LinuxCNC
V2.9.0~pre0+git20221023.7a5beabae0, 24 okt. 2022

Inhaltsverzeichnis

Teil I

Erste Schritte & Konfiguration

Kapitel 1

Erste Schritte mit LinuxCNC

1.1 Über LinuxCNC

1.1.1 Die Software

- LinuxCNC (Enhanced Machine Control) ist ein Softwaresystem zur Computersteuerung von Werkzeugmaschinen wie Fräs- und Drehmaschinen, Robotern wie Puma und Scara und anderen computergesteuerten Maschinen mit bis zu 9 Achsen.
- LinuxCNC ist freie Software mit offenem Quellcode. Aktuelle Versionen von LinuxCNC sind vollständig unter der GNU General Public License und Lesser GNU General Public License (GPL und LGPL) lizenziert
- LinuxCNC bietet:
  - einfaches Entdecken und Testen ohne Installation mit der LiveCD
  - einfache Installation von der Live-CD
  - benutzerfreundliche grafische Konfigurationsassistenten zum schnellen Erstellen einer maschinenspezifischen Konfiguration
  - direkt verfügbar als reguläre Pakete in den letzten Veröffentlichungen von Debian (seit Bookworm) und Ubuntu (seit Kinetic Kudu)
  - eine grafische Benutzeroberfläche (GUI) (es stehen sogar mehrere GUIs zur Auswahl)
  - ein Tool zur Erstellung einer grafischen Benutzeroberfläche (Glade)
  - ein Interpreter für *G-Code* (die Programmiersprache für RS-274-Werkzeugmaschinen)
  - ein System zur Bewegungsplanung in Echtzeit mit Vorausschau
  - 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
- Es bietet keine Zeichnungsfunktionen (CAD - Computer Aided Design) oder G-Code-Generierung aus der Zeichnung (CAM - Computer Automated Manufacturing).
- Er kann bis zu 9 Achsen gleichzeitig bewegen und unterstützt eine Vielzahl von Schnittstellen.
- Die Steuerung kann echte Servos (analog oder PWM) mit der Feedback-Schleife durch die LinuxCNC-Software auf dem Computer, oder Open-Loop mit Schritt-Servos oder Schrittmotoren betreiben.

- 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.
- Unterstützung für nicht-kartesische Bewegungssysteme wird über benutzerdefinierte Kinematikmodule bereitgestellt. Zu den verfügbaren Architekturen gehören Hexapoden (Stewart-Plattformen und ähnliche Konzepte) und Systeme mit Drehgelenken für die Bewegung wie PUMA- oder SCARA-Roboter.
- LinuxCNC läuft auf Linux mit Echtzeit-Erweiterungen.

1.1.2 Das Betriebssystem

LinuxCNC ist als gebrauchsfertige Pakete für die Ubuntu- und Debian-Distributionen verfügbar.

1.1.3 Hilfe erhalten

1.1.3.1 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, in den IRC zu gelangen, ist die Verwendung des eingebetteten Clients auf dieser Seite.

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 lösen kann.
- Haben Sie etwas Geduld, wenn Sie auf eine Antwort warten, denn 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*: http://pastebin.com/, http://pastie.org/, https://gist.github.com/
- *Für Bilder*: http://imagebin.org/, http://imgur.com/, http://bayimg.com/
- *Für Dateien*: https://filedropper.com/, http://filefactory.com/, http://1fichier.com/

1.1.3.2 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

1.1.3.3 Web-Forum

Ein Webforum finden Sie unter https://forum.linuxcnc.org oder über den Link oben auf der Homepage von linuxcnc.org.

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.

1.1.3.4 LinuxCNC-Wiki

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

Die von den Benutzern gepflegte LinuxCNC Wiki Seite enthält eine Fülle von Informationen und Tipps: http://wiki.linuxcnc.org

1.1.3.5 Fehlerberichte

Melden Sie Fehler an den LinuxCNC Link:github bug tracker.

1.2 Systemvoraussetzungen

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.

- 700 MHz x86-Prozessor (1,2 GHz x86-Prozessor empfohlen) oder Raspberry Pi 4 oder besser.
- Um LinuxCNC 2.8 und Debian Buster von der LiveCD auszuführen, sollte das System 64-Bit-fähig sein.
- 512 MB oder mehr RAM
- 8 GB Festplatte

- 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.
- Eine Netzwerk- oder 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 LiveCD informieren. Bei älterer Hardware kann es von Vorteil sein, eine ältere Version der LiveCD zu wählen, wenn diese verfügbar ist.

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.

1.2.2.1 Preempt-RT mit dem Paket *linuxcnc-uspace*

Preempt-RT is the newest of the realtime systems, and is also the version that is closest to a mainline kernel. Preempt-RT kernels are available as precompiled packages from the main repositories. The search term ”PREEMPT_RT” will find them, and one can be downloaded and installed just like any other package. Preempt-RT will generally have the best driver support and is the only option for systems using the Mesa ethernet-connected hardware driver cards. In general preempt-rt has the worst latency of the available systems, but there are exceptions.

1.2.2.2 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.

1.2.2.3 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.

1.2.2.4 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

1.2.3.1 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.

1.2.3.2 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 LinuxCNC erhalten

Dieser Abschnitt beschreibt den empfohlenen Weg zum Herunterladen und zur Neuinstallation von LinuxCNC. Es gibt auch Alternate Install Methods für die Abenteuerlustigen. Wenn Sie eine bestehende Installation haben, die Sie aktualisieren möchten, gehen Sie stattdessen zum Abschnitt Updating LinuxCNC.

Anmerkung
LinuxCNC benötigt 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 Tabelle unten für Details.

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 Prozesses 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.

1.3.1 Das Festplattenabbild (engl. kurz image) herunterladen

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

1.3.1.1 Normales Herunterladen

Für x86-PCs laden Sie die Live/Installations-CD herunter, indem Sie hier klicken:

http://www.linuxcnc.org/iso/linuxcnc-2.8.2-buster.iso

Für den Raspberry Pi ist ein vollständiges SD-Karten-Image hier verfügbar:

http://www.linuxcnc.org/iso/linuxcnc-2.8.1-pi4.zip (dies wird automatisch auf 2.8.2 aktualisiert)

Diese kann mit dem normalen Pi Installationsprozess installiert werden, auch mit der Raspberry Pi Imager App.

Es wird berichtet, dass dieses SD-Image nicht mit dem Raspberry Pi4 8GB Modell funktioniert. Beachten Sie auch, dass diese Version des SD-Images den verfügbaren Speicher auf 3 GB begrenzt, da dies notwendig ist, um es zu überzeugen, dass sowohl WiFi als auch USB auf einigen Versionen des Pi funktionieren. Sie können versuchen, diese Begrenzung aufzuheben, indem Sie die Datei config-rt.txt im Boot-Verzeichnis bearbeiten. Wenn Sie nach der Änderung nicht mehr booten können, dann können Sie die Datei wieder bearbeiten, indem Sie die SD-Karte in einen anderen Computer einlegen (vielleicht sogar einen Pi mit einem USB-Kartenleser).

1.3.1.2 Herunterladen mit zsync

zsync ist eine Download-Anwendung, die unterbrochene Downloads effizient wieder aufnimmt und große Dateien mit kleinen Änderungen effizient überträgt (wenn Sie eine ältere lokale Kopie haben). Verwenden Sie zsync, wenn Sie Probleme beim Herunterladen des Bildes mit der Methode Normales Herunterladen haben.

zsync unter Linux

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

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

- Führen Sie dann diesen Befehl aus, um die Iso auf Ihren Computer herunterzuladen

```
zsync http://www.linuxcnc.org/iso/linuxcnc-2.8.2-buster.iso
```

  oder

```
zsync http://www.linuxcnc.org/iso/linuxcnc-2.8.1-pi4.zip.zsync
```

zsync unter Windows Es gibt eine Windows-Portierung von zsync. Sie funktioniert als Konsolenanwendung. Sie kann heruntergeladen werden von:

https://www.assembla.com/spaces/zsync-windows/documents

1.3.1.3 Überprüfen des Abbilds

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

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

   ```
   md5sum linuxcnc-2.8.2-buster.iso
   ```

   oder

```
sha256sum linuxcnc-2.8.2-buster.iso
```

2. Vergleichen Sie dann mit diesen Prüfsummen

```
md5sum: 8a6e6abd2c792c3e06fbee0ed049ed41
sha256sum: 0bfeac3ddfe1bdbf5ca4dad84eeec165741d3f253a16b75e4405c06b7b489700
```

Überprüfen Sie md5sum auf Windows oder Mac Windows und Mac OS X werden nicht mit einem md5sum-Programm ausgeliefert, aber es gibt Alternativen. Weitere Informationen finden Sie unter: How To MD5SUM

1.3.2 Schreiben des Abbilds auf ein bootfähiges Gerät

Das Raspbery Pi-Image ist ein komplettes SD-Karten-Image und sollte auf eine SD-Karte in der normale Weg geschrieben werden.

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

Schreiben des Images auf ein USB-Speichergerät unter Linux

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 `dmesg`, nachdem Sie das Gerät angeschlossen haben. `/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-2.8.2-buster.iso of=/dev/sde
```

Schreiben des Abbilds auf ein USB-Speichergerät unter Mac OSX

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=/path-to.iso of=/dev/rdiskN bs=1m
```

5. Bitte beachten Sie, dass dieser Vorgang sehr lange dauern kann und dass Sie während des Vorgangs keine Rückmeldung erhalten.

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. Laden Sie Infra Recorder, ein kostenloses und quelloffenes Bildbrennprogramm, herunter und installieren Sie es: http://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 Menü ”Aktionen” und dann ”Bild brennen”.

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)

1.3.3 Testen von LinuxCNC

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.

Anmerkung
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 der 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.

1.3.4 LinuxCNC installieren

Um LinuxCNC von der LiveCD zu installieren, wählen Sie beim Booten ”Install (Graphical)”.

1.3.5 Updates für 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.

Warnung
Aktualisieren Sie das Betriebssystem nicht, wenn Sie dazu aufgefordert werden. Sie sollten jedoch Betriebssystem *Aktualisierungen* akzeptieren, insbesondere Sicherheitsaktualisierungen.

1.3.6 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.

1.3.7 Alternative Installationsmethoden

Der einfachste und bevorzugte Weg, LinuxCNC zu installieren, ist die Verwendung des Live/Installations-Images wie oben beschrieben. Diese Methode ist so einfach und zuverlässig, wie wir es machen können, und ist für Anfänger und erfahrene Benutzer gleichermaßen geeignet. Allerdings wird dies in der Regel ersetzen alle bestehenden Betriebssystem.

Zusätzlich werden für erfahrene Benutzer, die mit der Debian-Systemadministration vertraut sind (Installations-Images finden, apt-Quellen manipulieren, Kernel-Flavors ändern usw.), Neuinstallationen 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 Buster	amd64 & i386	LinuxCNC direkt nach der Installation	linuxcnc-uspace	nur Simulation
Debian Buster	amd64 & armhf	preemp-rt	linuxcnc-uspace	Maschinensteuerung und -simulation
Debian Buster	amd64	RTAI	linuxcnc	Maschinensteuerung (bekannte Probleme)
Debian Jessie	amd64 & i386	LinuxCNC direkt nach der Installation	linuxcnc-uspace	nur Simulation
Debian Wheezy	i386	RTAI	linuxcnc	Maschinensteuerung und -simulation
Debian Wheezy	amd64 & i386	Preempt-RT	linuxcnc-uspace	Maschinensteuerung und -simulation
Debian Wheezy	amd64 & i386	LinuxCNC direkt nach der Installation	linuxcnc-uspace	nur Simulation

Distribution	Architektur	Kernel	Paket-Name	Typische Verwendung
Ubuntu Precise	i386	RTAI	linuxcnc	Maschinensteuerung und -simulation
Ubuntu Precise	amd64 & i386	LinuxCNC direkt nach der Installation	linuxcnc-uspace	nur Simulation

Anmerkung
LinuxCNC v2.8 wird auf Ubuntu Lucid oder älter nicht unterstützt.

Preempt-RT-Kernel Die Preempt-rt-Kernel sind für Debian aus dem regulären debian.org-Archiv verfügbar. Der preempt-rt Kernel für RaspBerry Pi ist im LinuxCNC Repository 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 Buster: `deb http://linuxcnc.org buster base`
- Debian Wheezy: `deb http://linuxcnc.org wheezy base`
- Ubuntu Precise: `deb http://linuxcnc.org precise base`

Anmerkung
Debian Wheezy und Ubuntu Precise sind beide sehr alt und haben ihren Supportzeitraum hinter sich. Es wird dringend davon abgeraten, beide für eine Neuinstallation zu verwenden und ein Upgrade einer bestehenden Installation ernsthaft in Betracht zu ziehen.

Das Buster/RTAI-Paket ist nur für amd64 verfügbar, aber es gibt nur sehr wenige überlebende Systeme, die nicht mit einem 64-Bit-Betriebssystem arbeiten können.

Warnung
Es gibt bekannte Probleme mit dem 64-Bit RTAI 5.2 Kernel mit dieser Version von LinuxCNC. Das System wird gelegentlich einfrieren. Allerdings hat dies bisher nur beim Beenden des Systems gesehen worden. Während des Betriebs scheint das System stabil zu sein. Dennoch sollte es an dieser Stelle als experimentell betrachtet werden.

Anmerkung
Wenn Sie sich entscheiden, den RTAI 5.2-Kernel zu verwenden und ein Problem außerhalb der oben beschriebenen Umstände feststellen, melden Sie es bitte umgehend den Projektentwicklern.

1.3.7.1 Installation auf Debian Buster (mit Preempt-RT-Kernel)

1. Installieren Sie Debian Buster (Debian 10), Version amd64. Sie können das Installationsprogramm hier herunterladen: https://www.debian.org/releases/buster/
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.

3. Wählen Sie dann „Installieren“ oder „Grafische Installation“.

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

---

1. 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
```

2. Install the Preempt-RT kernel and modules

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

3. Starten Sie neu und wählen Sie den Linux-Kernel 4.19.0-9-rt-amd64. Dies könnte im Untermenü ”Erweiterte Optionen für Debian Buster” in Grub versteckt sein. Wenn Sie sich anmelden, überprüfen Sie, ob `PREEMPT RT` durch den folgenden Befehl gemeldet wird.

```
uname -v
```

4. Öffnen Sie Menü Anwendungen > System > Synaptic Package Manager, suchen Sie nach *linux-image* und klicken Sie 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.

5. Fügen Sie den LinuxCNC Archive Signing Key zu Ihrem apt keyring hinzu, indem Sie

```
sudo apt-key adv --keyserver hkp://keys.openpgp.org --recv-key 3cb9fd148f374fef
Alternate keyserver: keyserver.ubuntu.com
```

6. Fügen Sie das apt-Repository hinzu:

```
echo deb http://linuxcnc.org/ buster base 2.8-rtpreempt | sudo tee -a /etc/apt/sources. ↩
    list.d/linuxcnc.list
echo deb-src http://linuxcnc.org/ buster base 2.8-rtpreempt | sudo tee -a /etc/apt/ ↩
    sources.list.d/linuxcnc.list
```

7. Aktualisieren Sie die Paketliste von linuxcnc.org

```
sudo apt-get update
```

8. Installieren Sie uspace (vor der Installation von uspace kann ein Neustart erforderlich sein)

```
sudo apt-get install linuxcnc-uspace
```

9. Optional können Sie Mesaflash installieren, wenn Sie eine Mesa-Karte verwenden.

```
sudo apt install mesaflash
```

1.3.7.2 Installation unter Debian Buster (mit experimentellem RTAI-Kernel)

Warnung
Dieser Kernel hat bekannte Stabilitätsprobleme. Er scheint zuverlässig zu laufen, sobald LinuxCNC geladen ist. Allerdings wurden Kernel-Panics beim Herunterfahren des Systems gesehen.

1. Dieser Kernel und die LinuxCNC-Version können auf der LiveDVD-Installation installiert werden, oder alternativ auf einer neuen Installation von Debian Buster 64-bit, wie oben beschrieben
2. Fügen Sie den LinuxCNC Archive Signing Key zu Ihrem apt Keyring hinzu (nicht notwendig, wenn Sie den Echtzeitmodus eines LinuxCNC Live-CD-Images umschalten)

```
# Alternativer Schlüsselserver: keyserver.ubuntu.com
sudo apt-key adv --keyserver hkp://keys.openpgp.org --recv-key 3cb9fd148f374fef
```

3. Fügen Sie das apt-Repository hinzu:

```
echo deb http://linuxcnc.org/ buster base 2.8-rt | sudo tee /etc/apt/sources.list.d/ ↩
    linuxcnc.list
echo deb-src http://linuxcnc.org/ buster base 2.8-rt | sudo tee -a /etc/apt/sources. ↩
    list.d/linuxcnc.list
```

4. Aktualisieren Sie die Paketliste von linuxcnc.org

```
sudo apt-get update
```

5. Installieren Sie den neuen Echtzeit-Kernel, RTAI und die rtai-Version von linuxcnc

```
sudo apt-get install linux-image-4.19.195-rtai-amd64
```

6. Installation der RTAI-Anwendungsschicht

```
sudo apt-get install rtai-modules-4.19.195
```

7. Installieren Sie LinuxCNC (eventuell ist ein Neustart vor der Installation erforderlich)

```
sudo apt-get install linuxcnc
```

Starten Sie den Rechner neu und stellen Sie sicher, dass das System mit dem neuen Kernel 4.19.195-rtai bootet.

1.3.7.3 Installieren auf Raspbian 10

1. Laden Sie ein Standard-Raspbian-Image auf eine SD-Karte herunter und installieren Sie es auf die übliche Weise
2. Starten Sie den Pi und öffnen Sie ein Terminal
3. Fügen Sie den LinuxCNC Archive Signing Key zu Ihrem apt Schlüsselbund hinzu

```
# Alternativer Schlüsselserver: keyserver.ubuntu.com
sudo apt-key adv --keyserver hkp://keys.openpgp.org --recv-key 3cb9fd148f374fef
```

4. Fügen Sie das apt-Repository hinzu:

```
echo deb http://linuxcnc.org/ buster base 2.8-rtpreempt | sudo tee /etc/apt/sources. ↩
    list.d/linuxcnc.list
```

5. Aktualisieren Sie die Paketliste von linuxcnc.org

```
sudo apt-get update
```

6. den Echtzeit-Kernel installieren

```
sudo apt-get install linux-image-4.19.71-rt24-v7l+
```

7. Installieren Sie linuxcnc (vor der Installation kann ein Neustart erforderlich sein)

```
sudo apt-get install linuxcnc-uspace
```

1.3.7.4 Installieren unter Ubuntu Precise

1. Installieren Sie Ubuntu Precise 12.04 x86 (32-bit). Jede Variante sollte funktionieren (normales Ubuntu, Xubuntu, Lubuntu, etc.). 64-Bit (AMD64) wird derzeit nicht unterstützt. Sie können das Installationsprogramm hier herunterladen: http://releases.ubuntu.com/precise/ Beachten Sie die Warnungen, dass diese Version nicht mehr unterstützt wird. Aber es ist ein Weg, um LinuxCNC mit einem gut getesteten RTAI-Kernel zu installieren.
2. Führen Sie die folgenden Schritte aus, um den Rechner mit den neuesten Paketen in Ubuntu Precise auf den neuesten Stand zu bringen.

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

3. Fügen Sie den LinuxCNC Archive Signing Key zu Ihrem apt keyring hinzu, indem Sie

```
# Alternativer Schlüsselserver: keyserver.ubuntu.com
sudo apt-key adv --keyserver hkp://keys.openpgp.org --recv-key 3cb9fd148f374fef
```

4. Hinzufügen einer neuen Apt-Quelle

```
sudo add-apt-repository ”deb http://linuxcnc.org/ precise base 2.8-rt”
```

5. Holen Sie die Paketliste von linuxcnc.org.

```
sudo apt-get update
```

6. Installieren Sie den RTAI-Kernel und die Module durch Ausführen von

```
sudo apt-get install linux-image-3.4-9-rtai-686-pae rtai-modules-3.4-9-rtai-686-pae
```

7. Wenn Sie in der Lage sein wollen, LinuxCNC aus dem Quellcode zu bauen, indem Sie das Git Repo benutzen, führen Sie auch aus

```
sudo apt-get install linux-headers-3.4-9-rtai-686-pae
```

8. Starten Sie neu und vergewissern Sie sich, dass Sie mit dem rtai-Kernel booten. Wenn Sie sich anmelden, stellen Sie sicher, dass der Kernel-Name 3.4-9-rtai-686-pae lautet.

```
uname -r
```

9. Ausführen

```
sudo apt-get install linuxcnc
```

1.4 Ausführen von LinuxCNC

1.4.1 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.

1.4.2 Konfigurationsstarter

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
- *gmoccapy* - Touchscreen-GUI
- *gscreen* - Touchscreen-GUI
- *low_graphics* - Tastatur-GUI
- pyvcp_demo’ - Virtuelle Kontrolltafeln Python
- *qtvcp_screens* - Mit Qt5 und Python erstellte Anleitungen
- *tklinuxcnc* - Tastatur- und Maus-Gui (wird nicht mehr gepflegt)
- *touchy* - 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
- pluto
- servotogo
- vigilant
- vitalsystems

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
- plasmac
- scortbot erIII
- sherline
- smithy
- tormach

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

Die *Apps-Elemente* sind normalerweise 1) Dienstprogramme, die kein Starten von linuxcnc erfordern, oder 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-test - Standard-Test zur Bestimmung der Latenz
  - latency-plot - Streifendiagramm
  - latency-histogram - Histogram
- parport - Anwendungen zum Testen von parport.
- pyvcp - Beispiele für pyvcp-Anwendungen.
- xhc-hb04 – Anwendungen zum Testen eines drahtlosen USB-MPG xhc-hb04

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

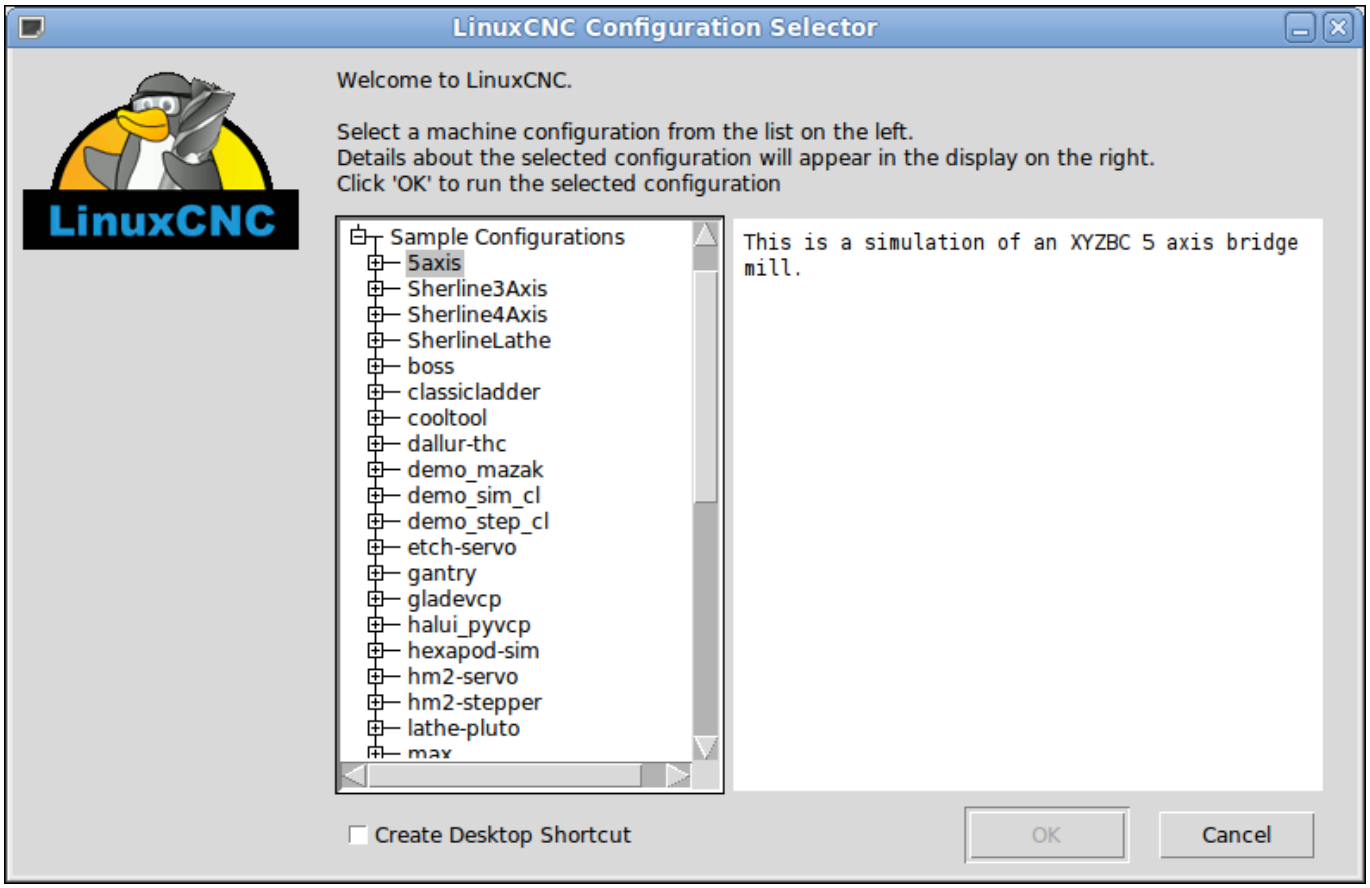


Abbildung 1.1: LinuxCNC-Konfigurationsauswahl

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.

1.4.3 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.

1.4.4 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.

1.4.5 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.

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
```

1.5 Aktualisieren von LinuxCNC

Die Aktualisierung von LinuxCNC auf eine neue Nebenversion (d.h. auf eine neue Version in der gleichen stabilen Serie, z.B. von 2.7.0 auf 2.7.1) 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.

1.5.1 Upgrade auf die neue Version

Dieser Abschnitt beschreibt, wie Sie LinuxCNC von Version 2.7 auf die neue Version 2.8.0 aktualisieren. Es wird davon ausgegangen, dass Sie eine bestehende 2.7 Installation haben, die Sie aktualisieren möchten.

Um LinuxCNC von einer Version älter als 2.7 zu aktualisieren, müssen Sie zuerst upgrade your old install to 2.7, 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 oder Debian Wheezy 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 bzw. 2018 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 2.7 auf 2.8 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 2.7 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:       wheezy
```

Sie sollten Debian Stretch oder Wheezy (wie oben) oder Ubuntu Precise verwenden. Es sind auch Pakete für Debian Jessie oder Debian Buster verfügbar, falls Sie bereits eines davon verwenden.

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

```
uname -r
4.19.0-9-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

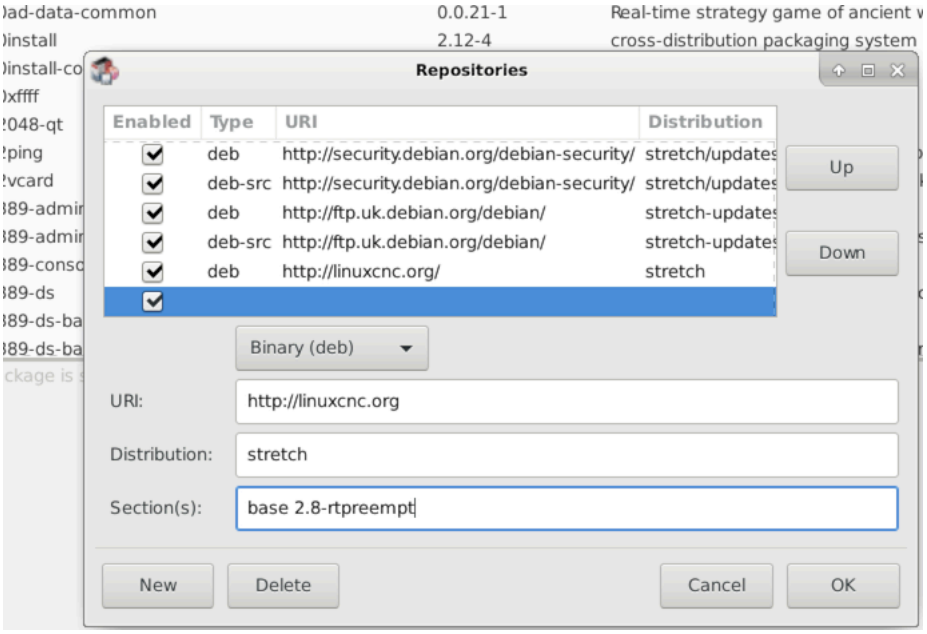
Wenn Sie `-rtai-` im Kernel-Namen sehen, dann laufen Sie mit RTAI-Echtzeit. Siehe unten für die LinuxCNC Version zu installieren.

1.5.1.1 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 Paketmanager*.
    - 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:

Plattform	**apt Quellzeile**
Debian Stretch	`deb http://linuxcnc.org stretch base 2.8-rtpreempt`
Debian Wheezy	`deb http://linuxcnc.org wheezy base 2.8-rt`
Ubuntu Precise	`deb http://linuxcnc.org precise base 2.8-rt`
Debian Jessie - preempt	`deb http://linuxcnc.org jessie base 2.8-rtpreempt`
Debian Jessie - RTAI	`deb http://linuxcnc.org jessie base 2.8-rt`
Debian Buster - preempt	`deb http://linuxcnc.org buster base 2.8-rtpreempt`
Debian-Knacker - RTAI	`deb http://linuxcnc.org buster base 2.8-rt`



- 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”.

1.5.1.2 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 Wheezy und Stretch verwenden beide 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.

1.5.1.3 Ubuntu Precise

- 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.

1.5.2 Aktualisieren ohne Netzwerk

Um ohne Netzwerkverbindung zu aktualisieren, müssen Sie die .deb-Datei herunterladen und mit dpkg installieren. Die .debs können unter http://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:       buster
```

Wählen Sie das Betriebssystem aus der Liste und dann die gewünschte Hauptversion wie 2.8-rt für RTAI oder 2.8-rtpreempt für preempt-rt.

Wählen Sie als Nächstes den Computertyp aus, den Sie haben: binary-amd64 für jeden 64-Bit-x86, binary-i386 für 32-Bit, binary-armhf (32-Bit) oder binary-arm64 (64-Bit) für Raspberry Pi.

Wählen Sie dann die gewünschte Version am Ende der Liste aus, z.B. *linuxcnc-uspace_2.8.0_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_2.8.0.deb*, dann öffnen Sie ein Terminal und installieren es mit dem Paketmanager mit diesem Befehl:

```
sudo dpkg -i linuxcnc_2.8.0.deb
```

1.5.3 Aktualisieren von Konfigurationsdateien (für 2.8.x)

Die neue Version von LinuxCNC unterscheidet sich von der Version 2.7 in einigen Punkten, die möglicherweise Änderungen an Ihrer Maschinenkonfiguration erfordern.

Der Hauptunterschied ist, dass LinuxCNC keine Annahmen mehr darüber macht, welches Gelenk welche Achse steuert. Diese Änderung wird ”joint-axes” genannt nach dem Namen des Entwicklungszweigs, in dem die Änderungen ihren Anfang nahmen. Diese Änderung hat in der Entwicklung seit mindestens 2010 gewesen, und wurde nun letztlich in der 2.8er Release integriert.

1.5.3.1 Verteilungskonfigurationen (Aktualisierungen für joints_axes)

Die LinuxCNC-Distribution enthält viele Beispielkonfigurationen, die in Verzeichnishierarchien organisiert sind: by_machine, by_interface, und sim (simulierte Maschinen). Diese Konfigurationen werden oft als Ausgangspunkt für die Erstellung einer neuen Konfiguration, als Beispiele für die Studie, oder als komplette simulierte Maschinen, die ohne spezielle Hardware oder Echtzeit-Kernel ausgeführt werden können.

Die Konfigurationsdateien in diesen Verzeichnisbäumen wurden für die für die joints_axes Updates erforderlichen Änderungen aktualisiert.

1.5.3.2 Automatische Aktualisierungen (update_ini-Skript für joints_axes)

Da die joints_axes-Updates eine Reihe von Änderungen an den Benutzer-INI-Dateien und den zugehörigen HAL-Dateien erfordern, wird ein Skript namens update_ini bereitgestellt, das die Benutzerkonfigurationen automatisch konvertiert.

Dieses Skript wird aufgerufen, wenn ein Benutzer eine bestehende Konfiguration zum ersten Mal nach der Aktualisierung von LinuxCNC startet. Das Skript sucht in der Benutzer-INI-Datei nach einem [EMC]VERSION-Element. Wenn dieses Element 1) nicht existiert, oder 2) existiert und auf den historischen CVS-Wert ”$Revision$” gesetzt ist, oder ein numerischer Wert kleiner als 1.1 ist, dann wird das update_ini Skript einen Dialog öffnen, um anzubieten, die Benutzerdateien zu bearbeiten, um eine aktualisierte Konfiguration zu erstellen. Wenn der Benutzer zustimmt, wird die Konfiguration aktualisiert.

Wenn die Benutzerkonfiguration beispielsweise bigmill.ini heißt, werden die Datei bigmill.ini und die mit ihr verbundenen lokalen HAL-Dateien bearbeitet, um die Änderungen an joints_axes zu übernehmen. Alle Dateien der ursprünglichen Konfiguration werden in einem neuen Verzeichnis gespeichert, das nach der ursprünglichen Konfiguration mit dem Suffix ”.old” benannt ist (im Beispiel bigmill.old).

Das Skript update_ini behandelt alle gängigen Benutzerelemente, die in einfachen Maschinen mit Identitätskinematik zu finden sind. Weniger gängige Elemente aus komplexeren Maschinen werden möglicherweise nicht automatisch konvertiert. Beispiele für komplexe Maschinenkonfigurationen sind:

- Portale mit zwei Gelenken für eine Achse
- Maschinen mit Handsteuerung (engl. jogwheels)
- Roboter mit nicht-identischer Kinematik
- Konfigurationen mit haltcl-Dateien

In den folgenden Unterabschnitten und im Abschnitt ”HAL-Änderungen” werden die Punkte aufgeführt, die zusätzliche Benutzeränderungen an INI- oder HAL-Dateien erfordern können.

1.5.3.3 Unterstützung mehrerer Spindeln

LinuxCNC unterstützt nun bis zu 8 Spindeln (und kann für mehr neu kompiliert werden). Bestehender G-Code wird ohne Änderung laufen und die meisten Konfigurationen werden standardmäßig auf einzelne Spindeln. Um mehr als eine Spindel zu spezifizieren, setzen Sie den [TRAJ]SPINDLES= Eintrag in der INI-Datei **und** fügen Sie den num_spindles= Parameter für das Bewegungsmodul ein (gesetzt entweder mit [EMCMOT]EMCMOT = motmod num_spindles= oder in einem HAL-Datei loadrt Eintrag für motmod).

Der Parameter num_spindles= des Bewegungsmoduls und die Einstellungen für [TRAJ]SPINDLES= müssen übereinstimmen.

Die Namen der Spindelsteuerungspins wurden geändert, damit die Spindeln mehr wie Achsen und Gelenke aussehen. motion.spindle-speed-out heißt zum Beispiel jetzt spindle.0.speed-out. Das automatische Update-Skript wird diese Änderungen übernehmen. Um zusätzliche Spindeln zu steuern, akzeptieren die G- und M-Codes, welche die Spindeldrehzahl steuern, jetzt ein zusätzliches ”$”-Argument, zum Beispiel M3 $2, um die dritte Spindel zu starten. Es sollte möglich sein, benutzerdefinierte G-Codes zu erstellen, die zu jeder anderen Mehrspindelsteuerung passen. Siehe die G-Code- und M-Code-Handbücher für Code-Änderungen und man motion für die HAL-Pin-Änderungen.

1.5.3.4 TRAJ Geschwindigkeiten, Beschleunigungen Namen

Mit der Aufnahme der Funktionalität joints_axes wurden einige Namen geändert, um die verfügbare Funktionalität zu verdeutlichen.

```
war: [TRAJ]MAX_VELOCITY ist: [TRAJ]MAX_LINEAR_VELOCITY
war: [TRAJ]DEFAULT_VELOCITY ist: [TRAJ]DEFAULT_LINEAR_VELOCITY

war: [TRAJ]MAX_ACCELERATION ist: [TRAJ]MAX_LINEAR_ACCELERATION
war: [TRAJ]DEFAULT_ACCELERATION ist: [TRAJ]DEFAULT_LINEAR_ACCELERATION
```

1.5.3.5 Kinematik-Module

Die Kinematikmodule gentrivkins und gantrykins wurden entfernt, da ihre Funktionalität nun im aktualisierten Modul trivkins verfügbar ist.

Das Modul gentrivkins war nur in früheren joints_axes-Zweigen verfügbar. Für die Konvertierung ist es notwendig, den Namen zu ändern.

Beispiele für HAL-Dateien:

```
war: loadrt gentrivkins
 ist: loadrt trivkins

war: loadrt gentrivkins koordinaten=xyyz
 ist: loadrt trivkins koordinaten=xyyz
```

Konfigurationen, die gantrykins verwenden, sollten aktualisiert werden, um trivkins zu verwenden, wobei der Parameter kinstype= auf BOTH (für KINEMATICS_BOTH) gesetzt wird.

Beispiel einer HAL-Datei:

```
war: loadrt gantrykins Koordinaten=xyyz
 ist: loadrt trivkins Koordinaten=xyyz kinstype=BOTH
```

Weitere Informationen finden Sie in der Manpage von trivkins (*$ man trivkins*)

Note: Die am meisten unterstützte Verwendung für die Angabe von Kinematik in joints_axes ist die Festlegung von Werten im Abschnitt [KINS] der Konfigurations-INI-Datei und die anschließende Bezugnahme auf diese Werte innerhalb der angegebenen [HAL]HALFILES (.hal .tcl-Dateien). Zum Beispiel:

```
INI-Datei:    [KINS]
            KINEMATICS = trivkins
            JOINTS = 3
            ...

HAL-Datei:    loadrt [KINS]KINEMATICS

haltcl Datei: loadrt $::KINS(KINEMATICS)
```

1.5.3.6 Drehmaschinen-Konfigurationen

Vor der Einführung von joints_axes wurden Drehmaschinen oft so konfiguriert, als wären sie dreiachsige (XYZ) Maschinen mit einer unbenutzten Achse (Y). Dies war praktisch für die gemeinsame Nutzung von HAL-Dateien (insbesondere für Simulationskonfigurationen), erforderte jedoch die Angabe von [TRAJ]AXES =3, einen ”Dummy”-Abschnitt AXIS_Y und Vorkehrungen für die Referenzierung der unbenutzten Y-Koordinate. Diese Vorkehrungen sind nicht mehr erforderlich oder empfehlenswert.

Historische Drehmaschinenkonfigurationen verwendeten die Standardoptionen für das trivkins Kinematikmodul. Diese Standardoptionen konfigurieren alle Achsenbuchstaben (XYZABCUVW). Mit der Einbindung von joints_axes setzt eine geeignetere Kinematik-Spezifikation die Koordinaten auf genau die verwendeten (XZ) und setzt die Anzahl der Gelenke entsprechend auf 2. Es besteht keine Notwendigkeit für einen [AXIS_Y]-Abschnitt in der INI-Datei und es müssen nur zwei [JOINT_N]-Abschnitte definiert werden.

Beispiel für INI-Dateien für eine Drehmaschine (es werden nur die für die Kinematik relevanten Abschnitte gezeigt):

```
[KINS]
KINEMATICS = trivkins coordinates=xz
JOINTS = 2

[TRAJ]
COORDINATES = XZ
...

[AXIS_X]
...

[AXIS_Z]
...

[JOINT_0]
...

[JOINT_1]
...
```

Beachten Sie, dass einige Simulationskonfigurationen möglicherweise noch immer die historischen Drehbankkonfigurationen verwenden.

1.5.3.7 Konsistente Gelenke/Achsen-Spezifikationen

INI-Datei-Elemente für die Verwendung von Gelenken und Achsen müssen konsistent sein.

Das Bewegungskinematikmodul, das normalerweise mit *[KINS]KINEMATICS*= geladen wird, muss eine Anzahl von Gelenken verwenden, die der mit *[KINS]JOINTS*= angegebenen Anzahl entspricht.

Das Kinematikmodul muss Achsenbuchstaben implementieren, die mit der vom Aufgabenmodul verwendeten Spezifikation *[TRAJ]COORDINATES*= übereinstimmen.

Beispiele:

Dreiachsige kartesische Maschine mit trivkins (KINEMATICS_IDENTITY):

```
[KINS]KINEMATICS  = trivkins
[KINS]JOINTS      = 3
[TRAJ]COORDINATES = XYZ
```

Zweiachsige Drehmaschine mit trivkins (KINEMATICS_IDENTITY) mit nicht aufeinanderfolgenden Achsenbuchstaben:

```
[KINS]KINEMATICS  = trivkins coordinates=XZ
[KINS]JOINTS      = 2
[TRAJ]COORDINATES = XZ
```

Gantry unter Verwendung von trivkins mit doppelten Achsenbuchstaben und KINEMATICS_BOTH, um eine individuelle Gelenkpositionierung (für die Referenzfahrt) zu ermöglichen:

```
[KINS]KINEMATICS  = trivkins coordinates=XYYZ kinstype=BOTH
[KINS]JOINTS      = 4
[TRAJ]COORDINATES = XYYZ
```

Gantry mit Trivkins (KINEMATICS_BOTH) mit doppelten Achsenbuchstaben und einer Drehachse mit übersprungenen Achsenbuchstaben (A,B übersprungen):

```
[KINS]KINEMATICS  = trivkins coordinates=XYYZC kinstype=BOTH
[KINS]JOINTS      = 5
[TRAJ]COORDINATES = XYYZC
```

Linearer Delta-Roboter mit nicht-identischen Kins (KINEMATICS_BOTH), der im kartesischen Rahmen mit einer zusätzlichen Drehkoordinate arbeitet:

```
[KINS]KINEMATICS  = lineardeltakins
[KINS]JOINTS      = 4
[TRAJ]COORDINATES = XYZA
```

Note: Einige universelle Kinematikmodule (wie trivkins) implementieren Identitätskinematik mit Unterstützung für Koordinatenangaben (Achsenbuchstaben). Achsenbuchstaben können weggelassen werden. Achsenbuchstaben können dupliziert werden. Gelenke werden den Achsenbuchstaben auf definierte Weise zugewiesen (”$ man trivkins”).

Note: Für das Laden von trivkins-Modulen dürfen Sie keine Leerzeichen um das =-Zeichen oder Buchstaben einfügen:

```
This:     [KINS]KINEMATICS = trivkins coordinates=XZ
NOT This: [KINS]KINEMATICS = trivkins coordinates = X Z
```

Note: Benutzerdefinierte Kinematikmodule, die Nicht-Identitätskinematiken implementieren (wie lineardeltakins), definieren maschinenspezifische Beziehungen zwischen einem Satz von Koordinaten und einem Satz von Gelenken. Normalerweise berechnen benutzerdefinierte Kinematikmodule die Beziehungen zwischen Gelenken und Achsen innerhalb des benutzerdefinierten Moduls, aber es ist wichtig, konsistente Einstellungen für die zugehörigen INI-Elemente zu verwenden: *[KINS]JOINTS* und *[TRAJ]COORDINATES*. Die Details werden in der Regel in der Manpage des Moduls erklärt (z.B. *$ man lineardeltakins*).

1.5.3.8 Referenzierfahrt-Sequenzen

Negative Werte können für die INI-Datei-Elemente verwendet werden mit der Bezeichnung [JOINT_n]HOMI
Vor der Aufnahme von joints_axes bedeutete ein Wert von -1 oder das Weglassen des Elements an, dass keine Sequenz anwendbar war. Jetzt wird nur noch das Weglassen des Eintrags für diesen Zweck verwendet. Siehe das Kapitel: Referenzierfahrt Konfiguration für weitere Informationen.

1.5.3.9 Sperrender (engl. locking) rotierender Indexers (Updates für joints_axes)

Bei joints_axes ist ein Indexer ein Gelenk, das referenziert werden kann (Gelenkmodus), aber auch aus dem G-Code heraus entriegelt werden muss. Dies erfordert eine Eins-zu-eins-Entsprechung zwischen einem einzelnen Gelenk und einer Achse.

Geben Sie die Gelenknummer an, die einer Drehachse (L = A, B oder C) mit einer INI-Datei-Einstellung für die Achse entspricht:

```
[AXIS_L]LOCKING_INDEXER_JOINT = Gelenknummer_für_Indexer
```

Geben Sie an, dass es sich bei der Verbindung um einen sperrenden (engl. locking) Indexer handelt, indem Sie in der INI-Datei eine Einstellung für die Verbindung vornehmen (*N* steht für joint_number_for_indexe

```
[JOINT_N]LOCKING_INDEXER = 1
```

Es können HAL Pins erstellt werden, um die Verwendung eines sperrenden (engl. locking) Indikator-Gelenks zu koordinieren:

```
joint.N.unlock (BIT-Ausgang vom HAL)
joint.N.is-unlocked (BIT-Eingang zum HAL)
```

Um diese HAL-Pins für sperrende Gelenke zu erzeugen, müssen alle Gelenke, die als sperrende Indexer verwendet werden, mit dem Parameter *unlock_joints_mask* für das motmod-Modul angegeben werden. (bit0(LSB)==>Gelenk0, bit1==>Gelenk1, usw.)

```
[EMCMOT]
EMCMOT = motmod unlock_joints_mask=BITMASK
```

Nehmen wir als Beispiel eine Maschine mit einer Trivkins-Kinematik mit den Koordinaten XYZB, wobei B ein Verriegelungsindexer ist. Bei der Trivkins-Kinematik werden den angegebenen Koordinaten fortlaufend Gelenknummern (beginnend mit 0) zugewiesen (die Buchstaben der Achsenkoordinaten können weggelassen werden). In diesem Beispiel: X==>Gelenk0, Y==>Gelenk1, Z==>Gelenk2, B==>Gelenk3. Die Maske zur Angabe von Gelenk 3 ist 000001000 (binär) == 0x08 (hexadezimal)

Die erforderlichen INI-Datei-Einträge für dieses trivkins XYZB-Beispiel sind:

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

[TRAJ]
COORDINATES = XYZB
...

[EMCMOT]
EMCMOT = motmod unlock_joints_mask=0x08
...

[AXIS_B]
LOCKING_INDEXER_JOINT = 3
...

[JOINT_3]
LOCKING_INDEXER = 1
...
```

Für komplexere Kinematiken wählen Sie die Gelenknummer nach Bedarf aus - es muss eine Eins-zu-Eins-Entsprechung zwischen der Drehachse und der Gelenknummer bestehen.

(Weitere Informationen zu motmod finden Sie in der motion man-Seite (*$ man motion*))

1.5.3.10 Strengere INI-Datei Syntax

Zeilen mit numerischen INI-Variablen sind nicht mehr erlaubt, um Text am Ende zu haben. In früheren Versionen von LinuxCNC wurde jeglicher Text nach der Zahl stillschweigend ignoriert, aber ab

dieser Version ist ein solcher Text völlig unzulässig. Dies schließt Rautezeichen (”#”) ein, die in dieser Position ein Teil des Wertes sind, nicht ein Kommentarzeichen.

Beispielsweise werden Zeilen wie diese nicht mehr akzeptiert:

```
MAX_VELOCITY = 7.5 # Dies ist die maximale Geschwindigkeit der Achse.
```

Sie könnten wie folgt in zwei Zeilen ausgedrückt werden:

```
# Dies ist die maximale Geschwindigkeit der Achse.
MAX_VELOCITY = 7.5
```

1.5.3.11 [TRAJ] Einstellungen

In den Versionen 2.7.x sind die Einstellungen für die Trajektorien-Planung ([TRAJ]) enthalten:

```
[TRAJ]
DEFAULT_ACCELERATION
MAX_ACCELERATION
```

Zwischenzeitlich wurden Arbeiten für unterschiedliche lineare und winklige Elemente vorbereitet, indem diese Elemente umbenannt wurden:

```
[TRAJ]
DEFAULT_LINEAR_ACCEL
MAX_LINEAR_ACCEL
```

Da diese abgekürzten Namen nicht mit anderen Namenskonventionen und der Implementierung des update_ini-Skripts übereinstimmten, wurde die vorläufige Namensgebung korrigiert und verwendet:

```
[TRAJ]
DEFAULT_LINEAR_ACCELERATION
MAX_LINEAR_ACCELERATION
```

Anmerkung
Eine Unterstützung für die Angabe von Winkelvorgaben und Maximalbeschleunigungen bei der Flugbahnplanung wurde nicht implementiert.

1.5.4 HAL Änderungen (Aktualisierungen für joints_axes 2.8.x)

1.5.4.1 Joggen mit dem Handrad oder MPG (manueller Impulsgeber)

Vor der Einbindung der joints_axes-Updates wurde das Joggen der Räder nur im Gelenkmodus unterstützt und mit HAL-Pins gesteuert:

```
bit   IN  axis.M.jog-enable
float IN  axis.M.jog-scale
s32   IN  axis.M.jog-counts
bit   IN  axis.M.jog-vel-mode
```

wobei ”M” eine Zahl ist, die einem Achsenbuchstaben entspricht (0==>X, 1==>Y, usw.)

Durch die Einbindung von joints_axes-Updates ist das Joggen der Räder für Gelenke im Joint-Modus und für jede Achsenkoordinate im Teleop-Modus verfügbar. Die zur Verfügung stehenden HAL-Pins sind:

```
bit   IN  joint.N.jog-enable
float IN  joint.N.jog-scale
s32   IN  joint.N.jog-counts
bit   IN  joint.N.jog-vel-mode

bit   IN  axis.L.jog-enable
float IN  axis.L.jog-scale
s32   IN  axis.L.jog-counts
bit   IN  axis.L.jog-vel-mode
```

wobei N eine Gelenknummer und L ein Achsenbuchstabe ist.

Um ein Handrad in Identitäts-Kins-Konfigurationen zu verwenden, bei denen es eine Eins-zu-Eins-Entsprechung zwischen einer Gelenknummer und einem Achsenbuchstaben gibt, kann es sinnvoll sein, die entsprechenden HAL-Pins zu verbinden. Zum Beispiel, wenn Gelenk 1 genau dem Achsenbuchstaben y entspricht:

```
net jora_1_y_enable   => joint.1.jog-enable => axis.y.jog-enable
net jora_1_y_scale    => joint.1.jog-scale  => axis.y.jog-scale
net jora_1_y_counts   => joint.1.jog-counts => axis.y.jog-counts
net jora_1_y_vel-mode => joint.1.jog-counts => axis.y.jog-vel-mode
```

(Die Signalnamen jora_1_y_* sind Beispiele, die Namen vor der Konvertierung für joints_axes hängen von den spezifischen Konfigurationsdetails ab.)

Konfigurationen mit nicht identischer Kinematik und Konfigurationen, die doppelte Achsenbuchstaben verwenden (z. B. Portale, die mehr als ein Gelenk für eine Achsenkoordinate verwenden), erfordern eine geeignete unabhängige Steuerlogik, um sowohl die Gelenk- als auch die Teleop-(Welt-)Bewegung zu unterstützen.

1.5.4.2 INI HAL-Pins

HAL-Pins werden für INI-Dateielemente sowohl für Gelenke ([JOINT_N]) als auch für Achsen ([AXIS_L]) erstellt:

Für N = 0 ... [KINS](GELENKE-1):

```
INI file item              HAL pin name
[JOINT_N]BACKLASH          ini.N.backlash
[JOINT_N]FERROR            ini.N.ferror
[JOINT_N]MIN_FERROR        ini.N.min_ferror
[JOINT_N]MIN_LIMIT         ini.N.min_limit
[JOINT_N]MAX_LIMIT         ini.N.max_limit
[JOINT_N]MAX_VELOCITY      ini.N.max_velocity
[JOINT_N]MAX_ACCELERATION  ini.N.max_acceleration
[JOINT_N]HOME              ini.N.home
[JOINT_N]HOME_OFFSET       ini.N.home_offset
```

Für L = x y z a b c u v w:

```
INI file item              HAL pin name
[AXIS_L]MIN_LIMIT          ini.L.min_limit
[AXIS_L]MAX_LIMIT          ini.L.max_limit
[AXIS_L]MAX_VELOCITY       ini.L.max_velocity
[AXIS_L]MAX_ACCELERATION   ini.L.max_acceleration
```

Anmerkung
In früheren Versionen von LinuxCNC (vor joints_axes Updates), bezogen sich die HAL-Pin-Namen `ini._N_.*` sich Achsen mit 0==>x, 1==>y, usw. (Pins wurden für alle 9 Achsen erstellt). Weitere Informationen finden Sie in der Manpage von *milltask*.

1.5.5 HAL Änderungen (Aktualisierungen für joints_axes 2.8.x)

1.5.5.1 halcompile

Die Anzahl der names=-Instanzen war früher auf 16 begrenzt. Jetzt werden die Instanzen für Echtzeitkomponenten (loadrt) dynamisch zugewiesen, ohne dass eine Begrenzung eingebaut ist. Die Begrenzung auf 16 gilt weiterhin für names=-Elemente für Userspace-Komponenten (loadusr).

Für Komponenten, die ”personality” verwenden, kann die maximale Anzahl nun durch eine Kommandozeilenoption -P|--personalities eingestellt werden.

1.5.5.2 Parameter zu Pin Änderungen

Die folgenden HAL-Ausgangspins wurden von Parametern in Pins umgewandelt, so dass sie mit Signalen verbunden werden können:

```
motion.servo.last-period (letzte Periode des Servos in clks)
motion.servo.last-period_ns (Kernel-abhängige Verfügbarkeit)
```

1.5.6 Schnittstellenänderungen für joint_axes 2.8.x

1.5.6.1 Python LinuxCNC-Modul

Die jog()-Schnittstelle enthält ein *joint-flag* zur Angabe von joint (True) oder teleop (False) jogging:

```
jog(command, joint-flag, axis-or-joint-number, velocity[, distance])

jog(linuxcnc.JOG_STOP, joint-flag, axis-or-joint-number)
jog(linuxcnc.JOG_CONTINUOUS, joint-flag, joint-flag, velocity)
jog(linuxcnc.JOG_INCREMENT, joint-flag, axis-or-joint-number, velocity, distance)
```

1.5.7 GUIs (Aktualisierungen für joints_axes 2.8.x)

1.5.7.1 Hinweise zu Gelenk/Achsen-Jogging, Referenzfahrt und Kinematik

Mit der Aufnahme von joints_axes Updates, erzwingt LinuxCNC die Unterscheidungen von Gelenken und Achsen (Koordinaten Buchstaben) - aber einige GUIs (wie die AXIS GUI) verstecken diese Unterscheidungen für einfache Maschinen.

In den meisten Fällen kann man sich die Gelenke als *Motoren* vorstellen.

Die Beziehungen zwischen Gelenken und Achsenkoordinaten werden durch die mathematischen Kinematikfunktionen bestimmt, welche die Bewegung einer Maschine beschreiben.

Die Weltkoordinaten (X,Y,Z,A,B,C,U,V,W) werden durch die Anwendung von *FORWARD* Kinematikoperationen auf die Gelenkpositionen (Motor) bestimmt.

Bei Bewegungen im Weltraum (z. B. G-Code-Bewegungen) werden die erforderlichen Gelenk- (Motor-) Positionen durch Anwendung von *INVERSE* Kinematikoperationen auf die für die Bewegung im Weltraum angeforderten Koordinaten bestimmt.

Eine Bewegung im Weltraum ist nur *nach* der Referenzfahrt möglich.

Bei einfachen Maschinen (wie Fräsmaschinen und Drehmaschinen) gibt es eine Eins-zu-Eins-Entsprechung von Gelenken und Achsenkoordinatenbuchstaben. Bei einer XYZ-Fräse beispielsweise lauten die Beziehungen typischerweise: AchseX==Gelenk0, AchseY==Gelenk1, AchseZ=Gelenk2. Diese Korrespondenz wird als ”IDENTITY”-Kinematik bezeichnet und das verwendete Kinematikmodul ist in der Regel trivkins (trivial kinematics). (Siehe die Manpage von trivkins *$ man trivkins*)

Gelenk-Jogging (mit Gelenknummer 0,1,...) wird im Gelenkmodus verwendet (normalerweise nur *VOR* der Referenzfahrt). Wenn die Referenzfahrt abgeschlossen ist, wird der Jogging-Modus *AUTOMATISCH* vom Joint-Modus in den World-Modus umgeschaltet und das Achsen-Jogging (Koordinatenbuchstabe X,Y,...) wird verwendet. Dies ist für alle G-Code-Bewegungen geeignet, die durch MDI-Befehle oder G-Code-Programme angefordert werden.

Obwohl das Joggen im Gelenkmodus nach der Referenzfahrt oft nicht erforderlich ist, bieten einige GUIs (wie z. B. AXIS) eine Tastenkombination ("$"), mit der bei Maschinen, die eine 'nicht-Idenitäts"-Kinematik verwenden, zwischen Gelenk- und Weltmodus (Teleop) umgeschaltet werden kann.

In vielen häufigen Situationen ist gleichzeitiges Jogging nicht erforderlich, da die Referenzfahrt mit Referenzpunkt-Schalter durchgeführt wird und/oder die verschiedenen Referenzfahrt-Methoden von LinuxCNC zur Verfügung gestellt werden. Man schaltet einfach die Maschine ein, gibt den Befehl Referenzfahrt-Alle, die Maschine referenziert und wechselt automatisch in den Weltmodus. Siehe Referenzfahrt-Konfiguration.

Bei Maschinen, die keine Referenzfahrtschalter verwenden, kann es erforderlich sein, vor der Referenzfahrt jedes einzelnen Gelenk manuell in den Gelenkmodus zu schalten. Es ist auch möglich, die sofortige Referenzfahrt (siehe Referenzfahrt-Dokumente) für Gelenke zu verwenden, die keine Referenzfahrt zu einer festen Position erfordern.

Obwohl ein GUI die Unterscheidung zwischen Gelenken und Achsen für "IDENTITY"-Kinematikmaschinen ausblenden kann, ist es in der Regel wichtig, die Referenzfahrt abzuschließen, um Programme auszuführen oder von einer Benutzeroberfläche bereitgestellte Funktionen zu nutzen.

Standardmäßig deklariert sich das trivkins-Modul mit einer *IDENTITY*-Kinematik. Die Unterscheidung zwischen Gelenk- und Weltoperationen kann bei der Verwendung von trivkins in der grafischen Benutzeroberfläche von AXIS sichtbar gemacht werden, indem der Kinematik-Typ mit "kinstype=both" auf einen "Nicht-IDENTITY"-Typ gesetzt wird. Die Einstellung *both* zeigt an, dass sowohl Vorwärts- als auch Inverskinematikfunktionen verfügbar sind und dass solche GUI-Bestimmungen, die eine Unterscheidung von Gelenken und Achsenbuchstaben verbergen, nicht verwendet werden sollten. Geben Sie zum Beispiel für eine xyz-Konfiguration an:

```
[KINS]
KINEMATICS = trivkins coordinates=xyz kinstype=both
```

Bei dieser Einstellung wird Identitätskinematik verwendet, aber die AXIS-GUI wird:

1. Gelenknummern vor Referenzfahrt anzeigen
2. Achsenbuchstaben nach erfolgreicher Referenzfahrt anzeigen
3. Unterstützung des Umschaltens zwischen Gelenk- und Teleop-Modus mit der *$*-Taste

1.5.7.2 Halui

Halui unterstützt nun Teleop-Jogging, was zu einigen geänderten Pin-Namen und zahlreichen neuen Namen für Jogging-bezogene Pins führt.

In der Manpage (*$ man halui*) finden Sie alle Pin-Namen.

Neue Pins für Teleop-Jogging sind:

```
new: halui.axis.jog-speed
new: halui.axis.jog-deadband

neu: halui.axis.L.plus
neu: halui.axis.L.minus
      ... usw.
```

wobei *L* ein Buchstabe ist, der einem der durch [TRAJ]COORDINATES angegebenen Achsenbuchstaben entspricht, oder *selected* für die durch die halui.axis.*L*.select-Pins ausgewählte Achse.

Alle Pins für gemeinsames Joggen wurden aus Gründen der Spezifität umbenannt:

```
war: halui.jog-speed ist: halui.joint.jog-speed
war: halui.jog-deadband ist: halui.joint.jog-deadband

was: halui.jog.N.plus         ist: halui.joint.N.plus
was: halui.jog.N.minus        ist: halui.joint.N.minus
      ...  usw.                    ... usw.
```

wobei *N* eine Gelenknummer (0 … *num_noints*-1) oder *selected* für das durch die halui.joint.*N*.select Pins ausgewählte Gelenk ist.

Die HAL-Pins für ”ausgewählte” Gelenke wurden umbenannt, damit sie mit verwandten Pins konsistent sind.

```
war: halui.joint.selected.is_homed
ist: halui.joint.selected.is-homed

war: halui.joint.selected.on-soft-limit
ist: halui.joint.selected.on-soft-min-limit
```

1.5.7.3 AXIS GUI

Die AXIS GUI unterstützt weiterhin Identitätskinematik-Konfigurationen. Diese Steuerung verbirgt die Unterscheidung von Achsen und Gelenken, um die Anzeige und Verwendung von einfachen Maschinen zu vereinfachen.

Einige Maschinen, typischerweise Gantrys, können eine Konfiguration verwenden, bei der einem Achsenbuchstaben mehr als ein Gelenk zugeordnet ist. Dies kann mit dem trivkins Kinematikmodul unter Verwendung wiederholter Koordinatenbuchstaben erfolgen. Beispiel: Eine Maschine, die mit INI-Einstellungen wie folgt konfiguriert ist:

```
[KINS]
KINEMATICS = trivkins coordinates=XYYZ kinstype=BOTH
...
[TRAJ]
COORDINATES = XYYZ
...
```

Diese Maschine hat nach der Referenzfahrt eine Eins-zu-Eins-Entsprechung zwischen einem einzelnen Achsenbuchstaben (Y) und einem Gelenkpaar (1,2). Die Verwendung von *kinematics=BOTH* ermöglicht die Steuerung einzelner Gelenke im Gelenkmodus *falls erforderlich*.

Die AXIS GUI unterstützt Konfigurationen mit Nicht-Identitäts-Kinematiken:

1. Tastenbindung (*$*) zum Umschalten des Gelenk- oder Teleop-Modus
2. Vorschau Tab-Anzeige von Gelenken oder Achsen je nach Gelenk- oder Teleop-Modus
3. Vorschau Registerkarte mit Anzeige der Symbole ”Referenzfahrt” und ”Limit” im Gelenk-Modus
4. Vorschau Registerkarte Anzeige der Symbole *Alle Referenziert* (engl. all homed) und *Beliebiges Limit* (engl. any limit) im Teleop-Modus
5. DRO Tab-Anzeige des Gelenks oder der Achsen je nach Gelenk- oder Teleop-Modus
6. Jogging wird sowohl im Gelenk- als auch im Teleop-Bewegungsmodus unterstützt.
7. Externe Änderungen des Bewegungsmodus ”Joint/Teleop” werden erkannt.

Bei Identitätskinematiken werden ”Referenzpunkt” (engl. home)-Symbole für den entsprechenden (eins-zu-eins) Achsenbuchstaben angezeigt, wenn ein Gelenk referenziert ist.

Bei Nicht-Identitätskinematiken werden ”Referenzpunkt”-Symbole für einzelne Gelenke angezeigt, wenn ein Gelenk im Gelenkanzeigemodus referenziert ist. Das Symbol ”Alle referenziert” wird für alle Achsenbuchstaben angezeigt, wenn ALLE Gelenke im Weltanzeigemodus referenziert sind.

Bei Identitätskinematiken werden ”Limit”-Symbole für den entsprechenden (eineindeutigen) Achsenbuchstaben angezeigt, wenn ein Gelenklimit aktiv ist.

Bei Nicht-Identitätskinematiken werden ”Limit”-Symbole für einzelne Gelenke angezeigt, wenn das Gelenklimit im Gelenkanzeigemodus aktiv ist. Ein ”Any-Limit”-Symbol wird angezeigt, wenn sich ein beliebiges Gelenk im Teleop-Anzeigemodus an einem Limit befindet.

In der AXIS-Bedienoberfläche werden die Jogging-Tasten den Achsen auf eine konfigurierbare Weise zugewiesen. Für 3-Achsen-Maschinen, XYZA-Maschinen und Drehmaschinen ist die Voreinstellung die gleiche wie in 2.7. Bei anderen Maschinen werden die 4 Tipptastenpaare den ersten 4 Achsen zugewiesen, die in der Reihenfolge XYZ ABC UVW existieren. Diese Zuweisungen können durch neue INI-Datei-Direktiven im Abschnitt [DISPLAY] der INI-Datei gesteuert werden.

Beachten Sie, dass die für das Jogging verwendeten Parameter bei Maschinen mit nicht identischer Kinematik möglicherweise nicht für beide Modi geeignet sind.

1.5.7.4 TkLinuxCNC

Die TkLinuxCNC-GUI unterstützt sowohl Identitäts- als auch Nicht-Identitätskinematiken, enthält GUI-Radiobuttons und eine Tastenbindung (*$*) zum Umschalten zwischen Gelenk- und Teleop-Modus. Externe Änderungen des Gelenk- oder Teleop-Bewegungsmodus werden erkannt. Jogging wird sowohl im Gelenk- als auch im Teleop-Bewegungsmodus unterstützt. Beachten Sie, dass die für das Jogging verwendeten Parameter bei Maschinen mit nicht identischer Kinematik möglicherweise nicht für beide Modi geeignet sind.

OpenGL wird von TkLinuxCNC nicht verwendet, so dass es verwendet werden kann, um Probleme und Systemabhängigkeiten zu isolieren, die mit moderneren GUIs wie AXIS offengelegt werden.

Die rudimentäre Backplot-GUI ist für die Verwendung mit identischen kinematischen (xyz) Maschinenkonfigurationen verfügbar.

Der Code von emcsh.cc stellt die von TkLinuxCNC verwendeten Tcl-Befehle bereit. Die Befehle sind für Tcl-Anwendungen als Tcl-Paket mit dem Namen *Linuxcnc* verfügbar. Eine Reihe von Befehlen erforderte bisher die Verwendung eines numerischen Arguments zur Angabe einer Achsenkoordinate (0-->X, 1-->Y, ..., 8-->W). Diese Befehle wurden vereinfacht, so dass nur noch der Koordinatenbuchstabe als Argument verwendet wird.

Befehle, die jetzt ein Koordinaten-Buchstaben-Argument verwenden, sind:

1. emc_pos_offset
2. emc_abs_cmd_pos
3. emc_abs_act_pos
4. emc_rel_cmd_pos
5. emc_rel_act_pos
6. emc_tool_offset
7. emc_probed_pos

1.5.7.5 Touchy

Die Touchy-GUI unterstützt weiterhin die Identitätskinematik-Konfigurationen, die sie vor der Integration von joints_axes unterstützt hat. Joggen wird im Teleop-Modus ausgeführt.

1.5.7.6 Gscreen

Die Gscreen-GUI unterstützt weiterhin die Identitätskinematik-Konfigurationen, die sie vor der Integration von joints_axes unterstützt hat. Jogging wird im Teleop-Modus durchgeführt.

1.5.7.7 GMOCCAPY

Die GMOCCAPY-GUI unterstützt weiterhin die Identitätskinematik-Konfigurationen, die sie vor der Integration von joints_axes unterstützt hat. Joggen wird im Teleop-Modus ausgeführt.

1.5.7.8 `shuttlexpress`-Treiber umbenannt in `shuttle`

Der HAL-Treiber für das Contour Designs ShuttleXpress-Gerät wurde von ”shuttlexpress” in einfach ”shuttle” umbenannt. Wenn Ihre HAL-Dateien eine Variante von ”loadusr shuttlexpress” enthalten, ersetzen Sie ”shuttlexpress” durch ”shuttle”.

Der ShuttlePRO, eine größere Version des ShuttleXpress, wird nun unterstützt, so dass der alte Treibername nicht mehr korrekt ist.

1.5.7.9 linuxcncrsh

”Referenzfahrt aller Achsen” (engl. home all) wird jetzt mit dem Unterbefehl ”Set Home” (Referenzpunkt setzen) unterstützt, indem -1 für die Gelenknummer verwendet wird.

Die Jogging-Befehle wurden geändert, um sowohl das gemeinsame (freie) als auch das Teleop-Jogging (Welt) zu ermöglichen.

```
war: set jog      joint_number             speed
ist: set jog      joint_number|axis_letter speed

war: set jog_incr joint_number             speed increment
ist: set jog_incr joint_number|axis_letter speed increment

war: set jog_stop
ist: set jog_stop joint_number|axis_letter
```

Anmerkung
Test auf Teleop-Modus mit dem Befehl: `get teleop_enable`
Wenn TELEOP_ENABLE=YES, benutze axis_letter;
Andernfalls joint_number verwenden

Anmerkung
Früher hat der Befehl *set jog 0 1.234* die nullte Achse (X) mit der geforderten Geschwindigkeit=1.234 in jedem Modus (frei oder teleop) verfahren. Dieser Befehl versucht nun, das nullte Gelenk (Joint0) zu verfahren, vorausgesetzt der Modus ist frei (nicht teleop). Um die X-Achse zu verfahren, muss der Modus teleop sein und der entsprechende Befehl lautet: *set jog x 1.234*.

1.5.8 Werkzeuge (engl. tools)

1.5.8.1 Kalibrierung (emccalib.tcl)

The calibration/tuning tool now supports stanzas:

`[JOINT_N], [AXIS_L], [SPINDLE_S], [TUNE]`

wobei *N* eine Gelenknummer (0 … ([KINS]JOINTS-1)), *L* ein Achsenkoordinatenbuchstabe (X,Y,Z,A,B,C,U,V,W
und *S* eine Spindelnummer (0 … 9) ist.

Anmerkung
Die Anzahl der zulässigen Spindeln beträgt 8, aber ältere Konfigurationen können eine Stanza [SPINDLE_9] enthalten, die nicht mit einer tatsächlichen Spindelanzahl in Verbindung steht.

Anmerkung
The [TUNE] stanza may be used for specifying tunable items not relevant to the other supported stanzas.

1.5.9 Veraltete GUIs (entfernt für 2.8.x)

Die GUIs *mini*, *keystick* und *xlinuxcnc* wurden in Verbindung mit Updates für joints_axes entfernt. Der gesamte zugehörige Quellcode, die Beispiele und die Dokumentation sind im Git-Repository verfügbar.

1.5.10 Veraltete GUIs (markiert bei 2.8.x)

Die Benutzeroberfläche *linuxcncIcd* ist ein Kandidat für die Entfernung. Sollte diese Komponente entfernt werden, bleiben alle zugehörigen Quellcodes, Beispiele und Dokumentationen im Git-Repository verfügbar.

1.5.11 Simulator-Konfigurationen (Aktualisierungen für Gelenke Achsen 2.8.x)

1.5.11.1 Vor joints_axes

Vor der Integration von joints_axes unterstützten die in den Sim-Konfigurationen verwendeten HAL-Dateien typischerweise eine gewöhnliche Fräsmaschine - ein kartesisches System mit trivialer Kinematik und drei Achsen mit der Bezeichnung *X Y Z*. Typische HAL-Datei-Einträge:

```
[HAL]
HALFILE = core_sim.hal
HALFILE = sim_spindle_encoder.hal
HALFILE = axis_manualtoolchange.hal
HALFILE = simulated_home.hal
```

Drehbankkonfigurationen hatten oft dieselben HAL-Dateien und verwendeten die zweckmäßige Methode, 3 Achsen mit unbenutztem ”Y” anzugeben. Komplexere Sim-Konfigurationen boten je nach Konfigurationszweck spezifische Sätze von HAL-Dateien.

1.5.11.2 Nach joints_axes

Mit der Einbindung der joints_axes-Funktionalität nutzen viele der in der Distribution enthaltenen Simulationen nun die Vorteile einer Allzweck-HAL-Datei, die automatisch zahlreiche Konfigurationen unterstützt. Eine typische sim config HALFILE-Spezifikation ist:

```
[HAL]
HALFILE = LIB:basic_sim.tcl
```

Die HALFILE basic_sim.tcl unterstützt eine Reihe von häufig benötigten Funktionen für eine beliebige Anzahl von Gelenken, die angegeben werden durch:

```
[KINS]
...
JOINTS = anzahl_an_gelenken
...
```

Zu den unterstützten Funktionen gehören:

1. *ddts* — die HAL-Komponenten des Differenzierers werden für jedes Gelenk geladen und verbunden (und xy, xyz für Trivkins-Maschinen)
2. *simulated_home* — eine sim_home_switch HAL-Komponente wird geladen und für jedes Gelenk verbunden. Die Homing-Bedingungen werden durch die üblichen [JOINT_n]HOME_* INI-Dateielemente spezifiziert.
3. *use_hal_manualtoolchange* – der Benutzerbereich hal_manualtoolchange Komponente wird geladen und verbunden.
4. *sim_spindle* — die sim_spindle-Komponente wird geladen und mit zusätzlich geladenen HAL-Komponen verbunden, um die Trägheit einer rotierenden Spindelmasse zu simulieren.

Die Funktionen sind standardmäßig aktiviert, können aber mit folgenden Optionen ausgeschlossen werden: *-no_make_ddts*, *-no_simulated_home*, *-no_use_hal_manualtoolchange*, *-no_sim_spindle*.

Um beispielsweise die Erstellung von ddts wegzulassen:

```
HALFILE = LIB:basic_sim.tcl -no_make_ddts
```

Das Weglassen einer oder mehrerer Kernfunktionen ermöglicht das Testen ohne diese Funktion oder das Hinzufügen neuer HALFILEs zur Implementierung oder Erweiterung der Funktionalität.

Wenn LIB:basic_sim.tcl verwendet wird, so wird eine entsprechende HAL-Datei erstellt (im Konfigurationsverzeichnis) erstellt, um die ausgegebenen halcmd-Befehle anzuzeigen. Der Name der Datei Name basiert auf dem Namen der INI-Datei mit angehängtem *_cmds*, dem Basisnamen und einer konventionellen *.hal*-Dateierweiterung. Beispiel:

```
inifilename:            example.ini
equivalent_halfilename: example_cmds.hal
```

Die entsprechende HAL-Datei ersetzt frühere Instanzen von Dateien mit demselben Dateinamen. Es werden INI-Datei-Variablen, die in der INI-Datei angegeben und von halcmd interpretiert werden, automatisch in der erstellten HAL-Datei ersetzt. Wenn es [HAL]HALFILEs gibt, die vor LIB:basic_sim.tcl angegeben sind, so werden deren halcmd Befehle ebenfalls einbezogen.

Die entsprechende HAL-Datei kann verwendet werden, um eine neue Konfiguration zu erstellen, die auf der mit LIB:basic_sim.tcl erstellten ursprünglichen Konfiguration basiert:

1. Führen Sie die Simulatorkonfiguration aus, um eine neue äquivalente HAL-Datei zu erstellen, z. B.: ”example_cmds.hal”.

2. Um diese neue äquivalente HAL-Datei in der ursprünglichen INI-Datei der Simulatorkonfiguration (oder einer Kopie davon) zu verwenden, bearbeiten Sie sie, um sie zu ändern:

```
[HAL]
HALFILE = LIB:basic_sim.tcl other_parameters
```

zu:

```
[HAL]
HALFILE = ./example_cmds.hal
```

Alle von LIB:basic_sim.tcl hergestellten Komponenten und Verbindungen können mit halcmd angezeigt werden. Die gesamte HAL-Konfiguration (mit Ausnahme der mit loadusr geladenen Userspace-Komponenten) kann mit in einer Datei gespeichert werden:

```
$ halcmd save > hal.save
```

Die Verwendung von LIB:basic_sim.tcl reduziert den Aufwand für die Erstellung einer Simulationskonfiguration, da sie die meisten der erforderlichen Komponentenladungen und HAL-Verbindungen übernimmt.

Die Sim-Konfiguration *Sample Configurations/sim/axis/minimal_xyz.ini* demonstriert eine funktionierende xyz-Konfiguration, die LIB:basic_sim.tcl mit einer minimalen Anzahl von INI-Datei-Einstellungen verwendet.

1.5.12 Verschiedene Updates für 2.8.x

Commits zu unveröffentlichten Entwicklungs-Zweigen (engl. branches) können Änderungen vornehmen, von denen Tester und early adopters der unveröffentlichten Software profitieren (oder Fehler feststellen).

1.5.12.1 Bewegungs-Pins (engl. motion pins)

Neue Pins (weitere Informationen finden Sie in der Motion-Manpage):

--- axis.L.jog-accel-fraction joint.N.jog-accel-fraction ---

1.5.12.2 HAL-Pins

Namensänderungen:

```
war: axis.L.vel-cmd
ist: axis.l.teleop-vel-cmd
```

Neue Pins:

```
motion.homing-inhibit (siehe motion manpage)
```

1.5.12.3 HAL-Komponenten

1. siggen: neuer Pin *reset* zum Setzen von Ausgangssignalwerten auf vordefinierten Zustand
2. biquad: Pins’type,f0,Q,s1,s2’ waren zuvor Parameter
3. userkins: Vorlage für benutzerdefinierte Kinematikmodule mit halcompile

1.5.12.4 XHC-HB04 vorbereitete Unterstützung

Entfernen Sie den ungenutzten Pin *jogenable-off*.

Fügen Sie den Pin *amux-enable* hinzu, so dass die gemultiplexten Beschleunigungsreduktionen nun durch die UND-Verknüpfung der Pins *is-manual* und *amux-enable* aktiviert werden. Diese beiden Pins sind normalerweise mit *halui.mode.is-manual* bzw. *halui.mode.is-teleop* verbunden.

Signal pendant:jogenable-off für entfernten Pin *pendant_util.jogenable-off* entfernen.

Unterstützung neuer Bewegungspins für reduzierte Beschleunigungen (axis.*L*.jog-accel-fraction, joint.*N*.jog accel-fraction) für das Joggen der Räder. Die Verwendung von [APPLICATIONS]APP=xhc-hb04-accels wird nicht mehr unterstützt. Reduzierte Beschleunigungen werden nur noch für das Rad-Jogging angewendet (nicht für nml-Befehle, die von GUIs ausgegeben werden).

1.5.12.5 XHC-WHB04B-6 pendant Unterstützung

Siehe die Dokumentation für die Komponente xhc-whb04b-6.

1.5.12.6 bldc3_hall

Die Komponente bldc_hall3 wurde entfernt. Die Komponente **bldc** ist flexibler und besser getestet.

1.5.12.7 [JOINT_n] HOME_SEQUENCE Startwerte

Die Werte der Startsequenz können nur 0, 1 (oder -1) sein. Weitere Informationen finden Sie in der Dokumentation zur ”Referenzfahrt-Konfiguration”.

1.5.12.8 [JOINT_n]HOME_SEQUENCE Negative Werte

Gelenke mit einer negativen HOME_SEQUENCE dürfen nicht im Gelenkmodus verfahren werden, um eine Fehlausrichtung (Racking) in üblichen Gantry-Konfigurationen zu verhindern. Wie immer müssen Maschinen mit einem beliebigen Kinematik-Typ vor der Aktivierung des konventionellen Weltmodus-Jogging referenziert werden.

1.5.12.9 TWOPASS-Unterstützung für komplexe loadrt config= items

Twopass-Unterstützung für loadrt config modparams mit mehreren durch Leerzeichen getrennten und in Anführungszeichen eingeschlossenen Einstellungen hinzugefügt. Beispiel:

```
loadrt hm2_eth board_ip=10.10.10.10 config=”num_encoders=2 num_pwmgens=2 num_stepgens=3”
```

1.5.13 Änderungen nach 2.8.x (Master-Zweig Entwicklung)

Der Master-Zweig ist mit einer Vorabversion gekennzeichnet, normalerweise 2.9~pre*

1.5.13.1 Python3 und GTK3

2.9 wurde auf Python3 und GTK3 umgestellt. Dies betrifft Sie nur, wenn Sie benutzerdefinierte Glade- oder Python-Handler in Ihrer Konfiguration haben.

1. Führen Sie py3clean in Ihrem Konfigurationsverzeichnis aus, um alle temporären Dateien zu entfernen.
2. Führen Sie py3clean in Ihrem LinuxCNC-Quellverzeichnis aus, wenn Sie aus den Quellen kompilieren.
3. Führen Sie 2to3 -w für alle .py-Dateien aus, die Sie geschrieben haben.
4. Stellen Sie sicher, dass der Interpreter in der ersten Zeile des Skripts Python 3 und nicht Python 2 ist.
5. Öffnen Sie die UI-Datei in Glade, und speichern Sie sie. Es sollte alles, was konvertiert werden kann, automatisch konvertieren und Ihnen Warnungen geben.

1.5.13.2 LinuxCNC-Startup

Das Hauptskript **linuxcnc** unterstützt eine neue Option (-H dirname) zur Angabe eines zusätzlichen benutzerdefinierten Verzeichnisses für HAL-Dateien. Dieses Verzeichnis wird vor der üblichen Suche nach 1) dem INI-Verzeichnis und 2) dem System-HAL-Dateibibliotheksverzeichnis durchsucht.

1.5.13.3 G-Code Änderungen

G43.2 (zusätzliche Offsets) akzeptiert nun transiente Offsets, die sowohl durch Achsenwörter als auch aus der Werkzeugtabelle hinzugefügt werden sollen.

1.5.13.4 Konfigurations-Updates

Neu: [JOINT_n]HOME_INDEX_NO_ENCODER_RESET — unterstützt Encoder mit einem Index, der bei Empfang eines Indeximpulses nach Aktivierung von index_enable nicht zurückgesetzt wird.

axis.py Vorgabe für [DISPLAY]GEOMETRY war: ”XYZBCUVW”,ist: ”XYZABCUVW”

1.5.13.5 Code-Aktualisierungen

Die Verwaltung der internen Speicherung von Werkzeugdaten und die Kommunikation derselben zwischen EMCIO und TASK wurde überarbeitet, so dass nun Memory-Mapped Storage verwendet wird. Die frühere Verwendung von nml-Nachrichten für Tooldaten ist veraltet und wird möglicherweise vor einer neuen Version entfernt.

Codeverweise auf die sequenziellen Indizes für interne Werkzeugdaten wurden geklärt, aber die alten Variablennamen für **selected_pocket** und **current_pocket** bleiben bestehen. Variablen mit diesen Namen beziehen sich auf den sequenziellen Index für interne Werkzeugdaten und nicht auf eine tatsächliche Taschennummer. Diese Variablennamen können in Zukunft umbenannt werden und erfordern Änderungen an benutzerdefinierten Python-Remap-Anwendungen, welche die Handhabung der Werkzeuge ändern.

Eine neue optionale Schnittstelle unterstützt die Verwaltung von Werkzeugdaten durch eine externe Datenbankanwendung.

Die Datei ioControl_v2.cc, die das Userspace-Programm iov2 bereitstellt, hat keinen Maintainer und ihre Verwendung ist veraltet - sie wird möglicherweise vor der nächsten Version entfernt.

Unterstützung für Rückwärtslauf im Trajektorienplaner, den Aufgaben- und Bewegungsmodulen, der Python-Schnittstelle, der AXIS-GUI und der Test-Suite hinzugefügt.

Die maximale Anzahl der Gelenke (EMCMOT_MAX_JOINTS) wurde von 9 auf 16 erhöht. Die AXIS GUI unterstützt nun die Anzeige von bis zu 16 Gelenken.

Ein neuer motmod-Parameter (num_extrajoints) spezifiziert Gelenke, die mit konventionellen Referenzfahrtmethoden angefahren werden, aber nach der Referenzfahrt über neue HAL-Pins (joint.*N*.posthome cmd) gesteuert werden. Solche Gelenke können von unabhängigen Bewegungsplanern/-controllern in HAL verwaltet und von G-Code aus mit Hilfe von benutzerdefinierten M-Codes manipuliert werden. Weitere Informationen finden Sie in der Manpage motion.

Eine Referenzfahrt (engl. homing)-API wird von src/emc/motion/homing.h bereitgestellt, um benutzerdefinierten Homing-Code zu unterstützen, der src/emc/motion/homing.c durch eine benutzerdefinierte homing.c-Datei ersetzt.

Das Bewegungsmodul unterstützt Kinematikmodule, die neue Funktionen kinematicsSwitchable() und kinematicsSwitch() definieren, um ihren Kinematik-Typ zu wechseln. Ein HAL-Pin, motion.switchkinstype, ist für die Verwendung solcher Kinematikmodule vorgesehen.

Kinematikmodule, die Kinematikumschaltung implementieren, verwenden das Objekt switchkins.o, um die erforderlichen rtapi_main() und zugehörige Funktionen bereitzustellen. Kinematikmodule, die Kinematikumschaltung nicht unterstützen, verwenden das Makro ”KINS_NOT_SWITCHABLE”, das von kinematics.h bereitgestellt wird.

Mehrere Kinematikmodule sind nun zwischen ihrer gleichnamigen Kinematik und einem alternativen Identitätskinematikmodus umschaltbar.

Kinematische Module, die Switchkins unterstützen:

1. xyzac-trt-kins table-rotary-tilting (ersetzt xyzac-trt-kins)
2. xyzbc-trt-kins table-rotary-tilting (ersetzt xyzbc-trt-kins)
3. genserkins verallgemeinerte Kinematik mit seriellen Gliedern
4. genhexkins verallgemeinerte parallele Hexapod-Kinematik
5. Scarakins Scara Roboter
6. pumakins puma roboter
7. 5axiskins bridgemill (xyzbcw 6 Achsen)

Die obigen switchkins-Module (und trivkins) unterstützen einen coordinates=-Parameter, der optional eine geordnete Menge von Koordinatenbuchstaben angibt, die nacheinander Gelenknummern zugeordnet werden (beginnend mit joint0).

Die obigen switchkins-Module enthalten Vorkehrungen für die Kompilierzeitunterstützung eines zusätzlichen benutzerspezifischen Kinematik-Typs, der in der make-Befehlszeile durch die Umgebungsvariable userkfuncs identifiziert wird. (Siehe src/Makefile)

Das Userspace-Testprogramm bin/genserkins wurde in eine einzige Datei (ugenserkins.c) isoliert, da die ursprüngliche Quelldatei (genserkins.c) für die Unterstützung von switchkins überarbeitet wurde. Das Userspace-Testprogramm wurde nicht aktiv gepflegt und seine Verwendung ist veraltet. Die Datei ugenserkins.c wird möglicherweise in Zukunft entfernt werden.

Der Trajektorienplaner ist jetzt als ladbares Modul implementiert (Standard:tpmod). Ein alternativer (vom Benutzer erstellter) Planer kann mit der INI-Einstellung [TRAJ]TPMOD= modulename oder der Option *linuxcnc -t modulename* geladen werden. Die Beispieldatei src/hal/components/tpcomp.com illustriert eine Methode zur Erstellung eines Moduls mit halcompile.

Die Referenzfahrt (engl. homing)-Funktionen werden jetzt durch ein ladbares Modul implementiert (Standard:homemod). Ein alternativer (vom Benutzer erstellter) Planer kann mit der INI-Einstellung

[EMCMOT]HOMEMOD=modulename oder der Option *linuxcnc -m modulename* geladen werden. Die Beispieldatei src/hal/components/homecomp.comp ist ein Minimalbeispiel für ein Homing-Modul, das mit halcompile erstellt werden kann.

lib/hallib/sim_lib.tcl: Simulation des Encoder-Index, wenn [JOINT_n]HOME_USE_INDEX angegeben ist.

lib/python/vismach.py: neuer HAL-Pin vismach.plotclear

1.5.13.6 HAL

sim_home_switch: I/O-Pin für Index-Aktivierung hinzugefügt

motion.feed-upm — aktueller Vorschub in Einheiten pro Minute

1.5.13.7 Konfigurationen

[DISPLAY]GEOMETRY Einstellungen, die das *!* Zeichen geben an, dass die angezeigten Drehungen G5x-, G92-Offsets berücksichtigen.

sim/configs/axis/axis_9axis: Demonstriert den simulierten Encoder-Index

1.5.14 Änderungen nach 2.8.x

Künftige Versionen dieses Dokuments werden die Änderungen berücksichtigen, die nach der letzten Version 2.8.x am Entwicklungszweig vorgenommen wurden.

1.6 Linux FAQ

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

1.6.1 Automatische Anmeldung

1.6.1.1 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.

1.6.1.2 Ubuntu

Wenn Sie LinuxCNC mit der Ubuntu LiveCD 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).

1.6.2 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.

1.6.3 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.

1.6.4 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.

Anmerkung
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.

1.6.5 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.

1.6.6 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.

1.6.6.1 Der Weg über die Befehlszeile

Öffnen Sie ein Terminal und geben Sie ein

```
sudo gedit
```

Öffnen Sie die Datei mit Datei > Öffnen > Bearbeiten

1.6.6.2 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

1.6.6.3 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.

1.6.7 Terminal-Befehle

1.6.7.1 Arbeitsverzeichnis

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

```
pwd
```

1.6.7.2 Wechsel von Verzeichnissen

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
```

1.6.7.3 Auflisten von Dateien in einem Verzeichnis

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

```
dir
```

oder

```
ls
```

1.6.7.4 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.

1.6.7.5 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.

1.6.7.6 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
```

1.6.8 Bequemlichkeiten

1.6.8.1 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.

1.6.9 Hardware-Probleme

1.6.9.1 Hardware-Informationen

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

```
lspci -v
```

1.6.9.2 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

1.6.10 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.
```

Kapitel 2

Allgemeine Benutzerinformationen

2.1 User Foreword

LinuxCNC is modular and flexible. These attributes lead many to see it as a confusing jumble of little things and wonder why it is the way it is. This page attempts to answer that question before you get into the thick of things.

LinuxCNC started at the National Institute of Standards and Technology in the USA. It grew up using Unix as its operating system. Unix made it different. Among early Unix developers there grew a set of code writing ideas that some call the Unix way. These early LinuxCNC authors followed those ways.

Eric S. Raymond, in his book The Art of Unix Programming, summarizes the Unix philosophy as the widely-used engineering philosophy, ”Keep it Simple, Stupid” (KISS Principle). He then describes how he believes this overall philosophy is applied as a cultural Unix norm, although unsurprisingly it is not difficult to find severe violations of most of the following in actual Unix practice:

- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Separation: Separate policy from mechanism; separate interfaces from engines.[1]

Mr. Raymond offered several more rules but these four describe essential characteristics of the LinuxCNC motion control system.

The **Modularity** rule is critical. Throughout these handbooks you will find talk of the interpreter or task planner or motion or HAL. Each of these is a module or collection of modules. It’s modularity that allows you to connect together just the parts you need to run your machine.

The **Clarity** rule is essential. LinuxCNC is a work in progress — it is not finished nor will it ever be. It is complete enough to run most of the machines we want it to run. Much of that progress is achieved because many users and code developers are able to look at the work of others and build on what they have done.

The **Composition** rule allows us to build a predictable control system from the many modules available by making them connectable. We achieve connectability by setting up standard interfaces to sets of modules and following those standards.

The **Separation** rule requires that we make distinct parts that do little things. By separating functions debugging is much easier and replacement modules can be dropped into the system and comparisons easily made.

[1] Found at http://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

What does the Unix way mean for you as a user of LinuxCNC. It means that you are able to make choices about how you will use the system. Many of these choices are a part of machine integration, but many also affect the way you will use your machine. As you read you will find many places where you will need to make comparisons. Eventually you will make choices, ”I’ll use this interface rather than that” or, “I’ll write part offsets this way rather than that way.” Throughout these handbooks we describe the range of abilities currently available.

As you begin your journey into using LinuxCNC we offer two cautionary notes:[2]

- Paraphrasing the words of Doug Gwyn on UNIX: ”LinuxCNC was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things.”
- Likewise the words of Steven King: ”LinuxCNC is user-friendly. It just isn’t promiscuous about which users it’s friendly with.”

A series of videos on YouTube provide plenty of evidence a transition to LinuxCNC is possible no matter what your regular computer operating system may be. That said, with the advent of additive manufacturing like 3D printing there is an increasing interest by the broader IT community in CNC machining and it should be possible to find someone with complementary skills/equipment near to you to jointly overcome the initial hurdles.

2.2 LinuxCNC User Introduction

2.2.1 Einführung

This document is focused on the use of LinuxCNC, it is intended for readers who have already installed and configured it. Some information on installation is given in the following chapters. The complete documentation on installation and configuration can be found in the integrator’s manual.

2.2.2 How LinuxCNC Works

LinuxCNC is a suite of highly-customisable applications for the control of a Computer Numerically Controlled (CNC) mills and lathes, 3D printers, robots, laser cutters, plasma cutters and other automated devices. It is capable of providing coordinated control of up to 9 axes of movement.

At its heart, LinuxCNC consists of several key components that are integrated together to form one complete system:

- a Graphical User Interface (GUI), which forms the basic interface between the operator, the software and the CNC machine itself;
- the Hardware Abstraction Layer (HAL), which provides a method of linking all the various internal virtual signals generated and received by LinuxCNC with the outside world, and
- the high level controllers that coordinate the generation and execution of motion control of the CNC machine, namely the motion controller (EMCMOT), the discrete input/output controller (EMCIO) and the task executor (EMCTASK).

The below illustration is a simple block diagram showing what a typical 3-axis CNC mill with stepper motors might look like:

[2] Found at http://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

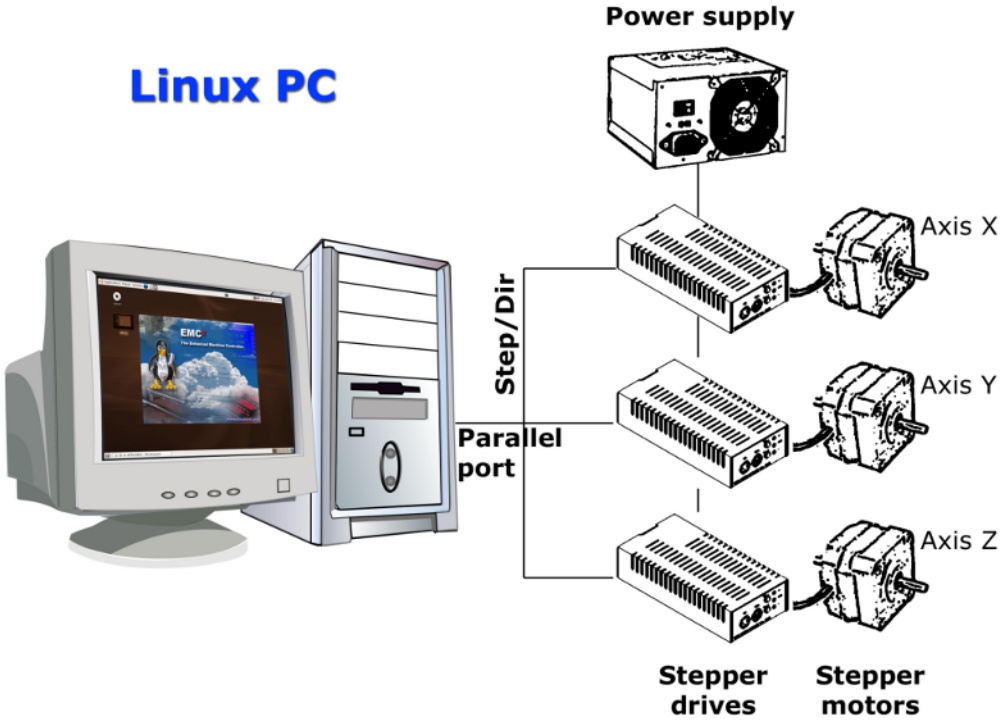


Abbildung 2.1: Simple LinuxCNC Controlled Machine

A computer running LinuxCNC sends a sequence of pulses via the parallel port to the stepper drives, each of which has one stepper motor connected to it. Each drive receives two independent signals; one signal to command the drive to move its associated stepper motor in a clockwise or anti-clockwise direction, and a second signal that defines the speed at which that stepper motor rotates.

While a stepper motor system under parallel port control is illustrated, a LinuxCNC system can also take advantage of a wide variety of dedicated hardware motion control interfaces for increased speed and I/O capabilities. A full list of interfaces supported by LinuxCNC can be found on the Supported Hardware page of the Wiki.

In most circumstances, users will create a configuration specific to their mill setup using either the Stepper Configuration Wizard (for CNC systems operating using the computers' parallel port) or the Mesa Hardware Wizard (for more advanced systems utilising a Mesa Anything I/O PCI card). Running either wizard will create several folders on the computers' hard drive containing a number of configuration files specific to that CNC machine, and an icon placed on the desktop to allow easy launching of LinuxCNC.

For example, if the Stepper Configuration Wizard was used to create a setup for the 3-axis CNC mill illustrated above entitled *My_CNC*, the folders created by the wizard would typically contain the following files:

- **Folder:** `My_CNC`
  - `My_CNC.ini`
    The INI file contains all the basic hardware information regarding the operation of the CNC mill such as the number of steps each stepper motor must turn to complete one full revolution, the maximum rate at which each stepper may operate at, the limits of travel of each axis or the configuration and behaviour of limit switches on each axis.
  - `My_CNC.hal`
    This HAL file contains information that tells LinuxCNC how to link the internal virtual signals to

physical connections beyond the computer. For example, specifying pin 4 on the parallel port to send out the Z axis step direction signal, or directing LinuxCNC to cease driving the X axis motor when a limit switch is triggered on parallel port pin 13.

- **`custom.HAL`**
  Customisations to the mill configuration beyond the scope of the wizard may be performed by including further links to other virtual points within LinuxCNC in this HAL file. When starting a LinuxCNC session, this file is read and processed before the GUI is loaded. An example may include initiating Modbus communications to the spindle motor so that it is confirmed as operational before the GUI is displayed.
- **`custom_postgui.hal`**
  The custom_postgui HAL file allows further customisation of LinuxCNC, but differs from custom.HAL in that it is processed after the GUI is displayed. For example, after establishing Modbus communications to the spindle motor in custom.hal, LinuxCNC can use the custom_postgui file to link the spindle speed readout from the motor drive to a bargraph displayed on the GUI.
- **`postgui_backup.hal`**
  This is provided as a backup copy of the custom_postgui.hal file to allow the user to quickly restore a previously-working postgui HAL configuration. This is especially useful if the user wants to run the Configuration Wizard again under the same *My_CNC* name in order to modify some parameters of the mill. Saving the mill configuration in the Wizard will overwrite the existing custom_postgui file while leaving the postgui_backup file untouched.
- **`tool.tbl`**
  A tool table file contains a parameterised list of any cutting tools used by the mill. These parameters can include cutter diameter and length, and is used to provide a catalogue of data that tells LinuxCNC how to compensate its motion for different sized tools within a milling operation.

- **Folder: `nc_files`**
  The nc_files folder is provided as a default location to store the G-code programs used to drive the mill. It also includes a number of subfolders with G-code examples.

2.2.3 Graphical User Interfaces

A graphical user interface is the part of the LinuxCNC that the machine tool operator interacts with. LinuxCNC comes with several types of user interfaces which may be chosen from by editing certain fields contained in the INI file:

ACHSE
AXIS, the standard keyboard GUI interface. This is also the default GUI launched when a Configuration Wizard is used to create a desktop icon launcher:

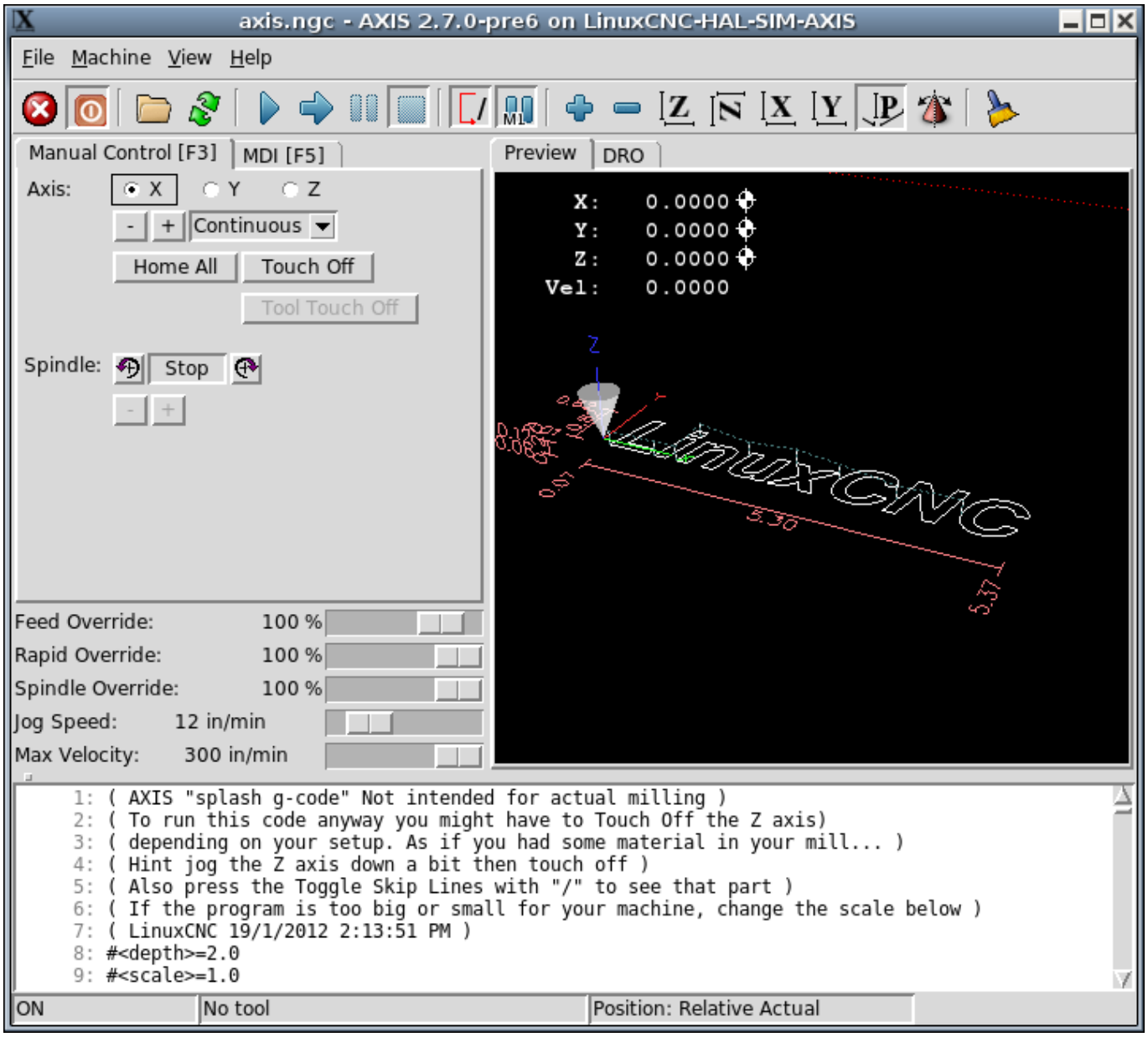


Abbildung 2.2: AXIS, the standard keyboard GUI interface

Touchy

Touchy, a touch screens GUI:

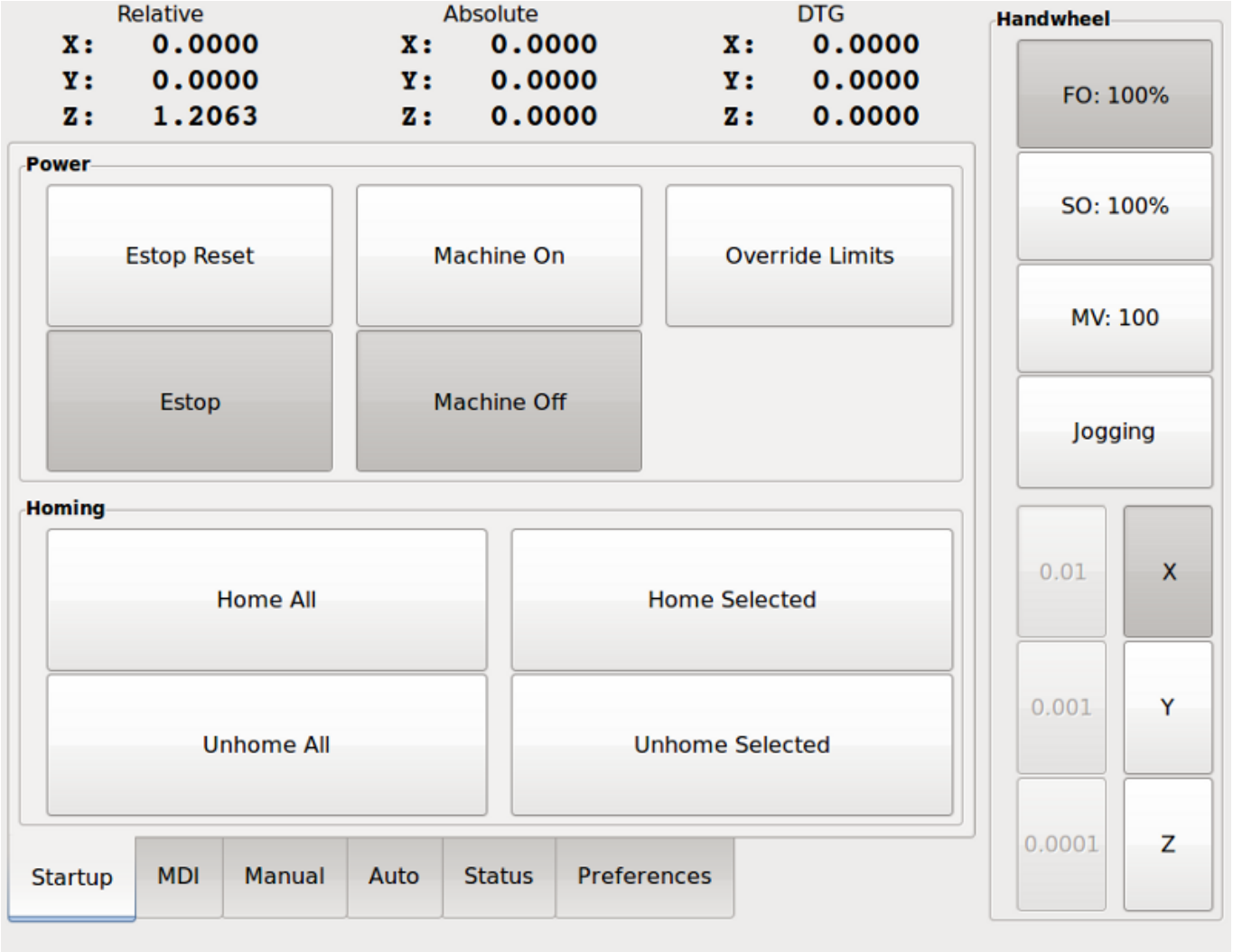


Abbildung 2.3: Touchy, a touch screen GUI

Gscreen

Gscreen, a user-configurable touch screen GUI:

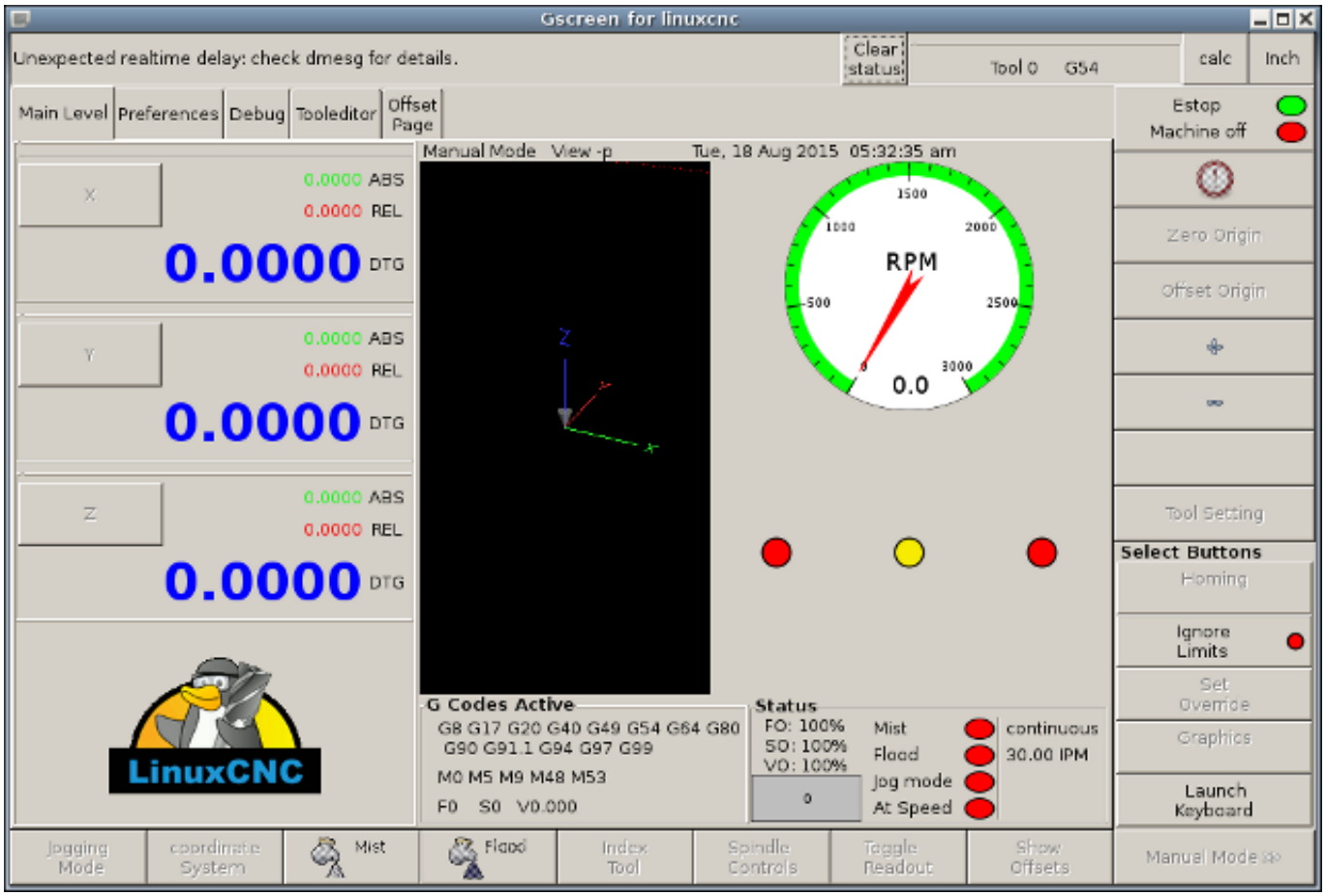


Abbildung 2.4: Gscreen, a configurable base touch screen GUI

GMOCCAPY

GMOCCAPY, a touch screen GUI based on Gscreen. GMOCCAPY is also designed to work equally well in applications where a keyboard and mouse are the preferred methods of controlling the GUI:

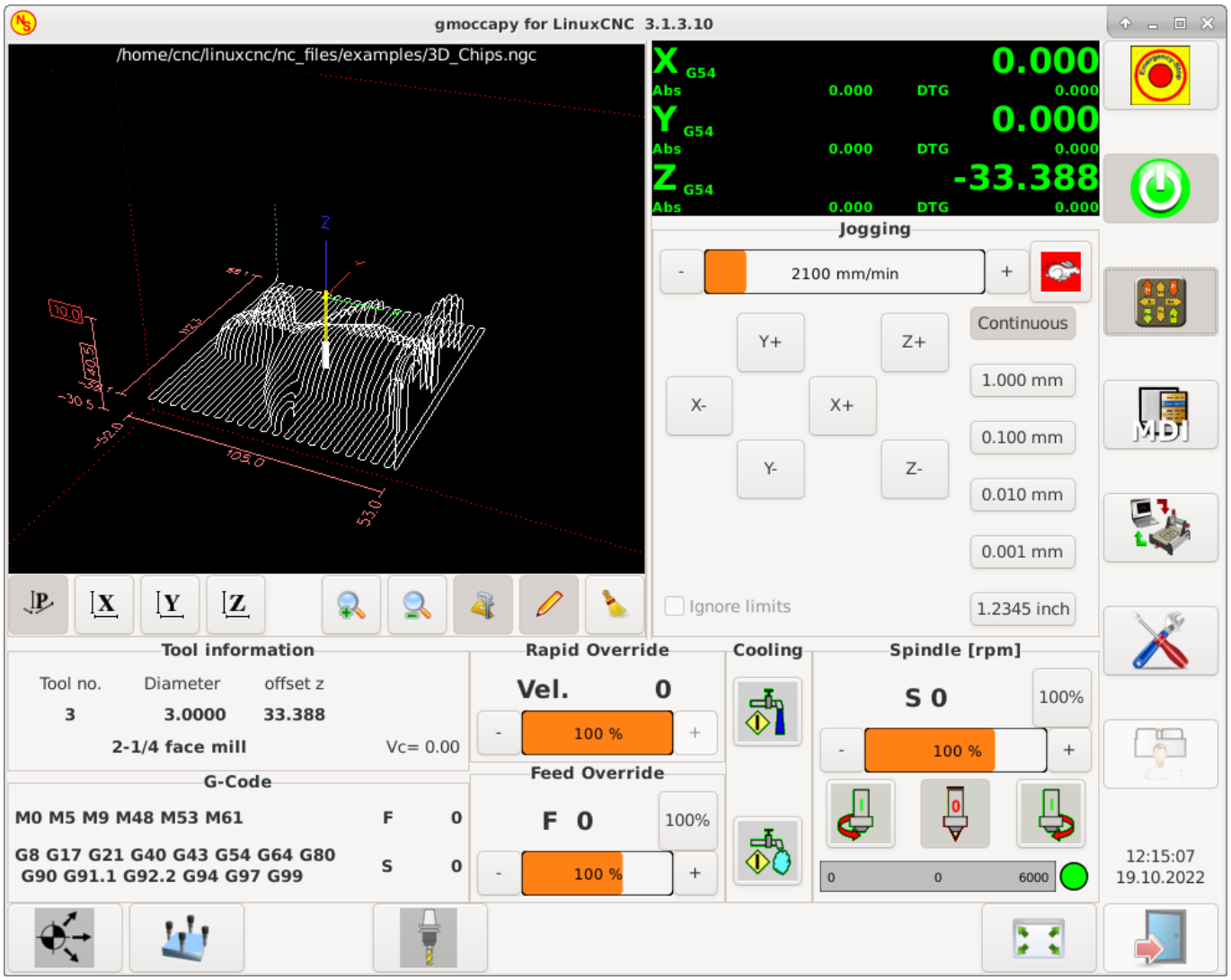


Abbildung 2.5: GMOCCAPY, a touch screen GUI based on Gscreen

NGCGUI

NGCGUI, a subroutine GUI that provides wizard-style programming of G code. NGCGUI may be run as a standalone program or embedded into another GUI as a series of tabs. The following screen shot shows NGCGUI embedded into Axis:

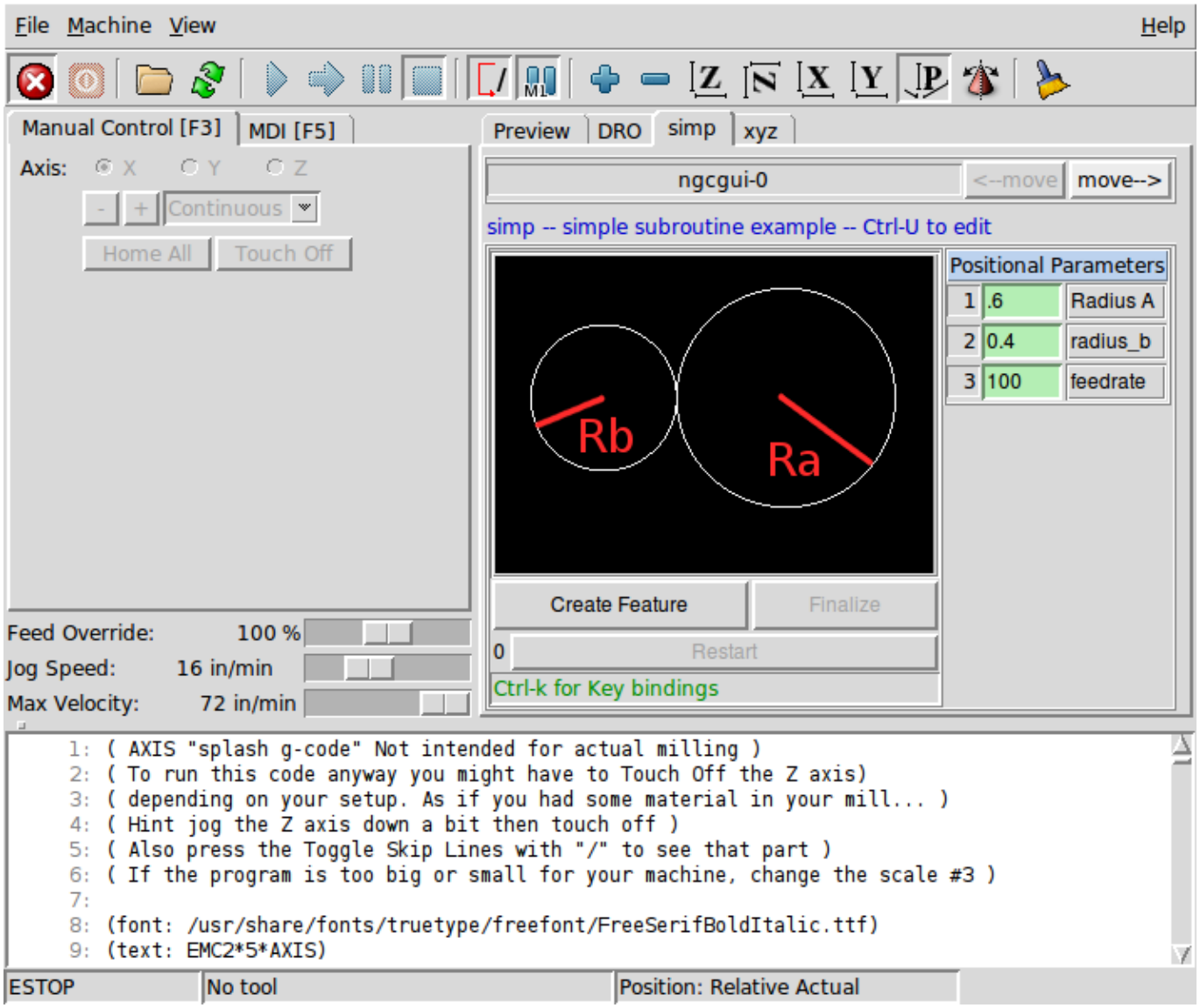


Abbildung 2.6: NGCGUI, a graphical interface integrated into AXIS

TkLinuxCNC

TkLinuxCNC, another interface based on Tcl/Tk. Once the most popular interface after AXIS.

Abbildung 2.7: TkLinuxCNC graphical interface

Xemc
an X-Window program

halui
A HAL based user interface allowing to control LinuxCNC using buttons and switches

linuxcncrsh
A telnet based user interface allowing to send commands from remote computers.

2.2.4 Virtuelle Schalttafeln

As mentioned above, many of LinuxCNC's GUIs may be customized by the user. This may be done to add indicators, readouts, switches or sliders to the basic appearance of one of the GUIs for increased flexibility or functionality. Two styles of Virtual Control Panel are offered in LinuxCNC:

PyVCP
PyVCP, a Python-based virtual control panel that can be added to the Axis GUI. PyVCP only utilises virtual signals contained within the Hardware Abstraction Layer, such as the spindle-at-speed indicator or the Emergency Stop output signal, and has a simple no-frills appearance. This makes it an excellent choice if the user wants to add a Virtual Control Panel with minimal fuss.

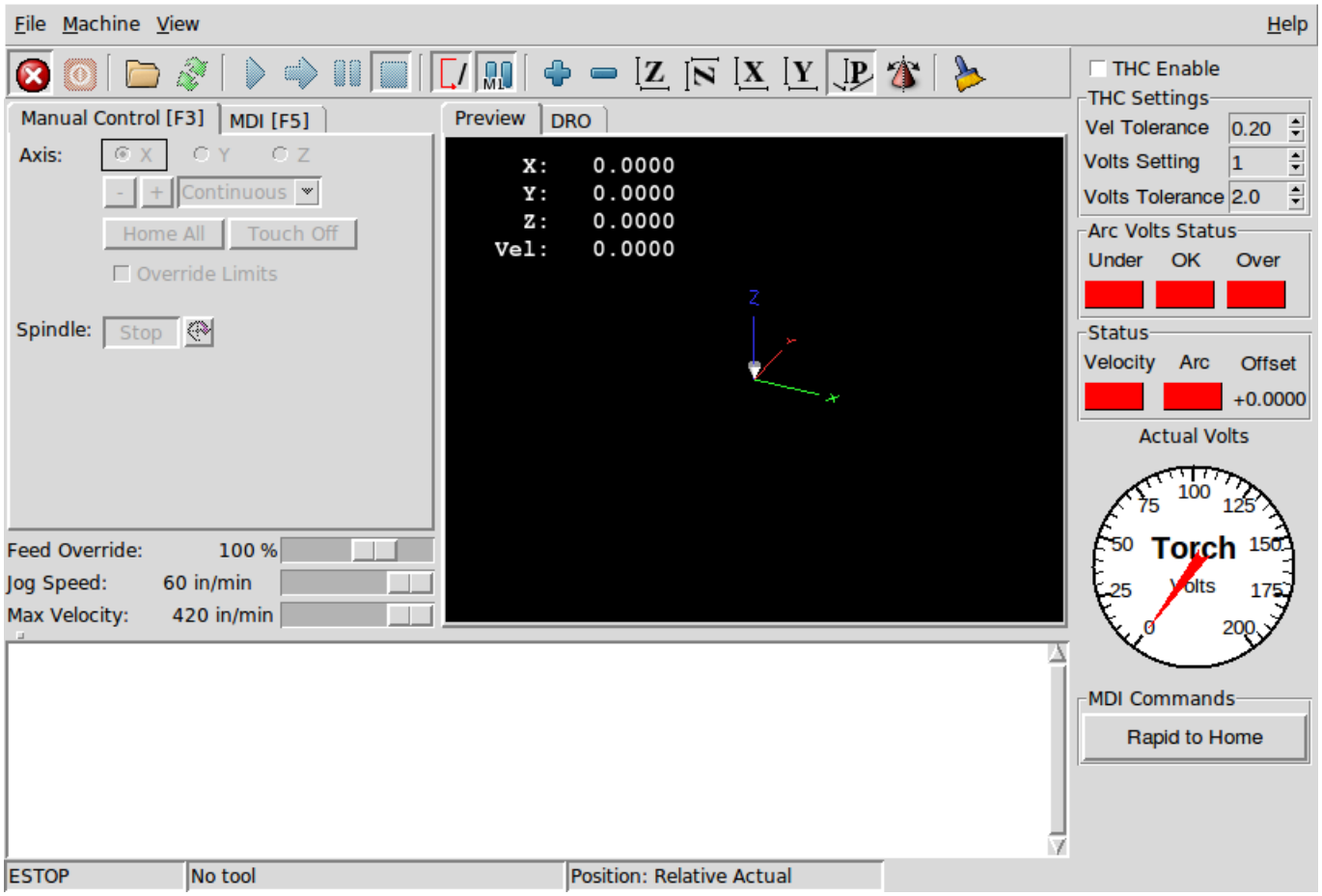


Abbildung 2.8: PyVCP Example Embedded Into AXIS GUI

GladeVCP
GladeVCP, a Glade-based virtual control panel that can be added to the Axis or Touchy GUIs. GladeVCP has the advantage over PyVCP in that it is not limited to the display or control of HAL virtual signals, but can include other external interfaces outside LinuxCNC such as window or network events. GladeVCP is also more flexible in how it may be configured to appear on the GUI:

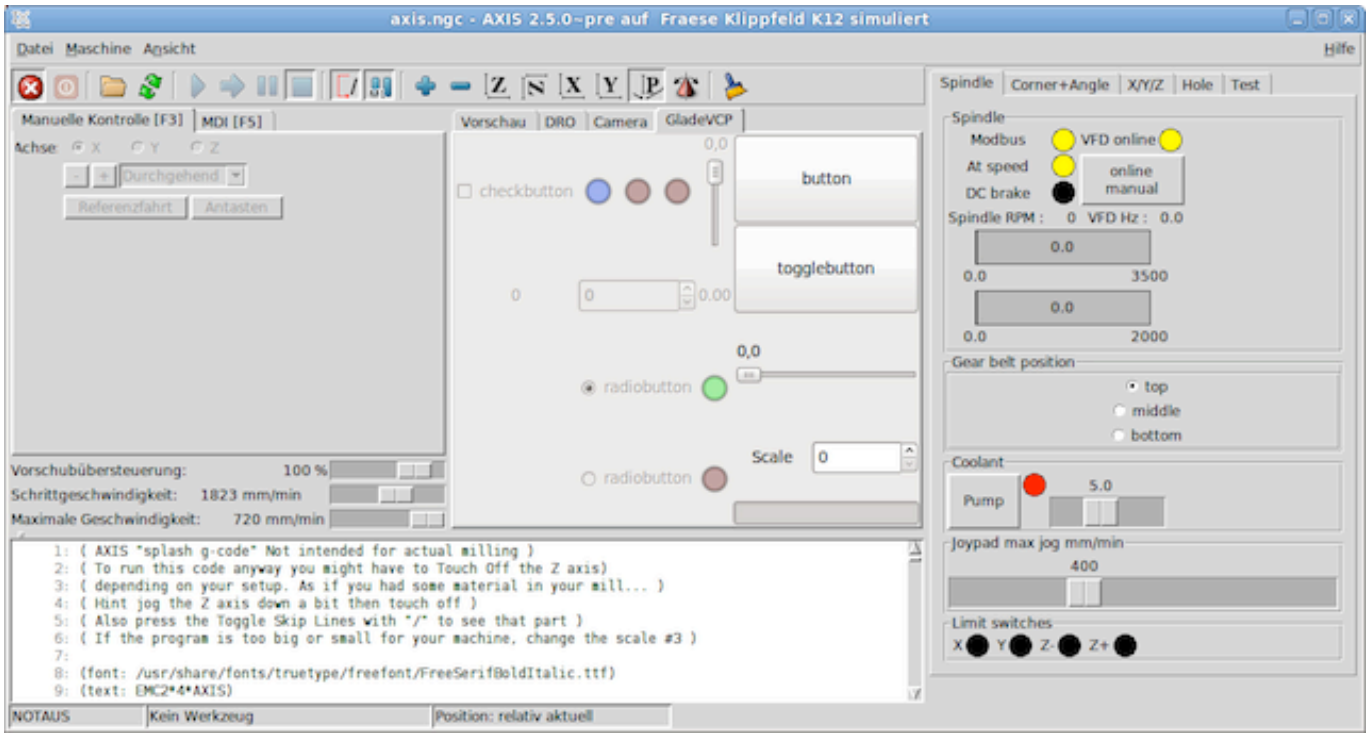


Abbildung 2.9: GladeVCP Example Embedded Into AXIS GUI

2.2.5 Sprachen

LinuxCNC uses translation files to translate LinuxCNC User Interfaces into many languages including French, German, Italian, Finnish, Russian, Romanian, Portuguese and Chinese. Assuming a translation has been created, LinuxCNC will automatically use whatever native language you log in with when starting the Linux operating system. If your language has not been translated, contact a developer on IRC, the mailing list or the User Forum for assistance.

2.2.6 Think Like a CNC Operator

This manual does not pretend to teach you how to use a lathe or a milling machine. Becoming an experienced operator takes a lot of time and requires a lot of work. An author once said, *We learn by experience, if one possesses it all*. Broken tools, vices attacked and the scars are evidence of the lessons learned. A beautiful finish, tight tolerances and caution during the work are evidence of lessons learned. No machine nor program can replace human experience.

Now that you start working with the LinuxCNC software, you have to put yourself in the shoes of an operator. You must be in the role of someone in charge of a machine. It’s a machine that will wait for your commands and then execute the orders that you will give it. In these pages, we will give the explanations which will help you to become a good CNC operator with LinuxCNC.

2.2.7 Modes of Operation

When LinuxCNC is running, there are three different major modes used for inputting commands. These are Manual, Auto, and Manual Data Input (MDI). Changing from one mode to another makes a big difference in the way that the LinuxCNC control behaves. There are specific things that can be done in one mode that cannot be done in another. An operator can home an axis in manual mode but

not in auto or MDI modes. An operator can cause the machine to execute a whole file full of G-codes in the auto mode but not in manual or MDI.

In manual mode, each command is entered separately. In human terms a manual command might be ”turn on coolant” or ”jog X at 25 inches per minute”. These are roughly equivalent to flipping a switch or turning the hand wheel for an axis. These commands are normally handled on one of the graphical interfaces by pressing a button with the mouse or holding down a key on the keyboard. In auto mode, a similar button or key press might be used to load or start the running of a whole program of G-code that is stored in a file. In the MDI mode the operator might type in a block of code and tell the machine to execute it by pressing the <return> or <enter> key on the keyboard.

Some motion control commands are available concurrently and will cause the same changes in motion in all modes. These include Abort, Emergency Stop, and Feed Rate Override. Commands like these should be self explanatory.

The AXIS user interface hides some of the distinctions between Auto and the other modes by making auto-commands available at most times. It also blurs the distinction between Manual and MDI, because some Manual commands like Touch Off are actually implemented by sending MDI commands. It does this by automatically changing to the mode that is needed for the action the user has requested.

2.3 Important User Concepts

This chapter covers important user concepts that should be understood before attempting to run a CNC machine with G-code.

2.3.1 Trajectory Control

2.3.1.1 Trajectory Planning

Trajektorie Planung, im Allgemeinen, ist das Mittel, mit dem LinuxCNC folgt dem Pfad von Ihrem G-Code-Programm angegeben, während immer noch innerhalb der Grenzen 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 reality, the whole move can’t be made at F10, since the machine must accelerate from a stop, move toward X=1, and then decelerate to stop again. Sometimes part of the move is done at F10, but for many moves, especially short ones, the specified feed rate is never reached at all. Having short moves in your G-code can cause your machine to slow down and speed up for the longer moves if the *naive cam detector* is not employed with G64 P*n*.

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.

2.3.1.2 Bahnverfolgung

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.

2.3.1.3 Programmieren des Planers

Die Befehle zur Steuerung der Trajektorie lauten wie folgt:

`G61`
: (Exact Path Mode) `G61` visits the programmed point exactly, even though that means it might temporarily come to a complete stop in order to change direction to the next programmed point.

`G61.1`
: (Exact Stop Mode) `G61.1` tells the planner to come to an exact stop at every segment's end. The path will be followed exactly but complete feed stops can be destructive for the part or tool, depending on the specifics of the machining.

`G64`
: (Blend Without Tolerance Mode) `G64` is the default setting when you start LinuxCNC. G64 is just blending and the naive cam detector is not enabled. G64 and G64 P0 tell the planner to sacrifice path following accuracy in order to keep the feed rate up. This is necessary for some types of material or tooling where exact stops are harmful, and can work great as long as the programmer is careful to keep in mind that the tool's path will be somewhat more curvy than the program specifies. When using G0 (rapid) moves with G64 use caution on clearance moves and allow enough distance to clear obstacles based on the acceleration capabilities of your machine.

`G64 P- Q-`
: (Blend With Tolerance Mode) This enables the *naive cam detector* and enables blending with a tolerance. If you program G64 P0.05, you tell the planner that you want continuous feed, but at programmed corners you want it to slow down enough so that the tool path can stay within 0.05 user units of the programmed path. The exact amount of slowdown depends on the geometry of the programmed corner and the machine constraints, but the only thing the programmer needs to worry about is the tolerance. This gives the programmer complete control over the path following compromise. The blend tolerance can be changed throughout the program as necessary. Beware that a specification of G64 P0 has the same effect as G64 alone (above), which is necessary for backward compatibility for old G-code programs. See the G64 section of the G-code chapter.

Blending without tolerance
: The controlled point will touch each specified movement at at least one point. The machine will never move at such a speed that it cannot come to an exact stop at the end of the current movement (or next movement, if you pause when blending has already started). The distance from the end point of the move is as large as it needs to be to keep up the best contouring feed.

Naive CAM Detector
: Successive G1 moves that involve only the XYZ axes that deviate less than Q- from a straight line are merged into a single straight line. This merged movement replaces the individual G1

movements for the purposes of blending with tolerance. Between successive movements, the controlled point will pass no more than P- from the actual endpoints of the movements. The controlled point will touch at least one point on each movement. The machine will never move at such a speed that it cannot come to an exact stop at the end of the current movement (or next movement, if you pause when blending has already started) On G2/3 moves in the G17 (XY) plane when the maximum deviation of an arc from a straight line is less than the G64 Q- tolerance the arc is broken into two lines (from start of arc to midpoint, and from midpoint to end). those lines are then subject to the naive cam algorithm for lines. Thus, line-arc, arc-arc, and arc-line cases as well as line-line benefit from the *naive cam detector*. This improves contouring performance by simplifying the path.

In the following figure the blue line represents the actual machine velocity. The red lines are the acceleration capability of the machine. The horizontal lines below each plot is the planned move. The upper plot shows how the trajectory planner will slow the machine down when short moves are encountered to stay within the limits of the machines acceleration setting to be able to come to an exact stop at the end of the next move. The bottom plot shows the effect of the Naive Cam Detector to combine the moves and do a better job of keeping the velocity as planned.

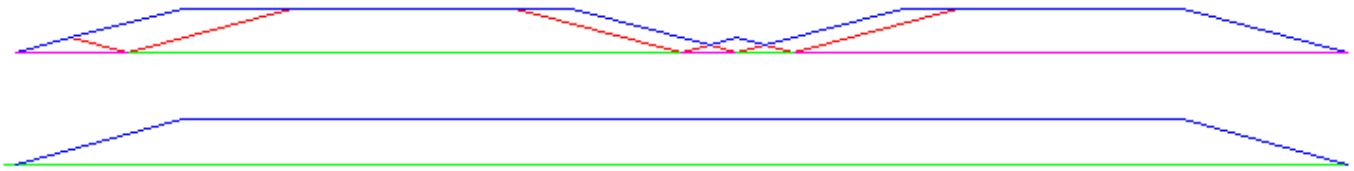


Abbildung 2.10: Naive CAM Detector

2.3.1.4 Planning Moves

Make sure moves are *long enough* to suit your machine/material. Principally because of the rule that the machine will never move at such a speed that it cannot come to a complete stop at the end of the current movement, there is a minimum movement length that will allow the machine to keep up a requested feed rate with a given acceleration setting.

The acceleration and deceleration phase each use half the ini file MAX_ACCELERATION. In a blend that is an exact reversal, this causes the total axis acceleration to equal the ini file MAX_ACCELERATION. In other cases, the actual machine acceleration is somewhat less than the ini file acceleration

To keep up the feed rate, the move must be longer than the distance it takes to accelerate from 0 to the desired feed rate and then stop again. Using A as **1/2** the ini file MAX_ACCELERATION and F as the feed rate **in units per second**, the acceleration time is $\mathbf{t_a = F/A}$ and the acceleration distance is $\mathbf{d_a = F*t_a/2}$. The deceleration time and distance are the same, making the critical distance $\mathbf{d = d_a + d_d = 2 * d_a = F^2/A}$.

For example, for a feed rate of 1 inch per second and an acceleration of **10 inches/sec^2**, the critical distance is $\mathbf{1^2/10 = 1/10 = 0.1}$ **inches**.

For a feed rate of 0.5 inch per second, the critical distance is $\mathbf{5^2/100 = 25/100 = 0.025}$ inches.

2.3.2 G-code

2.3.2.1 Defaults

When LinuxCNC first starts up many G- and M-codes are loaded by default. The current active G- and M-codes can be viewed on the MDI tab in the *Active G-Codes:* window in the AXIS interface. These

G- and M-codes define the behavior of LinuxCNC and it is important that you understand what each one does before running LinuxCNC. The defaults can be changed when running a G-Code file and left in a different state than when you started your LinuxCNC session. The best practice is to set the defaults needed for the job in the preamble of your G-Code file and not assume that the defaults have not changed. Printing out the G-Code Quick Reference page can help you remember what each one is.

2.3.2.2 Feed Rate

How the feed rate is applied depends on if an axis involved with the move is a rotary axis. Read and understand the Feed Rate section if you have a rotary axis or a lathe.

2.3.2.3 Tool Radius Offset

Tool Radius Offset (G41/42) requires that the tool be able to touch somewhere along each programmed move without gouging the two adjacent moves. If that is not possible with the current tool diameter you will get an error. A smaller diameter tool may run without an error on the same path. This means you can program a cutter to pass down a path that is narrower than the cutter without any errors. See the Cutter Compensation Section for more information.

2.3.3 Referenzfahrt (engl. homing)

After starting LinuxCNC each axis must be homed prior to running a program or running a MDI command.

If your machine does not have home switches a match mark on each axis can aid in homing the machine coordinates to the same place each time.

Once homed your soft limits that are set in the ini file will be used.

If you want to deviate from the default behavior, or want to use the Mini interface you will need to set the option NO_FORCE_HOMING = 1 in the [TRAJ] section of your ini file. More information on homing can be found in the Integrator Manual.

2.3.4 Tool Changes

There are several options when doing manual tool changes. See the [EMCIO] section for information on configuration of these options. Also see the G28 and G30 section of the G-code chapter.

2.3.5 Koordinatensysteme

The Coordinate Systems can be confusing at first. Before running a CNC machine you must understand the basics of the coordinate systems used by LinuxCNC. In depth information on the LinuxCNC Coordinate Systems is in the Coordinate System Section of this manual.

2.3.5.1 G53 Machine Coordinate

When you home LinuxCNC you set the G53 Machine Coordinate System to 0 for each axis homed.

No other coordinate systems or tool offsets are changed by homing.

The only time you move in the G53 machine coordinate system is when you program a G53 on the same line as a move. Normally you are in the G54 coordinate system.

2.3.5.2 G54-59.3 User Coordinates

Normally you use the G54 Coordinate System. When an offset is applied to a current user coordinate system a small blue ball with lines will be at the machine origin when your DRO is displaying *Position: Relative Actual* in Axis. If your offsets are temporary use the Zero Coordinate System from the Machine menu or program *G10 L2 P1 X0 Y0 Z0* at the end of your G-code file. Change the *P* number to suit the coordinate system you wish to clear the offset in.

- Offsets stored in a user coordinate system are retained when LinuxCNC is shut down.
- Using the *Touch Off* button in Axis sets an offset for the chosen User Coordinate System.

2.3.5.3 When You Are Lost

If you're having trouble getting 0,0,0 on the DRO when you think you should, you may have some offsets programmed in and need to remove them.

- Move to the Machine origin with G53 G0 X0 Y0 Z0
- Clear any G92 offset with G92.1
- Use the G54 coordinate system with G54
- Set the G54 coordinate system to be the same as the machine coordinate system with G10 L2 P1 X0 Y0 Z0 R0
- Turn off tool offsets with G49
- Turn on the Relative Coordinate Display from the menu

Now you should be at the machine origin X0 Y0 Z0 and the relative coordinate system should be the same as the machine coordinate system.

2.3.6 Machine Configurations

The following diagram shows a typical mill showing direction of travel of the tool and the mill table and limit switches. Notice how the mill table moves in the opposite direction of the Cartesian coordinate system arrows shown by the *Tool Direction* image. This makes the *tool* move in the correct direction in relation to the material.

Note also the position of the limit switches and the direction of activation of their cams. Several combinations are possible, for example it is possible (contrary to the drawing) to place a single fixed limit switch in the middle of the table and two mobile cams to activate it. In this case the limits will be reversed, +X will be on the right of the table and -X on the left. This inversion does not change anything from the point of view of the direction of movement of the tool.

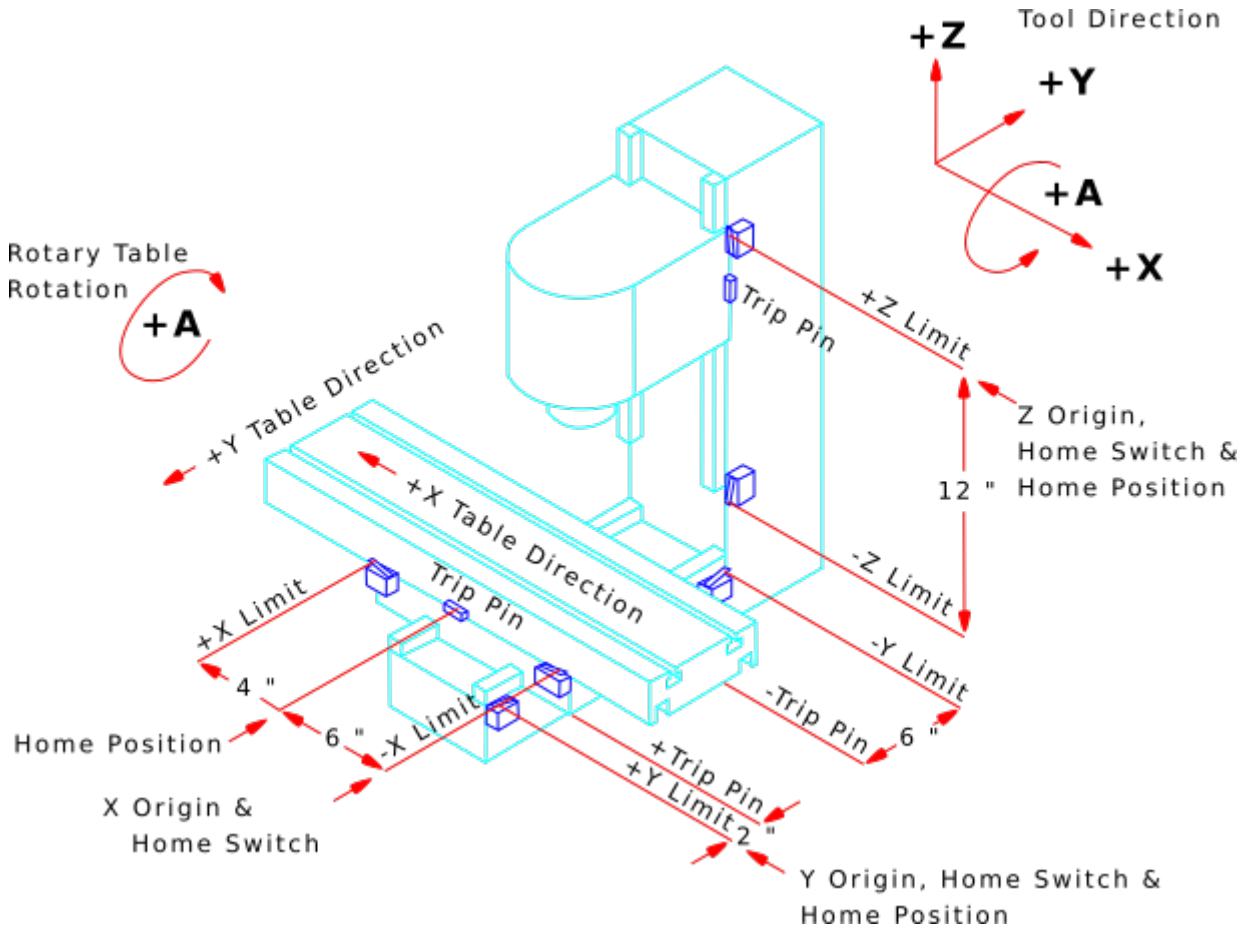


Abbildung 2.11: Typical Mill Configuration

The following diagram shows a typical lathe showing direction of travel of the tool and limit switches.

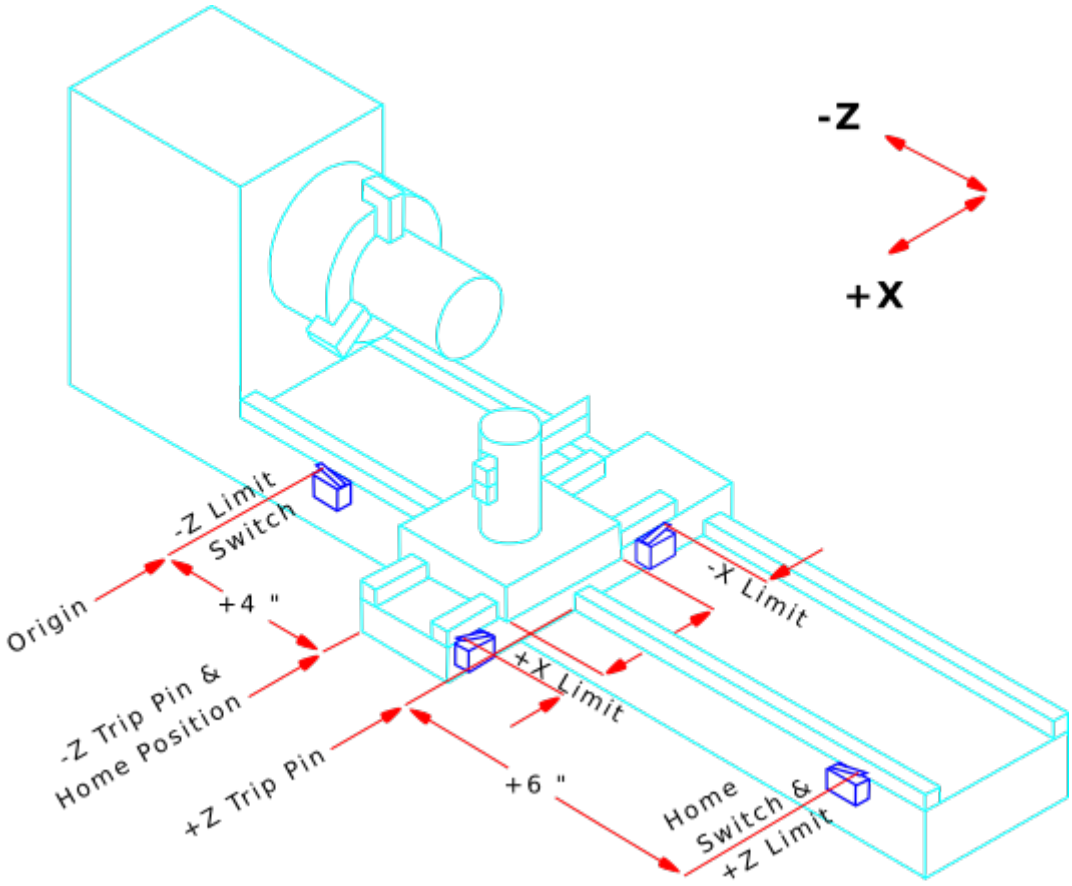


Abbildung 2.12: Typical Lathe Configuration

2.4 Starting LinuxCNC

2.4.1 Running LinuxCNC

LinuxCNC is started with the script file *linuxcnc*.

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

linuxcnc script options

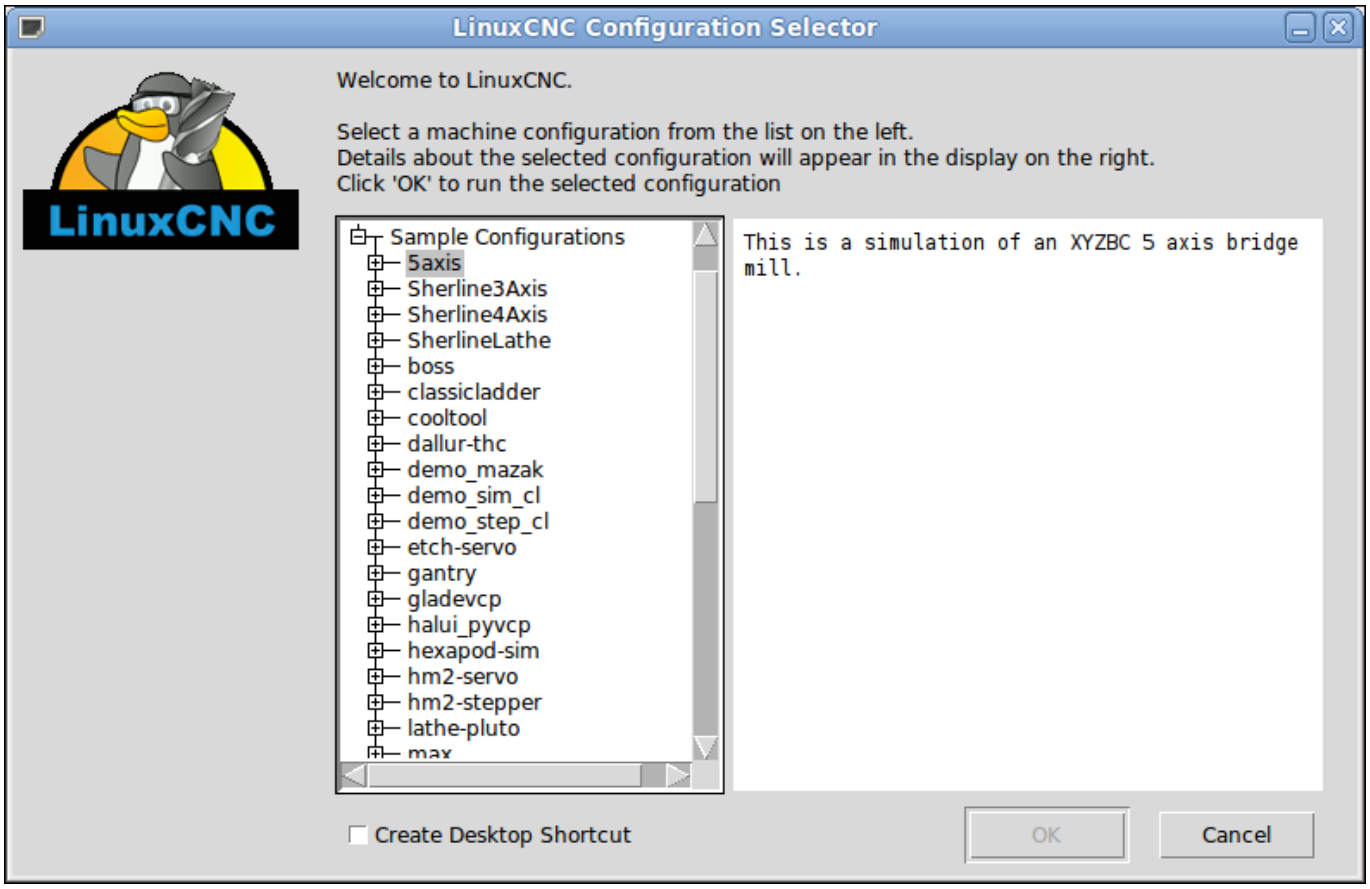
```
linuxcnc: Run LINUXCNC

Usage:
  $ linuxcnc -h
    This help

  $ linuxcnc [Options]
    Choose the configuration inifile graphically

  $ linuxcnc [Options] path/to/your_ini_file
    Name the configuration inifile using its path

  $ linuxcnc [Options] -l
    Use the previously used configuration inifile
```

```
Options:
    -d: Turn on ”debug” mode
    -v: Turn on ”verbose” mode
    -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 Halfiles before searching
                  ini directory and system library:
                  /home/git/linuxcnc-dev/lib/hallib
Note:
    The -H ”dirname” option may be specified multiple times
```

If the linuxcnc script is passed an INI file it reads the INI file and starts LinuxCNC. The INI file [HAL] section specifies the order of loading up HAL files if more than one is used. Once the HAL=xxx.hal files are loaded then the GUI is loaded then the POSTGUI=.xxx.hal file is loaded. If you create PyVCP or GladeVCP objects with HAL pins you must use the postgui HAL file to make any connections to those pins. See the [HAL] section of the INI configuration for more information.

2.4.1.1 Configuration Selector

If no INI file is passed to the linuxcnc script it loads the configuration selector so you can choose and save a sample configuration. Once a sample configuration has been saved it can be modified to suit your application. The configuration files are saved in linuxcnc/configs directory.



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.

2.5.1.1 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.

Anmerkung
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.

Anmerkung
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.

Primäre lineare AchsenAchseprimär linear primär linear Die X-, Y- und Z-Achse erzeugen lineare Bewegungen in drei zueinander orthogonalen Richtungen.

Sekundäre lineare AchsenAchsesekundäre lineare sekundäre lineare Die Achsen U, V und W erzeugen eine lineare Bewegung in drei zueinander orthogonalen Richtungen. Normalerweise sind X und U parallel, Y und V parallel und Z und W parallel.

RotationsachsenAchsenrotierend rotierend Die A-, B- und C-Achsen erzeugen eine Winkelbewegung (Rotation). Normalerweise dreht sich A um eine Linie parallel zu X, B um eine Linie parallel zu Y und C um eine Linie parallel zu Z.

2.5.1.2 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.

2.5.1.3 Kühlmittel

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

2.5.1.4 Vorschub- und Drehzahl-Override

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.

2.5.1.5 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).

2.5.1.6 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

2.5.2.1 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).

2.5.2.2 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.]

2.5.2.3 Gesteuerter Punkt (engl. controlled point)

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).

2.5.2.4 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.

2.5.2.5 Vorschubgeschwindigkeit

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).

1. Wenn sich eine der Achsen XYZ bewegt, wird F in Einheiten pro Minute im kartesischen System XYZ angegeben, und alle anderen Achsen (ABCUVW) bewegen sich so, dass sie koordiniert starten und stoppen.
2. Andernfalls, wenn sich UVW bewegt, wird F in Einheiten pro Minute im kartesischen System von UVW angegeben, und alle anderen Achsen (ABC) bewegen sich so, dass sie koordiniert starten und stoppen.
3. Andernfalls handelt es sich um eine reine Drehbewegung, und das F-Wort wird in Dreheinheiten im pseudokartesischen ABC-System angegeben.

2.5.2.6 Kühlung

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

2.5.2.7 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.

2.5.2.8 Einheiten

Die Einheiten für Abstände entlang der X-, Y- und Z-Achse können in Millimetern oder Zoll gemessen werden. Die Einheiten für alle anderen an der Maschinensteuerung beteiligten Größen können nicht geändert werden. Verschiedene Größen verwenden unterschiedliche spezifische Einheiten. Die Spindeldrehzahl wird in Umdrehungen pro Minute gemessen. Die Positionen der Rotationsachsen werden in Grad gemessen. Vorschubgeschwindigkeiten werden in aktuellen Längeneinheiten pro Minute oder Grad pro Minute oder Längeneinheiten pro Spindelumdrehung ausgedrückt, wie in Abschnitt G93 G94 G95 beschrieben.

2.5.2.9 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.

2.5.2.10 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.

2.5.2.11 Werkzeug-Karussell

Jedem Steckplatz im Werkzeugkarussell wird ein oder null Werkzeuge zugewiesen.

2.5.2.12 Werkzeugwechsel

Einem Bearbeitungszentrum kann der Befehl zum Werkzeugwechsel gegeben werden.

2.5.2.13 Paletten-Shuttle

Die beiden Paletten können auf Befehl ausgetauscht werden.

2.5.2.14 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.

2.5.2.15 Pfad-Steuerungs-Modus (engl. path control mode)

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.

2.5.3.1 Vorschub- und Geschwindigkeits-Neufestsetzungs-Schalter

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.

2.5.3.2 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.

2.5.3.3 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.

Tabelle 2.1: 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 Drehbank-Benutzerinformationen

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 Drehwerkzeugausrichtung

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+.

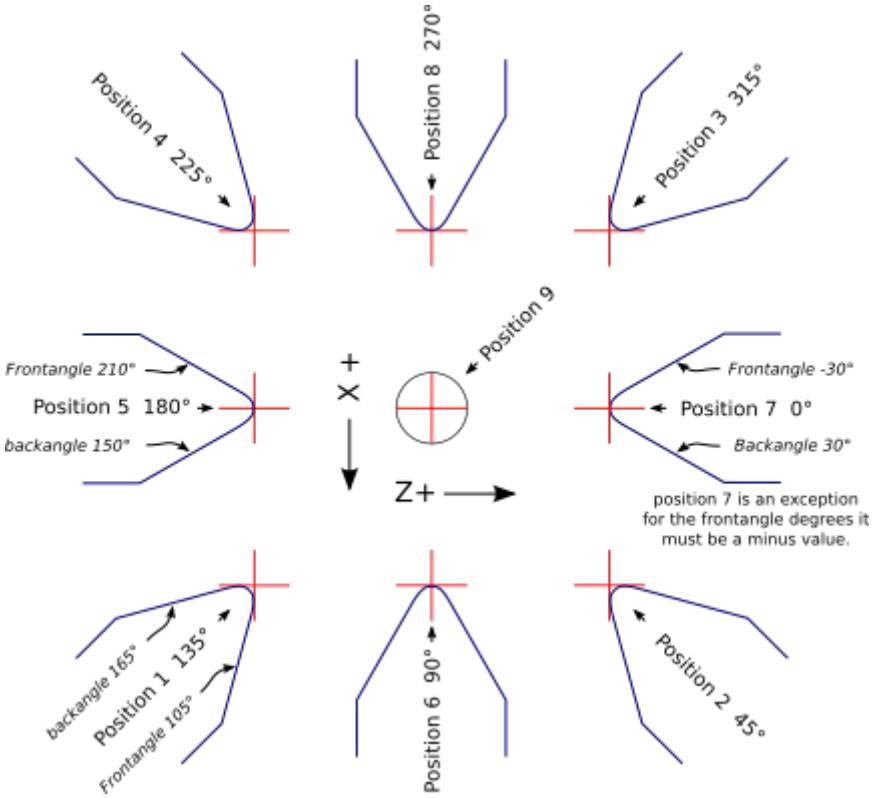
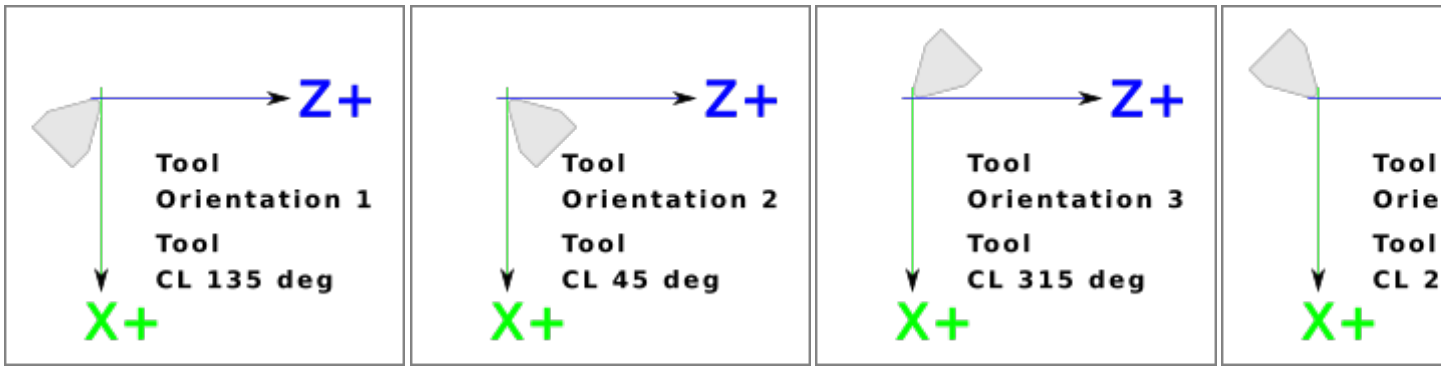


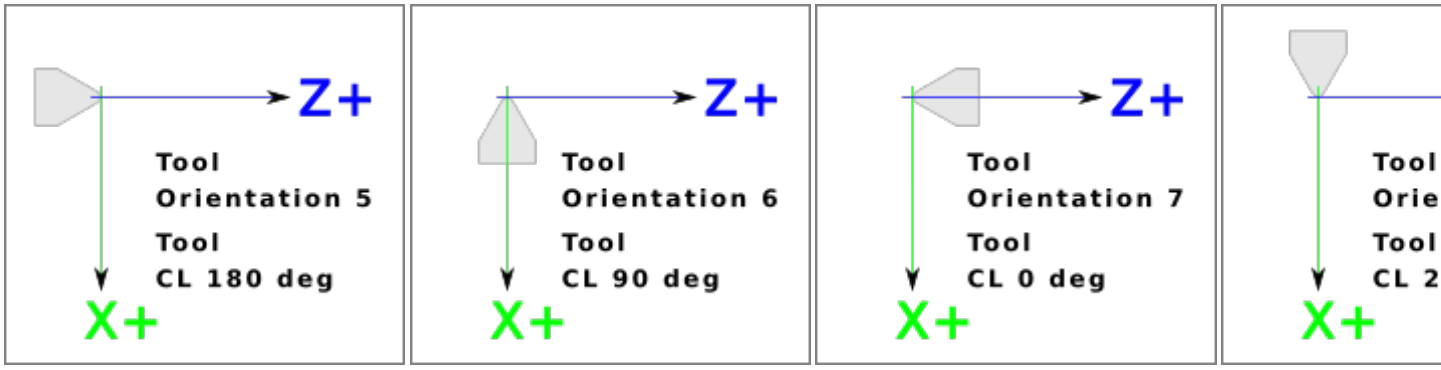
Abbildung 2.13: Ausrichtung des Drehwerkzeugs

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

Werkzeugpositionen 1, 2, 3 & 4Werkzeugpositionen 123 & 4 23 & 4 3 & 4



Werkzeugpositionen 5, 6, 7 & 8Werkzeugpositionen 567 & 8 67 & 8 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, Tn und G43. Für weitere Informationen zu den Optionen für die Werkzeugberührung in AXIS siehe Tool Touch Off.

2.6.4.1 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 *T*n *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.

2.6.4.2 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 *T*n *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.

2.6.4.3 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 *T*n *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.

2.6.6.1 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.

2.6.6.2 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

2.6.7.1 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.

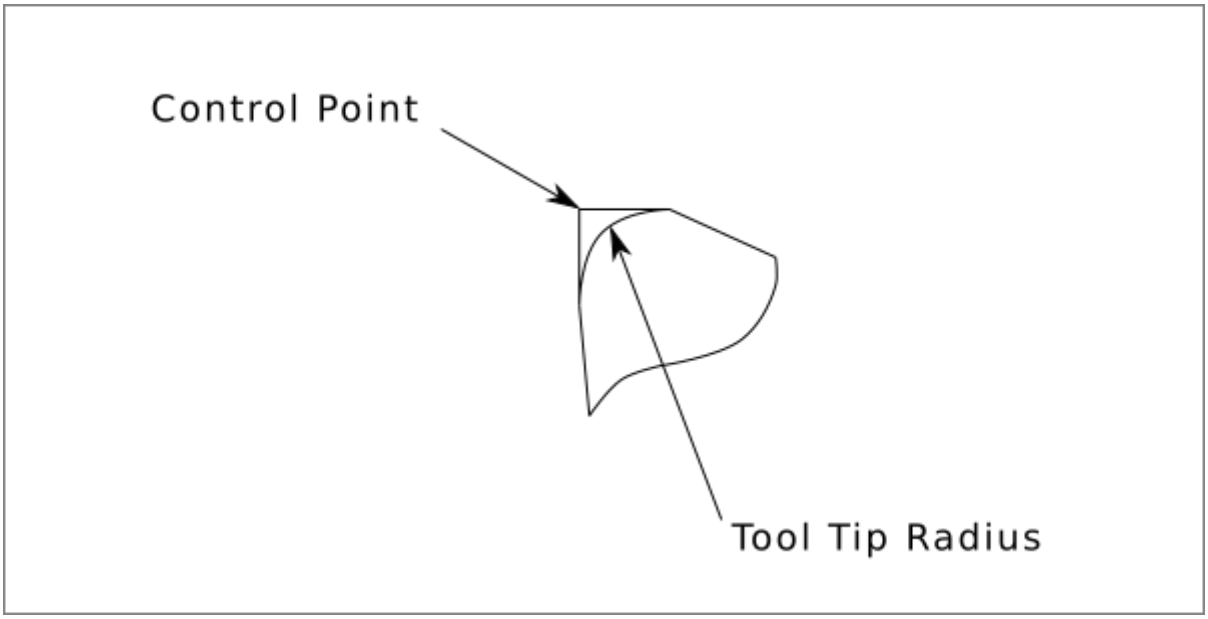


Abbildung 2.14: Kontrollpunkt

2.6.7.2 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.

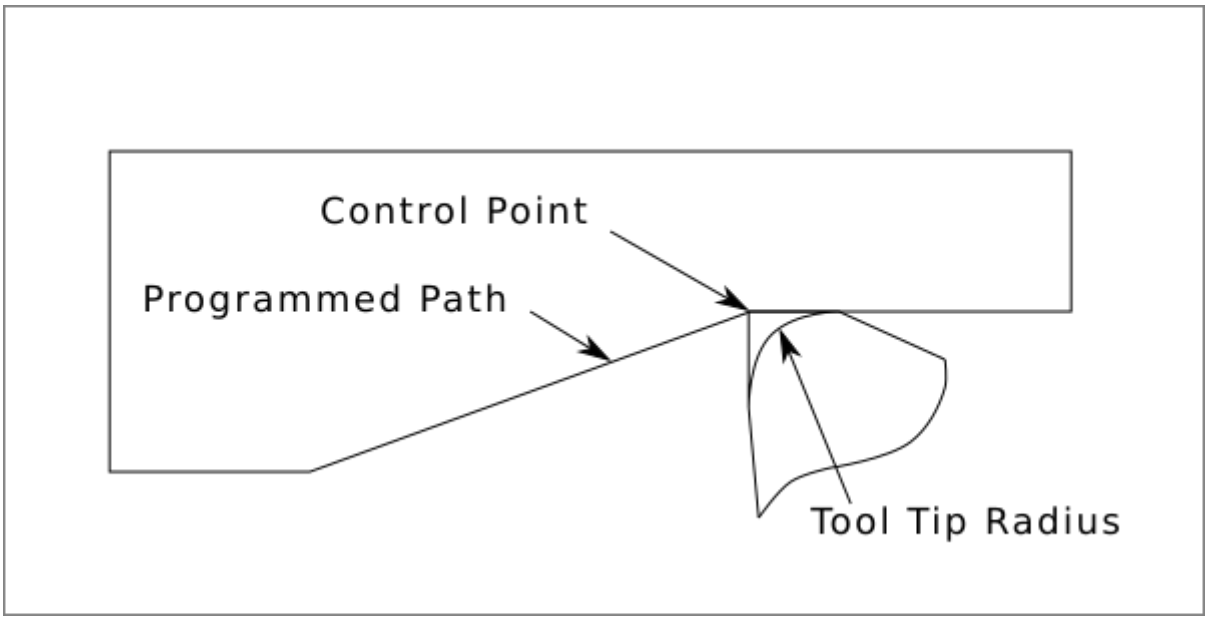


Abbildung 2.15: Rampe Eingang

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.

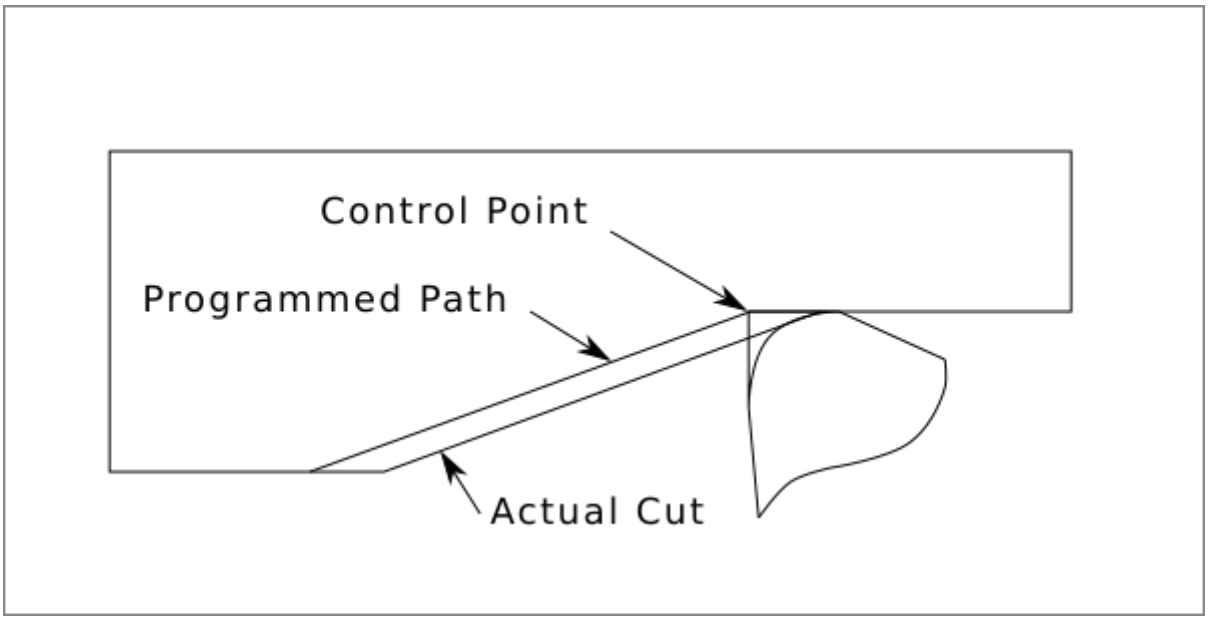


Abbildung 2.16: Rampenpfad

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.

2.6.7.3 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.

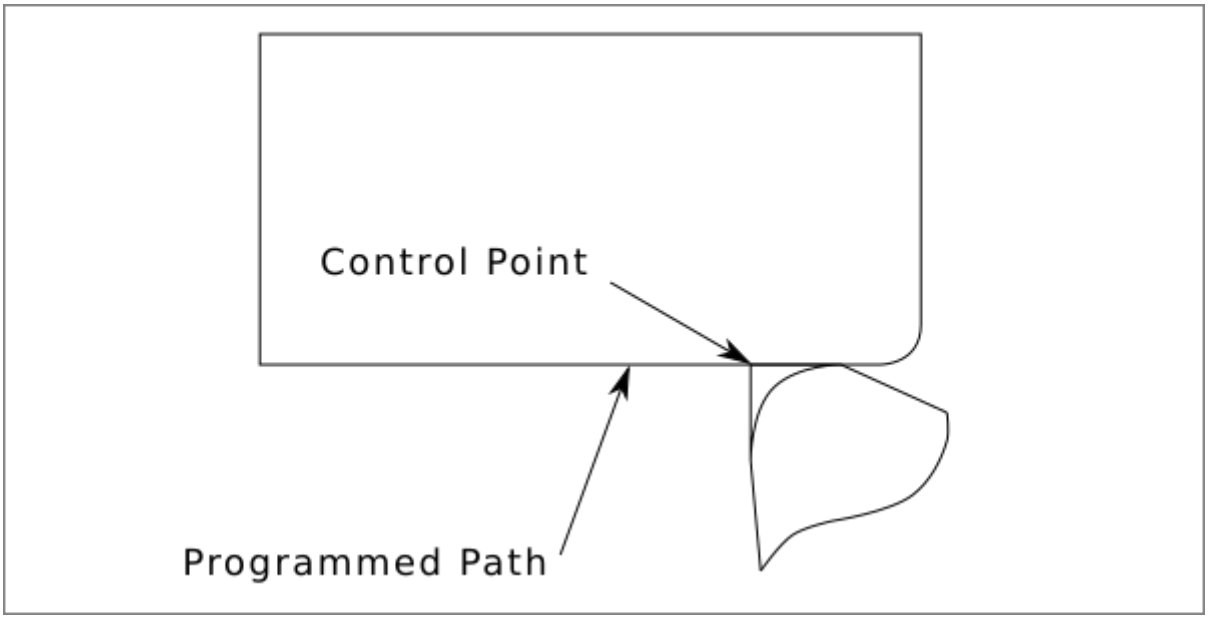


Abbildung 2.17: Turning Cut

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.

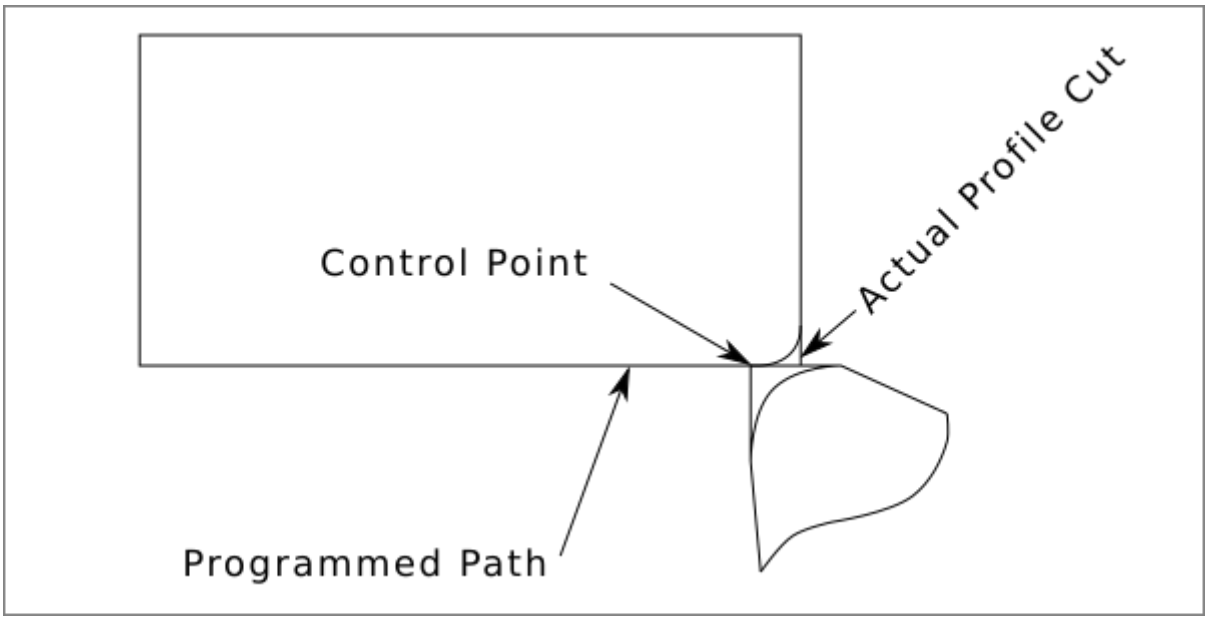


Abbildung 2.18: Radiusschnitt

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

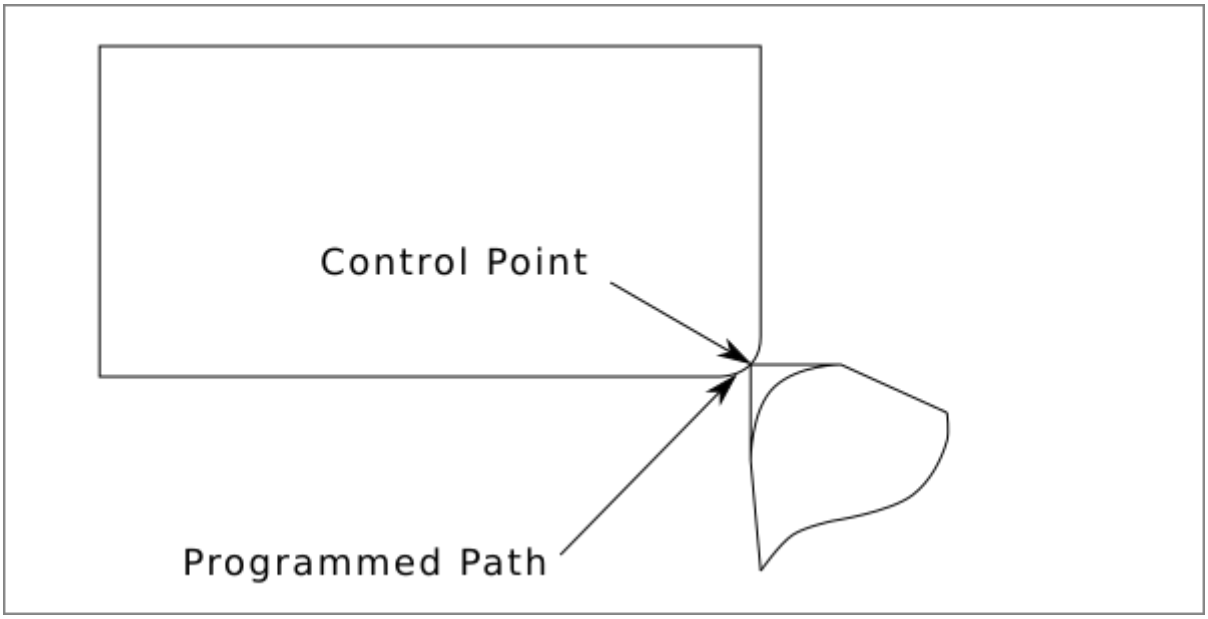


Abbildung 2.19: Radiusschnitt

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.

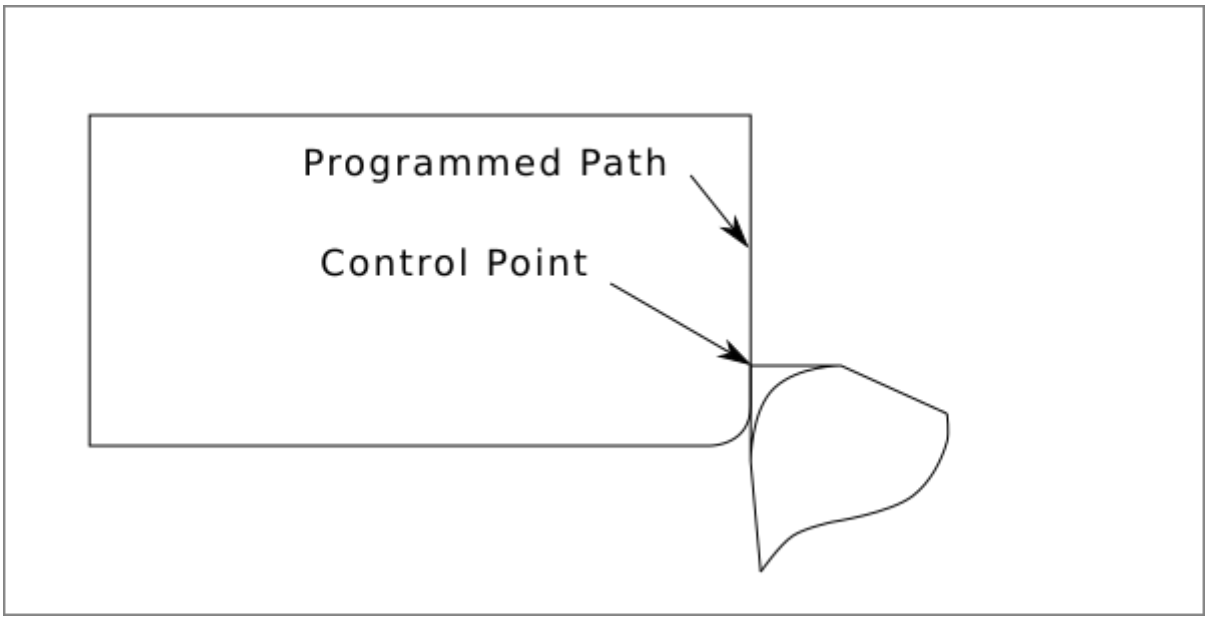


Abbildung 2.20: Face Cut

2.6.7.4 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.

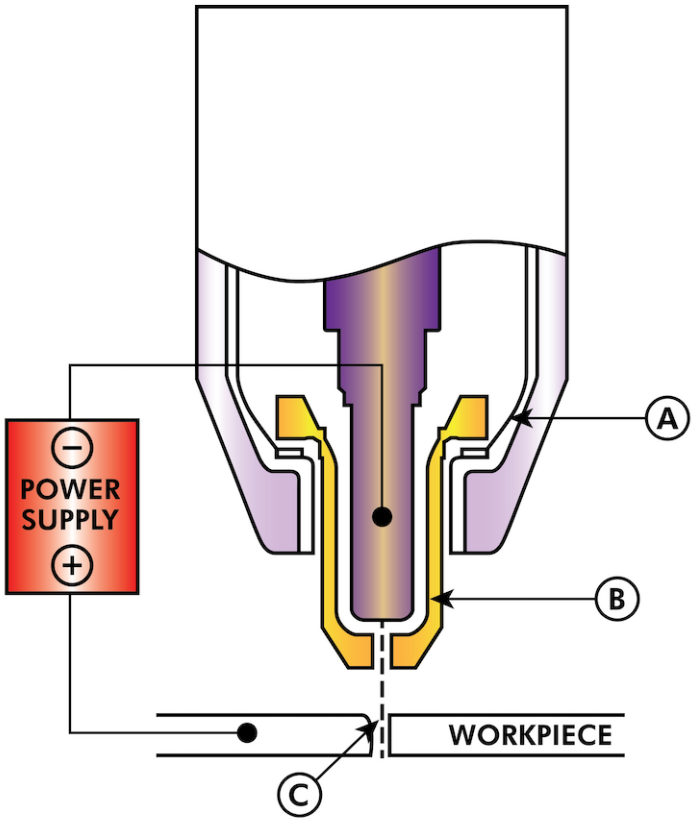
2.7 Plasmaschneiden Primer für LinuxCNC Benutzer

2.7.1 Was ist Plasma?

Plasma is a fourth state of matter, an ionised gas which has been heated to an extremely high temperature and ionised so that it becomes electrically conductive. The plasma arc cutting and gouging processes use this plasma to transfer an electrical arc to the workpiece. The metal to be cut or removed is melted by the heat of the arc and then blown away. While the goal of plasma arc cutting is the separation of the material, plasma arc gouging is used to remove metals to a controlled depth and width.

Plasma torches are similar in design to the automotive spark plug. They consist of negative and positive sections separated by a center insulator. Inside the torch, the pilot arc starts in the gap between the negatively charged electrode and the positively charged tip. Once the pilot arc has ionised the plasma gas, the superheated column of gas flows through the small orifice in the torch tip, which is focused on the metal to be cut.

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…

2.7.2.1 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.

2.7.2.2 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.

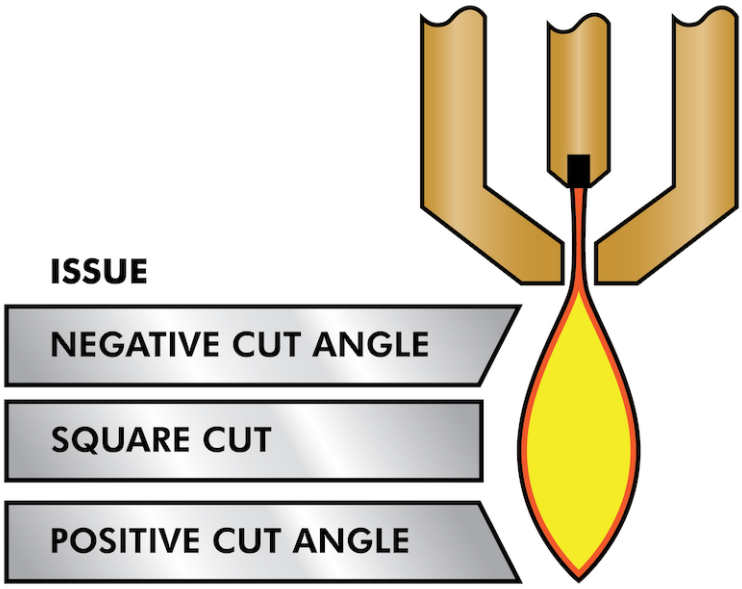
Bei CNC-Luftplasmasystemen der Einstiegsklasse wird der Blowback-Stil bevorzugt, um elektrische Interferenzen mit der Elektronik und Standard-PCs zu minimieren, aber bei größeren Maschinen ab 200 Ampere ist der Hochfrequenzstart immer noch die Regel. Diese erfordern PCs und Elektronik auf Industrieniveau, 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

Plasma operations on CNC machines is quite unique in comparison to milling or turning and is a bit of an orphan process. Uneven heating of the material from the plasma arc will cause the sheet to bend and buckle. Most sheets of metal do not come out of the mill or press in a very even or flat state. Thick sheets (30 mm plus) can be out of plane as much as 50 mm to 100 mm. Most other CNC G-code operations will start from a known reference or a piece of stock that has a known size and shape and the G-code is written to rough the excess off and then finally cut the finished part. With plasma the unknown state of the sheet makes it impossible to generate G-code that will cater for these variances in the material.

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.

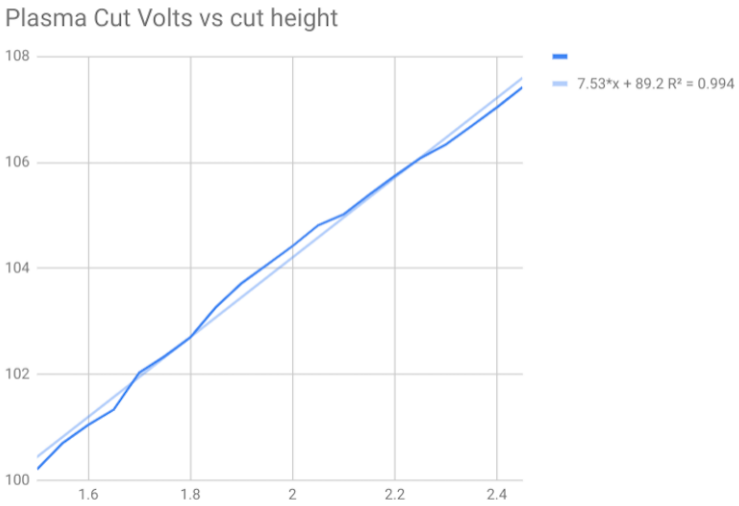
- **Der Abstand zwischen Brenner und Werkstück kann die Kantenfase beeinflussen**.



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

Anmerkung
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.



This graph was prepared from a sample of about 16,000 readings at varying cut height and the regression analysis shows 7.53 V/mm with 99.4% confidence. In this particular instance this sample was taken from an Everlast 50 A machine being controlled by LinuxCNC.

Torch voltage then becomes an ideal process control variable to use to adjust the cut height. Let's assume for simplicity that voltage changes by 10 V/mm. This can be restated to be 1 Volt per 0.1 mm (0.004"). Major plasma machine manufacturers (eg Hypertherm, Thermal Dynamics and ESAB), produce cut charts that specify the recommended cut height and estimated arc voltage at this height as well as some additional data. So if the arc voltage is 1 V higher than the manufacturers specification, the controller simply needs to lower the torch by 0.1 mm (0.004") to move back to the desired cut height. A torch height control unit (THC) is traditionally used to manage this process.

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
- Dry Contacts for ArcOK
- Terminals for Arc On switch
- Raw arc voltage or divided arc voltage output
- Optionally a RS485 interface if using a Hypertherm plasma cutter and want to control it from the LinuxCNC console.
- 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

Most THC units are external devices and many have a fairly crude "bit bang" adjustment method. They provide two signals back to the LinuxCNC controller. One turns on if the Z axis should move up and the other turns on if the Z axis should move down. Neither signal is true if the torch is at the correct height. The popular Proma 150 THC is one example of this type of THC. The LinuxCNC THCUD component is designed to work with this type of THC.

With the release of the Mesa THCAD voltage to frequency interface, LinuxCNC was able to decode the actual torch voltage via an encoder input. This allowed LinuxCNC to control the Z axis and eliminate

external hardware. Early implementations utilising the THCAD replicated the “bit bang” approach. The LinuxCNC THC component is an example of this approach.

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.

The inclusion of external offsets in LinuxCNC V2.8 allowed plasma control in LinuxCNC to rise to a whole new level. External Offsets refers to the ability to apply an offset to the axis commanded position external to the motion controller. This is perfect for plasma THC control as a method to adjust the torch height in real time based on our chosen process control methodology. Following a number of experimental builds, the Plasmac configuration was incorporated into LinuxCNC 2.8. QtPlasmaC has superceded Plasmac in LinuxCNC 2.9. This has been an extremely ambitious project and many people around the globe have been involved in testing and improving the feature set. QtPlasmaC is unique in that its design goal was to support all THCs including the simple bit bang ones through to sophisticated torch voltage control if the voltage is made available to LinuxCNC via a THCAD or some other voltage sensor. What’s more, QtPlasmaC is designed to be a stand alone system that does not need any additional G-code subroutines and allows the user to define their own cut charts that are stored in the system and accessible by a drop-down.

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.

A simple and effective ArcOK signal can be achieved with a simple reed relay. Wrap 3 turns of one of the plasma cutter’s thick cables (eg the material clamp cable) around it. Place the relay in an old pen tube for protection and connect one side of the relay to field power and the other end to your ArcOK input pin.

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, die erreicht werden können; Strommessung zur Erkennung eines erhöhten

Motordrehmoments, eines ”Schwimmer”-Schalters und einer elektrischen oder ”ohmischen” Sensorschaltung, die geschlossen wird, wenn die Brennerabschirmung das Material berührt. Die Stromerfassung ist keine praktikable Technik für DIY-Tische, aber Schwimmerschalter und ohmsche Erfassung werden im Folgenden diskutiert:

2.7.7.1 Gleitende Schalter (engl. float switches)

The torch is mounted on a sliding stage that can move up when the torch tip contacts the material surface and trigger a switch or sensor. Often this is achieved under G-code control using the G38 Commands. If this is the case, then after initial probing, it is recommended to probe away from the surface until the probe signal is lost at a slower speed. Also, ensure the switch hysteresis is accounted for.

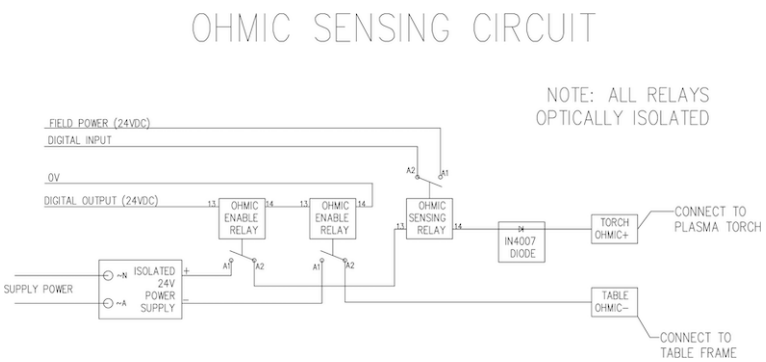
Unabhängig von der verwendeten Sondierungsmethode wird dringend empfohlen, einen gleitenden Schalter einzubauen, damit ein Ausweich- oder Sekundärsignal vorhanden ist, um eine Beschädigung des Brenners bei einem Absturz zu vermeiden.

2.7.7.2 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.

The following is an example circuit that has been proven to work and is compatible with the LinuxCNC QtPlasmaC configuration.



2.7.7.3 Hypersensing with a MESA THCAD-5

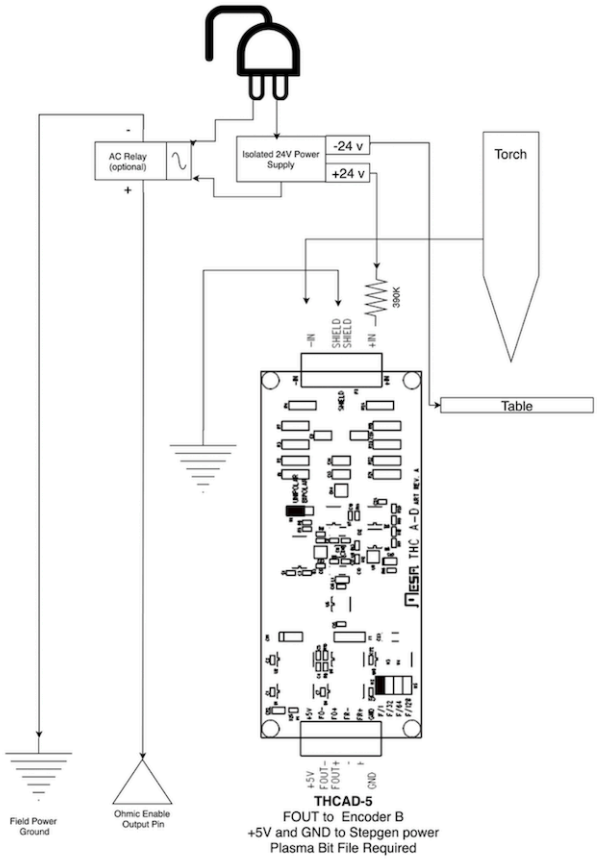
Eine ausgefeiltere Methode der Materialerkennung, bei der die Relais und Dioden entfallen, ist die Verwendung eines weiteren THCAD-5 zur Überwachung der Spannung des Materialerkennungsschaltkreises über eine isolierte Stromversorgung. Dies hat den Vorteil, dass das THCAD für die feindliche elektrische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

The THCAD is sensitive enough to see the ramp up in circuit voltage as contact pressure increases. The ohmic.comp component included in LinuxCNC can monitor the sensing voltage and set a voltage threshold above which it is deemed contact is made and an output is enabled. By monitoring the voltage, a lower “break circuit” threshold can be set to build in strong switch hysteresis. This minimises false triggering. In our testing, we found the material sensing using this method was more sensitive and robust as well as being simpler to implement the wiring. One further advantage is using software outputs instead of physical I/O pins is that it frees up pins to use for other purposes. This advantage is helpful to get the most out of the Mesa 7i96 which has limited I/O pins.

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



We used a 15 W Mean Well HDR-15 Ultra Slim DIN Rail Supply 24 V DIN rail based isolated power supply. This is a double insulated Isolation Class II device that will withstand any arc voltage that might be applied to the terminals.

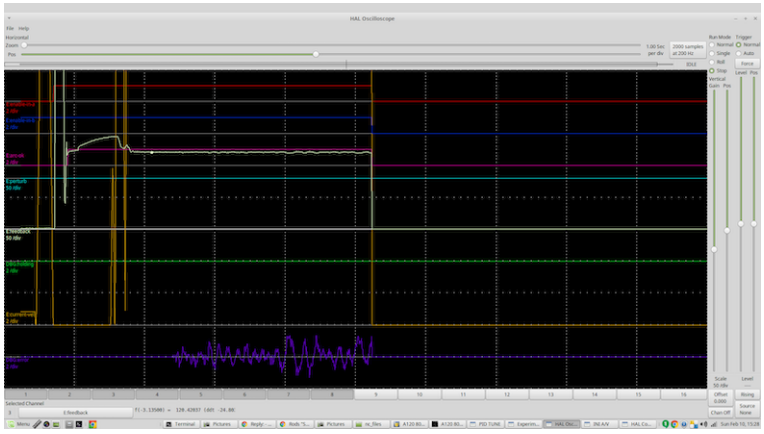
2.7.7.4 Beispiel HAL-Code für Hypersensing

Der folgende HAL-Code kann in die custom.hal von QtPlasmaC eingefügt werden, um die ohmsche Abtastung am Encoder 2 eines 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 debounce.0.2.in
net ohmic-true ohmicsense.ohmic-on => debounce.0.2.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.



It is important for the plasma controller to “wait it out” before auto sampling the torch voltage and commencing THC control. If enabled too early, the voltage will be above the desired cut Volts and the torch will be driven down in an attempt to address a perceived over-height condition.

In our testing this varies between machines and material from 0.5 to 1.5 seconds. Therefore a delay of 1.5 s after a valid ArcOK signal is received before enabling THC control is a safe initial setting. If you want to shorten this for a given material, LinuxCNC's Halscope will allow you to plot the torch voltage and make informed decisions about the shortest safe delay is used.

Anmerkung
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.

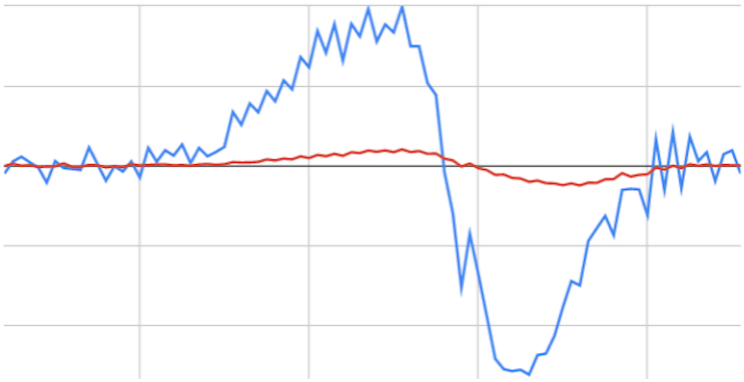
2.7.11 Corner Lock / Velocity Anti-Dive

The LinuxCNC trajectory planner is responsible for translating velocity and acceleration commands into motion that obey the laws of physics. For example, motion will slow when negotiating a corner. Whilst this is not a problem with milling machines or routers, this poses a particular problem for plasma cutting as the arc voltage increases as motion slows. This will cause the THC to drive the torch down. One of the enormous advantages of a THC control embedded within the LinuxCNC motion controller is that it knows what is going on at all times. So it becomes a trivial matter to monitor the current velocity (`motion.current-velocity`) and to hold THC operation if it falls below a set threshold (e.g., 10% below the desired feedrate).

2.7.12 Void / Kerf Crossing

If the plasma torch passes over a void while cutting, arc voltage rapidly rises and the THC responds by violent downward motion which can smash the torch into the material possibly damaging it. This is a situation that is difficult to detect and handle. To a certain extent it can be mitigated by good nesting techniques but can still occur on thicker material when a slug falls away. This is the one problem that has yet to be solved within the LinuxCNC open source movement.

One suggested technique is to monitor the rate of change in torch Volts over time (dv/dt) because this parameter is orders of magnitude higher when crossing a void than what occurs due to normal warpage of the material. The following graph shows a low resolution plot of dv/dt (in blue) while crossing a void. The red curve is a moving average of torch Volts.



So it should be possible to compare the moving average with the dv/dt and halt THC operation once the dv/dt exceeds the normal range expected due to warpage. More work needs to be done in this area to come up with a working solution in LinuxCNC.

2.7.13 Hole And Small Shape Cutting

It is recommended that you slow down cutting when cutting holes and small shapes.

John Moore says: “If you want details on cutting accurate small holes look up the sales sheets on Hypertherm’s ”True Hole Technology” also look on PlasmaSpider, user seanp has posted extensively on his work using simple air plasma.

The generally accepted method to get good holes from 37mm dia. and down to material thickness with minimal taper using an air plasma is:

1. Use recommended cutting current for consumables.
2. Use fixed (no THC) recommended cutting height for consumables.
3. Cut from 60% to 70% of the recommended feed rate of consumables and materials.
4. Start lead in at or near center of hole.
5. Use perpendicular lead in.
6. No lead out, either a slight over burn or early torch off depending on what works best for you.

You will need to experiment to get exact hole size because the kerf with this method will be wider than your usual straight cut.”

This slow down can be achieved by manipulating the feed rate directly in your post processor or by using adaptive feed and an analog pin as input. This lets you use M67/M68 to set the percentage of desired feed to cut at.

- Knowing The Feedrate

From the preceding discussion it is evident that the plasma controller needs to know the feed rate set by the user. This poses a problem with LinuxCNC because the Feedrate is not saved by LinuxCNC after the G-code is buffered and parsed. There are two approaches to work around this:

1. Remap the F command and save the commanded feedrate set in G-code via an M67/M68 command
2. Storing the cut charts in the plasma controller and allow the current feedrate be queried by the G-code program (as QtPlasmaC does)

One experimental LinuxCNC branch that would be useful for plasma cutting was the state tags branch. This adds a “tag” that is available to motion containing the current feed and speed rates for all active motion commands. It has been merged and will be in LinuxCNC v2.9

2.7.14 I/O Pins For Plasma Controllers

Plasma cutters require several additional pins. In LinuxCNC, there are no hard and fast rules about which pin does what. In this discussion we will assume the plasma inverter has a CNC interface and the controller card has active high inputs are in use (e.g., Mesa 7i76e).

Plasma tables can be large machines and we recommend that you take the time to install separate max/min limit switches and homing switches for each joint. The exception might be the Z axis lower limit. When a homing switch is triggered the joint decelerates fairly slowly for maximum accuracy. This means that if you wish to use homing velocities that are commensurate with table size, you can overshoot the initial trigger point by 50-100 mm. If you use a shared home/limit switch, you have to move the sensor off the trigger point with the final HOME_OFFSET or you will trigger a limit switch fault as the machine comes out of homing. This means you could lose 50 mm or more of axis travel with shared home/limit switches. This does not happen if separate home and limit switches are used.

The following pins are usually required (note that suggested connections may not be appropriate for a QtPlasmaC configuration):

2.7.14.1 Arc OK (input)

- Inverter closes dry contacts when a valid arc is established
- Connect Field power to one Inverter ArcOK terminal.
- Connect other Inverter Ok Terminal to input pin.
- Usually connected to one of the `motion.digital-_<nn>_` pins for use from G-code with M66

2.7.14.2 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.
- If a mechanical relay is used, connect a flyback diode (e.g., IN400x series) across the coil terminals with the band on the diode pointing towards the output pin
- 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.
- Usually connected to `spindle.0.on`

Warnung
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;

2.7.14.3 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.
- Usually Connected to `motion.probe-input`

2.7.14.4 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 a non-QtPlasmaC configuration, usually triggered by a `motion.digital-out-_<nn>_` so it can be controlled in G-code by M62/M63/M64/M65

2.7.14.5 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.
- Usually connected to `motion.probe-input` and may be or'd with the float switch.

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 der Mesa 7i76e dem 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.

2.7.14.6 Brenner-Abreißsensor (engl. torch breakaway sensor)

- As mentioned earlier, a breakaway sensor should be installed that is triggered if the torch crashes and falls off.
- Usually, this would be connected to `halui.program-pause` so the fault can be rectified and the program resumed.

2.7.15 G-code For Plasma Controllers

Most plasma controllers offer a method to change settings from G-code. LinuxCNC support this via M67/M68 for analog commands and M62-M65 for digital (on/off commands). How this is implemented is totally arbitrary. Lets look at how the LinuxCNC QtPlasmaC configuration does this:

2.7.15.1 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
```

Anmerkung
Users with a very large number of entries in the QtPlasmaC Materials Table may need to increase the Q parameter (e.g., from Q1 to Q2).

2.7.15.2 Aktivieren/Deaktivieren des THC-Betriebs:

```
M62 P2 will disable THC (synchronised with motion)
M63 P2 will enable THC (synchronised with motion)
M64 P2 will disable THC (immediately)
M65 P2 will enable THC (immediately)
```

2.7.15.3 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
```

2.7.15.4 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
```

Anmerkung
Integrators should familiarise themselves with the LinuxCNC documentation for the various LinuxCNC G-code commands mentioned above.

2.7.16 External Offsets and Plasma Cutting

External Offsets were introduced to LinuxCNC with version 2.8. By external, it means that we can apply an offset external to the G-code that the trajectory planner knows nothing about. It easiest to explain with an example. Picture a lathe with an external offset being applied by a mathematical formula to machine a lobe on a cam. So the lathe is blindly spinning around with the cut diameter set to a fixed diameter and the external offset moves the tool in and out to machine the cam lobe via an applied external offset. To configure our lathe to machine this cam, we need to allocate some portion of the axis velocity and acceleration to external offsets or the tool can’t move. This is where the ini variable OFFSET_AV_RATIO comes in. Say we decide we need to allocate 20% of the velocity

and acceleration to the external offset to the Z axis. We set this equal to 0.2. The consequence of this is that your maximum velocity and acceleration for the Lathe's Z axis is only 80% of what it could be.

External offsets are a very powerful method to make torch height adjustments to the Z axis via a THC. But plasma is all about high velocities and rapid acceleration so it makes no sense to limit these parameters. Fortunately in a plasma machine, the Z axis is either 100% controlled by the THC or it isn't. During the development of LinuxCNC's external offsets it was recognised that Z axis motion by G-code and by THC were mutually exclusive. This allows us to trick external offsets into giving 100 % of velocity and acceleration all of the time. We can do this by doubling the machine's Z axis velocity and acceleration settings in the ini file and set OFFSET_AV_RATIO = 0.5. That way 100% of the maximum velocity and acceleration will be available for both probing and THC.

Example: On a metric machine with a NEMA23 motor with a direct drive to a 5 mm ball screw, 60 mm/s maximum velocity and 700 mm/s^2 acceleration were determined to be safe values without loss of steps. For this machine, set the Z axis in the ini file as follows:

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

The joint associated with this axis would have the velocity and acceleration variables set as follows:

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

For further information about external offsets (for version 2.8 or later) please read the [AXIS_<letter>] Section of the INI file document and External Axis Offsets in the LinuxCNC documentation.

2.7.17 Reading Arc Voltage With The Mesa THCAD

The Mesa THCAD board is a remarkably well priced and accurate voltage to frequency converter that is designed for the hostile noisy electrical environment associated with plasma cutting. Internally it has a 0-10 V range. This range can be simply extended by the addition of some resistors as described in the documentation. This board is available in three versions, the newer THCAD-5 with a 0-5 V range, the THCAD-10 with a 0-10 Volt range and the THCAD-300 which is pre-calibrated for a 300 Volt extended range. Each board is individually calibrated and a sticker is applied to the board that states the frequency at 0 Volts and full scale. For use with LinuxCNC, it is recommended that the 1/32 divisor be selected by the appropriate link on the board. In this case, be sure to also divide the stated frequencies by 32. This is more appropriate for the 1 kHz servo thread and also allows more time for the THCAD to average and smooth the output.

There is a lot of confusion around how to decode the THCAD output so lets consider the Mesa 7i76e and the THCAD-10 for a moment with the following hypothetical calibration data:

- Full scale 928,000 Hz (1/32 29,000 Hz)
- 0 volt 121,600 Hz (1/32 3,800 Hz)

Because the full scale is 10 Volts, then the frequency per Volt is:

(29,000 Hz - 3,800 Hz) / 10 V = 2,520 Hz per Volt

So assuming we have a 5 Volt input, the calculated frequency would be:

(2520 Hz/V * 5 V) + 3,800 Hz = 16,400 Hz

So now it should be fairly clear how to convert the frequency to its voltage equivalent:

Volts = (frequency (Hz) - 3,800 Hz) / 2,520 Hz/V

2.7.17.1 THCAD Connections

On the high voltage side:

- Connect the divided or raw arc voltage to I_N+ and I_N-
- Connect the interconnect cable shield to the Shield connection.
- Connect the other Shield terminal to frame ground.

Assuming it is connected to a Mesa 7I76E, connect the output to the spindle encoder input:

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

2.7.17.2 THCAD Initial Testing

Make sure you have the following lines in your ini file (assuming a Mesa 7I76E):

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

Power up your controller and open Halshow (AXIS: Show Homing Configuration), drill down to find the `hm2_7i76e.0.encoder.00.velocity pin`. With 0 Volts applied, it should be hovering around the 0 Volt frequency (3,800 in our example). Grab a 9 Volt battery and connect it to I_N+ and I_N-. For a THCAD-10 you can now calculate the expected velocity (26,480 in our hypothetical example). If you pass this test, then you are ready to configure your LinuxCNC plasma controller.

2.7.17.3 Which Model THCAD To Use

The THCAD-5 is useful if you intend to use it for ohmic sensing. There is no doubt the THCAD-10 is the more flexible device and it is easy to alter the scaling. However, there is one caveat that can come into play with some cheaper plasma cutters with an inbuilt voltage divider. That is, the internal resistors may be sensed by the THCAD as being part of its own external resistance and return erroneous results. For example, the 16:1 divider on the Everlast plasma cutters needs to be treated as 24:1 (and 50:1 becomes 75:1). This is not a problem with more reputable brands (e.g., Thermal Dynamics, Hypertherm, ESAB etc). So if you are seeing lower than expected cutting voltages, it might be preferable to reconfigure the THCAD to read raw arc voltage.

Remembering that plasma arc voltages are potentially lethal, here are some suggested criteria.

Pilot Arc Start Because there is not likely to be any significant EMI, you should be able to safely install the THCAD in your control panel if you have followed our construction guidelines.

- If you do not have a voltage divider, either install scaling resistors inside the plasma cutter and install the THCAD in the control panel or follow the suggestions for HF start machines.
- If you have a voltage divider, install a THCAD-10 in your control panel. We've had no problems with this configuration with a 120 A Thermal Dynamics plasma cutter.

HF Start Install the THCAD at the inverter as the frequency signal is far more immune to EMI noise.

- If you do not have a voltage divider and you have room inside the plasma cutter, install a THCAD-300 inside the plasma cutter.
- If you do not have a voltage divider and you do not have room inside the plasma cutter, install a THCAD-10 in a metal case outside the plasma cutter and install 50% of the scaling resistance on each of the I_N+ and I_N- inside the plasma cutter case so no lethal voltages come out of the case.
- If you have a voltage divider, install a THCAD-10 in a metal case outside the plasma cutter

Raw Arc voltage presented on a connector In this case, regardless of the arc starting method, there are probably already resistors included in the circuitry to avoid lethal shocks so a THCAD-10 is advised so this resistance (typically 200 kΩ) can be accounted for when choosing a scaling resistor as these resistors will distort the voltage reported by the THCAD-300.

2.7.18 Post Processors And Nesting

Plasma is no different to other CNC operations in that it is:

1. Designed in CAD (where it is output as a DXF or sometimes SVG format).
2. Processed in CAM to generate final G-code that is loaded to the machine
3. Cutting the parts via CNC G-code commands.

Some people achieve good results with Inkscape and G-code tools but SheetCam is a very well priced solution and there are a number of post processors available for LinuxCNC. SheetCam has a number of advanced features designed for plasma cutting and for the price, is a no brainer for anybody doing regular plasma cutting.

2.7.19 Designing For Noisy Electrical Environments

Plasma cutting is inherently an extremely hostile and noisy electrical environment. If you have EMI problems things won't work correctly. You might fire the torch and the computer will reboot in a more obvious example, but you can have any number of other odd symptoms. They will pretty much all happen only when the torch is cutting - often when it is first fired.

Therefore, system builders should select components carefully and design from the ground up to cope with this hostile environment to avoid the impact of Electro-Magnetic Interference (EMI). Failure to do this could result in countless hours of fruitless troubleshooting.

Choosing ethernet boards such as the Mesa 7I76E or the cheaper 7I96 helps by allowing the PC to be located away from the electronics and the plasma machine. This hardware also allows the use of 24 Volt logic systems which are much more noise tolerant. Components should be mounted in a metal enclosure connected to the mains earth. It is strongly recommended that an EMI filter is installed on the mains power connection. The simplest way is to use a EMI filtered mains power IEC connector commonly used on PC's and electric appliances which allows this to be achieved with no extra work. Plan the layout of components in the enclosure so that mains power, high voltage motor wires and logic signals are kept as separate as possible from each other. If they do have to cross, keep them at 90 degrees.

Peter Wallace from Mesa Electronics suggests: "If you have a CNC compatible plasma source with a voltage divider, I would mount the THCAD inside your electronics enclosure with all the other motion hardware. If you have a manual plasma source and you are reading raw plasma voltage, I would mount the THCAD as close to the plasma source as possible (even inside the plasma source case if it fits.) In this case, make sure that all low side THCAD connections are fully isolated from the plasma source. If you use a shielded box for the THCAD, the shield should connect to your electronic enclosure ground, not the plasma source ground."

It is recommended to run a separate earth wire from motor cases and the torch back to a central star grounding point on the machine. Connect the plasma ground lead to this point and optionally an earth rod driven into the ground as close as possible to the machine (particularly if its a HF start plasma machine).

External wiring to motors should be shielded and appropriately sized to handle the current passing through the circuit. The shield should be left unconnected at the motor end and earthed at the control box end. Consider using an additional pin on any connectors into the control box so the earth can be extended through into the control box and earthed to the chassis right at the stepper/servo motor controller itself.

We are aware of at least one commercial system builder who has had problems with induced electrical noise on the ohmic sensing circuit. Whilst this can be mitigated by using ferrite beads and coiling the cable, adding a feed through power line filter is also recommended where the ohmic sensing signal enters the electronics enclosure.

Tommy Berisha, the master of building plasma machines on a budget says: “If on a budget, consider using old laptop power bricks. They are very good, filtering is good, completely isolated, current limited (this becomes very important when something goes wrong), and fitting 2 or 3 of them in series is easy as they are isolated (be aware that some do have the grounding wired to the negative output terminal, so. It has to be disconnected, simply done by using a power cable with no ground contacts)”.

2.7.20 Water Tables

The minimum water level under the cut level of the torch should be around 40 mm, having space under slats is nice so the water can level and escape during cutting, having a bit of water above the metal plate being cut is really nice as it gets rid of the little bit of dust, running it submerged is the best way but not preferable for systems with part time use as it will corrode the torch. Adding baking soda to the water will keep the table in a nice condition for many years as it does not allow corrosion while the slats are under water and it also reduces the smell of water vapour. Some people use a water reservoir with a compressed air inlet so they can push the water from the reservoir up to the water table on demand and thus allow changes in water levels.

2.7.21 Downdraft Tables

Many commercial tables utilise a down draft design so fans are used to suck air down through the slats to capture fumes and sparks. Often tables are zoned so only a section below the torch is opened to the outgoing vent, often using air rams and air solenoids to open shutters. Triggering these zones is relatively straightforward if you use the axis or joint position from one of the motion pins and the lincurve component to map downdraft zones to the correct output pin.

2.7.22 Designing For Speed And Acceleration

In plasma cutting, speed and acceleration are king. The higher the acceleration, the less the machine needs to slow down when negotiating corners. This implies that the gantry should be as light as possible without sacrificing torsional stiffness. A 100 mm x 100 mm x 2 mm aluminium box section has equivalent torsional stiffness to an 80 mm x 80 mm T slot extrusion yet is 62% lighter. So does the convenience of T slots outweigh the additional construction?

2.7.23 Distance Travelled Per Motor Revolution

Stepper motors suffer from resonance and a direct drive pinion is likely to mean that the motor is operating under unfavourable conditions. Ideally, for plasma machines a distance of around 15-25 mm per motor revolution is considered ideal but even around 30 mm per revolutions is still acceptable. A 5 mm pitch ball screw with a 3:1 or 5:1 reduction drive is ideal for the Z axis.

2.7.24 QtPlasmaC LinuxCNC Plasma Configuration

The QtPlasmaC which is comprised of a HAL component (plasmac.hal) plus a complete configurations for the QtPlasmaC GUI has received considerable input from many in the LinuxCNC Open Source movement that have advanced the understanding of plasma controllers since about 2015. There has been much testing and development work in getting QtPlasmaC to its current working state. Everything from circuit design to G-code control and configuration has been included. Additionally, QtPlasmaC supports external THC's such as the Proma 150 but really comes into its own when paired with a Mesa controller as this allows the integrator to include the Mesa THCAD voltage to frequency converter which is purpose built to deal with the hostile plasma environment.

QtPlasmaC is designed to stand alone and includes the ability to include your cutting charts yet also includes features to be used with a post processor like SheetCam.

The QtPlasmaC system is now included in Version 2.9 and above of LinuxCNC. Its now quite mature and has been significantly enhanced since the first version of this guide was written. QtPlasmaC will define LinuxCNC's plasma support for many years to come as it includes all of the features a proprietary high end plasma control system at an open source price.

2.7.25 Hypertherm RS485 Control

Some Hypertherm plasma cutters have a RS485 interface to allow the controller (e.g., LinuxCNC) to set amps.pressure and mode. A number of people have used a userspace component written in Python to achieve this. More recently, QtPlasmaC now supports this interface natively. Refer to the QtPlasmaC documentation for how to use it.

The combination of a slow baud rate used by Hypertherm and the userspace component, make this fairly slow to alter machine states so it generally not viable to change settings on the fly while cutting.

When selecting a RS485 interface to use at the PC end, users have reported that USB to RS485 interfaces are not reliable. Good reliable results have been achieved using a hardware based RS232 interface (eg PCI/PCIe or motherboard port) and an appropriate RS485 converter. Some users have reported success with a Sunix P/N: SER5037A PCI RS2322 card a generic XC4136 RS232 to RS485 converter (which may sometimes include a USB cable as well).

2.7.26 Post Processors For Plasma Cutting

CAM programs (Computer Aided Manufacture) are the bridge between CAD (Computer Aided Design) and the final CNC (Computer Numerical Control) operation. They often include a user configurable post processor to define the code that is generated for a specific machine or dialect of G-code.

Many LinuxCNC users are perfectly happy with using Inkscape to convert .SVG vector based files to G-code. If you are using a plasma for hobby or home use, consider this option. However, if your needs are more complex, probably the best and most reasonably priced solution is SheetCam. SheetCam supports both Windows and Linux and post processors are available for it including the QtPlasmaC configuration. SheetCam allows you to nest parts over a full sheet of material and allows you to configure toolsets and code snippets to suit your needs. SheetCam post processors are text files written in the Lua programming language and are generally easy to modify to suit your exact requirements. For further information, consult the SheetCam web site and their support forum.

Another popular post-processor is included with the popular Fusion360 package but the included post-processors will need some customisation.

LinuxCNC is a CNC application and discussions of CAM techniques other than this introductory discussion are out of scope of LinuxCNC.

Kapitel 3

Konfigurationsassistenten

3.1 Schrittmotor-Konfigurations-Assistent

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

Abbildung 3.1: 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

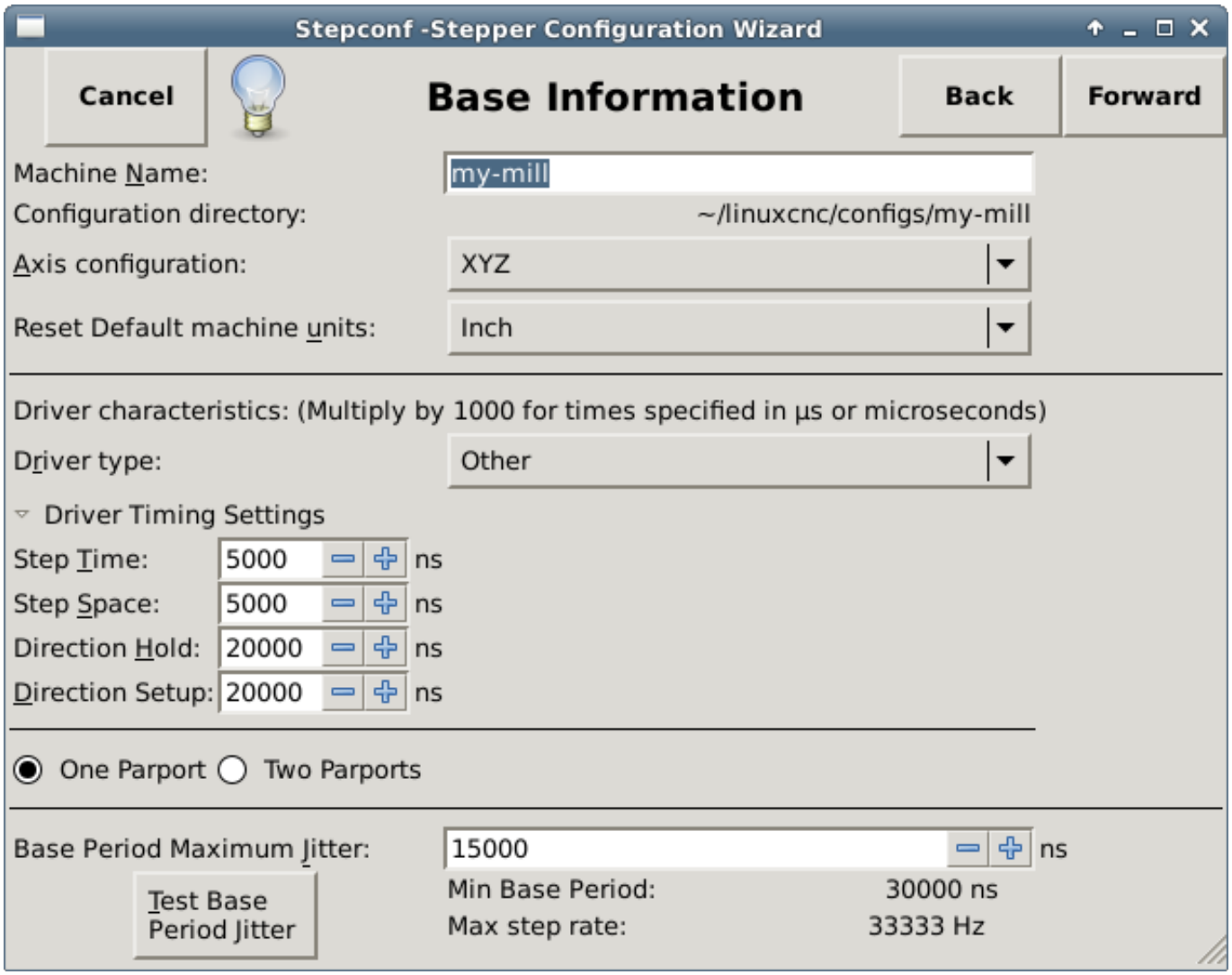


Abbildung 3.2: Seite mit grundlegenden Informationen

- *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 <sec:latency-test,Latenz-Test>>. * *Max Step Rate* - StepConf berechnet die Max Step Rate automatisch auf der Grundlage der eingegebenen Treibermerkmale und des Latenz-Test-Ergebnisses. * *Min Base Period* - StepConf ermittelt die Min Base Period automatisch auf der Grundlage der eingegebenen Treibermerkmale und des Latenz-Test-Ergebnisses.

3.1.4 Latenz-Test

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. Führen Sie ein OpenGL-Programm wie glxgears aus. Die Idee ist, den PC auf Herz und Nieren zu prüfen, während der Latenz-Test ermittelt, was die schlimmsten Werte sind. Lassen Sie den Test mindestens ein paar Minuten laufen. Je länger Sie den Test laufen lassen, desto eher werden auch seltene Ereignisse erfasst, die möglicherweise in kürzeren Abständen auftreten. Dies ist ein Test nur für Ihren Computer, es muss also keine Hardware angeschlossen sein, um den Test durchzuführen.

Warnung
Versuchen Sie nicht, LinuxCNC zu starten, während der Latenztest läuft.

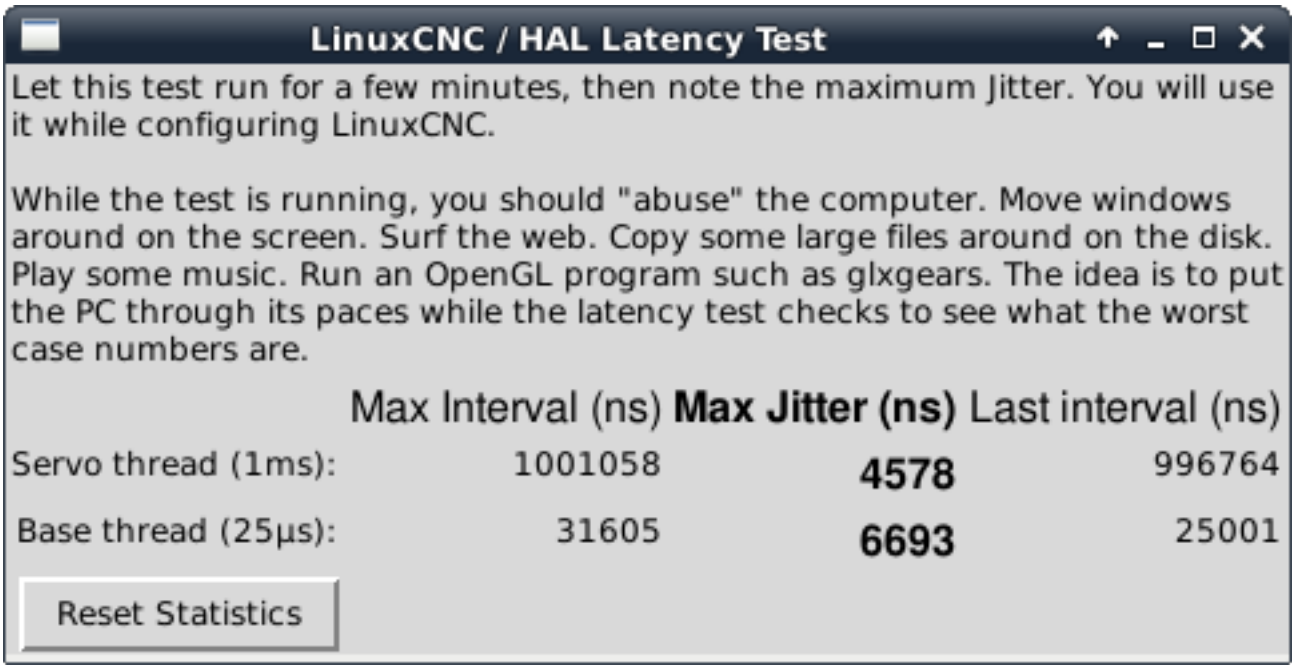


Abbildung 3.3: Latenz-Test

Die Latenzzeit gibt an, wie lange der PC braucht, um seine Arbeit zu unterbrechen und auf eine externe Anfrage zu reagieren. In unserem Fall ist die Anfrage der periodische *Herzschlag* (engl. heartbeat), der als Zeitreferenz für die Schrittimpulse dient. Je geringer die Latenzzeit ist, desto schneller kann der Herzschlag ausgeführt werden, und desto schneller und gleichmäßiger werden die Schrittimpulse sein.

Die Latenzzeit ist weitaus wichtiger als die CPU-Geschwindigkeit. Die CPU ist nicht der einzige Faktor, der die Latenzzeit bestimmt. Motherboards, Grafikkarten, USB-Anschlüsse, SMI-Probleme und eine Reihe anderer Faktoren können die Latenz beeinträchtigen.

Fehlersuche bei SMI-Problemen (LinuxCNC.org Wiki)

Behebung von durch SMI verursachten Echtzeitproblemen unter Ubuntu

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?FixingSMIIssues

The important numbers are the *max jitter*. In the example above 9075 nanoseconds (ns), or 9.075 microseconds (µs), is the highest jitter. Record this number, and enter it in the Base Period Maximum Jitter box.

If your Max Jitter number is less than about 15-20 µs (15000-20000 ns), the computer should give very nice results with software stepping. If the max latency is more like 30-50 µs, you can still get good results, but your maximum step rate might be a little disappointing, especially if you use microstepping or have very fine pitch leadscrews. If the numbers are 100 µs or more (100,000 ns), then the PC is not a good candidate for software stepping. Numbers over 1 millisecond (1,000,000 ns) mean the PC is not a good candidate for LinuxCNC, regardless of whether you use software stepping or not.

3.1.5 Einrichtung der parallelen Schnittstelle

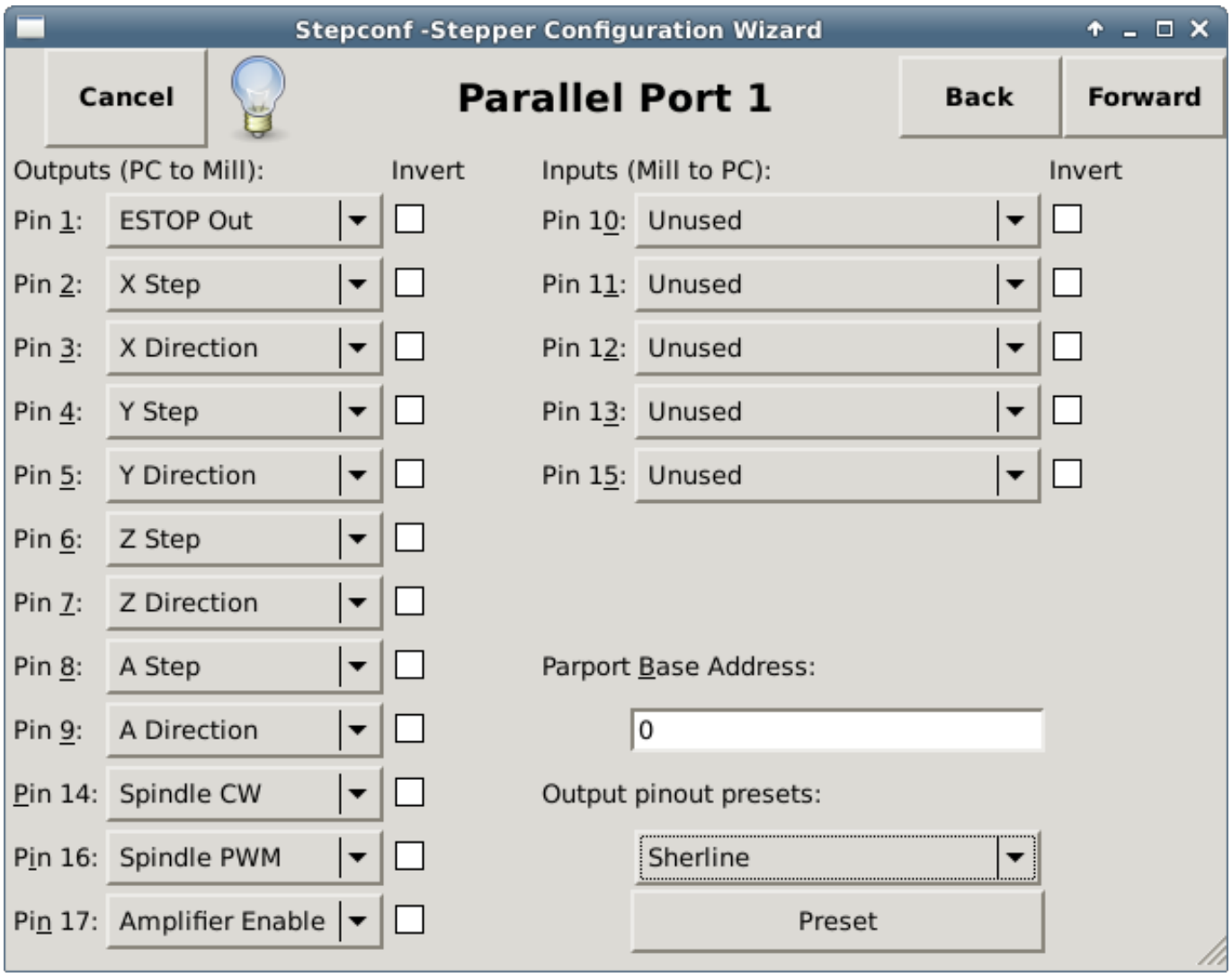


Abbildung 3.4: Parallele Schnittstelle Setup-Seite

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.

3.1.6 Einrichtung des zweiten parallelen Ports

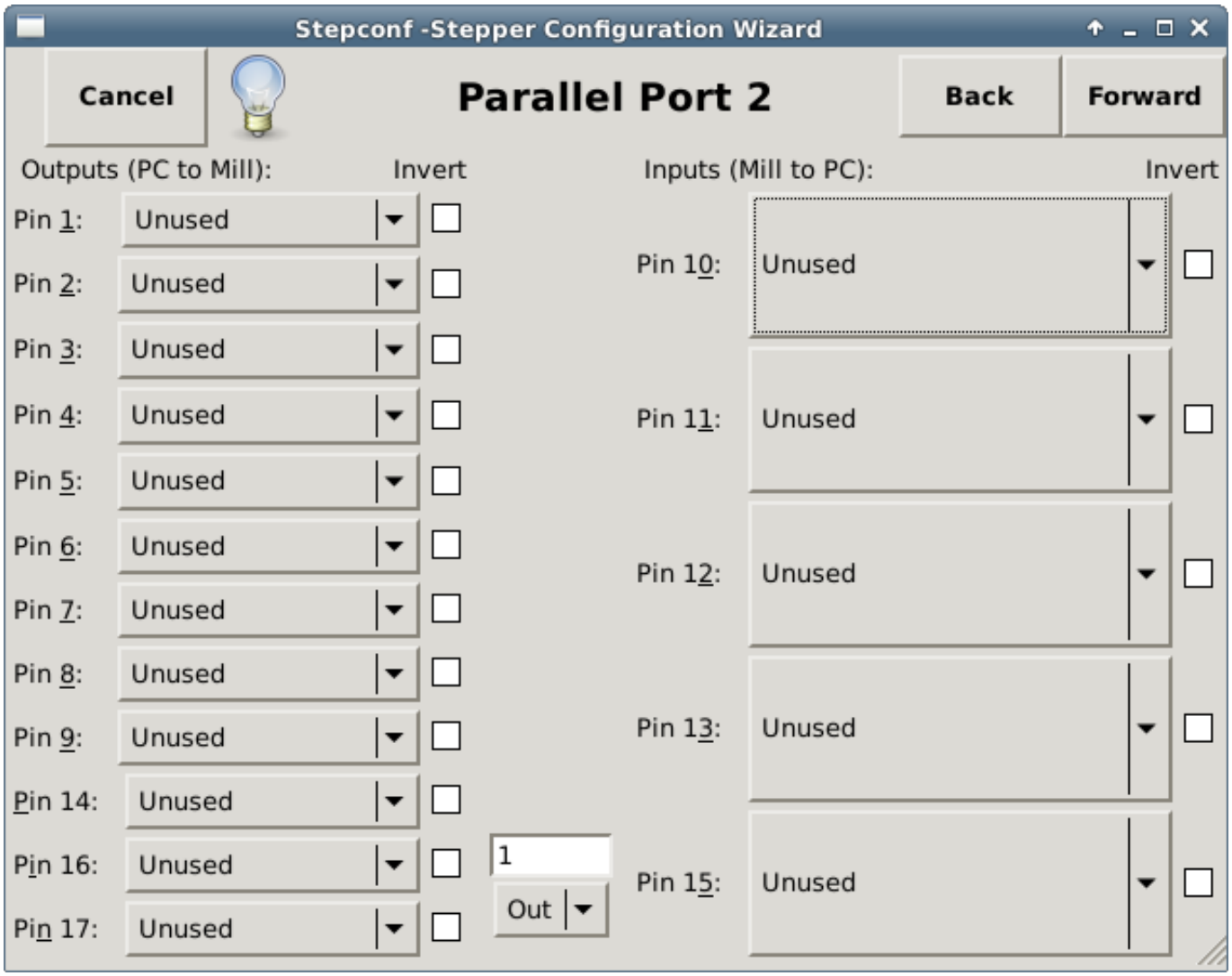


Abbildung 3.5: Einrichten von Parallel Port 2

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.

3.1.7 Achsenkonfiguration

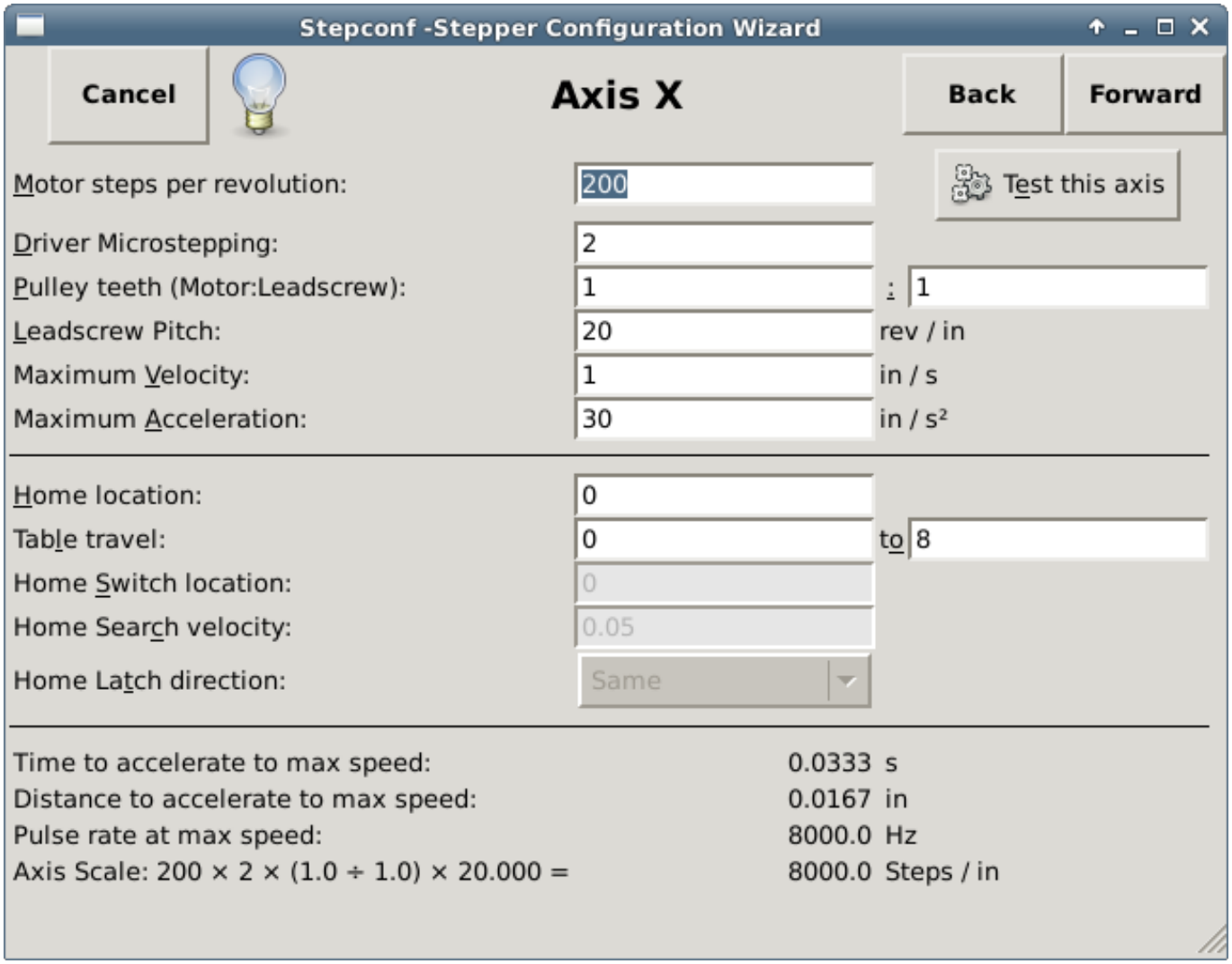


Abbildung 3.6: Achsenkonfiguration am Bildschirm

- *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. 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 <sub:finding-maximum-velocity,Maximale Geschwindigkeit bestimmen>> zur Einstellung der Geschwindigkeit und Maximale Beschleunigung bestimmen zur Einstellung der Beschleunigung.
- *Referenzunkt* - 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. Die Ausgangsposition muss innerhalb des *Tabellenverfahrwegs* 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.

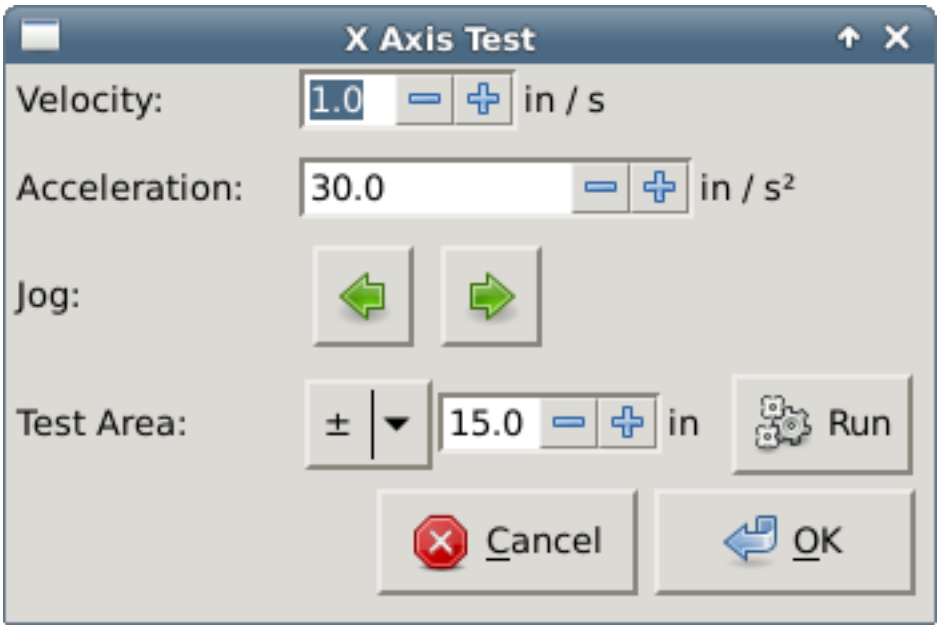


Abbildung 3.7: Achsen-Test

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

Wichtig
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.

3.1.7.1 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.

3.1.7.2 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.

3.1.8 Spindel-Konfiguration

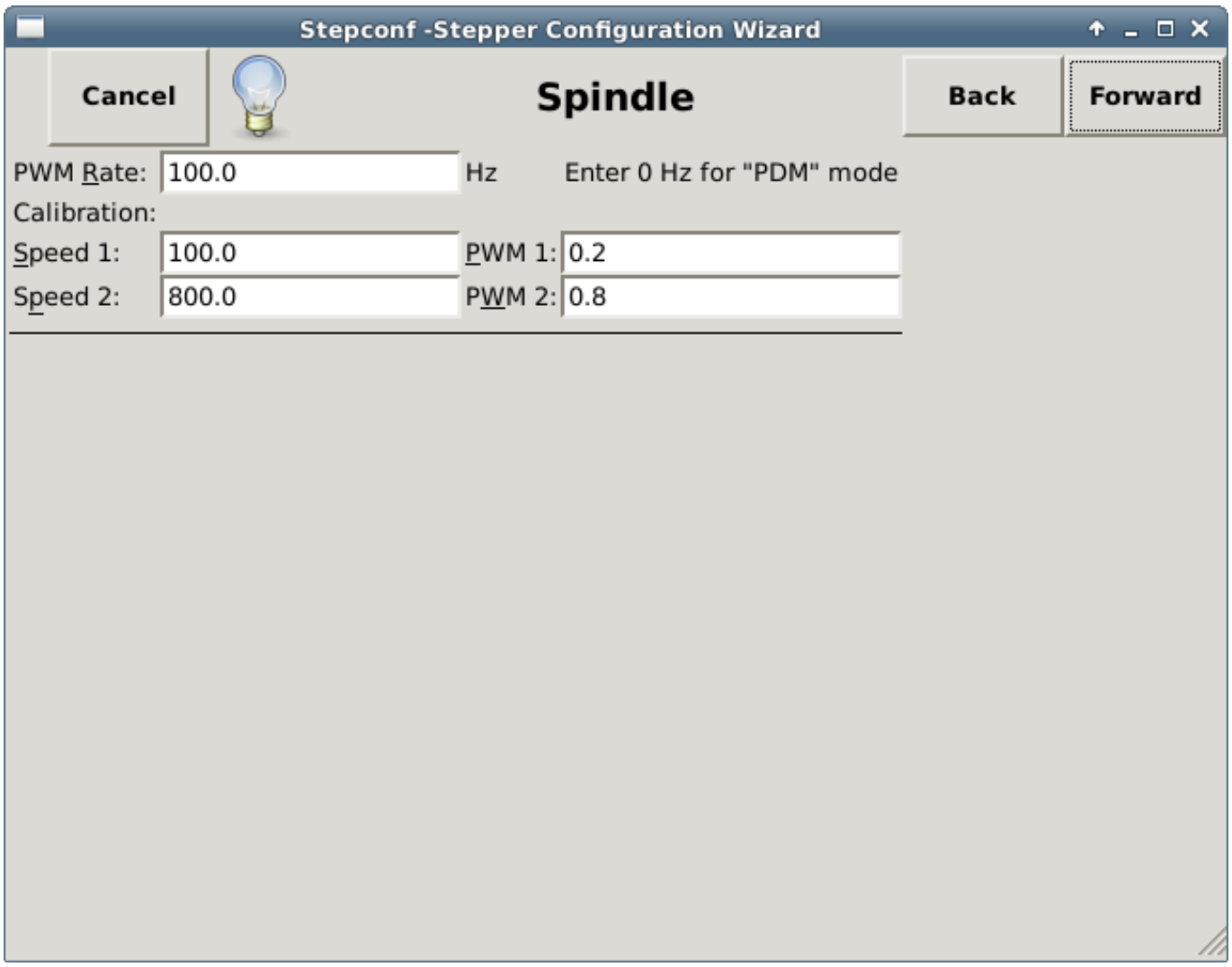


Abbildung 3.8: Seite zur Spindelkonfiguration

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

3.1.8.1 Spindeldrehzahlregelung

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.

3.1.8.2 Spindel-synchronisierte 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.

3.1.8.3 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.

3.1.9 Optionen

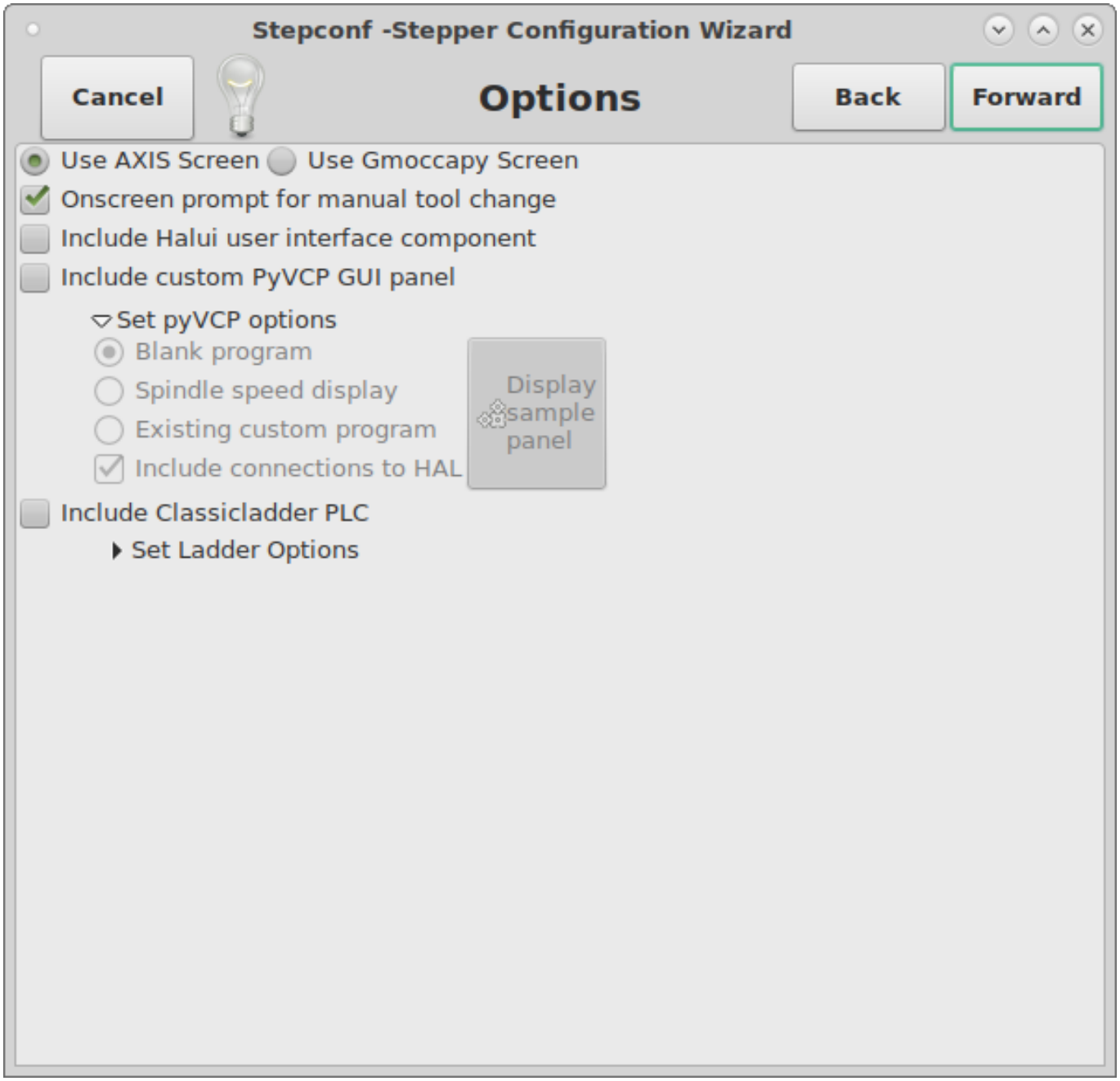


Abbildung 3.9: Erweiterte Optionen bei der Konfiguration

- *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 <cha:classicladder,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.

3.1.10 Vollständige Maschinenkonfiguration

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

3.1.11 Achsen Verfahrwege und Referenzpunkte

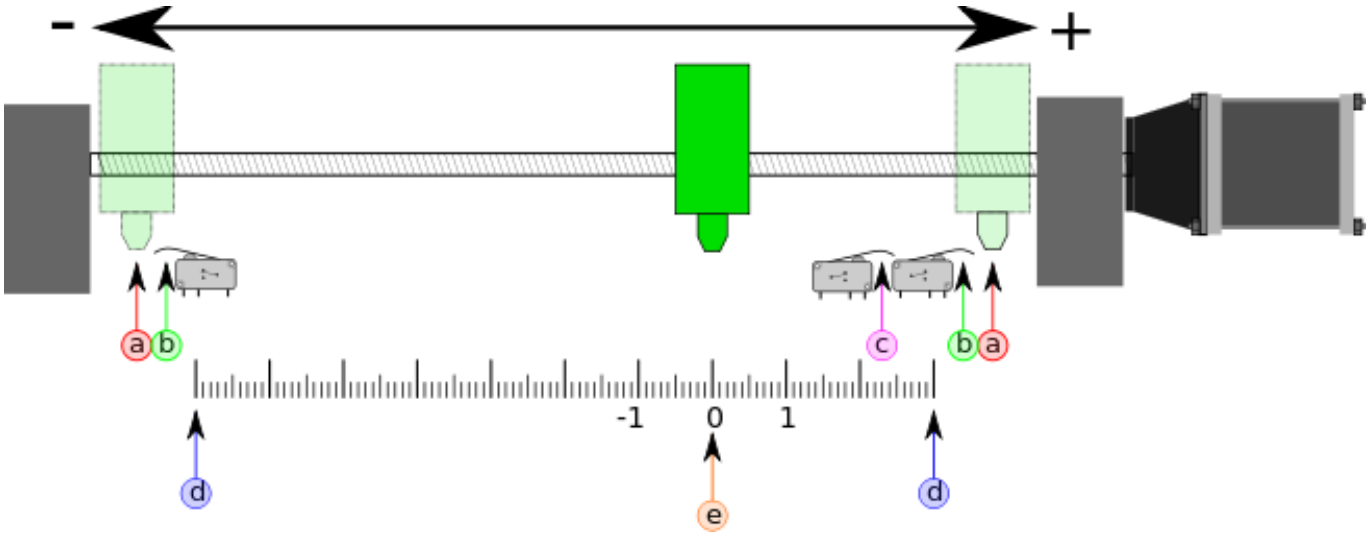


Abbildung 3.10: Achsen Verfahrwege und Referenzpunkt

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

Warnung
[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.

3.1.11.1 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.

3.1.11.2 Betrieb ohne Referenzschalter (engl. home switches)

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.

3.1.11.3 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.

The figures below show the general idea of wiring multiple switches to a single input pin. In each case, when one switch is actuated, the value seen on INPUT goes from logic HIGH to LOW. However, LinuxCNC expects a TRUE value when a switch is closed, so the corresponding *Invert* box must be checked on the pinout configuration page. The pull up resistor show in the diagrams pulls the input high until the connection to ground is made and then the input goes low. Otherwise the input might float between on and off when the circuit is open. Typically for a parallel port you might use 47 kΩ;.

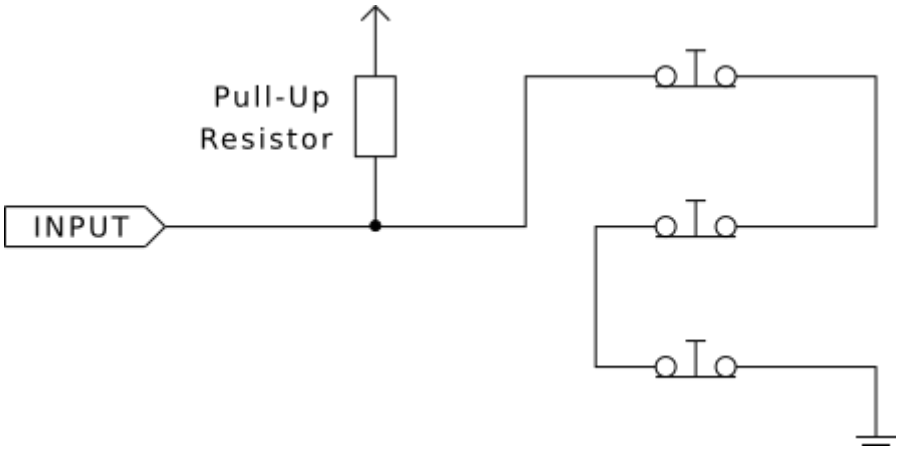


Abbildung 3.11: Normalerweise geschlossene Schalter (engl. normally closed, N/C) in Reihe verdrahtet (vereinfachtes Diagramm)

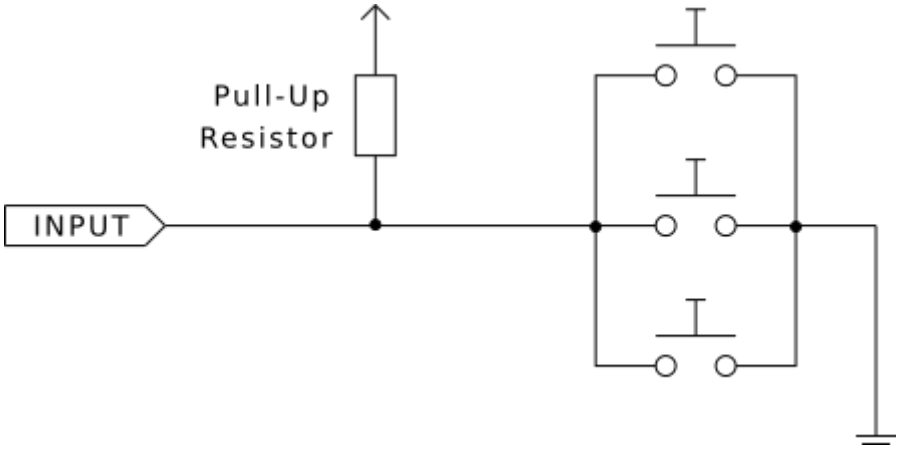


Abbildung 3.12: Normalerweise offene Schalter (engl. normally open switches, N/O) parallel verdrahtet (vereinfachte Darstellung)

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.

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

3.2.1 Schritt für Schritt Anleitung

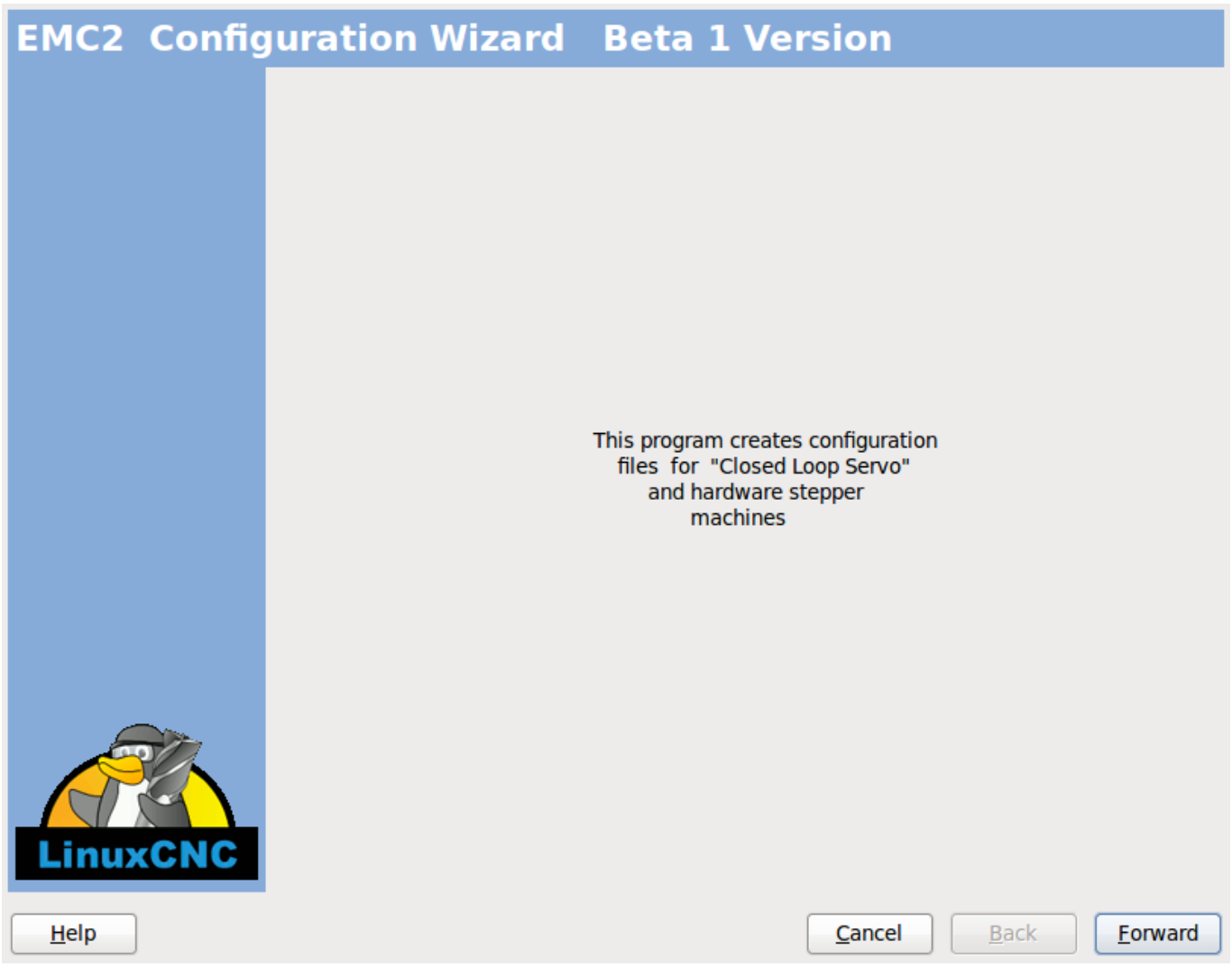


Abbildung 3.13: PnCconf Startfenster

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

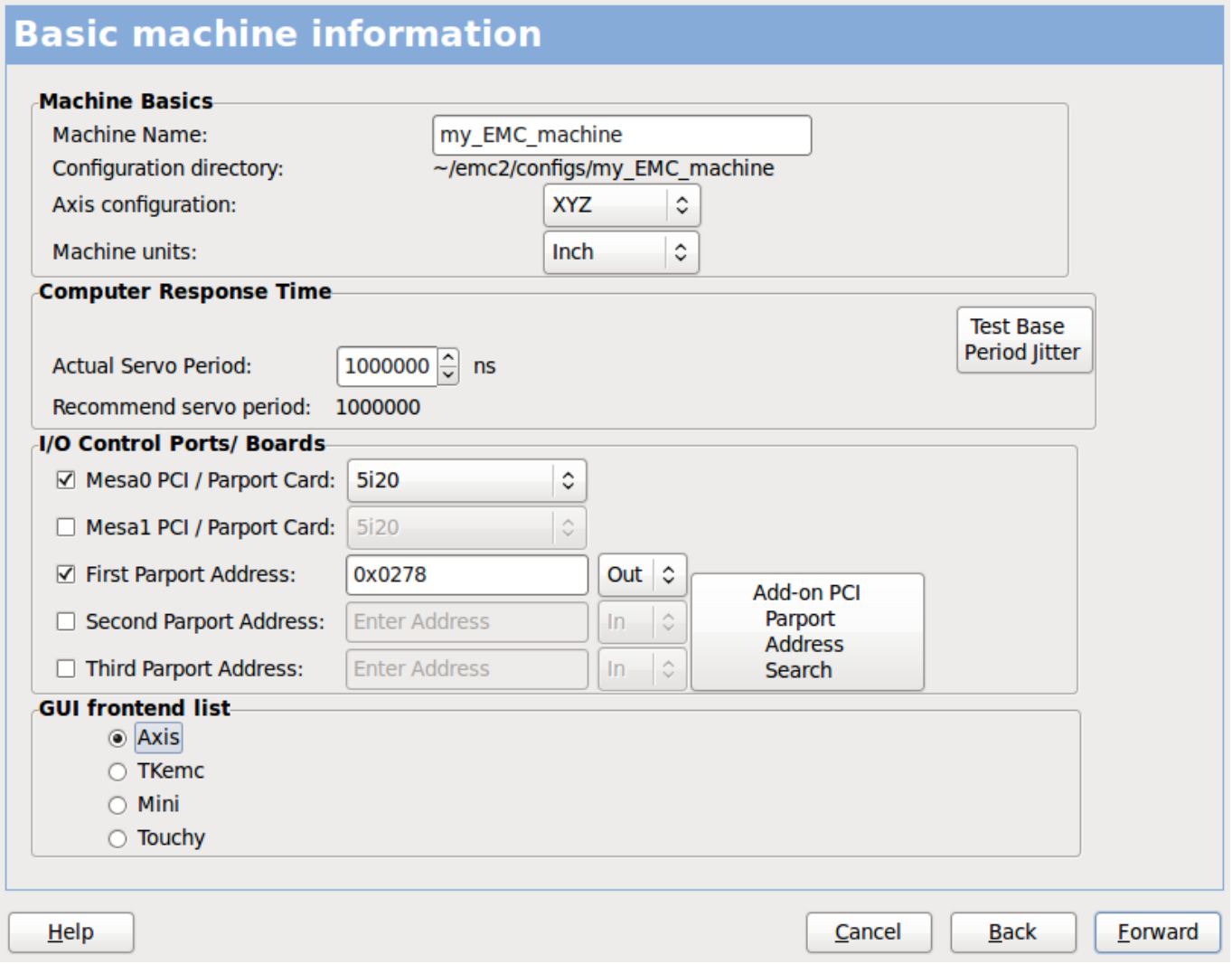


Abbildung 3.14: PnCconf Basic

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) Wählen Sie eine Achsenkonfiguration - damit wählen Sie aus, welche Art von Maschine Sie bauen und welche Achsen verfügbar sind. Die Auswahl der Maschineneinheiten ermöglicht die Eingabe von metrischen oder imperialen Einheiten auf den folgenden Seiten.

Tipp

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

Reaktionszeit des Computers

Die Servoperiode gibt den Herzschlag des Systems vor. Latenz bezieht sich auf die Menge der

Zeit, die der Computer länger als diese Periode sein kann. Genau wie eine Eisenbahn, LinuxCNC erfordert alles auf eine sehr enge und konsistente Zeitlinie oder schlechte Dinge passieren. LinuxCNC erfordert und verwendet ein *Echtzeit* Betriebssystem, das nur bedeutet, dass es hat eine niedrige Latenz (LReaktionszeit) hat. Bbei der Durchführung von LinuxCNCs Berechnungen können diese nicht durch Anforderungen niedrigerer Priorität (wie Benutzereingaben auf dem Bildschirm Tasten oder Zeichnung usw.) unterbrochen werden.

Das Testen der Latenzzeit ist sehr wichtig und ein wichtiger Punkt, den man frühzeitig überprüfen sollte. Glücklicherweise können wir der Mesa-Karte die Arbeit übertragen, für die eine besonders schnelle Reaktionszeit erforderlich ist (Encoder Zählen und PWM-Generierung). Somit können wir viel mehr Latenz ertragen, als wenn wir die parallele Schnittstelle für diese Dinge nutzen. Der Standard-Test in LinuxCNC ist die Überprüfung der BASE-Perioden Latenz (auch wenn wir nicht mit einer Basis-Periode arbeiten). Wenn Sie die *test base period jitter* Taste drücken, startet dies das Latenz-Test-Fenster (Sie können auch laden Sie diese direkt aus der Anwendungen / CNC-Panel.) Der Test soll ein paar Minuten laufen, aber je länger, desto besser. 15 Minuten sind das Minimum, und über Nacht ist noch besser. In dieser Zeit sollten Sie den Computer nutzen, um Dinge zu laden, das Netz zu nutzen, USB zu verwenden usw. Wir möchten die schlimmste Latenzzeit ermitteln und herausfinden, ob eine bestimmte Aktivität die Latenzzeit beeinträchtigt. Wir müssen uns den Jitter der Basisperiode ansehen. Alles, was unter 20000 liegt, ist hervorragend - man könnte mit der Maschine sogar schnelles Software-Stepping machen. 20000 - 50000 ist immer noch gut für Software-Stepping und für uns in Ordnung. 50000 - 100000 ist wirklich nicht so toll, könnte aber immer noch mit Hardware-Karten verwendet werden, die schnelle Reaktionszeiten ermöglichen. Alles unter 100000 ist also für uns brauchbar. Wenn die Latenzzeit enttäuschend ist oder Sie regelmäßig Schluckauf haben, können Sie sie vielleicht noch verbessern.

Tipp
Es ist ein Benutzer kompiliert Liste der Ausrüstung und die Latenz auf der LinuxCNC Wiki erhalten: http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Latency-Test Bitte beachten Sie Ihre Informationen auf der Liste hinzufügen. Auch auf dieser Seite sind Links zu Informationen über die Behebung einiger Latenzprobleme.

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
Sie müssen mindestens eine Mesa-Karte auswählen, da PnCconf die parallelen Ports nicht für die Zählung von Messgeräten oder die Ausgabe von Schritt- oder PWM-Signalen konfiguriert. Die in der Auswahlbox verfügbaren Mesa-Karten basieren darauf, was PnCconf an Firmware auf den Systemen findet. Es besteht die Möglichkeit, benutzerdefinierte Firmware hinzuzufügen und/oder bestimmte Firmware oder Karten mithilfe einer Einstellungsdatei auf eine schwarze Liste zu setzen (zu ignorieren). Wenn keine Firmware gefunden wird, zeigt PnCconf eine Warnung an und verwendet eine interne Beispiel-Firmware - ein Testen ist nicht möglich. Wenn Sie zwei PCI-Mesa-Karten auswählen, gibt es derzeit keine Möglichkeit vorherzusagen, welche Karte 0

und welche 1 ist - Sie müssen testen - das Verschieben der Karten könnte die Reihenfolge ändern. Wenn Sie mit zwei Karten konfigurieren, müssen beide Karten installiert sein, damit die Tests funktionieren.

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.

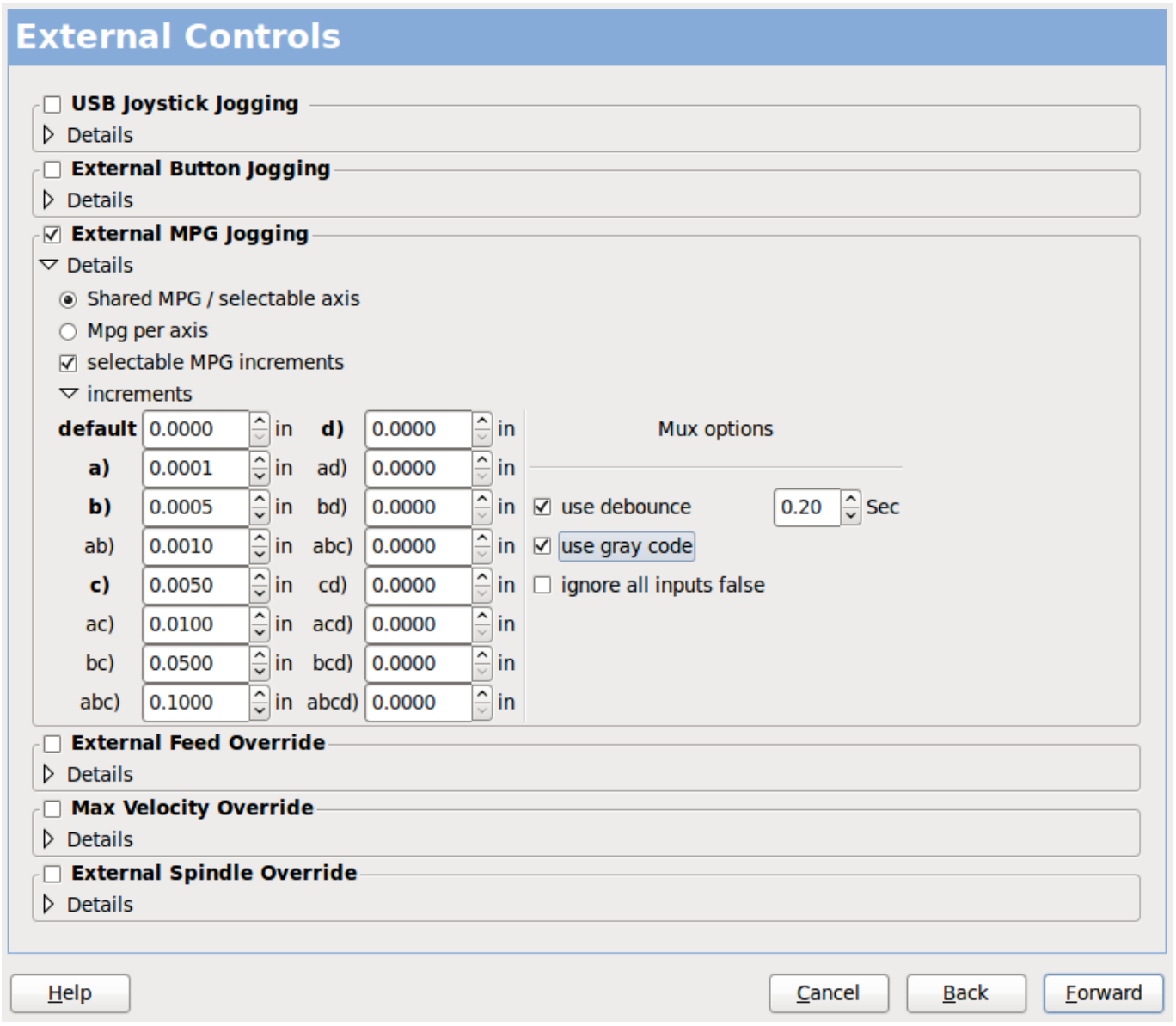


Abbildung 3.15: Externe Steuerelemente

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.

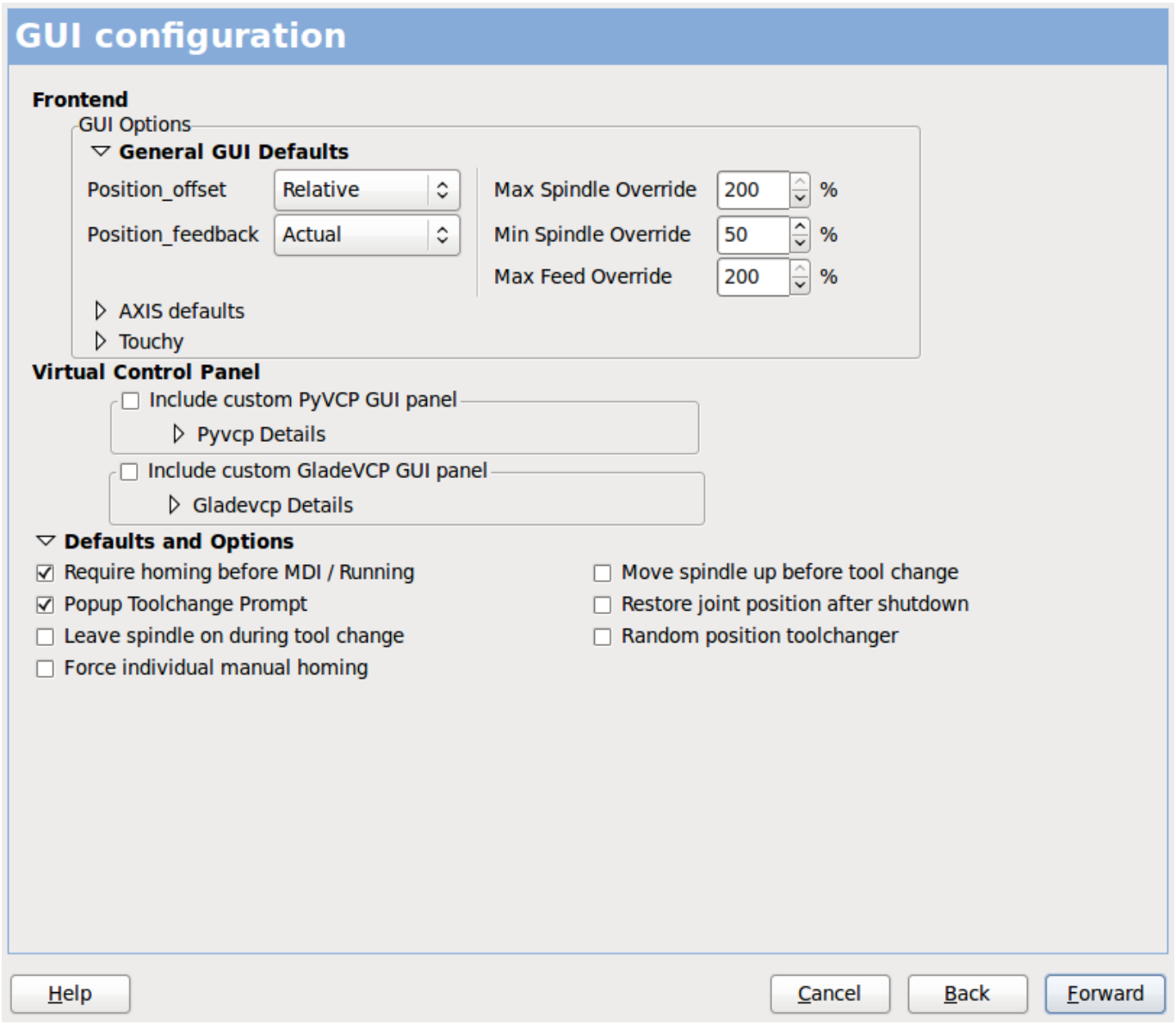


Abbildung 3.16: GUI-Konfiguration

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:

http://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.

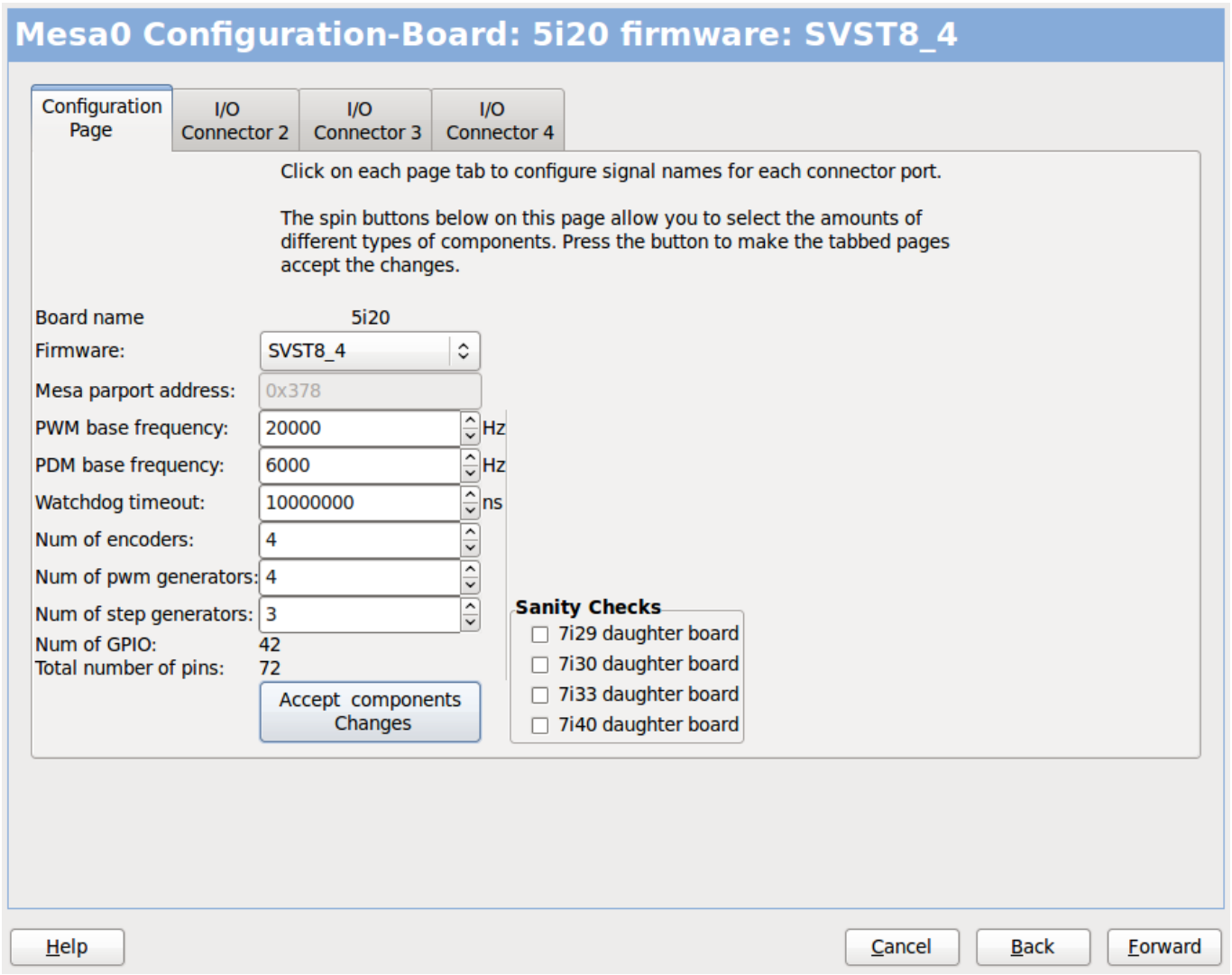


Abbildung 3.17: Mesa Board Konfiguration

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.

Anmerkung
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

Wichtig
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.

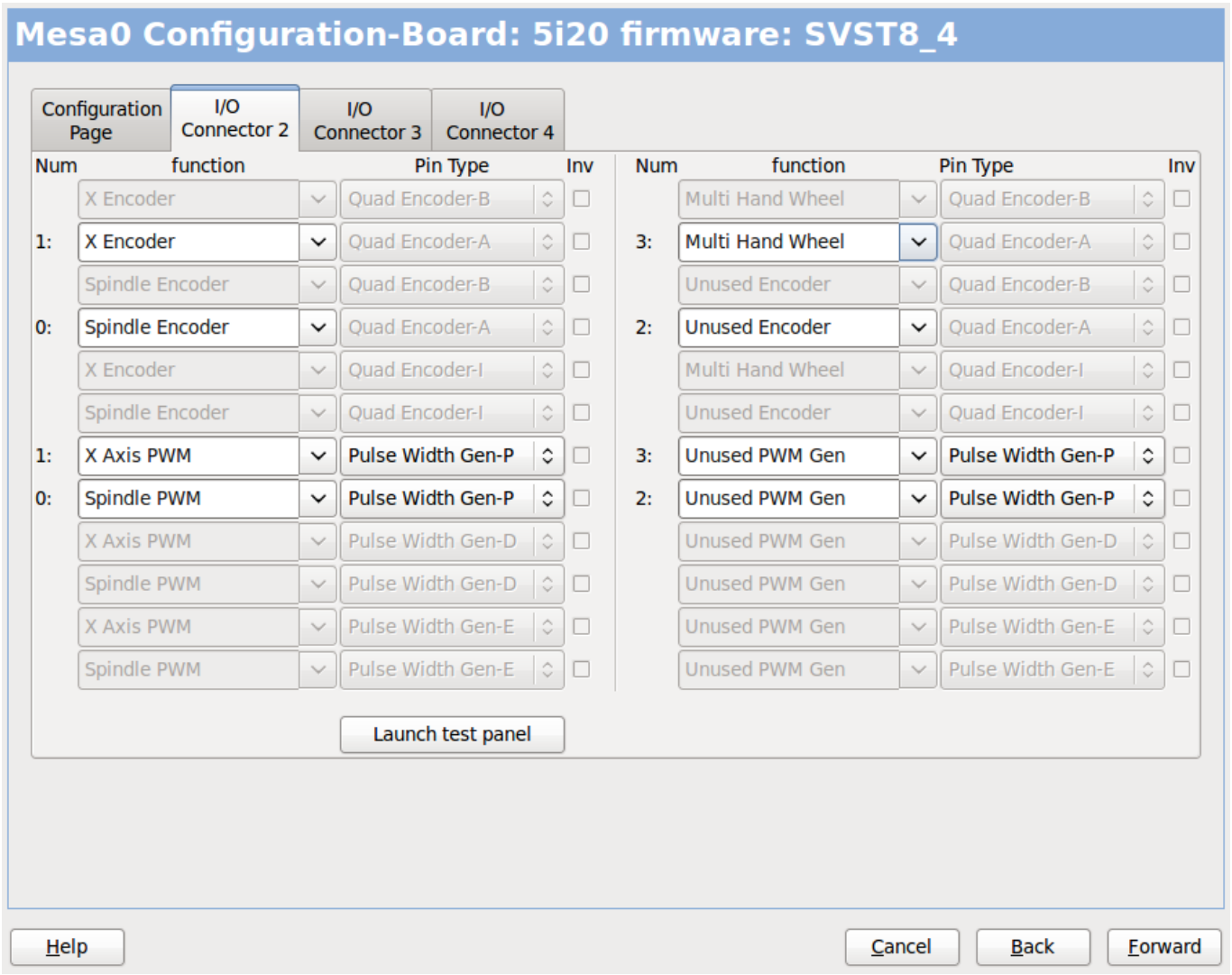


Abbildung 3.18: Mesa I/O C2 Einrichtung

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.

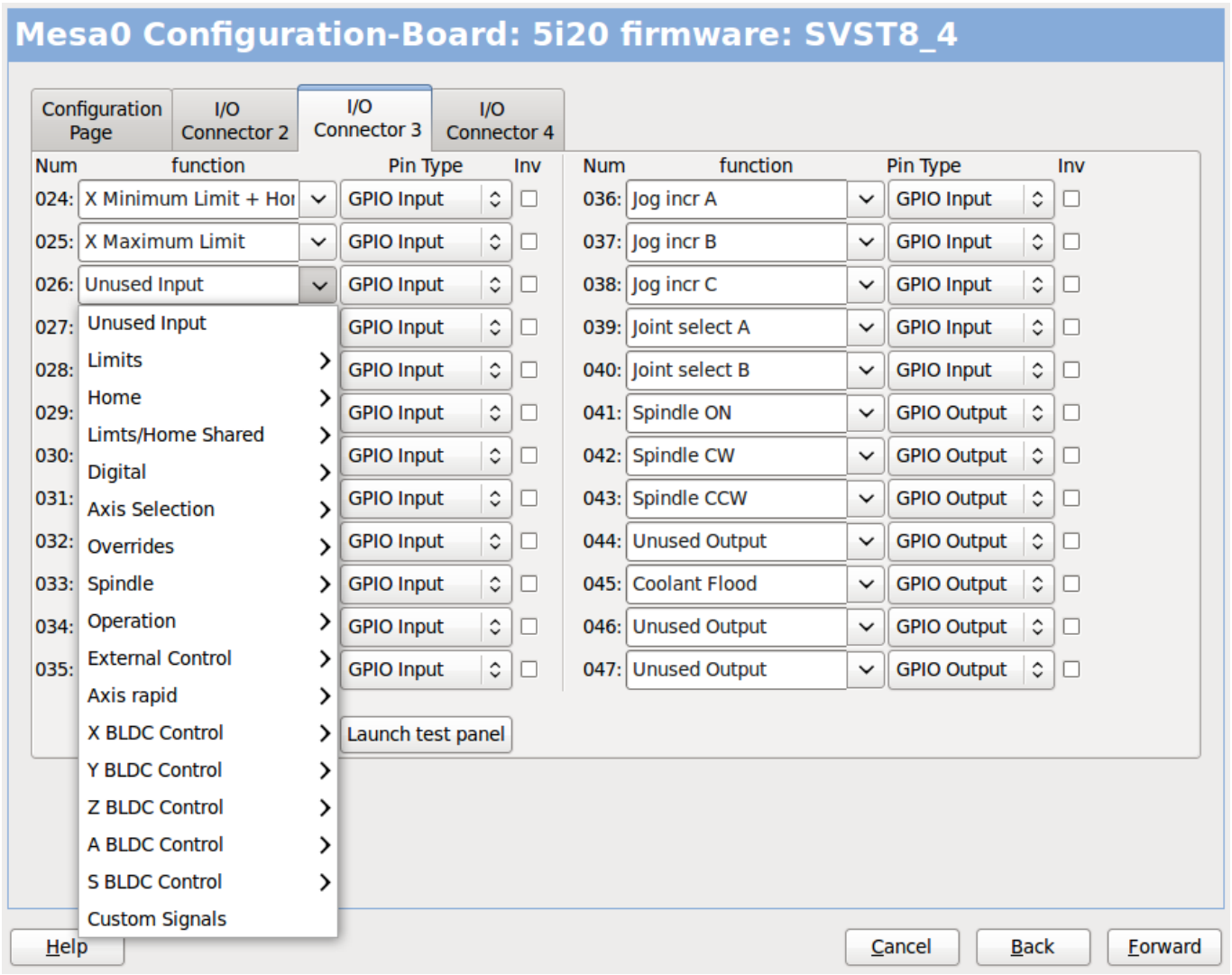


Abbildung 3.19: Mesa I/O C3 Einrichtung

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.

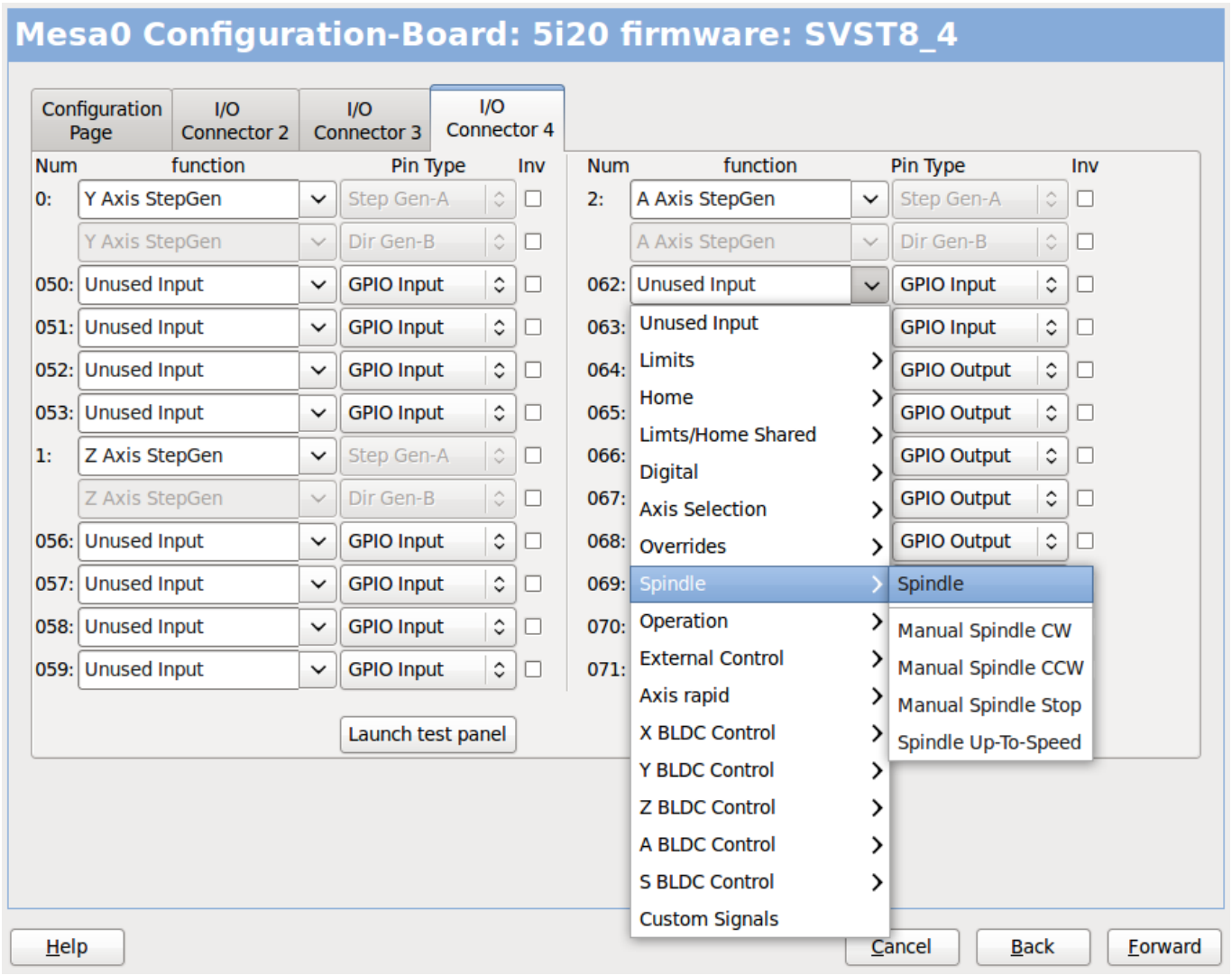
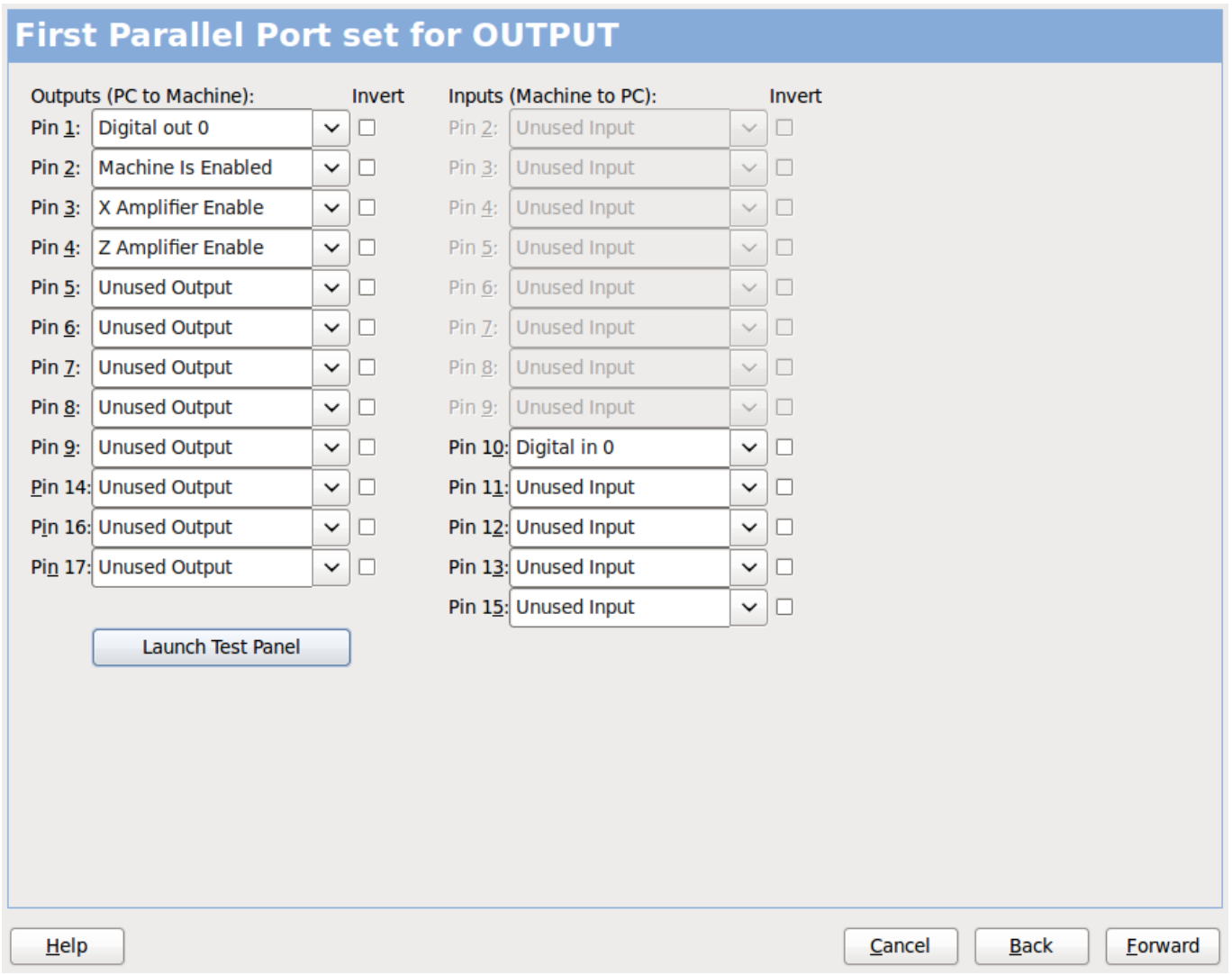


Abbildung 3.20: Mesa I/O C4 Einrichtung

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

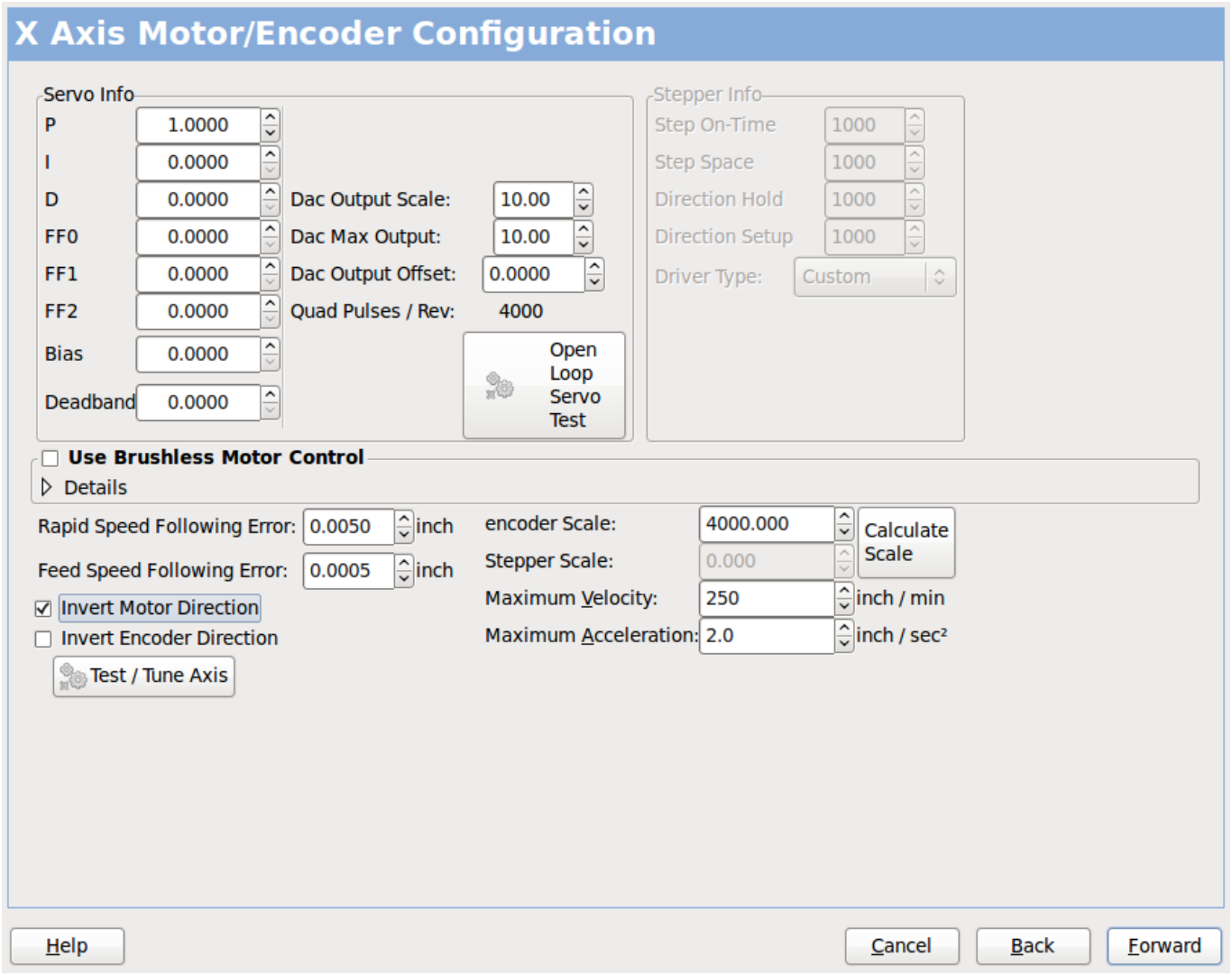


Abbildung 3.21: Konfiguration des Achsantriebs

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.

Warnung
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:

Tabelle 3.1: 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 Koeffi-

zienten 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 (Stepper-)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 http://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.

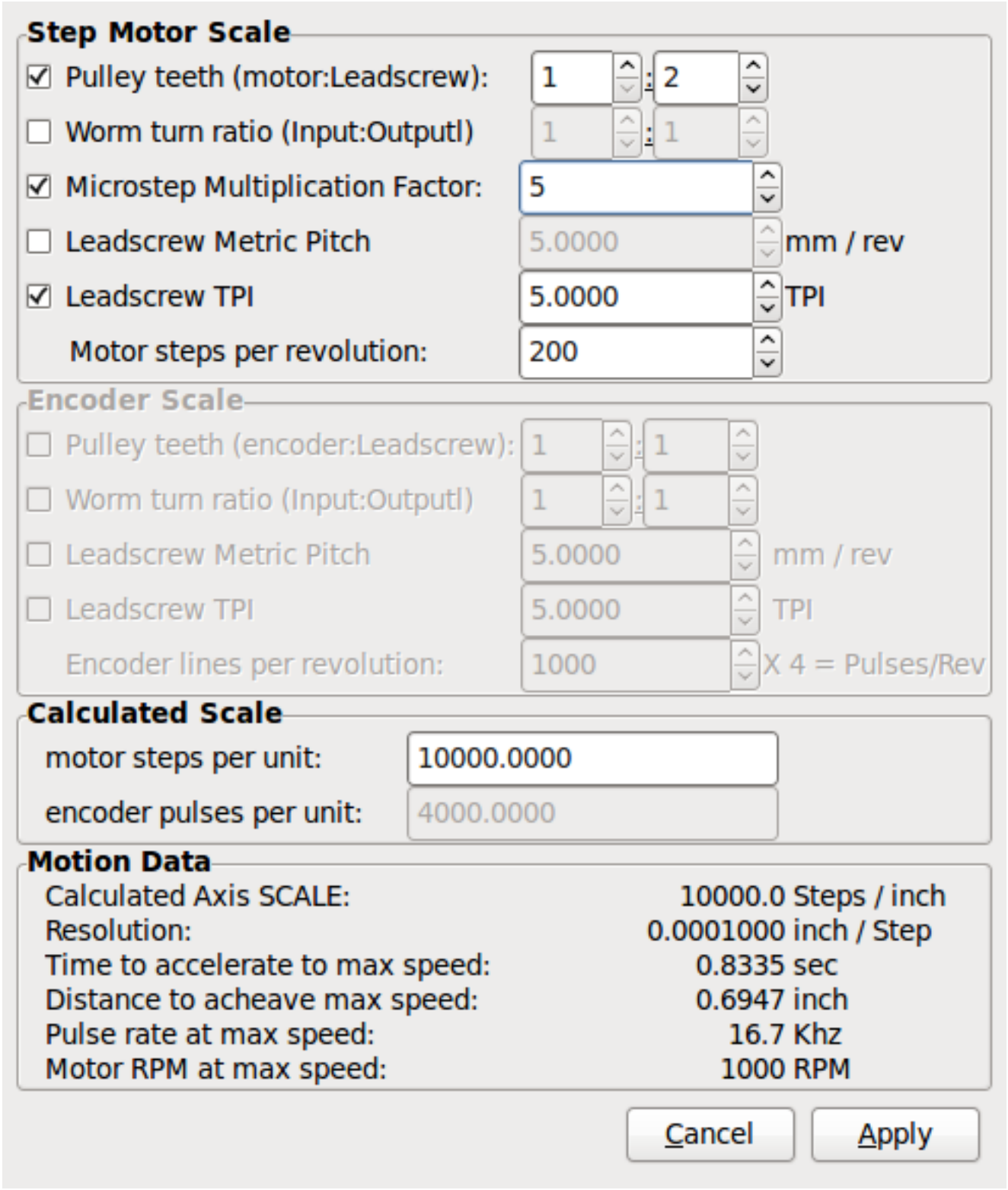


Abbildung 3.22: Berechnung der Achsenskala

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 er-

fordert das Zahnradverhältnis. Wenn Sie mit der Skala zufrieden sind, drücken Sie auf Anwenden, andernfalls auf Abbrechen und geben Sie die Skala direkt ein.

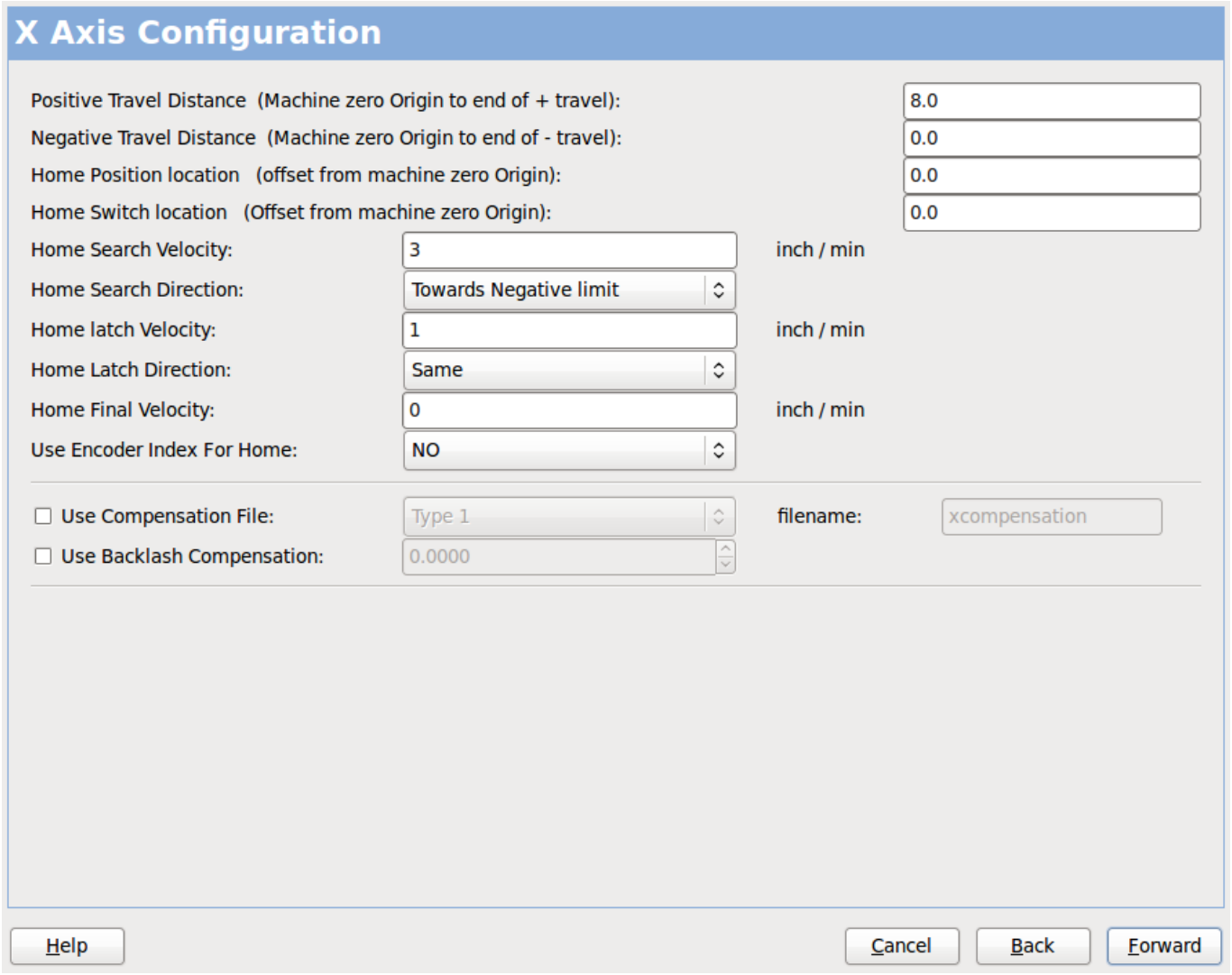


Abbildung 3.23: Konfiguration der Achsen

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.

Wichtig
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<sub:ini:sec:axis-letter,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 <sub:ini:sec:axis-letter,Achsen-Abschnitt>> des INI Kapitels.

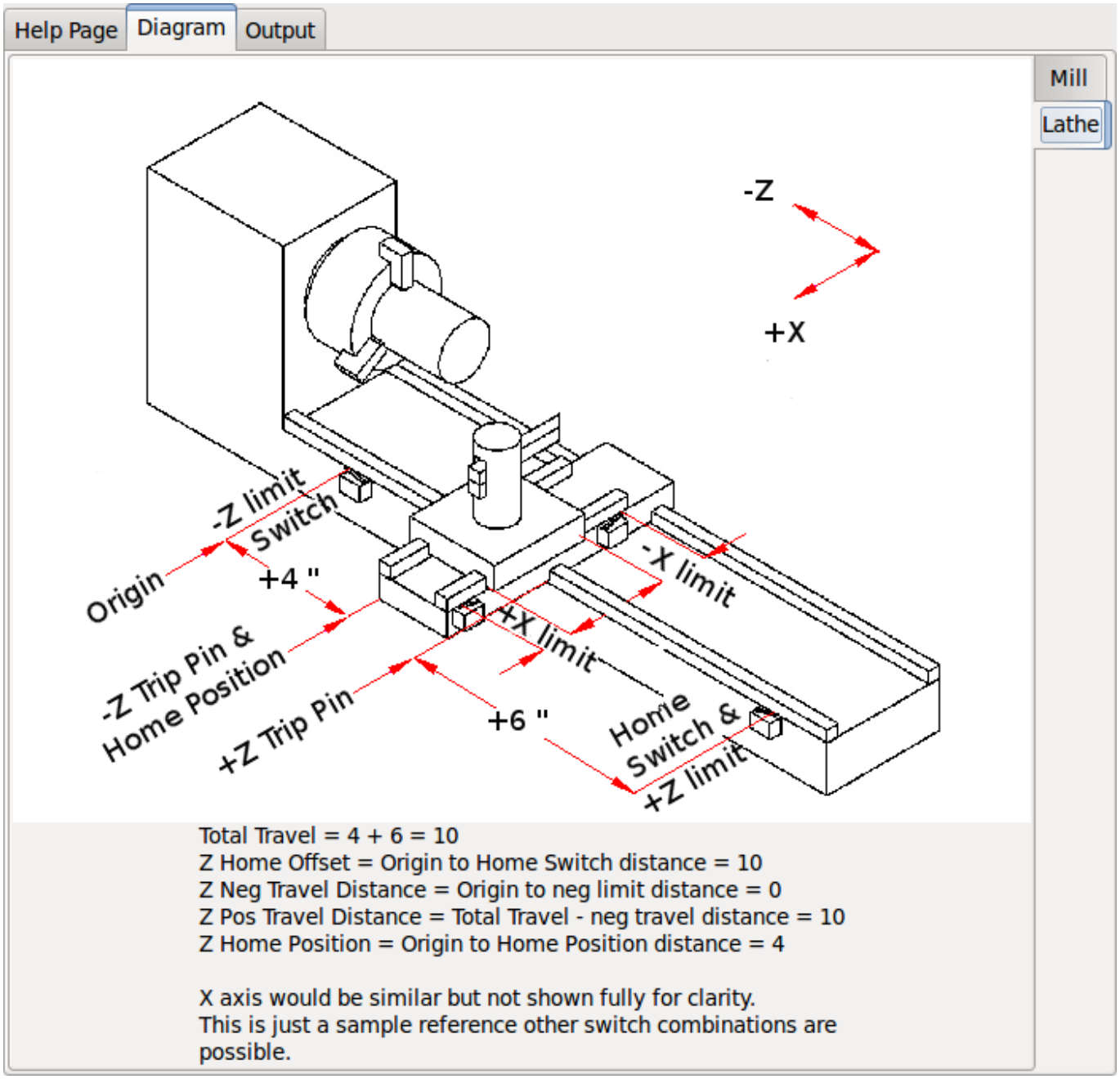


Abbildung 3.24: AXIS-Hilfsdiagramm

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.

Tipp
Viele der Optionen auf dieser Seite werden nur angezeigt, wenn auf den vorherigen Seiten die richtige Option ausgewählt wurde!

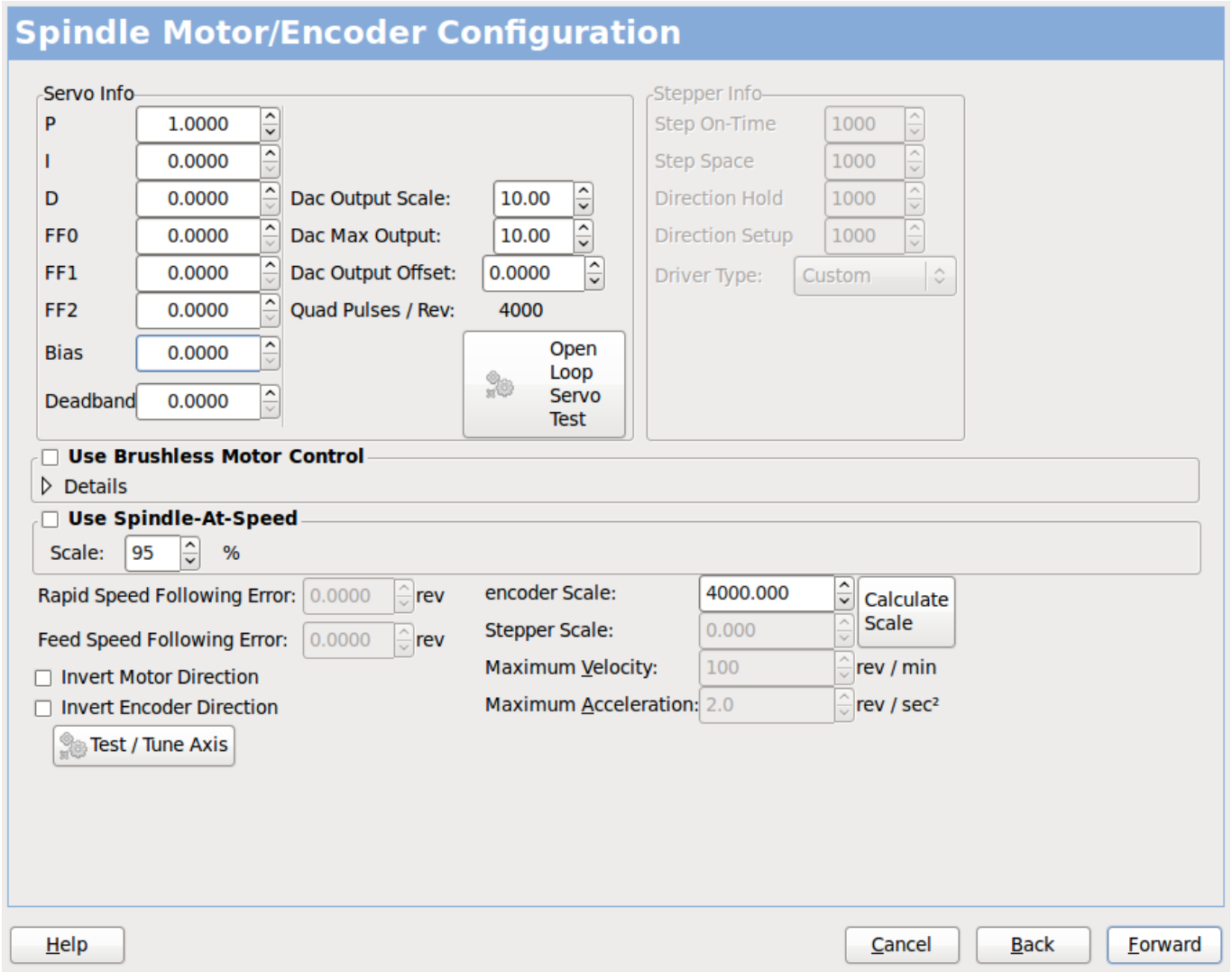


Abbildung 3.25: Spindelmotor/Encoder-Konfiguration

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 <sec:hm2-encoder,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 <cha:classicladder,ClassicLadder>> im Handbuch.

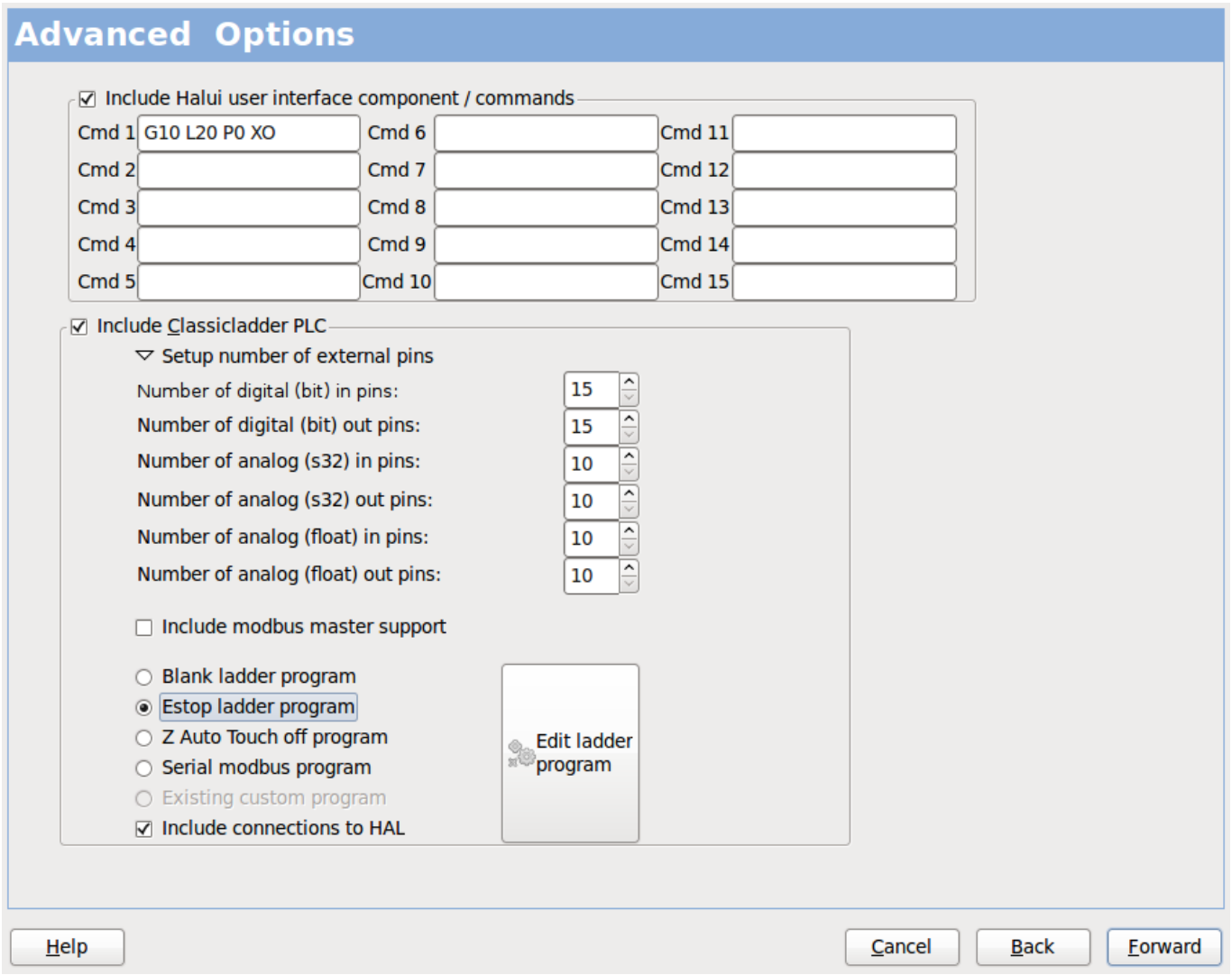


Abbildung 3.26: PnCconf, erweiterte Optionen

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.

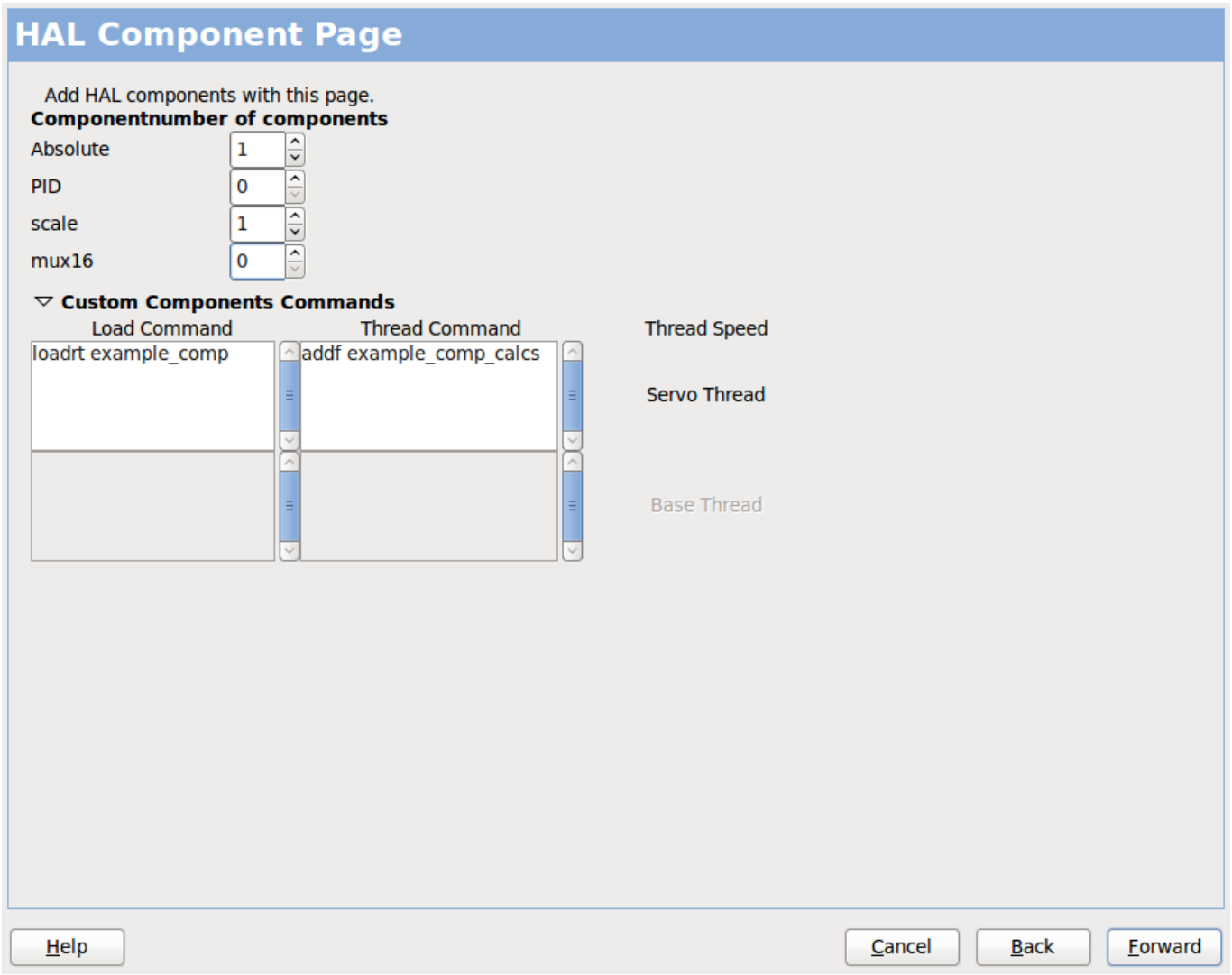


Abbildung 3.27: HAL-Komponenten

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.

Kapitel 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.

4.1.1.1 Installiert

Wenn Ihr LinuxCNC von der LiveCD 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*

4.1.1.2 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*.

Anmerkung
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.

Anmerkung
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)

4.1.3.1 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.

[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

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 Hochzeit. 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 man 15,5 µs ausprobiert, stellt man fest, dass der Computer träge ist, also begnügt man sich mit 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-Pin 5+16=21 µs , und wir müssen uns keine Sorgen mehr machen, dass der Antrieb aufgrund der Latenzzeit in die falsche Richtung schaltet.

Weitere Informationen zu stepgen finden Sie im Abschnitt stepgen.

4.1.3.2 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

4.1.4.1 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.

[2] dirhold bezieht sich auf einen Parameter, der die Länge der Richtungshaltezeit einstellt.

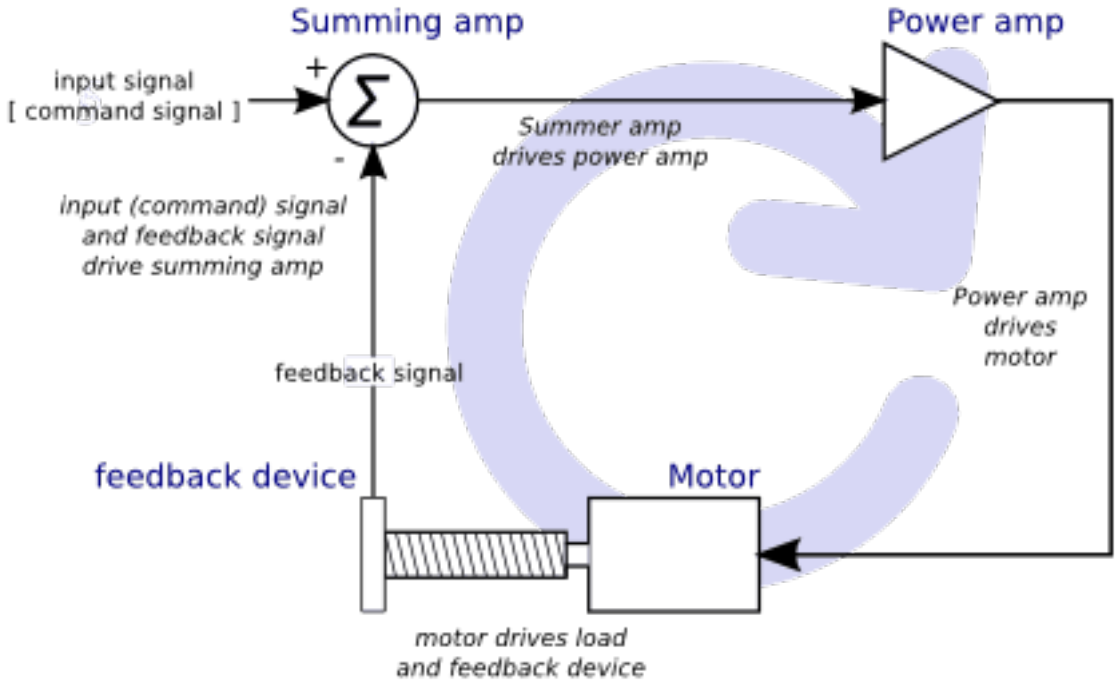


Abbildung 4.1: Servo Loop

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.

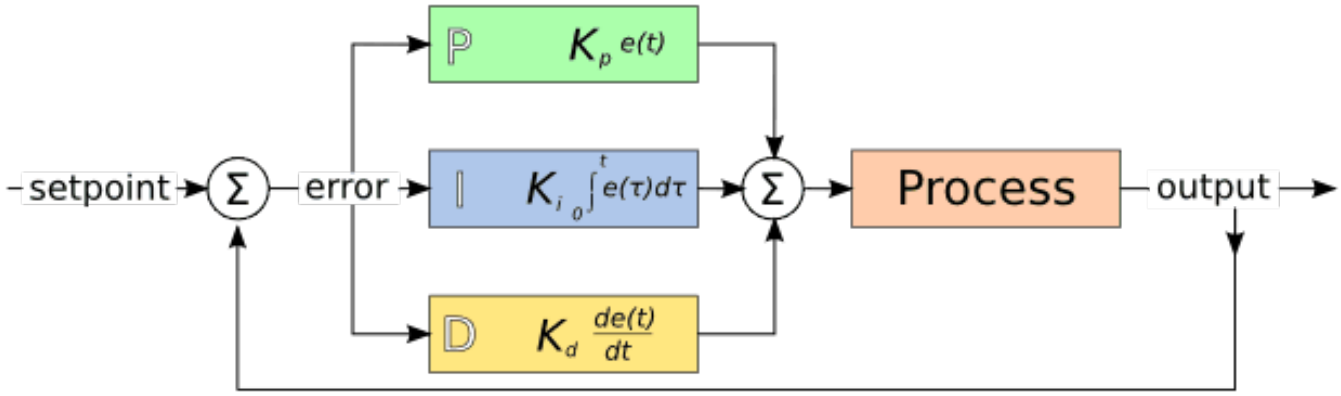


Abbildung 4.2: PID-Schleife

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.

4.1.4.2 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.

4.1.4.3 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).

4.1.4.4 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.

4.1.4.5 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.

4.1.4.6 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 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.

4.1.5.1 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.

4.2 Latenzprüfung

4.2.1 Latenz-Test

Dieser Test ist der erste Test, der an einem PC durchgeführt werden sollte, um festzustellen, ob er in der Lage ist, eine CNC-Maschine zu steuern.

Die Latenz gibt an, wie lange der PC benötigt, um seine Arbeit zu stoppen und auf eine externe Anfrage zu reagieren. Für LinuxCNC ist die Anforderung BASE_THREAD, die den periodischen ”Herzschlag” erzeugt, der als Timing-Referenz für die Schrittimpulse dient. Je niedriger die Latenz, desto schneller können Sie den Herzschlag laufen lassen und desto schneller und sanfter werden die Schrittimpulse sein.

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 RTAI-Latenztests.

Die Erzeugung von Schrittimpulsen in der Software hat einen sehr großen Vorteil - sie ist kostenlos. So gut wie 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

Der beste Weg, um herauszufinden, wie gut Ihr PC wird lrun LinuxCNC ist es, die HAL-Latenz-Test laufen. Um den Test 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
```

Nachdem Sie einen Latenztest durchgeführt haben, sollten Sie etwa Folgendes sehen:

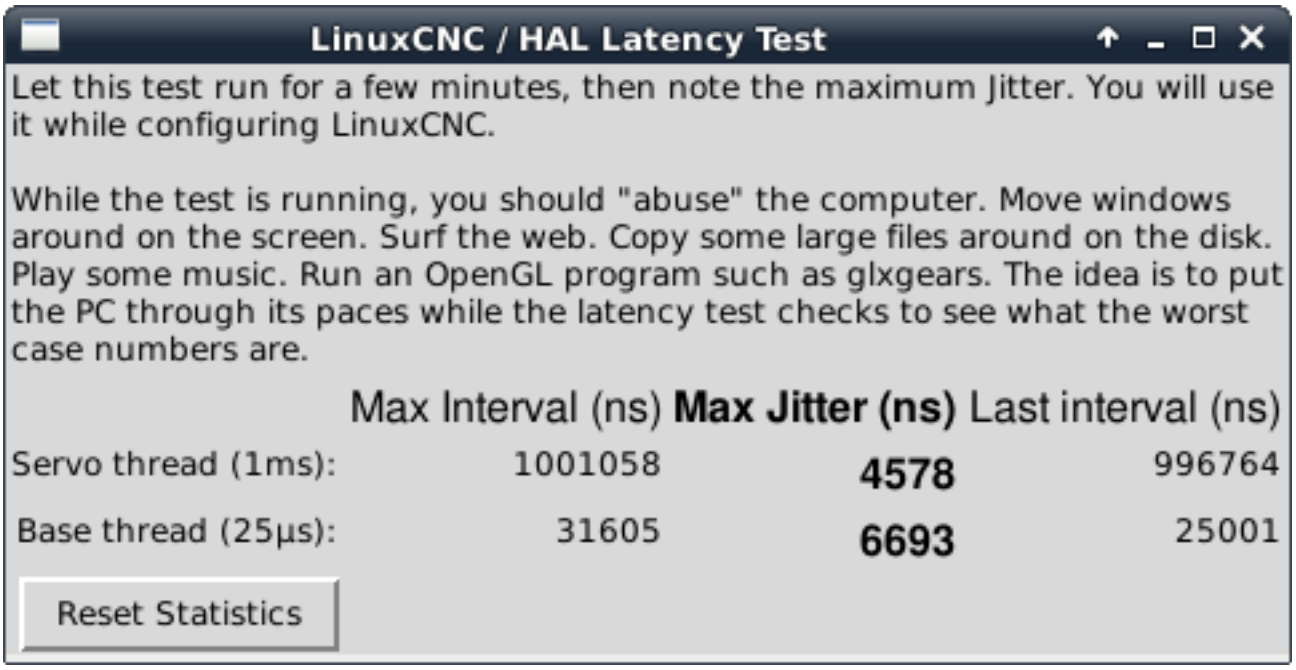


Abbildung 4.3: HAL-Latenz-Test

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.

Anmerkung
Führen Sie LinuxCNC oder Stepconf nicht aus, während der Latenztest läuft.

Die wichtige Zahl für das Software-Stepping ist der ”maximale Jitter” des Basis-Threads. Im obigen Beispiel sind das 6693 Nanosekunden, oder 6,693 Mikrosekunden. 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 http://wiki.linuxcnc.org/cgi-bin/wiki.pl?FixingSMIIssues

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 us 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.

Anmerkung
Wenn Sie hohe Zahlen erhalten, gibt es möglicherweise Möglichkeiten, sie zu verbessern. Ein anderer PC hatte eine sehr schlechte Latenz (mehrere 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.

Zusätzliche Kommandozeilen-Tools sind für die Untersuchung der Latenz verfügbar wenn LinuxCNC nicht läuft.

4.2.2 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]

Options:
      --base nS  (base  thread interval, default:   25000)
      --servo nS (servo thread interval, default: 1000000)
      --time mS  (report interval, default: 1000)
      --relative (relative clock time (default))
      --actual   (actual clock time)
```

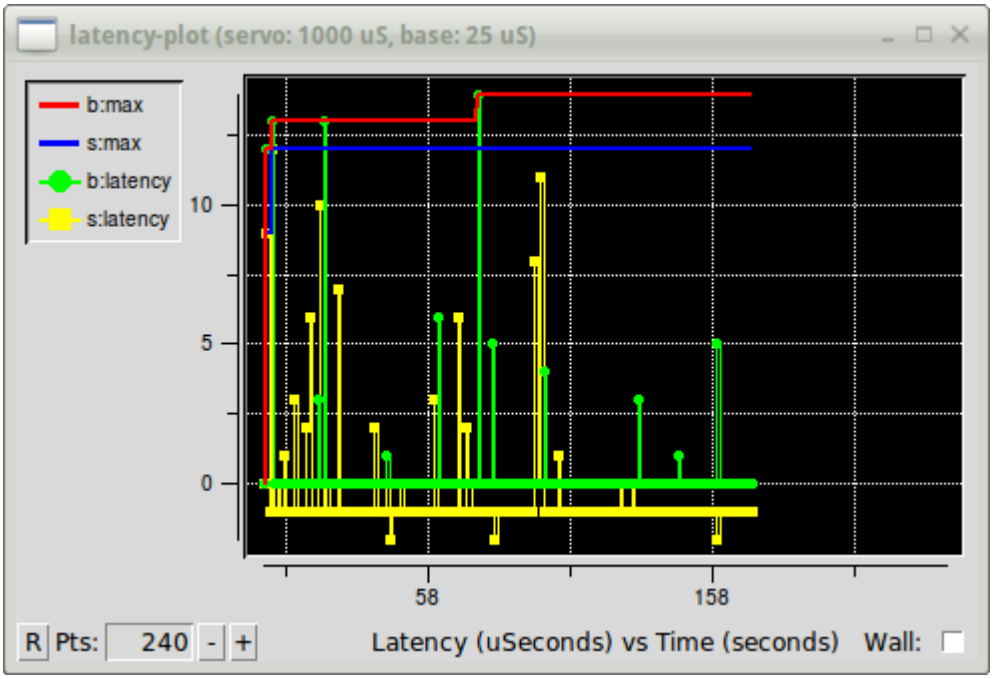


Abbildung 4.4: `latency-plot`-Fenster

4.2.3 Latenz-Histogramm

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, Voreinstellung: 25000, min: 5000)
  --servo ns (Servo-Thread-Intervall, Voreinstellung: 1000000, Mindestwert: 25000)
  --bbinsize ns (Basis-Bin-Größe, Voreinstellung: 100
  --sbinsize ns (Servo-Bin-Größe, 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)

Notes:
  LinuxCNC and HAL should not be running, stop with halrun -U.
  Large number of bins and/or small binsizes will slow updates.
  For single thread, specify --nobase (and options for servo thread).
  Measured latencies outside of the +/- bin range are reported
  with special end bars.  Use --show to show count for
  the off-chart [pos|neg] bin
```

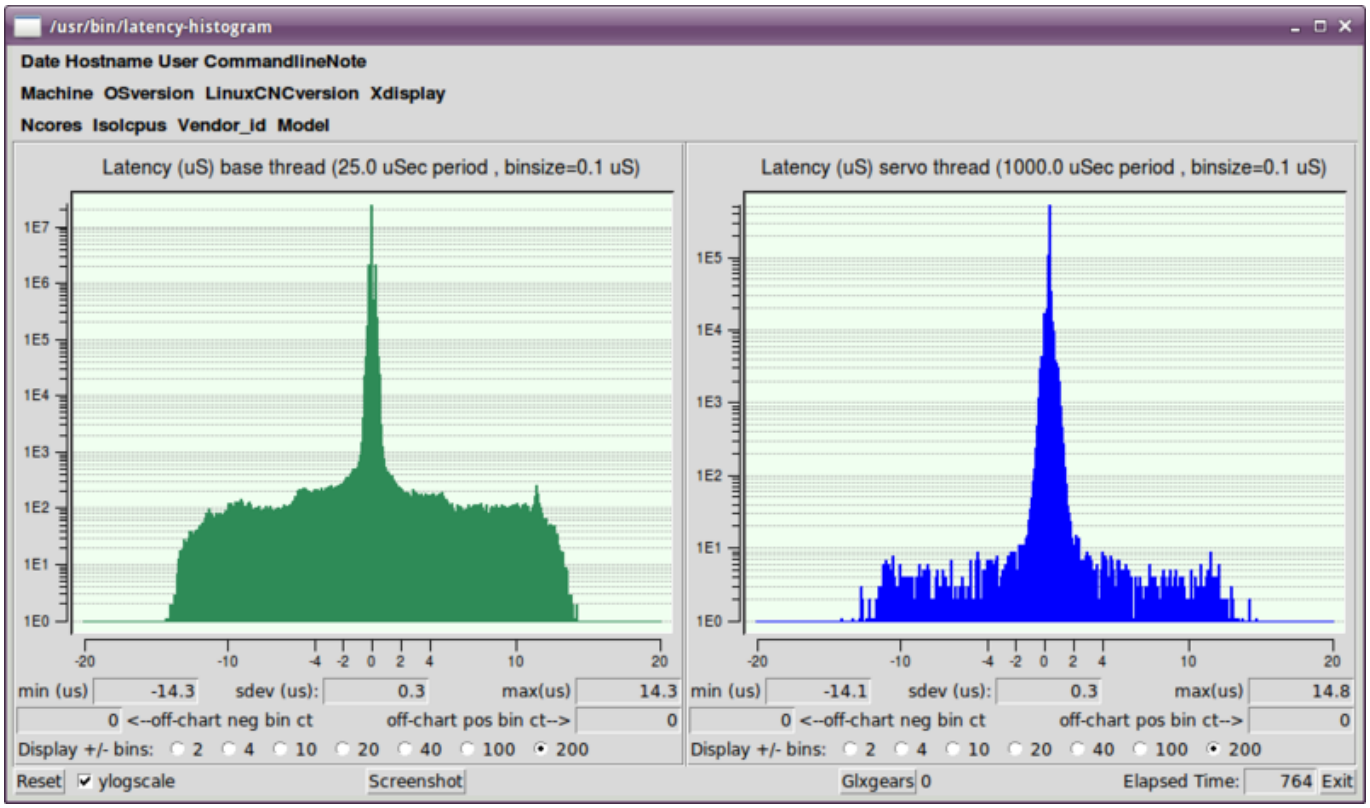


Abbildung 4.5: *Latenz-Histogramm*-Fenster

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.

4.3.1.1 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 us 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.

4.3.1.2 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.

4.3.1.3 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.

4.3.1.4 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.

4.3.1.5 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.

4.4.1.1 Kommentare

Eine Kommentarzeile wird mit einem ; oder einem # eingeleitet. Wenn der INI-Leser eines dieser Zeichen am Anfang einer Zeile sieht, wird der Rest der Zeile von der Software ignoriert. Kommentare können verwendet werden, um zu beschreiben, was ein INI-Element tun wird.

```
; 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.

Beachten Sie, dass die Zeichen "#" und ";" innerhalb einer Variablen nicht für Kommentare stehen:

```
FALSCH = Wert # und ein Kommentar

# Korrekter Kommentar
CORRECT = Wert
```

4.4.1.2 Abschnitte

Zusammenhängende Teile einer INI-Datei sind in Abschnitte unterteilt. Der Name eines Abschnitts ist in Klammern eingeschlossen, etwa so *[DIESER ABSCHNITT]*. Die Reihenfolge der Abschnitte ist unerheblich. Die Abschnitte beginnen mit dem Abschnittsnamen und enden mit dem nächsten Abschnittsnamen.

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

4.4.1.3 Variablen

Eine Variablenzeile besteht aus einem Variablennamen, einem Gleichheitszeichen (=) und einem Wert. Als Wert wird alles vom ersten nicht-weißen Leerzeichen nach dem = bis zum Ende der Zeile übergeben, so dass Sie Leerzeichen in Stringsymbole einbetten können, wenn Sie dies wollen oder müssen. Ein Variablenname wird oft auch als Schlüsselwort bezeichnet.

Beispiel für eine Variable

```
MACHINE = My Machine
```

Eine variable Zeile kann mit einem terminalen Backslash (\) auf mehrere Zeilen erweitert werden. Es sind maximal MAX_EXTEND_LINES (==20) zulässig. Nach dem abschließenden Backslash-Zeichen darf kein Leerzeichen stehen.

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
```

Boolean Variables Boolean values can be on one of `TRUE`, `YES` or `1` for true/enabled and one of `FALSE`, `NO` or `0` for false/disabled. The case is ignored.

In den folgenden Abschnitten wird jeder Abschnitt der Konfigurationsdatei anhand von Beispielwerten für die Konfigurationszeilen erläutert.

Variablen, die von LinuxCNC verwendet werden, müssen immer die Sektionsnamen und Variablennamen wie gezeigt verwenden. Im folgenden Beispiel wird die Variable *MACHINE* mit dem Wert *My Machine* belegt.

4.4.1.4 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

```
[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
```

Anmerkung
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]v`
Das folgende Beispiel zeigt eine einfache Z-Achsen-Antastroutine für eine Oberfräse oder ein Fräswerk unter Verwendung einer Tastplatte.

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>
```

4.4.1.5 Include-Dateien

Durch die Angabe einer #INCLUDE-Anweisung in einer INI-Datei kann der Computer dazu angehalten werden, an dieser Stelle zunächst den Inhalt der angegebenen Datei zu berücksichtigen.

#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 /)
- einen Dateinamen, der sich auf den Wohnort des Benutzers bezieht (beginnt mit ~)

Mehrere #INCLUDE-Direktiven werden unterstützt.

#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
```

Die #INCLUDE-Direktiven werden nur für eine Erweiterungsebene unterstützt - eine bereits inkludierte Datei darf keine weiteren Dateien einschließen. Die empfohlene Dateierweiterung ist .inc. Verwenden Sie nicht die Dateierweiterung .ini für eingeschlossene Dateien.

4.4.2 INI-Datei Abschnitte

4.4.2.1 [EMC] Abschnitt

- *VERSION = 1.1* - Die Versionsnummer für die Konfiguration. Jeder andere Wert als 1.1 führt dazu, dass die Konfigurationsprüfung ausgeführt wird und versucht, die Konfiguration auf den neuen Typ der Gelenkachsen-Konfiguration zu aktualisieren.
- *MACHINE = My Controller* - Dies ist der Name des Controllers, der in den meisten grafischen Oberflächen oben angezeigt wird. Sie können hier einfügen was Sie wollen, solange es nur eine Zeile lang ist.
- *DEBUG = 0* - Debug-Level 0 bedeutet, dass keine Meldungen ausgegeben werden, wenn LinuxCNC von einem Terminal ausgeführt wird. Debug-Flags sind normalerweise nur für Entwickler nützlich. Siehe src/emc/nml_intf/debugflags.h für andere Einstellungen.

4.4.2.2 [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* – Der Dateiname der ausführbaren Datei, welche die zu verwendende Benutzeroberfläche bereitstellt. Prominente gültige Optionen sind (alle in Kleinbuchstaben): *axis*, *touchy*, *gmoccapy*, *gscreen*, *tklinuxcnc*, *qtvcp*, *qtvcp-qtdragon* oder *qtvcp-qtplasmac*.
- *POSITION_OFFSET = RELATIVE* - Das Koordinatensystem (RELATIVE oder MACHINE), das beim Start der Benutzeroberfläche auf dem DRO angezeigt wird. Das RELATIVE Koordinatensystem spiegelt die derzeit gültigen G92- und G5x-Koordinatenoffsets wider.
- *POSITION_FEEDBACK = COMMANDED* - Der Koordinatenwert (COMMANDED oder ACTUAL), der auf der externen Digitalanzeige (DRO) angezeigt werden soll, wenn die Benutzeroberfläche startet. In Axis kann dies über das Menü View geändert werden. Die COMMANDED-Position ist die von LinuxCNC angeforderte Position. Die IST-Position ist die von den Motoren zurückgemeldete Position (engl. feedback position), wenn sie wie die meisten Servosysteme solche Funktion haben. Normalerweise wird der COMMANDED-Wert verwendet.
- *DRO_FORMAT_MM = %+08.6f* - Setzt die Standard-DRO-Formatierung im metrischen Modus außer Kraft (normalerweise 3 Dezimalstellen, aufgefüllt mit Leerzeichen auf 6 Ziffern nach links). Das obige Beispiel füllt mit Nullen auf, zeigt 6 Dezimalstellen an und erzwingt die Anzeige eines +-Zeichens für positive Zahlen. Die Formatierung folgt der Python-Praxis. https://docs.python.org/-2/library/string.html#format-specification-mini-language gibt einen Fehler aus, wenn das Format keine Fließkommazahlen akzeptieren kann.
- *DRO_FORMAT_IN = % 4.1f* - Überschreibt die Standard-DRO-Formatierung im imperialen Modus (normalerweise 4 Dezimalstellen, aufgefüllt mit Leerzeichen auf 6 Ziffern nach links) - das obige Beispiel zeigt nur eine Dezimalstelle an. Die Formatierung folgt der Python-Praxis. https://docs.python.org/-2/library/string.html#format-specification-mini-language Ein Fehler wird ausgelöst, wenn das Format keine Fließkommazahlen akzeptieren kann.
- *CONE_BASESIZE = .25* - Überschreibt die Standardkegel-/Werkzeugbasisgröße von .5 in der Grafikanzeige

- *MAX_FEED_OVERRIDE = 1.2* - Der maximale Vorschub-Override, den der Benutzer auswählen kann. Der Wert 1.2 (bitte mit Dezimalpunkt, nicht Komma) bedeutet 120% des programmierten Vorschubs.
- *MIN_SPINDLE_OVERRIDE = 0.5* - Der minimale Spindel-Override, den der Benutzer auswählen kann. 0.5 (immer mit Dezimalpunkt, nicht Komma) bedeutet 50% der programmierten Spindeldrehzahl. (Dies wird verwendet, um die minimale Spindeldrehzahl einzustellen).
- MIN_SPINDLE_0_OVERRIDE = 0.5’ - Der minimale Spindel-Override, den der Benutzer auswählen kann. 0,5 bedeutet 50% der programmierten Spindeldrehzahl. (Dies wird verwendet, um die minimale Spindeldrehzahl einzustellen). Bei Mehrspindelmaschinen gibt es Einträge für jede Spindelnummer. Wird nur von den QtVCP-basierten Benutzeroberflächen verwendet.
- *MAX_SPINDLE_OVERRIDE = 1.0* - Der maximale Spindel-Override, den der Benutzer auswählen kann. 1.0 bedeutet 100% der programmierten Spindeldrehzahl.
- *MAX_SPINDLE_0_OVERRIDE = 1.0* - Der maximale Vorschub-Override, den der Benutzer wählen kann. 1.2 bedeutet 120% der programmierten Vorschubgeschwindigkeit. Bei Mehrspindelmaschinen gibt es Einträge für jede Spindelnummer. Wird nur von den QtVCP-basierten Benutzeroberflächen verwendet.
- *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* - Der Standardspeicherort für G-Code-Dateien und der Speicherort für benutzerdefinierte M-Codes. Dieser Speicherort wird nach dem Dateinamen vor dem Unterprogramm-Pfad und dem Benutzer-M-Pfad durchsucht, wenn er im Abschnitt [RS274NGC] angegeben wurde.
- *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.

Anmerkung
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*

Anmerkung
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 <sub:ini:sec:tr
units>> 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 um 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* - Jeder nicht leere Wert (einschließlich ”0”) bewirkt, dass die Achse den Drehmaschinenmodus mit Draufsicht und mit Radius und Durchmesser auf dem DRO anzeigt.
- *BACK_TOOL_LATHE = 1* - Jeder nicht leere Wert (einschließlich ”0”) bewirkt, dass die Achse den ”Back Tool Lathe Modus” mit invertierter X-Achse verwendet.
- *FOAM = 1* - Jeder nicht leere Wert (einschließlich ”0”) veranlasst die Achse, die Anzeige für den Schaumstoffschneidermodus zu ändern.
- *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. Wenn das Sonderzeichen ”**!**” irgendwo in der Sequenz auftaucht, werden die Drehungen für die Buchstaben der A-, B- und C-Achse unter Berücksichtigung der auf X, Y und Z angewandten Offsets (G5x, G92) durchgeführt.
  8. 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.
  9. 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.
  10. Bei Drehungen, die nach einem Versatz angewendet werden, kann die Bestimmung ”**!**” verwendet werden, um in Bezug auf den aktuellen Maschinenversatz zu wirken. Beispiel: GEOMETRY=!CXYZ übersetzt den Kontrollpunkt in die X, Y, Z-Position und führt dann eine C-Drehung um den Maschinenursprung aus, der um die aktuellen G5x,G92 X, Y, Z-Offsets verschoben ist. (Das Ändern von Offsets kann ein Neuladen des Programms erfordern).
  11. Beispiel für UVW-Verschiebung: GEOMETRY=XYZUVW bewirkt, dass UVW im Koordinatensystem des Werkzeugs und XYZ im Koordinatensystem des Materials verschoben wird.

12. 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.

Anmerkung
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.

Anmerkung
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* - The name of an optional, configuration-specific Python file sourced by the axis GUI instead of the user-specific file `~/.axisrc`.

Anmerkung
Der folgende Abschnitt [DISPLAY] (engl. für Anzeige) wird nur von der TKLinuxCNC-Schnittstelle verwendet.

- *HELP_FILE = tklinucnc.txt* - Pfad zur Hilfedatei.

4.4.2.3 [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
```

Anmerkung
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 http://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.

4.4.2.4 [RS274NGC] Abschnitt

- *PARAMETER_FILE = myfile.var* - Die Datei, 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 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.
- *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. [ANZEIGE]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 M1xx wird der erste bei der Suche gefundene ausführbare M1xx verwendet.

Anmerkung
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* Voreinstellung 0 Erlaubt das automatische Löschen des G92-Offsets beim Start der Konfiguration.
- *DISABLE_FANUC_STYLE_SUB = 0* Standardwert 0 Wenn es einen Grund gibt, Fanuc-Unterprogramme zu deaktivieren, setzen Sie diesen Wert auf 1.

Anmerkung
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
```

Anmerkung
[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*, Default 0, Bestimmt log_level (Voreinstellung: -1)
- *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.

4.4.2.5 [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 die Motion Abschnitt.

- *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] Die Variable HOMEMOD ist optional. Wenn sie angegeben ist, wird ein bestimmtes (vom Benutzer erstelltes) Modul anstelle des Standardmoduls (homemod) verwendet. Modulparameter (home_parms) können einbezogen werden, wenn sie von dem angegebenen Modul unterstützt werden. Die Einstellung kann von der Befehlszeile aus mit der Option -m überschrieben werden ($ linuxcnc -h)

4.4.2.6 [TASK] Abschnitt

- *TASK = milltask* - Gibt den Namen der *Task* ausführbaren Datei an. Die ausführbare Datei *task* erledigt verschiedene Aufgaben, wie z.B. die Kommunikation mit den Benutzeroberflächen über NML, die Kommunikation mit dem Echtzeit-Bewegungsplaner über nicht-HAL Shared Memory und die Interpretation von G-Code. Derzeit gibt es nur eine ausführbare Aufgabe, die für 99,9 % der Benutzer sinnvoll ist: milltask.
- *CYCLE_TIME = 0.010* - Der Zeitraum in Sekunden, in dem TASK ausgeführt wird. Dieser Parameter wirkt sich auf das Abfrageintervall aus, wenn auf den Abschluss einer Bewegung gewartet wird, wenn ein Pausenbefehl ausgeführt wird und wenn ein Befehl von einer Benutzeroberfläche angenommen wird. In der Regel ist es nicht erforderlich, diese Zahl zu ändern.

4.4.2.7 [HAL] Abschnitt

- *HALFILE = example.hal* - Führt die Datei *example.hal* beim Start aus.

  HALFILE = example.hal - Führt die Datei *example.hal* beim Starten aus. Wenn *HALFILE* mehrfach angegeben wird, werden die Dateien in der Reihenfolge ausgeführt, in der sie in der INI-Datei stehen. Fast alle Konfigurationen haben mindestens eine *HALFILE*, und Steppersysteme haben typischerweise zwei solcher Dateien, eine zur Spezifikation der allgemeinen Stepperkonfiguration (*core_stepper.hal*) und eine für die Beschreibung der Pinbelegung der Maschine (*xxx_pinout.hal*).

  HALFILES werden durch eine Suche gefunden. Wenn die benannte Datei in dem Verzeichnis gefunden wird, das die INI-Datei enthält, wird sie verwendet. Wird die benannte Datei nicht in diesem INI-Dateiverzeichnis 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 <cha:haltcl,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-Elemente spezifizieren Dateien, die HAL-Komponenten laden und Signalverbindungen zwischen Komponentenpins herstellen. Häufige Fehler sind 1) das Fehlen der addf-Anweisung, die benötigt wird, um die Funktion(en) einer Komponente zu einem Thread hinzuzufügen, 2) unvollständige Signal-(Netz-)Spezifizierungen. 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 Popup-GUI zu melden:

```
HALFILE = LIB:halcheck.tcl [nopopup]
```

Anmerkung
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 für das Laden von HAL-Komponenten. Bei der TWOPASS-Verarbeitung werden die [HAL]HALFILE=-Zeilen in zwei Durchgängen verarbeitet. Im ersten Durchlauf (pass0) werden alle HALFILES gelesen und mehrfache 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* - Die TWOPASS-Funktion 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 erscheinen. *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).

4.4.2.8 [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.

4.4.2.9 [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 = value* - Sekunden, die vor dem Start anderer Anwendungen gewartet werden. Eine Verzögerung kann erforderlich sein, wenn eine Anwendung Abhängigkeiten von [HAL]POSTGUI_HALFILE-Aktionen oder von durch das GUI erstellten HAL Pins hat (Standard 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 Verzeichnis inifile 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
```

4.4.2.10 Abschnitt [TRAJ]

Warnung
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* - Neuen TP einschalten. Wenn auf 0 gesetzt, verwendet TP parabolisches Blending (1 Segment vorausschauend). Standardwert 1.
- *ARC_BLEND_FALLBACK_ENABLE = 0* - Optionaler Rückgriff auf parabolische Blends, wenn die geschätzte Geschwindigkeit höher ist. Diese Schätzung ist jedoch grob, und es scheint, dass die Deaktivierung eine bessere Leistung erzielt. Standardwert 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 Sekunden 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*).
- *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>* - Gibt die *Maschineneinheiten* für Rotationsachsen an. Mögliche Auswahlen sind *deg*, *degree* (360 pro Kreis), *rad*, *radian* (2*π pro Kreis), *grad*, oder *gon* (400 pro Kreis). Dies hat keinen Einfluss auf die Winkeleinheiten des NC-Codes. In RS274NGC werden die A-, B- und C-Wörter immer in Grad ausgedrückt.
- *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.
- *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* - Das Standardverhalten ist für LinuxCNC, um den Benutzer zu zwingen, eine Referenzfahrt durchzuführen, bevor ein MDI-Befehl oder ein Programm ausgeführt wird. Normalerweise ist nur Jogging vor der Referenzfahrt erlaubt. Bei Konfigurationen, die Identitätskinematiken verwenden, erlaubt die Einstellung NO_FORCE_HOMING = 1 dem Benutzer, MDI-Bewegungen auszuführen und Programme zu starten, ohne die Maschine zuerst zu referenzieren. Bei Schnittstellen, die Identitätskinematiken ohne Referenzfahrtmöglichkeit verwenden, muss diese Option auf 1 gesetzt werden.

Warnung
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. Hinweis: 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.

4.4.2.11 [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*

4.4.2.12 [AXIS_<letter>] Abschnitt

Der <letter> (engl. Buchstabe) gibt einen der folgenden Buchstaben an: X Y Z A B C U V W

- *MAX_VELOCITY = 1.2* - Maximale Geschwindigkeit für diese Achse in <sub:ini:sec:traj,Maschineneinheite
  pro Sekunde.
- *MAX_ACCELERATION = 20.0* - Maximale Beschleunigung für diese Achse in Maschineneinheiten pro Sekunde zum Quadrat.
- *MIN_LIMIT = -1000* - Der minimale Grenzwert (Softlimit) für die Achsenbewegung, in Maschineneinheiten. Wenn dieser Grenzwert überschritten wird, bricht die Steuerung die Achsenbewegung ab. Die Achse muss referenziert werden, bevor MIN_LIMIT in Kraft tritt. Für eine Rundachse (Typ A,B,C) mit unbegrenzter Drehung, für die im Abschnitt [AXIS_<letter>] kein MIN_LIMIT angegeben ist, wird ein Wert von -1e99 verwendet.
- *MAX_LIMIT = 1000* - Der maximale Grenzwert (Softlimit) für die Achsenbewegung in Maschineneinheiten. Wenn dieser Grenzwert überschritten wird, bricht die Steuerung die Achsenbewegung ab. Die Achse muss referenziert werden, bevor MAX_LIMIT in Kraft tritt. Für eine Rundachse (Typ A,B,C) mit unbegrenzter Drehung, für die im Abschnitt [AXIS_<letter>] kein MAX_LIMIT angegeben ist, wird ein Wert von 1e99 verwendet.
- *WRAPPED_ROTARY = 1* - Wenn dies für eine ANGULAR-Achse auf 1 gesetzt wird, bewegt sich die Achse um 0-359,999 Grad. Positive Zahlen bewegen die Achse in eine positive Richtung und negative Zahlen bewegen die Achse in eine negative Richtung.

- *LOCKING_INDEXER_JOINT = 4* - Dieser Wert wählt ein Gelenk aus, das für einen verriegelnden Indexer für die angegebene Achse <Buchstabe> verwendet wird. In diesem Beispiel ist das Gelenk 4, was der B-Achse für ein XYZAB-System mit Trivkins-Kinematik (Identität) entsprechen würde. Bei einer G0-Bewegung für diese Achse wird eine Entriegelung mit dem joint.4.unlock-Pin eingeleitet, dann wird auf den joint.4.is-unlocked-Pin gewartet und dann das Gelenk mit der für dieses Gelenk vorgesehenen Eilgeschwindigkeit bewegt. Nach der Bewegung wird der joint.4.unlock auf false gesetzt und die Bewegung wartet, bis joint.4.is-unlocked auf false gesetzt wird. Die Bewegung mit anderen Gelenken ist nicht erlaubt, wenn ein gesperrtes Drehgelenk bewegt wird. Um die Entriegelungsstifte zu erstellen, verwenden Sie den Parameter motmod:

```
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* - Wenn ungleich Null, aktiviert dieses Element die Verwendung von HAL-Eingangsstiften für externe Achsen-Offsets:

```
axis.<letter>.eoffset-enable
axis.<letter>.eoffset-count
axis.<letter>.eoffset-scale
```

Siehe das Kapitel: *cha:external-offsets,'External Axis Offsets>>* für Informationen zur Verwendung.

4.4.2.13 [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 Koordinaten=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: *$ man kins*

- *TYPE = LINEAR* - Die Art der Verbindung, entweder LINEAR oder ANGULAR (engl. für Winkel- bestimmt).
- *UNITS = INCH* - Falls angegeben, hat diese Einstellung Vorrang vor der zugehörigen [TRAJ]-Einheit-Einstellung. (z.B. [TRAJ]LINEAR_UNITS wenn der TYP dieses Gelenks LINEAR ist, [TRAJ]ANGULAR_UNIT wenn der TYP dieses Gelenks ANGULAR ist)
- *MAX_VELOCITY = 1.2* - Maximale Geschwindigkeit für dieses Gelenk in Maschineneinheiten pro Sekunde.
- *MAX_ACCELERATION = 20.0* - Maximale Beschleunigung für diese Achse in Maschineneinheiten pro Sekunde zum Quadrat.
- *BACKLASH = 0.0000* - Umkehrspiel in Maschineneinheiten. Der Kompensationswert für das Spiel kann verwendet werden, um kleine Unzulänglichkeiten in der zum Antrieb eines Gelenks verwendeten Hardware auszugleichen. Wenn das Spiel zu einem Gelenk hinzugefügt wird und Sie Schrittmotoren verwenden, muss STEPGEN_MAXACCEL auf das 1,5- bis 2-fache der MAX_ACCELERATION für das Gelenk erhöht werden. Ein übermäßiger Spielausgleich kann dazu führen, dass ein Gelenk bei Richtungsänderungen ruckelt. Wenn ein COMP_FILE für ein Gelenk angegeben ist, wird BACKLASH nicht verwendet.
- *COMP_FILE = file.extension* - Die Kompensationsdatei besteht aus einer Abbildung der Positionsinformationen für das Gelenk. Die Werte der Kompensationsdatei sind in Maschineneinheiten angegeben. Jeder Satz von Werten steht in einer Zeile, getrennt durch ein Leerzeichen. Der erste Wert ist der Nennwert (die befohlene Position). Der zweite und dritte Wert hängt von der Einstellung von COMP_FILE_TYPE ab. Punkte, die zwischen den Nennwerten liegen, werden zwischen den beiden Nennwerten interpoliert. Die Kompensationsdateien müssen mit dem kleinsten Sollwert beginnen und in aufsteigender Reihenfolge bis zum größten Wert der Sollwerte geführt werden. Dateinamen sind Groß-und Kleinschreibung und können Buchstaben und / oder Zahlen enthalten. Derzeit ist die Grenze innerhalb LinuxCNC für 256 Triplets pro Gelenk.

  Wenn COMP_FILE für eine Verbindung angegeben ist, wird BACKLASH nicht verwendet.
- *COMP_FILE_TYPE = 0 oder 1* - Bestimmt den Typ der Kompensationsdatei. Der erste Wert ist die nominale (befohlene) Position für beide Typen.
  Ein *COMP_FILE_TYPE* muss für jede *COMP_FILE* angegeben werden.

  - *Typ 0:* Der zweite Wert gibt die Ist-Position bei Bewegung des Gelenks in positiver Richtung an (steigender Wert) und 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* - Die Mindestgrenze für die Gelenkbewegung in Maschineneinheiten. Wenn diese Grenze erreicht ist, bricht die Steuerung die Gelenkbewegung ab. Für ein Drehgelenk mit unbegrenzter Drehung, für das kein MIN_LIMIT im Abschnitt [JOINT_N] angegeben ist, wird der Wert -1e99 verwendet.

- *MAX_LIMIT = 1000* - Die maximale Grenze für die Gelenkbewegung in Maschineneinheiten. Wenn diese Grenze erreicht ist, bricht die Steuerung die Bewegung des Gelenks ab. Für ein Drehgelenk mit unbegrenzter Drehung, für das im Abschnitt [JOINT_N] kein MAX_LIMIT angegeben ist, wird der Wert 1e99 verwendet.

Anmerkung
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.

Anmerkung
Die Einstellungen [JOINT_N]MIN_LIMIT, MAX_LIMIT werden beim Joggen im Gelenkmodus vor der Ausführung der Referenzfahrt erzwungen. Nach der Referenzfahrt werden die [AXIS_L]MIN_LIMIT,MAX_LIMIT-Koordinatengrenzen 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 auch die zugehörige Github-Fehlermeldung (engl. issue) #97.

- *MIN_FERROR = 0.010* - Dies ist der Wert in Maschineneinheiten, um den das Gelenk bei sehr niedrigen Geschwindigkeiten von der befohlenen Position abweichen darf. Wenn MIN_FERROR kleiner als FERROR ist, erzeugen die beiden eine Rampe von Fehlerauslösepunkten. Man kann sich dies wie ein Diagramm vorstellen, bei dem eine Dimension die Geschwindigkeit und die andere die zulässige Schleppabweichung ist. Mit zunehmender Geschwindigkeit steigt auch der Betrag des Schleppfehlers in Richtung des FERROR-Wertes.
- *FERROR = 1.0* - FERROR ist der maximal zulässige Schleppfehler in Maschineneinheiten. Übersteigt die Differenz zwischen Soll- und Ist-Position diesen Wert, deaktiviert der Regler die Servoberechnungen, setzt alle Ausgänge auf 0.0 und schaltet die Verstärker ab. Wenn MIN_FERROR in der .ini-Datei vorhanden ist, werden geschwindigkeitsproportionale Schleppfehler verwendet. In diesem Fall ist der maximal zulässige Schleppfehler proportional zur Geschwindigkeit, wobei FERROR für die mit [TRAJ]MAX_VELOCITY eingestellte Geschwindigkeit gilt und die Schleppfehler für langsamere Geschwindigkeiten proportional kleiner sind. Der maximal zulässige Schleppfehler wird immer größer sein als MIN_FERROR. Dadurch wird verhindert, dass kleine Schleppfehler bei stillstehenden Achsen die Bewegung ungewollt abbrechen. Kleine Schleppfehler werden aufgrund von Vibrationen usw. immer vorhanden sein.
- *LOCKING_INDEXER = 1* - Gibt an, dass das Gelenk als verriegelnder Indexer verwendet wird.

Diese Parameter beziehen sich auf die Ausführung der Referenzfahrt (engl. homing), für eine bessere Erklärung lesen Sie das Referenzfahrt-Konfiguration Kapitel.

- *HOME = 0.0* - Die Position, die das Gelenk nach Abschluss der Referenzfahrt anfahren wird.
- *HOME_OFFSET = 0.0* - Die gemeinsame Position des Referenzschalters oder Indeximpulses, in <sub:ini:sec:traj,Maschineneinheiten>>. Wenn der Referenzpunkt während der Referenzfahrt gefunden wird, ist dies die Position, die diesem Punkt zugewiesen wird. Bei gemeinsamer Nutzung von Referenzfahrt- und Endschaltern und bei Verwendung einer Referenzfahrt-Sequenz, die den Referenzfahrt-/Endschalter im umgeschalteten Zustand belässt, kann der Referenzfahrt-Offset verwendet werden, um die Position des Referenzfahrtschalters auf einen anderen Wert als 0 festzulegen, wenn die Referenzfahrtposition 0 sein soll.

- *HOME_SEARCH_VEL = 0.0* - Anfängliche Referenzfahrtgeschwindigkeit in Maschineneinheiten pro Sekunde. Das Vorzeichen gibt die Fahrtrichtung an. Ein Wert von Null bedeutet, dass die aktuelle Position als Ausgangsposition für die Maschine angenommen wird. Wenn Ihre Maschine keine Home-Schalter hat, sollten Sie diesen Wert auf Null belassen.
- *HOME_LATCH_VEL = 0.0* - Referenzfahrtgeschwindigkeit in Maschineneinheiten pro Sekunde zur Endschalter-Auslöse-Position. Das Vorzeichen gibt die Fahrtrichtung an.
- *HOME_FINAL_VEL = 0.0* - Geschwindigkeit in Maschineneinheiten pro Sekunde von der Home-Latch-Position zur Home-Position. Wird der Wert 0 belassen oder ist er nicht im Gelenk enthalten, wird die Eilgeschwindigkeit verwendet. Muss eine positive Zahl sein.
- *HOME_USE_INDEX = NO* - Wenn der für dieses Gelenk verwendete Encoder einen Indeximpuls hat und die Bewegungskarte (engl. motion card) dieses Signal vorsieht, können Sie diese Option auf ja setzen. Wenn dies der Fall ist, hat dies Auswirkungen auf die Art des verwendeten Referenzfahrtmusters. Gegenwärtig können Sie mit Schrittmotoren keine Referenzfahrt mit Index durchführen, es sei denn, Sie verwenden StepGen im Geschwindigkeitsmodus und PID.
- *HOME_INDEX_NO_ENCODER_RESET = NO* - Verwenden Sie YES, wenn der Encoder, der für dieses Gelenk verwendet wird, seinen Zähler nicht zurücksetzt, wenn ein Indeximpuls nach der Aktivierung des Gelenk index_enable HAL-Pins erkannt wird. Gilt nur für HOME_USE_INDEX = YES.
- *HOME_IGNORE_LIMITS = NO* - Wenn Sie den Endschalter als Referenzfahrtschalter und den Endschalter verwenden, sollte dies auf YES gesetzt werden. Wenn diese Einstellung auf YES gesetzt ist, wird der Endschalter für dieses Gelenk bei der Referenzfahrt ignoriert. Sie müssen Ihre Referenzfahrt so konfigurieren, dass sich der Referenzfahrt-/Endschalter am Ende der Referenzfahrt nicht im umgeschalteten Zustand befindet, da Sie sonst nach der Referenzfahrt einen Endschalterfehler erhalten.
- *HOME_IS_SHARED = <n>* - Wenn der Home-Eingang von mehr als einem Gelenk geteilt wird, setzen Sie <n> auf 1, um zu verhindern, dass die Referenzfahrt gestartet wird, wenn einer der geteilten Schalter bereits geschlossen ist. Setzen Sie <n> auf 0, um eine Referenzfahrt zu ermöglichen, wenn ein Schalter geschlossen ist.
- *HOME_ABSOLUTE_ENCODER = 0 | 1 | 2* - Wird verwendet, um anzuzeigen, dass das Gelenk einen Absolutwertgeber verwendet. Bei einer Anforderung zur Referenzfahrt wird der aktuelle Gelenkwert auf den *HOME_OFFSET* Wert gesetzt. Wenn die *HOME_ABSOLUTE_ENCODER* Einstellung 1 ist, macht die Maschine die übliche Endbewegung zum *HOME* Wert. Wenn die *HOME_ABSOLUTE_ENCOD* Einstellung 2 ist, wird keine solche abschließende Bewegung ausgeführt.
- *HOME_SEQUENCE = <n>* - Definiert die ”Home All” Sequenz, d.h. die Fahrt zum Referenzpunkt für alle Achsen. <n> muss bei 0 oder 1 oder -1 beginnen. Zusätzliche Sequenzen können mit um 1 aufsteigenden Zahlen (in absoluten Werten) angegeben werden. Das Überspringen von Sequenznummern ist nicht erlaubt. Bei Weglassen einer HOME_SEQUENCE wird das Gelenk von der Funktion ”Home All” nicht referenziert. Mehrere Gelenke können gleichzeitig referenziert werden, indem man die gleiche Sequenznummer für mehrere Gelenke angibt. Eine negative Sequenznummer wird verwendet, um die letzte Bewegung für alle Gelenke mit dieser (negativen oder positiven) Sequenznummer zu verschieben. Für weitere Informationen siehe: HOME SEQUENCE.
- *VOLATILE_HOME = 0* - Wenn aktiviert (auf 1 gesetzt), wird das Gelenk bei ausgeschalteter Maschinenstromversorgung oder bei eingeschaltetem Not-Halt nicht geortet. Dies ist nützlich, wenn Ihre Maschine über Home-Schalter verfügt und keine Positionsrückmeldung hat, wie z. B. bei einer schritt- und richtungsgesteuerten Maschine.

Diese Parameter sind relevant für Gelenke, die von Servos gesteuert werden.

Warnung
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 <sub:ini:custom,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’ - Wie nah ist nah genug, um den Motor als in Position zu betrachten, 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/sec, 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 spiegeln 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.

Tabelle 4.1: 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:

$$input\,scale = 2000\frac{counts}{rev} * 10\frac{rev}{inch} = 20000\frac{counts}{inch}$$

Diese Parameter sind relevant für Gelenke, die von Schrittmotoren gesteuert werden.

Warnung
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 <sub:ini:custom,Benutzerdefiniert Abschnitte und Variablen>>.

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:

$$input\,scale = \frac{2\,steps}{1.8\,degrees} * 360\,\frac{degree}{rev} * 10\,\frac{rev}{inch} = 4000\,\frac{steps}{inch}$$

Anmerkung
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.

4.4.2.14 [SPINDLE_<num>] Abschnitt(e)

Die <num> gibt die Spindelnummer 0 … (num_spindles-1) an. Der Wert von *num_spindles* wird gesetzt durch [TRAJ]SPINDLES=

- *MAX_VELOCITY = 20000* Die maximale Spindeldrehzahl (in U/min) für die angegebene Spindel. Optional.
- *MIN_VELOCITY = 3000* Die minimale Spindeldrehzahl (in U/min) für die angegebene Spindel. Optional. Viele Spindeln haben eine Mindestdrehzahl, unter der sie nicht betrieben werden sollten. Jeder Spindeldrehzahlbefehl, der unter diesem Grenzwert liegt, wird auf diesen Grenzwert /erhöht/.
- *MAX_REVERSE_VELOCITY = 20000* Diese Einstellung wird standardmäßig auf MAX_VELOCITY gesetzt, wenn sie weggelassen wird. Sie kann in Fällen verwendet werden, in denen die Spindeldrehzahl im Rückwärtsgang begrenzt ist. Für Spindeln, die nicht im Rückwärtsgang laufen dürfen, wird sie auf Null gesetzt. In diesem Zusammenhang bezieht sich "max" auf die absolute Größe der Spindeldrehzahl.

- *MIN_REVERSE_VELOCITY = 3000* Diese Einstellung entspricht MIN_VELOCITY, jedoch für die umgekehrte Spindeldrehung. Ist dieser Wert nicht angegeben, wird sie standardmäßig auf MIN_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 grafische Benutzeroberflächen die Dinge anders handhaben können.
- *HOME_SEARCH_VELOCITY = 100* - FIXME: Spindel-Referenzfahrt funktioniert noch nicht Setzt die Referenzfahrtgeschwindigkeit (U/min) für die Spindel. Die Spindel dreht sich während der Referenzfahrt mit dieser Geschwindigkeit, bis der Spindelindex gefunden ist. Dann wird die Spindelposition auf Null gesetzt. Beachten Sie, dass es keinen Sinn macht, wenn die Spindel-Ausgangsposition einen anderen Wert als Null hat, daher ist dies auch nicht vorgesehen.
- *HOME_SEQUENCE = 0* - FIXME: Spindel-Referenzfahrt funktioniert noch nicht Steuert, wo in der allgemeinen Referenzfahrt-Sequenz die Spindel-Referenzfahrt-Drehungen stattfinden. Setzen Sie HOME_SEARCH_VELOCITY auf Null, um Spindeldrehungen während der Referenzfahrt zu vermeiden.

4.4.2.15 [EMCIO] Abschnitt

- *EMCIO = io* - Name des E/A (engl. I/O)-Controller-Programms.
- *CYCLE_TIME = 0.100* - Die Zeitspanne in Sekunden, in der EMCIO läuft. Ein Wert von 0,0 oder eine negative Zahl bedeutet, dass EMCIO überhaupt nicht schlafen soll. Normalerweise besteht keine Notwendigkeit, diese Zahl zu ändern.
- *TOOL_TABLE = tool.tbl* - Die Datei mit den Werkzeuginformationen, die im Benutzerhandbuch beschrieben sind.
- *DB_PROGRAM = db_program* - Pfad zu einem ausführbaren Programm, das Werkzeugdaten verwaltet. (Wenn ein DB_PROGRAM angegeben ist, wird ein TOOL_TABLE-Eintrag ignoriert)
- *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* - Die Spindel bleibt während des Werkzeugwechsels eingeschaltet, wenn der Wert 1 ist. Nützlich für Drehmaschinen oder Maschinen, bei denen sich das Material in der Spindel und nicht im Werkzeug befindet.
- *TOOL_CHANGE_QUILL_UP = 1* - Die Z-Achse wird vor dem Werkzeugwechsel auf den Maschinennullpunkt gefahren, wenn der Wert 1 ist. Dies ist dasselbe wie die Ausführung eines G0 G53 Z0.
- *TOOL_CHANGE_AT_G30 = 1* - Die Maschine wird auf den durch die Parameter 5181-5186 für G30 definierten Referenzpunkt gefahren, wenn der Wert 1 ist. Weitere Informationen finden Sie unter G-Code Parameter und G-Code G30-G30.1.
- *RANDOM_TOOLCHANGER = 1* - Dies ist für Maschinen, die das Werkzeug nicht in die Tasche zurücklegen können, aus der es stammt. Zum Beispiel Maschinen, die das Werkzeug in der aktiven Tasche mit dem Werkzeug in der Spindel austauschen.

4.5 Konfiguration der Referenzfahrt (engl. homing)

4.5.1 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.

Anmerkung
Es ist zwar möglich, LinuxCNC ohne Referenzschalter/Referenzfahrt oder Endschalter zu verwenden, aber die zusätzliche Sicherheit der Softlimits wird dadurch zunichte gemacht.

4.5.2 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.

Anmerkung
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.

4.5.3 Separater Home-Schalter Beispiel-Layout

Dieses Beispiel zeigt minimale und maximale Endschalter mit einem separaten Home-Schalter.

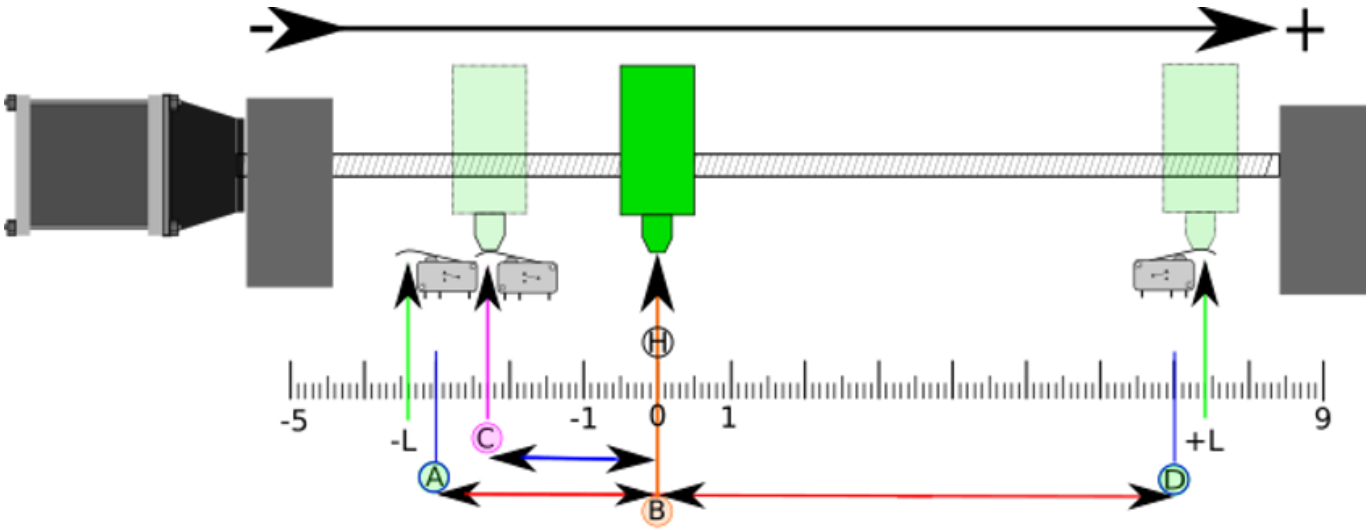


Abbildung 4.6: Demonstratives separates Schalterlayout

- 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.

Anmerkung
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.

4.5.4 Gemeinsamer End-/Hauptschalter Beispiel-Layout

Dieses Beispiel zeigt einen maximalen Endschalter und einen kombinierten minimalen End-/Referenzschalte

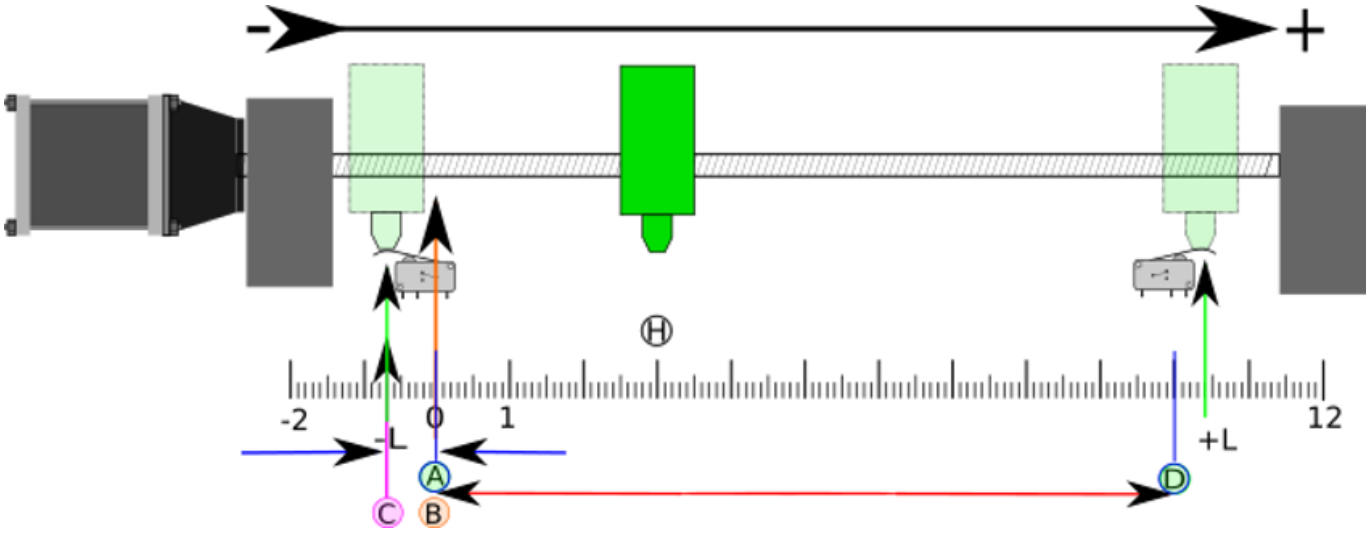


Abbildung 4.7: Beispiel für ein Layout mit geteilten Schaltern

- 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.

4.5.5 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.

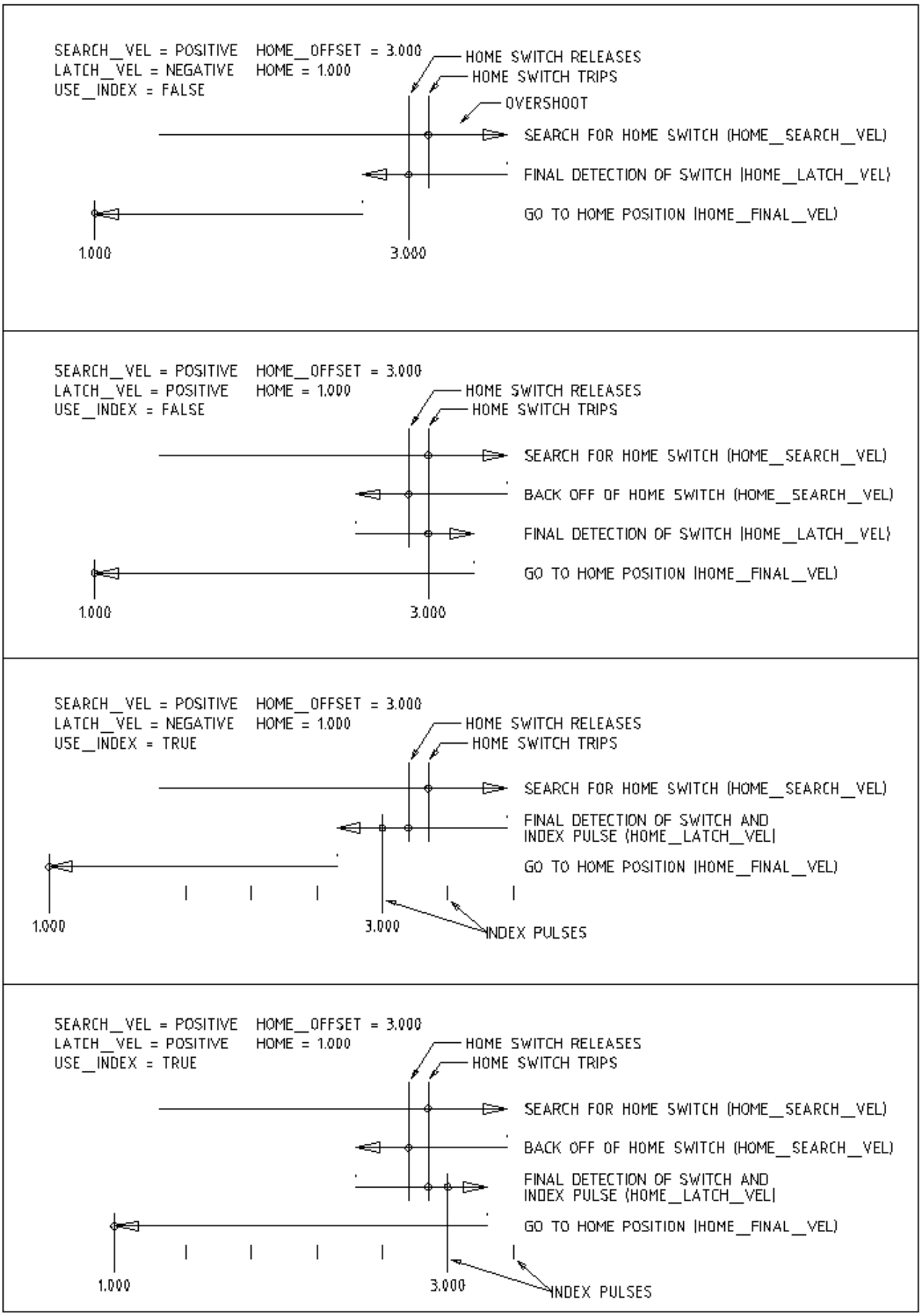


Abbildung 4.8: Referenzfahrt-Abläufe

4.5.6 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

Anmerkung
Alle anderen Kombinationen können zu einem Fehler führen.

4.5.6.1 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.

4.5.6.2 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.

4.5.6.3 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.

4.5.6.4 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.

4.5.6.5 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.

Anmerkung
HOME_USE_INDEX erfordert Verbindungen in Ihrer HAL-Datei zu joint.n.index-enable vom encoder.n.index-enable.

4.5.6.6 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.

4.5.6.7 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.

Anmerkung
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.

4.5.6.8 Referenzpunkt (engl. 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.

Anmerkung
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.

4.5.6.9 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.

4.5.6.10 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_ENCOI* 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
```

Anmerkung
Eine HOME_IS_SHARED-Einstellung wird stillschweigend ignoriert.

Anmerkung
Eine Aufforderung, für ein Gelenk die Referenzfahrt zu wiederholen, wird stillschweigend ignoriert.

4.5.6.11 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.

Anmerkung
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
```

4.5.6.12 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.

4.5.6.13 LOCKING_INDEXER

Handelt es sich bei diesem Gelenk um einen verriegelnden Drehindexer, wird es vor der Referenzfahrt entriegelt und danach verriegelt.

4.5.6.14 Unmittelbare Referenzfahrt (engl. immediate homing)

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 Achsen-GUI gedrückt wird, dann sind die folgenden INI-Einträge für dieses Gelenk erforderlich.

1. HOME_SEARCH_VEL = 0
2. HOME_LATCH_VEL = 0
3. HOME_USE_INDEX = NO
4. HOME ist gleich HOME_OFFSET
5. HOME_SEQUENCE = 0 (oder andere gültige Sequenznummer)

Anmerkung
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).

4.5.6.15 Vermeiden einer Referenzfahrt

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_float_s32 names=heimat_sequenz_s32
setp home_sequenz_mux.in0 -1
setp home_sequenz_mux.in1 1
addf home_sequence_mux servo-thread
addf home_sequence_s32 servo-thread
...
net home_seq_float <= home_sequence_mux.out
net home_seq_float => home_sequence_s32.in
net home_seq_s32 <= home_sequence_s32.out
net home_seq_s32 => ini.0.home_sequence
net home_seq_s32 => ini.1.home_sequence
...
# allow_jjog: von einem virtuellen Bedienfeld oder Hardware-Schalter erzeugter Pin
net hsequence_select <= allow_jog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

Anmerkung
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.

Anmerkung
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/.

4.6 I/O Control V2

4.6.1 Beschreibung

Die E/A-Steuerung übernimmt E/A-Aufgaben wie Kühlmittel, Werkzeugwechsel, Notaus (engl. E-stop) und Schmiermittel. Die Signale werden im Userspace mit G-Code ein- und ausgeschaltet oder im Falle des Notaus in HAL.

I/O Control V2 bietet mehr Unterstützung für die Kommunikation mit dem Werkzeugwechsler.

- Von LinuxCNC verursachter Abbruch und Werkzeugwechslerfehler: iocontrol bricht eine laufende Wechseloperation zuverlässig ab (Werkzeugwechsel bestätigt). Ein Werkzeugwechsler kann jederzeit einen Fehler melden, der zum Abbruch des nächsten M6 führt. Wenn beispielsweise ein Werkzeugwechsler während eines Vorbereitungsvorgangs eine leere Tasche vorfindet, sollte er iocontrol einen Fehler melden können, und iocontrol sollte entsprechend reagieren, wenn der M6-Wechselvorgang ausgeführt wird.
- Abbruch-/Fehlerursache kommunizieren: Lassen Sie iocontrol wissen, warum der Werkzeugwechsler einen Fehler verursacht hat und warum iocontrol abgebrochen hat. Dies ist für UI-Zwecke. Es wäre ein Kandidat für einen #5xxx Parameter und eine selektive Anzeige in der Benutzeroberfläche.

- Keine Race Conditions zwischen iocontrol und Werkzeugwechsler: Das Protokoll zwischen iocontrol und Werkzeugwechsler muss eindeutig sein, welche Operation signalisiert wird und ob eine Änderungsoperation abgebrochen oder abgeschlossen wird.
- Konsistente Ansicht des Status: Beide Parteien müssen zu jedem Zeitpunkt eine einheitliche Sicht auf den Stand der Dinge in Bezug auf abgebrochene und abgeschlossene Vorgänge sowie auf die Anzahl der Werkzeuge und deren Platz haben.
- Handshaked Signalisierung eines Abbruchs/Fehlers: Nach der Signalisierung eines Abbruchs von LinuxCNC an den Werkzeugwechsler, oder bei einem Fehler den der Werkzeugwechsler angezeigt, wird ein Handshake erwartet, um zuverlässige Signalisierung zu gewährleisten, und optional lockstep Verhalten zu erzwingen. Handshaking ist optional und kann in HAL überbrückt werden, wenn es nicht benötigt wird.
- Rückwärtskompatibilität: Ein Werkzeugwechsler, der die iocontrol emc-abort-Zeile ignoriert und an der alten Handhabung festhält, wird ”weiterhin funktionieren” (vorbehaltlich einer Race Condition)

Wenn Sie strenge Zeitvorgaben haben oder einfach mehr E/A benötigen, sollten Sie stattdessen die Echtzeit-E/A verwenden, die von motion bereitgestellt wird.

4.6.2 Anwendung

INI-Datei Optionen:

[EMCIO] Abschnitt

PROTOCOL_VERSION = 2
Der Standardwert ist 2. Mit der Einstellung 1 wird das alte iocontrol-Verhalten emuliert.

EMCIO = iov2 -support-start-change
Sie müssen das Start-Change-Protokoll explizit aktivieren, indem Sie die -support-start-change-Option hinzufügen; andernfalls bleibt der start-change-Pin low und start-change-ack wird ignoriert. Der Grund dafür ist eine besser Abwärtskompatibilität.

[TASK] Abschnitt

IO_ERROR
Printf-angepasste Vorlage für die Anzeige von Bedienerfehlern (negative Werkzeugwechsler-Fehlercode
Keine Anführungszeichen erforderlich. Beispiel: `IO_ERROR = Wechslerfehler %d`. Standard: Werkzeugwechsel Fehler %d.

[EMC] Abschnitt

DEBUG
Um eine (ziemlich detaillierte) Spur zu erhalten, setzen Sie entweder das RCS-Debugging-Flag (0x00000200), um RCS-Debugging-Meldungen in ganz LinuxCNC einzuschalten, oder verwenden Sie das neue iocontrol-Debugging-Bit (0x00001000) nur für iov2-Meldungen.

4.6.3 Pins

- iocontrol.0.coolant-flood (Bit, Out) TRUE wenn Kühlmittelflut angefordert wird
- iocontrol.0.coolant-mist (Bit, Out) TRUE wenn Kühlmittelnebel angefordert wird
- *iocontrol.0.emc-enable-in* - (Bit, in) Sollte FALSE sein, wenn eine externe Not-Aus-Bedingung vorliegt.

- iocontrol.0.lube (Bit, Out) TRUE wenn Schmiermittel angefordert wird. Dieser Pin wird mit TRUE angesteuert, wenn die Steuerung aus dem Not-Aus-Zustand kommt und der Befehl ”Lube On” an die Steuerung gesendet wird. Er wird mit FALSE angesteuert, wenn der Regler in den Notaus-Zustand geht und wenn der Befehl ”Lube Off” an den Regler gesendet wird.
- iocontrol.0.lube_level (Bit, In) Sollte auf FALSE gesetzt werden, wenn der Schmierstofftank leer ist.
- iocontrol.0.tool-change (Bit, Out) TRUE wenn ein Werkzeugwechsel angefordert wird
- iocontrol.0.tool-changed (Bit, In) Sollte auf TRUE gesetzt werden, wenn ein Werkzeugwechsel abgeschlossen ist.
- iocontrol.0.tool-number (s32, Out) Aktuelle Werkzeugnummer
- iocontrol.0.tool-prep-number (s32, Out) Die Nummer des nächsten Werkzeugs, aus dem RS274NGC T-Wort
- iocontrol.0.tool-prep-pocket (s32, Out) Dies ist die Platznummer (Speicherplatz im Werkzeugspeicher) des Werkzeugs, das durch das letzte T-Wort angefordert wurde.
- iocontrol.0.tool-prepare (Bit, Out) TRUE, wenn eine Tn-Werkzeugvorbereitung angefordert wird
- iocontrol.0.tool-prepared (Bit, In) Sollte auf TRUE gesetzt werden, wenn eine Werkzeugvorbereitung abgeschlossen ist.
- iocontrol.0.user-enable-out (Bit, Out) FALSE, wenn eine interne Estop-Bedingung vorliegt
- iocontrol.0.user-request-enable (Bit, Out) TRUE, wenn der Benutzer die Aufhebung der Sperre beantragt hat

Zusätzliche Pins hinzugefügt durch I/O Control V2

- emc-abort: (Bit, Out) signalisiert emc-verursachten Abbruch an Werkzeugwechsler.
- emc-abort-ack: (Bit, In) Bestätigung vom Werkzeugwechsler für das vorherige Signal, oder gejumpert auf abort-tool-change, wenn nicht im Werkzeugwechsler verwendet. Hinweis: Nach der Signalisierung eines emc-abort blockiert iov2 bis emc-abort-ack ausgelöst wird.
- emc-reason: (S32, Out) Übermittelt die Ursache für den EMC-verursachten Abbruch an den Werkzeugwechsler. Verwendung: UI informational. Gültig während emc-abort TRUE.
- start-change: (Bit, Out) wird zu Beginn einer M6-Operation ausgelöst, bevor eine Spindel-Aus-, Quill-Up- oder Move-to-toolchange-Position-Operation ausgeführt wird.
- start-change-ack: (Bit, In) Quittungsleitung (engl. acknowledgement line) für start-change.
- Werkzeugwechsler-Störung: (Bit, In) Werkzeugwechsler meldet Störung. Diese Leitung wird laufend überwacht. Ein Fehler schaltet ein Flag in iocontrol um, das sich in dem fehlerhaften Pin des Werkzeugwechslers widerspiegelt.
- toolchanger-fault-ack: (Bit, Out) Handshake-Leitung für obiges Signal. wird von iov2 gesetzt, nachdem die obige Fehlerleitung TRUE erkannt und abgewertet wurd, wenn toolchanger-fault fällt. Es steht dem Werkzeugwechsler frei, das Ack zu interpretieren; das Lesen der -ack-Leitungen stellt sicher, dass der Fehler empfangen wurde und darauf reagiert wird.
- toolchanger-reason: (S32, In) Übermittlung des Grundcodes für den vom Toolchanger verursachten Fehler an iov2. Verwendung: Signal, ob das Programm fortgesetzt oder abgebrochen werden soll, plus UI-Informationen, falls negativ. Wird während des Werkzeugwechsler-Fehlers gelesen TRUE. Werte ungleich Null führen zu einer Meldung des Achsenbedieners oder einer Fehlermeldung, siehe unten.
- toolchanger-faulted: (Bit, Out) signalisiert, dass die toolchanger-notify-Leitung umgeschaltet hat und der toolchanger-reason-code im Fehlerbereich lag. Der nächste M6 bricht ab.
- toolchanger-clear-fault: (Bit, In) setzt TC-Fehlerzustand zurück. Setzt toolchanger-faulted zurück, wenn toolchanger-notify auf FALSE steht. Verwendung: UI - z.B. Schaltfläche Fehlerzustand löschen.

4.6.4 Parameter

- iocontrol.0.tool-prep-index (s32, RO) IO's interner Array-Index des vorbereiteten Werkzeugs, das durch das letzte T-Wort angefordert wurde. 0, wenn kein Werkzeug vorgerüstet ist. Bei Maschinen mit Zufallswerkzeugwechsler ist dies die Platznummer des Werkzeugs (d.h. dieselbe wie der Tool-Prep-Pocket-Pin), bei Maschinen ohne Zufallswerkzeugwechsler ist dies eine kleine Ganzzahl, die der Position des Werkzeugs in der internen Darstellung der Werkzeugtabelle entspricht. Dieser Parameter kehrt nach einem erfolgreichen Werkzeugwechsel M6 auf 0 zurück.

4.6.5 Kommunikation

Wenn LinuxCNC signalisiert einen Abbruch aus irgendeinem Grund, wird dies in der emc-abort und emc-reason Pins reflektiert. Es wird erwartet, dass der Werkzeugwechsler den emc-abort-Pin bestätigt, indem er den emc-abort-ack-Pin anhebt - iov2 wird blockieren, bis dies geschehen ist. Wenn Sie die Abort-Handshake-Funktion nicht benötigen, jumpen Sie sie wie folgt:

```
net emc-abort-ack iocontrol.0.emc-abort iocontrol.0.emc-abort-ack
```

Der emc-reason-Pin wird als gültig angesehen, wenn emc-abort TRUE ist.

Die Auslöser-Codes sind wie folgt für LinuxCNC intern generiert Abbrüche (siehe emc.hh ca Zeile 321):

- 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_USER = 100

iov2 fügt einen weiteren Code hinzu, nämlich EMC_ABORT_BY_TOOLCHANGER_FAULT = 101, der signalisiert wird, wenn ein M6 abbricht, weil der Pin "toolchanger-faulted" TRUE ist.

Um Werkzeugwechsler-Fehler an LinuxCNC zu signalisieren, verdrahten Sie den Toolchanger-Fault-Pin und optional die Toolchanger-Reason- und Toolchanger-Ack-Pins.

Der Toolchanger-Fault löst den Fehlerzustand aus, der sich im Toolchanger-Faulted-Pin widerspiegelt. Dieser Zustand kann durch Aktivierung des Pins "toolchanger-clear-fault" gelöscht werden, sofern der Pin "toolchanger-fault" FALSE ist.

Der Wert des toolchanger-reason Pins wird wie folgt verwendet:

- toolchanger-reason > 0 : Der Werkzeugwechsel wird nicht abgeschlossen und das Programm wird fortgesetzt, jedoch wird der Parameter #5060 auf 1,0 gesetzt, um den Fehler anzuzeigen. Der Parameter #5601 enthält den Wert des Werkzeugwechslergrundes.
  - toolchanger-reason = 0 : das Programm wird abgebrochen
  - toolchanger-reason < 0 : Das Programm wird abgebrochen und eine Fehlermeldung wird mit Hilfe der Vorlage IO_ERROR angezeigt.

Die Verwendung des Pins "toolchanger-fault-ack" ist optional. Er wird TRUE, wenn toolchanger-fault ausgelöst wird und der toolchanger-reason-Pin von iov2 gelesen wurde.

4.7 Konfiguration der Drehmaschine

4.7.1 Standard-Ebene

Wenn LinuxCNC's Interpreter wurde zuerst geschrieben, war es für Mühlen konzipiert. Das ist, warum die Standard-Ebene ist XY (G17). Eine normale Drehmaschine verwendet nur 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 <cha:gmoccapy,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 <sec:latency-test,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					
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.

Beispiel 4.1 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:

$$\frac{200 motor steps}{1 motor rev} \times \frac{10 microsteps}{1 motor step} \times \frac{2 motor revs}{1 leadscrew rev} \times \frac{1 leadscrew revs}{0.2000 inch} = \frac{20{,}000 microsteps}{inch}$$

Beispiel 4.2 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 LinuxCNC war, dass man die Pinbelegung ohne Neukompilierung des Quellcodes angeben konnte’t. LinuxCNC ist viel flexibler, und jetzt (dank der Hardware Abstraction Layer) können Sie leicht angeben, welches Signal geht wo. Siehe die <cha:basic-hal-reference,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.

Anmerkung
Wir stellen nur eine Achse vor, um sie kurz zu halten, 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.

4.9.3.1 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.

4.9.3.2 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 Fußnote:[der schnellste Thread im LinuxCNC-Setup, normalerweise wird der Code alle paar zehn Mikrosekunden ausgeführt].
- 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.

4.9.3.3 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.

4.9.3.4 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-invert TRUE
```

4.9.3.5 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.

4.9.3.6 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.

4.9.3.7 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 Stepper-Diagnose

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

4.10.1.1 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.

4.10.1.2 Keine Stepper bewegen sich

Viele Laufwerke haben einen Freigabe-Pin oder benötigen eine Ladungspumpe, um den Ausgang zu aktivieren.

4.10.1.3 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

4.10.2.1 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.

4.10.2.2 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.

Anmerkung
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

4.10.3.1 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
```

Kapitel 5

HAL (Hardware Abstraction Layer)

5.1 HAL-Einführung

HAL steht 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. Der ”Hardware”-Teil kommt daher, dass HAL ursprünglich entwickelt wurde, um die Konfiguration von LinuxCNC für eine Vielzahl von Hardware-Geräten zu erleichtern. Viele der Bausteine sind Treiber für Hardware-Geräte. HAL kann jedoch mehr, als nur Hardware-Treiber zu konfigurieren.

5.1.1 HAL-Systementwurf

HAL basiert auf traditionellen Systementwurfstechniken.

HAL basiert auf denselben Prinzipien, die auch für die Entwicklung von Hardware-Schaltungen und -Systemen verwendet werden, so dass es sinnvoll ist, zunächst diese Prinzipien zu untersuchen.

Jedes System, auch eine CNC-Maschine, besteht aus miteinander verknüpften 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 für den Betrieb eines Werkzeugwechslers usw. sein. Der Maschinenbauer muss diese Teile auswählen, montieren und miteinander verdrahten, um ein komplettes System zu erhalten.

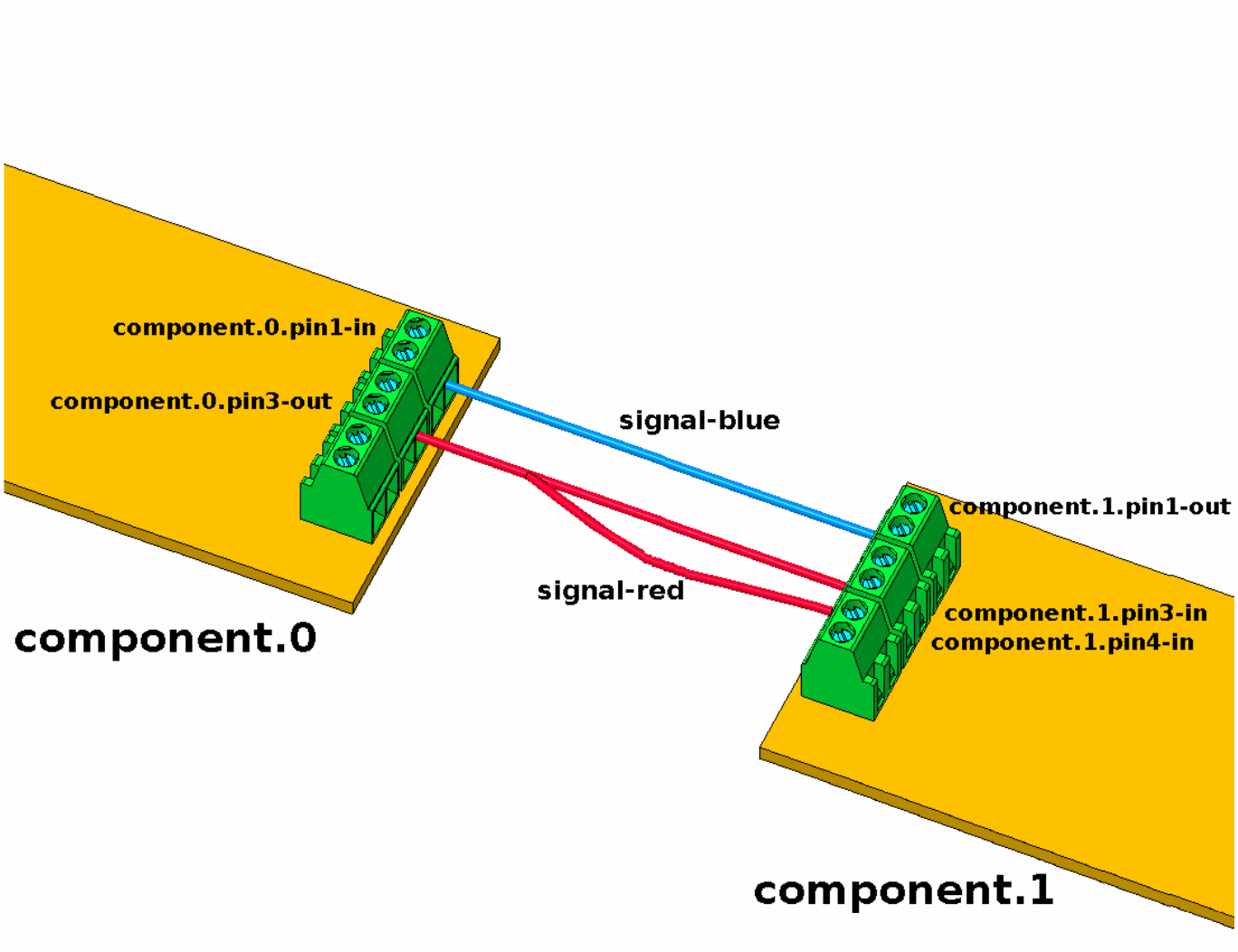


Abbildung 5.1: HAL-Konzept - Verbinden wie elektrische Schaltkreise.

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
```

5.1.1.1 Teileauswahl

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.

5.1.1.2 Verbindungsentwurf

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.

5.1.1.3 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.

5.1.1.4 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.

5.1.1.5 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.

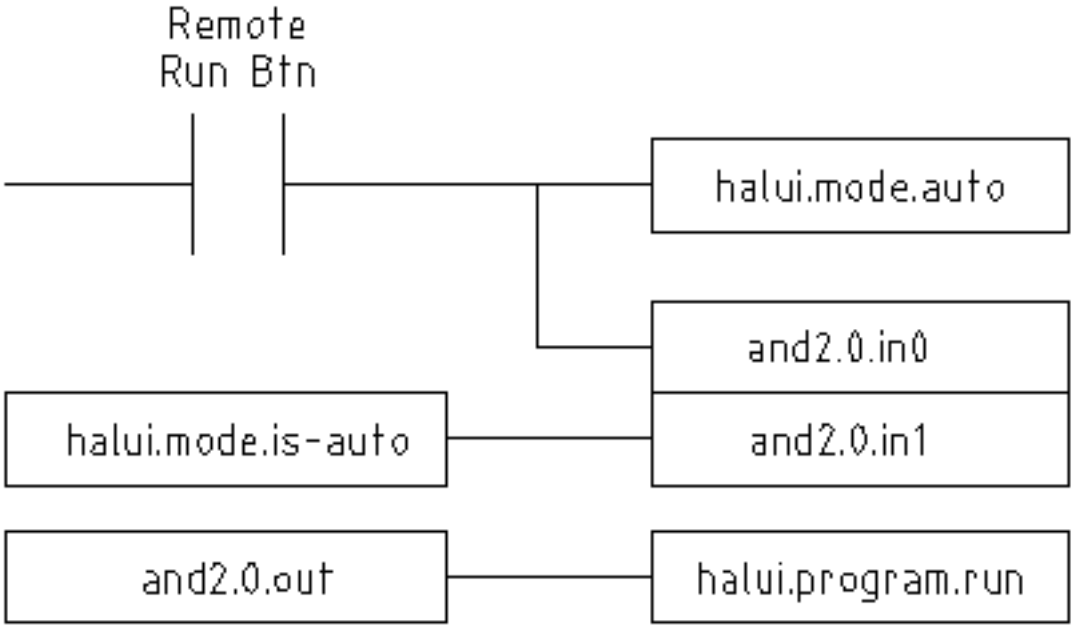


Abbildung 5.2: Remote-Start-Beispiel (Schema)

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.

5.1.2 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 *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 Verhaltensweisen, das installiert und nach Bedarf miteinander verbunden werden kann.

Anmerkung
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 einen Notstopp. 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.

Physikalischer Pin (engl. physical pin)
Viele E/A-Geräte haben echte physische Pins oder Anschlüsse, die mit externer Hardware verbunden sind, z. B. die Pins eines Parallelport-Anschlusses. Um Verwirrung zu vermeiden, werden diese als ”physische Pins” bezeichnet. Dies sind die Dinge, die in die reale Welt *hineinragen*.

Anmerkung
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-V-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:

- bit - ein einzelner TRUE/FALSE- oder ON/OFF-Wert
- float - eine 64-Bit-Fließkommazahl mit einer Auflösung von etwa 53 Bit und einem Dynamikbereich von über 1000 Bit.
- u32 - eine 32-Bit-Ganzzahl ohne Vorzeichen, zulässige Werte sind 0 bis 4.294.967.295
- s32 - eine 32-Bit-Ganzzahl mit Vorzeichen, zulässige Werte sind -2.147.483.647 bis +2.147.483.647

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. Es können Funktionen 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.

5.1.3 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.

5.1.4 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. 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 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.

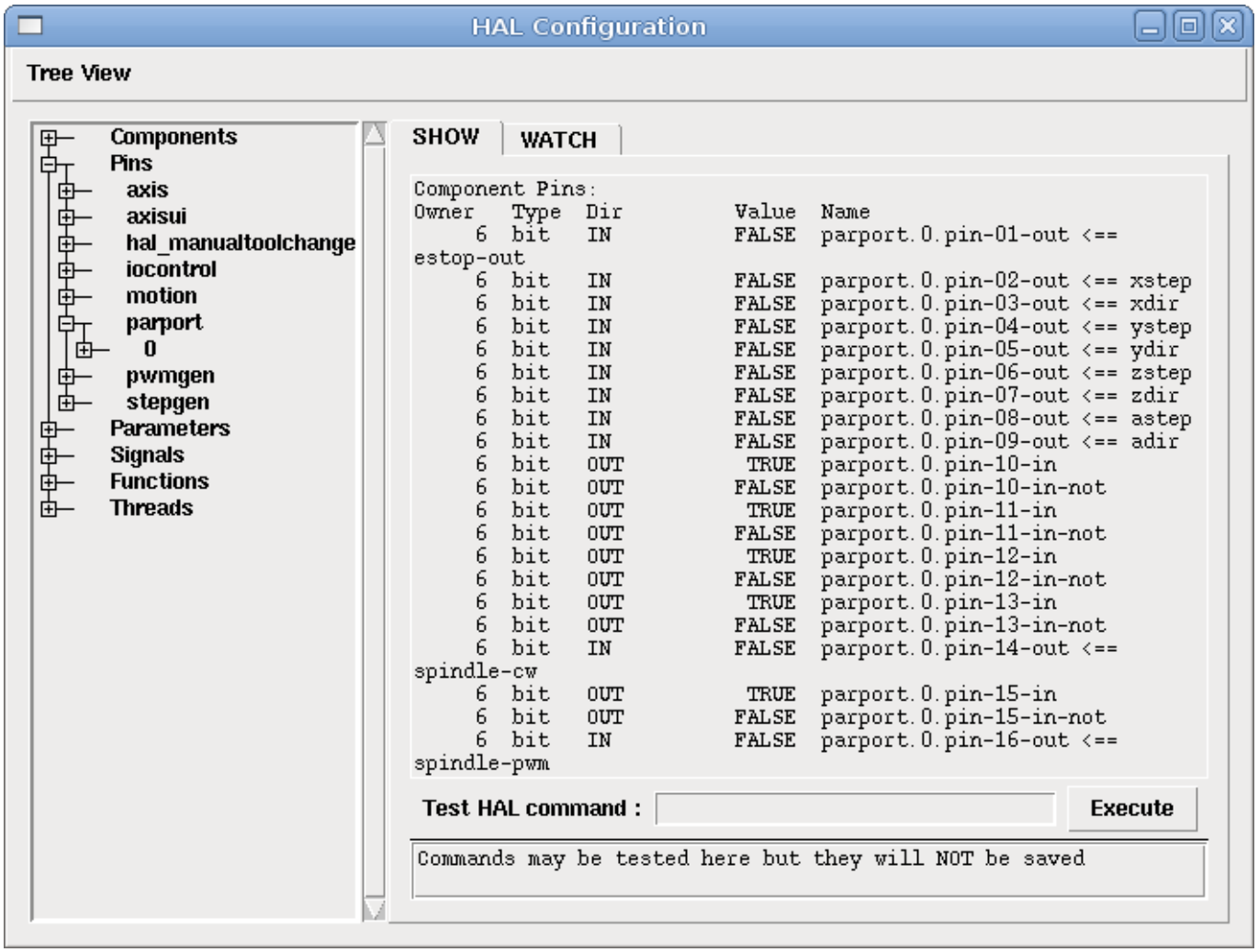


Abbildung 5.3: HAL-Konfigurationsfenster

5.2.1.1 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 Userspace-Komponente in den Echtzeitbereich laden.

loadrt Syntax und Beispiel

```
loadrt <component> <options>
loadrt mux4 count=1
```

5.2.1.2 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 *functname* zum 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* bedeutet Position in Bezug auf das Ende des Threads. Beispiel: *1* ist der Anfang des Fadens, *-1* ist das Ende des Fadens, *-3* ist die drittletzte Position.

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  NO       0   or2.0
```

Sie müssen eine Funktion aus einer HAL-Echtzeitkomponente zu einem Thread hinzufügen, damit die Funktion mit der Rate des Threads aktualisiert wird.

Normalerweise gibt es zwei Threads, wie in diesem Beispiel gezeigt. Einige Komponenten verwenden Fließkomma-Mathematik und müssen zu einem Thread hinzugefügt werden, der Fließkomma-Mathematik unterstützt. Das *FP* zeigt an, ob die Fließkomma-Mathematik in diesem Thread unterstützt wird.

```
$ 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  NO            base-thread (        0,        0 )
```

- Basis-Thread (der Hochgeschwindigkeits-Thread): Dieser Thread bearbeitet Aufgaben, die eine schnelle Reaktion erfordern, wie z. B. die Erzeugung von Schrittimpulsen und das Lesen und Schreiben der parallelen Schnittstelle. Er unterstützt keine Fließkomma-Mathematik.
- Servo-Thread (der Slow-Speed-Thread): Dieser Thread verarbeitet Elemente, die eine langsamere Reaktion tolerieren können, wie den Motion-Controller, ClassicLadder und den Motion-Command-Handler, und unterstützt Fließkomma-Mathematik.

addf Syntax und Beispiel

```
addf <function> <thread>
addf mux4.0 servo-thread
```

Anmerkung
Wenn die Komponente einen Fließkomma-Thread benötigt, ist dies normalerweise der langsamere Servo-Thread.

5.2.1.3 loadusr

Mit dem Befehl *loadusr* wird eine User-Space-HAL-Komponente geladen. Userspace-Programme sind eigene Prozesse, die optional über Pins und Parameter mit anderen HAL-Komponenten kommunizieren. Sie können keine Echtzeitkomponenten in den Benutzerraum 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.

loadusr Syntax und Beispiele

```
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*.

5.2.1.4 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.

net Syntax und Beispiel

```
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.

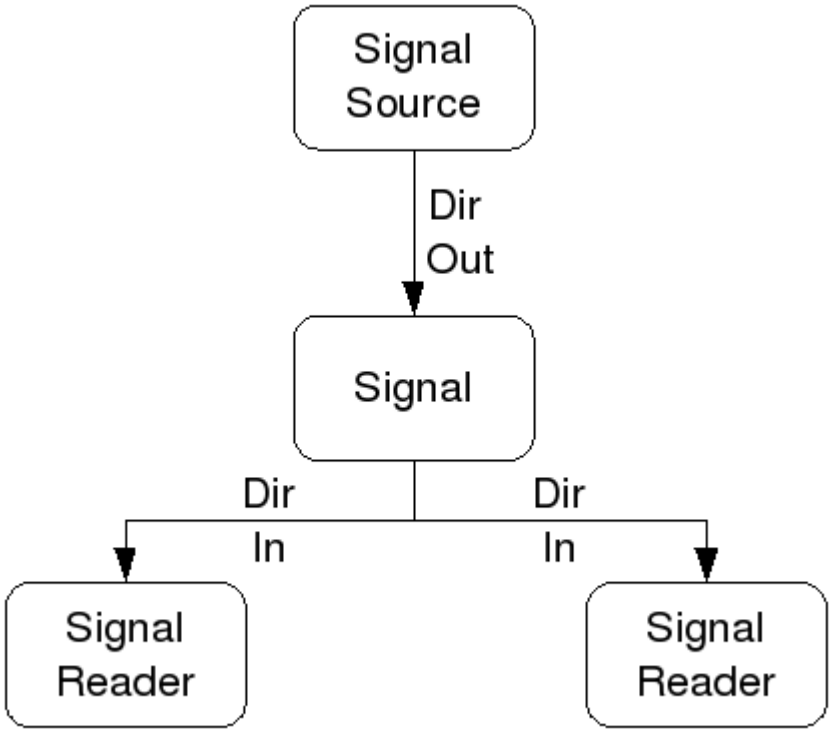


Abbildung 5.4: Signalrichtung (engl. signal direction)

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.

5.2.1.5 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.

setp Syntax und Beispiel

```
setp <pin/parameter-name> <value>
setp parport.0.pin-08-out TRUE
```

5.2.1.6 sets

Der Befehl *sets* setzt den Wert eines Signals.

setzt Syntax und Beispiel:

```
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

5.2.1.7 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.

unlinkp Syntax und Beispiel

```
unlinkp <pin-name>
unlinkp parport.0.pin-02-out
```

5.2.1.8 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.

Der Befehl *linksp* stellt eine *Verbindung* (engl. connection) zwischen einem Signal und einem Pin her.

linksp Syntax und Beispiel

```
linksp <signal-name> <pin-name>
linksp X-step parport.0.pin-02-out
```

Der Befehl *linksp* wurde durch den Befehl *net* abgelöst.

Der Befehl *linkps* stellt eine *Verbindung* zwischen einem Pin und einem Signal her. Er ist der gleiche wie linksp, aber die Argumente sind vertauscht.

linkps Syntax und Beispiel

```
linkps <pin-name> <signal-name>
linkps parport.0.pin-02-out X-Step
```

Der Befehl *linkps* wurde durch den Befehl *net* abgelöst.

der Befehl *newsig* erzeugt ein neues HAL-Signal mit dem Namen <signame> und dem Datentyp <type>. Der Typ muss *bit*, *s32*, *u32* oder *float* sein. Fehler, wenn <signame> bereits existiert.

newsig Syntax und Beispiel

```
newsig <signame> <type>
newsig Xstep bit
```

Weitere Informationen finden Sie im HAL-Handbuch oder in den Man Pages für `halrun`.

5.2.2 HAL Data

5.2.2.1 Bit

Ein Bitwert ist ein Ein oder Aus.

- bit values = true oder 1 und false oder 0 (True, TRUE, oder true sind alles gültige Werte)

5.2.2.2 Float

Ein ”Float” ist eine Gleitkommazahl. Das heißt, der Dezimalpunkt kann nach Bedarf verschoben werden.

- Float-Werte = ein 64-Bit-Fließkommawert mit einer Auflösung von etwa 53 Bit und einem Dynamikbereich von über 2^{10} (etwa 1000) Bit.

Weitere Informationen über Gleitkommazahlen finden Sie unter:

http://en.wikipedia.org/wiki/Floating_point

5.2.2.3 s32

Eine *s32*-Zahl ist eine ganze Zahl, die einen negativen oder positiven Wert haben kann.

- s32-Werte = ganzzahlige Werte von -2147483648 bis 2147483647

5.2.2.4 u32

Eine ”u32”-Zahl ist eine ganze Zahl, die nur positiv ist.

- u32-Werte = Ganzzahlige Zahlen von 0 bis 4294967295

5.2.3 HAL Files

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 HAL Parameter

Zwei Parameter werden automatisch zu jeder HAL-Komponente hinzugefügt, wenn sie erstellt wird. Mit diesen Parametern können Sie die Ausführungszeit einer Komponente festlegen.

`.time`	Zeit ist die Anzahl der CPU-Zyklen, die für die Ausführung der Funktion benötigt wurden.
`.tmax`	Tmax ist die maximale Anzahl von CPU-Zyklen, die zur Ausführung der Funktion benötigt wurden.

tmax” ist ein Lese-/Schreibparameter, so dass der Benutzer ihn auf 0 setzen kann, um die erste Initialisierung der Ausführungszeit der Funktion loszuwerden.

5.2.5 Grundlegende logische 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.

For further components see HAL Components List or the man pages.

5.2.5.1 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 (bit, in)
and2.N.in1 (bit, in)
and2.N.out (bit, out)
```

Tabelle 5.1: und2 Wahrheitstabelle

in0	in1	out
False	False	False
True	False	False
False	True	False
True	True	True

5.2.5.2 not

Die Komponente ”not” ist ein Bit-Inverter.

Syntax

```
not [count=n] | [names=name1[,name2...]]
```

Funktionen

```
not.all
not.n
```

Pins

```
not.n.in (bit, in)
not.n.out (bit, out)
```

Tabelle 5.2: not Wahrheitstabelle

in	out
True	False
False	True

5.2.5.3 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 (bit, in)
or2.n.in1 (bit, in)
or2.n.out (bit, out)
```

Tabelle 5.3: or2 Wahrheitstabelle

in0	in1	out
True	False	True
True	True	True
False	True	True
False	False	False

5.2.5.4 xor2

Die *xor2*-Komponente ist ein XOR-Gatter (exklusives ODER) mit zwei Eingängen.

Syntax

```
xor2[count=n] | [names=name1[,name2...]]
```

Funktionen

```
xor2.n
```

Pins

```
xor2.n.in0 (bit, in)
xor2.n.in1 (bit, in)
xor2.n.out (bit, out)
```

Tabelle 5.4: xor2-Wahrheitstabelle

in0	in1	out
True	False	True
True	True	False
False	True	True
False	False	False

5.2.6 Logikbeispiele

and2-Beispiel für das Anschließen von zwei Eingängen an einen Ausgang

```
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 dem obigen Beispiel wird eine Kopie von ”and2” in den Echtzeitbereich geladen und dem Servo-Thread hinzugefügt. Als nächstes wird `Pin-11` des parallelen Anschlusses mit dem `in0`-Bit des and-Gatters verbunden. Als nächstes wird `Pin-12` mit dem `in1`-Bit des and-Gatters verbunden. Zuletzt verbinden wir das Ausgangsbit ”and2” mit dem parallelen Anschluss ”Pin-14”. Wenn also nach der Wahrheitstabelle für ”and2” Pin 11 und Pin 12 eingeschaltet sind, dann ist der Ausgangs-Pin 14 eingeschaltet.

5.2.7 Konvertierungskomponenten

5.2.7.1 weighted_sum

Die weighted_sum (engl. für gewichtete Summe) wandelt eine Gruppe von Bits in eine ganze Zahl um. Die Umwandlung ist die Summe der ”Gewichte” der vorhandenen Bits plus eines eventuellen Offsets. Sie ähnelt der *binär kodierten Dezimalzahl*, hat aber mehr Möglichkeiten. Das *Hold*-Bit unterbricht die Eingabeverarbeitung, so dass sich der *Summen*-Wert nicht mehr ändert.

Syntax für das Laden der Komponente weighted_sum

```
loadrt weighted_sum wsum_sizes=size[,size,...]
```

Erzeugt Gruppen von `weighted_sum`s, jede mit der angegebenen Anzahl von Eingabebits (Größe).

Um die ”weighted_sum” zu aktualisieren, muss der ”process_wsums” an einen Thread angehängt werden.

add process_wsums function

```
addf process_wsums servo-thread
```

Which updates the `weighted_sum` component.

Im folgenden Beispiel, einer Kopie des AXIS HAL Konfigurationsfensters, sind die Bits *0* und *2* TRUE, sie haben keinen Offset. Das Gewicht (”weight”) von Bit 0 ist 1, das von Bit 2 ist 4, die Summe ist also 5.

Beispiel zu weighted_sum

```
Komonenten-Pins:
Owner   Typ   Richtung  Wert   Name
    10  bit   In           TRUE  wsum.0.bit.0.in
    10  s32   I/O             1  wsum.0.bit.0.weight
    10  bit   In          FALSE  wsum.0.bit.1.in
    10  s32   I/O             2  wsum.0.bit.1.weight
    10  bit   In           TRUE  wsum.0.bit.2.in
    10  s32   I/O             4  wsum.0.bit.2.weight
    10  bit   In          FALSE  wsum.0.bit.3.in
    10  s32   I/O             8  wsum.0.bit.3.weight
    10  bit   In          FALSE  wsum.0.hold
    10  s32   I/O             0  wsum.0.offset
    10  s32   Out             5  wsum.0.sum
```

5.3 HAL TWOPASS

5.3.1 TWOPASS

LinuxCNC 2.5 unterstützt TWOPASS Verarbeitung von LinuxCNC Konfigurationsdateien, die in der Modularisierung und Lesbarkeit von LinuxCNC-Dateien helfen können. (LinuxCNC-Dateien sind in einem LinuxCNC INI-Datei in der LinuxCNC Stanza als [LinuxCNC]HALFILE=filename angegeben).

Normalerweise muss ein Satz von einer oder mehreren LinuxCNC-Konfigurationsdateien eine einzige, eindeutige loadrt-Zeile verwenden, um ein Kernel-Modul zu laden, das mehrere Instanzen einer Komponente behandeln 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:

```
loadrt and2 count=3
```

was zu den Komponenten and2.0, and2.1 und 2.2 führt.

Konfigurationen sind besser lesbar, wenn Sie die Option names= für Komponenten angeben, bei denen sie unterstützt wird, z. B.:

```
loadrt and2 names=aa,ab,ac
```

was zu den Komponenten aa,ab,ac führt.

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-Richtlinie für die Komponente finden und aktualisieren.

Die TWOPASS-Verarbeitung wird durch Einfügen eines INI-Datei-Parameters im Abschnitt [HAL] aktiviert:

```
[HAL]

TWOPASS = anystring
```

Dabei kann ”anystring” eine beliebige Zeichenfolge sein, die nicht null ist. Mit dieser Einstellung können Sie mehrere Spezifikationen haben wie:

```
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 mehrfache Auftritte von loadrt-Anweisungen für jedes Modul kumuliert. Die Benutzerkomponenten (loadusr) werden der Reihe nach geladen, aber keine anderen LinuxCNC-Befehle werden im ersten Durchgang ausgeführt.

Anmerkung
Benutzerkomponenten 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), und zwar mit einer Anzahl, die der Gesamtzahl entspricht, wenn die Option *count*= verwendet wird, oder mit allen angegebenen Einzelnamen, wenn die Option *names*= verwendet wird.

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]POSTGU
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 halt-cl, 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.

Anmerkung
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
```

Anmerkung
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-Tutorial

5.4.1 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.

5.4.2 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/man halcmd
```

5.4.2.1 Notation

For this tutorial, commands for the operating system are typically shown without the prompt provided by the UNIX shell, i.e typically a dollar sign ($) or a hash/double cross (#). When communicating directly with the HAL through `halcmd` or `halrun`, the prompts are shown in the examples. The terminal window is in *Applications/Accessories* from the main Ubuntu menu bar.

Terminal Command Example - prompts

```
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
```

5.4.2.2 Befehl-Vervollständigung durch Tabulator-Taste

Your version of `halcmd` may include tab-completion. Instead of completing file names as a shell does, it completes commands with HAL identifiers. You will have to type enough letters for a unique match. Try pressing tab after starting a HAL command:

Befehl-Vervollständigung durch Tabulator-Taste

```
halcmd: loa<TAB>
halcmd: load
halcmd: loadrt
halcmd: loadrt cou<TAB>
halcmd: loadrt counter
```

5.4.2.3 Die RTAPI-Umgebung

RTAPI stands for Real Time Application Programming Interface. Many HAL components work in real-time, and all HAL components store data in shared memory so realtime components can access it. Regular Linux does not support realtime programming or the type of shared memory that HAL needs. Fortunately, there are realtime operating systems (RTOS's) that provide the necessary extensions to Linux. Unfortunately, each RTOS does things a little differently.

Um diese Unterschiede zu beseitigen, hat das LinuxCNC-Team die RTAPI entwickelt, die einen einheitlichen Weg für Programme bietet, um mit dem RTOS zu kommunizieren. Wenn Sie ein Programmierer sind, der an den Interna von LinuxCNC arbeiten will, sollten Sie vielleicht *linuxcnc/src/rtapi/rtapi.h* studieren, um die API zu verstehen. Aber wenn Sie eine normale Person sind, ist alles, was Sie über RTAPI wissen müssen, dass es (und das RTOS) in den Speicher Ihres Computers geladen werden muss, bevor Sie etwas mit HAL machen.

5.4.3 Ein einfaches Beispiel

5.4.3.1 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.

For the first example, we will use a HAL component called *siggen*, which is a simple signal generator. A complete description of the *siggen* component can be found in the SigGen section of this Manual. It is a realtime component, implemented as a Linux kernel module. To load the "`siggen`" component, use the HAL command `loadrt`.

Laden von siggen

```
halcmd: loadrt siggen
```

5.4.3.2 Untersuchung der HAL

Now that the module is loaded, it is time to introduce `halcmd`, the command line tool used to configure the HAL. This tutorial will introduce only a selection of halcmd features. For a more complete description try `man halcmd`, or see the reference in HAL Commands section of this document. The first halcmd feature is the *show* command. This command displays information about the current state of the HAL. To show all installed components:

Show Components with halrun/halcmd

```
halcmd: show comp

    Loaded HAL Components:
    ID      Type  Name                                PID   State
    3       RT    siggen                                    ready
    2       User  halcmd2177                          2177  ready
```

Since *halcmd* itself is also a HAL component, it will always show up in the list. The number after "`halcmd`" in the component list is the UNIX process ID. It is possible to run more than one copy of halcmd at the same time (in different terminal windows for example), so the PID is added to the end of the name to make it unique. The list also shows the *siggen* component that we installed in the previous step. The *RT* under *Type* indicates that *siggen* is a realtime component. The *User* under *Type* indicates it is a user space component.

Next, let's see what pins `siggen` makes available:

Pins anzeigen

```
halcmd: show pin

Component Pins:
Owner   Type   Dir         Value  Name
     3  float  IN              1  siggen.0.amplitude
     3  bit    OUT         FALSE  siggen.0.clock
     3  float  OUT             0  siggen.0.cosine
     3  float  IN              1  siggen.0.frequency
     3  float  IN              0  siggen.0.offset
     3  float  OUT             0  siggen.0.sawtooth
     3  float  OUT             0  siggen.0.sine
     3  float  OUT             0  siggen.0.square
     3  float  OUT             0  siggen.0.triangle
```

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 nine pins. Of these pins eight are floating point and one is bit (boolean). Six carry data out of the *siggen* component and three are used to transfer settings into the component. 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
     3  s32   RO              0   siggen.0.update.time
     3  s32   RW              0   siggen.0.update.tmax
```

The *show param* command shows all the parameters in the HAL. Right now, each parameter has the default value it was given when the component was loaded. Note the column labeled *Dir*. The parameters labeled *-W* are writable ones that are never changed by the component itself, instead they are meant to be changed by the user to control the component. We will see how to do this later. Parameters labeled *R-* are read only parameters. They can be changed only by the component. Finally, parameter 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 parameters `siggen.0.update.time` and `siggen.0.update.tmax` are for debugging purposes and won't be covered in this section.

Most realtime components export one or more functions to actually run the realtime code they contain. Let's see what function(s) *siggen* exported:

Show Functions with 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 requires floating point. It is not currently linked to any threads, so *users* is zero [1].

5.4.3.3 Echtzeitcode zum Laufen bringen

To actually run the code contained in the function `siggen.0.update`, we need a realtime thread. The component called *threads* that is used to create a new thread. Lets create a thread called ”test-thread” with a period of 1 ms (1,000 µs or 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, and which can handle floating point functions. The next step is to connect the function to the thread:

Funktion hinzufügen

```
halcmd: addf siggen.0.update test-thread
```

Up till now, we’ve been using `halcmd` only to look at the HAL. However, this time we used the `addf` (add function) command to actually change something in the HAL. We told *halcmd* to add the function `siggen.0.update` to the thread *test-thread*, and if we look at the thread list again, we see that it succeeded:

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
                 1 siggen.0.update
```

There is one more step needed before the *siggen* component starts generating signals. When the HAL is first started, the thread(s) are not actually running. This is to allow you to completely configure the system before the realtime code starts. Once you are happy with the configuration, you can start the realtime code like this:

```
halcmd: start
```

Jetzt läuft der Signalgenerator. Schauen wir uns seine Ausgangspins an:

[1] CodeAddr and Arg fields were used during development and should probably disappear.

```
halcmd: show pin

Komponenten-Pins:
Owner   Type  Dir         Value  Name
     3  float IN              1  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
     3  float OUT    -0.1640929  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
     3  float IN              0  siggen.0.offset
     3  float OUT    -0.4475303  siggen.0.sawtooth
     3  float OUT     0.9864449  siggen.0.sine
     3  float OUT            -1  siggen.0.square
     3  float OUT    -0.1049393  siggen.0.triangle
```

Und schauen wir noch einmal hin:

```
halcmd: show pin

Komponenten Pins:
Owner   Type  Dir         Value  Name
     3  float IN              1  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
     3  float OUT     0.0507619  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
     3  float IN              0  siggen.0.offset
     3  float OUT     -0.516165  siggen.0.sawtooth
     3  float OUT     0.9987108  siggen.0.sine
     3  float OUT            -1  siggen.0.square
     3  float OUT    0.03232994  siggen.0.triangle
```

We did two `show pin` commands in quick succession, and you can see that the outputs are no longer zero. The sine, cosine, sawtooth, and triangle outputs are changing constantly. The square output is also working, however it simply switches from +1.0 to -1.0 every cycle.

5.4.3.4 Ändern von Parametern

The real power of HAL is that you can change things. For example, we can use the `setp` command to set the value of a parameter. Let's change the amplitude of the signal generator from 1.0 to 5.0:

Pin einstellen (engl. set)

```
halcmd: setp siggen.0.amplitude 5
```

Überprüfen Sie die Parameter und Pins erneut

```
halcmd: show param

Parameter:
Owner   Type  Dir         Value  Name
     3  s32   RO           1754  siggen.0.update.time
     3  s32   RW          16997  siggen.0.update.tmax

halcmd: show pin

Komponenten-Pins:
Owner   Type  Dir         Value  Name
     3  float IN              5  siggen.0.amplitude
     3  bit   OUT         FALSE  siggen.0.clock
     3  float OUT     0.8515425  siggen.0.cosine
     3  float IN              1  siggen.0.frequency
```

```
     3  float IN              0  siggen.0.offset
     3  float OUT      2.772382  siggen.0.sawtooth
     3  float OUT     -4.926954  siggen.0.sine
     3  float OUT             5  siggen.0.square
     3  float OUT      0.544764  siggen.0.triangle
```

Note that the value of parameter `siggen.0.amplitude` has changed to 5, and that the pins now have larger values.

5.4.3.5 Speichern der HAL-Konfiguration

Most of what we have done with `halcmd` so far has simply been viewing things with the *show* command. However two of the commands actually changed things. As we design more complex systems with HAL, we will use many commands to configure things just the way we want them. HAL has the memory of an elephant, and will retain that configuration until we shut it down. But what about next time? We don't want to manually enter a bunch of commands every time we want to use the system.

Saving the configuration of the entire HAL with a single command.

```
halcmd: save

# Komponenten
loadrt threads name1=test-thread period1=1000000
loadrt siggen
# Pin-Aliase
# Signale
# Netze
# Parameterwerte
setp siggen.0.update.tmax 14687
# Echtzeit-Thread/Funktions-Verknüpfungen
addf siggen.0.update test-thread
```

The output of the `save` command is a sequence of HAL commands. If you start with an *empty* HAL and run all these commands, you will get the configuration that existed when the *save* command was issued. To save these commands for later use, we simply redirect the output to a file:

Save configuration to a file with halcmd

```
halcmd: save all saved.hal
```

5.4.3.6 Halrun beenden

When you're finished with your HAL session type `exit` at the "`halcmd:`" prompt. This will return you to the system prompt and close down the HAL session. Do not simply close the terminal window without shutting down the HAL session.

HAL beenden

```
halcmd: exit
```

5.4.3.7 Wiederherstellung der HAL-Konfiguration

To restore the HAL configuration stored in the file "saved.hal", we need to execute all of those HAL commands. To do that, we use "`-f _<file name>_`" which reads commands from a file, and "`-I`" (upper case i) which shows the halcmd prompt after executing the commands:

Ausführen einer gespeicherten Datei

```
halrun -I -f saved.hal
```

Notice that there is not a ”start” command in saved.hal. It’s necessary to issue it again (or edit the file saved.hal to add it there).

5.4.3.8 HAL aus dem Speicher entfernen

If an unexpected shutdown of a HAL session occurs you might have to unload HAL before another session can begin. To do this type the following command in a terminal window.

Removing HAL

```
halrun -U
```

5.4.4 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.

It allows to observe the pins, signals or parameters by displaying the current value of these entities. It is very easy to use application for graphical environments. In a console type:

```
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
```

At this point we have the siggen component loaded and running. It’s time to start halmeter.

Halmeter starten

```
halcmd: loadusr halmeter
```

The first window you will see is the ”Select Item to Probe” window.

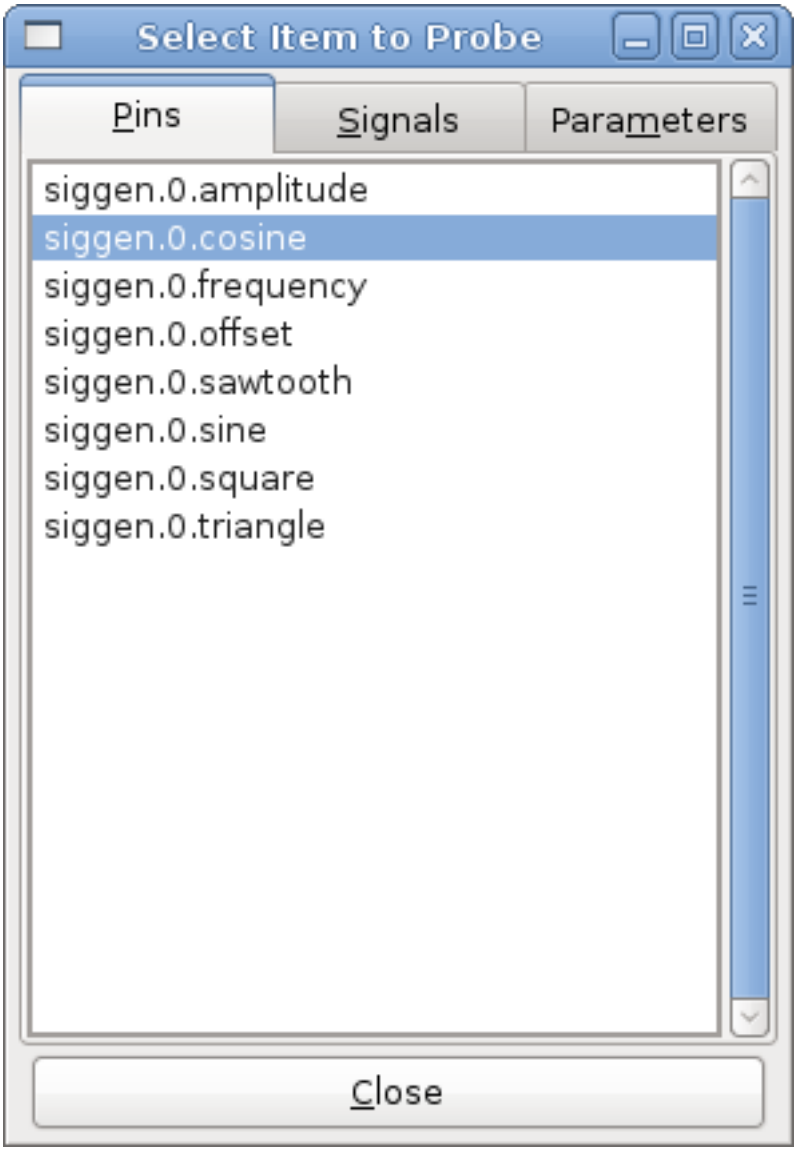


Abbildung 5.5: Halmeter Auswahlfenster

This dialog has three tabs. The first tab displays all of the HAL pins in the system. The second one displays all the signals, and the third displays all the parameters. We would like to look at the pin

`siggen.0.cosine` first, so click on it then click the ”Close” button. The probe selection dialog will close, and the meter looks something like the following figure.

Abbildung 5.6: Halmeter-Fenster

To change what the meter displays press the ”Select” button which brings back the ”Select Item to Probe” window.

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.

5.4.5 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
```

5.4.5.1 Installieren der Komponenten

Now we are going to load the step pulse generator component. For a detailed description of this component refer to the stepgen section of the Integrator Manual. In this example we will use the *velocity* control type of StepGen. For now, we can skip the details, and just run the following commands.

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 fp1=0 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). The fast thread doesn’t support floating point functions.

As before, we can use `halcmd show` to take a look at the HAL. This time we have a lot more pins and parameters than before:

```
halcmd: show pin

Component Pins:
Owner   Type  Dir         Value  Name
     4  float IN              1  siggen.0.amplitude
     4  bit   OUT         FALSE  siggen.0.clock
     4  float OUT             0  siggen.0.cosine
     4  float IN              1  siggen.0.frequency
     4  float IN              0  siggen.0.offset
     4  float OUT             0  siggen.0.sawtooth
     4  float OUT             0  siggen.0.sine
     4  float OUT             0  siggen.0.square
     4  float OUT             0  siggen.0.triangle
     3  s32   OUT             0  stepgen.0.counts
     3  bit   OUT         FALSE  stepgen.0.dir
     3  bit   IN          FALSE  stepgen.0.enable
     3  float OUT             0  stepgen.0.position-fb
     3  bit   OUT         FALSE  stepgen.0.step
     3  float IN              0  stepgen.0.velocity-cmd
     3  s32   OUT             0  stepgen.1.counts
     3  bit   OUT         FALSE  stepgen.1.dir
     3  bit   IN          FALSE  stepgen.1.enable
     3  float OUT             0  stepgen.1.position-fb
     3  bit   OUT         FALSE  stepgen.1.step
     3  float IN              0  stepgen.1.velocity-cmd

halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name
     4  s32   RO              0  siggen.0.update.time
     4  s32   RW              0  siggen.0.update.tmax
     3  u32   RW     0x00000001  stepgen.0.dirhold
     3  u32   RW     0x00000001  stepgen.0.dirsetup
     3  float RO              0  stepgen.0.frequency
     3  float RW              0  stepgen.0.maxaccel
     3  float RW              0  stepgen.0.maxvel
     3  float RW              1  stepgen.0.position-scale
     3  s32   RO              0  stepgen.0.rawcounts
     3  u32   RW     0x00000001  stepgen.0.steplen
     3  u32   RW     0x00000001  stepgen.0.stepspace
     3  u32   RW     0x00000001  stepgen.1.dirhold
     3  u32   RW     0x00000001  stepgen.1.dirsetup
     3  float RO              0  stepgen.1.frequency
     3  float RW              0  stepgen.1.maxaccel
     3  float RW              0  stepgen.1.maxvel
     3  float RW              1  stepgen.1.position-scale
     3  s32   RO              0  stepgen.1.rawcounts
     3  u32   RW     0x00000001  stepgen.1.steplen
     3  u32   RW     0x00000001  stepgen.1.stepspace
     3  s32   RO              0  stepgen.capture-position.time
     3  s32   RW              0  stepgen.capture-position.tmax
     3  s32   RO              0  stepgen.make-pulses.time
     3  s32   RW              0  stepgen.make-pulses.tmax
     3  s32   RO              0  stepgen.update-freq.time
     3  s32   RW              0  stepgen.update-freq.tmax
```

5.4.5.2 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
```

To see the effect of the `net` command, we show the signals again.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
float             0  X-vel <== siggen.0.cosine
```

When a signal is connected to one or more pins, the show command lists the pins immediately following the signal name. The *arrow* shows the direction of data flow - in this case, data flows from pin `siggen.0.cosine` to signal `X-vel`. Now let’s connect the `X-vel` to the velocity input of a step pulse generator.

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

We can also connect up the Y axis signal `Y-vel`. It is intended to run from the sine output of the signal generator to the input of the second step pulse generator. The following command accomplishes in one line what two `net` commands accomplished for `X-vel`.

```
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)
float             0  X-vel <== siggen.0.cosine
                            ==> stepgen.0.velocity-cmd
float             0  Y-vel <== siggen.0.sine
                            ==> stepgen.1.velocity-cmd
```

The *show sig* command makes it clear exactly how data flows through the HAL. For example, the *X-vel* signal comes from pin `siggen.0.cosine`, and goes to pin `stepgen.0.velocity-cmd`.

5.4.5.3 Einrichten der Echtzeitausführung - Threads und Funktionen

Thinking about data flowing through ”wires” makes pins and signals fairly easy to understand. Threads and functions are a little more difficult. Functions contain the computer instructions that actually get things done. Thread are the method used to make those instructions run when they are needed. First let’s look at the functions available to us.

```
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
```

In general, you will have to refer to the documentation for each component to see what its functions do. In this case, the function `siggen.0.update` is used to update the outputs of the signal generator. Every time it is executed, it calculates the values of the sine, cosine, triangle, and square outputs. To make smooth signals, it needs to run at specific intervals.

Die anderen drei Funktionen beziehen sich auf die Schrittimpulsgeneratoren.

The first one, `stepgen.capture_position`, is used for position feedback. It captures the value of an internal counter that counts the step pulses as they are generated. Assuming no missed steps, this counter indicates the position of the motor.

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. Unlike the others, it does not need floating point math.

The last function, `stepgen.update-freq`, is responsible for doing scaling and some other calculations that need to be performed only when the frequency command changes.

What this means for our example is that we want to run `siggen.0.update` at a moderate rate to calculate the sine and cosine values. Immediately after we run `siggen.0.update`, we want to run `stepgen.update_freq` to load the new values into the step pulse generator. Finally we need to run `stepgen.make_pulses` as fast as possible for smooth pulses. Because we don’t use position feedback, we don’t need to run `stepgen.capture_position` at all.

We run functions by adding them to threads. Each thread runs at a specific rate. Let’s see what threads we have available.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
      49849  NO                   fast (        0,        0 )
```

The two threads were created when we loaded `threads`. The first one, *slow*, runs every millisecond, and is capable of running floating point functions. We will use it for `siggen.0.update` and `stepgen.update_freq`. The second thread is *fast*, which runs every 50 microseconds (µs), and does not support floating point. 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
```

After we give these commands, we can run the `show thread` command again to see what happened.

```
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.

5.4.5.4 Parameter einstellen

We are almost ready to start our HAL system. However we still need to adjust a few parameters. By default, the siggen component generates signals that swing from +1 to -1. For our example that is fine, we want the table speed to vary from +1 to -1 inches per second. However the scaling of the step pulse generator isn't quite right. By default, it generates an output frequency of 1 step per second with an input of 1.0. It is unlikely that one step per second will give us one inch per second of table movement. Let's assume instead that we have a 5 turn per inch leadscrew, connected to a 200 step per rev stepper with 10x microstepping. So it takes 2000 steps for one revolution of the screw, and 5 revolutions to travel one inch. That means the overall scaling is 10000 steps per inch. We need to multiply the velocity input to the step pulse generator by 10000 to get the proper output. That is exactly what the parameter `stepgen.n.velocity-scale` is for. In this case, both the X and Y axis have the same scaling, so we set the scaling parameters for both to 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
```

This velocity scaling means that when the pin `stepgen.0.velocity-cmd` is 1.0, the step generator will generate 10000 pulses per second (10 kHz). With the motor and leadscrew described above, that will result in the axis moving at exactly 1.0 inches per second. This illustrates a key HAL concept - things like scaling are done at the lowest possible level, in this case in the step pulse generator. The internal signal `X-vel` is the velocity of the table in inches per second, and other components such as `siggen` don't know (or care) about the scaling at all. If we changed the leadscrew, or motor, we would change only the scaling parameter of the step pulse generator.

5.4.5.5 Ausführen!

We now have everything configured and are ready to start it up. Just like in the first example, we use the `start` command.

```
halcmd: start
```

Although nothing appears to happen, inside the computer the step pulse generator is cranking out step pulses, varying from 10 kHz forward to 10 kHz reverse and back again every second. Later in this tutorial we'll see how to bring those internal signals out to run motors in the real world, but first we want to look at them and see what is happening.

5.4.6 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 Echtzeitteil, der als Kernelmodul geladen wird, und einem Benutzerteil, der die grafische Benutzeroberfläche und die Anzeige bereitstellt. Sie müssen sich darüber jedoch keine Gedanken machen, da der Benutzerteil automatisch anfordert, dass der Echtzeitteil

geladen wird. 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.

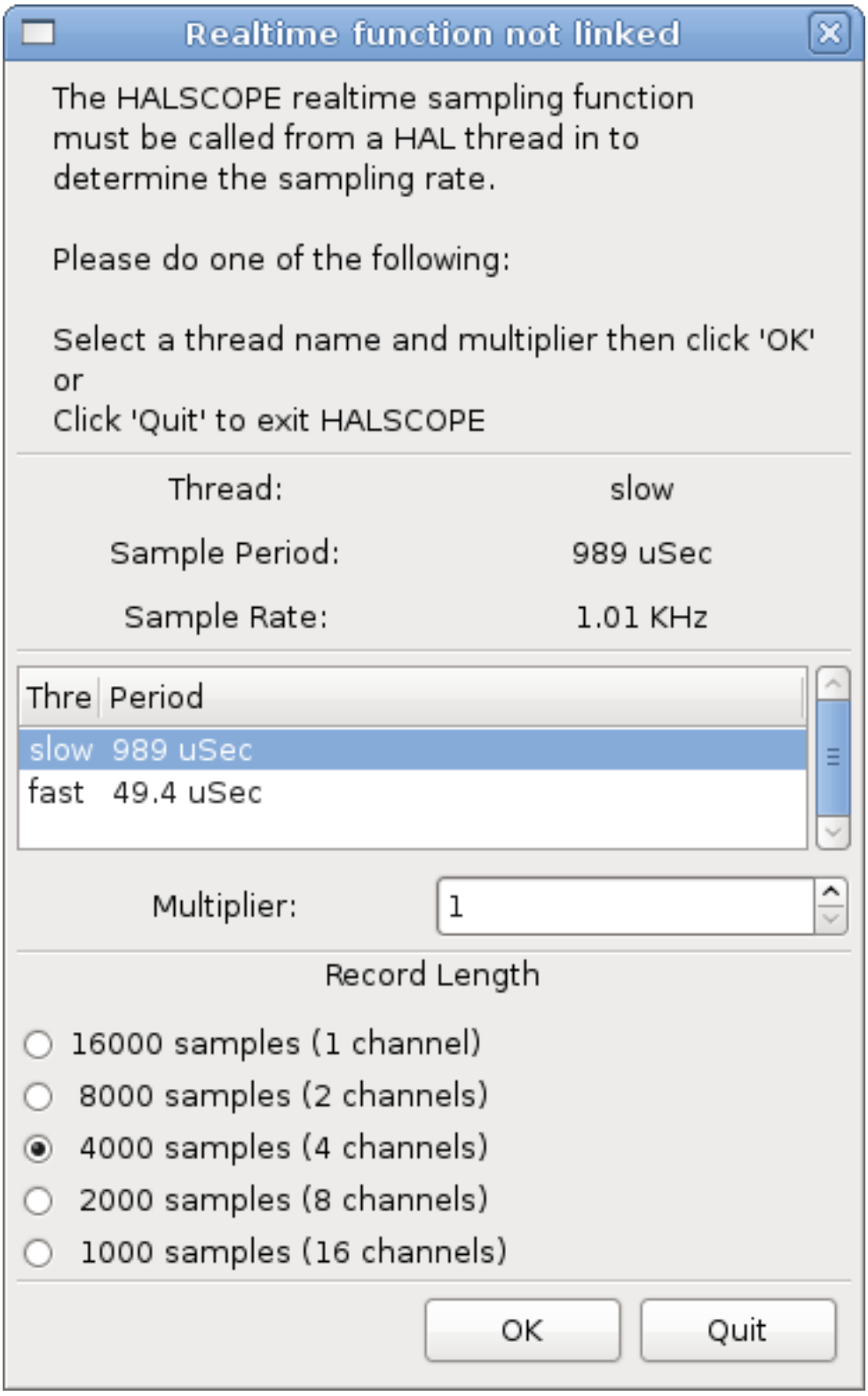


Abbildung 5.7: Dialog *Echtzeitfunktion nicht verknüpft*

This dialog is where you set the sampling rate for the oscilloscope. For now we want to sample once per millisecond, so click on the 989 µs thread *slow* and leave the multiplier at 1. We will also leave the record length at 4000 samples, so that we can use up to four channels at one time. When you select

a thread and then click *OK*, the dialog disappears, and the scope window looks something like the following figure.

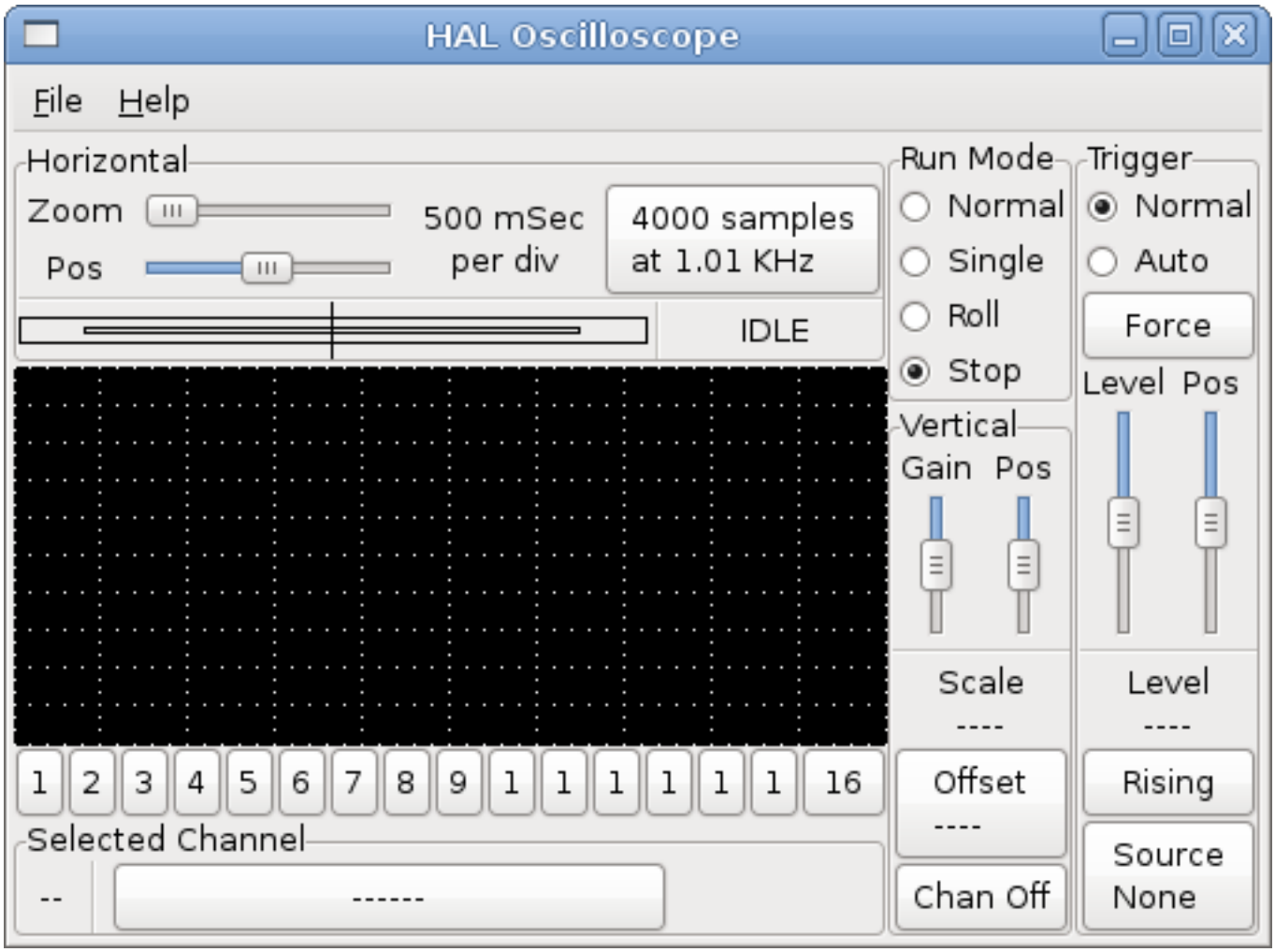


Abbildung 5.8: Fenster für den anfänglichen Geltungsbereich

5.4.6.1 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).

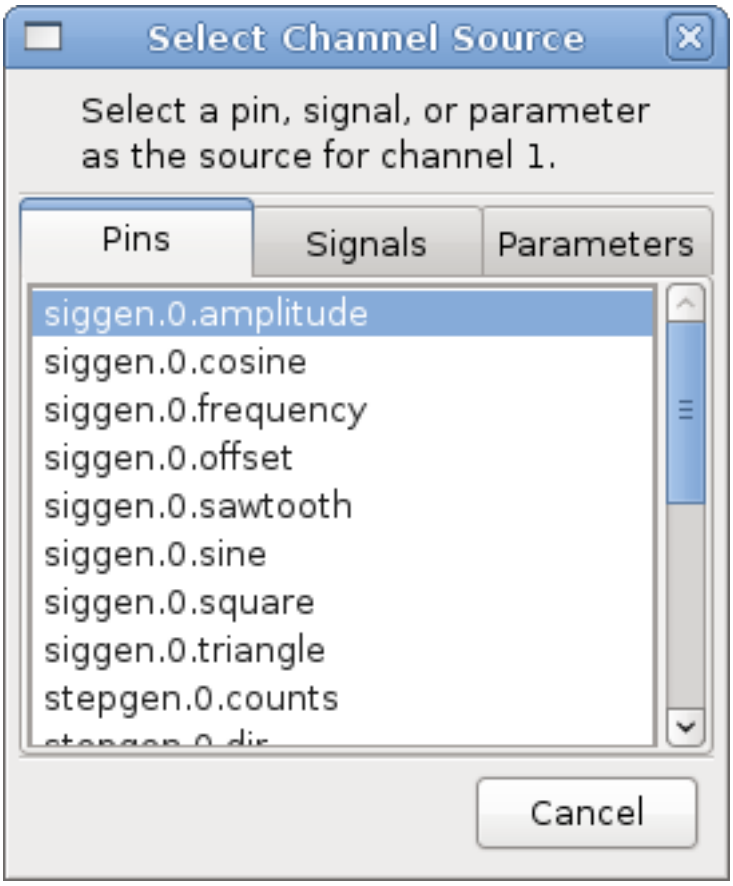


Abbildung 5.9: Kanalquelle auswählen

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.

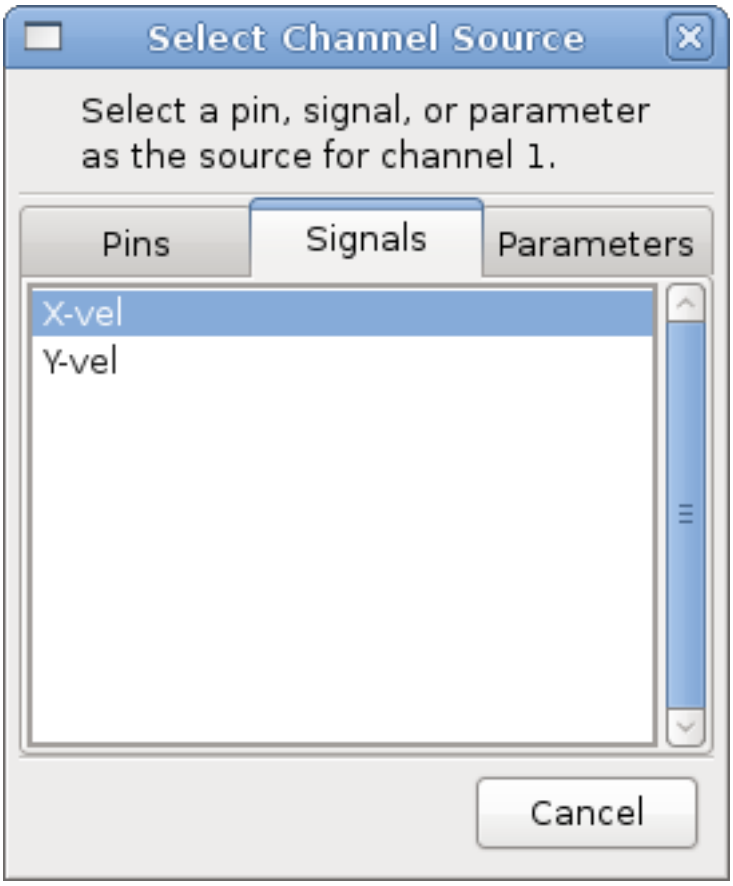


Abbildung 5.10: Signal auswählen

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.

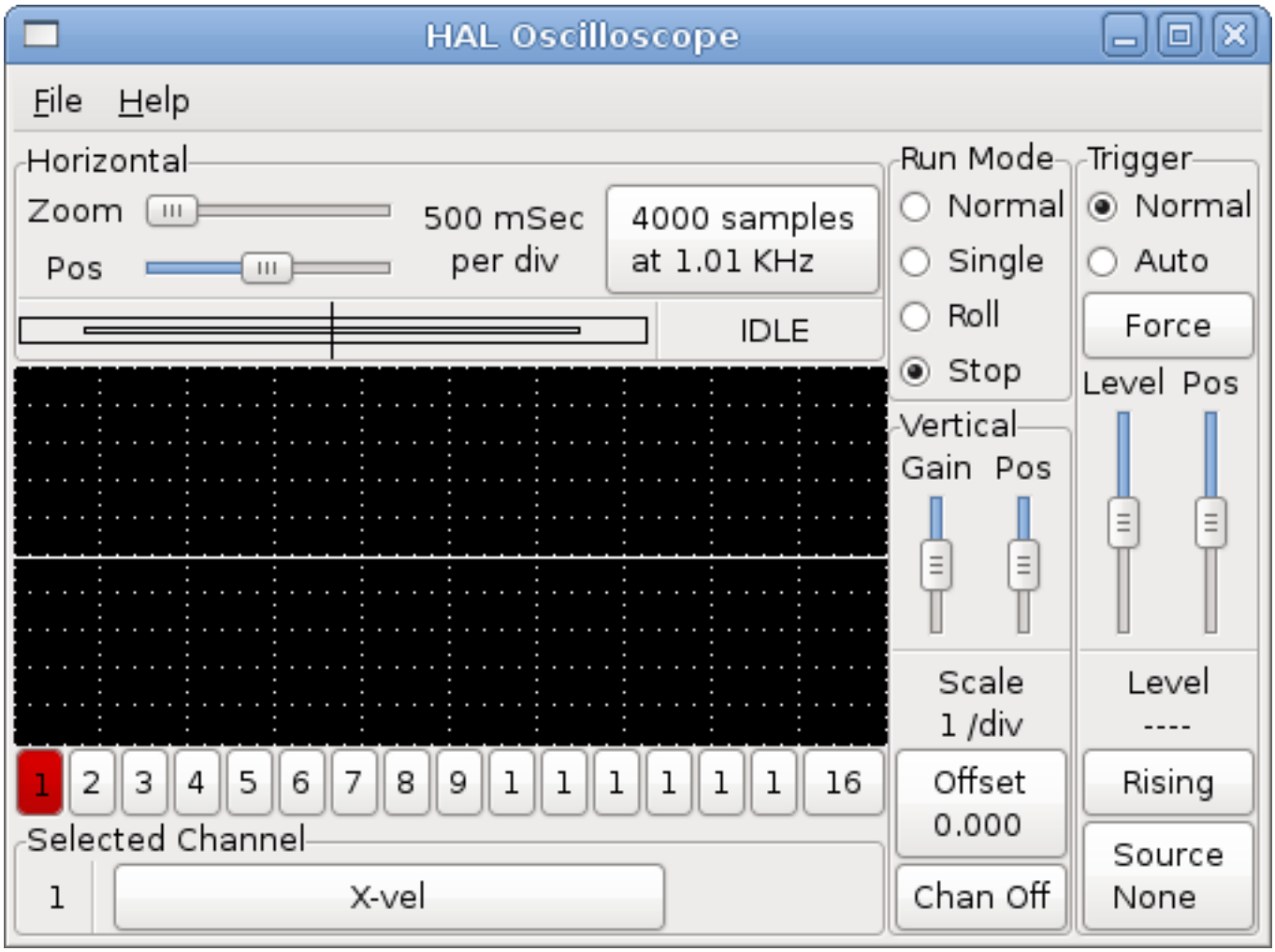


Abbildung 5.11: Halscope

To add a signal to channel 2, click the *2* button. When the dialog pops up, click the *Signals* tab, then click on *Y-vel*. We also want to look at the square and triangle wave outputs. There are no signals connected to those pins, so we use the *Pins* tab instead. For channel 3, select `siggen.0.triangle` and for channel 4, select `siggen.0.square`.

5.4.6.2 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.

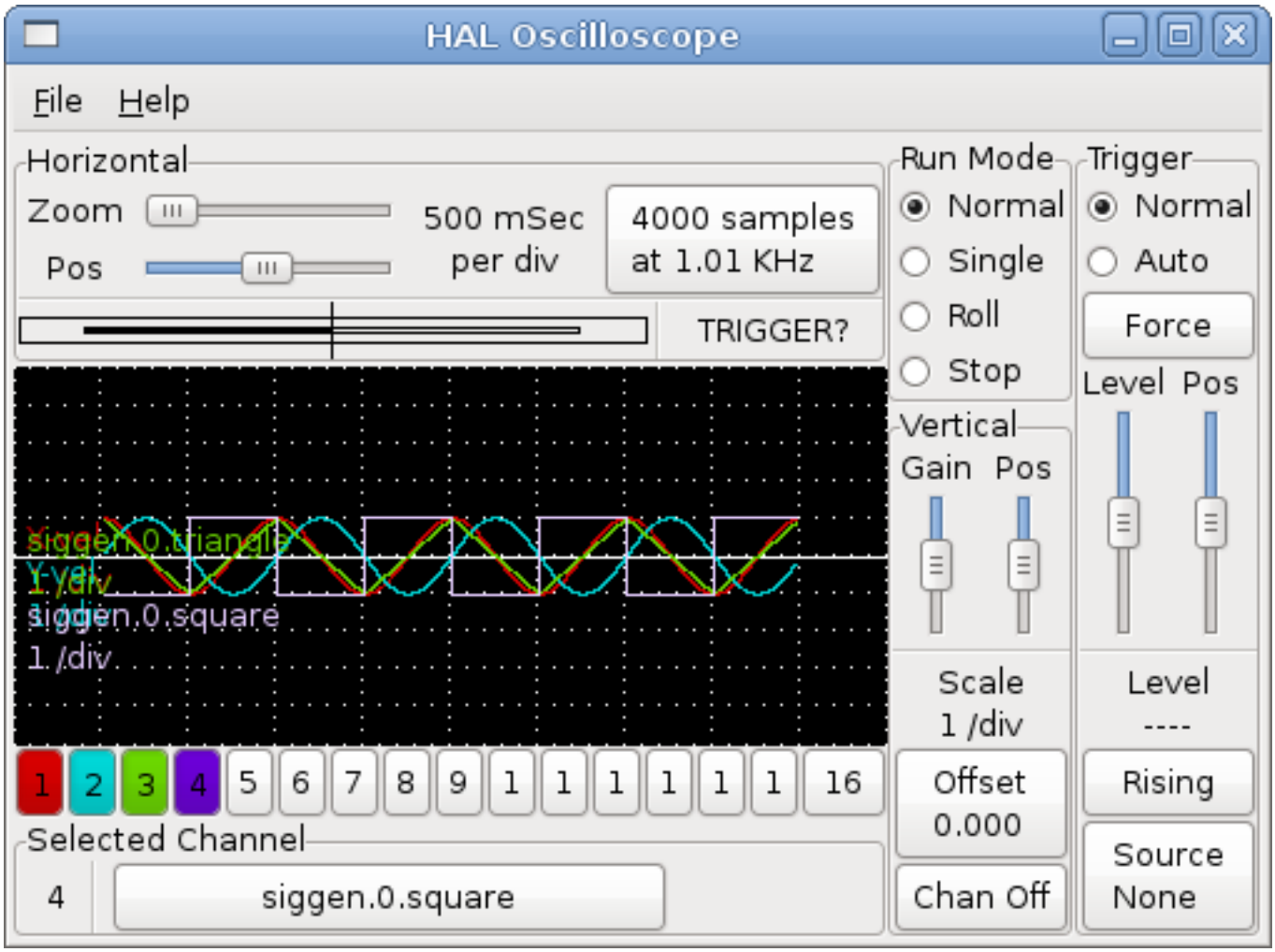


Abbildung 5.12: Erfasste Wellenformen

The *Selected Channel* box at the bottom tells you that the purple trace is the currently selected one, channel 4, which is displaying the value of the pin `siggen.0.square`. Try clicking channel buttons 1 through 3 to highlight the other three traces.

5.4.6.3 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.

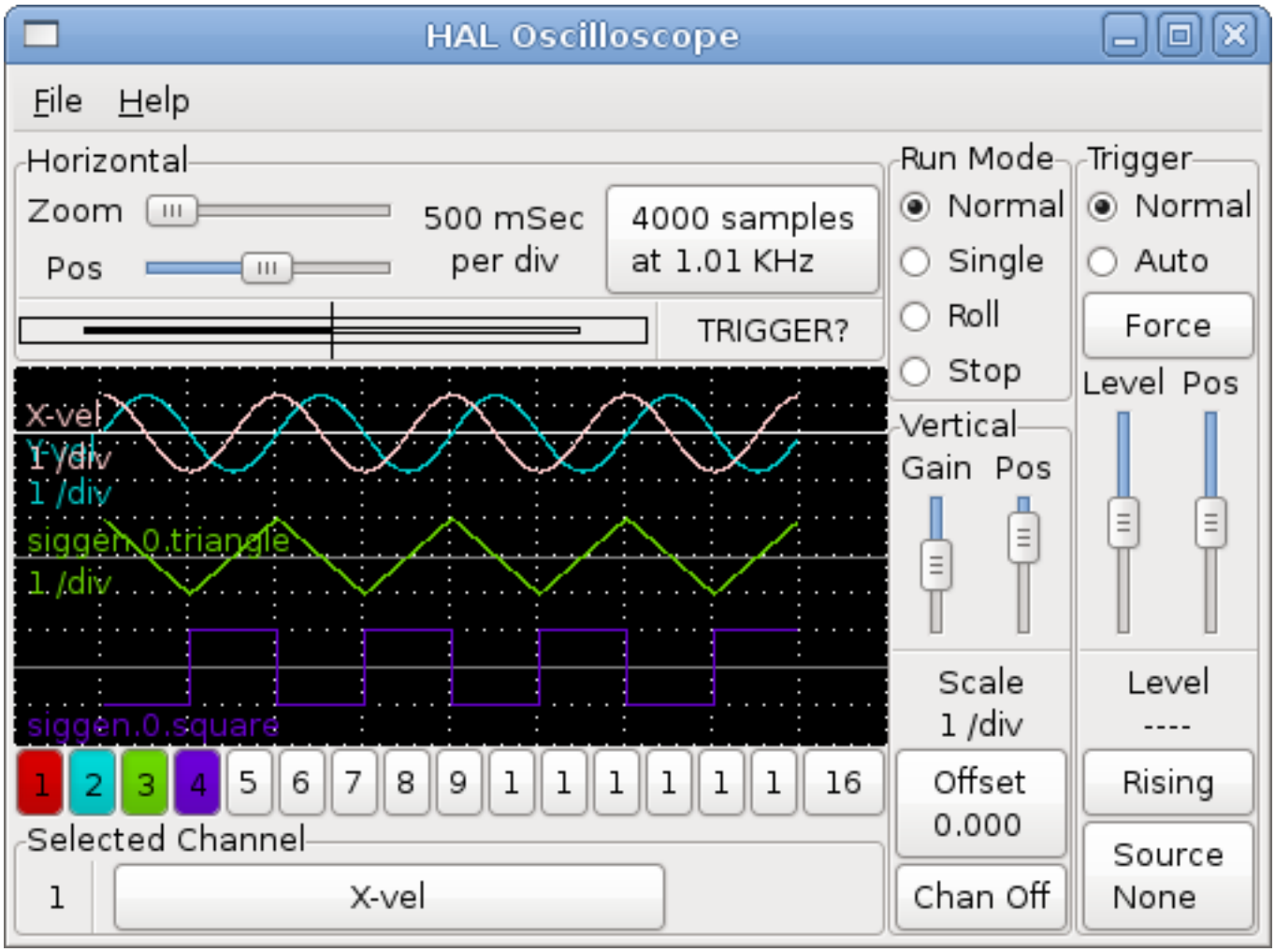


Abbildung 5.13: Vertikale Einstellung

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.

5.4.6.4 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.

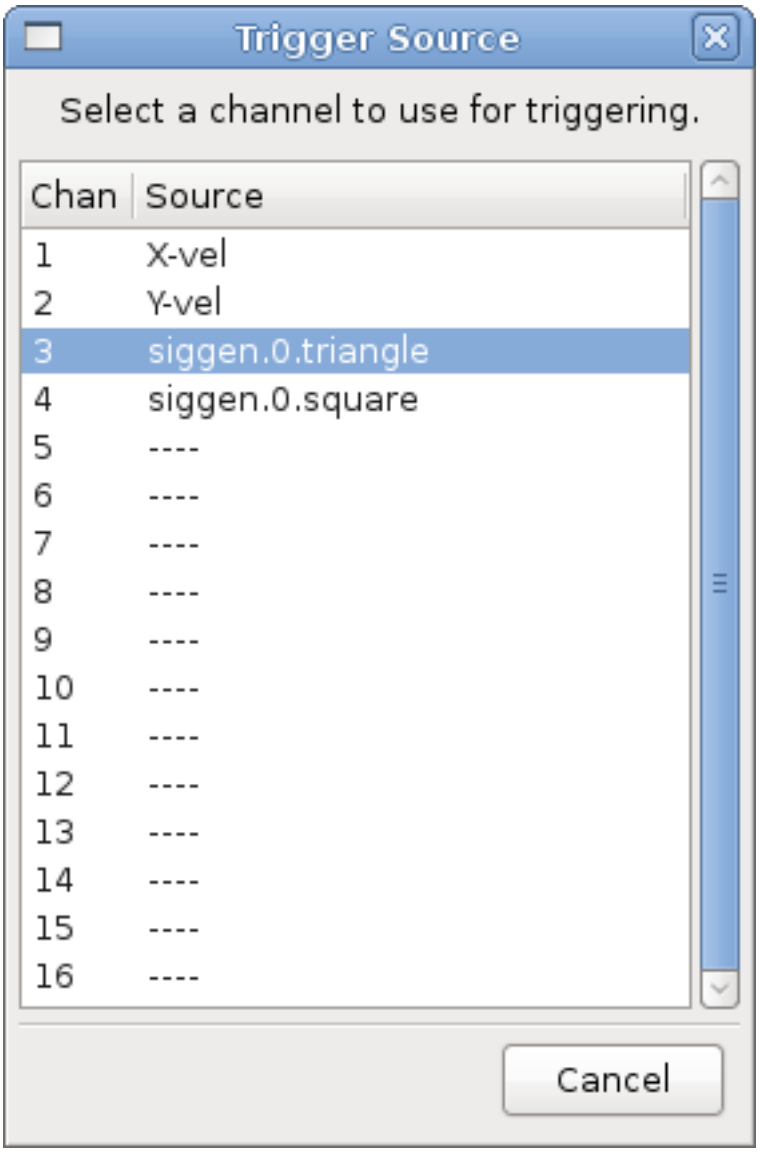


Abbildung 5.14: Dialogfeld *Triggerquelle* (engl. trigger source)

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.

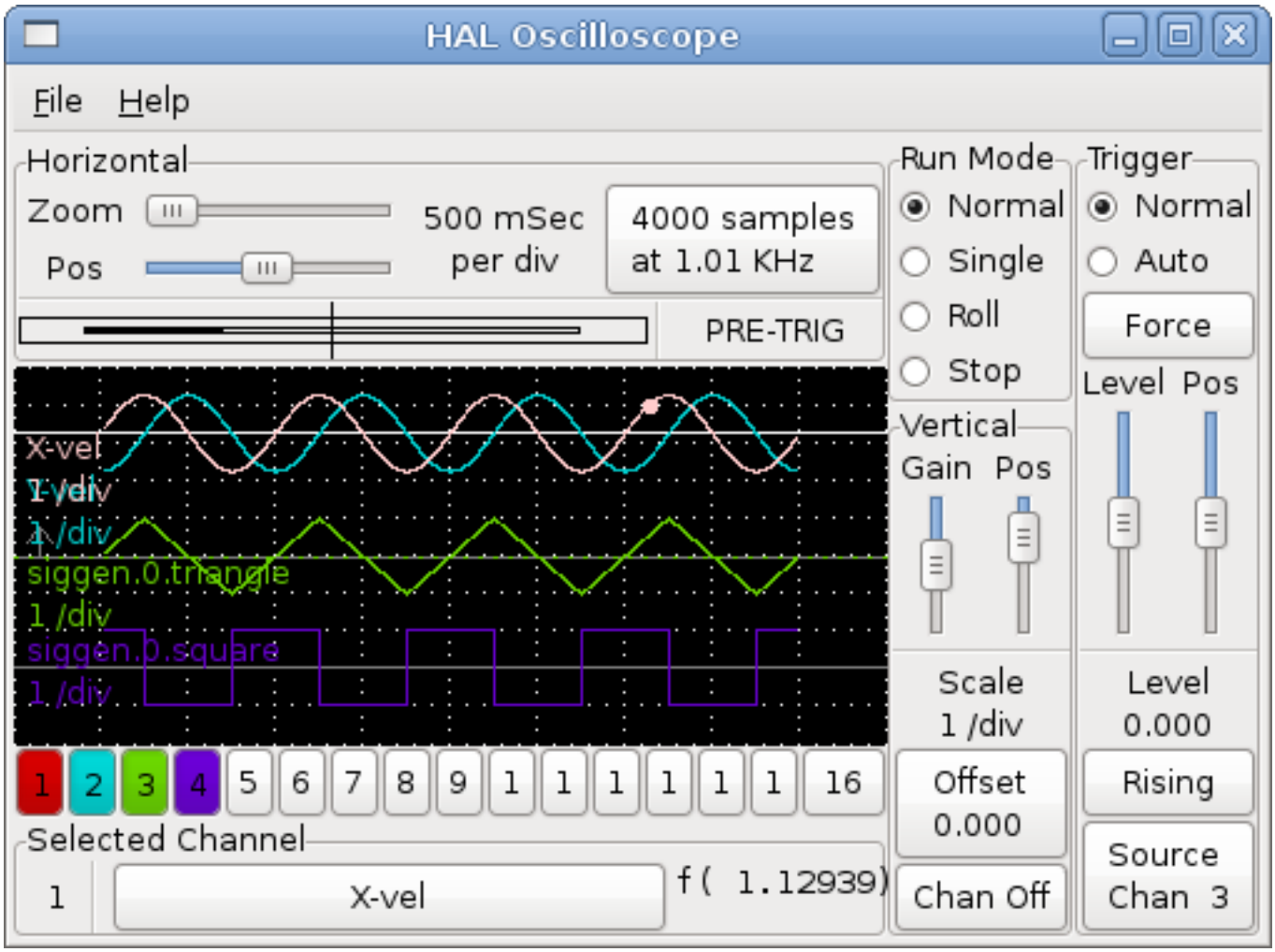


Abbildung 5.15: Wellenformen mit Triggerung

5.4.6.5 Horizontale Anpassungen

To look closely at part of a waveform, you can use the zoom slider at the top of the screen to expand the waveforms horizontally, and the position slider to determine which part of the zoomed waveform is visible. However, sometimes simply expanding the waveforms isn't enough and you need to increase the sampling rate. For example, we would like to look at the actual step pulses that are being generated in our example. Since the step pulses may be only 50 µs long, sampling at 1 kHz isn't fast enough. To change the sample rate, click on the button that displays the number of samples and sample rate to bring up the *Select Sample Rate* dialog figure. For this example, we will click on the 50 µs thread, *fast*, which gives us a sample rate of about 20 kHz. Now instead of displaying about 4 seconds worth of data, one record is 4000 samples at 20 kHz, or about 0.20 seconds.

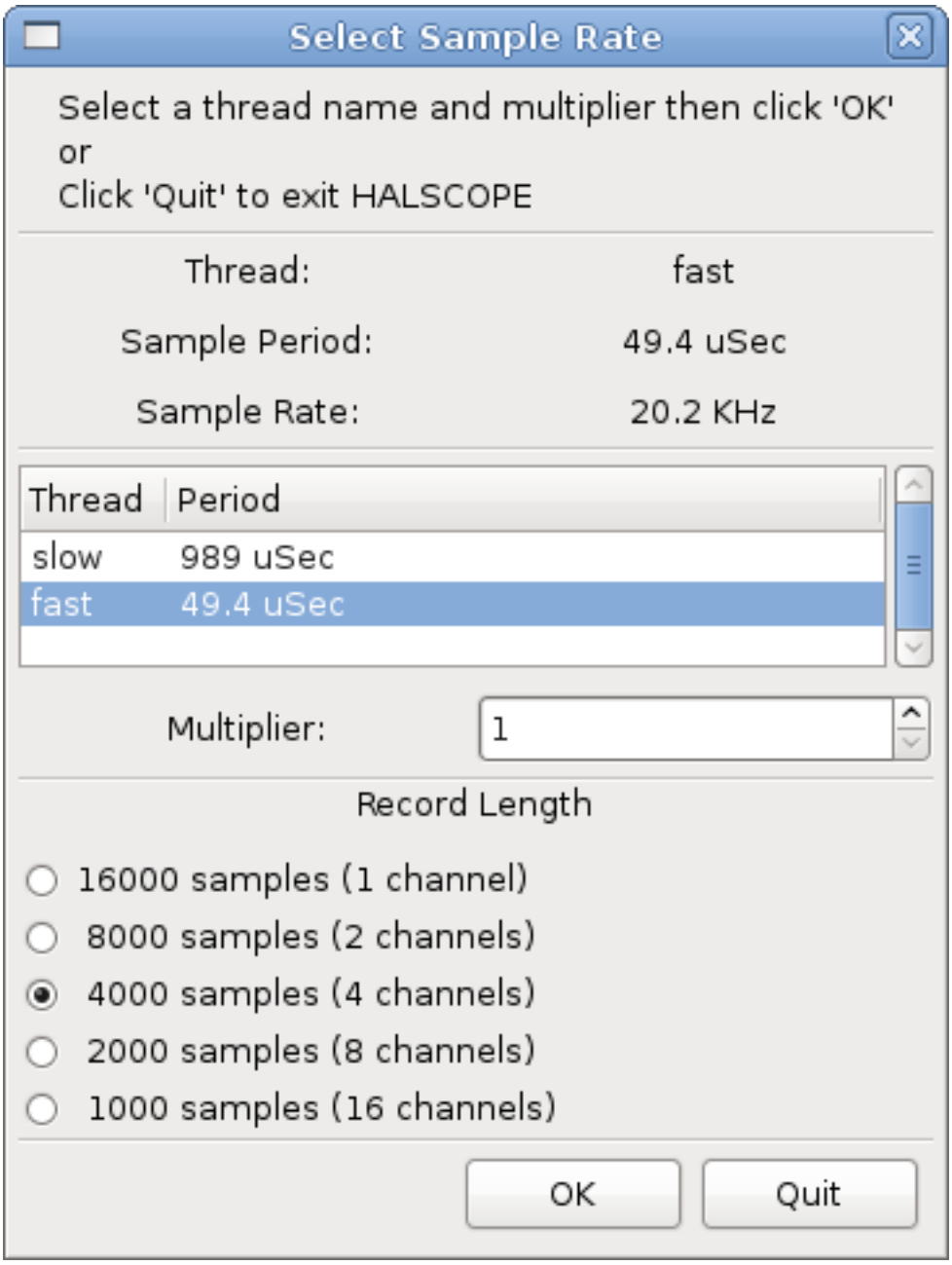


Abbildung 5.16: Dialogfeld für Abtastrate

5.4.6.6 Weitere Kanäle

Now let's look at the step pulses. Halscope has 16 channels, but for this example we are using only 4 at a time. Before we select any more channels, we need to turn off a couple. Click on the channel 2 button, then click the *Chan Off* button at the bottom of the *Vertical* box. Then click on channel 3, turn if off, and do the same for channel 4. Even though the channels are turned off, they still remember what they are connected to, and in fact we will continue to use channel 3 as the trigger source. To add new channels, select channel 5, and choose pin `stepgen.0.dir`, then channel 6, and select `stepgen.0.step`. Then click run mode *Normal* to start the scope, and adjust the horizontal zoom to 5 ms per division. You should see the step pulses slow down as the velocity command (channel 1) approaches zero, then the direction pin changes state and the step pulses speed up again. You might want toincrease the gain on

channel 1 to about 20 milli per division to better see the change in the velocity command. The result should look like the following figure.

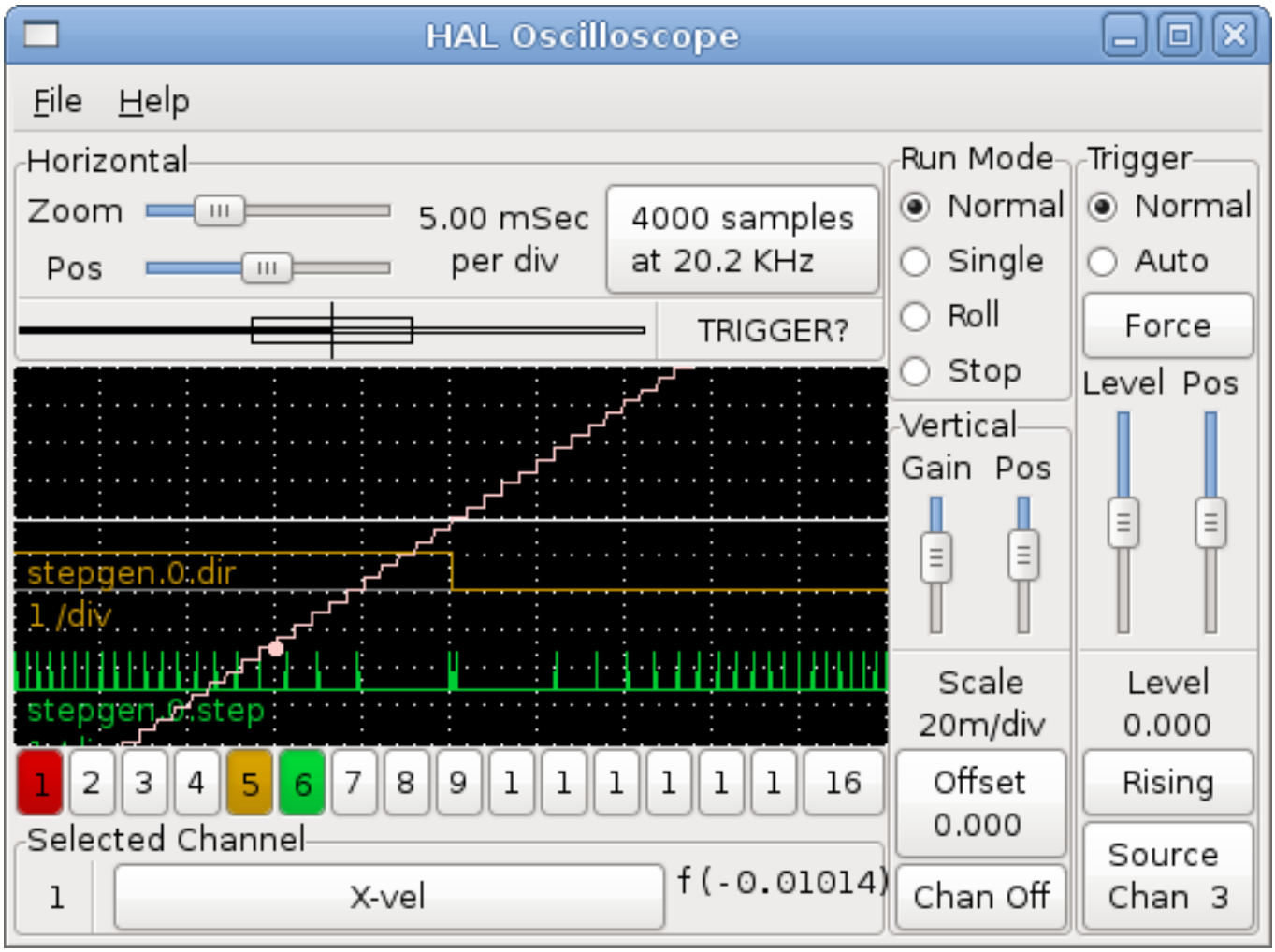


Abbildung 5.17: Schrittimpulse

5.4.6.7 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).

5.5 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.

5.5.1 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

```
<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>
```

The postgui.hal file

```
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 Axis-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

```
halcmd show pin or2
Component Pins:
Owner   Type  Dir         Value  Name
    22  bit   IN          FALSE  or2.0.in0 <== button-1
    22  bit   IN          FALSE  or2.0.in1 <== button-2
    22  bit   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.

5.5.2 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 die hal_manualtoolchange-Komponente laden, dann die iocontrol-Komponente ”tool change” an die hal_manualtoolchange-Komponente ”change” senden und die hal_manualtoolchange-Komponente ”changed” an die iocontrol-Komponente ”tool changed” zurücksenden. Ein Pin ist für einen externen Eingang vorgesehen, um anzuzeigen, 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
```

5.5.3 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 (Abschnitt 5.1.3).

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 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.

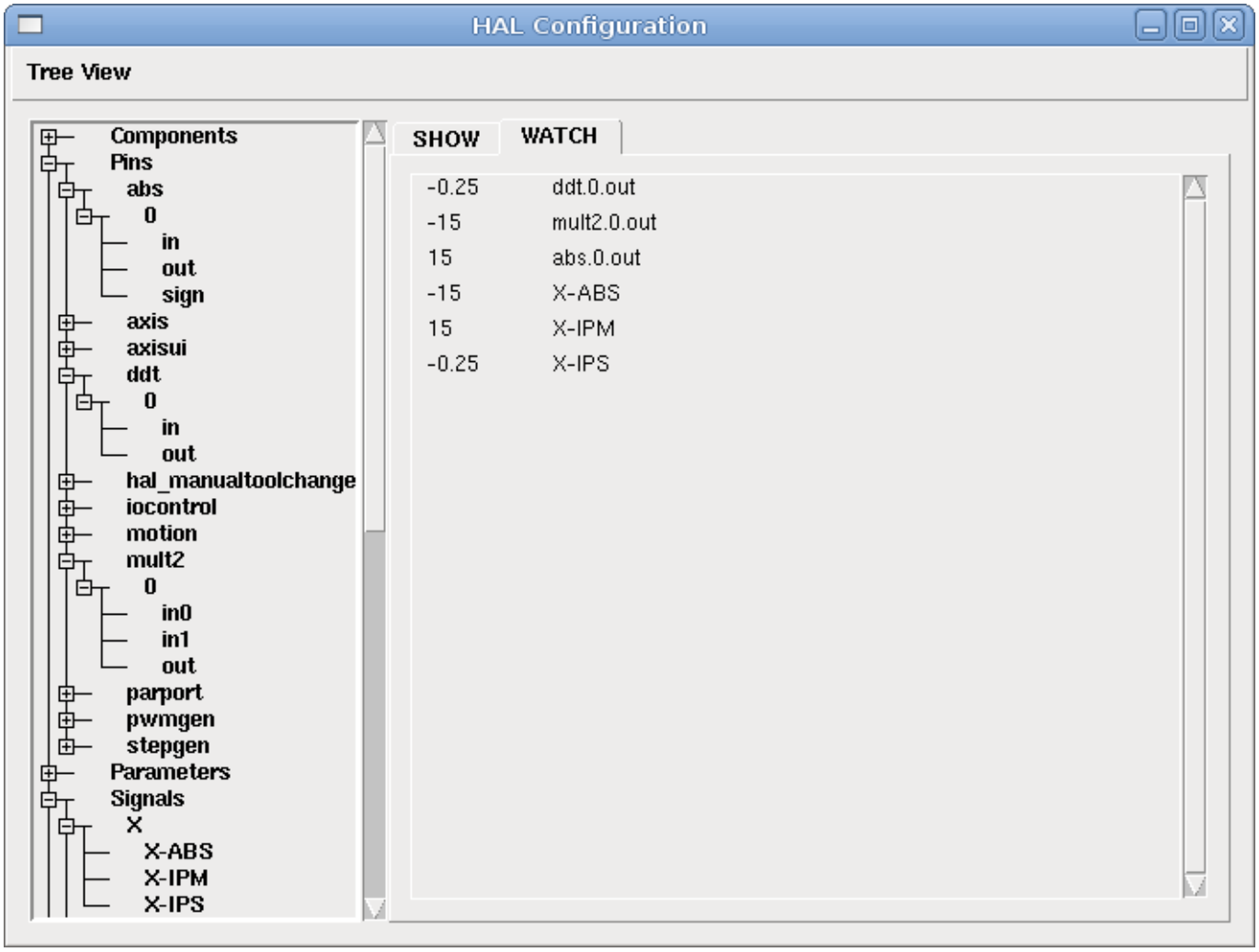


Abbildung 5.18: HAL: Geschwindigkeitsbeispiel

5.5.4 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 Spindel.N.speed-out durch einen Begrenzer vor dem PID senden, so dass der PID-Befehlswert langsamer auf neue Einstellungen wechselt.

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
```

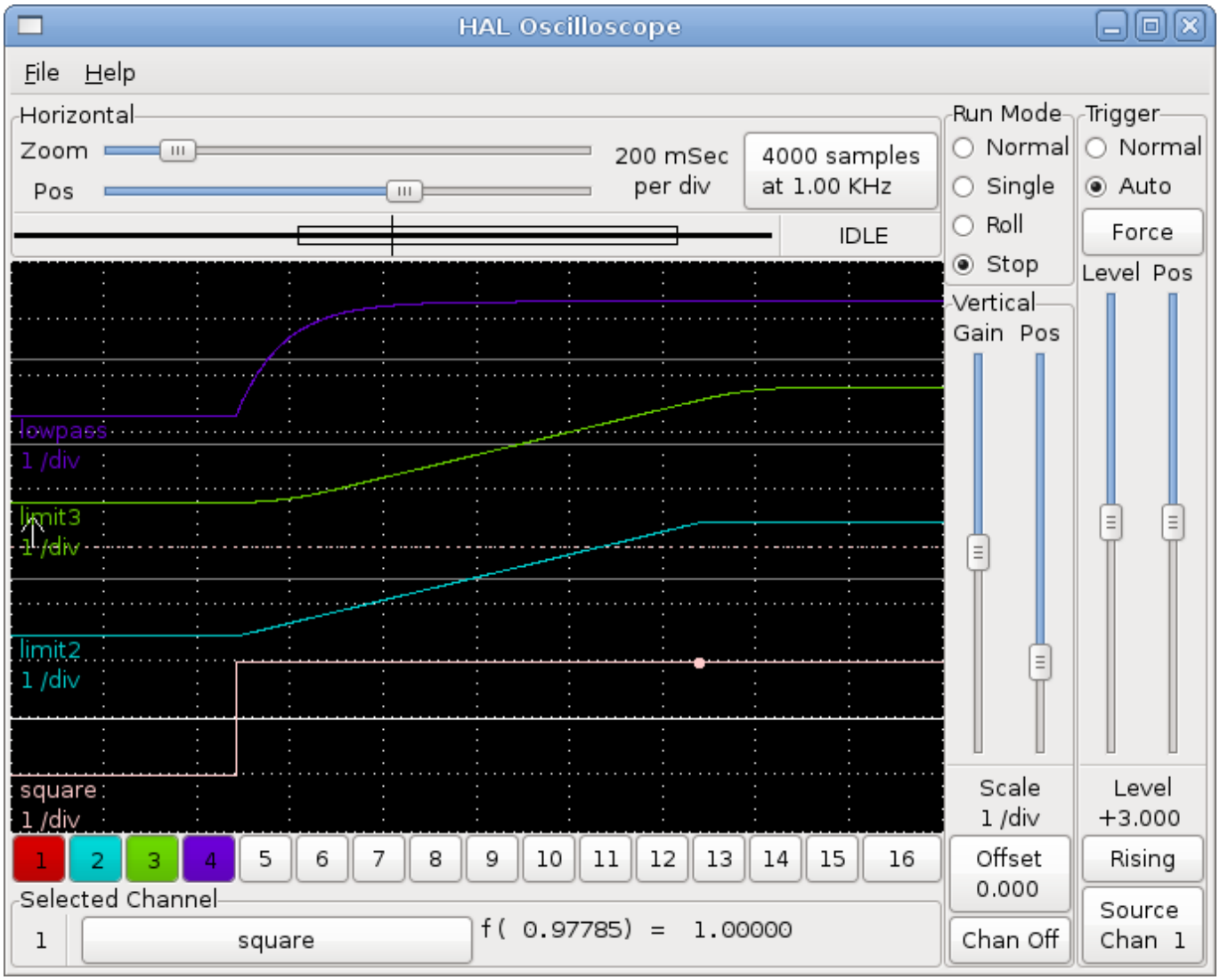
Wenn das HAL Oszilloskop zum ersten Mal startet, klicken Sie auf *OK*, um den Standardfaden zu akzeptieren.

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.

5.5.5 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
```

5.6 Kernkomponenten

Siehe auch die Man Pages zu *motion(9)*.

5.6.1 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.

Anmerkung
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 us) 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.

5.6.1.1 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
```

5.6.1.2 Pins

Diese Pins, Parameter und Funktionen werden durch das Echtzeitmodul *motmod* angelegt.

- *motion.adaptive-feed* - (float, in) Wenn der adaptive Vorschub mit *M52 P1* aktiviert ist, wird die befohlene Geschwindigkeit mit diesem Wert multipliziert. Dieser Effekt ist multiplikativ mit dem Vorschub-Override-Wert auf NML-Ebene und *motion.feed-hold*. Ab der Version 2.9 von LinuxCNC ist es möglich, einen negativen adaptiven Vorschubwert zu verwenden, für eine G-Code-Bahn in umgekehrter Richtung.
- *motion.analog-in-00* - (float, in) Diese Pins (00, 01, 02, 03 oder mehr, falls konfiguriert) werden von M66 gesteuert.
- *motion.analog-out-00* - (float, out) Diese Pins (00, 01, 02, 03 oder mehr, falls konfiguriert) werden von M67 oder M68 gesteuert.
- *motion.coord-error* - (bit, out) TRUE, wenn bei der Bewegung ein Fehler aufgetreten ist, z. B. das Überschreiten eines Softlimit (einer ”weichen Grenze”)
- *motion.coord-mode* - (bit, out) TRUE, wenn die Bewegung im *koordinierten Modus* ist, im Gegensatz zum *Teleop-Modus*
- *motion.current-vel* – (float, out) Die aktuelle Werkzeuggeschwindigkeit in Benutzereinheiten pro Sekunde.
- *motion.digital-in-00* - (bit, in) Diese Pins (00, 01, 02, 03 oder mehr, falls konfiguriert) werden von M62-65 gesteuert.
- *motion.digital-out-00* - (bit, out) Diese Pins (00, 01, 02, 03 oder mehr, falls konfiguriert) werden von der *M62-65* gesteuert.
- *motion.distance-to-go* - (float,out) Die verbleibende Distanz der aktuellen Bewegung.
- *motion.enable* - (bit, in) Wenn dieses Bit auf FALSE gesetzt wird, stoppt die Bewegung, die Maschine wird in den *machine off* Zustand versetzt und eine Meldung für den Bediener wird angezeigt. Für eine normale Bewegung muss dieses Bit auf TRUE gesetzt werden.
- *motion.feed-hold* - (Bit, in) Wenn die Vorschub-Stopp-Steuerung mit *M53 P1* aktiviert ist und dieses Bit TRUE ist, wird die Vorschubgeschwindigkeit auf 0 gesetzt.
- *motion.feed-inhibit* - (bit, in) Wenn dieses Bit TRUE ist, wird der Vorschub auf 0 gesetzt. Dies wird bei Spindelsynchronisationsbewegungen bis zum Ende der Bewegung verzögert.
- *motion.in-position* - (bit, out) TRUE wenn die Maschine in Position ist.
- *motion.motion-enabled* - (bit, out) TRUE wenn im *machine on* Zustand.
- *motion.motion-type* - (s32, out) Diese Werte sind aus src/emc/nml_intf/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* - (bit, out) TRUE, wenn sich die Maschine an einem Softlimit (buchstäblich auch: einer weichen Grenze) befindet.
- *motion.probe-input* - (bit, in) *G38.n* verwendet den Wert an diesem Pin, um den Moment zu bestimmen, in dem der Taster Kontakt hergestellt hat. TRUE für Tasterkontakt geschlossen (berührend), FALSE für Tasterkontakt offen.

- *motion.program-line* - (s32, out) Die aktuelle Programmzeile während der Ausführung. Null, wenn das Programm nicht läuft oder zwischen den Zeilen bei Einzelschritten.
- *motion.requested-vel* - (float, out) Die aktuell geforderte Geschwindigkeit in Benutzereinheiten pro Sekunde. Dieser Wert ist die F-Wort-Einstellung aus der G-Code-Datei, möglicherweise reduziert, um die Geschwindigkeits- und Beschleunigungsgrenzen der Maschine zu berücksichtigen. Der Wert an diesem Pin spiegelt nicht den Vorschub-Override oder andere Anpassungen wider.
- *motion.teleop-mode* - (bit, out) TRUE wenn die Bewegung im *teleop mode* (Fernsteuerungs-Modus) ist, im Gegensatz zum *coordinated mode* (Koordinaten-Modus)
- *motion.tooloffset.x … motion.tooloffset.w* - (float, out, einer pro Achse) zeigt den aktuellen Werkzeugversatz an; er kann aus der Werkzeugtabelle (*G43* aktiv) oder aus dem G-Code (*G43.1* aktiv) stammen
- *motion.on-soft-limit* - (bit, out) TRUE, wenn sich die Maschine an einem Softlimit (buchstäblich auch: einer weichen Grenze) befindet.
- *motion.probe-input* - (bit, in) *G38.n* verwendet den Wert an diesem Pin, um den Moment zu bestimmen, in dem der Taster Kontakt hergestellt hat. TRUE für Tasterkontakt geschlossen (berührend), FALSE für Tasterkontakt offen.
- *motion.program-line* - (s32, out) Die aktuelle Programmzeile während der Ausführung. Null, wenn das Programm nicht läuft oder zwischen den Zeilen bei Einzelschritten.
- *motion.requested-vel* - (float, out) Die aktuell geforderte Geschwindigkeit in Benutzereinheiten pro Sekunde. Dieser Wert ist die F-Wort-Einstellung aus der G-Code-Datei, möglicherweise reduziert, um die Geschwindigkeits- und Beschleunigungsgrenzen der Maschine zu berücksichtigen. Der Wert an diesem Pin spiegelt nicht den Vorschub-Override oder andere Anpassungen wider.
- *motion.teleop-mode* - (bit, out) TRUE wenn die Bewegung im *teleop mode* (Fernsteuerungs-Modus) ist, im Gegensatz zum *coordinated mode* (Koordinaten-Modus)
- *motion.tooloffset.x … motion.tooloffset.w* - (float, out, einer pro Achse) zeigt den aktuellen Werkzeugversatz an; er kann aus der Werkzeugtabelle (*G43* aktiv) oder aus dem G-Code (*G43.1* aktiv) stammen

5.6.1.3 Parameter

Viele dieser Parameter dienen als Hilfsmittel zur Fehlersuche und können jederzeit geändert oder entfernt werden.

- *motion-command-handler.time* - (s32, RO)
- *motion-command-handler.tmax* - (s32, RW)
- *motion-controller.time* - (s32, RO)
- *motion-controller.tmax* - (s32, RW)
- *motion.debug-bit-0* - (bit, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-bit-1* - (bit, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-float-0* - (float, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-float-1* - (float, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-float-2* - (float, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-float-3* - (float, RO) Dies wird zur Fehlersuche verwendet.

- *motion.debug-s32-0* - (s32, RO) Dies wird zur Fehlersuche verwendet.
- *motion.debug-s32-1* - (s32, RO) Dies wird zur Fehlersuche verwendet.
- *motion.servo.last-period* - (u32, RO) Die Anzahl der CPU-Zyklen zwischen den Aufrufen des Servo-Threads. Normalerweise ergibt diese Zahl geteilt durch die CPU-Geschwindigkeit die Zeit in Sekunden und kann verwendet werden, um festzustellen, ob der Echtzeit-Bewegungsregler seine Zeitvorgaben einhält
- *motion.servo.last-period-ns* - (float, RO)

5.6.1.4 Funktionen

Im Allgemeinen werden diese beiden Funktionen in der angegebenen Reihenfolge zum Servo-Thread hinzugefügt.

- *motion-command-handler* - Verarbeitet Bewegungsbefehle aus dem Benutzerbereich
- *motion-controller* - Führt die LinuxCNC Bewegungssteuerung aus

5.6.2 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.

5.6.2.1 Pins

- *spindle*.N.*at-speed* - (bit, in) Die Bewegung wird unter den folgenden Bedingungen angehalten, bis dieser Pin TRUE ist:
  - 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* - (bit, out) TRUE, wenn die Spindelbremse aktiviert werden soll.
- *spindle*.N.*forward* - (bit, out) TRUE wenn sich die Spindel vorwärts drehen soll.
- *spindle*.N.*index-enable* - (Bit, I/O) Für den korrekten Betrieb von spindelsynchronisierten Bewegungen muss dieser Pin mit dem Index-Enable-Pin des Spindelgebers verbunden werden.
- *spindle*.N.*inhibit* - (bit, in) Wenn dieses Bit TRUE ist, wird die Spindeldrehzahl auf 0 gesetzt.
- *spindle*.N.*on* - (bit, out) TRUE wenn sich die Spindel drehen soll.
- *spindle*.N.*reverse* - (bit, out) TRUE wenn sich die Spindel rückwärts drehen soll
- *spindle*.N.*revs* - (float, in) Für den korrekten Betrieb von spindelsynchronisierten Bewegungen muss dieses Signal mit dem Positionspin des Spindelgebers verbunden werden. Die Position des Spindelgebers sollte so skaliert werden, dass die Spindelumdrehungen bei jeder Umdrehung der Spindel im Uhrzeigersinn (*M3*) um 1,0 zunehmen.

- *spindle*.N.*speed-in* - (float, in) Rückmeldung der tatsächlichen Spindeldrehzahl in Umdrehungen pro Sekunde. Dies wird von der Bewegung mit Vorschub pro Umdrehung (*G95*) verwendet. Wenn Ihr Spindelgeber-Treiber nicht über einen Geschwindigkeitsausgang verfügt, können Sie einen geeigneten Ausgang erzeugen, indem Sie die Spindelposition durch eine *ddt* Komponente senden. Wenn Sie keinen Spindelgeber haben, können Sie *spindle*.N.*speed-out-rps* durchschleifen.
- *spindle*.N.*speed-out* - (float, out) Befohlene Spindeldrehzahl in Umdrehungen pro Minute. Positiv für Spindel vorwärts (*M3*), negativ für Spindel rückwärts (*M4*).
- *spindle*.N.*speed-out-abs* - (Float, out) Geforderte Spindeldrehzahl in Umdrehungen pro Minute. Dies ist immer eine positive Zahl.
- *spindle*.N.*speed-out-rps* - (float, out) Geforderte Spindeldrehzahl in Umdrehungen pro Sekunde. Positiv für Spindel vorwärts (*M3*), negativ für Spindel rückwärts (*M4*).
- *spindle*.N.*speed-out-rps-abs* - (float, out) Befohlene Spindeldrehzahl in Umdrehungen pro Sekunde. Dies ist immer eine positive Zahl.
- *spindle*.N.*orient-angle* - (float,out) Gewünschte Spindelausrichtung für M19. Wert des M19 R-Wort-Parameters plus dem Wert des [RS274NGC]ORIENT_OFFSET INI-Parameters.
- *spindle*.N.*orient-mode* - (s32,out) Gewünschter Rotationsmodus der Spindel M19. Voreinstellung 0.
- *spindle*.N.*orient* - (out,bit) Zeigt den Beginn des Spindelorientierungszyklus an. Wird von M19 gesetzt. Wird gelöscht durch M3, M4 oder M5. Wenn spindle-orient-fault während spindle-orient true nicht Null ist, schlägt der Befehl M19 mit einer Fehlermeldung fehl.
- *spindel*.N.*is-oriented* - (in, bit) Bestätigungs-(engl. acknowledge)-Pin für ”spindle-orient”. Schließt den Orientierungszyklus ab. Wenn ”spindle-orient” wahr war, als ”spindle-is-oriented” bestätigt wurde, wird der Pin ”spindle-orient” gelöscht und der Pin ”spindle-locked” bestätigt. Außerdem wird der Pin ”spindle-brake” aktiviert.
- *spindle*.N.*orient-fault* - (s32, in) Fehlercodeeingabe für den Orientierungszyklus. Jeder Wert ungleich Null führt zum Abbruch des Orientierungszyklus.
- *spindle*.N.*lock* - (bit, out) Spindel-Orientierung-durchgeführt-Pin. Zurückgesetzt durch M3, M4 oder 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.

5.6.3 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)*.

5.6.4 iocontrol

iocontrol - nimmt NML-E/A-Befehle entgegen, interagiert mit HAL im Userspace.

Die Signale werden im Userspace ein- und ausgeschaltet - wenn Sie strenge Timing-Anforderungen haben oder einfach mehr E/A benötigen, sollten Sie stattdessen die von Motion bereitgestellte echtzeitsynchronisierte E/A verwenden.

5.6.4.1 Pins

- *iocontrol.0.coolant-flood* - (bit, out) TRUE wenn Kühlmittelflut angefordert wird.
- *iocontrol.0.coolant-mist* - (bit, out) TRUE wenn Kühlmittelnebel angefordert wird.
- *iocontrol.0.emc-enable-in* - (Bit, in) Sollte FALSE sein, wenn eine externe Not-Aus-Bedingung vorliegt.
- *iocontrol.0.lube* - (bit, out) TRUE wenn lube befohlen wird.
- *iocontrol.0.lube_level* - (bit, in) Sollte auf TRUE gesetzt werden, wenn der Schmiermittelstand hoch genug ist.
- *iocontrol.0.tool-change* - (bit, out) TRUE wenn ein Werkzeugwechsel angefordert wird.
- *iocontrol.0.tool-changed* - (bit, in) Sollte TRUE sein, wenn ein Werkzeugwechsel abgeschlossen ist.
- *iocontrol.0.tool-number* - (s32, out) Die aktuelle Werkzeugnummer.
- *iocontrol.0.tool-prep-number* - (s32, out) Die Nummer des nächsten Werkzeugs, aus dem RS274NGC T-Wort.

- *iocontrol.0.tool-prepare* - (bit, out) TRUE wenn eine Werkzeugvorbereitung angefordert wird.
- *iocontrol.0.tool-prepared* - (bit, in) Sollte auf TRUE gesetzt werden, wenn eine Werkzeugvorbereitung abgeschlossen ist.
- *iocontrol.0.user-enable-out* - (bit, out) FALSE, wenn eine interne Not-Aus-Bedingung vorliegt.
- *iocontrol.0.user-request-enable* - (bit, out) TRUE, wenn der Benutzer die Freigabe des Notausschalters angefordert hat.

5.6.5 INI-Einstellungen

Eine Reihe von INI-Einstellungen werden als HAL Eingangspins zur Verfügung gestellt.

5.6.5.1 Pins

N bezieht sich auf eine Gelenknummer, *L* auf einen Achsenbuchstaben.

- ”ini.*N*.ferror’ - (float, in) [JOINT_N]FERROR
- *ini*.N.*min_ferror* - (float, in) [JOINT_N]MIN_FERROR
- *ini*.N.*backlash* - (float, in) [JOINT_N]BACKLASH
- *ini*.N.*min_limit* - (float, in) [JOINT_N]MIN_LIMIT
- *ini*.N.*max_limit* - (float, in) [JOINT_N]MAX_LIMIT
- *ini*.N.*max_velocity* - (float, in) [JOINT_N]MAX_VELOCITY
- *ini*.N.*max_acceleration* - (float, in) [JOINT_N]MAX_ACCELERATION
- *ini*.N.*home* - (float, in) [JOINT_N]HOME
- *ini*.N.*home_offset* - (float, in) [JOINT_N]HOME_OFFSET
- *ini*.N.*home_offset* - (s32, in) [JOINT_N]HOME_SEQUENCE
- *ini*.L.*min_limit* - (float, in) [AXIS_L]MIN_LIMIT
- *ini*.L.*max_limit* - (float, in) [AXIS_L]MAX_LIMIT
- *ini*.L.*max_velocity* - (float, in) [AXIS_L]MAX_VELOCITY
- *ini*.L.*max_acceleration* - (float, in) [AXIS_L]MAX_ACCELERATION

Anmerkung
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* - (bit, in) [TRAJ]ARC_BLEND_ENABLE
- *ini.traj_arc_blend_fallback_enable* - (bit, in) [TRAJ]ARC_BLEND_FALLBACK_ENABLE

- *ini.traj_arc_blend_gap_cycles* - (float, in) [TRAJ]ARC_BLEND_GAP_CYCLES
- *ini.traj_arc_blend_optimization_depth* - (float, in) [TRAJ]ARC_BLEND_OPTIMIZATION_DEPTH
- *ini.traj_arc_blend_ramp_freq* - (float, in) [TRAJ]ARC_BLEND_RAMP_FREQ

Anmerkung
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* - (float, in) [TRAJ]DEFAULT_ACCELERATION
- *ini.traj_default_velocity* - (float, in) [TRAJ]DEFAULT_VELOCITY
- *ini.traj_max_acceleration* - (float, in) [TRAJ]MAX_ACCELERATION

5.7 HAL Component List

5.7.1 Komponenten

Für die meisten der in der folgenden Liste aufgeführten Befehle gibt es Man Pages. Für einige gibt es erweiterte Beschreibungen, für andere eingeschränkte Beschreibungen. Anhand dieser Liste wissen Sie, welche Komponenten existieren, und Sie können ”man name” verwenden, um zusätzliche Informationen zu erhalten. Um die Informationen in der man-Seite zu sehen, geben Sie in einem Terminalfenster ein:

```
man axis (oder vielleicht 'man <n> axis', wenn Ihr System dies erfordert. n = 1 für ↩
    Userspace und 9 für Echtzeitkomponenten)
```

Anmerkung
Siehe auch den Abschnitt *Man Pages* unter dem Link:../index.html[docs main page] oder den Link:../man/[directory listing].

5.7.1.1 Benutzerschnittstellen (Userspace)

axis	AXIS LinuxCNC (ehemals ”Enhanced Machine Controller”) GUI
axis-remote	AXIS 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	Kurzbeschreibung
pyngcgui	Python-Implementierung von NGCGUI
touchy	AXIS - TOUCHY LinuxCNC Grafische Benutzeroberfläche

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

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
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

5.7.1.2 Bewegung (engl. motion) (Userspace)

io	iocontrol - interagiert mit HAL oder G-Code im Userspace
iocontrol	Interagiert mit HAL oder G-Code im Userspace
iov2	Interagiert mit HAL oder G-Code im Userspace
mdi	Senden von G-Code-Befehlen vom Terminal an die laufende LinuxCNC-Instanz
milltask	Userspace Task Controller für LinuxCNC
motion	Akzeptiert NML-Bewegungsbefehle, interagiert mit HAL in Echtzeit

5.7.1.3 Hardware-Treiber

elbpcom	Kommunikation mit Mesa-Ethernet-Karten
gs2_vfd	HAL Userspace Komponente für Automation Direct GS2 VFDs
hal_parport	Echtzeit-HAL-Komponente zur Kommunikation mit einer oder mehreren parallelen PC-Ports
hy_gt_vfd	HAL Userspace Komponente für Huanyang GT-Serien VFDs
hy_vfd	HAL Userspace Komponente für Huanyang VFDs
mb2hal	MB2HAL ist eine generische Userspace-HAL-Komponente zur Kommunikation mit einem oder mehreren Modbus-Geräten. Modbus RTU und Modbus TCP werden unterstützt.
mitsub_vfd	HAL Userspace-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 Userspace Komponente für SVD-P(S) VFDs

vfdb_vfd	HAL-Userspace-Komponente für Delta VFD-B Frequenzumrichter
vfs11_vfd	HAL-Userspace-Komponente für Toshiba-Schneider VF-S11 Frequenzumrichter
wj200_vfd	Hitachi WJ200 Modbus-Treiber
xhc-hb04	User-Space-HAL-Komponente für das xhc-hb04-Pendant
xhc-hb04-accels	Obsolete script for jogging wheel
xhc-whb04b-6	Userspace jog dial HAL component for the wireless XHC WHB04B-6 USB device

5.7.1.4 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
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
opto_ac5	Echtzeittreiber für opto22 PCI-AC5 Karten
pluto_servo	Pluto-P Treiber und Firmware für den Parallelport FPGA, für Servos
pluto_step	Pluto-P Treiber für den Parallelport FPGA, für Stepper
serport	Hardwaretreiber für die digitalen E/A-Bits der seriellen Schnittstelle des 8250 und 16550
sserial	hostmot2 - Smart Serial LinuxCNC HAL Treiber für die Mesa Electronics HostMot2 Smart-Serial Remote Karten
thc	Brennerhöhensteuerung mit einer Mesa THC-Karte oder einem beliebigen Analog-/Geschwindigkeitseingang

5.7.1.5 Dienstprogramme (Benutzerbereich)

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	Kurzbeschreibung
halreport	Erzeugt einen Bericht über den Status des HAL
halrmt	Kurzbeschreibung
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
latency-histogram	Plottet Histogramm der Maschinenlatenz
latency-plot	Eine weitere Möglichkeit, Latenzzahlen anzuzeigen
latency-test	Testen der Latenzzeit des Echtzeitsystems
pncconf	Konfigurationsassistent für Mesa-Karten
setsserial	Dienstprogramm zur Einstellung der Smart Serial NVRAM-Parameter. NOTE: Dieses recht klobige Dienstprogramm wird nicht mehr benötigt, außer zum Flashen neuer Smart-Serial-Remote-Firmware. Smart-Serial-Remote-Parameter können jetzt auf normale Weise in der HAL-Datei gesetzt werden.
sim_pin	GUI for displaying and setting one or more HAL inputs
stepconf	Ein Konfigurationsassistent für Maschinen mit parallelen Anschlüssen

5.7.1.6 Signalverarbeitung (Echtzeit)

and2	UND-Gatter mit zwei Eingängen. Damit der Ausgang wahr ist, müssen beide Eingänge wahr sein. (Link:../man/man9/and2.9.html[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
edge	Kanten-Detektor
estop_latch	E-stop latch
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
select8	8-Bit-Binär-Match-Detektor
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 - generate a high pulse of defined duration on rising edge
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)

abs_s32	Berechnet den Absolutwert und das Vorzeichen des Eingangssignals
abs	Berechnet den Absolutwert und das Vorzeichen des Eingangssignals
biquad	Biquad IIR-Filter
blend	Lineare Interpolation zwischen zwei Werten durchführen
comp	Komparator mit zwei Eingängen und Hysterese
constant	Verwendet einen Parameter, um den Wert eines Pins festzulegen
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.
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	Konvertiert die Anzahl (wahrscheinlich von einem Encoder) in einen Gleitkommawert.
lowpass	Tiefpassfilter
limit1	Begrenzt das Ausgangssignal auf einen Wert zwischen min und max. [2]
limit2	Begrenzt das Ausgangssignal auf einen Wert zwischen min und max. Begrenzung der 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	Auswahl aus einem von sechzehn Eingangswerten (multiplexer).
near	Bestimmt, ob zwei Werte annähernd gleich sind.
offset	Fügt einer Eingabe einen Offset hinzu und subtrahiert ihn vom Feedbackwert.
sample_hold	Probenahme und Halten.
scale	Wendet eine Skalierung und einen Offset auf seinen Eingang an.
sum2	Summe aus zwei Eingängen (jeweils mit einem Verstärkungsfaktor) und einem Offset.
timedelta	Komponente, die das Zeitverhalten bei der Thread-Planung misst.
updown	Zählt aufwärts oder abwärts, mit optionalen Grenzen und Wraparound-Verhalten.
wcomp	Fenster-Komparator.
weighted_sum	Gewichtete Summe, konvertiert eine Gruppe von Bits in eine ganze Zahl.
xhc_hb04_util	xhc-hb04 Komfort-Dienstprogramm

bin2gray	Konvertierung einer Zahl in die Gray-Code Repräsentation
bitslice	Konvertiert eine vorzeichenlose 32-Eingabe in einzelne Bits

[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 die ”Position”, ”Geschwindigkeit” und ”Beschleunigung” begrenzt sind

conv_bit_float	Konvertiert von bit in float.
conv_bit_s32	Konvertiert von bit nach s32.
conv_bit_u32	Konvertiert von bit nach u32.
conv_float_s32	Konvertiert von float nach s32.
conv_float_u32	Konvertiert von float nach u32
conv_s32_bit	Konvertiert von s32 in Bit
conv_s32_float	Konvertiert von s32 in float
conv_s32_u32	Konvertiert von s32 nach u32
conv_u32_bit	Konvertiert von u32 in Bit
conv_u32_float	Konvertiert von u32 in float
conv_u32_s32	Konvertiert von u32 nach s32
gray2bin	Konvertiert Gray-Code-Eingabe in Binärformat

5.7.1.7 Kinematiken (Echtzeit)

corexy_by_hal	CoreXY-Kinematiken
differential	Kinematik für ein Differentialgetriebe
gantry	LinuxCNC HAL component for driving multiple joints from a single axis
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	See trivkins
kins	Kinematik Definitionen für LinuxCNC.
lineardeltakins	Kinematik für pumaähnliche Roboter
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. Die Quelldatei, maxkins.c, kann ein nützlicher Ausgangspunkt für andere 5-Achsen-Systeme sein.
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
tripodkins	Die Gelenke stellen den Abstand des kontrollierten Punktes von drei vordefinierten Orten (den Motoren) dar, was drei Freiheitsgrade in der Position (XYZ) ergibt
trivkins	1:1-Entsprechung zwischen Gelenken und Achsen. Die meisten Standardfräs- und -drehmaschinen verwenden das triviale Kinematikmodul.
userkins	Vorlage für benutzerdefinierte Kinematiken

5.7.1.8 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 (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.

5.7.1.9 Sonstiges (Echtzeit)

comp Erstellen, kompilieren und installieren Sie LinuxCNC HAL Komponenten.
classicladder Echtzeit-Software-SPS (engl. PLC), die auf Kontaktplan-Logik basiert. Siehe Kapitel ClassicLadder für weitere Informationen.
threads Erzeugt harte Echtzeit-HAL-Threads.
charge_pump Erzeugt eine Rechteckwelle für den *Charge Pump*-Eingang einiger Controller-Boards. Die Funktion "Charge Pump" sollte zur Basis-Thread-Funktion hinzugefügt werden. Wenn sie aktiviert ist, dann ist der Ausgang eine Periode lang an und eine Periode lang aus. Zur Berechnung der Frequenz des Ausgangs wird 1/(Periodenzeit in Sekunden x 2) = Hz verwendet. Wenn Sie zum Beispiel eine Basisperiode von 100.000 ns haben, ist das 0,0001 Sekunden und die Formel wäre 1/(0,0001 x 2) = 5.000 Hz oder 5 kHz.
encoder_ratio Elektronisches Getriebe zur Synchronisierung zweier Achsen.
feedcomp Multipliziert die Eingabe mit dem Verhältnis von aktueller Geschwindigkeit und Vorschubgeschwindigkeit.
GladeVCP (Echtzeit) zeigt mit GTK/Glade erstellte virtuelle Kontrollfelder an
gearchange Wählt einen von zwei Geschwindigkeitsbereichen aus.
joyhandle Setzt nichtlineare Joypad-Bewegungen, Deadbands und Skalen.
sampler Probendaten von HAL in Echtzeit.
siggen Signal generator, see Description.
sim_encoder Simulated quadrature encoder, see Description.
sphereprobe Sondieren einer angenommenen Hemisphäre.
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).
threadtest Komponente zum Testen des Threadverhaltens.
time Kumulierte Laufzeit des Timers zählt HH:MM:SS des *aktiven* Eingangs.
watchdog Überwachen Sie einen bis zweiunddreißig Eingänge auf einen „Herzschlag".

5.7.2 HAL-API-Aufrufe

```
hal_add_funct_to_thread.3hal
hal_bit_t.3hal
hal_create_thread.3hal
hal_del_funct_from_thread.3hal
hal_exit.3hal
hal_export_funct.3hal
hal_float_t.3hal
hal_get_lock.3hal
hal_init.3hal
hal_link.3hal
hal_malloc.3hal
hal_param_bit_new.3hal
hal_param_bit_newf.3hal
hal_param_float_new.3hal
hal_param_float_newf.3hal
hal_param_new.3hal
hal_param_s32_new.3hal
```

hal_param_s32_newf.3hal
hal_param_u32_new.3hal
hal_param_u32_newf.3hal
hal_parport.3hal
hal_pin_bit_new.3hal
hal_pin_bit_newf.3hal
hal_pin_float_new.3hal
hal_pin_float_newf.3hal
hal_pin_new.3hal
hal_pin_s32_new.3hal
hal_pin_s32_newf.3hal
hal_pin_u32_new.3hal
hal_pin_u32_newf.3hal
hal_ready.3hal
hal_s32_t.3hal
hal_set_constructor.3hal
hal_set_lock.3hal
hal_signal_delete.3hal
hal_signal_new.3hal
hal_start_threads.3hal
hal_type_t.3hal
hal_u32_t.3hal
hal_unlink.3hal
intro.3hal
undocumented.3hal

5.7.3 RTAPI-Aufrufe

EXPORT_FUNCTION.3rtapi
MODULE_AUTHOR.3rtapi
MODULE_DESCRIPTION.3rtapi
MODULE_LICENSE.3rtapi
RTAPI_MP_ARRAY_INT.3rtapi
RTAPI_MP_ARRAY_LONG.3rtapi
RTAPI_MP_ARRAY_STRING.3rtapi
RTAPI_MP_INT.3rtapi
RTAPI_MP_LONG.3rtapi
RTAPI_MP_STRING.3rtapi
intro.3rtapi
rtapi_app_exit.3rtapi
rtapi_app_main.3rtapi
rtapi_clock_set_period.3rtapi
rtapi_delay.3rtapi
rtapi_delay_max.3rtapi
rtapi_exit.3rtapi
rtapi_get_clocks.3rtapi
rtapi_get_msg_level.3rtapi
rtapi_get_time.3rtapi
rtapi_inb.3rtapi
rtapi_init.3rtapi
rtapi_module_param.3rtapi
RTAPI_MP_ARRAY_INT.3rtapi
RTAPI_MP_ARRAY_LONG.3rtapi
RTAPI_MP_ARRAY_STRING.3rtapi
RTAPI_MP_INT.3rtapi

```
RTAPI_MP_LONG.3rtapi
RTAPI_MP_STRING.3rtapi
rtapi_mutex.3rtapi
rtapi_outb.3rtapi
rtapi_print.3rtap
rtapi_prio.3rtapi
rtapi_prio_highest.3rtapi
rtapi_prio_lowest.3rtapi
rtapi_prio_next_higher.3rtapi
rtapi_prio_next_lower.3rtapi
rtapi_region.3rtapi
rtapi_release_region.3rtapi
rtapi_request_region.3rtapi
rtapi_set_msg_level.3rtapi
rtapi_shmem.3rtapi
rtapi_shmem_delete.3rtapi
rtapi_shmem_getptr.3rtapi
rtapi_shmem_new.3rtapi
rtapi_snprintf.3rtapi
rtapi_task_delete.3rtpi
rtapi_task_new.3rtapi
rtapi_task_pause.3rtapi
rtapi_task_resume.3rtapi
rtapi_task_start.3rtapi
rtapi_task_wait.3rtapi
```

5.8 Beschreibungen der HAL-Komponenten

5.8.1 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 den Positionsmodus, das dritte den Geschwindigkeitsmodus. Steuermodus und Schrittart werden unabhängig voneinander eingestellt, und es kann jede beliebige Kombination gewählt werden.

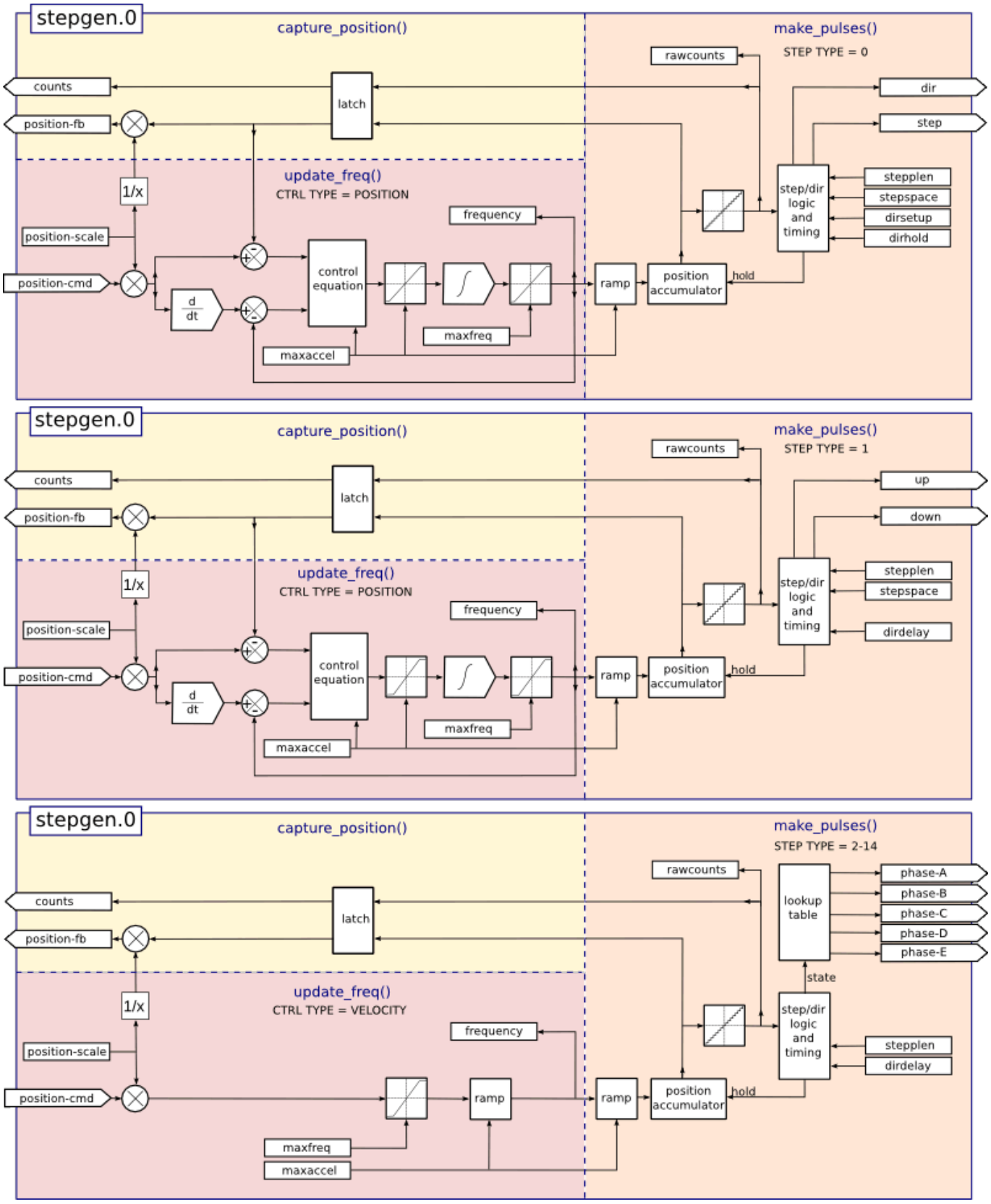


Abbildung 5.19: Schrittimpulsgenerator-Blockdiagramm Positionsmodus

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
```

Es werden drei Schrittgeneratoren installiert. Die ersten beiden verwenden den Schritttyp *0* (Schritt und Richtung) und laufen im Positionsmodus. Der letzte verwendet den Schritttyp *2* (Quadratur) und läuft im Geschwindigkeitsmodus. Der Standardwert für <*config-array*> ist *0,0,0*, wodurch drei Generatoren vom Typ *0* (Schritt/Richtung) installiert werden. Die maximale Anzahl von Schrittgeneratoren ist 8 (wie durch MAX_CHAN in stepgen.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.

Komponente ”stepgen” entfernen (engl. unload)

```
halcmd: unloadrt stepgen
```

5.8.1.1 Pins

Zur gewählten Schrift- und Steuerungsart.

- *(float) stepgen.*<chan>*.position-cmd* - Gewünschte Motorposition, in Positionseinheiten (nur Positionsmodus).
- *(float) stepgen.*<chan>*.velocity-cmd* - Gewünschte Motorgeschwindigkeit, in Positionseinheiten pro Sekunde (nur im Geschwindigkeitsmodus).
- *(s32) stepgen.*<chan>*.counts* - Feedback position in counts, updated by *capture_position()*.
- *(float) stepgen.* <chan>*.position-fb* - Feedback-Position in Positionseinheiten, aktualisiert durch *capture_position()*.
- *(bit) stepgen.*<chan>*.enable* - Aktiviert Ausgabeschritte - wenn false, werden keine Schritte erzeugt.
- *(bit) stepgen.*<chan>*.step* - Schrittimpulsausgang (nur Schritttyp 0).
- *(bit) stepgen.*<chan>*.dir* - Richtungsausgabe (nur Schritttyp 0).
- *(bit) stepgen.*<chan>*.up* - UP Pseudo-PWM Ausgang (nur Schritttyp 1).
- *(bit) stepgen.*<chan>*.down* - DOWN Pseudo-PWM-Ausgang (nur Schritttyp 1).
- *(bit) stepgen.*<chan>*.phase-A* - Ausgang Phase A (nur Schritttypen 2-14).

- *(bit) stepgen.*<chan>*.phase-B* - Ausgang Phase B (nur Schritttypen 2-14).
- *(bit) stepgen.*<chan>*.phase-C* - Phase-C-Ausgang (nur Schritttypen 3-14).
- *(bit) stepgen.*<chan>*.phase-D* - Phase-D-Ausgang (nur Schritttypen 5-14).
- *(bit) stepgen.*<chan>*.phase-E* - Phase-E-Ausgang (nur Schritttypen 11-14).

5.8.1.2 Parameter

- *(float) stepgen.<chan>.position-scale* - Schritte pro Positionseinheit. Dieser Parameter wird sowohl für die Ausgabe als auch für die Rückmeldung verwendet.
- *(float) stepgen.<chan>.maxvel* - Maximale Geschwindigkeit, in Positionseinheiten pro Sekunde. Wenn 0.0, hat keine Wirkung.
- *(float) stepgen.<chan>.maxaccel* - Maximale Beschleunigungs-/Verzögerungsrate, in Positionseinheiten pro Sekunde zum Quadrat. Wenn 0.0, hat keine Auswirkung.
- *(float) stepgen.<chan>.frequency* - Die aktuelle Schrittfrequenz in Schritten pro Sekunde.
- *(float) stepgen.<chan>.steplen* - Länge eines Schrittimpulses (Schritttyp 0 und 1) oder Mindestzeit in einem bestimmten Zustand (Schritttypen 2-14), in Nanosekunden.
- *(float) stepgen.<chan>.stepspace* - Mindestabstand zwischen zwei Schrittimpulsen (nur Schritttypen 0 und 1), in Nanosekunden. Wird auf 0 gesetzt, um die stepgen-Funktion *doublefreq* zu aktivieren. Um *doublefreq* zu verwenden, muss die parport reset function aktiviert sein.
- *(float) stepgen.<chan>.dirsetup* - Mindestzeit zwischen einem Richtungswechsel und dem Beginn des nächsten Schrittimpulses (nur Schritttyp 0), in Nanosekunden.
- *(float) stepgen.<chan>.dirhold* - Mindestzeit vom Ende eines Schrittimpulses bis zu einem Richtungswechsel (nur Schritttyp 0), in Nanosekunden.
- *(float) stepgen.<chan>.dirdelay* - Mindestzeit zwischen einem Schritt und einem Schritt in die entgegengesetzte Richtung (nur Schritttypen 1-14), in Nanosekunden.
- *(s32) stepgen.<chan>.rawcounts* - Die rohe Anzahl der Rückmeldungen, aktualisiert durch *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.

5.8.1.3 Schritttypen

Der Schrittgenerator unterstützt 15 verschiedene *Schrittfolgen*:

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 1ns, aber die automatische Rundung wird bei der ersten Ausführung des Codes wirksam. Da für einen Schritt *steplen* ns high und *stepspace* ns low benötigt werden, ist die maximale Frequenz 1.000.000.000 geteilt durch *(steplen+stepspace)*. Wird *maxfreq* höher als dieser Grenzwert eingestellt, 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.

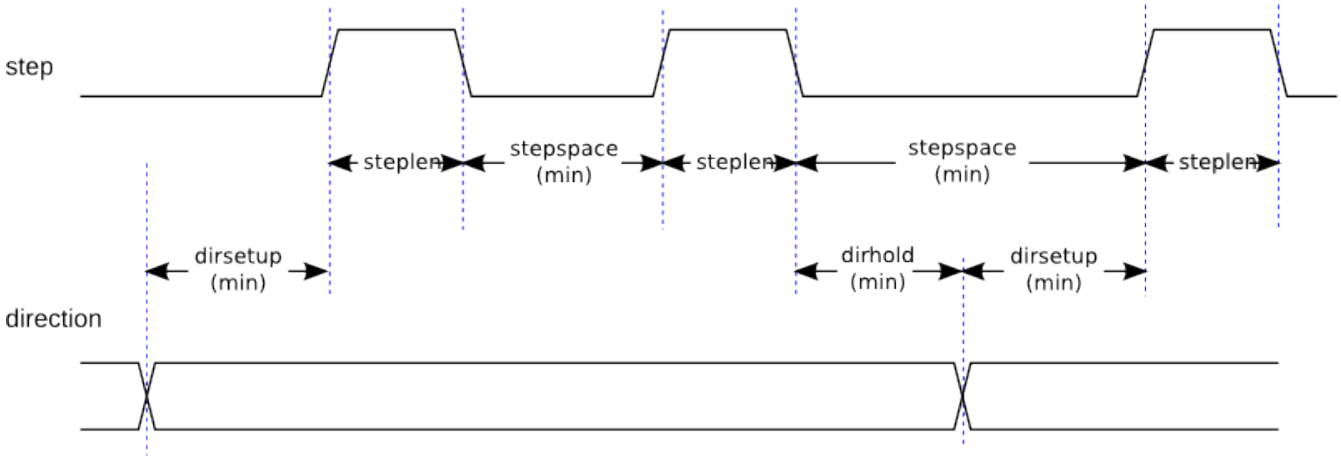


Abbildung 5.20: Schritt- und Richtungs-Timing (engl. step and direction timing)

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 Ausgang. Jeder Impuls ist *steplen* ns lang, und die Impulse sind durch mindestens *stepspace* ns getrennt. Die maximale Frequenz ist die gleiche 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.

Warnung
Verwenden Sie die Parport-Reset-Funktion nicht mit den Schritttypen 2 - 14. Unerwartete Ergebnisse können auftreten.

Schritt Typen 2 - 14 (engl. step type 2-14) Die Schritttypen 2 bis 14 sind zustandsabhängig und haben zwei bis fünf Ausgänge. Bei jedem Schritt wird ein Zustandszähler inkrementiert oder dekrementiert. Die Zwei-und-Drei-Phasen-, Vier-Phasen- und Fünf-Phasen-Schritte zeigen die Ausgangsmuster als Funktion des Zustandszählers. Die maximale Frequenz ist 1.000.000.000 geteilt durch *steplen*, und wie in den anderen Modi wird *maxfreq* gesenkt, wenn es über dem Grenzwert liegt.

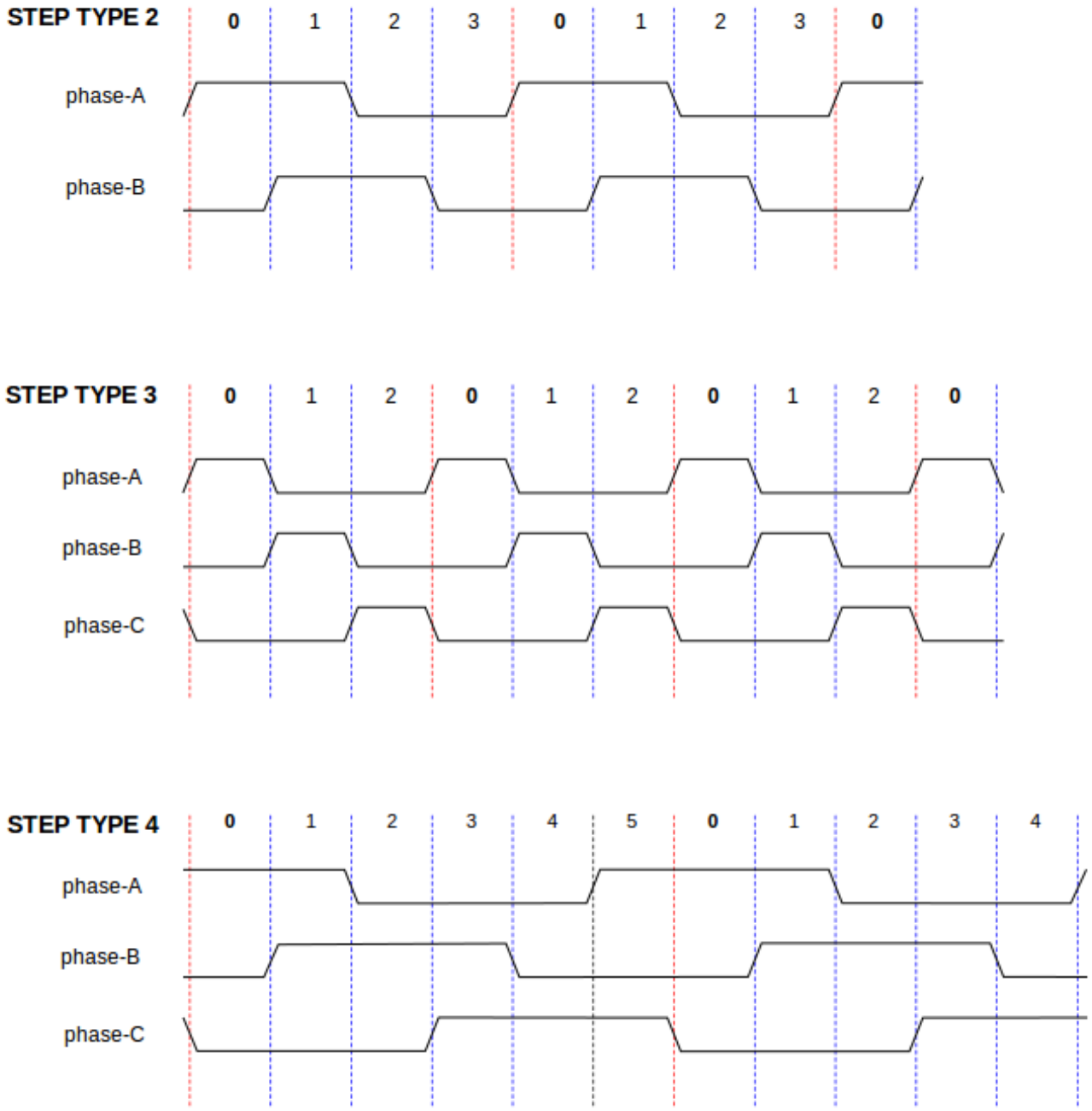


Abbildung 5.21: Zwei- und dreiphasige Schritttypen

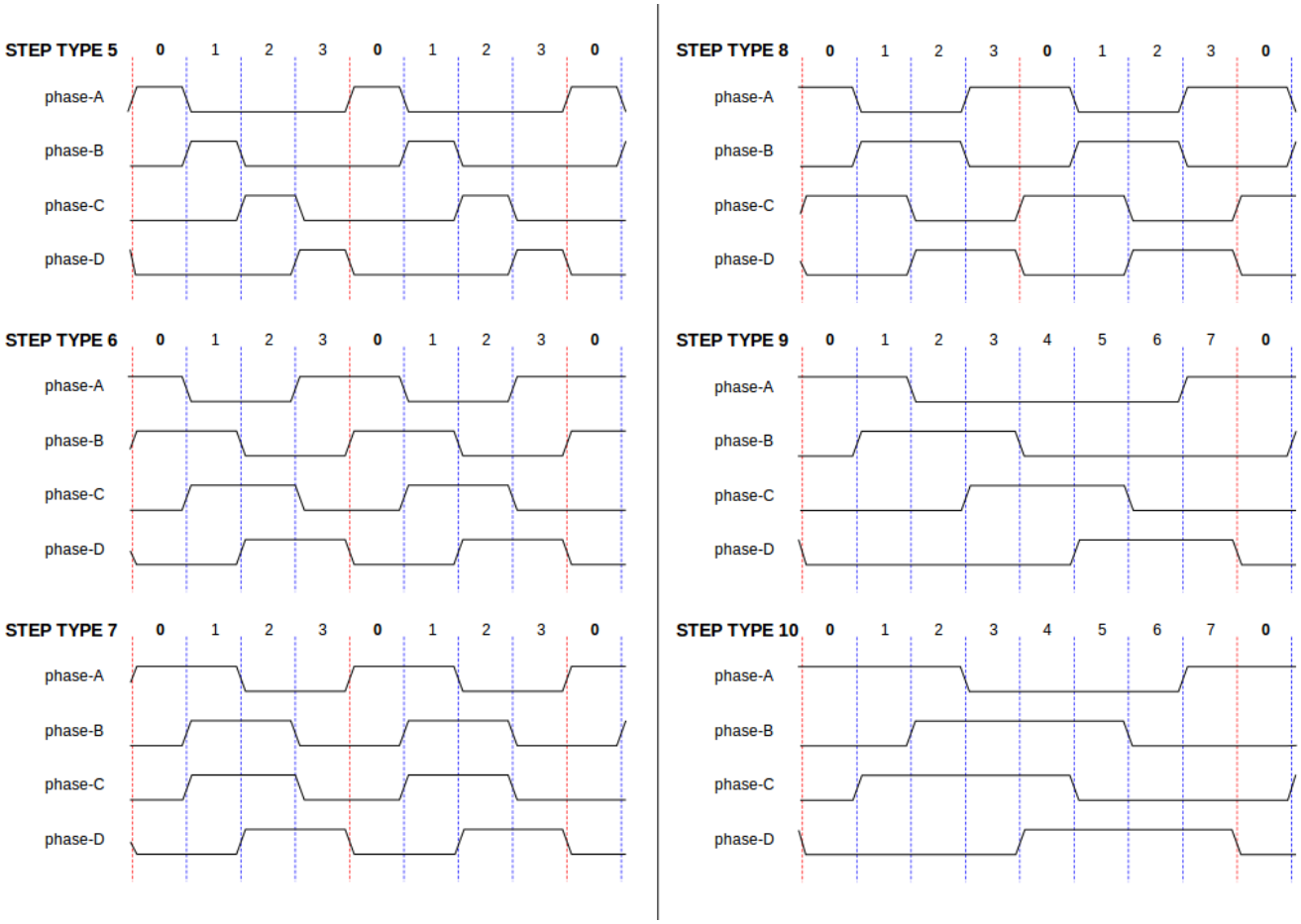


Abbildung 5.22: Vierphasige Schritttypen

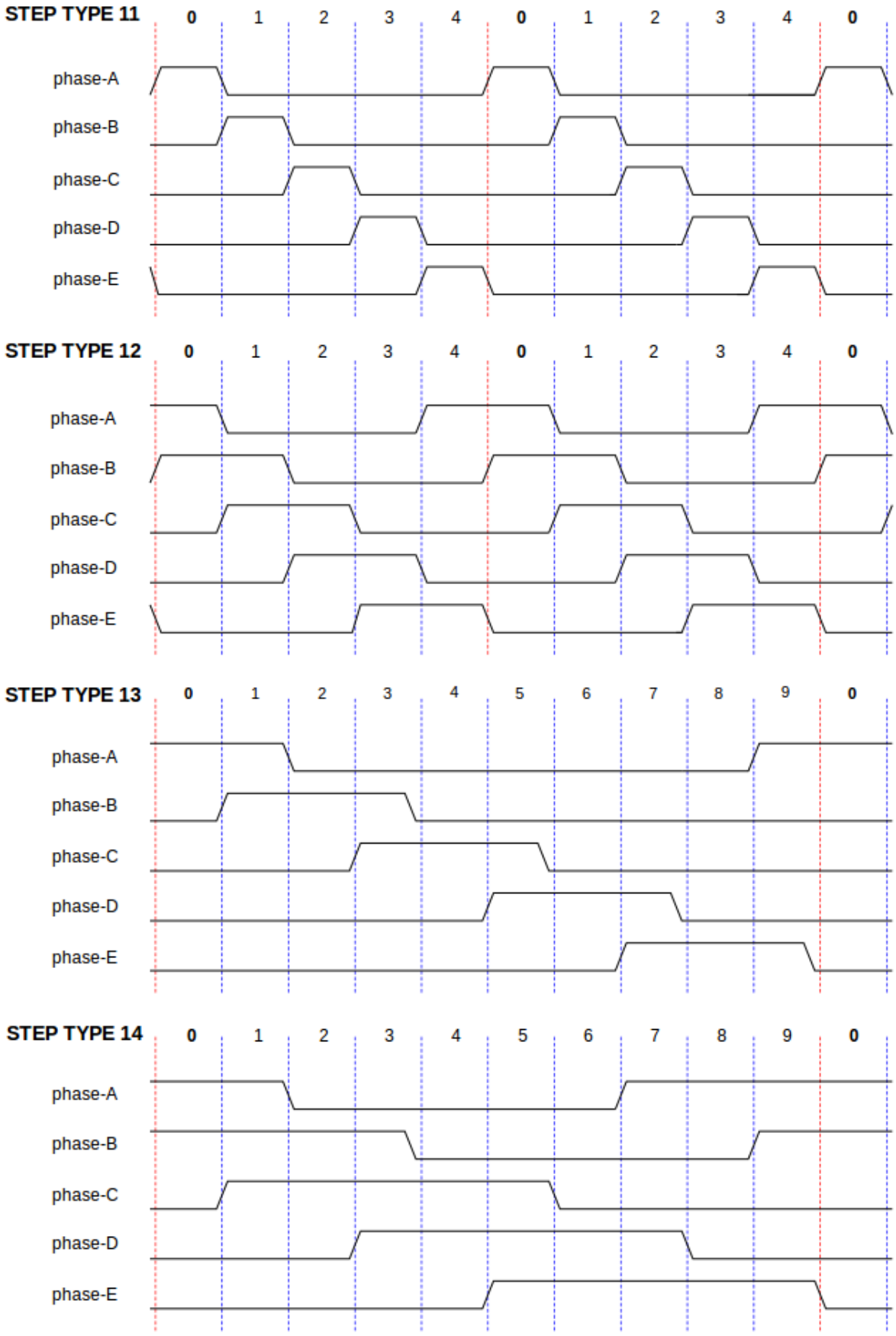


Abbildung 5.23: Fünf-Phasen-Schritttypen

5.8.1.4 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* - Hochgeschwindigkeitsfunktion zum Erzeugen und Zählen von Impulsen (kein Fließkomma).
- *(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.

5.8.2 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
```

5.8.2.1 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 Inputs werden als positive und negative PWM ausgegeben. Der Richtungspin ist false für positive Befehle und true für negative Befehle. Wenn Ihre Steuerung positive PWM sowohl für CW als auch für CCW benötigt, verwenden Sie den 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.

5.8.2.2 Pins

Jeder PWM-Generator hat die folgenden Pins:

- *(float) pwmgen.<chan>.value* - Befehlswert, in beliebigen Einheiten. Wird durch den Parameter *scale* skaliert (siehe unten).
- *(bit) pwmgen.<chan>.enable* - Aktiviert oder deaktiviert die PWM-Generatorausgänge.

Jeder PWM-Generator verfügt über einige dieser Pins, je nach gewähltem Ausgangstyp:

- *(bit) pwmgen.<chan>.pwm* - PWM- (oder PDM-) Ausgang, (nur Ausgangstyp 0 und 1).
- *(bit) pwmgen.<chan>.dir* - Richtungsausgabe (nur Ausgabetyp 1).
- *(bit) pwmgen.<chan>.up* - PWM/PDM-Ausgang für positiven Eingangswert (nur Ausgangstyp 2).
- *(bit) pwmgen.<chan>.down* - PWM/PDM-Ausgang für negativen Eingangswert (nur Ausgangstyp 2).

5.8.2.3 Parameter

- *(float) pwmgen.<chan>.scale* - Skalierungsfaktor zur Konvertierung von *value* von beliebigen Einheiten in Duty Cycle. Wenn z.B. scale auf 4000 gesetzt ist und der Eingangswert, der an pwmgen.<chan>.value übergeben wird, 4000 ist, dann wird es 100% Duty-Cycle (immer an) sein. Beträgt der Wert 2000, so handelt es sich um eine 50%ige 25Hz-Rechteckwelle.
- *(float) pwmgen.<chan>.pwm-freq* - Gewünschte PWM-Frequenz, in Hz. Wenn 0.0, wird PDM statt PWM erzeugt. Ist die Frequenz höher als die internen Grenzwerte, wird sie beim nächsten Aufruf von *update_freq()* auf den internen Grenzwert gesetzt. Falls ungleich Null und *dither* falsch, wird der nächste Aufruf von *update_freq()* auf das nächste ganzzahlige Vielfache der Periode der Funktion *make_pulses()* gesetzt.
- *(bit) pwmgen.<chan>.dither-pwm* - Bei true wird Dithering aktiviert, um durchschnittliche PWM-Frequenzen oder Tastverhältnisse zu erreichen, die mit reiner PWM nicht möglich sind. Bei false werden sowohl die PWM-Frequenz als auch das Tastverhältnis auf Werte gerundet, die genau erreicht werden können.

- *(float) pwmgen.<chan>.min-dc* - Minimales Tastverhältnis, zwischen 0,0 und 1,0 (das Tastverhältnis geht unabhängig von dieser Einstellung auf Null, wenn es deaktiviert wird).
- *(float) pwmgen.<chan>.max-dc* - Maximales Tastverhältnis, zwischen 0,0 und 1,0.
- *(float) pwmgen.<chan>.curr-dc* - Aktuelles Tastverhältnis - nach allen Begrenzungen und Rundungen (nur Lesen).

5.8.2.4 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* - Hochgeschwindigkeitsfunktion zur Erzeugung von PWM-Wellenformen (keine Fließkommazahlen). Die Hochgeschwindigkeitsfunktion *pwmgen.make-pulses* sollte im Basis-Thread (schnellster Thread) ausgeführt werden, je nach den Fähigkeiten des Computers zwischen 10 und 50 µs. Die Periode dieses Threads bestimmt die maximale PWM-Trägerfrequenz sowie die Auflösung der PWM- oder PDM-Signale. Wenn der Basis-Thread 50.000 ns beträgt, entscheidet das Modul alle 50 µs, ob es an der Zeit ist, den Zustand des Ausgangs zu ändern. Bei einem Tastverhältnis von 50 % und einer PWM-Frequenz von 25 Hz bedeutet dies, dass sich der Zustand des Ausgangs alle (1 / 25) Sekunden / 50 µs * 50 % = 400 Iterationen ändert. Das bedeutet auch, dass Sie 800 mögliche Tastverhältniswerte haben (ohne 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.

5.8.3 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.

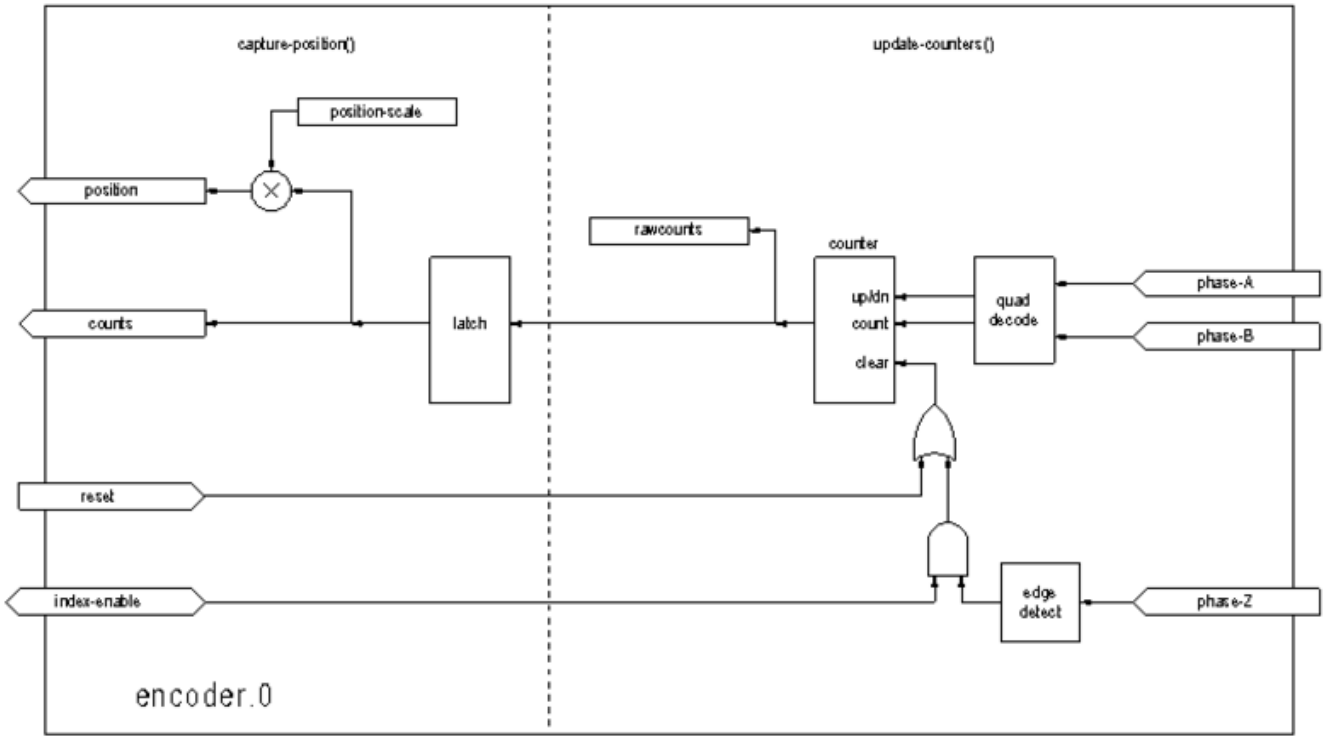


Abbildung 5.24: Encoderzähler-Blockdiagramm

Laden des Encoders

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<counters> ist die Anzahl der Encoderzähler, die Sie installieren möchten. Wenn *numchan* 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
```

5.8.3.1 Pins

- *encoder.<chan>.counter-mode* (Bit, I/O) (Voreinstellung: FALSE) - Aktiviert den Zählermodus. Bei true zählt der Zähler jede steigende Flanke des Phase-A-Eingangs und ignoriert den Wert an Phase-B. Dies ist nützlich, um den Ausgang eines einkanaligen (nicht-Quadratur-) Sensors zu zählen. Bei false zählt er im Quadraturmodus.

- *encoder.<chan>.missing-teeth* (s32, In) (Voreinstellung: 0) - Aktiviert die Verwendung des Fehlzahnindex. Dadurch kann ein einzelner IO-Pin sowohl Positions- als auch Indexinformationen liefern. Wenn das Geberrad 58 Zähne hat, von denen zwei fehlen, die so angeordnet sind, als wären es 60 (wie bei Kurbelwellensensoren in der Automobilindustrie üblich), dann sollte die Positionsskala auf 60 und die fehlenden Zähne auf 2 gesetzt werden. Um diesen Modus zu verwenden, sollte counter-mode auf true gesetzt werden. Dieser Modus eignet sich zum Gewindedrehen, aber nicht zum Gewindeschneiden.
- *encoder.<chan>.counts* (s32, Out) - Position in encoder counts.
- *encoder.<chan>.counts-latched* (s32, Out) - Zur Zeit nicht verwendet.
- *encoder.<chan>.index-enable* (bit, I/O) - Wenn True, werden *counts* und *position* bei der nächsten steigenden Flanke von Phase Z auf Null zurückgesetzt.
  Gleichzeitig wird *index-enable* auf Null zurückgesetzt, um anzuzeigen, dass die steigende Flanke aufgetreten ist. Der *Index-Enable*-Pin ist bidirektional. Wenn „index-enable“ False ist, wird der Phase-Z-Kanal des Encoders ignoriert und der Zähler zählt normal. Der Encoder-Treiber wird „index-enable“ niemals auf True setzen. Einige andere Komponenten können dies jedoch tun.
- *encoder.<chan>.latch-falling* (bit, In) (Standard: TRUE) - Derzeit nicht verwendet.
- *encoder.<chan>.latch-input* (bit, In) (Voreinstellung: TRUE) - Zur Zeit nicht verwendet.
- *encoder.<chan>.latch-rising* (bit, In) - Derzeit nicht verwendet.
- *encoder.<chan>.min-speed-estimate* (float, in) - Bestimmt die minimale wahre Geschwindigkeitsgröße, bei der die Geschwindigkeit als ungleich Null geschätzt und die Position interpoliert wird. Die Einheiten von *min-speed-estimate* sind die gleichen wie die Einheiten von *velocity* . Skalierungsfaktor, in Zählungen pro Längeneinheit. Wird dieser Parameter zu niedrig eingestellt, dauert es sehr lange, bis die Geschwindigkeit auf 0 zurückgeht, nachdem keine Geberimpulse mehr ankommen.
- *encoder.<chan>.phase-A* (bit, In) - Phase A of the quadrature encoder signal.
- *encoder.<chan>.phase-B* (bit, In) - Phase B of the quadrature encoder signal.
- encoder.<chan>.phase-Z’ (Bit, In) - Phase Z (Indeximpuls) des Quadratur-Encodersignals.
- *encoder.<chan>.position* (float, Out) - Position in skalierten Einheiten (siehe *position-scale*).
- *encoder.<chan>.position-interpolated* (float, Out) - Position in skalierten Einheiten, interpoliert zwischen Encoder-Zählungen.
  Die *position-interpolated* versucht, zwischen den Encoderzählungen zu interpolieren, basierend auf der zuletzt gemessenen Geschwindigkeit. Nur gültig, wenn die Geschwindigkeit annähernd konstant ist und über der *min-speed-estimate* liegt. Nicht für die Lageregelung verwenden, da der Wert bei niedrigen Geschwindigkeiten, bei Richtungsumkehr und bei Geschwindigkeitsänderungen falsch ist.
  Er ermöglicht jedoch die Verwendung eines Encoders mit niedrigem Impuls pro Umdrehung (einschließlich eines *Encoders* mit einem Impuls pro Umdrehung) für das Gewindeschneiden auf einer Drehmaschine und kann auch für andere Zwecke verwendet werden.
- *encoder. <chan>.position-latched (float, out)* - Wird derzeit nicht verwendet.
- *encoder.<chan>.position-scale (float, I/O)* – Skalierungsfaktor, in Zählungen pro Längeneinheit. Wenn beispielsweise die Positionsskala 500 beträgt, werden 1000 Zählwerte des Encoders als Position von 2.0 Einheiten gemeldet.
- *encoder.<chan>.rawcounts (s32, In)* - Die rohe Anzahl, wie durch Update-Zähler bestimmt. Dieser Wert wird häufiger aktualisiert als Anzahl und Position. Es ist auch unbeeinflusst von Reset oder dem Indeximpuls.
- *encoder.<chan>.reset* (bit, In) - Wenn True, werden *counts* und *position* sofort auf Null gesetzt.

- *encoder.<chan>.velocity* (float, Out) - Geschwindigkeit in skalierten Einheiten pro Sekunde. *encoder* verwendet einen Algorithmus, der das Quantisierungsrauschen im Vergleich zur einfachen Differenzierung des *position*-Ausgangs stark reduziert. Wenn der Wert der tatsächlichen Geschwindigkeit unter der geschätzten Mindestgeschwindigkeit liegt, ist die Geschwindigkeitsausgabe 0.
- encoder.<*chan*>.x4-mode (bit, I/O) (Voreinstellung: TRUE)' - Aktiviert den Times-4-Modus. Bei true zählt der Zähler jede Flanke der Quadraturwellenform (vier Zählungen pro vollem Zyklus). Bei false zählt er nur einmal pro vollem Zyklus. Im Zählermodus wird dieser Parameter ignoriert. Der 1x-Modus ist für einige Jogwheels nützlich.

5.8.3.2 Parameter

- *encoder.<chan>.capture-position.time (s32, RO)*
- *encoder.<chan>.capture-position.tmax (s32, RW)*
- *encoder.<chan>.update-counters.time (s32, RO)*
- *encoder.<chan>.update-counter.tmax (s32, RW)*

5.8.3.3 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* - Hochgeschwindigkeitsfunktion zum Zählen von Impulsen (kein Gleitkomma).
- *(funct) encoder.capture-position* - Funktion mit niedriger Geschwindigkeit zur Aktualisierung von Latches und Skalenposition.

5.8.4 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.

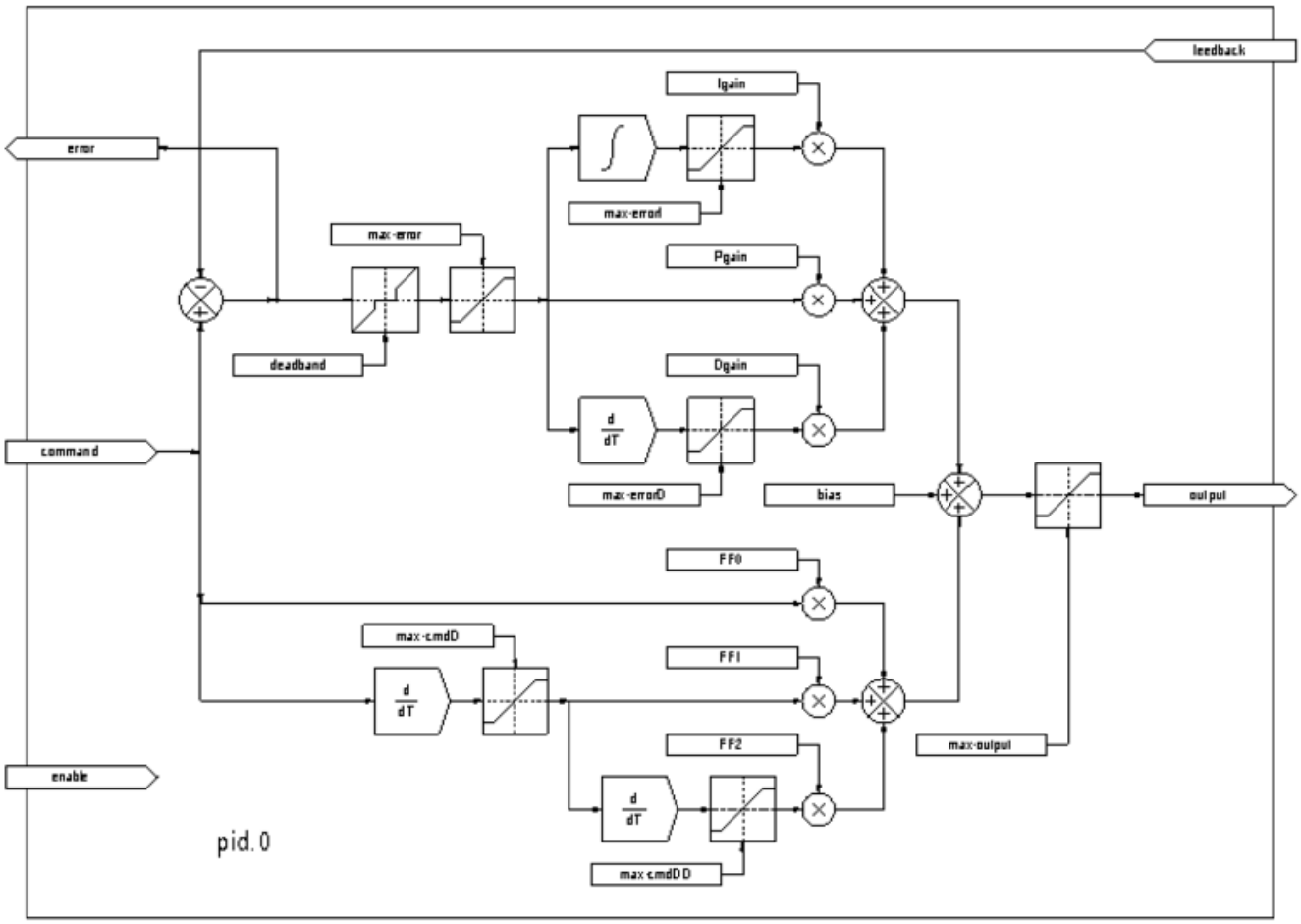


Abbildung 5.25: PID-Regelkreis-Blockdiagramm

PID laden

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

<Schleifen>’ ist die Anzahl der PID-Schleifen, die Sie installieren möchten. Wird *numchan* 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
```

5.8.4.1 Pins

Die drei wichtigsten Pins sind

- *(float) pid.<Loopnum>.command* - Die gewünschte Position, wie sie von einer anderen Systemkomponente befohlen wurde.
- *(Float) pid.<Schleifennummer>.Rückmeldung* - Die aktuelle Position, wie sie von einem Rückmeldegerät wie einem Encoder gemessen wird.
- *(float) pid.<loopnum>.output* - Ein Geschwindigkeitsbefehl, der versucht, von der aktuellen Position zur gewünschten Position zu gelangen.

Bei einer Positionsschleife sind ”Befehl” und ”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 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.

- *(float) pid.*<Schleifennummer>*.error* - Entspricht *.command* (gefordert) minus ’.feedback (Rückmeldung zu ist-Zustand).
- *(bit) pid.*<loopnum>*.enable* - Ein Bit, das die Schleife aktiviert. Wenn *.enable* falsch ist, werden alle Integratoren zurückgesetzt und der Ausgang wird auf Null gezwungen. Wenn *.enable* wahr ist, arbeitet die Schleife normal.

Pins zur Meldung der Sättigung. Eine Sättigung ist gegeben, wenn der Ausgang des PID-Blocks an seinem maximalen oder minimalen Grenzwert liegt.

- *(Bit) pid.* <loopnum>*.gesättigt* - True, wenn die Ausgabe gesättigt ist.
- *(float) pid.*<loopnum>*.saturated_s* – Die Zeit, zu der die Ausgabe zuerst gesättigt war.
- *(s32) pid.* <loopnum>*.saturated_count* - Die Dauer, seit der die Ausgabe gesättigt ist.

Die PID-Verstärkungen, Grenzwerte und andere ”abstimmbare” Merkmale des Regelkreises sind als Pins verfügbar, so dass sie dynamisch für erweiterte Abstimmungsmöglichkeiten angepasst werden können.

- *(float) pid.*<loopnum>*.Pgain* - Proportionale Verstärkung
- *(float) pid.*<loopnum>*.Igain* - Integrale Verstärkung
- *(float) pid.*<loopnum>*.Dgain* - Abgeleitete (engl. derivative) Verstärkung
- *(float) pid.*<loopnum>*.bias* - Konstanter Offset (engl. bias) am Ausgang
- *(float) pid.* <loopnum>*. FF0* - Feedforward nullter Ordnung - Ausgabe proportional zum Befehl (Position).
- *(float) pid.* <loopnum>*. FF1* - Feedforward erster Ordnung - Ausgabe proportional zur Ableitung des Befehls (Geschwindigkeit).
- *(float) pid.* <loopnum>*. FF2* - Feedforward zweiter Ordnung - Ausgabe proportional zur 2. Ableitung des Befehls (Beschleunigung).
- *(float) pid.*<loopnum>*.deadband* - Betrag des Fehlers, der ignoriert wird
- *(float) pid.* <loopnum>*.maxerror* - Fehlerbegrenzung

- *(float) pid.* <loopnum>*.maxerrorI* - Limit für Fehlerintegrator
- *(float) pid.* <loopnum>*.maxerrorD* - Limit für Fehlerableitung
- *(float) pid.*<loopnum>*.maxcmdD* - Begrenzung der Befehlsableitung
- *(float) pid.*<loopnum>*.maxcmdDD* - Begrenzung der 2. Ableitung des Befehls
- *(float) pid.* <loopnum>*.maxoutput* - Grenzwert für Ausgangswert

Alle *max**-Grenzwerte sind so implementiert, dass es keinen Grenzwert gibt, wenn der Wert dieses Parameters Null ist.

Wenn bei der Installation der Komponente *debug=1* angegeben wurde, werden vier zusätzliche Pins exportiert:

- *(float) pid.*<loopnum>*.errorI* - Integral des Fehlers.
- *(float) pid.*<loopnum>*.errorD* - Ableitung von error.
- *(float) pid.* <loopnum>*.commandD* - Ableitung des Befehls.
- *(float) pid.* <loopnum>*.commandDD* - 2. Ableitung des Befehls.

5.8.4.2 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* - Führt alle Berechnungen für eine einzelne PID-Schleife durch.

Wenn Sie den genauen Algorithmus zur Berechnung des Ausgangs der PID-Schleife verstehen wollen, lesen Sie die Abbildung PID-Regelkreis Block Diagram, die Kommentare am Anfang von *emc2/src/hal/compo* und natürlich den Code selbst. Die Schleifenberechnungen erfolgen in der C-Funktion *calc_pid()*.

5.8.5 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
```

5.8.5.1 Pins

- *(float) sim-encoder.*<chan-num>*.speed* - Der Geschwindigkeitsbefehl für die simulierte Welle.
- *(bit) sim-encoder.*<chan-num>*.phase-A* - Quadraturausgang.
- *(bit) sim-encoder.*<chan-num>*.phase-B* - Quadraturausgang.
- *(bit) sim-encoder.* <chan-num>*.phase-Z* - Index-Impulsausgang.

Wenn *.speed* positiv ist, liegt *.phase-A* vor *.phase-B*.

5.8.5.2 Parameter

- *(u32) sim-encoder.*<chan-num>*.ppr* - Impulse pro Umdrehung.
- *(float) sim-encoder.*<chan-num>*.scale* - Skalierungsfaktor für *speed*. Der Standardwert ist 1.0, was bedeutet, dass die Geschwindigkeit in Umdrehungen pro Sekunde angegeben wird. Ändern Sie den Wert auf 60 für Umdrehungen pro Minute, auf 360 für Grad pro Sekunde, 6,283185 für Bogenmaß pro Sekunde usw.

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.

5.8.5.3 Funktionen

Die Komponente exportiert zwei Funktionen. Jede Funktion wirkt auf alle simulierten Geber.

- *(funct) sim-encoder.make-pulses* - Hochgeschwindigkeitsfunktion zur Erzeugung von Quadraturimpulsen (kein Fließkomma).
- *(funct) sim-encoder.update-speed* - Funktion für niedrige Geschwindigkeit zum Lesen von *speed*, Skalieren und Einrichten von *make-pulses*.

5.8.6 Entprellung (engl. 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
```

werden drei Gruppen von Filtern installiert. Gruppe 0 enthält einen Filter, Gruppe 1 enthält vier Filter und Gruppe 2 enthält zwei Filter. Der Standardwert für <*config-string*> ist *”1”*, wodurch eine einzige Gruppe mit einem einzigen Filter installiert wird. Die maximale Anzahl von Gruppen ist 8 (wie durch MAX_GROUPS in debounce.c definiert). Die maximale Anzahl von Filtern in einer Gruppe ist nur durch den gemeinsamen Speicherplatz begrenzt. Jede Gruppe ist völlig unabhängig. Alle Filter in einer Gruppe sind identisch und werden alle von derselben Funktion gleichzeitig aktualisiert. In den folgenden Beschreibungen steht ”<G>” für die Gruppennummer und ”<F>” für die Filternummer innerhalb der Gruppe. Der erste Filter ist Gruppe 0, Filter 0.

Entladen der Entprellung

```
halcmd: unloadrt debounce
```

5.8.6.1 Pins

Jeder einzelne Filter hat zwei Pins.

- *(bit) debounce.*<G>.<F>*.in* - Eingang von Filter <*F*> in Gruppe <*G*>.
- *(bit) debounce.*<G>.<F>*.out* - Ausgang von Filter <*F*> in Gruppe <*G*>.

5.8.6.2 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.].

- *(s32) debounce.*<G>*.delay* - Filterverzögerung für alle Filter in der Gruppe <*G*>.

Die Filterverzögerung 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 zunehmender Verzögerung werden immer längere Störimpulse zurückgewiesen. Wenn *delay* 4 ist, werden alle Störungen zurückgewiesen, die kleiner oder gleich vier Thread-Perioden sind.

5.8.6.3 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*>.

5.8.7 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
```

5.8.7.1 Pins

Jeder Generator hat fünf Ausgangspins.

- *(float) siggen.*<chan>*.sine* - Ausgabe einer Sinuswelle.
- *(float) siggen.*<chan>*.cosine* - Ausgabe eines Kosinus.
- *(float) siggen.* <chan>*.sawtooth* - Sägezahn-Ausgang.
- *(float) siggen.*<chan>*.triangle* - Ausgabe einer Dreieckswelle.
- *(float) siggen.*<chan>*.square* - Ausgabe von Rechteckwellen.

Alle fünf Ausgänge haben die gleiche Frequenz, Amplitude und Offset.

Zusätzlich zu den Ausgangspins gibt es drei Steuerpins:

- *(float) siggen.*<chan>*.frequency* - Legt die Frequenz in Hertz fest, Standardwert ist 1 Hz.
- *(float) siggen.*<chan>*.amplitude* - Legt die Spitzenamplitude der Ausgangswellenformen fest, Standardwert ist 1.
- *(float) siggen.* <chan>*.offset* - Setzt den DC-Offset der Ausgangswellenformen, der Standardwert ist 0.

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.

5.8.7.2 Parameter

Keine. Fußnote:[Vor Version 2.1 waren Frequenz, Amplitude und Offset Parameter. Sie wurden in Pins geändert, um die Steuerung durch andere Komponenten zu ermöglichen.]

5.8.7.3 Funktionen

- *(funct) siggen.*<chan>*.update* - Berechnet neue Werte für alle fünf Ausgaben.

5.8.8 lut5

Die Komponente lut5 ist eine Logikkomponente mit 5 Eingängen, die auf einer Look-up-Tabelle basiert.

- lut5’ benötigt keinen Fließkomma-Thread.

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.

Tabelle 5.5: 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	
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.

Tabelle 5.6: 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*.

5.9 HAL-Komponentengenerator

5.9.1 Einführung

Das Schreiben einer HAL-Komponente kann ein langwieriger Prozess sein, der zum größten Teil aus Aufrufen der Funktionen *rtapi_* und *hal_* und der damit verbundenen Fehlerprüfung besteht. *halcompile* schreibt all diesen Code automatisch für Sie.

Das Kompilieren einer HAL-Komponente ist auch viel einfacher, wenn man *halcompile* benutzt, egal ob die Komponente Teil des LinuxCNC-Quellbaums ist oder nicht.

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 ”Berechne die Ableitung der Eingangsfunktion”;
pin in float in;
pin out float out;
variable double old;
function _;
license ”GPL”; // gibt GPL v2 oder höher an
;;
float tmp = in;
out = (tmp - old) / fperiod;
old = tmp;
```

5.9.2 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

```
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.

5.9.3 Verwendung einer Komponente

Komponenten müssen geladen und einem Thread hinzugefügt werden, bevor er verwendet werden kann.

Beispiel

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

More information on *loadrt* and *addf* can be found in the HAL Grundlagen.

Um Ihre Komponente zu testen, können Sie den Beispielen im HAL Tutorial folgen.

5.9.4 Definitionen

- *component* - Eine Komponente 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 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)

5.9.5 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.

5.9.6 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.

5.9.7 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

- *if CONDITION* (engl. für Bedingung)- Ein Ausdruck mit der Variablen *Persönlichkeit*, die ungleich Null ist, wenn der Pin oder Parameter erstellt werden soll
- *SIZE* - Eine Zahl, um die Größe eines Arrays anzugeben. Die Array-Elemente sind von 0 bis *SIZE*-1 nummeriert.
- *MAXSIZE : CONDSIZE* - Gibt die maximale Größe des Arrays an, gefolgt von einem Ausdruck, der die Variable *personality* einbezieht und der immer weniger als *MAXSIZE* ergibt. Wenn das Array erstellt wird, hat es die Größe *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.

  Die Dokumentationszeichenfolge hat das Format ”groff -man”. Für weitere Informationen über dieses Format siehe *groff_man(7)*. Denken Sie daran, dass *halcompile* Backslash-Escapes in Zeichenketten interpretiert, so dass Sie zum Beispiel die kursive Schriftart für das Wort *Beispiel* einstellen können:

```
”\\fIBeispiel\\fB”
```

  In diesem Fall sind r-Zeichenfolgen besonders nützlich, da die Backslashes in einer r-Zeichenfolge nicht verdoppelt werden müssen:

```
r”\fI-Beispiel\fB”
```

- *TYPE* - Einer der HAL-Typen: *bit*, *signed*, *unsigned* oder *float*. Die alten Namen ”s32” und ”u32” können ebenfalls verwendet werden, aber ”signiert” und ”unsigniert” werden bevorzugt.
- *PINDIRECTION* - Eine der folgenden Optionen: *in*, *out* oder *io*. Eine Komponente legt einen Wert für einen Out-Pin fest, liest einen Wert von einem ”In”-Pin und kann den Wert eines ”io”-Pins lesen oder festlegen.
- *PARAMDIRECTION* - Eine der folgenden: *r* oder *rw*. Eine Komponente legt einen Wert für einen *r*-Parameter fest und kann den Wert eines *rw*-Parameters lesen oder festlegen.
- *STARTVALUE* - Gibt den Anfangswert eines Pins oder Parameters an. Wenn nicht anders angegeben, ist der Standardwert ”0” oder ”FALSE”, abhängig vom Typ des Elements.
- *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.

5.9.7.1 HAL-Funktionen

- *fp* - Gibt an, dass die Funktion Gleitkommaberechnungen durchführt.
- *nofp* - Gibt an, dass nur Ganzzahlberechnungen durchgeführt werden. Wenn keines von beiden angegeben ist, wird *fp* angenommen. Weder *halcompile* noch gcc können die Verwendung von Fließkommaberechnungen in Funktionen, die mit *nofp* gekennzeichnet sind, erkennen, aber die Verwendung solcher Operationen führt zu undefiniertem Verhalten.

5.9.7.2 Optionen

Die derzeit definierten Optionen sind:

- *option singleton yes* - (Standardwert: 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>.Elementnam* 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* - (Voreinstellung: no) Normalerweise wird die Anzahl der zu erstellenden Instanzen im Modulparameter *count* angegeben; wenn *count_function* angegeben ist, wird stattdessen der von der Funktion *int get_count(void)* zurückgegebene Wert verwendet, und der Modulparameter *count* ist nicht definiert.
- *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* - (Voreinstellung: none) **veraltet** Wenn angegeben, hat jede Instanz der Komponente einen zugehörigen Datenblock des Typs *TYPE* (der ein einfacher Typ wie *float* oder der Name eines mit *typedef* erstellten Typs sein kann). In neuen Komponenten sollte stattdessen *variable* verwendet werden.
- option extra_setup yes' - (Standard: 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* - (Standard: nein) Falls angegeben, rufen Sie die durch *EXTRA_CLEANUP* definierte Funktion aus dem automatisch definierten *rtapi_app_exit* auf, oder wenn im automatisch definierten *rtapi_app_main* ein Fehler erkannt wird.
- *option userspace yes* - (Voreinstellung: no) Wenn diese Datei angegeben wird, beschreibt sie eine Userspace-Komponente (d.h. eine Nicht-Echtzeit-Komponente) und keine reguläre Komponente (d.h. eine Echtzeit-Komponente). Eine Userspace-Komponente darf 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* - (Standard: 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 "..."* - (Standard: "") Diese Option wird ignoriert, wenn die Option *userspace* (siehe oben) auf *no* gesetzt ist. Beim Linken einer Userspace-Komponente 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.
- *option extra_compile_args "..."* - (Standard: "") Diese Option wird ignoriert, wenn die Option *userspace* (siehe oben) auf *no* gesetzt ist. Beim Kompilieren einer Userspace-Komponente werden die angegebenen Argumente in die Befehlszeile des Compilers eingefügt.
- *option homemod yes* - (Standard: no) Modul ist ein benutzerdefiniertes Homing-Modul, das mit [EMCMOT]HOMEMOD=modulename geladen wird
- *option tpmod yes* - (Standard: no) Modul ist ein benutzerdefiniertes Trajektorienplanungsmodul (tp), das mit [TRAJ]TPMOD=modulename geladen wird

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.

5.9.7.3 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.

5.9.7.4 Datenspeicherung pro Instanz

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARRED = DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Deklarieren Sie eine Instanzvariable *STARREDNAME* vom Typ *CTYPE*, optional als Array von *SIZE*-Elementen und optional mit einem Standardwert *DEFAULT*.
  Elemente ohne *DEFAULT* werden auf alle Bits-Null initialisiert.
  CTYPE' ist ein einfacher Ein-Wort-C-Typ, wie *float*, *u32*, *s32*, int, etc.
  Der Zugriff auf Array-Variablen erfolgt über eckige Klammern.

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;
```

5.9.7.5 Kommentare

Einzeilige Kommentare im C++-Stil (`//...`) und mehrzeilige Kommentare im C-Stil (`/* ... */`) werden beide im Deklarationsabschnitt unterstützt.

5.9.8 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*

5.9.9 Bequemlichkeits-Makros

Basierend auf den Elementen im Deklarationsabschnitt erzeugt *halcompile* eine C-Struktur namens `struct __comp_state`. Anstatt jedoch auf die Mitglieder dieser Struktur zu verweisen (z.B. `*(inst->name)`), werden sie im Allgemeinen mit den untenstehenden Makros angesprochen. Die Details von `struct __comp_state` und diesen Makros können sich von einer Version von *halcompile* zur nächsten ändern.

- 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.
- 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.

- *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.
- *FOR_ALL_INSTS() {...}* - Für Userspace-Komponenten. Dieses Makro iteriert über alle definierten Instanzen. Innerhalb des Schleifenkörpers arbeiten die Makros *pin_name*, *parameter_name* und *data* wie in Echtzeitfunktionen.

5.9.10 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.

5.9.11 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 *--personality* mit halcompile. Zum Beispiel, um bis zu 128 Persönlichkeitszeiten zu erlauben:

```
[sudo] halcompile --personality=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
```

Anmerkung
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.

5.9.12 Kompilieren

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 Userspace-Simulator konfiguriert.

5.9.13 Kompilieren von Echtzeitkomponenten 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
```

Anmerkung
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.

Oder es kann in einem Schritt verarbeitet und kompiliert werden, wobei *example.ko* (oder *example.so* für den Simulator) im aktuellen Verzeichnis verbleibt:

```
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 rtexample.comp
```

Die resultierende Manpage „example.9“ kann angezeigt werden mit

```
man ./example.9
```

oder an einen Standardspeicherort für UNIX man pages kopiert.

5.9.14 Kompilieren von Userspace-Komponenten außerhalb des Quellbaums

halcompile’ kann Userspace-Komponenten verarbeiten, kompilieren, installieren und dokumentieren:

```
halcompile usrexample.comp
halcompile --compile usrexample.comp
[sudo] halcompile --install usrexample.comp
halcompile --document usrexample.comp
```

Dies funktioniert nur für *.comp*-Dateien, nicht für *.c*-Dateien.

5.9.15 Beispiele

5.9.15.1 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 float out;
param r float value = 1.0;
function _;
license ”GPL”; // bedeutet GPL v2 oder höher
;;
FUNCTION(_) { out = value; }
```

5.9.15.2 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 float sin_;
pin out float cos_;
pin in float theta;
function _;
license ”GPL”; // bedeutet GPL v2 oder höher
;;
#include <rtapi_math.h>
FUNCTION(_) { sin_ = sin(theta); cos_ = cos(theta); }
```

5.9.15.3 out8

Bei dieser Komponente handelt es sich um einen Treiber für eine ”fiktive” Karte mit der Bezeichnung ”out8”, die über 8 Pins mit digitalen Ausgängen verfügt, die als ein einziger 8-Bit-Wert behandelt werden. Es kann eine unterschiedliche Anzahl solcher Karten im System geben, und sie können sich an verschiedenen Adressen befinden. Der Pin wird *out_* genannt, weil *out* ein in *<asm/io.h>* verwendeter Bezeichner ist. Er veranschaulicht die Verwendung von *EXTRA_SETUP* und *EXTRA_CLEANUP*, um einen E/A-Bereich anzufordern und ihn dann im Fehlerfall oder beim Entladen des Moduls wieder freizugeben.

```
component out8;
pin out unsigned out_ ”Ausgabewert; es werden nur niedrige 8 Bit verwendet”;
param r unsigned ioaddr;

function _;

option count_function;
option extra_setup;
option extra_cleanup;
option constructable no;

license ”GPL”; // bedeutet GPL v2 oder höher
;;
```

```
#include <asm/io.h>

#define MAX 8
int io[MAX] = {0,};
RTAPI_MP_ARRAY_INT(io, MAX, "E/A-Adressen der out8-Karten");

int get_count(void) {
    int i = 0;
    for(i=0; i<MAX && io[i]; i++) { /* Nichts */ }
    return i;
}

EXTRA_SETUP() {
    if(!rtapi_request_region(io[extra_arg], 1, "out8")) {
        // Setze diesen I/O-Port auf 0, damit EXTRA_CLEANUP die IO-Ports nicht freigibt,
        // die nie angefordert wurden.
        io[extra_arg] = 0;
        return -EBUSY;
    }
    ioaddr = io[extra_arg];
    return 0;
}

EXTRA_CLEANUP() {
    int i;
    for(i=0; i < MAX && io[i]; i++) {
        rtapi_release_region(io[i], 1);
    }
}

FUNCTION(_) { outb(out_, ioaddr); }
```

5.9.15.4 hal_loop

```
component hal_loop;
pin out float example;
```

Dieses Komponentenfragment veranschaulicht die Verwendung des Präfixes "hal_" in einem Komponentennamen. "loop" ist der Name eines Standard-Linux-Kernelmoduls, so dass eine "loop"-Komponente möglicherweise nicht erfolgreich geladen werden kann, wenn das Linux-Modul "loop" ebenfalls auf dem System vorhanden ist.

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".

5.9.15.5 arraydemo

Diese Echtzeitkomponente veranschaulicht die Verwendung von Arrays fester Größe:

```
component arraydemo "4-Bit-Schieberegister";
pin in bit in;
pin out bit out-# [4];
funktion _ nofp;
licencse "GPL"; // bedeutet GPL v2 oder höher
;;
int i;
for(i=3; i>0; i--) out(i) = out(i-1);
out(0) = in;
```

5.9.15.6 rand

Diese Userspace-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 float out;
license "GPL"; // bedeutet GPL v2 oder höher
;;
#include <unistd.h>

void user_mainloop(void) {
    while(1) {
        usleep(1000);
        FOR_ALL_INSTS() out = drand48();
    }
}
```

5.9.15.7 logic

Diese Echtzeitkomponente zeigt, wie man "Persönlichkeit" verwendet, um Arrays variabler Größe und optionale Pins zu erstellen.

```
component logic "LinuxCNC HAL Komponente mit experimentellen Logikfunktionen";
pin in bit in-##[16 : personality & 0xff];
pin out bit and if personality & 0x100;
pin out bit or if personality & 0x200;
pin out bit xor if personality & 0x400;
function _ nofp;
description """
Experimentelle allgemeine 'Logikfunktion' Komponente. Kann 'und', 'oder' und 'xor' von bis ↩
    zu 16 Eingängen durchführen. Bestimmen Sie den richtigen Wert für 'Persönlichkeit' durch ↩
     Hinzufügen:
.IP \\(bu 4 Die Anzahl der Eingangsstifte, in der Regel von 2 bis 16
.IP \\(bu 256 (0x100), wenn der 'und'-Ausgang gewünscht ist
.IP \\(bu 512 (0x200), wenn der 'oder'-Ausgang erwünscht ist
IP \\(bu 1024 (0x400), wenn die 'xor'-Ausgabe (exklusives oder) gewünscht ist""";
license "GPL"; // bedeutet GPL v2 or höher
;;
FUNCTION(_) {
    int i, a=1, o=0, x=0;
    for(i=0; i < (personality & 0xff); i++) {
        if(in(i)) { o = 1; x = !x; }
        else { a = 0; }
    }
    if(personality & 0x100) and = a;
    if(personality & 0x200) or = o;
    if(personality & 0x400) xor = 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

5.9.15.8 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 s32 in;
pin out bit out1;
pin out bit out2;

function _;
license "GPL";
;;

// allgemeine Pin Set True Funktion
void set(hal_bit_t *p){
    *p = 1;
}

// allgemeine Pin Set False Funktion
void unset(hal_bit_t *p){
    *p = 0;
}

//Haupt-Funktion (engl. main)
FUNCTION(_) {
    if (in < 0){
        set(&out1);
        unset(&out2);
    }else if (in >0){
        unset(&out2);
        set(&out2);
    }else{
        unset(&out1);
        unset(&out2);
    }
}
```

Diese Komponente verwendet zwei allgemeine Funktionen, um einen HAL-Bit-Pin zu manipulieren, auf den sie referenziert ist.

5.9.16 Verwendung der Kommandozeile

Die Manpage halcompile enthält Details zum Aufruf von halcompile.

```
$ man halcompile
```

Eine kurze Zusammenfassung der Verwendung von halcompile finden Sie hier:

```
$ halcompile --help
```

5.10 HALTCL-Dateien

Die halcmd-Sprache eignet sich hervorragend für die Angabe von Komponenten und Verbindungen, bietet aber keine Berechnungsmöglichkeiten. Infolgedessen sind INI-Dateien in der Klarheit und Kürze eingeschränkt, die mit höheren Sprachen möglich ist.

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.

5.10.1 Kompatibilität

Die in HAL-Dateien verwendete halcmd-Sprache hat eine einfache Syntax, die eigentlich eine Teilmenge der leistungsfähigeren Allzweck-Skriptsprache Tcl ist.

5.10.2 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,
linkpp, 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
```

5.10.3 Haltcl INI-Datei-Variablen

Auf INI-Datei-Variablen kann sowohl mit halcmd als auch mit haltcl zugegriffen werden, allerdings mit unterschiedlicher Syntax.

LinuxCNC INI-Dateien verwenden SECTION und ITEM Spezifizierer, 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 oder andere Zeichen, die in Tcl eine besondere Bedeutung haben - 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]
```

5.10.4 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
```

5.10.5 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)}]
```

5.10.6 Haltcl Beispiele

Betrachten Sie das Thema "Stepgen Headroom". 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_MAX 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]
```

5.10.7 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
```

5.10.8 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.

5.11 Erstellen von Userspace-Python-Komponenten

5.11.1 Grundlegende Verwendung (engl. basic usage)

Eine Userspace-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, time
h = hal.component("passthrough")
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
h.newpin("out", hal.HAL_FLOAT, hal.HAL_OUT)
h.ready()
try:
    while 1:
        time.sleep(1)
        h['out'] = h['in']
except KeyboardInterrupt:
    raise SystemExit
```

Kopieren Sie das obige Listing in eine Datei mit dem Namen "passthrough", machen Sie es ausführbar (*chmod +x*), und legen Sie es in Ihren *$PATH*. Dann probieren Sie es aus:

```
halrun

halcmd: loadusr passthrough

halcmd: show pin

    Komponenten-Pins:
    Owner Typ  Richtung     Wert  Name
     03   float IN          0  passthrough.in
     03   float OUT         0  passthrough.out

halcmd: setp passthrough.in 3.14

halcmd: show pin

    Komponenten-Pins:
    Owner Typ   Richtung Wert  Name
     03   float IN       3.14  passthrough.in
     03   float OUT      3.14  passthrough.out
```

5.11.2 Userspace-Komponenten und Verzögerungen

Wenn Sie ”show pin” schnell eingegeben haben, 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 hier um eine Userspace-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; die Zuweisung könnte verzögert werden, bis dieser Speicher wieder ausgelagert wird.

So eignen sich Userspace-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).

5.11.3 Pins und Parameter erstellen

```
h = 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.

```
h.newpin(”in”, hal.HAL_FLOAT, hal.HAL_IN)
```

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.

Tabelle 5.7: Namen der HAL-Optionen

Pin- und Parametertypen:	HAL_BIT	HAL_FLOAT	HAL_S32	HAL_U32
Pin-Richtungen:	HAL_IN	HAL_OUT	HAL_IO	
Parameter-Richtungen:	HAL_RO	HAL_RW		

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*.

```
h.ready()
```

Sobald alle Pins und Parameter erstellt wurden, rufen Sie die Methode *.ready()* auf.

5.11.3.1 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.

5.11.4 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:

```
h.out = h.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:

```
h['out'] = h['in']
```

Um alle Pins mit ihren Werten zu sehen, gibt getpins alle Werte in einem Wörterbuch dieser Komponente zurück.

```
h.getpins()
>>>{'in': 0.0, 'out': 0.0}
```

5.11.4.1 Ansteuerung der Ausgangsstifte (HAL_OUT)

In regelmäßigen Abständen, in der Regel als Reaktion auf einen Timer, sollten alle HAL_OUT-Pins ”getrieben” werden, indem ihnen ein neuer Wert zugewiesen wird. Dies sollte unabhängig davon geschehen, ob sich der Wert von dem zuletzt zugewiesenen unterscheidet oder nicht. Beim Verbinden eines Pins mit einem Signal wird sein alter Ausgangswert nicht in das Signal kopiert, so dass der richtige Wert erst auf dem Signal erscheint, wenn die Komponente einen neuen Wert zuweist.

5.11.4.2 Ansteuerung von bidirektionalen (HAL_IO) Pins

Die obige Regel gilt nicht für bidirektionale Pins. Stattdessen sollte ein bidirektionaler Pin nur dann von der Komponente angesteuert werden, wenn die Komponente den Wert ändern möchte. In der kanonischen Encoder-Schnittstelle beispielsweise setzt die Encoder-Komponente den Pin *index-enable* nur auf **FALSE** (wenn ein Indeximpuls gesehen wird und der alte Wert **TRUE** ist), aber niemals auf **TRUE**. Das wiederholte Setzen des Pins auf **FALSE** könnte dazu führen, dass die andere angeschlossene Komponente sich so verhält, als ob ein weiterer Indeximpuls gesehen worden wäre.

5.11.5 Beenden

Eine *halcmd unload* Anforderung für die Komponente wird als *KeyboardInterrupt* Ausnahme geliefert. Wenn eine Entladeanforderung eintrifft, sollte der Prozess entweder in kurzer Zeit beendet werden oder die Methode *.exit()* für die Komponente aufrufen, wenn umfangreiche Arbeiten (wie das Lesen oder Schreiben von Dateien) durchgeführt werden müssen, um den Abschaltvorgang abzuschließen.

5.11.6 Hilfreiche Funktionen

5.11.6.1 ready (engl. für bereit)

Sagt dem HAL-System, dass die Komponente initialisiert ist. Sperrt das Hinzufügen von Pins.

5.11.6.2 unready (engl. für nicht bereit)

Ermöglicht es einer Komponente, Pins hinzuzufügen, nachdem *ready()* aufgerufen wurde. Man sollte *ready()* auf der Komponente danach aufrufen.

5.11.6.3 component_exists

Existiert die angegebene Komponente zu diesem Zeitpunkt.

Beispiel

```
hal.component_exists(”testpanel”)
```

5.11.6.4 component_is_ready (engl. für Komponente ist bereit)

Ist die angegebene Komponente zu diesem Zeitpunkt bereit.

Beispiel

```
hal.component_is_ready(”testpanel”)
```

5.11.6.5 get_msg_level

Abfrage der aktuellen Echtzeit-Nachrichten (engl. kurz msg)-Stufe.

5.11.6.6 set_msg_level

Legen Sie die aktuelle Echtzeit-MSG-Stufe fest. Wird zum Debuggen von Informationen verwendet.

5.11.6.7 verbinden

Verbinden Sie einen Pin mit einem Signal.

Beispiel

```
hal.connect(”pinname”,”signal_name”)
```

5.11.6.8 trennen

Trennen Sie einen Pin von einem Signal.

Beispiel

```
hal.disconnect(”Pinname”)
```

5.11.6.9 get_value

Lesen Sie einen Pin, einen Parameter oder ein Signal direkt.

Beispiel

```
value = hal.get_value("iocontrol.0.emc-enable-in")
```

5.11.6.10 get_info_pins()

Gibt eine Liste von Dicts aller Systempins zurück.

```
listOfDicts = hal.get_info_pins()
pinName1 = listOfDicts[0].get('NAME')
pinValue1 = listOfDicts[0].get('VALUE')
pinType1 = listOfDicts[0].get('TYPE')
pinDirection1 = listOfDicts[0].get('DIRECTION')
```

5.11.6.11 get_info_signals()

Gibt eine Liste von Dicts aller Systemsignale zurück.

```
listOfDicts = hal.get_info_signals()
signalName1 = listOfDicts[0].get('NAME')
signalValue1 = listOfDicts[0].get('VALUE')
driverPin1 = listOfDicts[0].get('DRIVER')
```

5.11.6.12 get_info_params()

Gibt eine Liste von Dicts mit allen Systemparametern zurück.

```
listOfDicts = hal.get_info_params()
paramName1 = listOfDicts[0].get('NAME')
paramValue1 = listOfDicts[0].get('VALUE')
paramDirection1 = listOfDicts[0].get('DIRECTION')
```

5.11.6.13 new_sig

Erstellt ein neues Signal des angegebenen Typs.

Beispiel

```
hal.new_sig("signalname",hal.HAL_BIT)
```

5.11.6.14 pin_has_writer

Ist der angegebene Pin mit einem treibenden Pin verbunden.
Gibt True oder False zurück.

```
h.in.pin_has_writer()
```

5.11.6.15 get_name

Ermittelt den Namen des HAL-Objekts.
Rückgabe einer Zeichenkette.

```
h.in.get_name()
```

5.11.6.16 get_type

Ermittelt den Typ des HAL-Objekts'.
Gibt eine ganze Zahl zurück.

```
h.in.get_type()
```

5.11.6.17 get_dir

Ermittelt den Richtungstyp des HAL-Objekts.
Gibt eine ganze Zahl zurück.

```
h.in.get_dir()
```

5.11.6.18 get

Ermittelt den Wert des HAL-Objekts.

```
h.in.get()
```

5.11.6.19 set

Setzt den Wert des HAL-Objekts.

```
h.out.set(10)
```

5.11.6.20 is_pin

Ist das Objekt ein Pin oder Parameter?
Gibt True oder False zurück.

```
h.in.is_pin()
```

5.11.6.21 sampler_base

TODO

5.11.6.22 stream_base

TODO

5.11.6.23 stream

TODO

5.11.6.24 set_p

Einstellen eines Pin-Wertes eines beliebigen Pins im HAL-System.

Beispiel

```
hal.set_p(”pinname”,”10”)
```

5.11.7 Konstanten

Verwenden Sie diese, um Details zu spezifizieren, und nicht den Wert, den sie enthalten.

- HAL_BIT
- HAL_FLOAT
- HAL_S32
- HAL_U32
- HAL_IN
- HAL_OUT
- HAL_RO
- HAL_RW
- MSG_NONE
- MSG_ALL
- MSG_DBG
- MSG_ERR
- MSG_INFO
- MSG_WARN

5.11.8 System-Informationen

Lesen Sie diese, um Informationen über das Echtzeitsystem zu erhalten.

- is_kernelspace
- is_rt
- is_sim
- is_userspace

5.11.9 Verwendung mit hal_glib in GladeVCP Handler

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 glib-Modul 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.HAL_BIT, ↩
            hal.HAL_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:" % (pin.get()))
        print("pin name= %s" % (pin.get_name()))
        print("pin type= %d" % (pin.get_type()))

        # dies kann außerhalb der Funktion aufgerufen werden
        self.example_trigger.get()
```

5.11.10 Verwendung mit hal_glib in QtVCP Handler

QtVCP nutzt die hal_glib Bibliothek, die verwendet werden kann, um 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 Modul importieren:

```
import hal
```

Erstellen Sie dann einen Pin und verbinden Sie ein *value_changed* (den Watcher) Signal mit einem Funktionsaufruf:

```
    ########################
    **** 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):
        self.hal = halcomp
        self.testPin = self.hal.newpin('test-pin', hal.HAL_BIT, hal.HAL_IN)
        self.testPin.value_changed.connect(lambda s: self.setTestPin(s))
```

Und eine aufzurufende Funktion haben. Dies zeigt, wie man den Pin-Wert und die Informationen erhält.

```
    ####################
    # general functions #
    ####################
    def setTestPin(self, data):
        print("Test pin value changed to:" % (data))
        print('halpin object =', self.w.sender())
        print('halpin name: ',self.sender().text())
        print('halpin type: ',self.sender().get_type())

        # dies kann außerhalb der Funktion aufgerufen werden
        print(self.testPin.get())
```

5.11.11 Projektideen

- Erstellen Sie ein externes Bedienfeld mit Tasten, Schaltern und Anzeigen. Schließen Sie alles an einen Mikrocontroller an, und verbinden Sie den Mikrocontroller über eine serielle Schnittstelle mit dem PC. Python hat ein sehr leistungsfähiges serielles Schnittstellenmodul namens pyserial (Ubuntu-Paketname "python3-serial", im Universum-Repository)
- Schließen Sie ein LCDProc-kompatibles LCD-Modul an und verwenden Sie es, um eine digitale Anzeige mit Informationen Ihrer Wahl anzuzeigen (Ubuntu-Paketname "lcdproc", im Universum-Repository)
- Erstellen eines virtuellen Bedienfelds mit einer beliebigen von Python unterstützten GUI-Bibliothek (Gtk, Qt, wxWindows usw.).

5.12 Kanonische Geräteschnittstellen (engl. canonical device interfaces)

5.12.1 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.

5.12.2 Digitaler Eingang

Der kanonische Digitaleingang (E/A-Typfeld: "digin") ist recht einfach.

5.12.2.1 Pins

(bit) in
Zustand des Hardware-Eingangs.

(bit) in-not
Invertierter Zustand des Eingangs.

5.12.2.2 Parameter

Keine

5.12.2.3 Funktionen

(funct) read
Lesen der Hardware und Setzen der HAL-Pins ”in” und ”in-not”.

5.12.3 Digitaler Ausgang

Der kanonische digitale Ausgang (engl. output) (E/A-Typfeld: `digout`) ist ebenfalls sehr einfach.

5.12.3.1 Pins

(bit) out
Wert, der (eventuell invertiert) an den Hardware-Ausgang geschrieben werden soll.

5.12.3.2 Parameter(((HAL Digital Output Parameter))

(bit) invert
Wenn TRUE, wird **out** vor dem Schreiben in die Hardware invertiert.

5.12.3.3 Funktionen

(funct) write
Lesen Sie **out** und **invert** und stellen Sie die Hardwareausgabe entsprechend ein.

5.12.4 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.

5.12.4.1 Pins

(float) Wert
Der Hardware-Messwert, skaliert gemäß den Parametern **Skala** und **Offset**.
Wert = ((Eingangsmesswert, in hardwareabhängigen Einheiten) * **Skala**) - **Offset**

5.12.4.2 Parameter

(Float) Skala (engl. scale)
Die Eingangsspannung (oder der Strom) wird mit **Skala** multipliziert, bevor sie als **Wert** ausgegeben wird.

(float) Offset
Dieser Wert wird nach Anwendung des Skalenmultiplikators von der Hardware-Eingangsspannung (oder dem Strom) subtrahiert.

(float) bit_weight
Der Wert eines niederwertigen Bits (LSB). Dies ist effektiv die Granularität des Eingangswertes.

(float) hw_offset
Der Wert, der am Eingang anliegt, wenn 0 Volt an den Eingangspin(s) angelegt wird.

5.12.4.3 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

5.12.5 Analogausgang

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.

5.12.5.1 Pins

(float) Wert
Der zu schreibende Wert. Der tatsächliche Wert, der an die Hardware ausgegeben wird, hängt von den Parametern Skala und Offset ab.

(bit) aktivieren (engl. enable)
Wenn false, dann wird 0 an die Hardware ausgegeben, unabhängig vom **value** Pin.

5.12.5.2 Parameter(HAL Analog Output-Parameter)

(float) Offset
Dies wird zu dem **Wert** (engl. value) hinzugefügt, bevor die Hardware aktualisiert wird.

(Float) Skala (engl. scale)
Dies sollte so eingestellt werden, dass eine Eingabe von 1 am **value**-Pin dazu führt, dass der Analogausgangspin 1 Volt anzeigt.

(float) 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.

(float) 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.

(float) bit_weight (optional)
Der Wert des niedrigstwertigen Bits (LSB) in Volt (oder mA bei Stromausgängen).

(float) hw_offset (optional)
Die tatsächliche Spannung (oder Stromstärke), die ausgegeben wird, wenn 0 in die Hardware geschrieben wird.

5.12.5.3 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.

5.13 HAL-Werkzeuge

5.13.1 Halcmd

Halcmd ist ein Kommandozeilen-Tool für die Manipulation des HAL. Es gibt eine ziemlich vollständige Manpage für Link:../man/man1/halcmd.1.html[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`.

5.13.2 Halmeter

Halmeter is 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.

You can have many halmeters running at the same time, if you want to monitor several items. If you want to launch a halmeter without tying up a shell window, type `halmeter &` to run it in the

background. You can also make halmeter start displaying a specific item immediately, by adding "`pin|sig|par[am]` <*name*>" to the command line. It will display the pin, signal, or parameter <*name*> as soon as it starts - if there is no such item, it will simply start normally. And finally, if you specify an item to display, you can add *-s* before the pin|sig|param to tell halmeter to use a small window. The item name will be displayed in the title bar instead of under the value, and there will be no buttons. This is useful when you want a lot of meters in a small amount of screen space.

Weitere Informationen finden Sie im Abschnitt Halmeter Tutorial.

Halmeter can be loaded from a terminal or from AXIS. Halmeter is faster than Halshow at displaying values. Halmeter has two windows, one to pick the pin, signal, or parameter to monitor and one that displays the value. Multiple Halmeters can be open at the same time. If you use a script to open multiple Halmeters you can set the position of each one with the argument `-g X Y` relative to the upper left corner of your screen. For example:

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

Siehe die Manpage für weitere Optionen und den Abschnitt Halmeter.

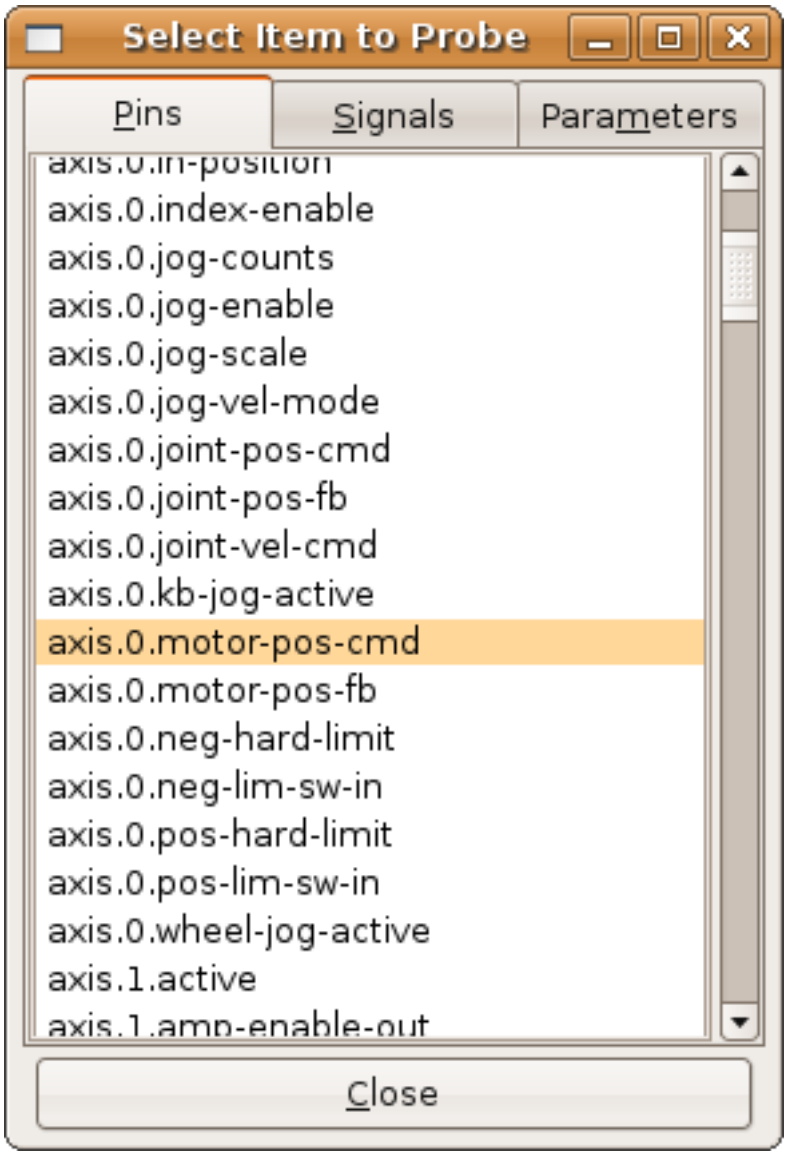


Abbildung 5.26: Halmeter-Auswahlfenster

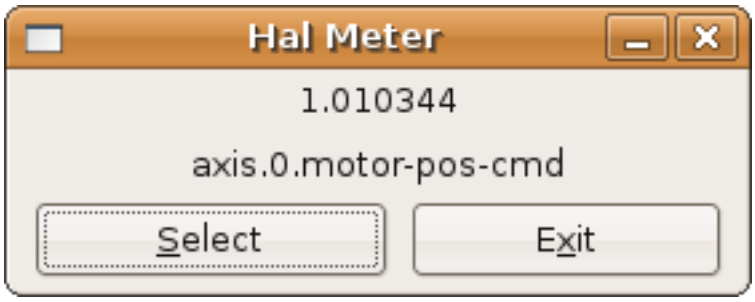


Abbildung 5.27: Halmeter-Watch Fenster

5.13.3 Halshow

Halshow (komplettee Beschreibung der Anwendung) kann von der Kommandozeile aus gestartet werden, um Details für ausgewählte Komponenten, Pins, Parameter, Signale, Funktionen und Threads einer laufenden HAL anzuzeigen. Die Registerkarte WATCH bietet eine kontinuierliche Anzeige der ausgewählten Pins, Parameter und Signale. Das Menü ”Datei” enthält Schaltflächen zum Speichern der Watch-Elemente in einer Watch-Liste und zum Laden einer bestehenden Watch-Liste. Die Elemente der Überwachungsliste können auch automatisch beim Starten geladen werden. Für die Verwendung in der Befehlszeile:

```
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.
```

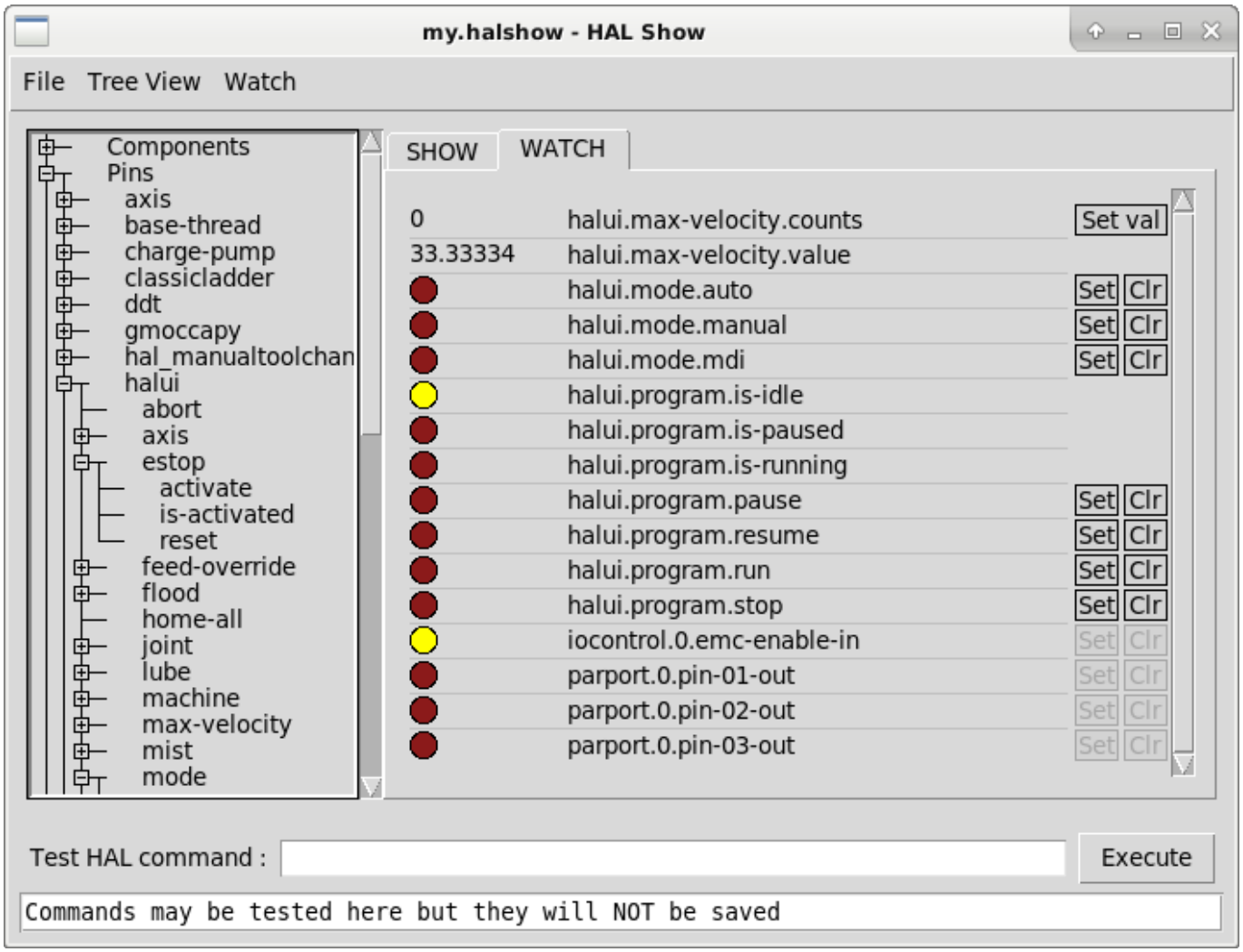


Abbildung 5.28: Halshow Watch Tab

A watchfile created using the *File/Save Watch List* menu item is formatted as a single line with tokens ”pin+”, ”param+”, ”sig=+” followed by the appropriate pin, param, or signal name. The token-name pairs are separated by a space character.

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.

5.13.4 Halscope

Halscope is an *oscilloscope* for the HAL. It lets you capture the value of pins, signals, and parameters as a function of time. Complete operating instructions should be located here eventually. For now, refer to section Abschnitt 5.4.6 in the tutorial chapter, which explains the basics.

The halscope ”File” menu selector provides buttons to save a configuration or open a previously saved configuration. When halscope is terminated, the last configuration is saved in a file named autosave.halscope.

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]
```

5.13.5 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)

  Note:  LinuxCNC (or a standalone HAL application) must be running
         A named item can specify a pin, param, or signal
         The item must be writable, e.g.:
            pin:    IN or I/O (and not connected to a signal with a writer)
            param:  RW
            signal: connected to a writable pin

         HAL-Elementtypen bit, s32, u32, float werden unterstützt

         When a bit 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 bit 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 &
```

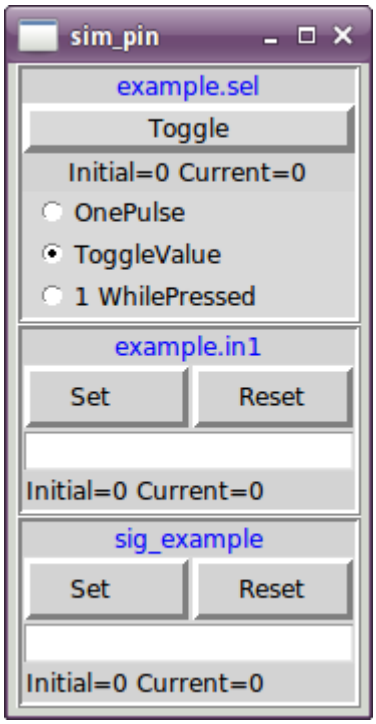


Abbildung 5.29: `sim_pin`-Fenster

5.13.6 simulate_probe (Sonde simulieren)

`simulate_probe` ist ein einfaches GUI, um die Aktivierung des Pins motion.probe-input zu simulieren. Verwendung:

```
simulate_probe &
```

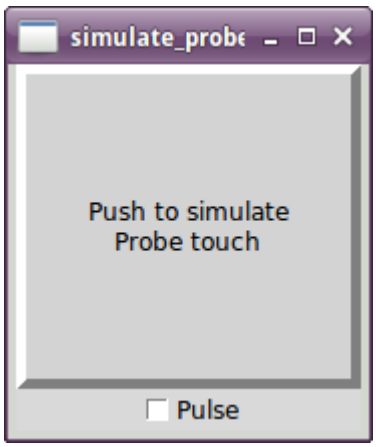


Abbildung 5.30: Fenster `simulieren_probe`

5.13.7 HAL Histogramm

`hal-histogram` ist ein Kommandozeilenprogramm zur Anzeige von Histogrammen für HAL-Pins.

Nutzung:

```
   hal-histogram --help | -?
or
   hal-histogram [Options] [pinname]
```

Tabelle 5.8: Options:

Option	**Wert**	**Beschreibung**
--minvalue	minvalue	minimum bin, Voreinstellung: 0
--binsize	binsize	binsize, Voreinstellung: 100
--nbins	nbins	Anzahl der Bins, Standard: 50
--logscale	0/1	y-Achse logarithmische Skala, Voreintellung: 1
--text	Hinweis	Textanzeige, Standard: ””
-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. If no pinname is specified, default is: `motion-command-handler.time`.
3. Diese App kann für 5 Pins geöffnet werden.
4. Unterstützt werden die Pintypen float, s32, u32, bit.
5. Der Pin muss mit einem Thread verbunden sein, der Fließkomma unterstützt. Für einen Basis-Thread kann dies die Verwendung von `loadrt motmod ... base_thread_fp=1` erfordern.

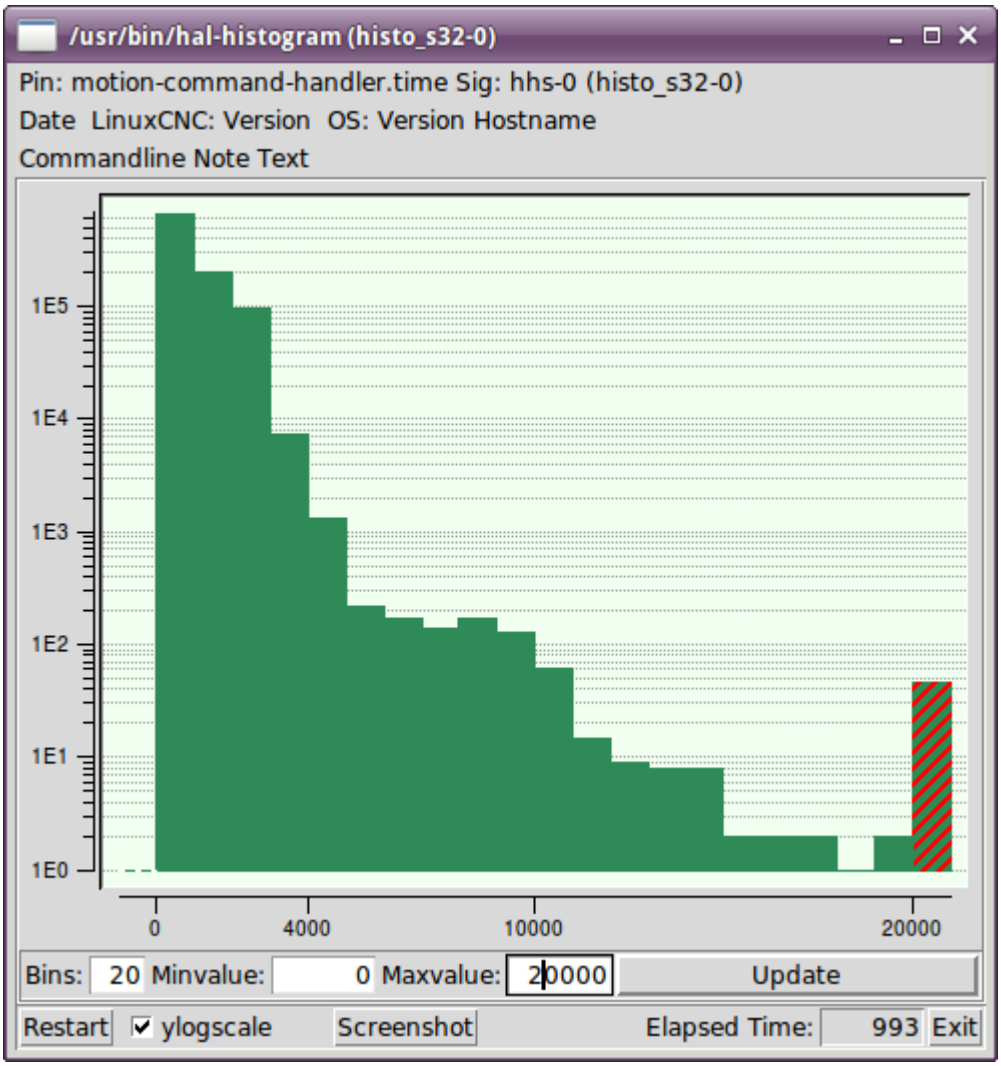


Abbildung 5.31: `hal-histogram`-Fenster

5.13.8 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. Pins von Userspace-Komponenten 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, User-Space-Pins usw. verarbeiten, ist die addf-Reihenfolge oft nicht kritisch. Außerdem kann das Timing der Wertänderungen von User-Space-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
```

Anmerkung
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 Userspace-Komponenten, denen Unterscheidungsmerkmale wie ein Präfix auf Basis des GUI-Programmnamens fehlen.

Fragwürdige Funktionen sind mit einem Fragezeichen ”?” gekennzeichnet.

Anmerkung
Komponentenstifte, die nicht mit einer bekannten Gewindefunktion verbunden werden können, melden die Funktion als ”Unbekannt”.

`halreport` generates a connections report (without pin types, and current values) for a running HAL application to aid in designing and verifying connections. This helps with the understanding what the source of a pin value is. Use this information with applications like `halshow`, `halmeter`, `halscope` or the `halcmd show` command in a terminal.

Kapitel 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.

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-Ohm-Widerstand von einem der Steuerpins nach 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 0V 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:

```
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.

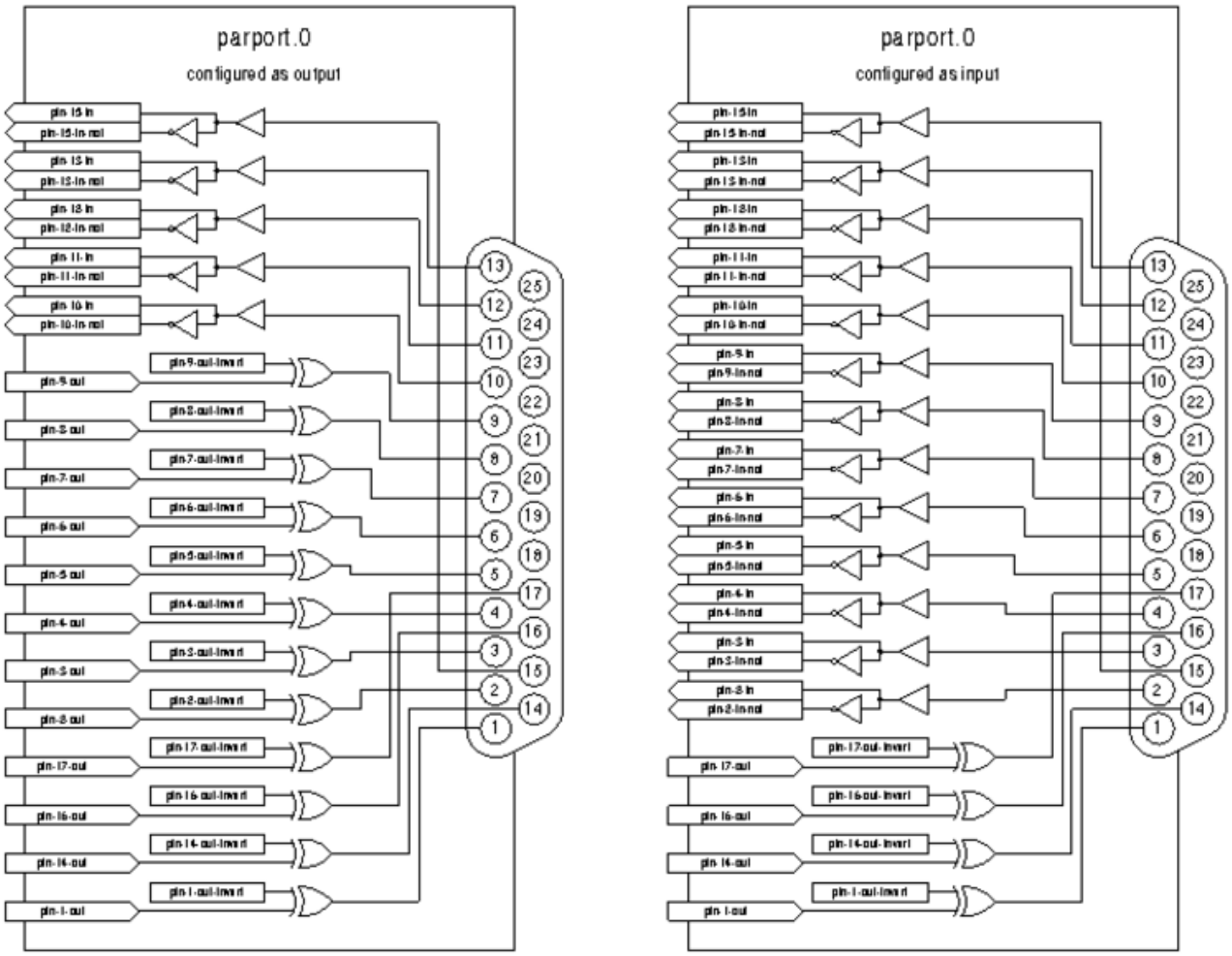


Abbildung 6.1: Parport-Blockdiagramm

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*.

Tabelle 6.1: 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

Tabelle 6.1: (continued)

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 +5V-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. Die Netmos-Karten sind Plug-N-Play, und könnte ihre Einstellungen ändern, je nachdem, welchen Steckplatz Sie sie in, so dass, wenn Sie \ "unter die Haube zu bekommen" und neu anordnen Dinge mögen, achten Sie darauf, diese Werte zu überprüfen, bevor Sie LinuxCNC starten.

6.1.3 Pins

- *parport.<p>.pin-<n>-out* (bit) Drives a physical output pin.
- parport.<p>.pin-<n>-in' (Bit) Verfolgt einen physikalischen Eingangs-Pin.
- parport.<p>.pin-<n>-in-not' (Bit) Verfolgt einen physischen Eingangs-Pin, aber invertiert.

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* (Bit) Invertiert einen Ausgangspin.
- *parport.<p>.pin-<n>-out-reset* (Bit) (nur für *out*-Pins) TRUE, wenn dieser Pin zurückgesetzt werden soll, wenn die *-reset*-Funktion ausgeführt wird.
- parport.<p>.reset-time' (U32) Die Zeit (in Nanosekunden) zwischen dem Setzen eines Pins durch *write* und dem Zurücksetzen durch die Funktion *reset*, wenn diese aktiviert ist.

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 auf TRUE aktiviert und durch FALSE deaktiviert. Wenn *-invert* TRUE ist, 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 <*portnum*> und aktualisiert die HAL *-in* und *-in-not* Pins.
- *parport.read-all* (funct) Liest die physischen Eingangspins aller Ports und aktualisiert die HAL *-in* und *-in-not* Pins.
- *parport.<p>.write* (funct) Liest die HAL *-out* Pins des Ports <*p*> und aktualisiert die physikalischen Ausgangspins dieses Ports.
- parport.write-all' (funct) Liest die HAL--*out*-Pins aller Ports und aktualisiert alle physischen 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. Wenn *-reset* TRUE ist, dann setzt die *reset*-Funktion den Pin auf den Wert von *-out-invert*. Dies kann in Verbindung mit *doublefreq* von stepgen verwendet werden, um einen Schritt pro Periode zu erzeugen. Der Schrittgenerator 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 Rücksetzzeit einstellen. So kann der Schritt in 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.

6.1.8.1 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 detected.
```

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 AX5214H Driver

Die Axiom Measurement & Control AX5214H ist eine digitale I/O-Karte mit 48 Kanälen. Sie wird an einen ISA-Bus angeschlossen und ähnelt einem Paar 8255-Chips. In der Tat könnte es ein Paar 8255-Chips sein, aber ich bin mir nicht sicher. Wenn jemand einen Treiber für einen 8255 entwickelt, sollte er sich den ax5214-Code ansehen, ein Großteil der Arbeit ist bereits erledigt.

6.2.1 Installation

```
loadrt hal_ax5214h cfg="<config-string>"
```

Der Konfigurationsstring besteht aus einer hexadezimalen Anschlussadresse, gefolgt von einer 8-stelligen Zeichenfolge aus "I" und "O", die Gruppen von Pins als Eingänge und Ausgänge festlegt. Die ersten beiden Zeichen legen die Richtung der ersten beiden 8-Bit-Blöcke von Pins (0-7 und 8-15) fest. Die nächsten beiden legen Blöcke von 4 Stiften fest (16-19 und 20-23). Das Muster wiederholt sich dann, zwei weitere Blöcke von 8 Bits (24-31 und 32-39) und zwei Blöcke von 4 Bits (40-43 und 44-47). Wenn mehr als eine Karte installiert ist, folgen die Daten für die zweite Karte auf die erste. Ein Beispiel: Der String *"0x220 IIIOIIOO 0x300 OIOOIOIO"* installiert Treiber für zwei Karten. Die erste Karte befindet sich an Adresse 0x220 und hat 36 Eingänge (0-19 und 24-39) und 12 Ausgänge (20-23 und 40-47). Die zweite Karte befindet sich an der Adresse 0x300 und hat 20 Eingänge (8-15, 24-31 und 40-43) und 28 Ausgänge (0-7, 16-23, 32-39 und 44-47). Es können bis zu 8 Karten in einem System verwendet werden.

6.2.2 Pins

- *(bit) ax5214.<boardnum>.out-<pinnum>* — Steuert einen physikalischen Ausgangspin.
- *(bit) ax5214.<boardnum>.in-<pinnum>* — Verfolgt einen physikalischen Eingangspin.
- *(bit) ax5214.<boardnum>.in-<pinnum>-not* — Verfolgt einen physikalischen Eingangspin, invertiert.

Für jeden Pin ist <boardnum> die Platinen-Nummer (beginnt bei Null) und <pinnum> die Nummer des E/A-Kanals (0 bis 47).

Beachten Sie, dass der Treiber von aktiven LOW-Signalen ausgeht. Dies ist erforderlich, damit Module wie OPTO-22 korrekt funktionieren (TRUE bedeutet Ausgang EIN oder Eingang unter Spannung). Wenn die Signale direkt ohne Pufferung oder Isolierung verwendet werden, muss die Inversion

berücksichtigt werden. Der In-HAL-Pin ist TRUE, wenn der physikalische Pin niedrig ist (OPTO-22-Modul unter Spannung), und FALSE, wenn der physikalische Pin hoch ist (OPTO-22-Modul aus). Der in-<pinnum>-not HAL-Pin ist invertiert - er ist FALSE, wenn der physikalische Pin low ist (OPTO-22-Modul unter Spannung). Durch Anschluss eines Signals an den einen oder anderen Pin kann der Benutzer den Zustand des Eingangs bestimmen.

6.2.3 Parameter

- *(bit) ax5214.<boardnum>.out-<pinnum>-invert* — Invertiert einen Ausgangspin.

Der Parameter -invert bestimmt, ob ein Ausgangspin aktiv high oder aktiv low ist. Wenn -invert auf FALSE steht, wird der physikalische Pin durch das Setzen von HAL out- pin TRUE auf low gesetzt, wodurch ein angeschlossenes OPTO-22-Modul eingeschaltet wird, und durch FALSE auf high gesetzt, wodurch das OPTO-22-Modul ausgeschaltet wird. Wenn -invert TRUE ist, wird durch das Setzen des HAL out-Pins TRUE der physikalische Pin auf high gesetzt und das Modul ausgeschaltet.

6.2.4 Funktionen

- *(funct) ax5214.<boardnum>.read* — Liest alle digitalen Eingänge auf einer Karte.
- *(funct) ax5214.<boardnum>.write* — Schreibt alle digitalen Ausgänge auf einer Karte.

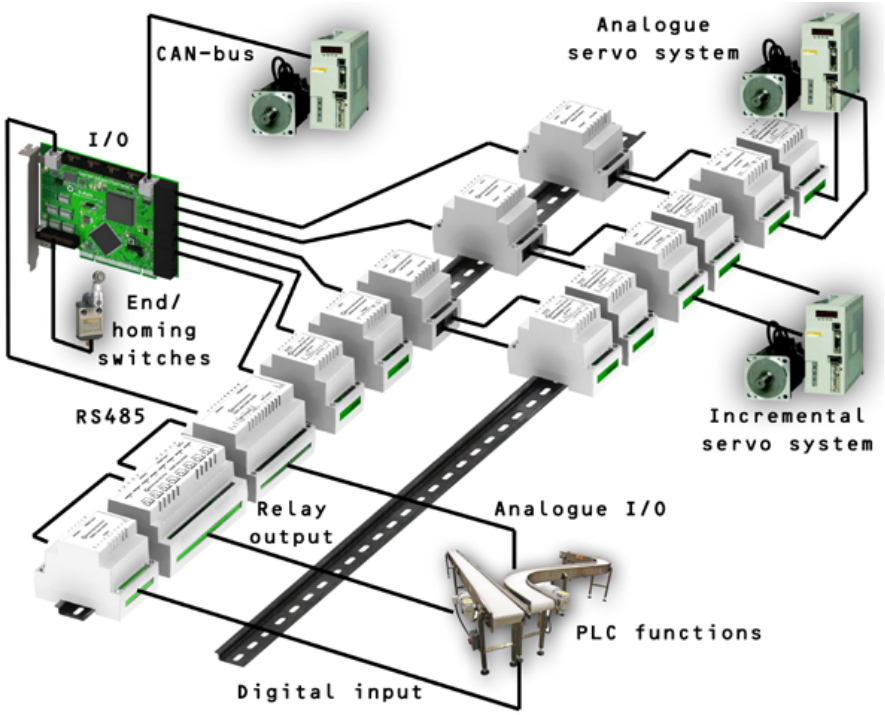
6.3 General Mechatronics Treiber

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 I/O expander modules: RS485 bus was designed for interfacing with compact DIN-rail mounted expander modules. An 8-channel digital input, an 8-channel relay output and an analogue I/O (4x +/-10 Volts output and 8x +/-5 Volts input) modules are available now. Up to 16 modules can be connected to the bus altogether.
- 20 optically isolated input pins: Six times three for the direct connection of two end switch and one homing sensor for each joint. And additionally, two optically isolated E-stop inputs.



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:

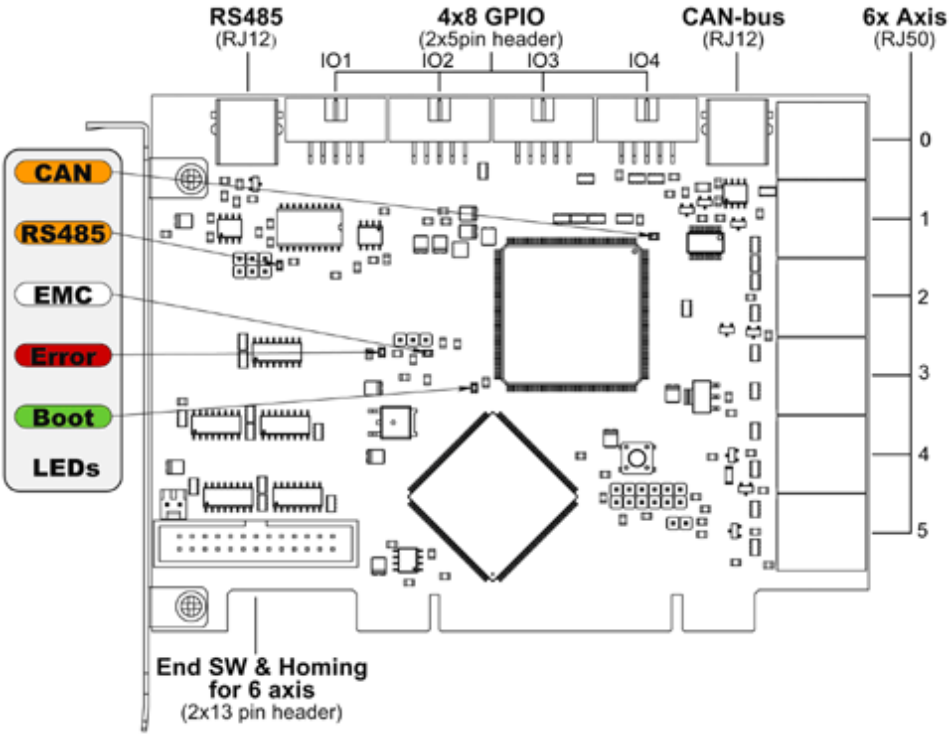


Abbildung 6.2: GM6-PCI-Kartenanschlüsse und LEDs

6.3.1 I/O-Anschlüsse

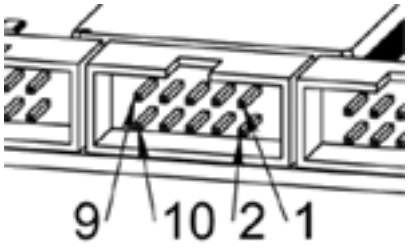


Abbildung 6.3: Pin-Nummerierung der GPIO-Anschlüsse

Tabelle 6.2: 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-Bewegungssteuerungs

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>
```

where <*gpio_con_no*> is from 0 to 3.

State of the first pin of the first GPIO connector on the GM6-PCI card.

```
gm.0.gpio.0.in-0
```

HAL pins are updated by function

```
gm.<card_no>.read
```

6.3.1.1 Pins

Tabelle 6.3: GPIO-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.in-<0-7>`	(bit, Out)	Input-Pin (wörtlich Eingabe-Pin)
`.in-not-<0-7>`	(bit, Out)	Negierter Eingangspin
`.out-<0-7>`	(bit, In)	Ausgangspin. Wird nur verwendet, wenn GPIO auf Ausgang eingestellt ist.

6.3.1.2 Parameter

Tabelle 6.4: GPIO-Parameter

Pins	Typ und Richtung	Parameterbeschreibung
`.is-out-<0-7>`	(bit, R/W)	Bei True wird der entsprechende GPIO auf Totem-Pol-Ausgang gesetzt, andernfalls auf hochohmigen Eingang.
`.invert-out-<0-7>`	(bit, R/W)	Wenn True, wird der Wert des Pins invertiert. Wird verwendet, wenn der Pin als Ausgang konfiguriert ist.

6.3.2 Achsen-Anschlüsse

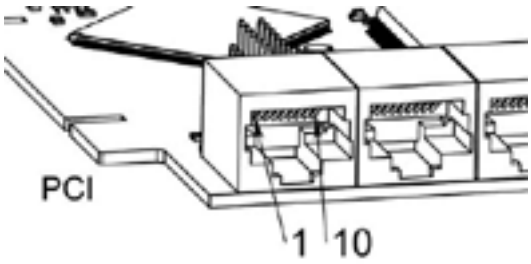


Abbildung 6.4: Pin-Nummerierung der Achsenverbinder

Tabelle 6.5: 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

6.3.2.1 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:

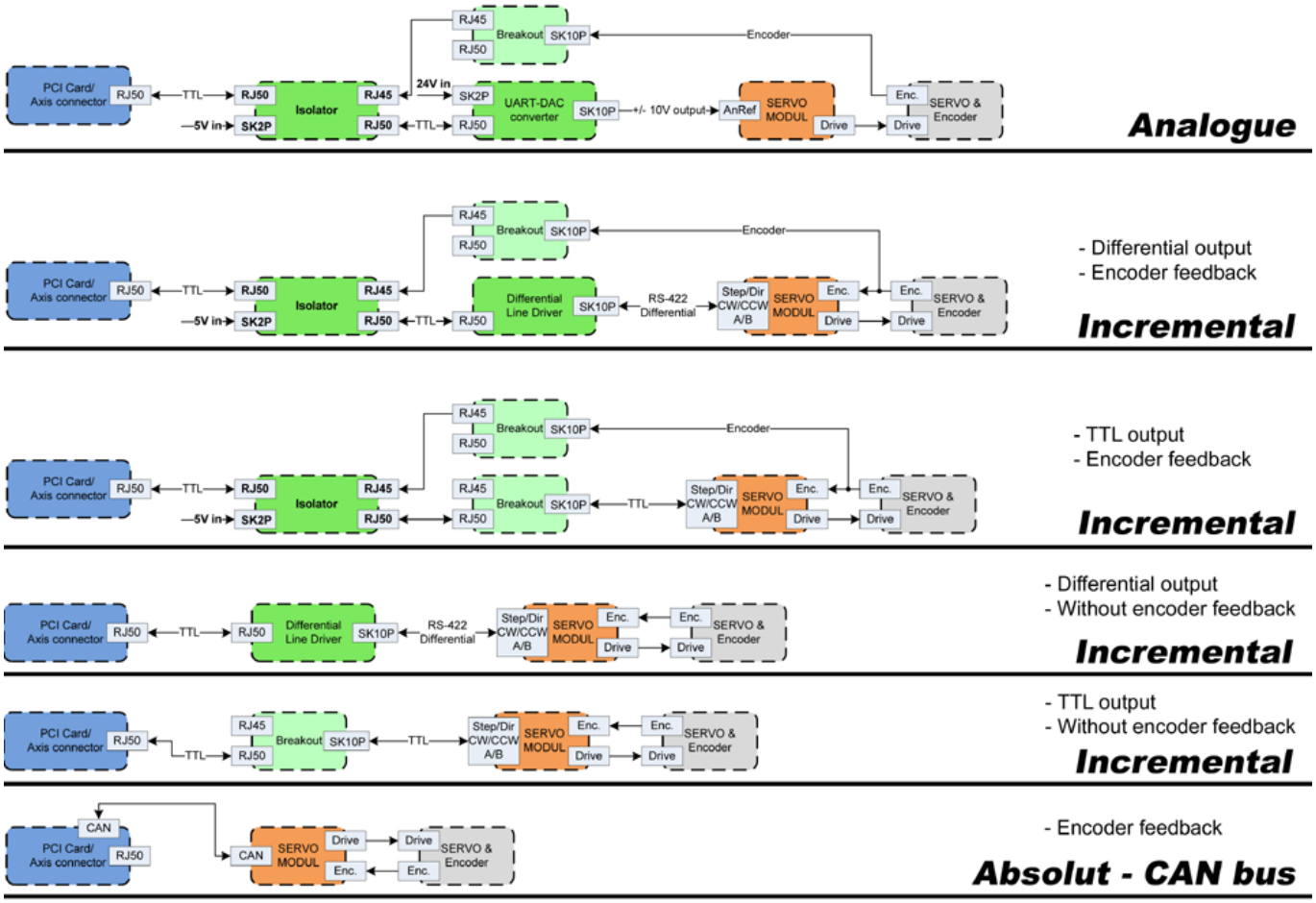


Abbildung 6.5: Servo-Achsen-Schnittstellen

6.3.2.2 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>
```

where *<axis_no>* is from 0 to 5. For example, `gm.0.encoder.0.position` refers to the position of encoder module of axis 0.

Die GM6-PCI-Karte zählt das Gebersignal unabhängig von LinuxCNC. HAL-Pins werden nach Funktion aktualisiert:

```
gm.<card_no>.read
```

Tabelle 6.6: Encoder-Pins

Pins	Typ und Richtung	Pin-Beschreibung
.reset	(bit, In)	Wenn True, setzt Anzahl und Position auf Null zurück.
.rawcounts	(s32, Out)	Die ”raw counts” ist dieselbe Zählung, aber unbeeinflusst von der Rückstellung oder dem Indeximpuls.
.counts	(s32, Out)	Position in Encoder-Zählungen (engl. counts).
.position	(float, Out)	Position in skalierten Einheiten (=.counts/.position-scale).
.index-enabled	(bit, 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-enabled pin is set to 0 and remains 0 until connected HAL pin does not set it.
.velocity	(float, Out)	Velocity in scaled units per second. GM encoder uses high frequency hardware timer to measure time between encoder pulses in order to calculate velocity. It greatly reduces quantization noise as compared to simply differentiating the position output. When the measured velocity is below min-speed-estimate, the velocity output is 0.

Tabelle 6.7: Encoder-Parameter

Parameter	Typ und Lesen/-Schreiben	Parameterbeschreibung
.counter-mode	(bit, R/W)	When True, the counter counts each rising edge of the channel-A input to the direction determined by channel-B. This is useful for counting the output of a single channel (non-quadrature) or step/dir signal sensor. When false, it counts in quadrature mode.
.index-mode	(bit, R/W)	When True and .index-enabled is also true, .counts and .position are rounded (based on .counts-per-rev) at rising edge of channel-I. This is useful to correct few pulses error caused by noise. In round mode, it is essential to set .counts-per-rev parameter correctly. When .index-mode is False and .index-enabled is true, .counts and .position are reset at channel-I pulse.

Tabelle 6.7: (continued)

Parameter	Typ und Lesen/-Schreiben	Parameterbeschreibung
.counts-per-rev	(s32, R/V)	Determine how many counts are between two index pulses. It is used only in round mode, so when both .index-enabled and .index-mode parameters are True. GM encoder process encoder signal in 4x mode, so for example in case of a 500 CPR encoder it should be set to 2000. This parameter can be easily measured by setting .index-enabled True and .index-mode False (so that .counts resets at channel-I pulse), than move axis by hand and see the maximum magnitude of .counts pin in halmeter.
.index-invert	(bit, R/W)	When True, channel-I event (reset or round) occur on falling edge of channel-I signal, otherwise on rising edge.
.min-speed-estimate	(float, R/W)	Determine the minimum measured velocity magnitude at which .velocity will be set as nonzero. Setting this parameter too low will cause it to take a long time for velocity to go to zero after encoder pulses have stopped arriving.
.position-scale	(float, R/W)	Scale in counts per length unit. .position=.counts/.position-scale. For example, if position-scale is 2000, then 1000 counts of the encoder will produce a position of 0.5 units.

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 process encoder in 4x mode, 4x500=2000
setp gm.0.encoder.0.index-invert 0         #
setp gm.0.encoder.0.min-speed-estimate 0.1 # in position unit/s
setp gm.0.encoder.0.position-scale 20000   # 10 encoder rev cause the machine to move one  ↩
    position unit (10x2000)
```

Connect encoder position to LinuxCNC joint position feedback

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

6.3.2.3 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>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.stepgen.0.position-cmd` refers to the position command of StepGen module of axis 0 on card 0.

Die GM6-PCI-Karte erzeugt unabhängig von LinuxCNC Schrittimpulse. Die HAL-Pins werden geupdatet durch die Funktion

```
gm.<card_no>.write
```

Tabelle 6.8: StepGen Modul Pins

Pins	**Typ und Richtung**	**Pin-Beschreibung**
`.enable`	(bit, In)	StepGen erzeugt nur dann Impulse, wenn dieser Pin ”true” ist.
`.count-fb`	(s32, Out)	Positionsrückmeldung in Zähleinheiten.
`.position-fb`	(float, Out)	Positionsrückmeldung in Positionseinheit.
`.position-cmd`	(float, In)	Die befohlene Position in Positionseinheiten. Wird nur im Positionsmodus verwendet.
`.velocity-cmd`	(float, In)	Geforderte Geschwindigkeit in Positionseinheiten pro Sekunde. Wird nur im Geschwindigkeitsmodus verwendet.

Tabelle 6.9: StepGen-Modul-Parameter

Parameter	**Typ und Lesen/-Schreiben**	**Parameterbeschreibung**
`.step-type`	(u32, 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`	(bit, R/W)	When True, .velocity-cmd is used as reference and velocityvcontrol calculate pulse rate output. When False, .position-cmd is used as reference and position control calculate pulse rate output.
`.invert-step1`	(bit, R/W)	Invertieren des Ausgangs von Kanal 1 (Step-Signal im StepDir-Modus)
`.invert-step2`	(bit, R/W)	Invertierung des Ausgangs von Kanal 2 (Dir-Signal im StepDir-Modus)
`.maxvel`	(float, R/W)	Maximum velocity in position units per second. If it is set to 0.0, .maxvel parameter is ignored.
`.maxaccel`	(float, R/W)	Maximum acceleration in position units per second squared. mf it is set to 0.0, .maxaccel parameter is ignored.
`.position-scale`	(float, R/W)	Skala in Schritten pro Längeneinheit.

Tabelle 6.9: (continued)

Parameter	**Typ und Lesen/-Schreiben**	**Parameterbeschreibung**
`.steplen`	(u32, R/W)	Länge des Schritt-Pulses (engl. step pulse) in Nanosekunden.
`.stepspace`	(u32, R/W)	Mindestzeit zwischen zwei Schrittimpulsen in Nanosekunden.
`.dirdelay`	(u32, R/W)	Minimum time between step pulse and direction change in nanoseconds.

Zur Ermittlung der entsprechenden Werte siehe die nachstehenden Zeitdiagramme:

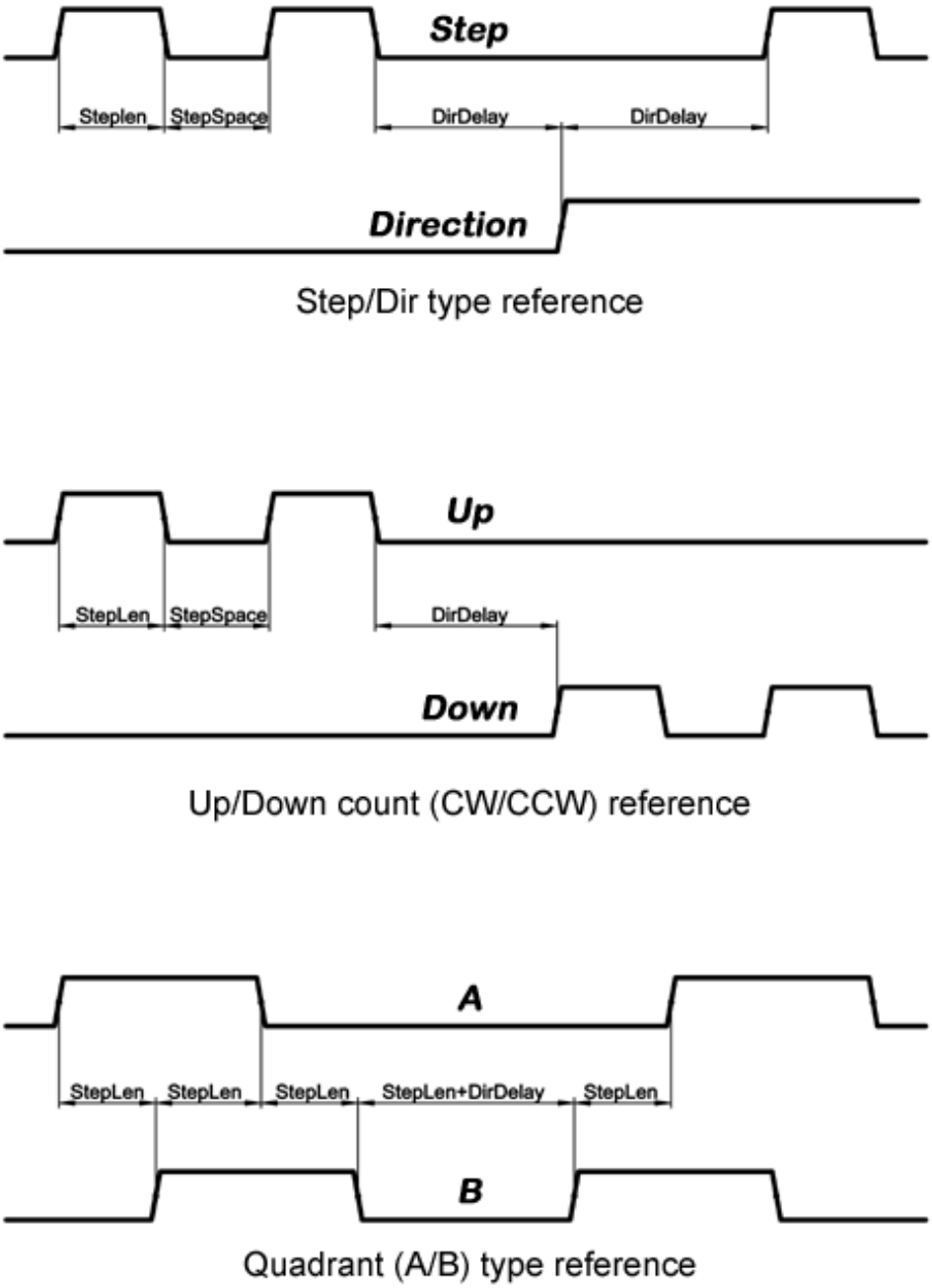


Abbildung 6.6: Referenzsignal-Zeitdiagramme

Setting StepGen module of axis 0 to generate 1000 step pulse per position unit

```
setp gm.0.stepgen.0.step-type 0         # 0:stepDir, 1:UpDown, 2:Quad
setp gm.0.stepgen.0.control-type 0      # 0:Pos. Kontrolle, 2:Geschw. Kontrolle
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.stepspace1000       # 1000 ns = 1 µs
```

```
setp gm.0.stepgen.0.dirdelay 2000       # 2000 ns = 2 µs
```

Connect StepGen to axis 0 position reference and enable pins

```
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
```

6.3.2.4 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
```

Tabelle 6.10: Aktivierungs- und Fehlersignal-Pins

Pins	Typ und Richtung	Pin-Beschreibung
gm.<card_no>.power-enable	(bit, In)	If this pin is True, * and Watch Dog Timer is not expired * and there is no power fault then power enable pins of axis- and CAN connectors are set to high, otherwise set to low.
gm.<card_no>.power-fault	(bit, Out)	Stromausfall-Eingang.

6.3.2.5 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>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.dac.0.value` refers to the output voltage of DAC module of axis 0.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.write
```

Tabelle 6.11: Achsen DAC-Pins

Pins	Typ und Richtung	Pin-Beschreibung
.enable	(bit, In)	Enable DAC output. When enable is false, DAC output is 0.0 V.
.value	(float, In)	Wert des DAC-Ausgangs in Volt.

Tabelle 6.12: Achsen-DAC-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
.offset	(float, R/W)	Der Offset wird zu dem Wert addiert, bevor die Hardware aktualisiert wird.
.high-limit	(float, R/W)	Maximum output voltage of the hardware in Volts.
.Untergrenze	(float, R/W)	Minimum output voltage of the hardware in Volts.
.invert-seriell	(float, R/W)	GM6-PCI card is communicating with DAC hardware via fast serial communication to highly reduce time delay compared to PWM. DAC module is recommended to be isolated which is negating serial communication line. In case of isolation, leave this parameter to default (0), while in case of none-isolation, set this parameter to 1.

6.3.3 CAN-Bus-Servoverstärker

The GM6-PCI motion control card has CAN module to drive CAN servo amplifiers. Implementation of higher level protocols like CANopen is further development. Currently GM produced power amplifiers has upper level driver which export pins and parameters to HAL. They receive position reference and provide encoder feedback via CAN bus.

The frames are standard (11 bit) ID frames, with 4 byte data length. The BAUD rate is 1 Mbit/s. The position command IDs for axis 0..5 are 0x10..0x15. The position feedback IDs for axis 0..5 are 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` refers to the output position of axis 0 in position units.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.write
```

6.3.3.1 Pins

Tabelle 6.13: CAN-Modul-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.enable`	(bit, In)	Aktivieren das Senden von Positionsreferenzen.
`.position-cmd`	(float, In)	Befohlene Position in Positionseinheiten.
`.position-fb`	(float, In)	Rückmeldung der Position in Positionseinheiten.

6.3.3.2 Parameter

Tabelle 6.14: CAN-Modul-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
`.position-scale`	(float, R/W)	Maßstab in Längeneinheiten.

6.3.4 Watchdog-Timer

Watchdog-Timer wird zurückgesetzt bei Funktion:

```
gm.<card_no>.read
```

6.3.4.1 Pins

Tabelle 6.15: Watchdog-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`gm.<card_no>.watchdog-expired`	(bit, 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.

6.3.4.2 Parameter

Tabelle 6.16: Watchdog-Parameter

Parameter	Typ und Richtung	Parameterbeschreibung
gm.<*card_no*>.watchdog-enable	(bit, R/W)	Enables watchdog timer. It is strongly recommended to enable the watchdog timer, because it can disable all the servo amplifiers by pulling down all enable signals in case of a PC error.
gm.<*card_no*>.watchdog-timeout	(float, R/W)	Time interval in within the gm.<*card_no*>.read function must be executed. The gm.<*card_no*>.read is typically added to servo-thread, so watch timeout is typically set to 3 times of the servo period.

6.3.5 End-, Referenzpunkt- und Notaus-Schalter

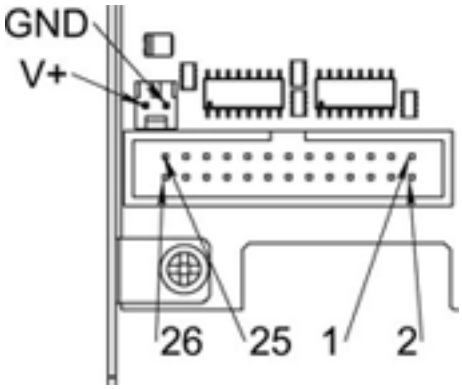


Abbildung 6.7: Pin-Nummerierung des Anschlusses für Referenzfahrt und Endschalter

Tabelle 6.17: Pinbelegung des End- und Referenzschalter-anschlusses

25	23	21	19	17	15	13	11	9	7	5	3	1
GND		1/End-	2/End+	2/Homing	3/End-	4/End+	4/Homing	5/End-	6/End+	6/Homing	Notaus 2	V+ (Ext.)

26	24	22	20	18	16	14	12	10	8	6	4	2
GND		1/End+	1/Homing	2/End-	3/End+	3/Homing	4/End-	5/End+	5/Homing	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>.joint.<axis_no>
```

where *<axis_no>* is from 0 to 5. For example, `gm.0.joint.0.home-sw-in` indicates the state of the axis 0 home switch.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.read
```

6.3.5.1 Pins

Tabelle 6.18: End- und Referenzschalter-Pins

Pins	Typ und Richtung	Pin-Beschreibung
`.home-sw-in`	(bit, Out)	Eingang des Referenzschalters
`.home-sw-in-not`	(bit, Out)	Negierter Referenzschaltereingang
`.neg-lim-sw-in`	(bit, Out)	Negativer Endschaltereingang
`.neg-lim-sw-in-not`	(bit, Out)	Negierter negativer Endschaltereingang
`.pos-lim-sw-in`	(bit, Out)	Positiver Endschaltereingang
`.pos-lim-sw-in-not`	(bit, Out)	Negierter positiver Endschaltereingang

6.3.5.2 Parameter

Tabelle 6.19: Parameter des Notaus-Schalters

Parameter	Typ und Richtung	Parameterbeschreibung
`gm.0.estop.0.in`	(bit, Out)	Notaus0 Eingang
`gm.0.estop.0.in-not`	(bit, Out)	Negierter Notaus 0-Eingang
`gm.0.estop.1.in`	(bit, Out)	Notaus 1 Eingang
`gm.0.estop.1.in-not`	(bit, Out)	Negierter Notaus 1-Eingang

6.3.6 Status-LEDs

6.3.6.1 CAN

Farbe: Orange

- Blinken während der Datenkommunikation.
- Ein, wenn einer der Puffer voll ist - Kommunikationsfehler.
- Aus, wenn keine Datenkommunikation stattfindet.

6.3.6.2 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.

6.3.6.3 EMC

Farbe: Weiß

- Blinken, wenn LinuxCNC läuft.
- Sonst aus.

6.3.6.4 Booten

Farbe: Grün

- Ein, wenn das System erfolgreich gebootet wurde.
- Sonst aus.

6.3.6.5 Fehler

Farbe: Rot

- Aus, wenn keine Störung im System vorliegt.
- Blinkt, wenn ein PCI-Kommunikationsfehler vorliegt.
- Ein, wenn der Watchdog-Timer übergelaufen ist.

6.3.7 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.

Automatic node recognizing 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.

Fault handling 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.

Connecting the nodes The modules on the bus have to be connected in serial topology, with termination resistors on the end. The start of the topology is the PCI card, and the end is the last module.

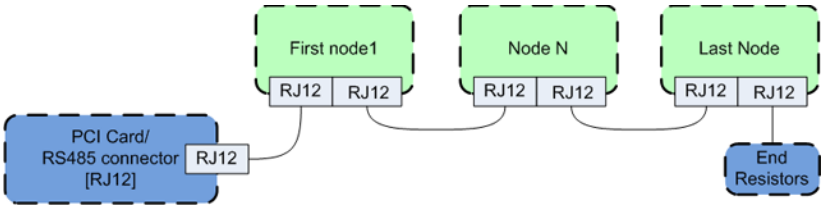


Abbildung 6.8: Anschließen der RS485-Knoten an die GM6-PCI-Karte

Addressing 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.

6.3.7.1 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>
```

where *<module ID>* is from 00 to 15.

Tabelle 6.20: Pins des Relaisausgangsmoduls

Pins	Typ und Richtung	Pin-Beschreibung
`.relay-<0-7>`	(bit, Out)	Ausgangspin für Relais

Tabelle 6.21: Parameter des Relaisausgangsmoduls

Parameter	Typ und Richtung	Parameterbeschreibung
.invert-relay-<0-7>	(bit, R/W)	Relais-Ausgangsstift negieren

HAL-Beispiel

```
  gm.0.rs485.0.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
```

6.3.7.2 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>
```

where <*module ID*> is from 00 to 15.

Tabelle 6.22: Pins des digitalen Eingangs-/Ausgangsmoduls

Pins	Typ und Richtung	Pin-Beschreibung
.in-<*0-7*>	(bit, Out)	Input-Pin (wörtlich Eingabe-Pin)
.in-not-<*0-7*>	(bit, Out)	Negierter Eingangspin

HAL-Beispiel

```
  gm.0.rs485.0.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
```

6.3.7.3 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>
```

where <*module ID*> is from 00 to 15.

Tabelle 6.23: DAC- & ADC-Modul-Pins

Pins	Typ und Richtung	Pin-Beschreibung
.adc-<*0-7*>	(float, Out)	Wert des ADC-Eingangs in Volt.
.dac-enable-<*0-3*>	(bit, In)	Enable DAC output. When enable is false then DAC output is set to 0.0 V.
.dac-<*0-3*>	(float, In)	Wert des DAC-Ausgangs in Volt.

Tabelle 6.24: Parameter des DAC- & ADC-Moduls

Parameter	Typ und Richtung	Parameterbeschreibung
.adc-scale-<*0-7*>	(float, R/W)	The input voltage will be multiplied bymscale before being output to .adc- pin.
.adc-offset-<*0-7*>	(float, R/W)	Der Offset wird von der Hardware-Eingangsspannung subtrahiert, nachdem der Skalenmultiplikator angewendet wurde.
.dac-offset-<*0-3*>	(float, R/W)	Der Offset wird zu dem Wert addiert, bevor die Hardware aktualisiert wird.
.dac-high-limit-<*0-3*>	(float, R/W)	Maximum output voltage of the hardware in Volts.
.dac-low-limit-<*0-3*>	(float, R/W)	Minimum output voltage of the hardware in Volts.

HAL-Beispiel

```
  gm.0.rs485.0.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
```

6.3.7.4 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.

Tabelle 6.25: Pins des Teach-Pendant-Moduls

Pins	Typ und Richtung	Pin-Beschreibung
.adc-<*0-5*>	(float, Out)	Wert des ADC-Eingangs in Volt.
.enc-reset	(bit, In)	Wenn True, setzt Anzahl und Position auf Null zurück.
.enc-counts	(s32, Out)	Position in Encoder-Zählungen (engl. counts).
.enc-rawcounts	(s32, Out)	Die rohen Zähldaten (engl. raw counts) ist die Zählung, die durch das Zurücksetzen nicht beeinflusst wird.
.enc-position	(float, Out)	Position in skalierten Einheiten (=.enc-counts/.enc-position-scale).
.in-<*0-7*>	(bit, Out)	Input-Pin (wörtlich Eingabe-Pin)
.in-not-<*0-7*>	(bit, Out)	Negierter Eingangspin

Tabelle 6.26: Parameter des Teach Pendant Moduls

Parameter	Typ und Richtung	Parameterbeschreibung
.adc-scale-<*0-5*>	(float, R/W)	Die Eingangsspannung wird mit der Skalierung multipliziert, bevor sie an den .adc-Pin ausgegeben wird.
.adc-offset-<*0-5*>	(float, R/W)	Der Offset wird von der Hardware-Eingangsspannung subtrahiert, nachdem der Skalenmultiplikator angewendet wurde.
.enc-position-scale	(float, R/W)	Maßstab in Längeneinheiten.

HAL-Beispiel

```
  gm.0.rs485.0.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
```

6.3.8 Errata

6.3.8.1 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.

6.4 GS2 VFD-Treiber

Dies ist ein Userspace-HAL-Programm für die GS2-Serie von VFD's bei Automation Direct. Fußnote:[In Europa kann das Äquivalent unter dem Markennamen Omron gefunden werden.]

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.

6.4.1 Kommandozeilen-Optionen

- *-b* oder *--bits* <n> (Standard: 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>* (Standard: 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.

Anmerkung
Bei seriellen Konfigurationsfehlern kann das Einschalten von verbose zu einer Flut von Timeout-Fehlern führen.

6.4.2 Pins

Dabei ist <*name*> ”gs2_vfd” oder der Name, der beim Laden mit der Option `-n` angegeben wurde:

- <name>.*DC-bus-volts* (float, out) Zwischenkreisspannung des VFD
- <name>.*at-speed* (bit, out), wenn der Antrieb die befohlene Geschwindigkeit erreicht
- <name>.*err-reset* (bit, in) Reset-Fehler, die an VFD gesendet werden
- <name>.*firmware-revision* (s32, out) vom VFD
- <name>.*frequency-command* (float, out) vom VFD
- <name>.*frequency-out* (float, out) aus dem VFD
- <name>.*is-stopped* (Bit, aus), wenn der Frequenzumrichter 0 Hz-Ausgang meldet
- <name>.*load-percentage* (float, out) vom VFD
- <name>.*motor-RPM* (float, out) vom VFD
- <name>.*output-current* (float, out) vom VFD
- <name>.*Ausgangsspannung* (float, out) vom VFD
- <name>.*power-factor* (float, out) vom VFD
- <name>.*scale-frequency* (float, out) vom VFD
- <name>.*speed-command* (float, in) an den VFD gesendete Geschwindigkeit in U/min. Es ist ein Fehler, eine Geschwindigkeit zu senden, die höher ist als die im VFD eingestellte Motor Max U/min (engl. RPM).
- <name>.*spindle-fwd* (bit, in) 1 für FWD (engl. kurz für forwärts) und 0 für REV (engl. kurz für rückwärts) an den VFD gesendet
- <name>.*spindle-rev* (bit, in) 1 für REV und 0 wenn aus
- <name>.*spindle-on* (bit, in) 1 für EIN und 0 für AUS an VFD gesendet
- <name>.*status-1* (s32, out) Antriebsstatus des VFD (siehe GS2-Handbuch)
- <name>.*status-2* (s32, out) Laufwerksstatus des Frequenzumrichters (siehe GS2-Handbuch)

Anmerkung
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).

6.4.3 Parameter

Dabei ist <*name*> gs2_vfd oder der Name, der beim Laden mit der Option `-n` angegeben wurde:

- <name>.*error-count* (s32, RW)
- <name>.*loop-time* (float, RW) wie oft der Modbus abgefragt wird (Standard: 0.1)
- <name>.*nameplate-HZ* (float, RW) Typenschild-Hz des Motors (Voreinstellung: 60)
- <name>.*nameplate-RPM* (float, RW) Typenschild-Drehzahl des Motors in U/min (Voreinstellung: 1730)
- <name>.*retval* (s32, RW) der Rückgabewert eines Fehlers in HAL
- <name>.*tolerance* (s32, RW) Geschwindigkeitstoleranz (Voreinstellung: 0.01)
- <name>.*ack-delay* (s32, RW) Anzahl der Lese-/Schreibzyklen vor der Überprüfung bei Geschwindigkeit (Voreinstellung: 2)

Ein Beispiel für die Verwendung dieser Komponente zum Antreiben einer Spindel finden Sie im Beispiel GS2 Spindel.

6.5 Mesa HostMot2-Treiber

6.5.1 Einführung

HostMot2 ist eine FPGA-Konfiguration, die von Mesa Electronics für ihre *Anything I/O*-Bewegungssteuerung 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.

6.5.2 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-Stifte): 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-Stifte): 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.

6.5.3 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.

6.5.4 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.

6.5.5 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:

6.5.5.1 Pins

- has_bit' - (bit i/o) True, wenn der Watchdog ein Bit hat, False, wenn der Watchdog kein Bit hat. Wenn der Watchdog ein Bit hat und das has_bit-Bit True ist, kann der Benutzer es auf False zurücksetzen, um den Betrieb wieder aufzunehmen.

6.5.5.2 Parameter

- timeout_ns' - (u32 read/write) Watchdog-Timeout, in Nanosekunden. Dieser Wert wird beim Laden des Moduls auf 5.000.000 (5 Millisekunden) initialisiert. Wenn zwischen den Aufrufen der hm2-Schreibfunktion mehr als diese Zeitspanne vergeht, wird der Watchdog aktiv.

6.5.6 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.)

Anmerkung
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.

6.5.7 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
```

Anmerkung
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.

6.5.8 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>/
```

6.5.9 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*.

6.5.10 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.

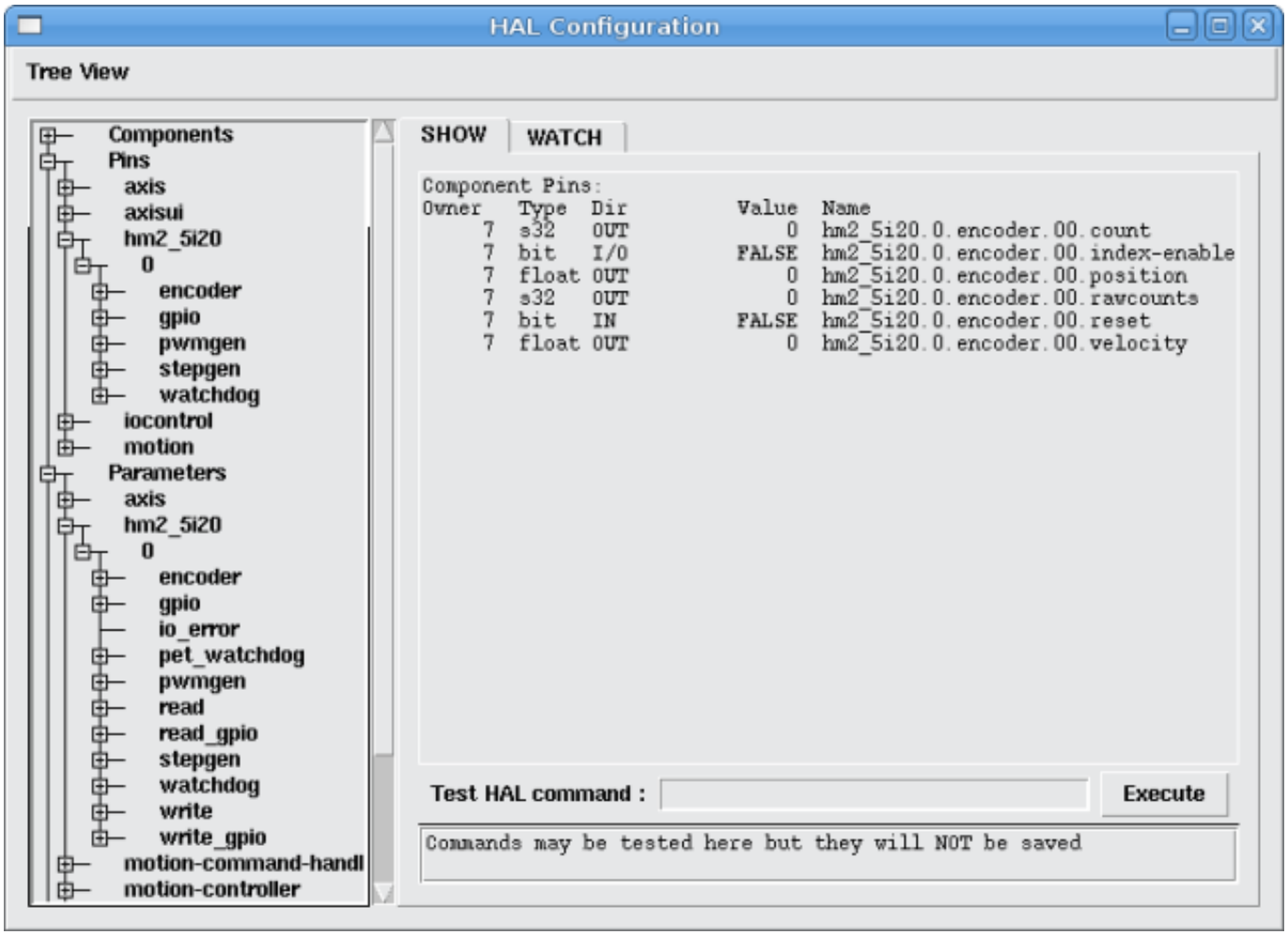


Abbildung 6.9: 5i20 HAL-Pins

6.5.11 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
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 Gates.)

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 Gates.)

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
SVST8_4	8	8	4 (tbl5)	0
SVST8_4IM2	8 (+IM)	8	4	8

4i68 (3-Port PC/104) Standardkonfigurationen (Die 4i68 hat 400k Gates.)

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
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.

6.5.12 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 geschrie-

ben, 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.

6.5.12.1 Pins

- *in* - (Bit, Out) Normaler Zustand des Hardware-Eingangs-Pins. Sowohl volle GPIO-Pins als auch I/O-Pins, die von aktiven Modulinstanzen als Eingänge verwendet werden, haben diesen Pin.
- *in_not* - (Bit, Out) Invertierter Zustand des Hardware-Eingangs-Pins. Sowohl volle GPIO-Pins als auch I/O-Pins, die von aktiven Modulinstanzen als Eingänge verwendet werden, haben diesen Pin.
- *out* - (Bit, In) Wert, der (möglicherweise invertiert) an den Hardware-Ausgangspin geschrieben werden soll. Nur volle GPIO-Pins haben diesen Pin.

6.5.12.2 Parameter

- *invert_output* - (Bit, RW) Dieser Parameter hat nur eine Auswirkung, wenn der Parameter *is_output* wahr ist. Wenn dieser Parameter wahr ist, wird der Ausgangswert des GPIOs der Inverse des Wertes am *out* HAL-Pin sein. Nur vollständige GPIO-Pins und I/O-Pins, die von aktiven Modulinstanzen als Ausgänge verwendet werden, haben diesen Parameter. Um einen aktiven Modul-Pin zu invertieren, müssen Sie den GPIO-Pin invertieren, nicht den Modul-Pin.
- *is_opendrain* - (Bit, RW) Dieser Parameter hat nur eine Auswirkung, wenn der Parameter *is_output* wahr ist. Wenn dieser Parameter false ist, verhält sich der GPIO wie ein normaler Ausgangspin: der I/O-Pin am Stecker wird auf den Wert getrieben, der durch den *out* HAL-Pin angegeben ist (möglicherweise invertiert), und der Wert der *in* und *in_not* HAL-Pins ist undefiniert. Wenn dieser Parameter true ist, verhält sich der GPIO wie ein Open-Drain-Pin. Das Schreiben von 0 an den *out* HAL-Pin treibt den I/O-Pin auf low, das Schreiben von 1 an den *out* HAL-Pin versetzt den I/O-Pin in einen hochohmigen Zustand. In diesem hochohmigen Zustand schwebt der I/O-Pin (schwach hochgezogen), und andere Geräte können den Wert treiben; der resultierende Wert am I/O-Pin ist an den *in* und *in_not* Pins verfügbar. Nur vollständige GPIO-Pins und I/O-Pins, die von aktiven Modulinstanzen als Ausgänge verwendet werden, haben diesen Parameter.
- *is_output* - (Bit, RW) Wenn auf 0 gesetzt, ist der GPIO ein Eingang. Der I/O-Pin wird in einen hochohmigen Zustand versetzt (schwach hochgezogen - engl. weakly pulled high), um von anderen Geräten angesteuert zu werden. Der logische Wert am I/O-Pin ist in den HAL-Pins *in* und *in_not* verfügbar. Schreibvorgänge auf den *out* HAL-Pin haben keine Auswirkungen. Wenn dieser Parameter auf 1 gesetzt ist, dann ist der GPIO ein Ausgang; sein Verhalten hängt dann von dem Parameter *is_opendrain* ab. Nur volle GPIO-Pins haben diesen Parameter.

6.5.13 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:

6.5.13.1 Pins

- *control-type* - (Bit, In) Schaltet zwischen Lageregelungsmodus (engl. position control mode) (0) und Geschwindigkeitsregelungsmodus (engl. velocity control mode) (1) um. Standardmäßig ist die Lageregelung (0) eingestellt.
- *counts* - (s32, Out) Rückmeldung der Position in counts (Anzahl der Schritte).
- *enable* - (Bit, In) Aktiviert Schritte am Ausgang. Wenn false, werden keine Schritte erzeugt.
- *position-cmd* - (Float, In) Zielposition der Stepperbewegung, in benutzerdefinierten Positionseinheiten.
- *position-fb* - (Float, Out) Positionsrückmeldung in benutzerdefinierten Positionseinheiten (counts / position_scale).
- *velocity-cmd* - (Float, In) Zielgeschwindigkeit der Schrittmotorbewegung, in benutzerdefinierten Positionseinheiten pro Sekunde. Dieser Pin wird nur verwendet, wenn sich der Stepgen im Geschwindigkeitsregelungsmodus befindet (control-type=1).
- *velocity-fb* - (Float, Out) Rückmeldung der Geschwindigkeit in benutzerdefinierten Positionseinheiten pro Sekunde.

6.5.13.2 Parameter

- *dirhold* - (u32, RW) Mindestdauer eines stabilen Richtungssignals nach dem Ende eines Schritts, in Nanosekunden.
- *dirsetup* - (u32, RW) Mindestdauer des stabilen Richtungssignals vor Beginn eines Schritts, in Nanosekunden.
- *maxaccel* - (Float, RW) Maximale Beschleunigung, in Positionseinheiten pro Sekunde pro Sekunde. Bei einem Wert von 0 begrenzt der Treiber seine Beschleunigung nicht.
- *maxvel* - (Float, RW) Maximale Geschwindigkeit, in Positionseinheiten pro Sekunde. Wird dieser Wert auf 0 gesetzt, wählt der Treiber die Höchstgeschwindigkeit auf der Grundlage der Werte von steplen und stepspace (zu dem Zeitpunkt, zu dem maxvel auf 0 gesetzt wurde).
- *position-scale* - (Float, RW) Konvertiert von Zählungen in Positionseinheiten. position = counts / position_scale
- *step_type* - (u32, RW) Ausgabeformat, wie das step_type modparam für die Software stegen(9) Komponente. 0 = Schritt/Dir, 1 = Auf/Ab, 2 = Quadratur. Im Quadraturmodus (step_type=2) gibt der stepgen einen kompletten Gray-Zyklus (00 -> 01 -> 11 -> 10 -> 00) für jeden *Schritt* aus, den er macht.
- *steplen* - (u32, RW) Dauer des Schrittsignals, in Nanosekunden.
- *stepspace* - (u32, RW) Minimaler Abstand zwischen Schrittsignalen, in Nanosekunden.

6.5.13.3 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* - (Bit, RW) Dieser Parameter hat nur eine Auswirkung, wenn der Parameter *is_output* wahr ist. Wenn dieser Parameter wahr ist, dann ist der Ausgangswert des GPIO der umgekehrte Wert des Wertes am *out* HAL-Pin.
- *is_opendrain* - (Bit, RW) Wenn dieser Parameter falsch ist, verhält sich der GPIO wie ein normaler Ausgangspin: der I/O-Pin am Anschluss wird auf den durch den *out* HAL-Pin spezifizierten Wert gesteuert (möglicherweise invertiert). Wenn dieser Parameter true ist, verhält sich der GPIO wie ein Open-Drain-Pin. Das Schreiben von 0 an den *out* HAL-Pin treibt den I/O-Pin auf low, das Schreiben von 1 an den *out* HAL-Pin versetzt den I/O-Pin in einen hochohmigen Zustand. In diesem hochohmigen Zustand schwebt der I/O-Pin (schwach hochgezogen), und andere Geräte können den Wert treiben; der resultierende Wert am I/O-Pin ist an den *in* und *in_not* Pins verfügbar. Nur vollständige GPIO-Pins und I/O-Pins, die von aktiven Modulinstanzen als Ausgänge verwendet werden, haben diesen Parameter.

6.5.14 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:

6.5.14.1 Pins

- *enable* - (Bit, In) Wenn true, setzt das pwmgen seinen Not-Enable-Pin auf false und gibt seine Impulse aus. Wenn *enable* falsch ist, setzt pwmgen seinen Not-Enable-Pin auf true und gibt keine Signale aus.
- *value* - (Float, In) Der aktuelle Wert des pwmgen-Befehls, in beliebigen Einheiten.

6.5.14.2 Parameter

- *output-type* - (s32, RW) Dies emuliert das output_type-Ladezeit-Argument für die Softwarekomponente pwmgen. Dieser Parameter kann zur Laufzeit geändert werden, aber in den meisten Fällen wollen Sie ihn beim Start setzen und dann in Ruhe lassen. Akzeptierte Werte sind 1 (PWM auf Out0 und Direction auf Out1), 2 (Up auf Out0 und Down auf Out1), 3 (PDM-Modus, PDM auf Out0 und Dir auf Out1) und 4 (Direction auf Out0 und PWM auf Out1, *for locked antiphase*).
- *scale* - (Float, RW) Skalierungsfaktor zur Umrechnung von *value* von beliebigen Einheiten in das Tastverhältnis: dc = value / scale. Das Tastverhältnis hat einen effektiven Bereich von -1,0 bis einschließlich +1,0, alles außerhalb dieses Bereichs wird abgeschnitten.

- *pdm_frequency* - (u32, 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, it will be clipped to the max supported frequency of the board.
- *pwm_frequency* - (u32, 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, 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.

6.5.14.3 Ausgangsparameter

The output pins of each PWMGen have two additional parameters. To find which I/O pin belongs to which output run `dmesg` as described above.

- `invert_output` - (Bit, 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` - (Bit, 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.

6.5.15 Encoder

Encoders have names like `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.`. `Instance` is a two-digit number that corresponds to the HostMot2 encoder instance number. There are *num_encoders* instances, starting with 00.

Each encoder uses three or four input I/O pins, depending on how the firmware was compiled. Three-pin encoders use A, B, and Index (sometimes also known as Z). Four-pin encoders use A, B, Index, and Index-mask.

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:

6.5.15.1 Pins

- `count` - (s32, Out) Number of encoder counts since the previous reset.

- `index-enable` - (Bit, 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` - (Float, Out) Encoder position in position units (count / scale).
- `rawcounts` - (s32, Out) Total number of encoder counts since the start, not adjusted for index or reset.
- `reset` - (Bit, 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` - (Float, Out) Estimated encoder velocity in position units per second.

6.5.15.2 Parameter

- `counter-mode` - (Bit, 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` - (Bit, 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` - (Bit, 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` - (Bit, 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` - (Bit, 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` - (Float, 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` - (Float, 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.

6.5.16 5I25 Configuration

6.5.16.1 Firmware

The 5I25 firmware comes preloaded for the daughter card it is purchased with. So the `firmware=xxx.BIT` is not part of the hm2_pci configuration string when using a 5I25.

6.5.16.2 Konfiguration

Example configurations of the 5I25/7I76 and 5I25/7I77 cards are included in the Configuration Selector.

Wenn Sie Ihre eigene Konfiguration erstellen möchten, zeigen die folgenden Beispiele, wie Sie die Treiber in die HAL-Datei laden.

5I25 + 7I76 Card

```
# 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 Card

```
# 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”
```

6.5.16.3 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 des Links:../man/man9/sserial.9.html[SSERIAL(9)].

6.5.16.4 7I77 Limits

Minlimit und Maxlimit sind Begrenzungen des Pin-Wertes (in diesem Fall des Analogausgangswertes), Fullscalemax ist der Skalierungsfaktor.

These are by default set to the analog in or analog range (most likely in Volts).

So for example on the 7I77 +-10 V analog outputs, the default values are:

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

If you wanted to scale the analog out of a channel to RPM for a 0 to 6000 RPM spindle with 0-10 V control you could set the limits like this:

minlimit: 0
maxlimit: 6000
maxfullscale: 6000
(dies würde verhindern, dass unerwünschte negative Ausgangsspannungen eingestellt werden)

6.5.17 Beispielkonfigurationen

Several example configurations for Mesa hardware are included with LinuxCNC. The configurations are located in the hm2-servo and hm2-stepper sections of the Configuration Selector. Typically you will need the board installed for the configuration you pick to load. The examples are a good place to start and will save you time. Just pick the proper example from the LinuxCNC Configuration Selector and save a copy to your computer so you can edit it. To see the exact pins and parameters that your configuration gave you, open the Show HAL Configuration window from the Machine menu, or do `dmesg` as outlined above.

6.6 MB2HAL

6.6.1 Einführung

MB2HAL ist eine generische Userspace-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 <cha:classicladder,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.

6.6.2 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-Benutzerspace-Komponente

b. Standard-Komponentenname: `loadusr -W mb2hal config=config_file.ini`

c. Benutzerdefinierter Komponentenname: `loadusr -Wn mymodule mb2hal config=config_file.ini`

6.6.3 Optionen

6.6.3.1 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)
`HAL_MODULE_NAME`	Zeichenfolge (engl. string)	Nein	Name des HAL-Moduls (Komponente). Standardmäßig steht 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.

6.6.3.2 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`.
`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`.

Wert	Typ	Erforderlich	Beschreibung
SERIAL_DELAY_MS	Integer	If LINK_TYPE=serial	Serial port delay 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.
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 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.

6.6.3.3 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

6.6.4 Beispiel Konfigurationsdatei

Klicken Sie zum Herunterladen auf den Link:mb2hal_HOWTO.ini[hier].

```
#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: 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.
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)
#    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)
#    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 repond (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
```

6.6.5 Pins

Anmerkung
Yellow = Neu in LinuxCNC 2.9

6.6.5.1 fnct_01_read_coils

- mb2hal.m.n.bit *bit out*
- mb2hal.m.n.bit-inv *bit out*

6.6.5.2 fnct_02_read_discrete_inputs

- mb2hal.m.n.bit *bit out*
- mb2hal.m.n.bit-inv *bit out*

6.6.5.3 fnct_03_read_holding_registers

- mb2hal.m.n.float *float out*
- mb2hal.m.n.int *s32 out*

6.6.5.4 fnct_04_read_input_registers

- mb2hal.m.n.float *float out*
- mb2hal.m.n.int *s32 out*

6.6.5.5 fnct_05_write_single_coil

- mb2hal.m.n.bit *bit in*

`NELEMENTS` muss 1 sein oder `PIN_NAMES` darf nur einen Namen enthalten.

6.6.5.6 fnct_06_write_single_register

- mb2hal.m.n.float *float in*
- mb2hal.m.n.int *s32 in*

NELEMENTS needs to be 1 or PIN_NAMES must contain just one name. Both pin values are added and limited to 65535 (UINT16_MAX). Use one and let the other open (read as 0).

6.6.5.7 fnct_15_write_multiple_coils

- mb2hal.m.n.bit *bit in*

6.6.5.8 fnct_16_write_multiple_registers

- mb2hal.m.n.float *float in*
- mb2hal.m.n.int *s32 in*

Beide Pin-Werte werden addiert und auf 65535 (UINT16_MAX) begrenzt. Verwenden Sie einen und lassen Sie den anderen offen (gelesen als 0).

m = HAL_TX_NAME oder Transaktionsnummer, falls nicht gesetzt, n = Elementnummer (NELEMENTS) oder Name aus PIN_NAMES

Beispiel:

- mb2hal.00.01.<type> (Transaktion=00, zweites Register=01 (00 ist das erste))
- mb2hal.TxName.01.<type> (HAL_TX_NAME=TxName, zweites Register=01 (00 ist das erste))

6.7 Mitsub VFD-Treiber

Dies ist ein Benutzerraum-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.

Anmerkung
Da es sich um ein Userspace-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 Estop-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)

6.7.1 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.

Anmerkung
Das Einschalten der Fehlersuche (engl. debugging) führt zu einer Flut von Text im Terminal.

6.7.2 Pins

Dabei steht <n> für `mitsub_vfd` oder den beim Laden vergebenen Namen.

- *<n>.fwd* (bit, in) True setzt Bewegung vorwärts, False setzt Bewegung rückwärts.
- *<n>.run* (bit, in) True setzt den VFD basierend auf dem .fwd-Pin in Bewegung.
- *<n>.debug* (bit, in) Gibt Debug-Informationen auf dem Terminal aus.
- *<n>.alarm* (bit, out) signalisiert einen Alarmzustand des VFD.
- *<n>.up-to-speed* (bit, out) wenn der Antrieb die Solldrehzahl erreicht hat (die Drehzahltoleranz ist im VFD eingestellt)
- *<n>.monitor* (bit, in) einige Modelle (z.B. E500) können den Status nicht überwachen - setzen Sie den Monitor-Pin auf false in diesem Fall werden Pins wie Up-to-Speed, Ampere, Alarm und Statusbits nicht aktualisiert.
- *<n>.motor-cmd* (float, in) dem VFD befohlene Geschwindigkeit (standardmäßig in Hertz skaliert).
- *<n>.motor-fb* (float, out) Rückgemeldete Gschwindigkeit vom VFD (standardmäßig auf Hertz skaliert).
- *<n>.motor-amps* (float, out) Aktuelle Stromstärke des Motors.
- *<n>.motor-power* (float, out) Aktuelle Ausgangsleistung des Motors.
- *<n>.scale-cmd* (float, in) Skaliert den Motor-Cmd-Pin auf beliebige Einheiten. Voreinstellung 1 = Hertz.
- *<n>.scale-fb* (float, in) Skaliert den Motor-fb-Pin auf beliebige Einheiten. Voreinstellung 1 = Hertz.
- *<n>.scale-amps* (float, in) Skaliert den Motor-Ampère-Pin auf beliebige Einheiten. Voreinstellung 1 = Ampere.
- *<n>.scale-power* (float, in) Skaliert den Motor-Leistungs-Pin auf beliebige Einheiten. default 1 = .
- *<n>.estop* (bit, in) versetzt den VFD in den Notaus-Status.
- *<n>.status-bit-N* (bit, out) N = 0 bis 7, Statusbits sind auf dem VFD vom Benutzer konfigurierbar. Bit 3 sollte auf ”Geschwindigkeit erreicht” (engl. at speed) und bit 7 auf ”Alarm” gesetzt werden. Andere können nach Bedarf gesetzt werden.

6.7.3 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
```

6.7.4 Konfigurieren des Mitsubishi VFD für die serielle Nutzung

6.7.4.1 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.

6.7.4.2 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.*

6.8 Motenc Treiber

Vital Systems Moenc-100 und Moenc-LITE

Die Vital Systems Motenc-100 und Motenc-LITE sind 8- und 4-Kanal-Servokontrollkarten. Die Motenc-100 bietet 8 Quadratur-Encoder-Zähler, 8 analoge Eingänge, 8 analoge Ausgänge, 64 (68?) digitale Eingänge und 32 digitale Ausgänge. Die Motenc-LITE hat nur 4 Encoderzähler, 32 digitale Eingänge und 16 digitale Ausgänge, aber immer noch 8 analoge Eingänge und 8 analoge Ausgänge. Der Treiber identifiziert automatisch die installierte Karte und exportiert die entsprechenden HAL-Objekte.

Installation:

```
loadrt hal_motenc
```

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 4 Karten in einem System verwendet werden.

6.8.1 Pins

In den folgenden Pins, Parametern und Funktionen ist <board> die ID der Karte. Gemäß den Namenskonventionen sollte die erste Karte immer eine ID von Null haben. Dieser Treiber setzt die ID jedoch auf der Grundlage eines Jumperpaares auf der Karte, so dass sie auch bei nur einer Karte ungleich Null sein kann.

- ”(S32) Motenc. <board>.enc-<channel>-count’ - Encoder-Position, in Zählungen.
- ”(float) motenc. <board>.enc-<channel>-position’ - Encoder-Position, in Benutzereinheiten.
- *(bit) motenc.<board>.enc-<channel>-index* - Aktueller Status des Indeximpulseingangs.
- *(bit) motenc.<board>.enc-<channel>-idx-latch* - Der Treiber setzt diesen Pin auf true, wenn er einen Indeximpuls hält (aktiviert durch latch-index). Wird durch Löschen von latch-index gelöscht.
- *(bit) motenc.<board>.enc-<channel>-latch-index* - Wenn dieser Pin true ist, setzt der Treiber den Zähler beim nächsten Indeximpuls zurück.
- *(bit) motenc.<board>.enc-<channel>-reset-count* - Wenn dieser Pin wahr ist, wird der Zähler sofort auf Null zurückgesetzt, und der Pin wird gelöscht.
- *(float) motenc.<board>.dac-<channel>-value* - Analoger Ausgangswert für DAC (in Benutzereinheiten, siehe -gain und -offset)

- *(float) motenc.<board>.adc-<channel>-value* - Vom ADC gelesener analoger Eingangswert (in Benutzereinheiten, siehe -gain und -offset)
- *(bit) motenc.<board>.in-<channel>* - Zustand des digitalen Eingangspins, siehe kanonischer Digitaleingang.
- *(bit) motenc.<board>.in-<channel>-not* - Invertierter Zustand des digitalen Eingangspins, siehe kanonischer Digitaleingang.
- *(bit) motenc.<board>.out-<channel>* - Wert, der in den digitalen Ausgang geschrieben werden soll, siehe kanonischer digitaler Ausgang.
- *(bit) motenc.<board>.estop-in* - Separater Notaus-Eingang, weitere Details erforderlich.
- *(bit) motenc.<board>.estop-in-not* - Invertierter Zustand des dedizierten Notaus-Eingangs.
- *(bit) motenc.<board>.watchdog-reset* - Bidirektional, - TRUE setzen, um Watchdog einmal zurückzusetzen, wird automatisch gelöscht.

6.8.2 Parameter

- *(float) motenc.<board>.enc-<channel>-scale* - Die Anzahl der Zählungen / Benutzereinheit (zur Umrechnung von Zählungen in Einheiten).
- *(float) motenc.<board>.dac-<channel>-offset* - Setzt den DAC-Offset.
- *(float) motenc.<board>.dac-<channel>-gain* - Setzt den DAC-Gain (engl. für Verstärkung) (Skalierung).
- *(float) motenc.<board>.dac-<channel>-offset* - Setzt den DAC-Offset.
- *(float) motenc.<board>.adc-<channel>-gain* - Setzt die ADC gain (engl. für Verstärkung) (Skalierung).
- *(bit) motenc.<board>.out-<channel>-invert* - Invertiert einen digitalen Ausgang, siehe kanonischer digitaler Ausgang.
- *(u32) motenc.<board>.watchdog-control* - Konfiguriert den Watchdog.
  Der Wert kann ein bitweises ODER der folgenden Werte sein:

Bit #	Wert	Bedeutung
0	1	Das Timeout beträgt 16 ms, wenn festgelegt, 8 ms, wenn es nicht festgelegt ist
1	2	
2	4	Watchdog ist aktiviert
3	8	
4	16	Watchdog wird automatisch durch DAC-Schreibvorgänge zurückgesetzt (die HAL dac-write Funktion)

Normalerweise sind die sinnvollen Werte 0 (Watchdog deaktiviert) oder 20 (8ms Watchdog aktiviert, durch dac-write gelöscht).

- *(u32) motenc.<board>.led-view* - Ordnet einen Teil der E/A den Onboard-LEDs zu.

6.8.3 Funktionen

- *(funct) motenc.<board>.encoder-read* - Liest alle Encoder-Zähler.
- *(funct) motenc.<board>.adc-read* - Liest die Analog-Digital-Wandler.

- *(funct) motenc.<board>.digital-in-read* - Liest digitale Eingänge.
- *(funct) motenc.<board>.dac-write* - Schreibt die Spannungen an die DACs.
- *(funct) motenc.<board>.digital-out-write* - Schreibt die digitalen Ausgänge.
- *(funct) motenc.<board>.misc-update* - Aktualisiert verschiedene Dinge.

6.9 Opto22 Treiber

PCI AC5 ADAPTER KARTE / HAL TREIBER

6.9.1 Die Adapterkarte

Dies ist eine Karte von Opto22 für die Anpassung des PCI-Ports an Solid-State-Relais-Racks wie die Standard- oder G4-Serie. Sie hat 2 Ports, die jeweils bis zu 24 Punkte steuern können, und verfügt über 4 LEDs auf der Karte. Die Ports sind mit 50-poligen Anschlüssen ausgestattet, die denen der Mesa-Karten entsprechen. Alle Relais-Racks/Breakout-Boards, die mit Mesa-Karten funktionieren, sollten mit dieser Karte funktionieren, wobei alle Encoder-Zähler, PWM usw. in Software ausgeführt werden müssen. Die AC5 hat keine *intelligente* Logik an Bord, sie ist nur ein Adapter.

Weitere Informationen finden Sie auf der Website des Herstellers:

http://www.opto22.com/site/pr_details.aspx?cid=4&item=PCI-AC5

Ich möchte Opto22 für die Freigabe von Informationen in ihrem Handbuch danken, die das Schreiben dieses Treibers erleichtert haben!

6.9.2 Der Treiber (engl. driver)

Dieser Treiber ist für die PCI AC5-Karte und funktioniert nicht mit der ISA AC5-Karte. Der HAL-Treiber ist ein echtzeitfähiges Modul. Er unterstützt 4 Karten (mehr Karten sind mit einer Änderung im Quellcode möglich). Laden Sie den Basistreiber wie folgt:

```
loadrt opto_ac5
```

Dadurch wird der Treiber geladen, der nach maximal 4 Karten sucht. Er setzt die E/A der 2 Ports jeder Karte auf eine Standardeinstellung. Die Standardkonfiguration ist für 12 Eingänge und 12 Ausgänge. Die Nummern der Pin-Namen entsprechen der Position auf dem Relais-Rack. Zum Beispiel würden die Pin-Namen für die Standard-E/A-Einstellung von Port 0 lauten:

- *opto_ac5.0.port0.in-00* - Sie würden von 00 bis 11 nummeriert werden
- *opto_ac5.0.port0.out-12* - Sie würden die Nummern 12 bis 23 tragen, Port 1 wäre analog.

6.9.3 Pins

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER] OUT bit* -
- *opto_ac5.[BOARDNUMMER].port[PORTNUMMER].in-[PINNUMMER]-not OUT bit* - Schließen Sie ein HAL-Bit-Signal an diesen Pin an, um einen I/O-Punkt von der Karte zu lesen. Die PINNUMMER steht für die Position im Relaisgestell. Z.B. PINNUMMER 0 ist Position 0 in einem Opto22-Relaisrack und wäre Pin 47 auf dem 50-poligen Stecker. Der -not-Pin ist invertiert, so dass LOW TRUE und HIGH FALSE ergibt.

- *opto_ac5.[BOARDNUMMER].port[PORTNUMMER].out-[PINNUMMER] IN bit* - Schließen Sie ein HAL-Bit-Signal an diesen Pin an, um auf einen E/A-Punkt der Karte zu schreiben. Die PINNUMMER steht für die Position im Relais-Rack, z.B. PINNUMMER 23 ist Position 23 in einem Opto22-Relais-Rack und wäre Pin 1 auf dem 50-poligen Pfostenverbinder.
- *opto_ac5.[BOARDNUMBER].led[NUMBER] OUT bit* - Schaltet eine der 4 Onboard-LEDs ein/aus. Die LEDs sind von 0 bis 3 nummeriert.

BOARDNUMBER kann 0-3 sein PORTNUMBER kann 0 oder 1 sein. Anschluss 0 liegt am nächsten an der Kartenhalterung.

6.9.4 Parameter

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER]-invert W bit* - Bei TRUE wird die Bedeutung des entsprechenden -out-Pins invertiert, so dass TRUE LOW und FALSE HIGH ergibt.

6.9.5 FUNKTIONEN

- *opto_ac5.0.digital-read* - Fügen Sie dies zu einem Thread hinzu, um alle Eingabepunkte zu lesen.
- *opto_ac5.0.digital-write* - Fügen Sie dies zu einem Thread hinzu, um alle Ausgangspunkte und LEDs zu schreiben.

Die Pin-Namen für die Standard-E/A-Einstellung von Anschluss 0 lauten zum Beispiel:

```
opto_ac5.0.port0.in-00
```

Sie würden von 00 bis 11 nummeriert werden.

```
opto_ac5.0.port0.out-12
```

Sie wären mit 12 bis 23 nummeriert, Port 1 wäre derselbe.

6.9.6 Konfigurieren von E/A-Ports (engl. I/O ports)

Um die Standardeinstellung zu ändern, laden Sie den Treiber etwa so:

```
loadrt opto_ac5 portconfig0=0xffff portconfig1=0xff0000
```

Natürlich passen Sie die Zahlen so an, dass sie mit dem gewünschten E/A (engl. I/P) übereinstimmen. Jeder Anschluss kann anders eingerichtet werden.

Auf diese Weise kann man die Nummer herausfinden: Die Konfigurationsnummer ist ein 32 Bit langer Code. Dieser teilt der Karte mit, welche E/A-Punkte Ausgang bzw. Eingang sind. Die unteren 24 Bits sind die E/A-Punkte eines Ports. Die 2 höchsten Bits sind für 2 der LEDs auf der Karte. Eine Eins in einer beliebigen Bitposition macht den E/A-Punkt zu einem Ausgang. Die beiden höchsten Bits müssen ausgegeben werden, damit die LEDs funktionieren. Der Treiber setzt die beiden höchsten Bits automatisch für Sie, wir werden nicht darüber sprechen.

Am einfachsten ist es, den Taschenrechner unter ANWENDUNGEN/ZUBEHÖR zu starten. Stellen Sie ihn auf wissenschaftlich ein (klicken Sie auf Ansicht). Stellen Sie ihn auf BINÄR (Optionsfeld Bin). Drücken Sie 1 für jeden gewünschten Ausgang und/oder Null für jeden Eingang. Denken Sie daran, dass der HAL-Pin 00 dem ganz rechten Bit entspricht. 24 Zahlen stehen für die 24 E/A-Punkte eines Ports. Für die Standardeinstellung (12 Eingänge und 12 Ausgänge) würden Sie also zwölfmal die 1 drücken (das sind die Ausgänge) und zwölfmal die 0 (das sind die Eingänge). Beachten Sie,

dass der erste E/A-Punkt das niedrigste (ganz rechte) Bit ist. (Dieses Bit entspricht dem HAL-Pin 00.) Sie sollten 24 Ziffern auf dem Bildschirm haben. Drücken Sie nun die Optionsschaltfläche HEX. Die angezeigte Zahl (fff000) ist die Konfigurationsportnummer (setzen Sie ein *0x* davor, um sie als HEX-Zahl zu kennzeichnen).

Ein anderes Beispiel: Einstellen des Ports auf 8 Ausgänge und 16 Eingänge (wie bei einer Mesa-Karte). Hier sind die 24 Bits in einer BINÄREN Zahl dargestellt. Bit 1 ist die ganz rechte Zahl:

16 Nullen für die 16 Eingänge und 8 Einsen für die 8 Ausgänge

```
000000000000000011111111
```

Das wird auf dem Taschenrechner in FF umgewandelt, so dass 0xff die Nummer ist, die beim Laden des Treibers für portconfig0 und/oder portconfig1 zu verwenden ist.

6.9.7 Pin-Nummerierung

Der HAL-Pin 00 entspricht dem Bit 1 (ganz rechts), das die Position 0 auf einem Opto22-Relaisgestell darstellt. HAL-Pin 01 entspricht Bit 2 (eine Stelle links vom rechten Rand), das die Position 1 auf einem Opto22-Relaismodul darstellt. HAL-Pin 23 entspricht Bit 24 (ganz links), das die Position 23 auf einem Opto22-Relaisrack darstellt.

HAL-Pin 00 wird an Pin 47 des 50-poligen Anschlusses jedes Ports angeschlossen. HAL-Pin 01 wird an Pin 45 des 50-poligen Anschlusses jedes Ports angeschlossen. HAL-Pin 23 wird an Pin 1 des 50-poligen Anschlusses jedes Ports angeschlossen.

Beachten Sie, dass Opto22 und Mesa entgegengesetzte Nummerierungssysteme verwenden: Opto22 Position 23 = Steckerpin 1, und die Position geht nach unten, wenn die Nummer des Steckerpins nach oben geht. Mesa Hostmot2 Position 1 = Steckerpin 1, und die Positionsnummer steigt, wenn die Nummer des Steckerpins nach oben geht.

6.10 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.

6.10.1 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.

6.10.2 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.

- *(Alle s32-Ausgaben) ppmc.<port>.encoder.<channel>.count* - Encoder-Position in Zählwerten.
- *(Alle s32-Ausgaben) ppmc.<port>.encoder.<channel>.delta* – Änderung der Zählwerte seit dem letzten Lesen, in rohen Encoder-Zähleinheiten.
- *(Alles Float-Ausgaben) 'ppmc.<port>.encoder.<channel>.velocity* - Geschwindigkeit skaliert in Benutzereinheiten pro Sekunde. Auf PPMC und USC wird dies von den rohen Encoder-Zählungen pro Servoperiode abgeleitet und wird daher von der Encoder-Granularität beeinflusst. Auf UPC-Platinen mit der Firmware 8/21/09 und später kann die Geschwindigkeitsschätzung durch Zeitstempelung der Encoderzählungen verwendet werden, um die Glattheit dieser Geschwindigkeitsausgabe zu verbessern. Dies kann in die PID-HAL-Komponente eingespeist werden, um eine stabilere Servoreaktion zu erzeugen. Diese Funktion muss in der HAL-Kommandozeile, die den PPMC-Treiber startet, mit der Option timestamp=0x00 aktiviert werden.
- *(Alle Float-Ausgänge) ppmc.<port>.encoder.<channel>.position* - Encoder-Position, in Benutzereinheiten.
- *(All bit bidir) ppmc.<port>.encoder.<channel>.index-enable* - Verbindung mit joint.#.index-enable für home-to-index. Dies ist ein bidirektionales HAL-Signal. Wird es auf true gesetzt, setzt die Encoder-Hardware den Zählerstand beim nächsten Encoder-Index-Impuls auf Null zurück. Der Treiber erkennt dies und setzt das Signal zurück auf false.
- *(PPMC float output) ppmc. <port>. DAC. <channel>.value* - sendet einen vorzeichenbehafteten Wert an den 16-Bit-Digital-Analog-Wandler auf der PPMC-DAC16-Platine, der die analoge Ausgangsspannung dieses DAC-Kanals befiehlt.
- *(UPC bit input) ppmc.<port>.pwm.<channel>.enable* - Aktiviert einen PWM-Generator.
- *(UPC float input) ppmc.<port>.pwm.<channel>.value* - Wert, der das Tastverhältnis der PWM-Wellenformen bestimmt. Der Wert wird durch *pwm.<channel>.scale* geteilt, und wenn das Ergebnis 0,6 ist, beträgt das Tastverhältnis 60 %, und so weiter. Negative Werte führen dazu, dass das Tastverhältnis auf dem absoluten Wert basiert, und der Richtungspin ist so eingestellt, dass er negativ anzeigt.
- *(USC-Bit-Eingang) ppmc.<port>.stepgen.<channel>.enable* - Aktiviert einen Schrittimpulsgenerator.
- *(USC float input) ppmc.<port>.stepgen.<channel>.velocity* - Wert, der die Schrittfrequenz bestimmt. Der Wert wird mit *stepgen.<channel>.scale* multipliziert, und das Ergebnis ist die Frequenz in Schritten pro Sekunde. Negative Werte führen dazu, dass die Frequenz auf dem absoluten Wert basiert, und der Richtungspin wird auf negativ gesetzt.
- *(All bit output) ppmc.<port>.din.<channel>.in* - Zustand des digitalen Eingangspins, siehe kanonischer Digitaleingang.
- *(All bit output) ppmc.<port>.din.<channel>.in-not* - Invertierter Zustand des digitalen Eingangspins, siehe kanonischer Digitaleingang.
- *(All bit input) ppmc.<port>.dout.<channel>.out* - Wert, der an den Digitalausgang geschrieben werden soll, siehe kanonischer Digitalausgang.
- *(Option Float-Eingang) ppmc.<port>.DAC8-<channel>.value* - In den Analogausgang zu schreibender Wert, Bereich von 0 bis 255. Dies sendet 8 Ausgangsbits an J8, an dem eine Spindel-DAC-Karte angeschlossen sein sollte. 0 entspricht null Volt, 255 entspricht 10 Volt. Die Polarität des Ausgangs kann auf immer Minus, immer Plus eingestellt werden oder durch den Zustand von SSR1 (Plus,

wenn eingeschaltet) und SSR2 (Minus, wenn eingeschaltet) gesteuert werden. Sie müssen extradac = 0x00 in der HAL-Befehlszeile angeben, die den PPMC-Treiber lädt, um diese Funktion auf der ersten USC ur UPC-Platine zu aktivieren.

- *(Option bit input) ppmc.<port>.dout.<channel>.out* - Wert, der an einen der 8 zusätzlichen digitalen Ausgangspins an J8 geschrieben werden soll. Sie müssen extradout = 0x00 in der HAL-Befehlszeile angeben, die den ppmc-Treiber lädt, um diese Funktion auf der ersten USC- oder UPC-Platine zu aktivieren. extradac und extradout schließen sich gegenseitig aus, da sie dieselben Signalleitungen für unterschiedliche Zwecke verwenden. Diese Ausgangspins werden nach den digitalen Standardausgängen der Karte aufgezählt.

6.10.3 Parameter

- *(All float) ppmc.<port>.encoder.<channel>.scale* - Die Anzahl der Zählungen / Benutzereinheit (zur Umrechnung von Zählungen in Einheiten).
- *(UPC float) ppmc.<port>.pwm.<channel-range>.freq* - Die PWM-Trägerfrequenz, in Hz. Gilt für eine Gruppe von vier aufeinanderfolgenden PWM-Generatoren, wie durch *<channel-range>* angegeben. Minimum ist 610Hz, Maximum ist 500 kHz.
- *(PPMC float) ppmc.<port>.DAC.<channel>.scale* - Legt die Skalierung des DAC16-Ausgangskanals so fest, dass ein Ausgangswert, der dem Wert 1/scale entspricht, einen Ausgang mit dem Wert + oder - Volt erzeugt. Wenn also der Skalierungsparameter 0,1 ist und Sie einen Wert von 0,5 senden, wird der Ausgang 5,0 Volt betragen.
- *(UPC float) ppmc.<port>.pwm.<channel>.scale* - Skalierung für PWM Generator. Wenn *scale* X ist, dann ist das Tastverhältnis 100%, wenn der *value* Pin X (oder -X) ist.
- *(UPC float) ppmc.<port>.pwm.<channel>.max-dc* - Maximales Tastverhältnis, von 0,0 bis 1,0.
- *(UPC float) ppmc.<port>.pwm.<channel>.min-dc* – Mindestarbeitszyklus von 0,0 bis 1,0.
- *(UPC float) ppmc.<port>.pwm.<channel>.duty-cycle* - Tatsächlicher Arbeitszyklus (wird hauptsächlich zur Fehlerbehebung verwendet.)
- *(UPC bit) ppmc.<port>.pwm.<channel>.bootstrap* - Wenn true, erzeugt der PWM-Generator eine kurze Sequenz von Impulsen beider Polaritäten, wenn Not-Aus falsch ist, um die Abschalt-Latches einiger PWM-Servoantriebe zurückzusetzen.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.setup-time* - Legt die Mindestzeit zwischen Richtungswechsel und Schrittimpuls fest, in Einheiten von 100 ns. Gilt für eine Gruppe von vier aufeinanderfolgenden Schrittgeneratoren, wie durch *<Kanalbereich>* angegeben. Es können Werte zwischen 200 ns und 25,5 µs angegeben werden.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-width* - Bestimmt die Breite der Schrittimpulse in Einheiten von 100 ns. Gilt für eine Gruppe von vier aufeinanderfolgenden Schrittgeneratoren, wie durch *<Kanalbereich>* angegeben. Es können Werte zwischen 200 ns und 25,5 µs angegeben werden.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-space-min* - Legt die Mindestzeit zwischen den Impulsen in Einheiten von 100 ns fest. Gilt für eine Gruppe von vier aufeinanderfolgenden Schrittgeneratoren, wie durch *<Kanalbereich>* angegeben. Es können Werte zwischen 200 ns und 25,5 µs angegeben werden. Die maximale Schrittfrequenz beträgt: $\frac{1}{100\text{ns} * (pulsewidth + pulsespacemin)}$
- *(USC float) ppmc.<port>.stepgen.<channel>.scale* - Skalierung für Schrittimpulsgenerator. Die Schrittfrequenz in Hz ist der absolute Wert von *velocity* (engl. für Geschwindkeit) * *scale* (engl. für Skala/Maßstab).

- *(USC float) ppmc.<port>.stepgen.<channel>.max-vel* - Der Höchstwert für *velocity*. Befehle, die größer als *max-vel* sind, werden gekappt. Gilt auch für negative Werte. (Der absolute Wert wird gekappt.)
- *(USC float) ppmc.<port>.stepgen.<channel>.frequency* - Tatsächliche Schrittimpulsfrequenz in Hz (wird meist zur Fehlersuche verwendet.)
- *(Option float) ppmc. <port>. DAC8. <channel>.scale* - Legt die Skalierung des zusätzlichen DAC-Ausgangs so fest, dass ein der Skalierung entsprechender Ausgangswert eine Größe von 10,0 V ergibt. (Das Vorzeichen des Ausgangs wird durch Jumper und/oder andere digitale Ausgänge gesetzt.)
- *(Optionsbit) ppmc.<Port>.dout.<Kanal>.invert* - Invertiert einen digitalen Ausgang, siehe kanonischer digitaler Ausgang.
- *(Optionsbit) ppmc. .<port>dout. <channel>.invert* - Invertiert einen digitalen Ausgangspin von J8, siehe Kanonischer Digitalausgang.

6.10.4 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.

6.11 Pluto P-Treiber

6.11.1 Allgemeine Informationen

Das Pluto-P ist ein FPGA-Board mit dem ACEX1K-Chip von Altera.

6.11.1.1 Anforderungen

1. Ein Pluto-P-Board
2. Ein EPP-kompatibler Parallelport, der im System-BIOS für den EPP-Modus konfiguriert ist, oder eine PCI-EPP-kompatible Parallelport-Karte.

Anmerkung
Das Pluto P Board benötigt den EPP-Modus. Netmos98xx-Chips funktionieren nicht im EPP-Modus. Die Pluto P Karte funktioniert auf einigen Computern und auf anderen nicht. Es gibt kein bekanntes Muster, welche Computer funktionieren und welche nicht.

Weitere Informationen über PCI EPP kompatible Parallelport-Karten finden Sie auf der LinuxCNC Supported Hardware Seite im Wiki.

6.11.1.2 Verbinder

- Bei der Auslieferung der Pluto-P-Platine ist der linke Steckverbinder vorgelötet, wobei sich der Schlüssel in der angegebenen Position befindet. Die anderen Anschlüsse sind unbestückt. Es scheint keinen standardmäßigen 12-poligen IDC-Stecker zu geben, aber einige der Stifte eines 16-poligen Steckers können von der Platine neben QA3/QZ3 herunterhängen.
- Der untere und der rechte Anschluss befinden sich auf demselben .1”-Raster, der linke Anschluss jedoch nicht. Wenn OUT2…OUT9 nicht benötigt werden, kann ein einzelner IDC-Stecker den unteren Stecker und die unteren beiden Reihen des rechten Steckers überbrücken.

6.11.1.3 Physikalische Stifte (engl.+ inzwischen auch deutsch: pins)

- Lesen Sie das ACEX1K-Datenblatt für Informationen über Eingangs- und Ausgangsspannungsschwellenwerte. Die Pins sind alle im *LVTTL/LVCMOS* Modus konfiguriert und sind generell mit 5V TTL-Logik kompatibel.
- Vor der Konfiguration und nach dem ordnungsgemäßen Verlassen von LinuxCNC werden alle Pluto-P-Pins mit schwachen Pull-ups (20 kΩ min, 50 kΩ max) tristiert. Wenn der Watchdog-Timer aktiviert ist (Standardeinstellung), werden diese Pins auch nach einer Unterbrechung der Kommunikation zwischen LinuxCNC und der Karte tristiert. Der Watchdog-Timer benötigt etwa 6,5 ms, um aktiviert zu werden. Allerdings können Software-Fehler in der pluto_servo-Firmware oder LinuxCNC die Pluto-P-Pins in einem undefinierten Zustand lassen.
- Im Modus PWM+RIchtung (engl. pwm+dir) ist Richtung (dir) standardmäßig HIGH für negative Werte und LOW für positive Werte. Um HIGH für positive Werte und LOW für negative Werte zu wählen, setzen Sie den entsprechenden Parameter dout-NN-invert auf TRUE, um das Signal zu invertieren.
- Der Indexeingang wird mit der steigenden Flanke getriggert. Erste Tests haben gezeigt, dass die QZx-Eingänge besonders rauschempfindlich sind, da sie alle 25 ns abgefragt werden. Es wurde eine digitale Filterung hinzugefügt, um Impulse zu filtern, die kürzer als 175 ns (sieben Abfragezeiten) sind. Eine zusätzliche externe Filterung an allen Eingangspins, wie z. B. ein Schmitt-Puffer oder Inverter, ein RC-Filter oder ein Differentialempfänger (falls zutreffend) wird empfohlen.
- Die Pins IN1…IN7 haben 22 Ω Vorwiderstände zu den entsprechenden FPGA-Pins. Keine anderen Pins haben irgendeine Art von Schutz für Spannungen oder Ströme außerhalb der Spezifikation. Es ist Sache des Integrators, eine geeignete Isolierung und einen entsprechenden Schutz hinzuzufügen. Herkömmliche Optoisolator-Karten mit parallelem Anschluss funktionieren aufgrund der bidirektionalen Natur des EPP-Protokolls nicht mit pluto_servo.

6.11.1.4 LED

- Wenn das Gerät unprogrammiert ist, leuchtet die LED schwach. Wenn das Gerät programmiert ist, leuchtet die LED entsprechend dem Tastverhältnis von PWM0 (*LED = UP0 xor DOWN0*) oder STEPGEN0 (*LED = STEP0 xor DIR0*).

6.11.1.5 Power

- Von VCC kann eine geringe Strommenge entnommen werden. Der verfügbare Strom hängt von der ungeregelten DC-Eingabe auf der Platine ab. Alternativ können dem FPGA über diese VCC-Pins regulierte +3,3 VDC zugeführt werden. Der erforderliche Strom ist noch nicht bekannt, liegt aber wahrscheinlich bei etwa 50 mA plus I/O-Strom.
- Der Regler auf der Pluto-P-Platine ist ein Low-Dropout-Typ. Wenn 5 V an der Netzbuchse anliegen, kann der Regler ordnungsgemäß arbeiten.

6.11.1.6 PC-Schnittstelle

- Es wird nur eine einzige pluto_servo oder pluto_step Karte unterstützt.

6.11.1.7 Neuerstellung der FPGA-Firmware

Die Unterverzeichnisse ”src/hal/drivers/pluto_servo_firmware/” und ”src/hal/drivers/pluto_step_firmware/” enthalten den Verilog-Quellcode sowie zusätzliche Dateien, die von Quartus für die FPGA-Firmware verwendet werden. Die Quartus II Software von Altera ist erforderlich, um die FPGA-Firmware neu zu erstellen. Um die Firmware aus den .hdl und anderen Quelldateien neu zu erstellen, öffnen Sie die *.qpf* Datei und drücken Sie CTRL-L. Dann kompilieren Sie LinuxCNC neu.

Wie der HAL-Hardwaretreiber unterliegt auch die FPGA-Firmware den Bedingungen der GNU General Public License.

Die kostenlose Version von Quartus II läuft nur unter Microsoft Windows, obwohl es offenbar eine kostenpflichtige Version gibt, die unter Linux läuft.

6.11.1.8 Für weitere Informationen

Einige zusätzliche Informationen dazu finden Sie unter KNJC LLC und unter Blog des Entwicklers.

6.11.2 Pluto-Servo

Das pluto_servo-System eignet sich für die Steuerung einer 4-Achsen-CNC-Fräse mit Servomotoren, einer 3-Achsen-Fräse mit PWM-Spindelsteuerung, einer Drehmaschine mit Spindel-Encoder, etc. Die große Anzahl von Eingängen ermöglicht einen vollständigen Satz von Endschaltern.

Dieser Treiber hat folgende Eigenschaften:

- 4 Quadraturkanäle mit 40 MHz Abtastrate. Die Zähler arbeiten im *4x* Modus. Die maximale nützliche Quadratur-Rate ist 8191 Zählungen pro LinuxCNC Servo-Zyklus, oder etwa 8 MHz für LinuxCNC Standard 1 ms Servo-Rate.
- 4 PWM-Kanäle, *up/down* oder *pwm+dir* Stil. 4095 Arbeitszyklen von -100% bis +100%, einschließlich 0%. Die PWM-Periode beträgt etwa 19,5 kHz (40 MHz / 2047). Ein PDM-ähnlicher Modus ist ebenfalls verfügbar.
- 18 digitale Ausgänge: 10 dedizierte, 8 gemeinsam genutzte mit PWM-Funktionen. (Beispiel: Eine Drehmaschine mit unidirektionaler PWM-Spindelsteuerung kann insgesamt 13 digitale Ausgänge verwenden)
- 20 digitale Eingänge: 8 dedizierte, 12 gemeinsam genutzte mit Quadraturfunktionen. (Beispiel: Eine Drehmaschine mit Indeximpuls nur an der Spindel kann insgesamt 13 digitale Eingänge verwenden.)
- EPP-Kommunikation mit dem PC. Die EPP-Kommunikation dauert bei den bisher getesteten Maschinen typischerweise etwa 100 µs und ermöglicht Servoraten über 1 kHz.

6.11.2.1 Pinbelegung

- *UPx* - Das *Up*- (Aufwärts-/Abwärtsmodus) oder *PWM*-Signal (PWM+Richtung-Modus) vom PWM-Generator X. Kann als digitaler Ausgang verwendet werden, wenn der entsprechende PWM-Kanal unbenutzt ist oder der Ausgang des Kanals immer negativ ist. Der entsprechende digitale Ausgang kann auf TRUE gesetzt werden, damit UPx aktiv low statt aktiv high ist.

- *DNx* - Das *down*- (Aufwärts/Abwärts-Modus) oder *Richtungs*-Signal (PWM+direction-Modus) vom PWM-Generator X. Kann als digitaler Ausgang verwendet werden, wenn der entsprechende PWM-Kanal unbenutzt ist oder der Ausgang des Kanals nie negativ ist. Der entsprechende digitale Ausgang kann auf TRUE gesetzt werden, damit DNx aktiv low statt aktiv high ist.
- *QAx, QBx* - Die A- und B-Signale für Quadraturzähler X. Kann als digitaler Eingang verwendet werden, wenn der entsprechende Quadraturkanal nicht verwendet wird.
- *QZx* - Das Z-Signal (Index) für Quadraturzähler X. Kann als digitaler Eingang verwendet werden, wenn die Indexfunktion des entsprechenden Quadraturkanals nicht verwendet wird.
- *INx* - Dedizierter digitaler Eingang#x
- *OUTx* - Dedizierter digitaler Ausgang #x
- *GND* - Masse (engl. ground)
- VCC' - +3,3V geregelter Gleichstrom (engl. regulated DC)

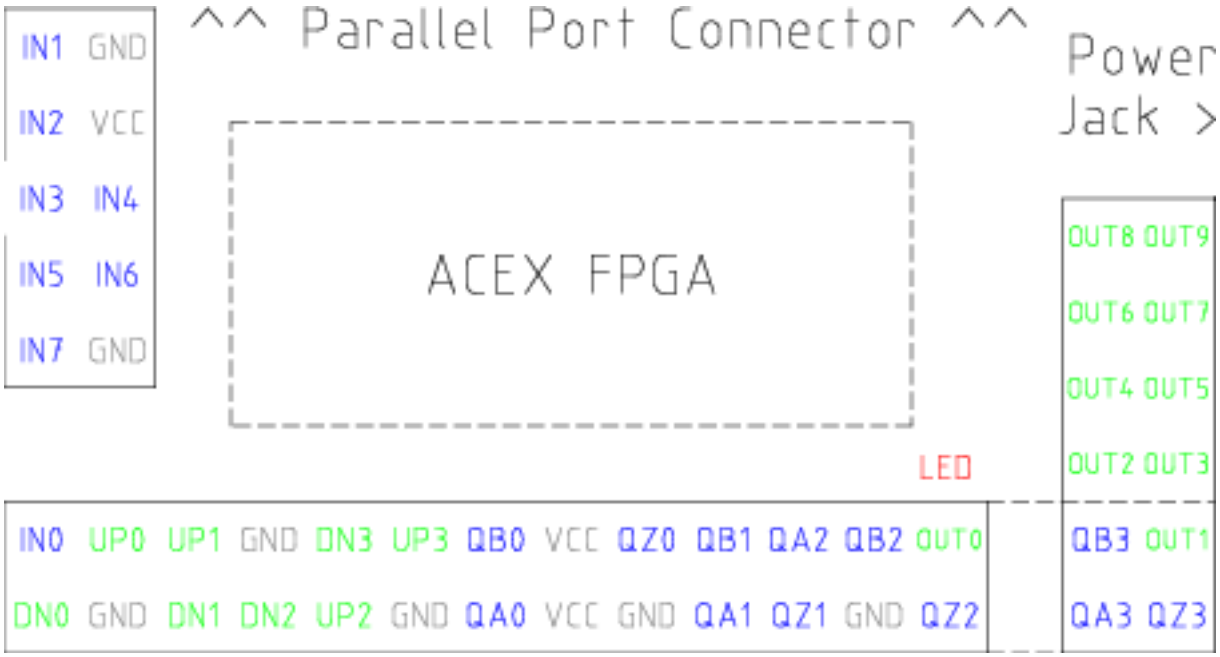


Abbildung 6.10: Pluto-Servo-Pinbelegung

Tabelle 6.27: Pluto-Servo Wechselnde Pin-Funktionen

Primäre Funktion	**Alternative Funktion**	**Verhalten bei Verwendung beider Funktionen**
UP0	PWM0	Wenn pwm-0-pwmdir TRUE ist, dann ist dieser Pin der PWM-Ausgang
	OUT10	geXORt (engl. XOR'd) mit UP0 oder PWM0
UP1	PWM1	Wenn pwm-1-pwmdir TRUE ist, dann ist dieser Pin der PWM-Ausgang
	OUT12	geXORt mit UP1 oder PWM1

Tabelle 6.27: (continued)

Primäre Funktion	**Alternative Funktion**	**Verhalten bei Verwendung beider Funktionen**
UP2	PWM2	Wenn pwm-2-pwmdir TRUE ist, dann ist dieser Pin der PWM-Ausgang
	OUT14	geXORt mit UP2 oder PWM2
UP3	PWM3	Wenn pwm-3-pwmdir TRUE ist, dann ist dieser Pin der PWM-Ausgang
	OUT16	geXORt mit UP3 oder PWM3
DN0	DIR0	Wenn pwm-0-pwmdir TRUE ist, dann ist dieser Pin der DIR-Ausgang
	OUT11	geXORt mit DN0 oder DIR0
DN1	DIR1	Wenn pwm-1-pwmdir TRUE ist, dann ist dieser Pin der DIR-Ausgang
	OUT13	geXORt mit DN1 oder DIR1
DN2	DIR2	Wenn pwm-2-pwmdir TRUE ist, dann ist dieser Pin der DIR-Ausgang
	OUT15	geXORt mit DN2 oder DIR2
DN3	DIR3	Wenn pwm-3-pwmdir TRUE ist, dann ist dieser Pin der DIR-Ausgang
	OUT17	geXORt mit DN3 oder DIR3
QZ0	IN8	Gleichen Wert lesen
QZ1	IN9	Gleichen Wert lesen
QZ2	IN10	Gleichen Wert lesen
QZ3	IN11	Gleichen Wert lesen
QA0	IN12	Gleichen Wert lesen
QA1	IN13	Gleichen Wert lesen
QA2	IN14	Gleichen Wert lesen
QA3	IN15	Gleichen Wert lesen
QB0	IN16	Gleichen Wert lesen
QB1	IN17	Gleichen Wert lesen
QB2	IN18	Gleichen Wert lesen
QB3	IN19	Gleichen Wert lesen

6.11.2.2 Input-Latching und Output-Aktualisierung

- Die PWM-Tastverhältnisse werden für jeden Kanal zu unterschiedlichen Zeiten aktualisiert.
- Die digitalen Ausgänge OUT0 bis OUT9 werden alle zur gleichen Zeit aktualisiert. Die digitalen Ausgänge OUT10 bis OUT17 werden gleichzeitig mit der PWM-Funktion aktualisiert, mit der sie

geteilt werden.

- Die digitalen Eingänge IN0 bis IN19 werden alle gleichzeitig gehalten (engl. latched).
- Die Quadraturpositionen werden für jeden Kanal zu unterschiedlichen Zeiten gespeichert.

6.11.2.3 HAL-Funktionen, Pins und Parameter

Eine Liste aller *loadrt*-Argumente, HAL-Funktionsnamen, Pin-Namen und Parameter-Namen befindet sich in der Manpage zu *pluto_servo.9*.

6.11.2.4 Kompatible Treiber-Hardware

Ein Schaltplan für eine 2A, 2-Achsen-PWM-Servoverstärkerplatine ist von der (der Softwareentwickler) erhältlich. Die L298 H-Bridge kann für Motoren bis zu 4A (ein Motor pro L298) oder bis zu 2A (zwei Motoren pro L298) mit einer Versorgungsspannung von bis zu 46V verwendet werden. Der L298 verfügt jedoch nicht über eine eingebaute Strombegrenzung, was bei Motoren mit hohen Stillstandsströmen ein Problem darstellt. Für höhere Ströme und Spannungen haben einige Anwender über Erfolge mit den integrierten High-Side/Low-Side-Treibern von International Rectifier berichtet.

6.11.3 Pluto Step

Pluto-step ist für die Steuerung einer 3- oder 4-Achsen-CNC-Fräse mit Schrittmotoren geeignet. Die große Anzahl von Eingängen ermöglicht einen vollständigen Satz von Endschaltern.

Die Karte hat folgende Merkmale:

- 4 *Schritt+Richtung*-Kanäle mit 312,5 kHz maximaler Schrittrate, programmierbarer Schrittlänge, Abstand und Richtungswechselzeiten
- 14 dedizierte digitale Ausgänge
- 16 dedizierte digitale Eingänge
- EPP-Kommunikation mit dem PC

6.11.3.1 Pinbelegung

- *STEPx* - Der *Step* (Clock) Ausgang des Stepgen-Kanals *x*
- *DIRx* - Die *Richtungs*-Ausgabe des Stepgen-Kanals *x*
- *INx* - Dedizierter digitaler Eingang#x
- *OUTx* - Dedizierter digitaler Ausgang #x
- *GND* - Masse (engl. ground)
- VCC’ - +3,3V geregelter Gleichstrom (engl. regulated DC)

Während der ”erweiterte Hauptanschluss” über eine Reihe von Signalen verfügt, die normalerweise auf einem Step & Direction DB25-Anschluss zu finden sind - 4 Schrittgeneratoren, 9 Eingänge und 6 Allzweckausgänge -, unterscheidet sich das Layout dieses Anschlusses von dem eines standardmäßigen 26-poligen Flachbandkabels zu einem DB25-Anschluss.

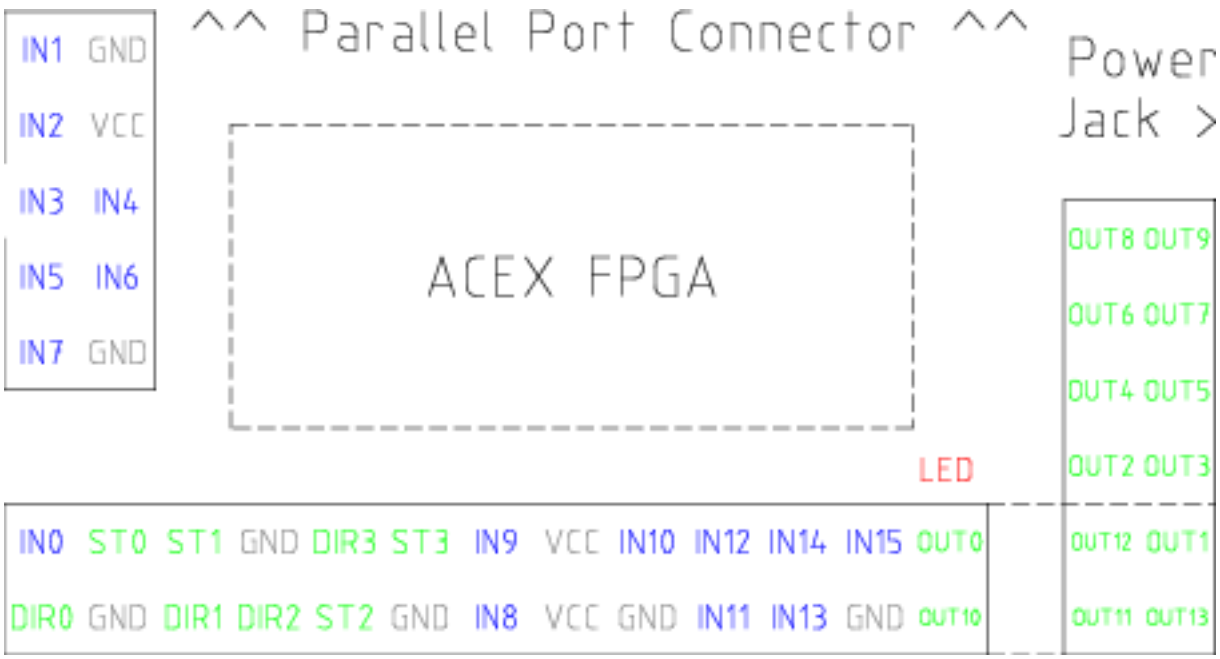


Abbildung 6.11: Pluto-Step Pinout

6.11.3.2 Input-Latching und Output-Aktualisierung

- Die Schrittfrequenzen für jeden Kanal werden zu unterschiedlichen Zeiten aktualisiert.
- Die digitalen Ausgänge werden alle gleichzeitig aktualisiert.
- Die digitalen Eingänge werden alle zur gleichen Zeit gehalten (engl. latched).
- Feedback-Positionen werden für jeden Kanal zu unterschiedlichen Zeiten gespeichert.

6.11.3.3 Schritt (engl. Step)-Wellenform-Timings

Die Firmware und der Treiber erzwingen Schrittlänge, Abstand und Richtungswechselzeiten. Die Zeiten werden auf das nächste Vielfache von 1,6 µs aufgerundet, mit einem Maximum von 49,6 µs. Die Zeitvorgaben sind dieselben wie bei der Softwarekomponente stepgen, mit der Ausnahme, dass ”dirhold” und ”dirsetup” zu einem einzigen Parameter ”dirtime” zusammengefasst wurden, der das Maximum der beiden Parameter sein sollte, und dass für alle Kanäle stets dieselben Schrittvorgaben gelten.

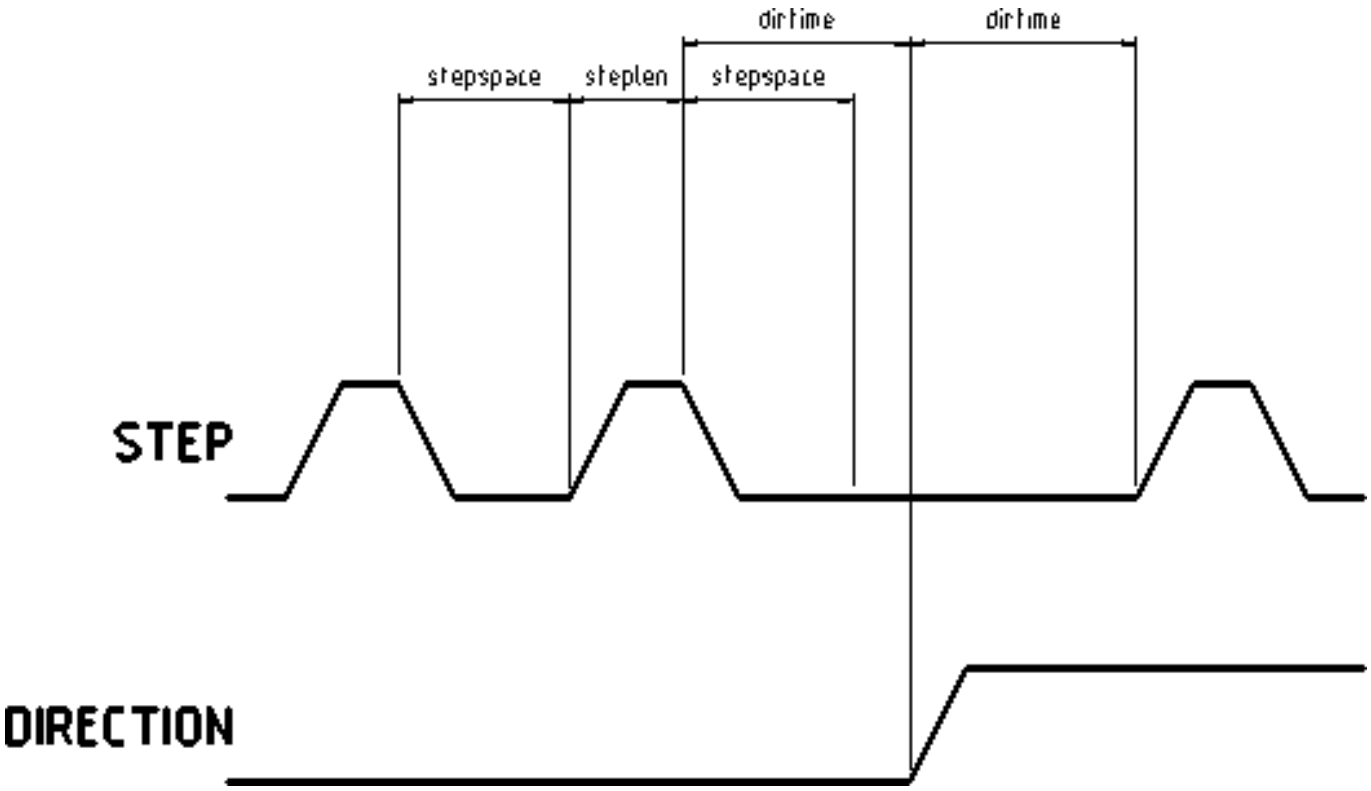


Abbildung 6.12: Pluto-Step-Timings

6.11.3.4 HAL-Funktionen, Pins und Parameter

Eine Liste aller *loadrt*-Argumente, HAL-Funktionsnamen, Pin-Namen und Parameter-Namen findet sich in der Manpage zu *pluto_step.9*.

6.12 Powermax Modbus-Treiber

Dies ist ein in Python geschriebenes HAL-Programm zur Steuerung von Hypetherm Powermax-Plasmaschnei
unter Verwendung des Modbus-ASCII-Protokolls über RS485.

Anmerkung
Da es sich um ein Userspace-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.

6.12.1 Pins

- **pmx485.mode-set** (bit, in) # Schneidmodus einstellen
- **pmx485.current-set** (bit, in) # Schneidstrom einstellen
- **pmx485.pressure-set** (bit, in) # Gasdruck einstellen
- **pmx485.enable** (bit, in) # Aktivierung der Komponente
- **pmx485.mode** (bit, out) # Schneidmodus-Feedback
- **pmx485.current** (bit, out) # Schneidstrom feedback
- **pmx485.pressure** (bit, out) # Gasdruck-Rückmeldung
- **pmx485.fault** (bit, out) # Powermax-Fehlercode
- **pmx485.status** (bit, out) # Verbindungsstatus
- **pmx485.current-min** (bit, out) # minimal zulässiger Strom
- **pmx485.current-max** (bit, out) # maximal zulässiger Strom
- **pmx485.pressure-min** (bit, out) # minimal zulässiger Gasdruck
- **pmx485.pressure-max** (bit, out) # maximal zulässiger Gasdruck

6.12.2 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** (engl. für *Modus gesetzt*)
- **current-set** (engl. für *Stromfluss gesetzt*)
- **pressure-set** (engl. für Druck-gesetzt)

Anmerkung
Ein **pressure-set**-Wert von Null ist gültig, der Powermax berechnet dann intern den erforderlichen Druck.

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:

- Setzen Sie alle Set-Pins auf Null: **mode-set**, **current-set** und **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.

6.12.3 Referenz:

- Hypertherm Anwendungshinweis #807220
  ”Powermax45 XP/65/85/105/125® Serielles Kommunikationsprotokoll”

6.13 Servo To Go-Treiber

Die Servo-To-Go (STG) ist eine der ersten PC Motion Control Karten von LinuxCNC unterstützt. Es ist eine ISA-Karte und es gibt in verschiedenen Varianten (alle von diesem Treiber unterstützt). Die Karte enthält bis zu 8 Kanäle von Quadratur-Encoder-Eingang, 8 Kanäle von analogen Ein- und Ausgängen, 32 Bit digitale E/A, ein Intervall-Timer mit Interrupt und ein Watchdog. Weitere Informationen finden Sie auf der Webseite Servo To Go.

Anmerkung
Uns liegen Berichte vor, dass die Operationsverstärker auf der Servo To Go-Karte nicht mit neueren ATX-Netzteilen funktionieren, die moderne DC-DC-Schaltwandler verwenden. Der Fehlermodus ist, dass die STG-Karte eine konstante Spannung ausgibt, unabhängig davon, was der Treiber ihr befiehlt. Ältere ATX-Netzteile mit linearen Spannungsreglern haben dieses Problem nicht und funktionieren problemlos mit den STG-Karten.

6.13.1 Installation

```
loadrt hal_stg [base=<address>] [num_chan=<nr>] [dio="<dio-string>"] \
                  [model=<model>]
```

Das Feld base address ist optional; wenn es nicht angegeben wird, versucht der Treiber, die Karte automatisch zu erkennen. Das Feld num_chan wird verwendet, um die Anzahl der auf der Karte verfügbaren Kanäle anzugeben; wird es nicht verwendet , dann wird die 8-Achsen-Version angenommen. Die Konfiguration der digitalen Ein- und Ausgänge wird durch einen Konfigurationsstring bestimmt, der beim Laden des Moduls an insmod übergeben wird. Das Format besteht aus einer vierstelligen Zeichenkette, welche die Richtung jeder Gruppe von Pins festlegt. Jedes Zeichen des Richtungsstrings ist entweder "I" oder "O". Das erste Zeichen bestimmt die Richtung von Anschluss A (Port A - DIO.0-7), das nächste die von Anschluss B (Port B - DIO.8-15), das nächste die von Anschluss C (Port C - DIO.16-23) und das vierte die von Anschluss D (Port D - DIO.24-31). Das Modellfeld kann verwendet werden, falls der Treiber nicht automatisch die richtige Kartenversion erkennt.

HINT: nach dem Starten des Treibers kann *dmesg* auf treiberrelevante Meldungen (z.B. automatisch erkannte Versionsnummer und Basisadresse) untersucht werden. Zum Beispiel:

```
loadrt hal_stg base=0x300 num_chan=4 dio="IOIO"
```

Dieses Beispiel installiert den STG-Treiber für eine Karte mit der Basisadresse 0x300, 4 Kanälen für Encoder-Feedback, DACs und ADCs sowie 32 Bits E/A, die wie folgt konfiguriert sind: die ersten 8 (Port A) als Eingang, die nächsten 8 (Port B) als Ausgang, die nächsten 8 (Port C) als Eingang und die letzten 8 (Port D) als Ausgang

```
loadrt hal_stg
```

Dieses Beispiel installiert den Treiber und versucht, die Kartenadresse und das Kartenmodell automatisch zu erkennen. Es installiert standardmäßig 8 Achsen zusammen mit einer Standard-E/A-Einstellung: Port A und B sind als Eingang, Port C und D als Ausgang konfiguriert.

6.13.2 Pins

- *stg.<channel>.counts* - (s32) Folgt den gezählten Encoder-Ticks.
- *stg.<channel>.position* - (float) Gibt eine konvertierte Position aus.
- *stg.<channel>.dac-value* - (float) Steuert die Spannung für den entsprechenden DAC.

- *stg.<channel>.adc-value* - (float) Folgt die gemessene Spannung vom entsprechenden ADC.
- *stg.in-<pinnum>* - (bit) Folgt einen physischen Eingangspin.
- *stg.in-<pinnum>-not* - (bit) Folgt einem physischen Eingangspin, aber invertiert.
- *stg.out-<pinnum>* - (Bit) Treibt einen physischen Ausgangspin an

Für jeden Pin ist <Kanal> die Achsennummer und <Pinnum> die logische Pin-Nummer des STG, wenn "IIOO" definiert ist. Es gibt 16 Eingangs-Pins (in-00 .. in-15) und 16 Ausgangs-Pins (out-00 .. out-15), und sie entsprechen den PORTs ABCD (in-00 ist PORTA.0, out-15 ist PORTD.7).

Der in-<pinnum>-HAL-Pin ist TRUE, wenn der physikalische Pin high ist, und FALSE, wenn der physikalische Pin low ist. Der in-<pinnum>-not HAL-Pin ist invertiert - er ist FALSE, wenn der physikalische Pin high ist. Durch Anschließen eines Signals an den einen oder anderen Pin kann der Benutzer den Zustand des Eingangs bestimmen.

6.13.3 Parameter

- *stg.<channel>.position-scale* - (float) Die Anzahl der Zählungen / Benutzereinheiten (zur Umrechnung von Zählungen in Einheiten).
- *stg.<channel>.dac-offset* - (float) Legt den Offset für den entsprechenden DAC fest.
- *stg.<channel>.dac-gain* - (float) Legt die Verstärkung des entsprechenden DAC fest.
- *stg.<channel>.adc-offset* - (float) Legt den Offset des entsprechenden ADC fest.
- *stg.<channel>.adc-gain* - (float) Legt die Verstärkung des entsprechenden ADC fest.
- *stg.out-<pinnum>-invert* - (bit) Invertiert einen Ausgangspin.

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 auf TRUE aktiviert und durch FALSE deaktiviert. Wenn *-invert* TRUE ist, wird der physikalische Pin durch das Setzen des HAL *-out*-Pins TRUE auf low gesetzt.

6.13.4 Funktionen

- *stg.capture-position* - Liest die Encoder-Zähler von der Achse <channel>.
- *stg.write-dacs* - Schreibt die Spannungen in die DACs.
- *stg.read-adcs* - Liest die Spannungen von den ADCs.
- *stg.di-read* - Liest physische In-Pins aller Ports und aktualisiert alle HAL In-<pinnum> und In- nicht <pinnum>Pins.
- *stg.do-write* - Liest alle HAL out-<pinnum> Pins und aktualisiert alle physischen Ausgangspins.

6.14 Shuttle

6.14.1 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.

Warnung
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 im Treiber eintrifft, verwendet der Treiber die vom Gerät gemeldete Jog-Wheel-Position, um den Zähler auf 0 zu initialisieren.
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 Jogwheel einwandfrei und bemerkt den ersten Klick.

6.14.2 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.

6.14.3 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-<ButtonNummer> **(bit out)**
Diese Pins sind wahr (1) wenn die Schaltfläche gedrückt wird.

<Prefix>.button-<ButtonNummer>-not **(bit out)**
Diese Pins haben den umgekehrten Zustand des Buttons, also sind sie True (1) wenn der Button nicht gedrückt wird.

<Präfix>.counts **(s32 out)**
Kumulierte Zählungen des Jogwheels (das innere Rad).

<Präfix>.spring-wheel-s32 **(s32 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.

<Präfix>.spring-wheel-f **(float 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.)

6.15 VFS11 VFD-Treiber

Dies ist ein User-Space-HAL-Programm zur Steuerung der S11-Serie von VFD's 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

6.15.1 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 ÞÞidof vfs11_vfd Ý`).

Anmerkung
Bei seriellen Konfigurationsfehlern kann das Einschalten von verbose zu einer Flut von Timeout-Fehlern führen.

6.15.2 Pins

Dabei steht <n> für `vfs11_vfd` oder den beim Laden mit der Option -n angegebenen Namen.

- *<n>.acceleration-pattern* (bit, in) wenn true, setzt Beschleunigungs- und Verzögerungszeiten wie in den Registern F500 bzw. F501 definiert. Wird in PID-Schleifen verwendet, um kürzere Rampenzeiten zu wählen und damit Schwingungen zu vermeiden.
- *<n>.alarm-code* (s32, out) ungleich Null, wenn der Antrieb im Alarmzustand ist. Bitmap zur Beschreibung der Alarminformationen (siehe Beschreibung des Registers FC91). Verwenden Sie err-reset (siehe unten), um den Alarm zu löschen.
- *<n>.at-speed* (bit, out) wenn der Antrieb die Solldrehzahl erreicht (siehe Drehzahltoleranz unten)
- *<n>.current-load-percentage* (float, out) gemeldet vom VFD
- *<n>.dc-brake* (bit, in) aktiviert die Gleichstrombremse. Schaltet auch das Spindel-ein Signal aus.
- *<n>.enable* (bit, in) gibt das VFD frei. Wenn false, werden alle Betriebsparameter weiterhin gelesen, aber die Steuerung wird freigegeben und die Bedienfeldsteuerung wird aktiviert (abhängig von der VFD-Einstellung).
- *<n>.err-reset* (bit, in) setzt Fehler (Alarme a.k.a Trip und e-stop Status) zurück. Das Zurücksetzen des VFD kann eine 2-Sekunden-Verzögerung verursachen bis der VFD neu gebootet ist und der Modbus wieder funktioniert.
- *<n>.estop* (bit, in) versetzt den VFD in den Notaus-Status. Kein Betrieb möglich, bis mit Err-Reset oder einen Neustart der Notaus rückgängig gemacht wird.
- *<n>.frequency-command* (float, out) aktuelle Zielfrequenz in Hz, wie durch den Drehzahl-Befehl (der in RPM angegeben ist), vom VFD eingestellt
- *<n>.frequency-out* (float, out) aktuelle Ausgangsfrequenz des VFD
- *<n>.inverter-load-percentage* (float, out) aktuelle Lastmeldung vom VFD
- *<n>.is-e-stopped* (bit, out) der VFD befindet sich im Notaus-Status (blinkendes „E“ auf dem Bedienfeld). Verwenden Sie err-reset, um das VFD neu zu starten und den Notaus-Status zu löschen.
- *<n>.is-stopped* (bit, out) true wenn der VFD 0 Hz Ausgang meldet
- *<n>.max-rpm* (Float, R) tatsächliche Drehzahlgrenze basierend auf der maximalen Frequenz, die der VFD erzeugen kann, und den Typenschildwerten des Motors. Wenn z. B. die Typenschild-HZ 50 und die Typenschild-Drehzahl 1410 beträgt, der Frequenzumrichter aber bis zu 80 Hz erzeugen kann, dann würde die maximale Drehzahl 2256 (80*1410/50) betragen. Die Frequenzgrenze wird beim Einschalten vom VFD abgelesen. Um die obere Frequenzgrenze zu erhöhen, müssen die Parameter UL und FH am Bedienfeld geändert werden. Anweisungen zum Einstellen der Höchstfrequenz finden Sie im Handbuch des VF-S11.
- *<n>.modbus-ok* (bit, out) true, wenn die Modbus-Sitzung erfolgreich aufgebaut ist und die letzten 10 Transaktionen ohne Fehler zurückgegeben wurden.
- *<n>.motor-RPM* (float, out) geschätzter aktueller U/min (engl. RPM)-Wert, vom VFD
- *<n>.output-current-percentage* (float, out) vom VFD
- *<n>.output-voltage-percentage* (float, out) vom VFD
- *<n>.output-voltage* (float, out) vom VFD
- *<n>.speed-command* (float, in) an den VFD gesendete Geschwindigkeit in U/min. Es ist ein Fehler, eine Geschwindigkeit zu senden, die höher ist als die im VFD eingestellte Motor Max RPM

- *<n>.spindle-fwd* (bit, in) 1 für FWD (engl. kurz für vorwärts) und 0 für REV (engl. kurz für rückwärts), gesendet an VFD
- *<n>.spindle-on* (bit, in) 1 für EIN und 0 für AUS an den VFD gesendet, nur bei Betrieb eingeschaltet
- *<n>.spindle-rev* (bit, in) 1 für EIN und 0 für AUS, nur bei Betrieb eingeschaltet
- *<n>.jog-mode* (bit, in) 1 für ON und 0 für OFF, aktiviert den VF-S11 *jog-mode*. Die Drehzahlregelung ist deaktiviert, und die Ausgangsfrequenz wird durch das Register F262 bestimmt (voreingestellt auf 5 Hz). Dies kann für die Spindelausrichtung nützlich sein. Im normalen Modus schaltet sich der VFD ab, wenn die Frequenz unter 12 Hz fällt.
- *<n>.status* (s32, out) Antrieb (engl. Drive)-Status des VFD (siehe TOSVERT VF-S11 Communications Function Instruction Manual, Register FD01). Eine Bitmap.
- *<n>.trip-code* (s32, out) Auslösecode, wenn VF-S11 im Auslösezustand ist.
- *<n>.error-count* (s32, out) Anzahl der Modbus-Transaktionen, die einen Fehler zurückgegeben haben
- *<n>.max-speed* (bit, in) ignoriert den Schleifenzeit (engl. loop-time)-Parameter und lässt Modbus mit maximaler Geschwindigkeit laufen, auf Kosten einer höheren CPU-Auslastung. Empfohlene Verwendung während der Spindelpositionierung.

6.15.3 Parameter

Dabei steht <n> für `vfs11_vfd` oder den beim Laden mit der Option -n angegebenen Namen.

- *<n>.frequency-limit* (float, RO) oberer Grenzwert, der vom VFD-Setup gelesen wird.
- *<n>.loop-time* (float, RW) wie oft der Modbus abgefragt wird (Standardintervall 0,1 Sekunden)
- *<n>.nameplate-HZ* (float, RW) Namen-/Typenschild in Hz des Motors (Standard 50). Dient zur Berechnung der Zielfrequenz (zusammen mit nameplate-RPM (engl. für U/min)) für einen durch den Drehzahlbefehl vorgegebenen Ziel-Drehzahlwert.
- *<n>.nameplate-RPM* (float, RW) (engl. für U/min) Namens-/Typenschild-Drehzahl des Motors (Standard 1410)
- *<n>.rpm-limit* (float, RW) weicher Grenzwert für die Motordrehzahl, der nicht überschritten werden darf (Standardwert ist die Angabe bei ”nameplate-RPM” (engl. für Typenschild-Drehzahl)).
- *<n>.tolerance* (float, RW) Drehzahltoleranz (Standardwert 0,01) zur Bestimmung, ob die Spindel auf Drehzahl ist (0,01 bedeutet: Ausgangsfrequenz liegt innerhalb von 1% der Sollfrequenz)

6.15.4 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
```

6.15.5 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
```

6.15.6 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 VFD’s finden Sie im ”TOSVERT VF-S11 Communications Function Instruction Manual” (Toshiba Dokumentnummer E6581222) und im ”TOSVERT VF-S11 Instruction manual” (Toshiba Dokumentnummer E6581158).

6.15.7 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.

6.15.8 Konfigurieren des VFS11 VFD für die Modbus-Nutzung

6.15.8.1 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".

6.15.8.2 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.

6.15.9 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.

Kapitel 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 Spindelfreigabe (engl. spindle enable)

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 Spindeldrehrichtung (engl. spindle direction)

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-Sanftanlauf (engl. 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

7.2.6.1 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
```

[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.

```
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
```

7.2.6.2 Spindel bei Drehzahl (engl. spindle at speed)

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-Schnittstellenka 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 Userspace-Komponente für die Automation Direct GS2 VFD's
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
```

Anmerkung
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-Kommunikations 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.

Kapitel 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 ist derzeit zusammen mit LinuxCNC verteilt werden. Wenn es Fragen/Probleme/Bugs bitte melden Sie sie 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 der 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:

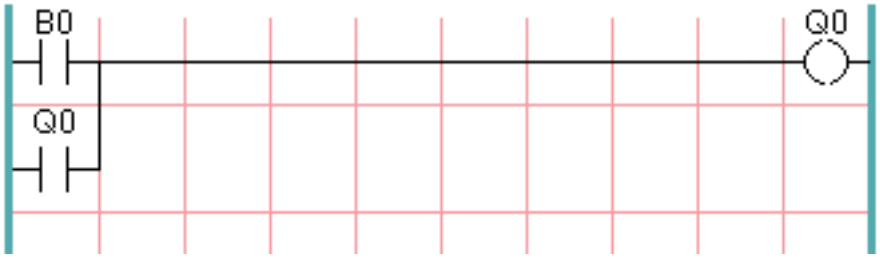
Der Eingang auf der linken Seite, B0, ein Schließer, ist mit der Spule (Ausgang) auf der rechten Seite, Q0, verbunden. Stellen Sie sich nun vor, dass am linken Ende eine Spannung angelegt wird, weil der Eingang B0 aktiv wird (z. B. weil der Eingang aktiviert ist oder der Benutzer den Schließer betätigt hat). Die Spannung hat einen direkten Weg zur Spule (Ausgang) rechts, Q0. Infolgedessen schaltet die Spule Q0 (Ausgang) von 0/aus/falsch auf 1/an/wahr. Wenn der Benutzer B0 loslässt, kehrt der Ausgang Q0 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 Q0 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 Q0 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 Q0 und verbinden ihn mit dem Tasterkontakt B0 aus unserem früheren Beispiel.

Werfen wir einen Blick darauf:

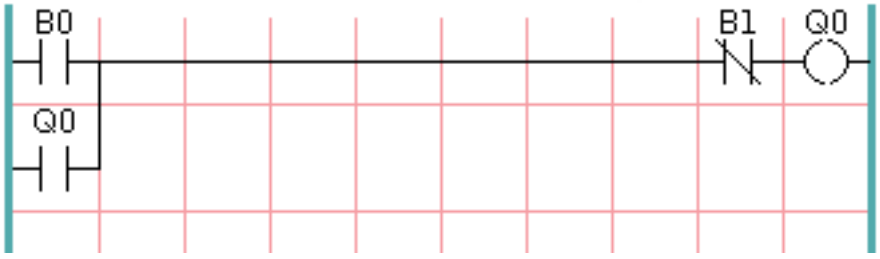


Wenn der Benutzer den Taster B0 drückt, schaltet sich wie zuvor die Spule Q0 ein. Und wenn die Spule Q0 eingeschaltet wird, dann wird der Schalter Q0 eingeschaltet. Jetzt kommt der interessante Teil. Wenn der Benutzer den Druckknopf B0 loslässt, bleibt die Spule Q0 nicht wie zuvor stehen. Das liegt daran, 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 B0, 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 Q0 zu unterbrechen. Fügen wir also einen Öffner-Taster direkt vor der Spule Q0 ein.

Das würde folgendermaßen aussehen:



Jetzt haben wir den Taster B1 für ”Aus” oder ”Stopp” hinzugefügt. Wenn der Benutzer ihn drückt, wird der Kontakt zwischen der Sprosse und der Spule unterbrochen. Wenn die Spule Q0 keinen Strom mehr hat, geht sie auf 0/aus/falsch. Wenn die Spule Q0 erlischt, dann erlischt auch der Schalter Q0, so dass der ”Haltekontakt” unterbrochen oder der Stromkreis ”entsiegelt” wird. Wenn der Benutzer die Stopptaste loslässt, wird der Kontakt von der Sprosse zur Spule Q0 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 / Kontaktplan 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 Echtzeitmodul classicladder_rt
- Das Benutzerraummodul (einschließlich einer grafischen Benutzeroberfläche) classicladder

8.2.3.1 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.

Anmerkung
Kontaktplan-Dateien (.clp) dürfen keine Leerzeichen im Namen enthalten.

8.2.3.2 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.

8.2.3.3 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.

Tabelle 8.1: 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
Anzahl der Bitpins der HAL-Eingänge	(numPhysInputs)	15
Anzahl der HAL-Ausgangsbit-Pins	(numPhysOutputs)	15
Anzahl der arithmetischen Ausdrücke	(numArithmExpr)	50
Anzahl der Abschnitte	(numSections)	10
Anzahl der Symbole	(numSymbols)	Auto
Anzahl der S32-Eingänge	(numS32in)	10
Anzahl der S32-Ausgänge	(numS32out)	10
Anzahl der Float-Eingänge	(numFloatIn)	10
Anzahl der Float-Ausgänge	(numFloatOut)	10

Objekte von größtem Interesse sind numPhysInputs, numPhysOutputs, numS32in und numS32out.

Das Ändern dieser Zahlen ändert die Anzahl der verfügbaren HAL-Bit-Pins. numPhysInputs und numPhysOutputs steuern, wie viele HAL-Bit-Pins (ein/aus) verfügbar sind. numS32in und numS32out steuern, wie viele HAL-Pins für ganze Zahlen mit Vorzeichen (+- Ganzzahlbereich) verfügbar sind.

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 Benutzermoduls

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.

Zum Laden des Benutzermoduls:

```
loadusr classicladder
```

Anmerkung
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.

8.2.5.1 Sektions-Manager

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Fenster des Abschnittsmanagers.

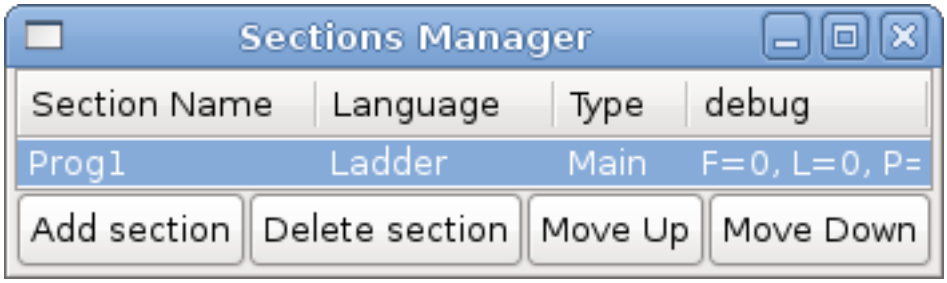


Abbildung 8.1: Sections Manager Standardfenster

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 Anrufspulen.

8.2.5.2 Abschnittsanzeige

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Abschnittsanzeigefenster. Es wird eine leere Sprosse angezeigt.

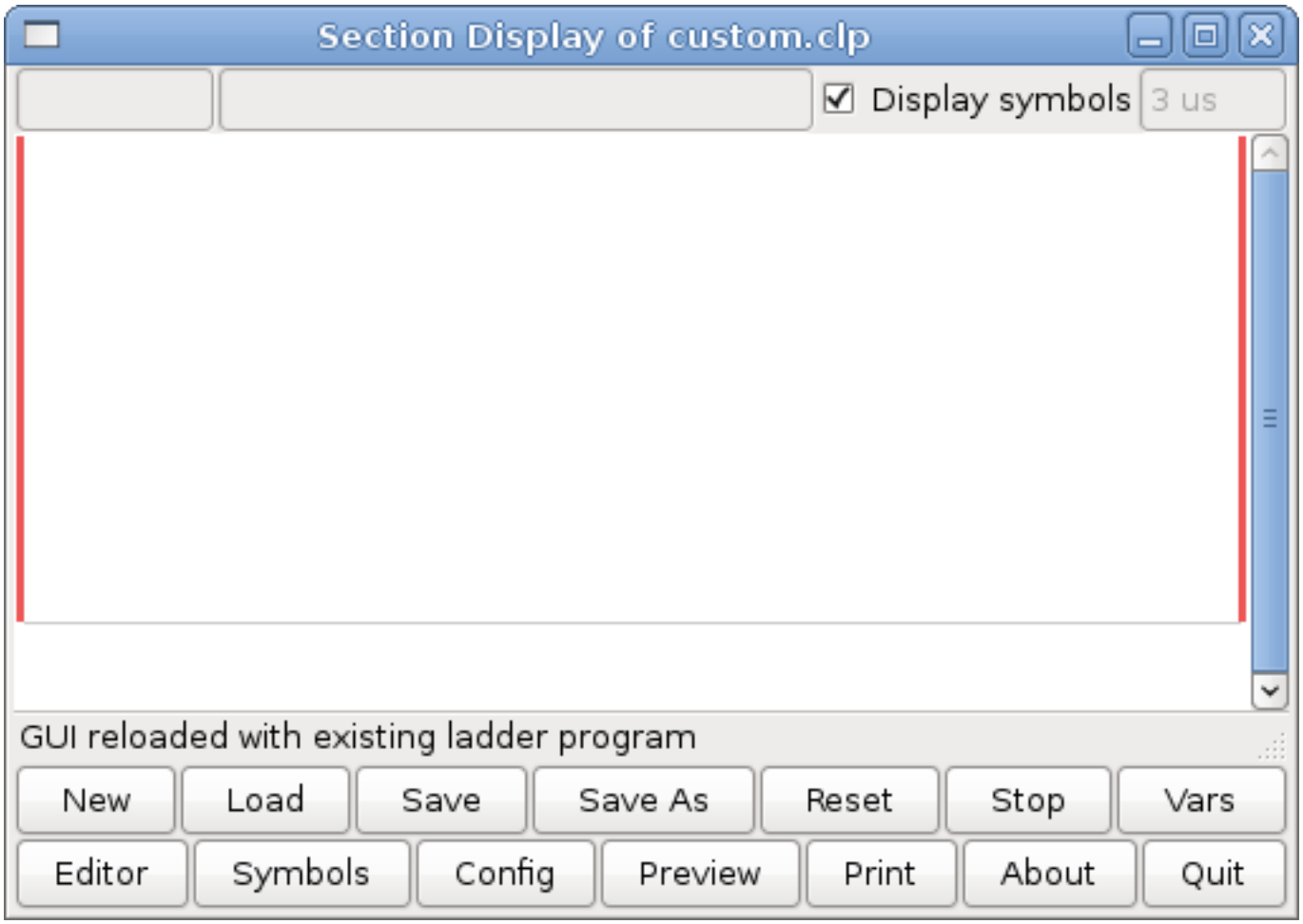


Abbildung 8.2: Standardfenster der Abschnittsanzeige

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).

Der Button Beenden (engl. quit) beendet das Benutzerprogramm, d. h. Modbus und das Display. Das Echtzeit-Kontaktplanprogramm läuft weiterhin im Hintergrund.

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.

8.2.5.3 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.

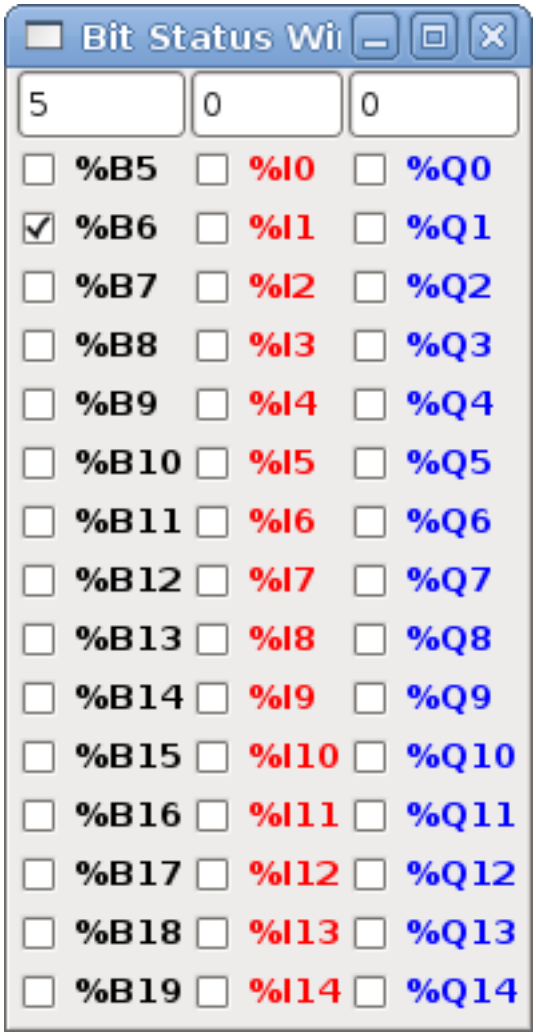


Abbildung 8.3: Bit-Status-Fenster

Im Bitstatusfenster werden einige der booleschen Variablen (ein/aus) angezeigt. Beachten Sie, dass alle Variablen mit dem %-Vorzeichen beginnen. Die %I-Variablen stellen HAL-Eingangsbit-Pins dar. Der %Q steht für die Relaisspule und die HAL-Ausgangsbitpins. Das %B steht für eine interne Relaisspule oder einen internen Kontakt. In den drei Bearbeitungsbereichen oben können Sie auswählen, welche 15 Variablen in jeder Spalte angezeigt werden. Wenn beispielsweise die Spalte %B Variable 15 Einträge hoch wäre und Sie oben in der Spalte 5 eingegeben haben, werden die Variablen %B5 bis %B19 angezeigt. Mit den Kontrollkästchen können Sie %B-Variablen manuell festlegen und deaktivieren, solange das Leiterprogramm sie nicht als Ausgaben festlegt. Alle Bits, die vom Programm

als Ausgaben festgelegt werden, wenn ClassicLadder ausgeführt wird, können nicht geändert werden und werden als aktiviert angezeigt, wenn sie aktiviert sind, und als deaktiviert, wenn sie deaktiviert sind.

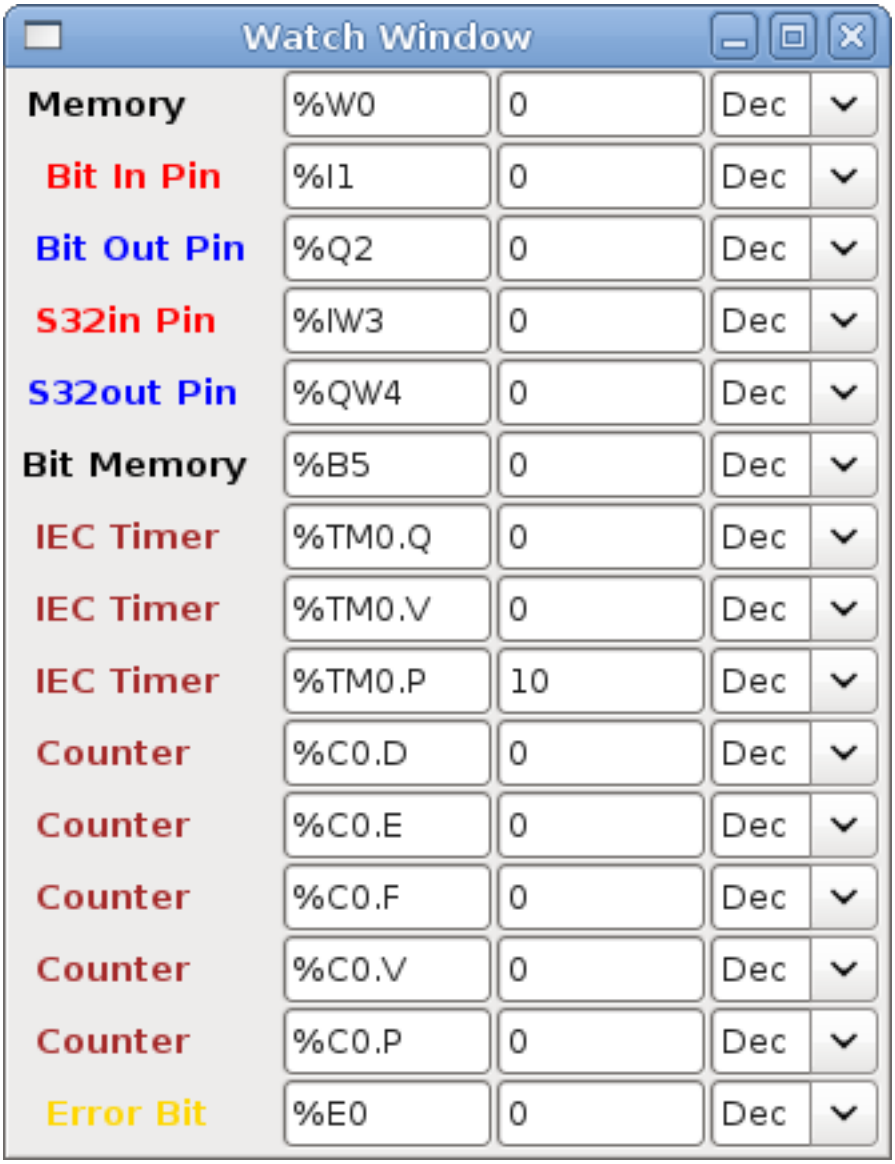


Abbildung 8.4: Schaufenster

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.

8.2.5.4 Symbol-Fenster

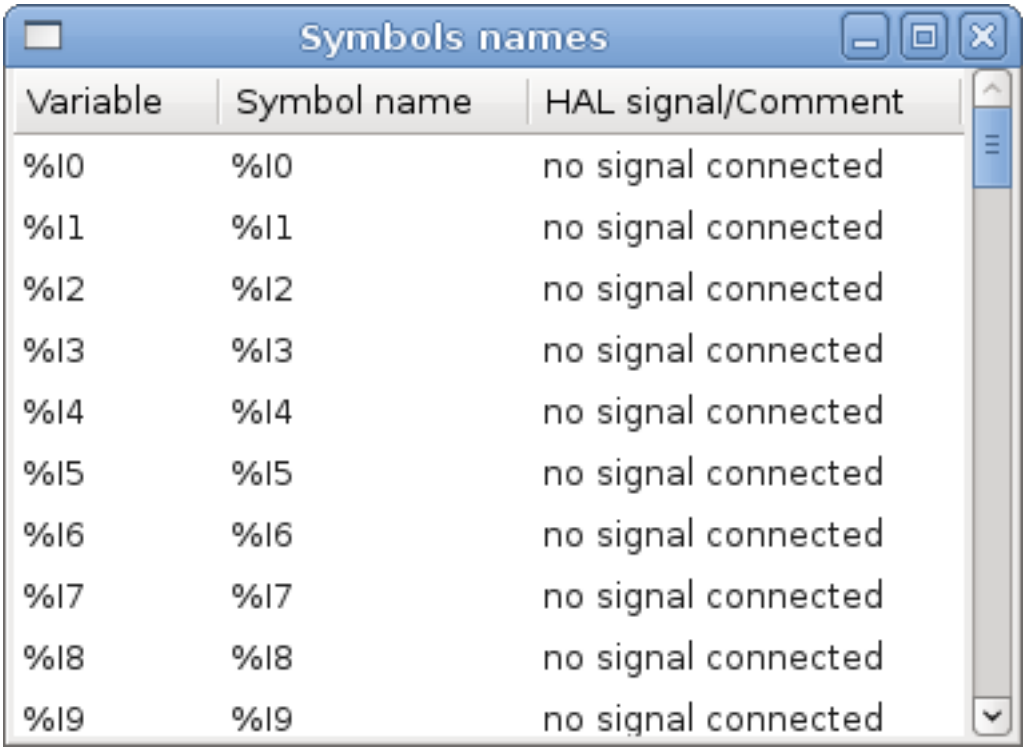


Abbildung 8.5: Fenster mit Symbolnamen

Dies ist eine Liste von ”Symbol”-Namen, die anstelle von Variablennamen im Schnittfenster angezeigt werden sollen, wenn das Kontrollkästchen ”Symbole anzeigen” aktiviert ist. Sie fügen den Variablennamen (denken Sie an das %-Symbol und Großbuchstaben), den Symbolnamen . Wenn an die Variable ein HAL-Signal angeschlossen werden kann (%I, %Q und %W - wenn Sie den s32-Pin mit dem Echtzeitmodul geladen haben), wird im Kommentarbereich der Name des aktuellen HAL-Signals oder das Fehlen eines solchen angezeigt. Symbolnamen sollten zur besseren Darstellung kurz gehalten werden. Denken Sie daran, dass Sie die längeren HAL-Signalnamen der Variablen %I, %Q und %W anzeigen lassen können, indem Sie sie im Abschnittsfenster anklicken. Auf diese Weise sollte man in der Lage sein, den Überblick darüber zu behalten, womit das Kontaktplanprogramm verbunden ist!

8.2.5.5 Das Editor-Fenster

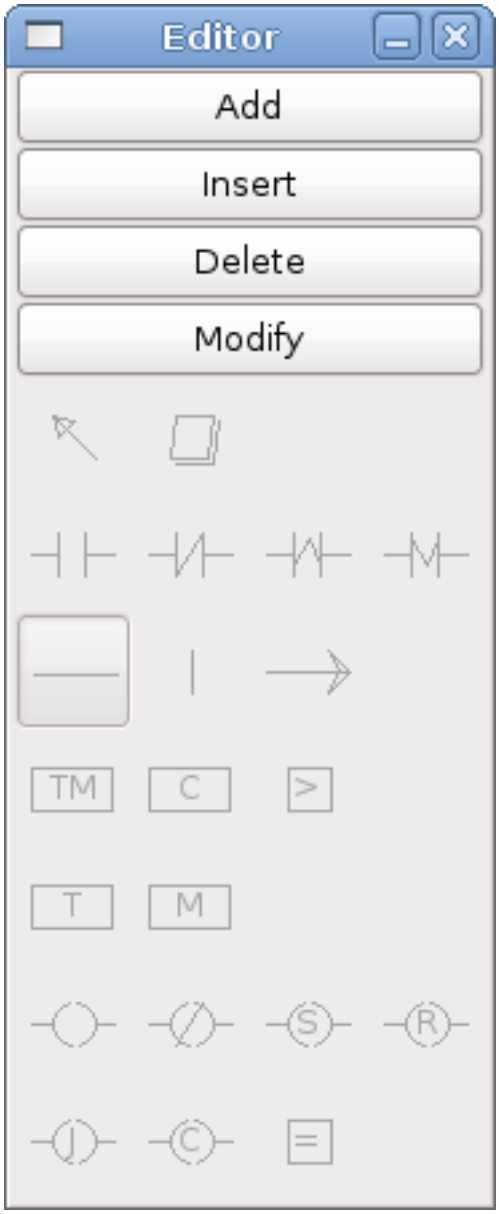


Abbildung 8.6: Editor-Fenster

- *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.
- *Variable Assignment* - erstellt einen Zuordnungsblock, 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.

8.2.5.6 Konfigurationsfenster

Das Konfigurationsfenster zeigt den aktuellen Projektstatus und enthält die Registerkarten für die Modbus-Einrichtung.

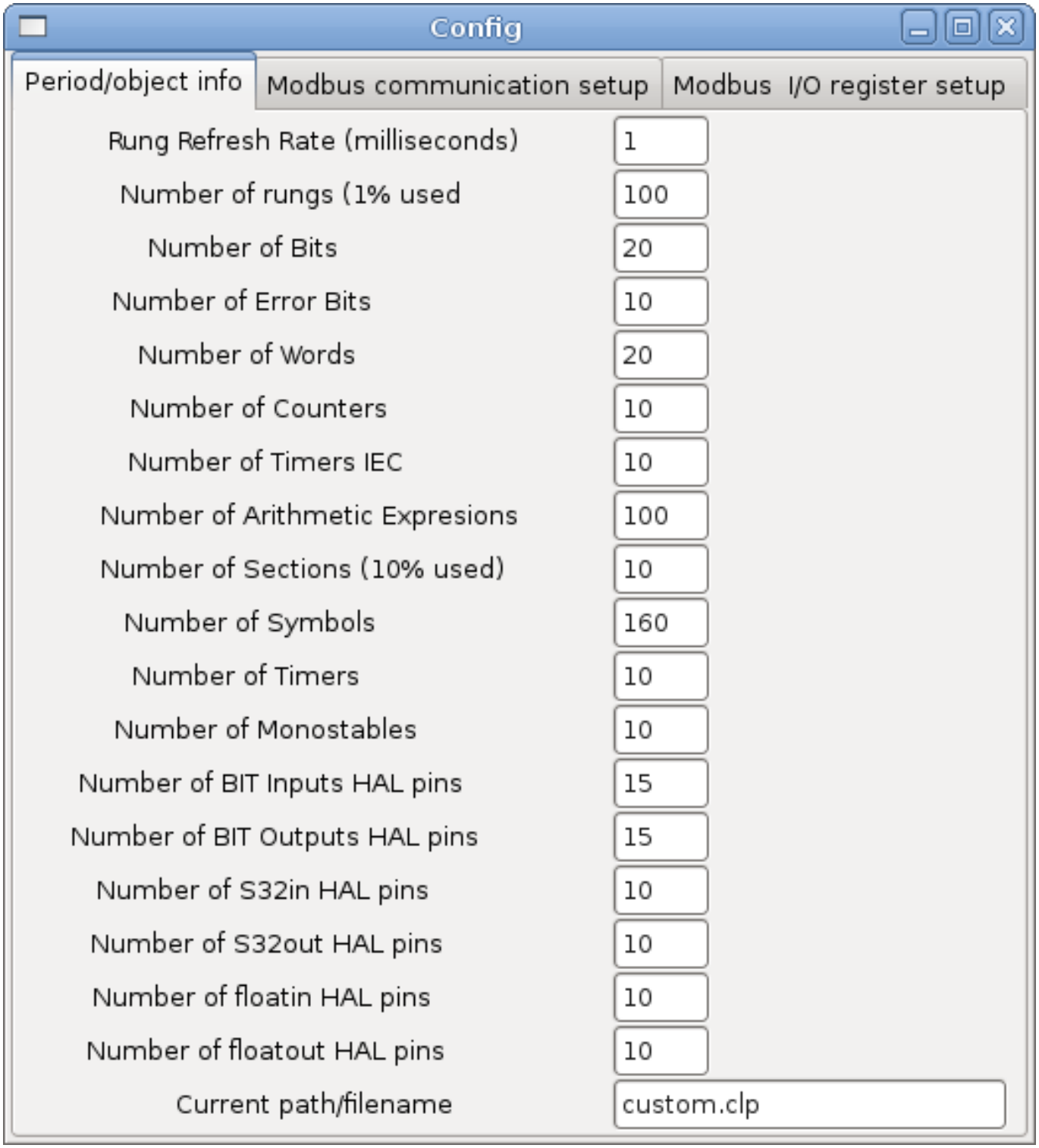


Abbildung 8.7: Konfigurationsfenster

8.2.6 SPS Objekte

8.2.6.1 KONTAKTE

Stellen Schalter oder Relaiskontakte dar. Sie werden durch den ihnen zugewiesenen variablen Buchstaben und die Nummer gesteuert.

The variable letter can be B, I, or Q and the number can be up to a three digit number eg. %I2, %Q3, or %B123. Variable I is controlled by a HAL input pin with a corresponding number. Variable B is for internal contacts, controlled by a B coil with a corresponding number. Variable Q is controlled by a Q coil with a corresponding number. (like a relay with multiple contacts). E.g. if HAL pin `classicladder.0.in-00` is true then %I0 N.O. contact would be on (closed, true, whatever you like to call it). If %B7 coil is *energized* (on, true, etc) then %B7 N.O. contact would be on. If %Q1 coil is *energized* then %Q1 N.O. contact would be on (and HAL pin `classicladder.0.out-01` would be true).

- 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.

8.2.6.2 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)

8.2.6.3 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)

8.2.6.4 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)

8.2.6.5 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.

- *%Cxx.D* - Zähler xx fertig (Boolesch, nur lesbar)
- *%Cxx.E* - Zähler xx leerer Überlauf (boolesch, schreibgeschützt)
- *%Cxx.F* - Zähler xx voller Überlauf (boolesch, nur Lesen)
- *%Cxx.V* - Zähler xx aktueller Wert (Ganzzahl, Lesen oder Schreiben)
- *%Cxx.P* - Zähler xx voreingestellt (Ganzzahl, lesen oder schreiben)

8.2.6.6 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.

8.2.6.7 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
```

Anmerkung
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.

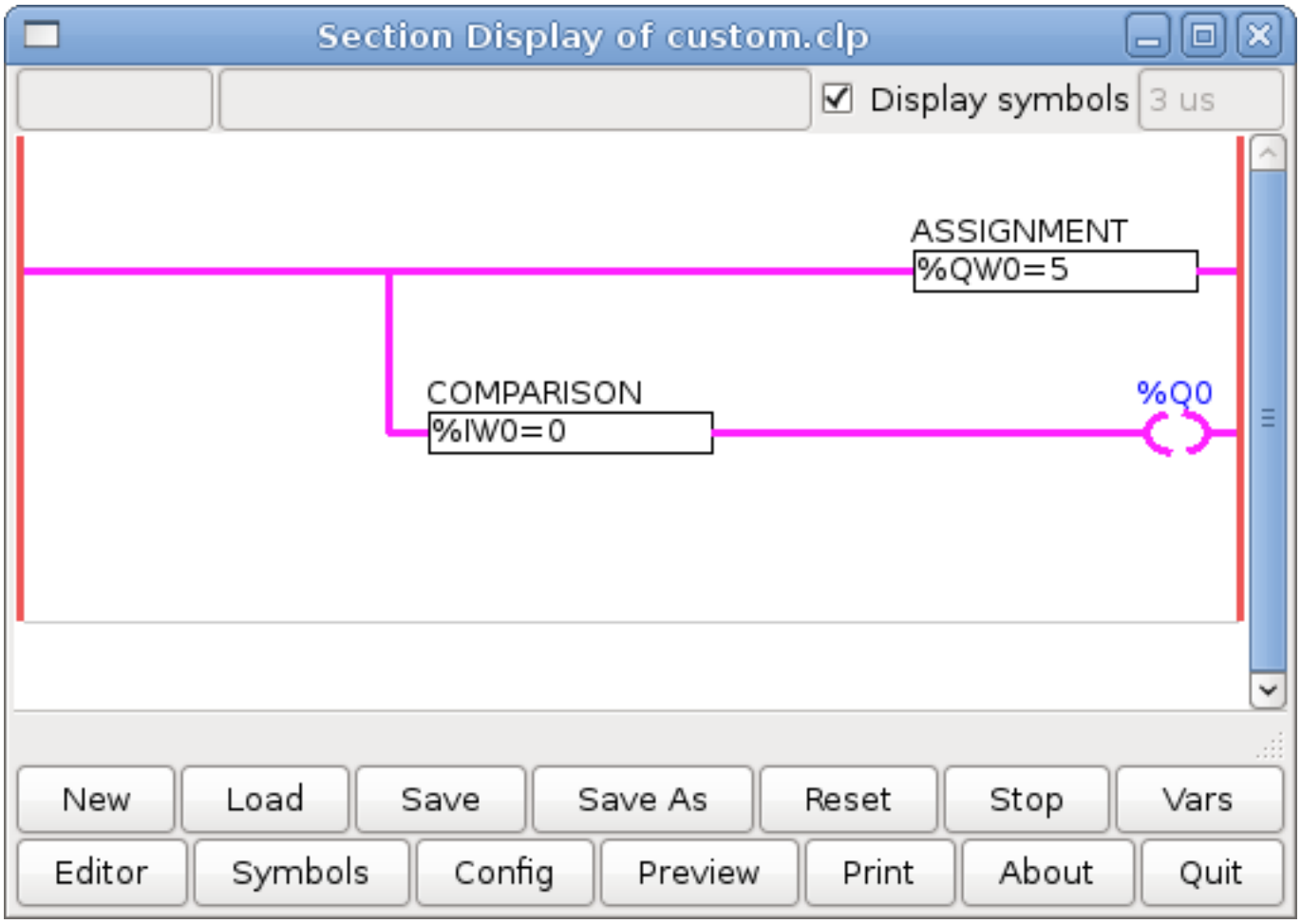


Abbildung 8.8: Beispiel für Zuordnen/Vergleichen von mit SPS

Abbildung 8.9: Beispiel für einen Zuweisungsausdruck

Abbildung 8.10: Beispiel für einen Vergleichsausdruck

8.2.6.8 SPULEN (engl. coils)

Spulen stellen Relaisspulen dar. Sie werden durch den ihnen zugewiesenen variablen Buchstaben und die Nummer gesteuert.

The variable letter can be B or Q and the number can be up to a three digit number, e.g., %Q3, or %B123. Q coils control HAL out pins, e.g. if %Q15 is energized then HAL pin classicladder.0.out-15 will be true. B coils are internal coils used to control program flow.

- 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)

Warnung
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!

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.

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 :

- *%Bxxx* - Bitspeicher xxx (boolesch)
- *%Wxxx* - Word-Speicher xxx (32 Bit Vorzeichen)
- *%IWxxx* - Wortspeicher xxx (S32 in Pin)
- *%QWxxx* - Word-Speicher xxx (S32-Ausgangspin)
- *%IFxx* - Word-Speicher xx (Float in Pin) (**konvertiert in S32 in ClassicLadder**)
- *%QFxx* - Word-Speicher xx (Float-Out-Pin) (**konvertiert in S32 in ClassicLadder**)
- *%Txx.R* - Timer xx läuft (boolesch, schreibgeschützt für Benutzer)
- *%Txx.D* - Timer xx erledigt (Boolescher Wert, nur für Benutzer)
- *%Txx.V* - Timer xx aktueller Wert (ganze Zahl, schreibgeschützt für Benutzer)
- *%Txx.P* - Timer xx Voreinstellung (ganze Zahl)
- *%TMxxx.Q* - Timer xxx fertig (Boolesch, lesen/schreiben)
- *%TMxxx.P* - Timer xxx Voreinstellung (ganze Zahl, Schreiblese)
- *%TMxxx.V* - Timer xxx Wert (ganze Zahl, lesender Schreibzugriff)
- *%Mxx.R* - Monostabiles xx läuft (boolesch)

- *%Mxx.V* - Monostabiler xx-Aktuellwert (Ganzzahl, nur vom Benutzer lesbar)
- *%Mxx.P* - Monostabile xx-Voreinstellung (Ganzzahl)
- *%Cxx.D* - Zähler xx fertig (Boolesch, Benutzer schreibgeschützt)
- *%Cxx.E* - Zähler xx leerer Überlauf (Boolean, nur vom Benutzer gelesen)
- *%Cxx.F* - Zähler xx voller Überlauf (Boolean, schreibgeschützter Benutzer)
- *%Cxx.V* - Zähler xx aktueller Wert (Ganzzahl)
- *%Cxx.P* - Zähler xx Voreinstellung (ganze Zahl)
- *%Ixxx* - Physikalischer Eingang xxx (Boolean) (HAL-Eingangsbit)
- *%Qxxx* - Physikalische Ausgabe xxx (Boolesch) (HAL-Ausgangsbit)
- *%Xxxx* - Aktivität von Schritt xxx (sequentielle Sprache)
- *%Xxxx.V* - Zeit der Aktivität in Sekunden von Schritt xxx (sequenzielle Sprache)
- *%Exx* - Fehler (Boolean, read write(wird überschrieben))
- *Indexed or vectored variables* - These are variables indexed by another variable. Some might call this vectored variables. Example: %W0[%W4] => if %W4 equals 23 it corresponds to %W23

8.2.8 GRAFCET (State Machine) Programmierung

Warnung
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

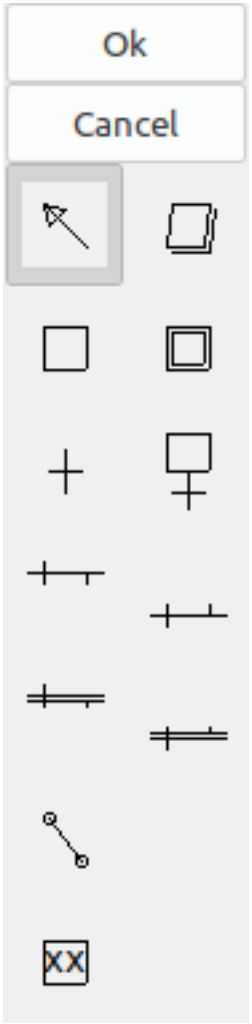


Abbildung 8.11: Fenster des Sequenzeditors

- *ORDINARY STEP* - hat für jeden eine eindeutige Nummer
- *STARTING STEP* - a sequential program must have one. This is where the program will start.
- *TRANSITION* - Dies 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 %Xxxx (z. B. %X5) wird verwendet, um festzustellen, ob ein Schritt aktiv ist. Die Variable %Xxxx.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:

- Da Modbus ein Userspace-Programm ist, kann es auf einem stark belasteten Computer zu Latenzproblemen kommen.
- 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 Userspace-Programms 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 ClassicLadder Benutzerprogramms keinen -- *modmaster* angeben, wird diese Seite nicht angezeigt.

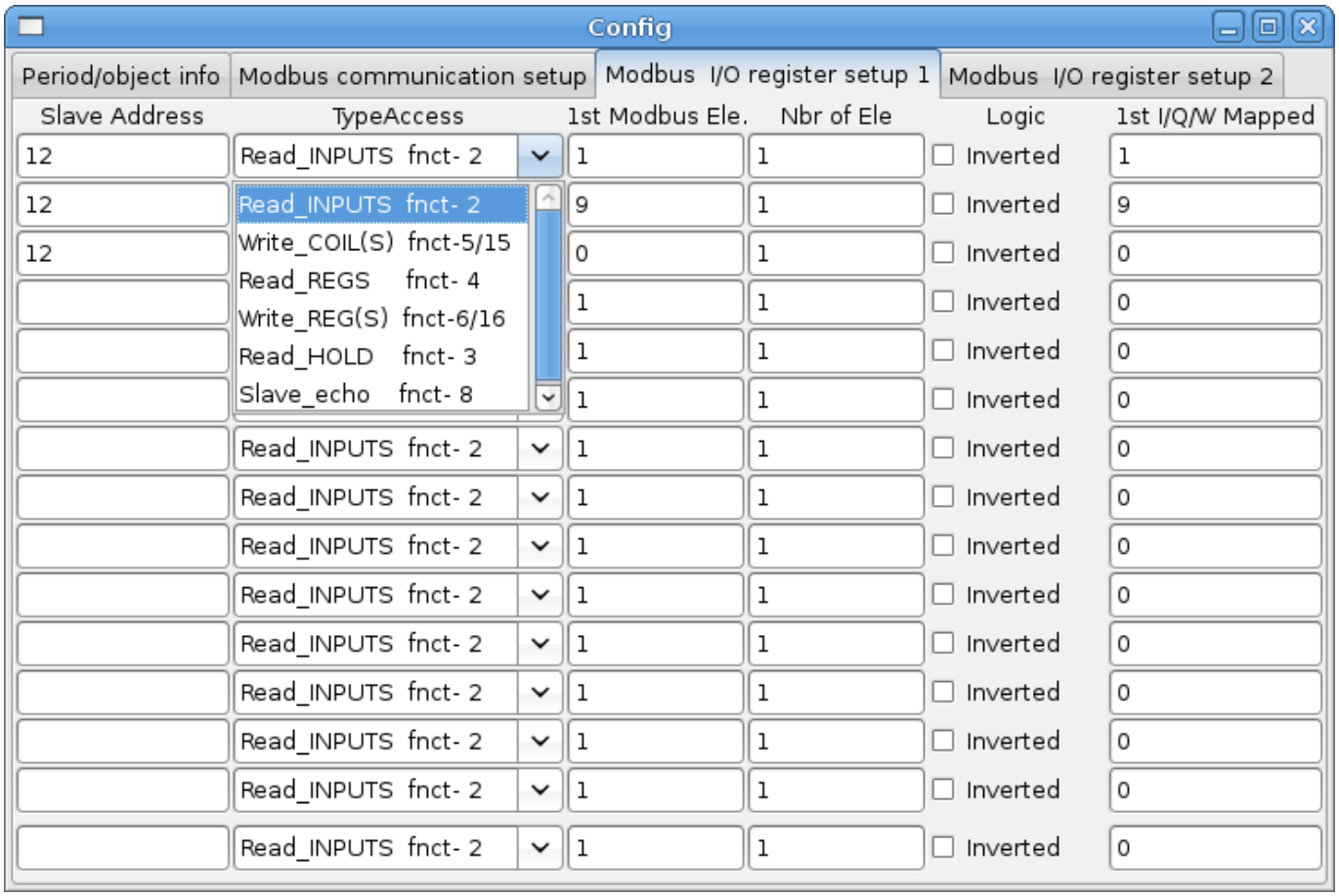


Abbildung 8.12: Modbus I/O-Konfiguration

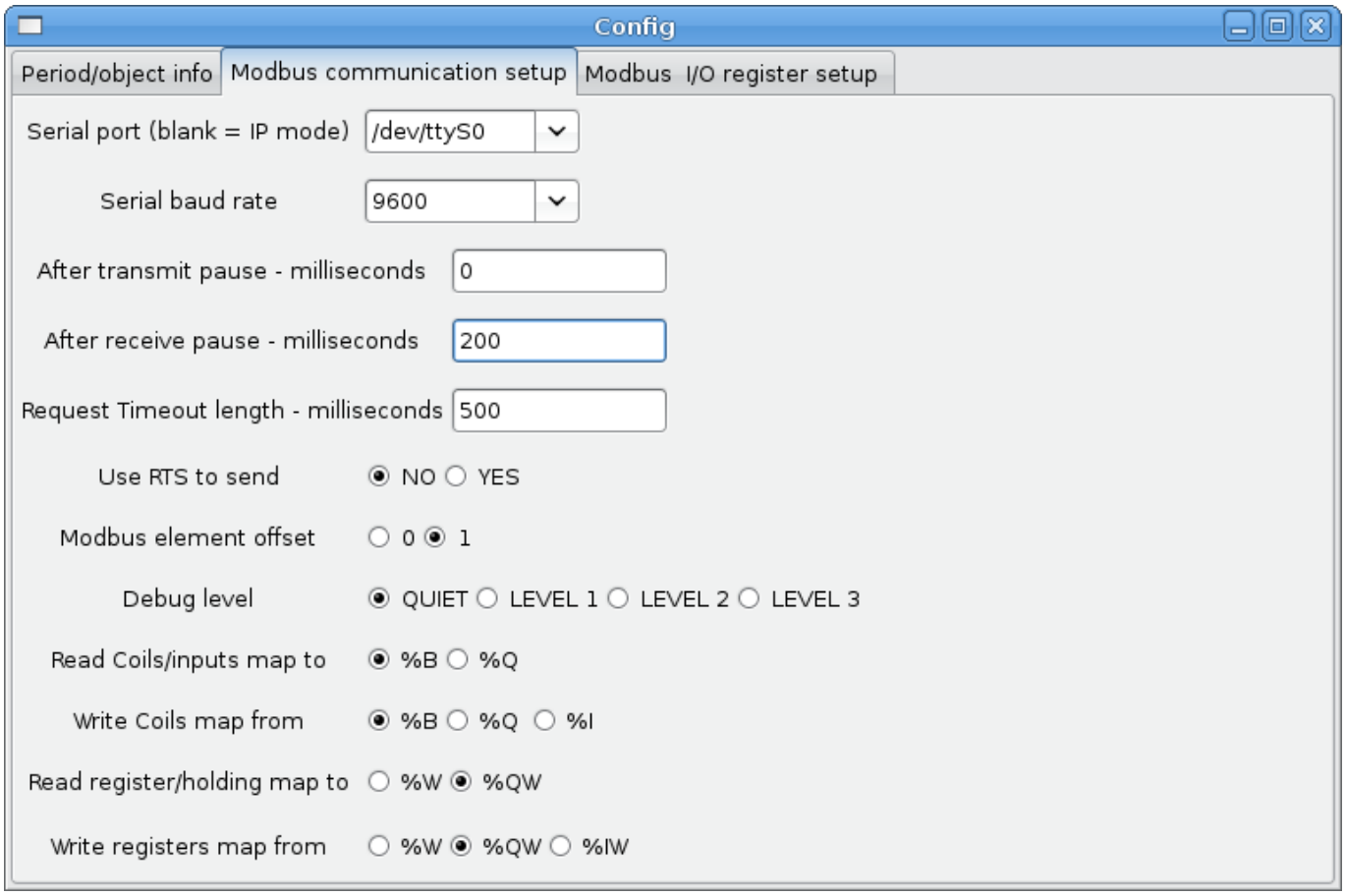


Abbildung 8.13: Modbus-Kommunikationskonfiguration

- *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 0. 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 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).

8.2.10.1 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, http://www.advantech und ein serielles Modbus/RTU-Produkt (http://www.ipac.ws).
- Siehe Beispiele: adam-6051 und modbus_rtu_serial.
- Weblinks: http://www.modbus.org und dieser interessante Link: http://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.

http://www.modbus.org/

8.2.10.2 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: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
Wenn Sie linuxcnc/classocladder von einem Terminal aus starten, werden die Modbus-Befehle und Slave-Antworten ausgedruckt.

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.

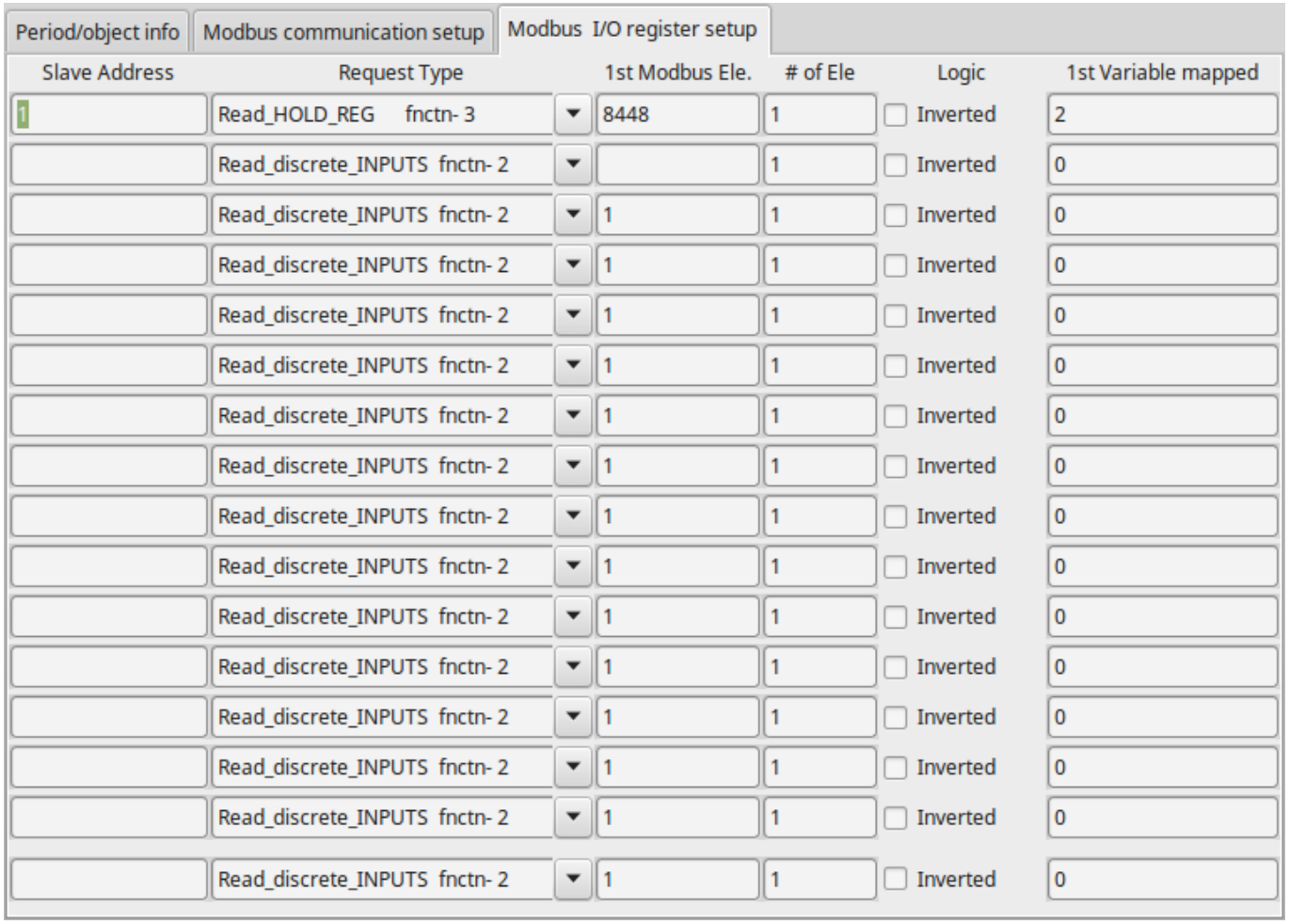


Abbildung 8.14: Modbus I/O-Register-Setup

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.

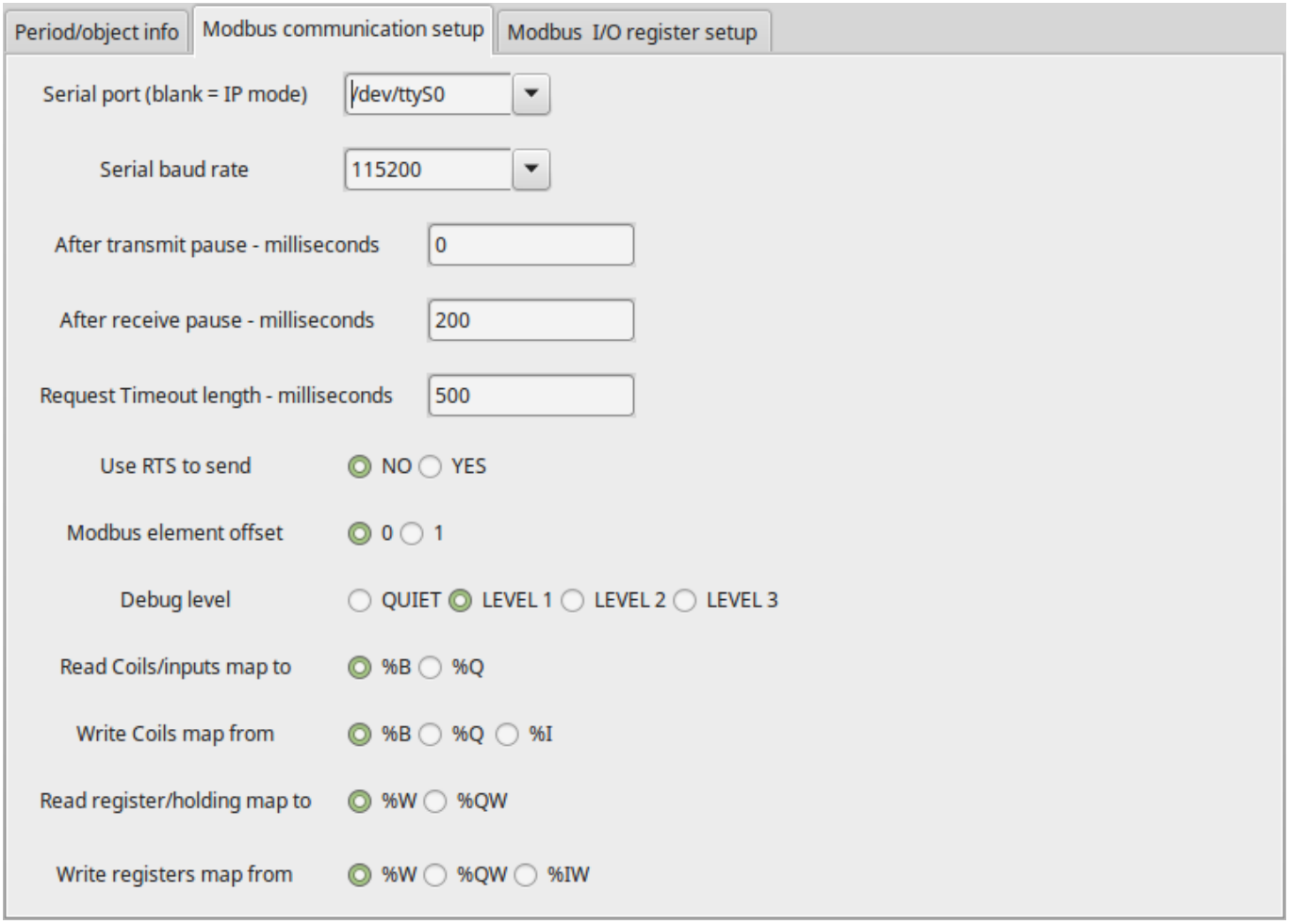


Abbildung 8.15: Einrichtung der Modbus-Kommunikation

8.2.11.1 Anfrage

Betrachten wir ein Beispiel für das Lesen eines Hold-Registers bei 8448 Decimal (0x2100 Hex). Blick in die Modbus-Protokollreferenz:

Tabelle 8.2: 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)

8.2.11.2 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:

Tabelle 8.3: 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)

8.2.11.3 Datenantwort

Blick auf das Protokoll über die Antwort:

Tabelle 8.4: Datenantwort für Funktionscode 3 (Lesen Holdingregister)

Name	**Anzahl Bytes**	**Wert (hex)**
Funktionscode	1 Byte	3 (0x03)
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.

8.2.11.4 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 Classic-Ladder 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 Benutzerprogramm 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”.

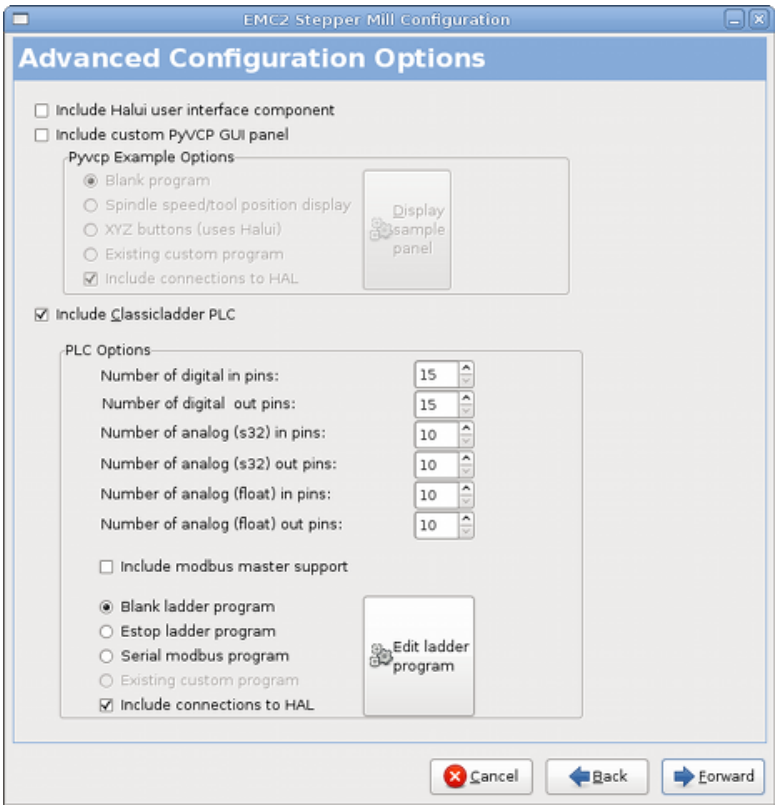


Abbildung 8.16: Stepconf ClassicLadder

8.2.12.1 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
```

8.2.12.2 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.

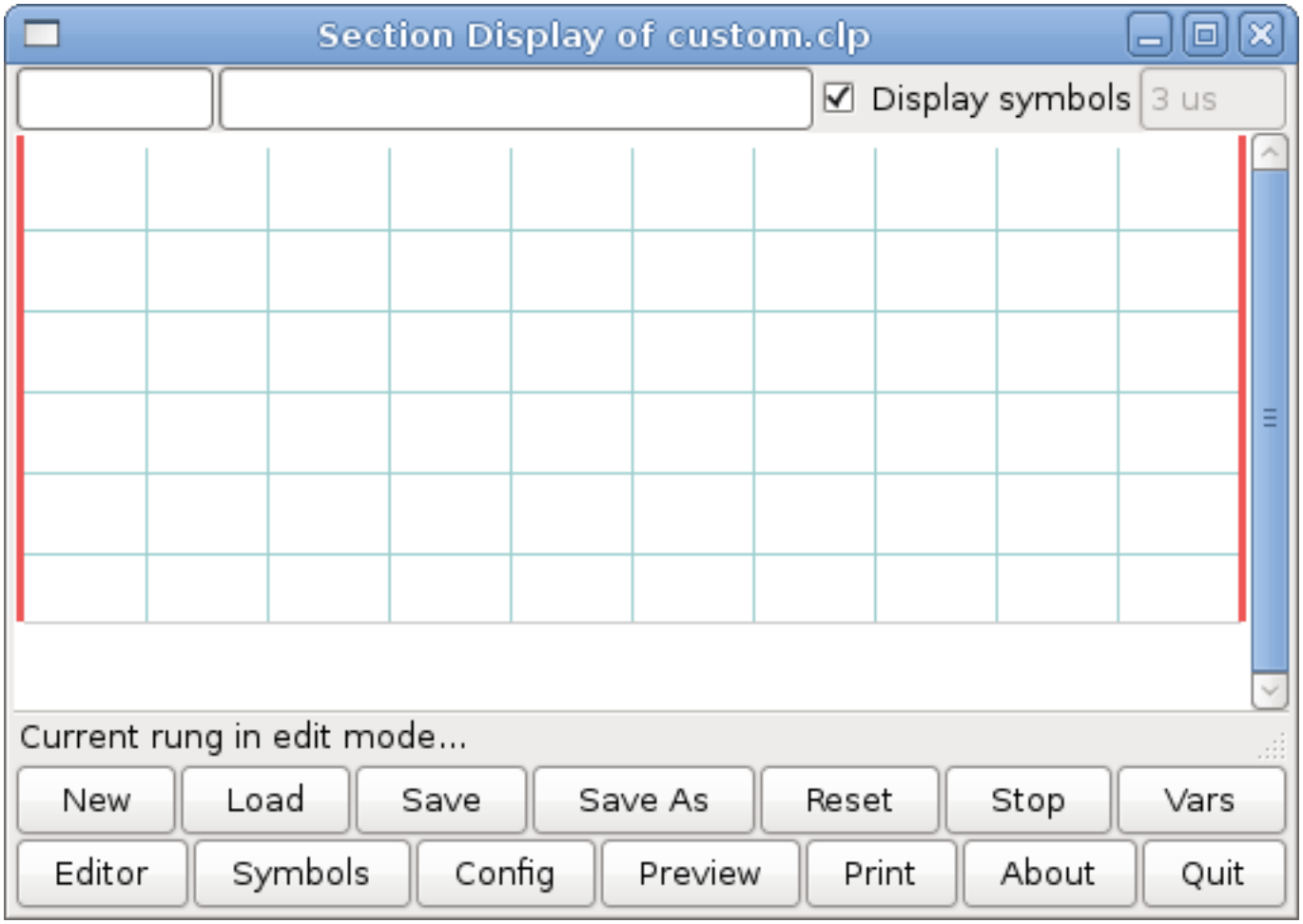


Abbildung 8.17: Abschnitt Display mit Grid

Klicken Sie nun auf den N.O.-Eingang im Editorfenster.

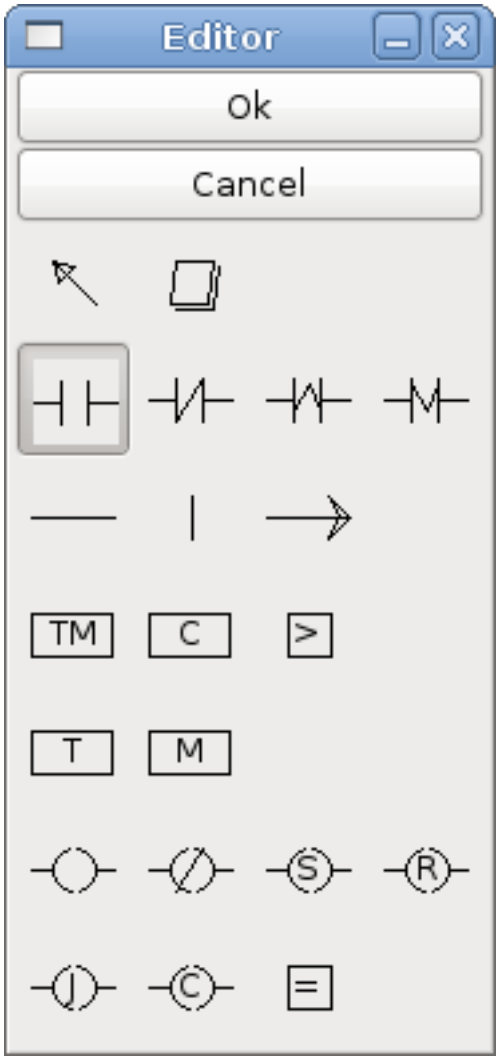


Abbildung 8.18: Editor-Fenster

Klicken Sie nun in das obere linke Raster, um den N.O.-Eingang in der Leiter zu platzieren.

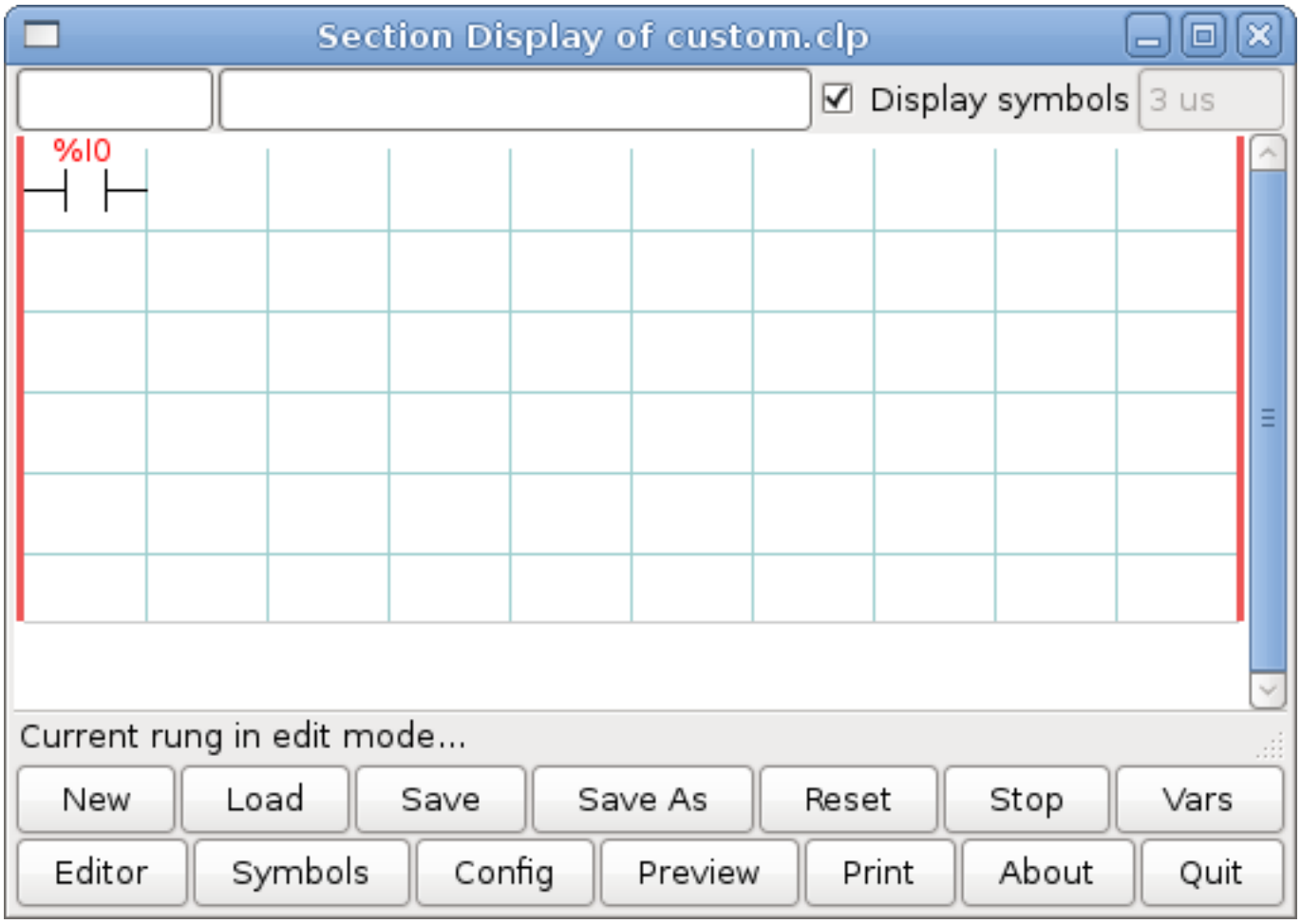


Abbildung 8.19: Abschnitt Display mit Input

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.

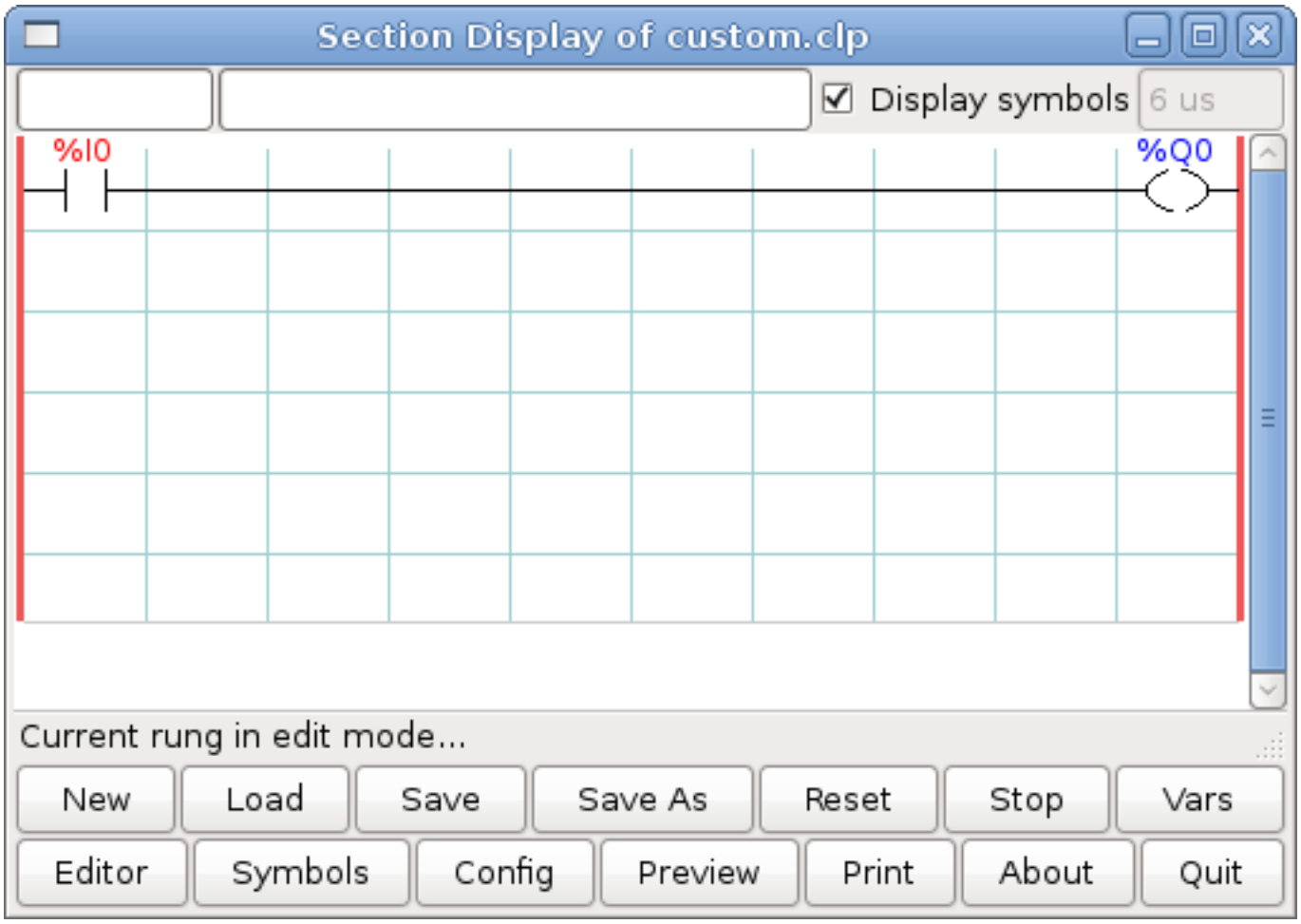


Abbildung 8.20: Abschnittsanzeige mit Sprosse

Klicken Sie nun im Editor-Fenster auf die Schaltfläche OK. Jetzt sollte Ihre Abschnittsanzeige wie folgt aussehen:

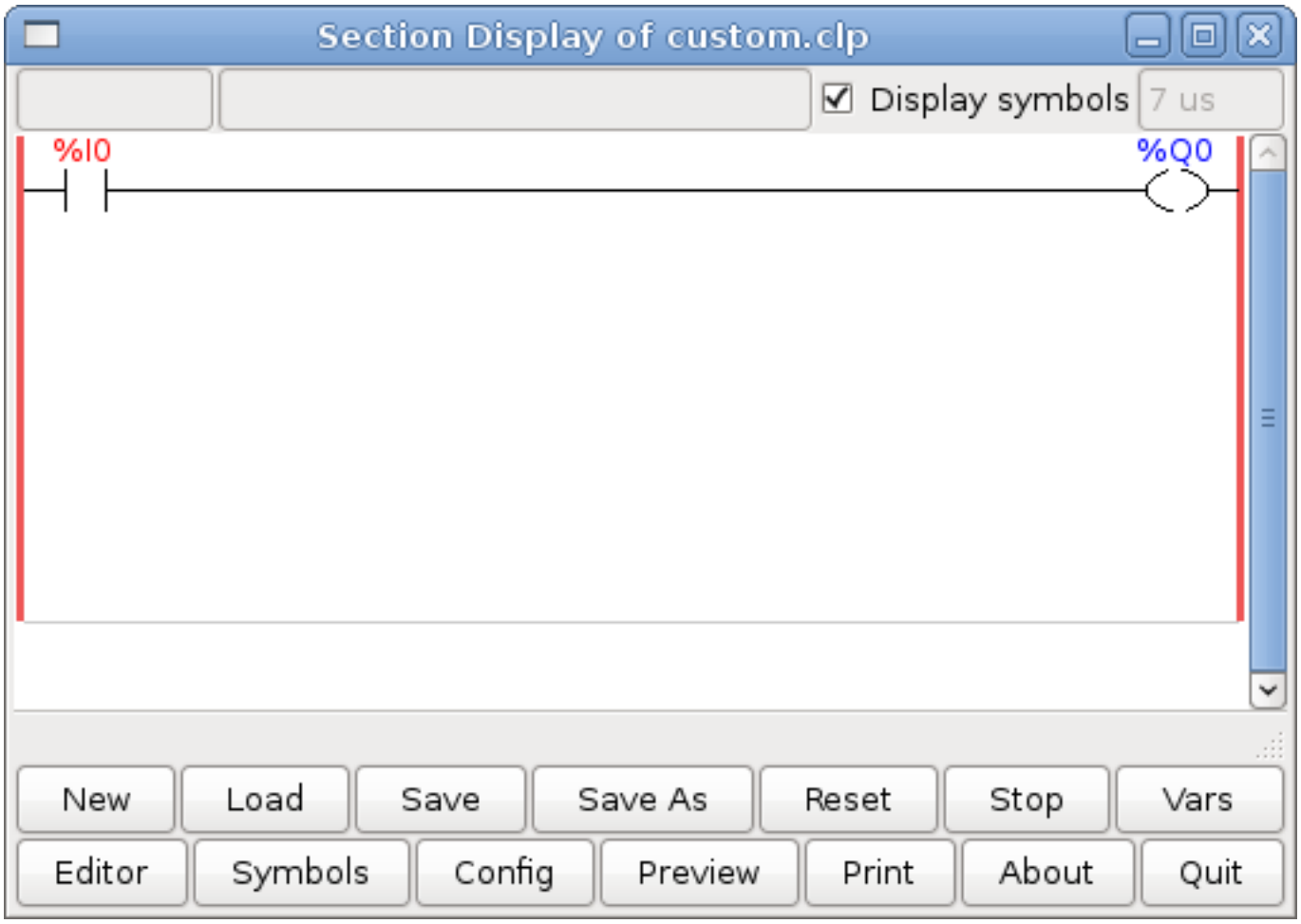


Abbildung 8.21: Abschnitt Anzeige Beendet

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.

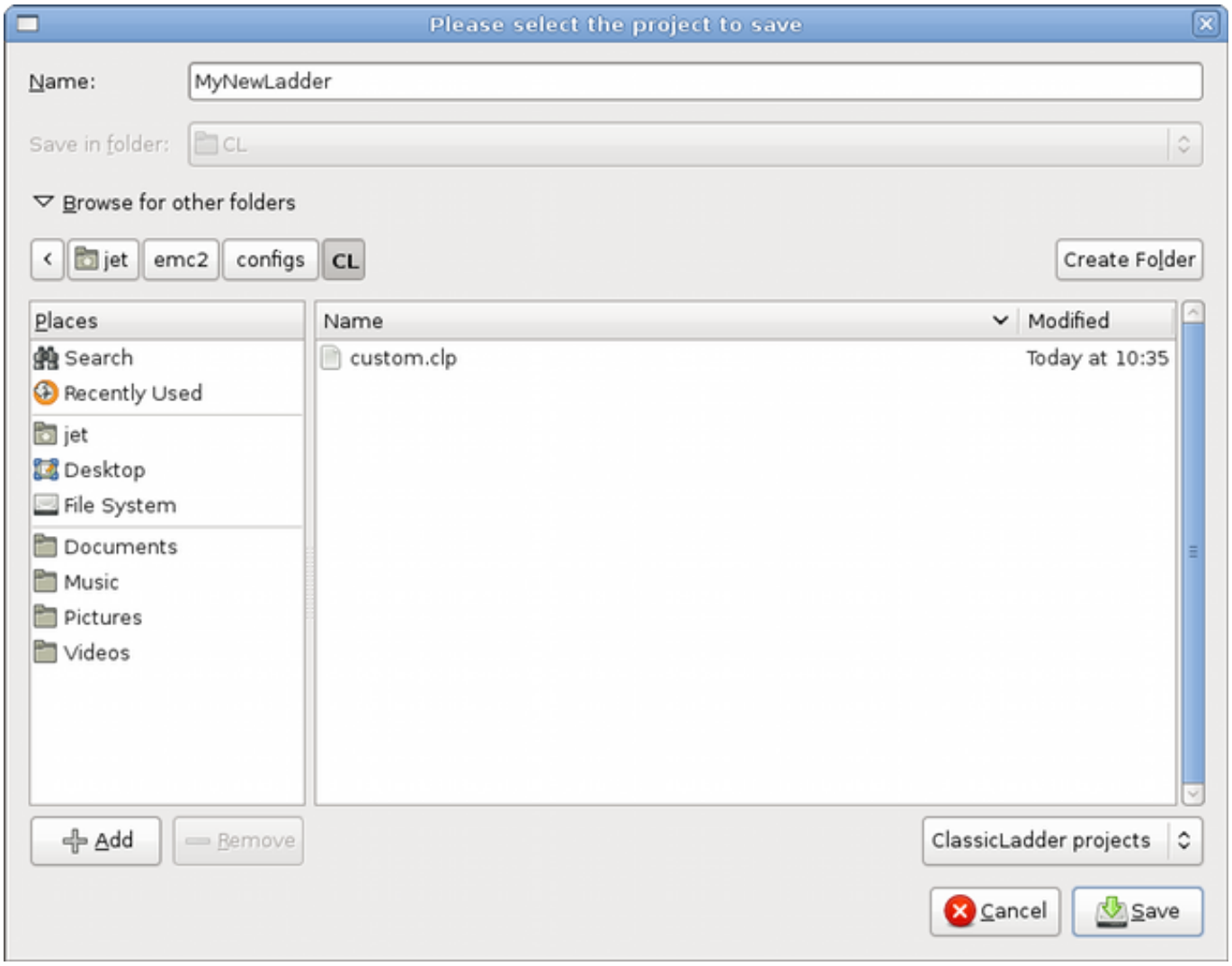


Abbildung 8.22: Speichern unter Dialog

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.

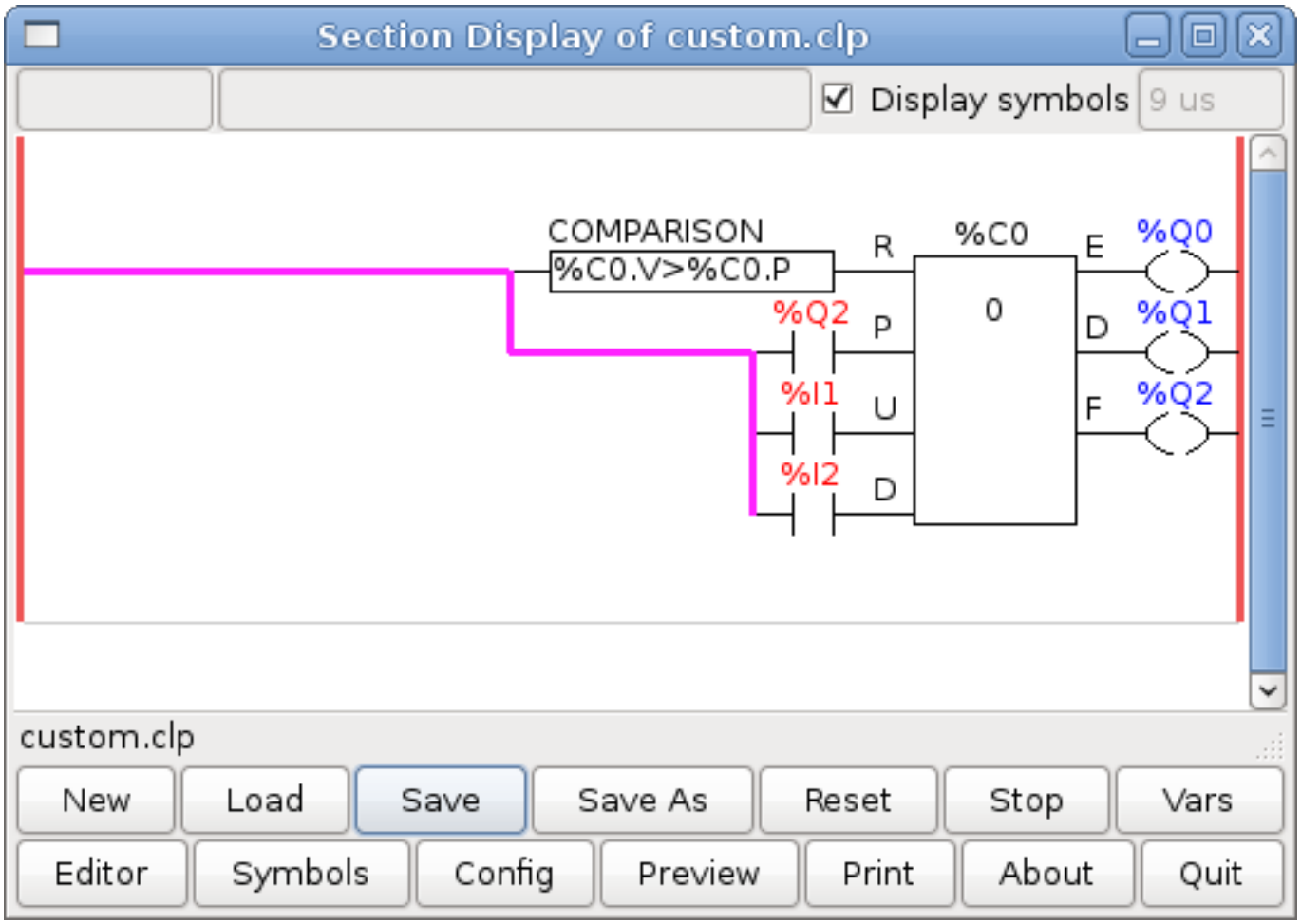


Abbildung 8.23: Umlaufender (engl. wrapping) Zähler

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.

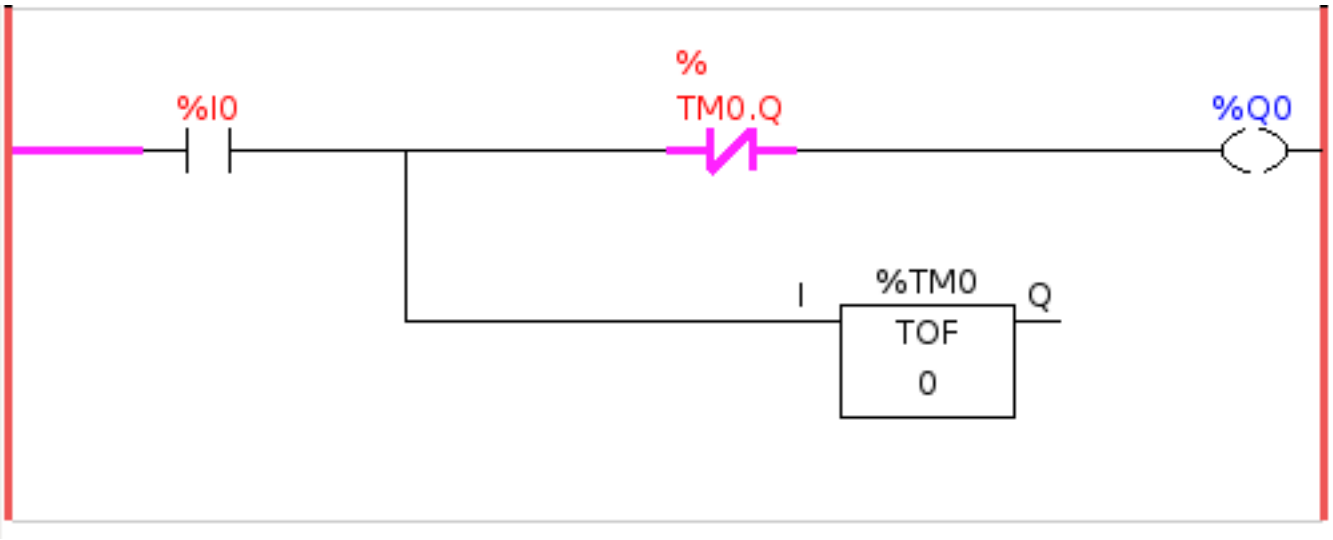


Abbildung 8.24: Extra-Impuls ablehnen

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.

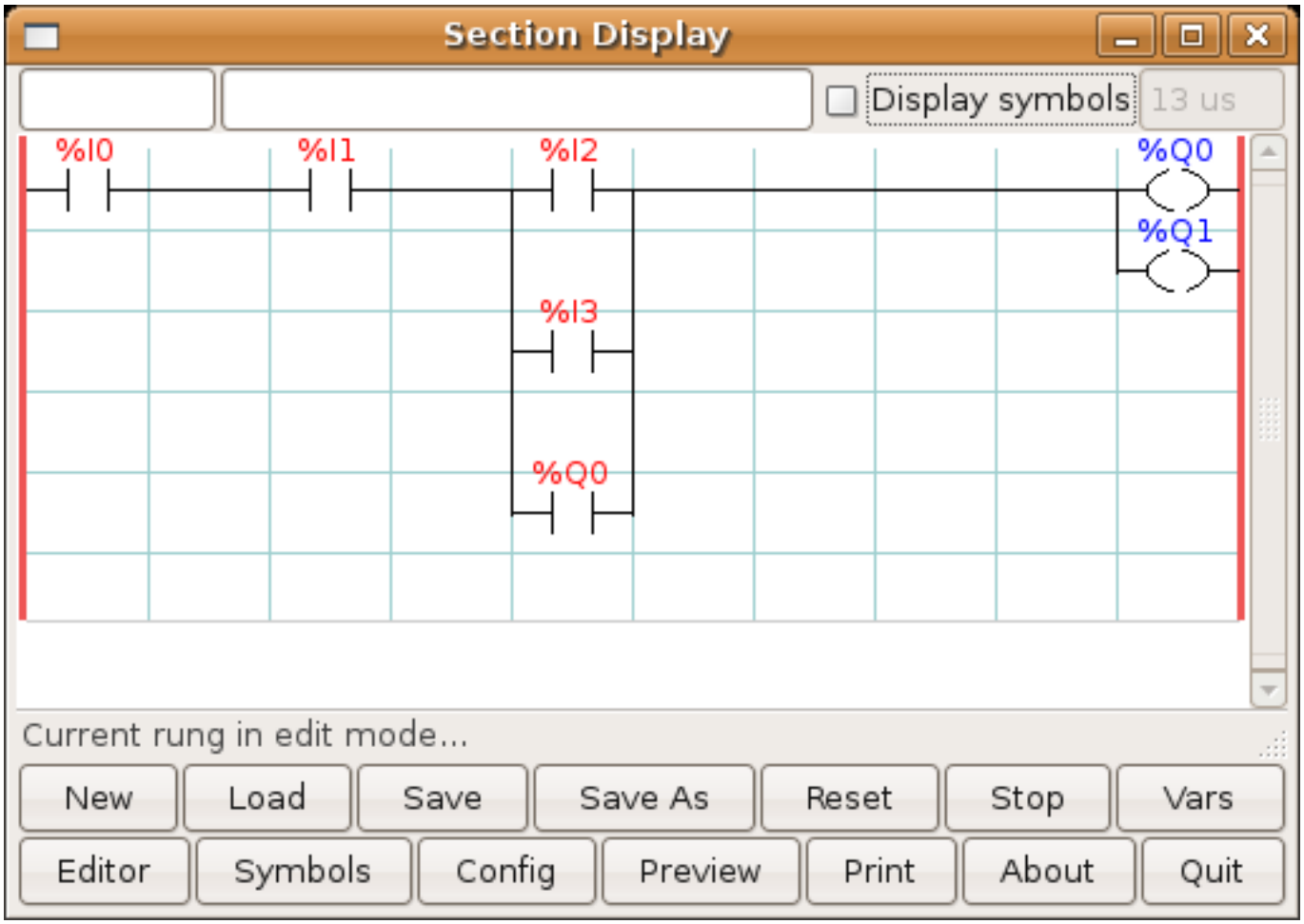


Abbildung 8.25: Anzeige des Notaus-Bereichs

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.

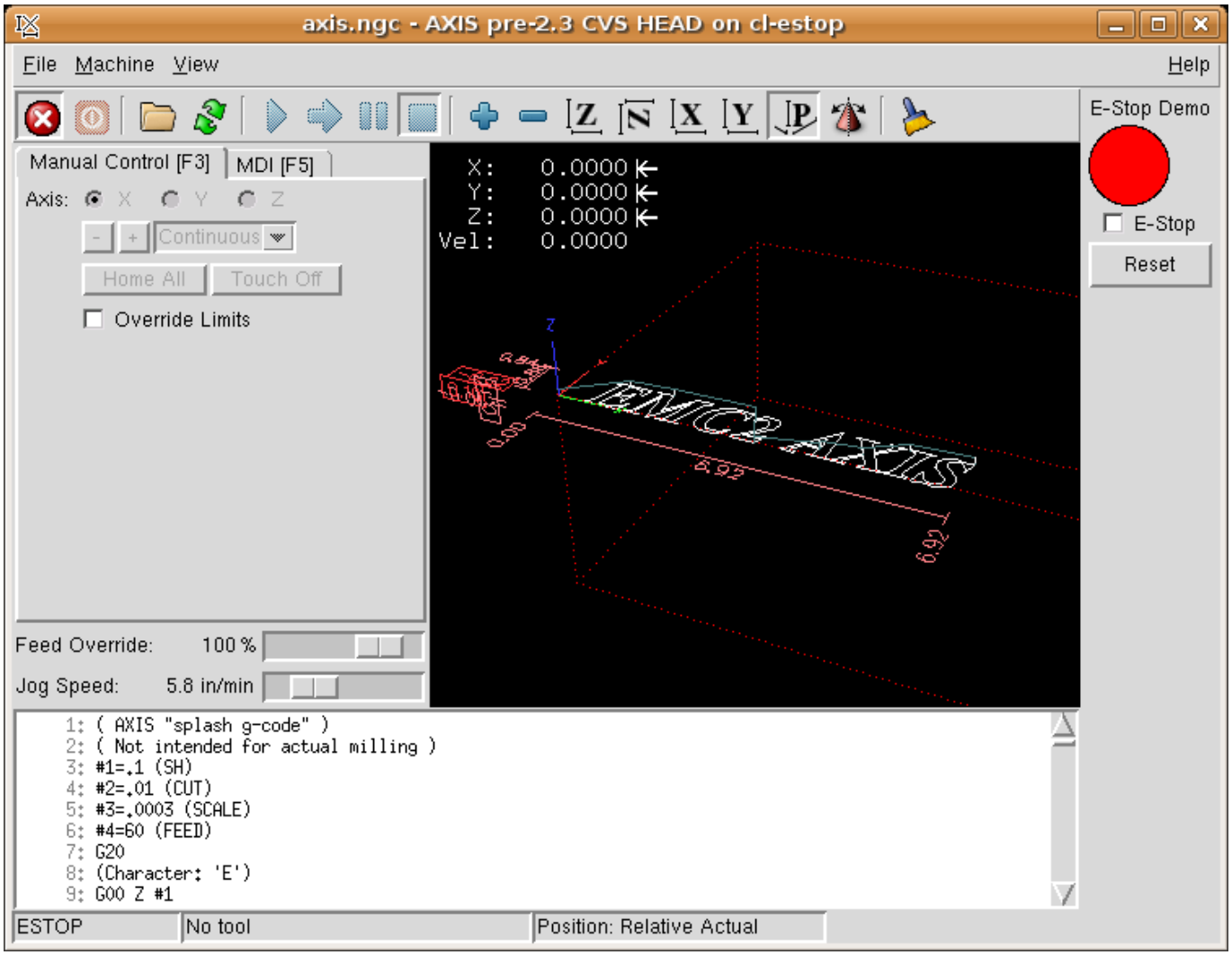


Abbildung 8.26: AXIS Notaus

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.

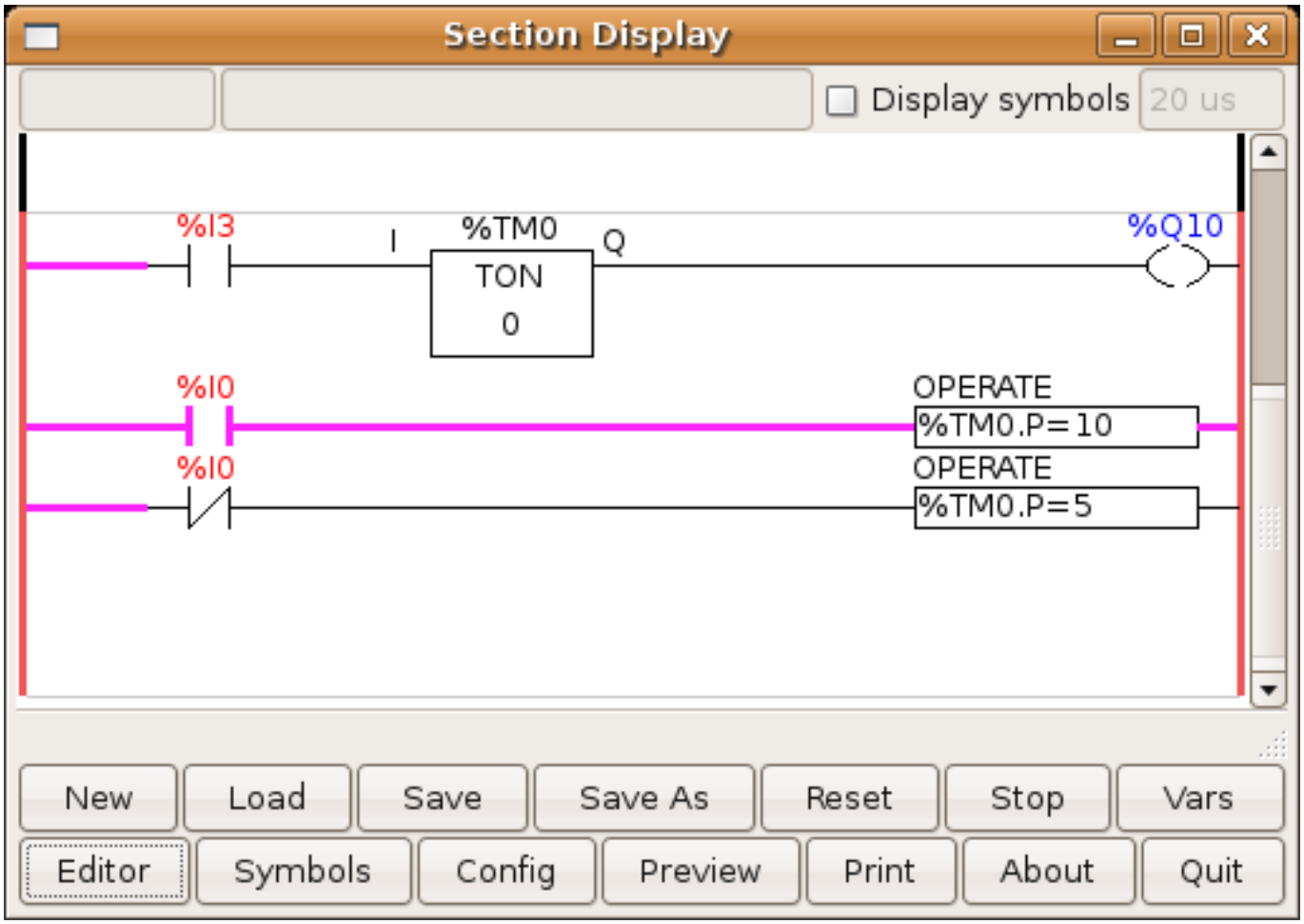


Abbildung 8.27: Beispiel für Timer/Bedienung

In diesem Fall ist %I0 wahr, so dass der voreingestellte Wert des Timers 10 ist. Wäre %I0 falsch, wäre der voreingestellte Zeitgeberwert 5.

Kapitel 9

Fortgeschrittene Themen

9.1 Kinematiken

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].

9.1.1.1 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.

[1] Kinematik: eine Zwei-Wege-Funktion, um aus dem kartesischen Raum in den Gelenkraum zu transformieren.

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:

[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.

```
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

There can be quite a few types of machine setups (robots: puma, scara; hexapods etc.). Each of them is set up using linear and rotary joints. These joints don’t usually match with the Cartesian coordinates, therefore we need a kinematics function which does the conversion (actually 2 functions: forward and inverse kinematics function).

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.

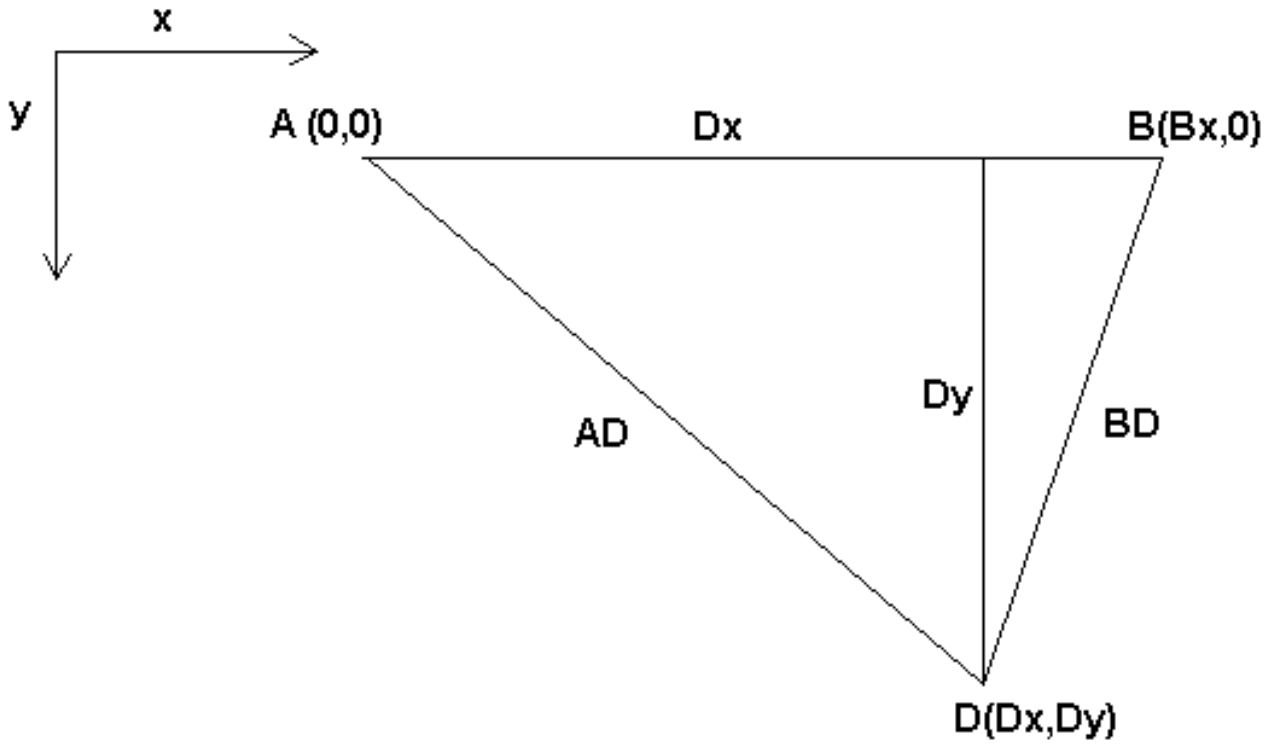


Abbildung 9.1: Zweibein-Einrichtung

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.

9.1.3.1 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;
```

9.1.3.2 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. KINEMATICS_FORWARD_ONLY
4. KINEMATICS_INVERSE_ONLY

Anmerkung
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.

Anmerkung
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.

9.1.4.1 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 Einrichten ”modifizierter” Denavit-Hartenberg (DH)-Paramete für ”genserkins”

9.2.1 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.

Anmerkung
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.

Anmerkung
Es kann Fehler und/oder Mängel geben - Nutzung auf eigene Gefahr!

9.2.2 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*).

9.2.3 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.

9.2.4 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** : positive distance, along X, between two joint axes specified in *machine units* (mm or inch) defined in the system’s INI file.
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.

9.2.5 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.

9.2.6 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).

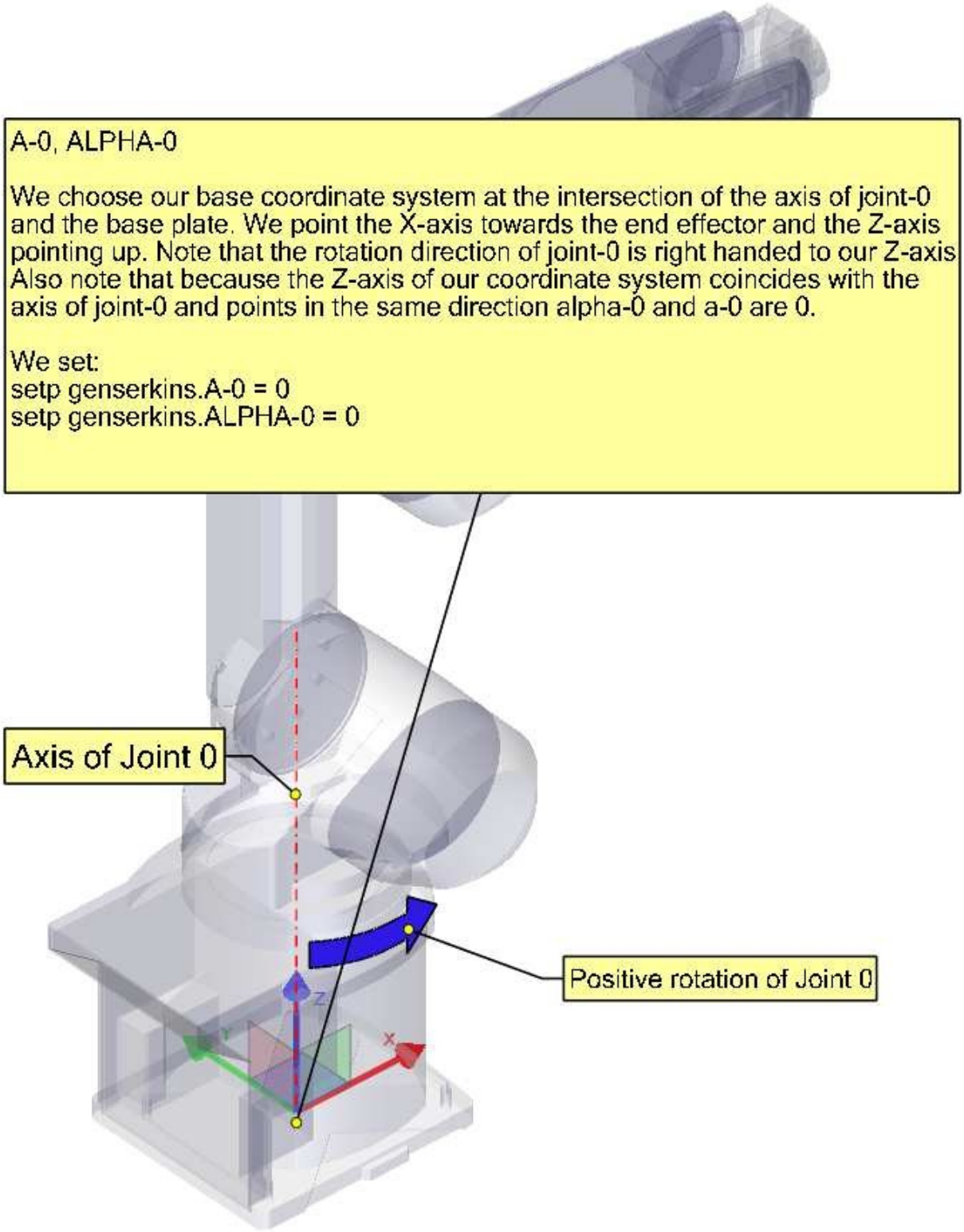
Und so sitzt der TCP-CS des Endeffektors in der Mitte des Handflansches.

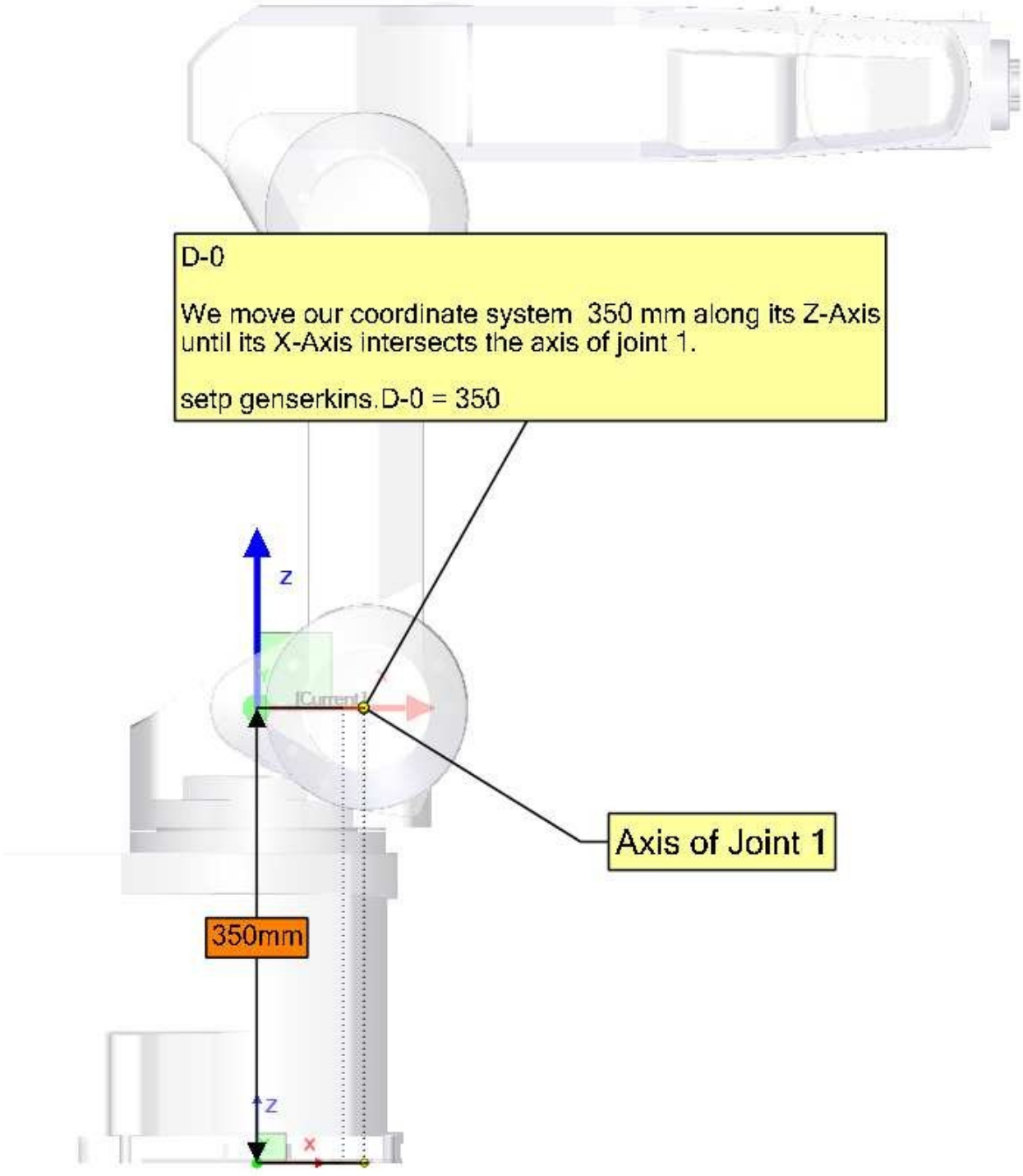
9.2.7 Sonderfälle

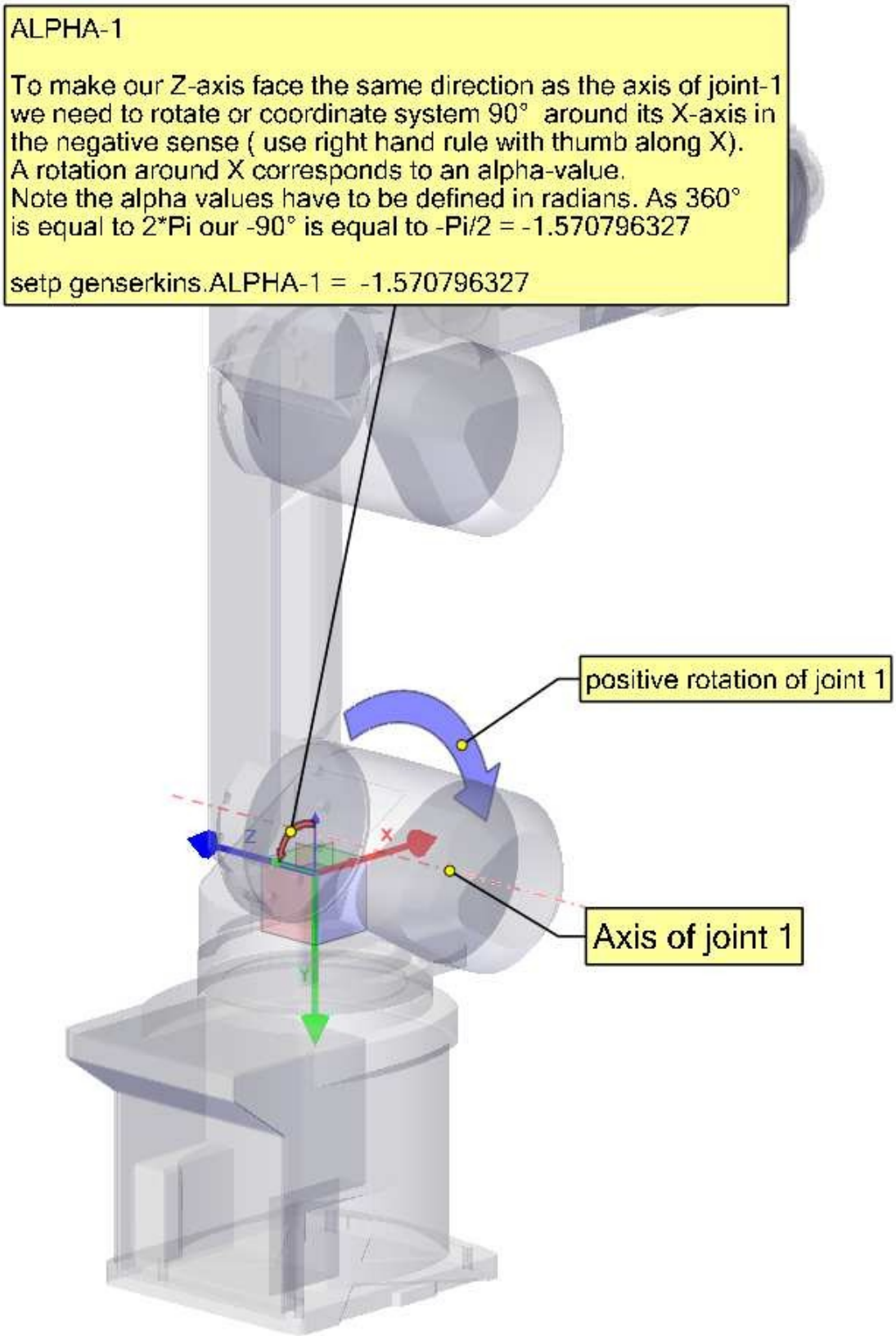
Wenn die nächste Gelenkachse parallel zur letzten ist, kann man den Wert für den d-Parameter beliebig wählen, aber es hat keinen Sinn, ihn anders als 0 zu setzen.

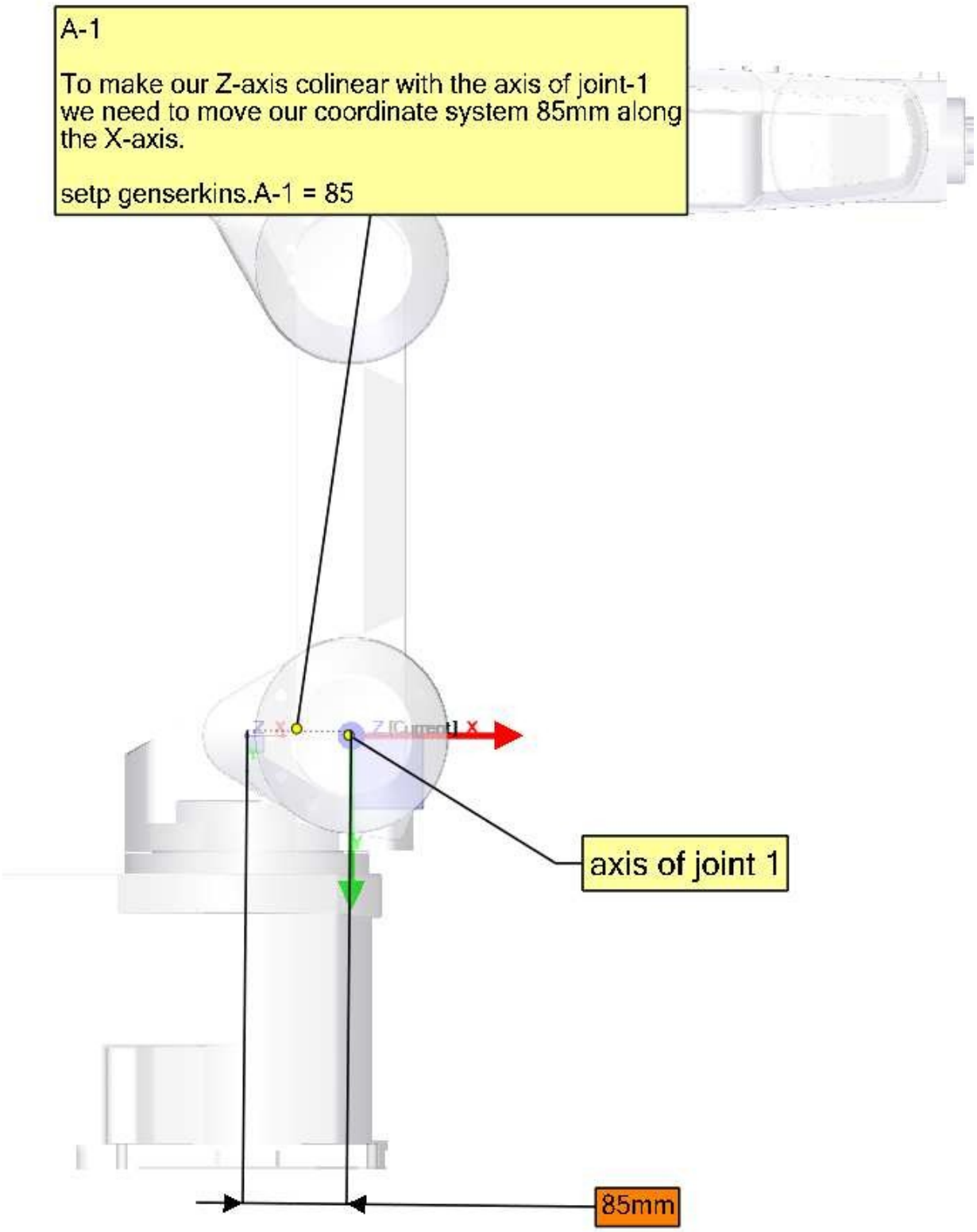
9.2.8 Detailliertes Beispiel (RV-6SL)

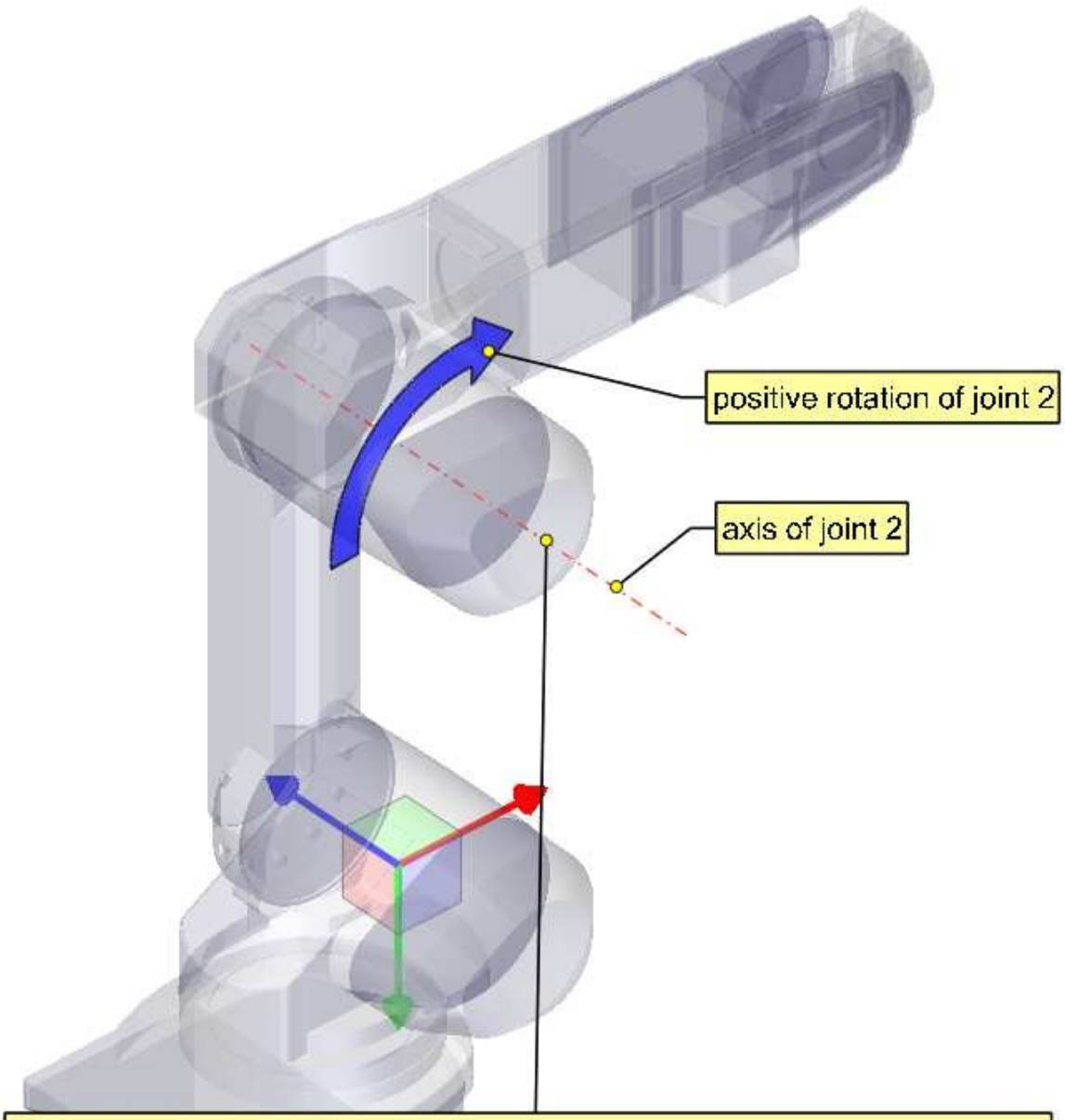
Im Folgenden wird eine Methode beschrieben, wie man die erforderlichen ”modifizierten DH-Parameter” für einen Mitsubishi RV-6SDL ableitet und wie man die Parameter in der HAL-Datei einstellt, um sie mit der ”genserkins”-Kinematik in LinuxCNC zu verwenden. Die erforderlichen Abmessungen werden am besten 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
Y
X

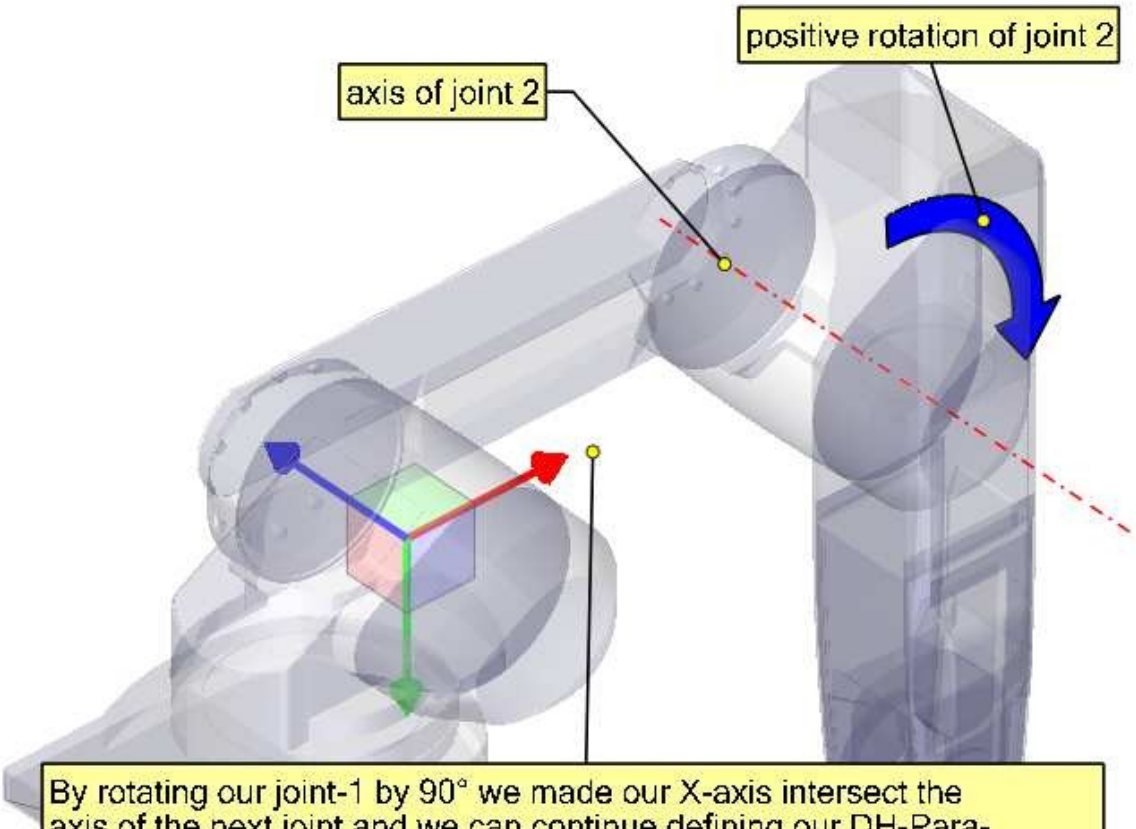

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Axis of Joint 1
350mm
z
x


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

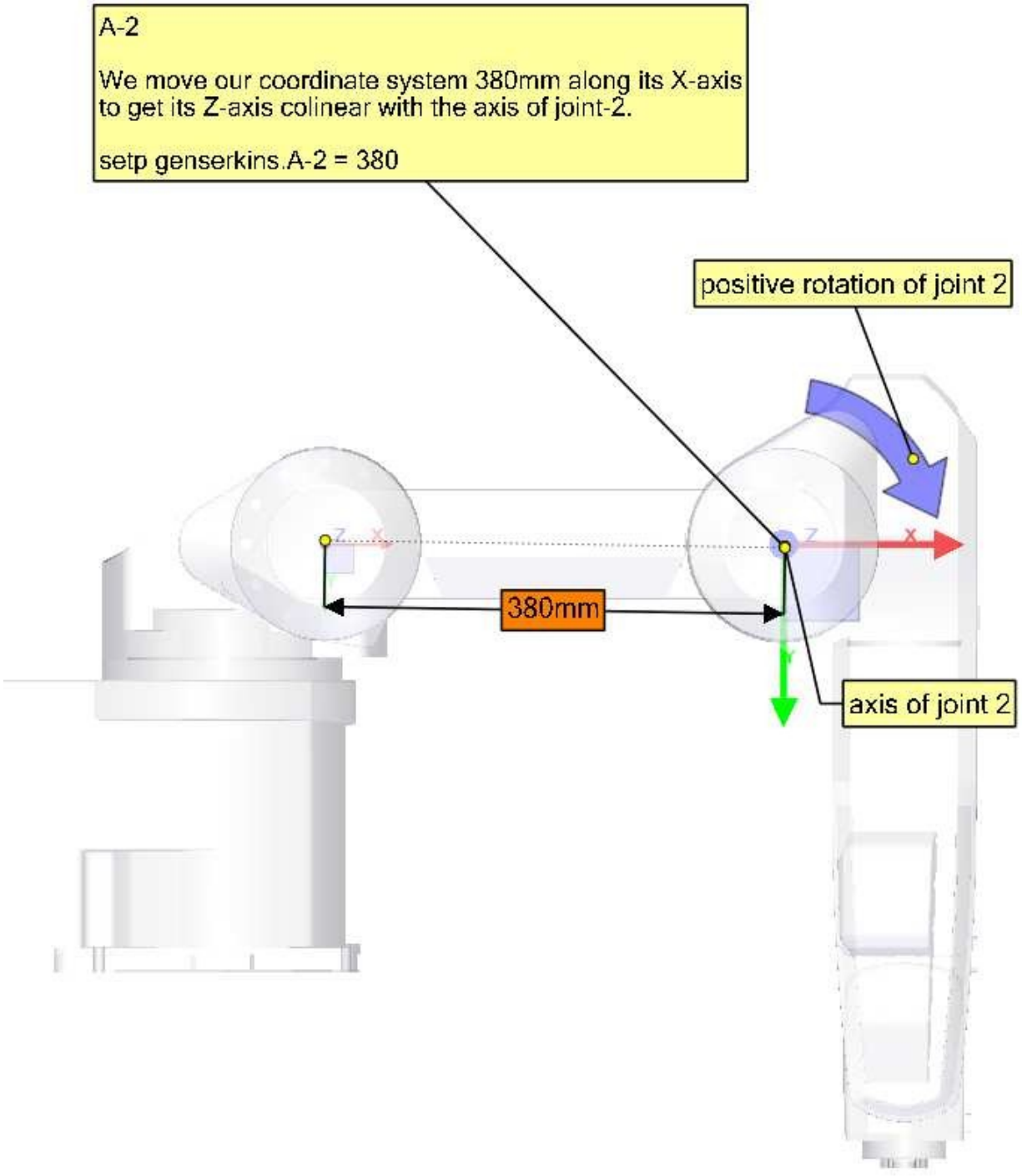
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

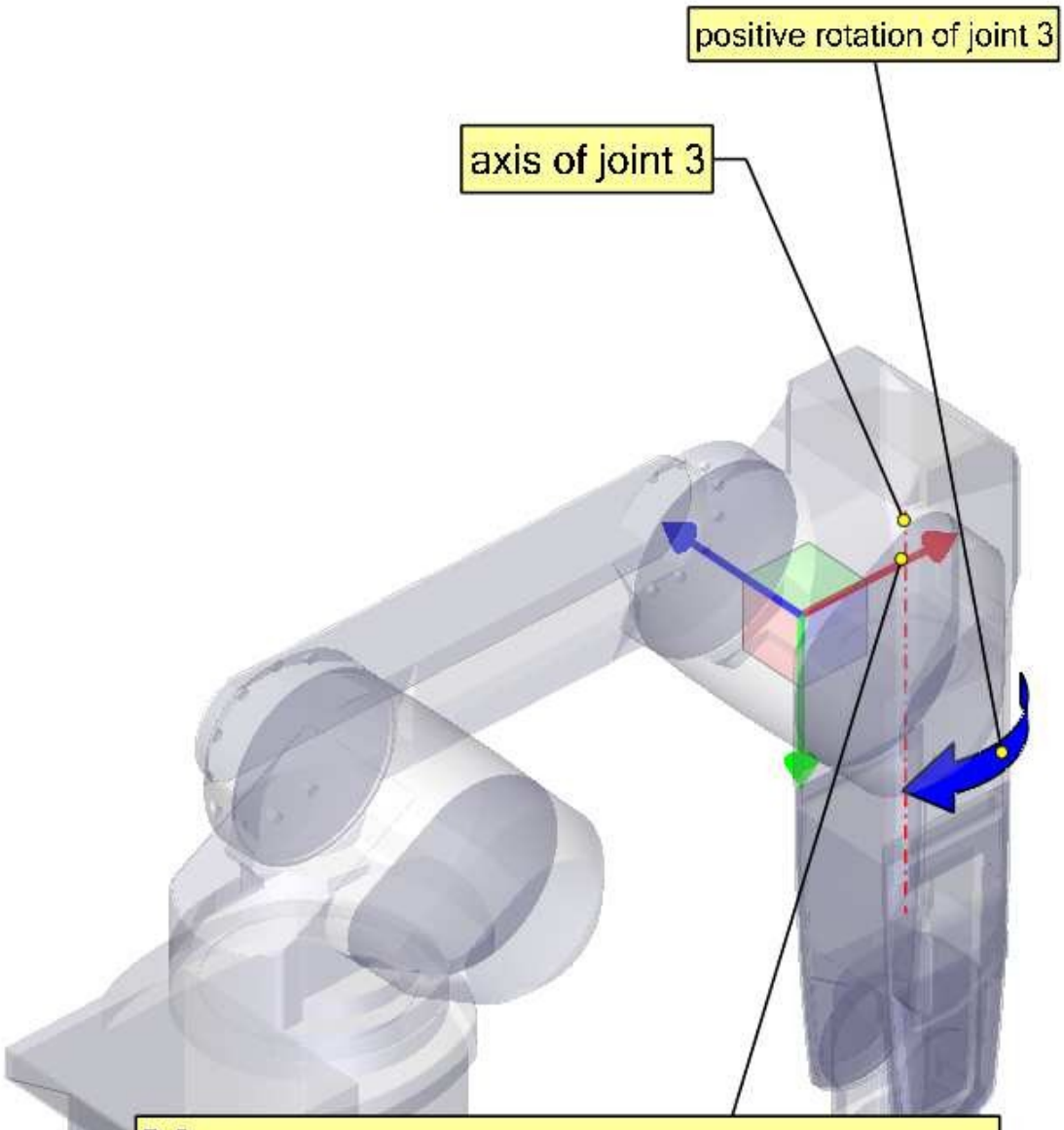
setp genserkins.D-1 = 0

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

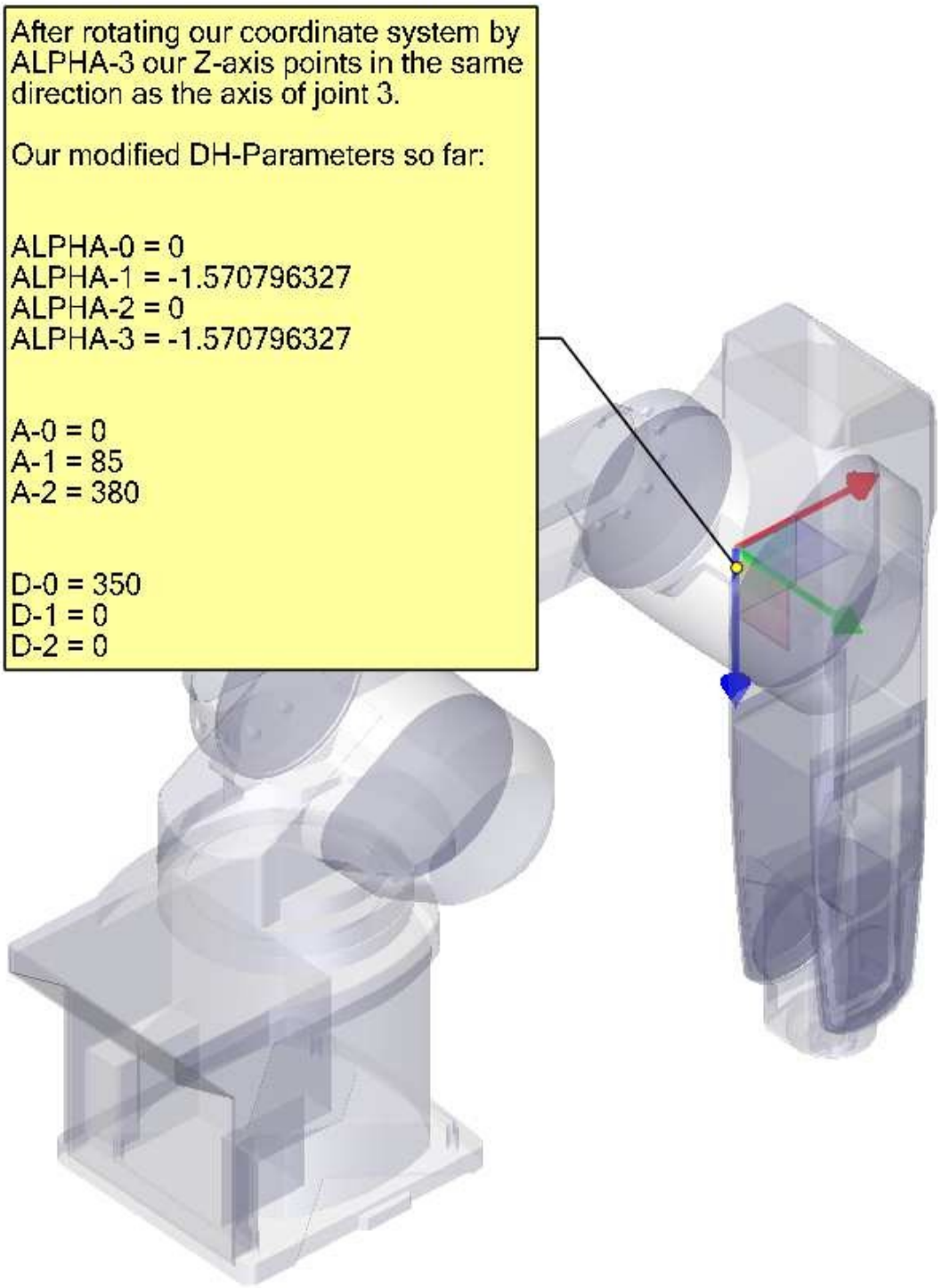
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

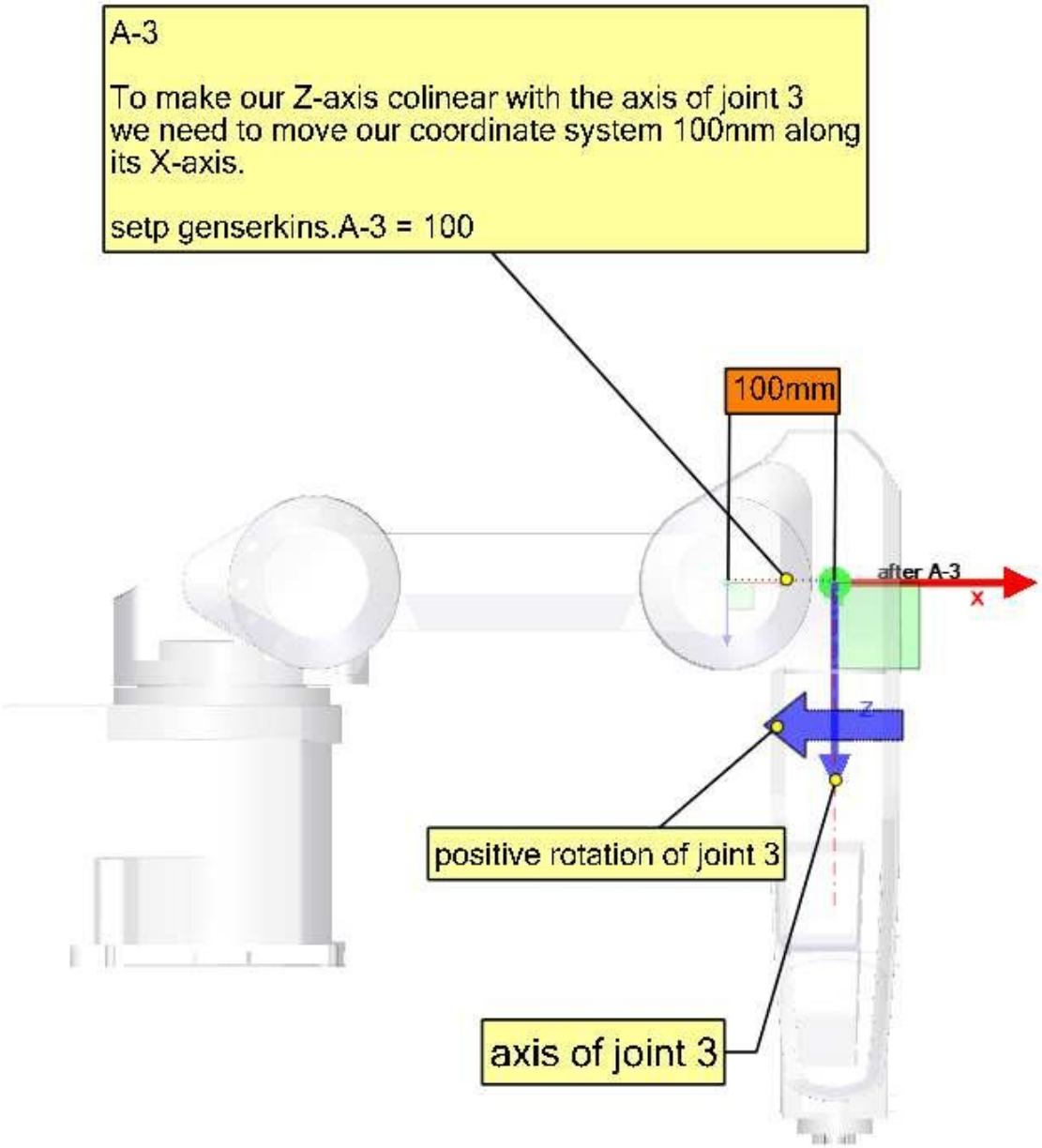
setp genserkins.D-2 = 0

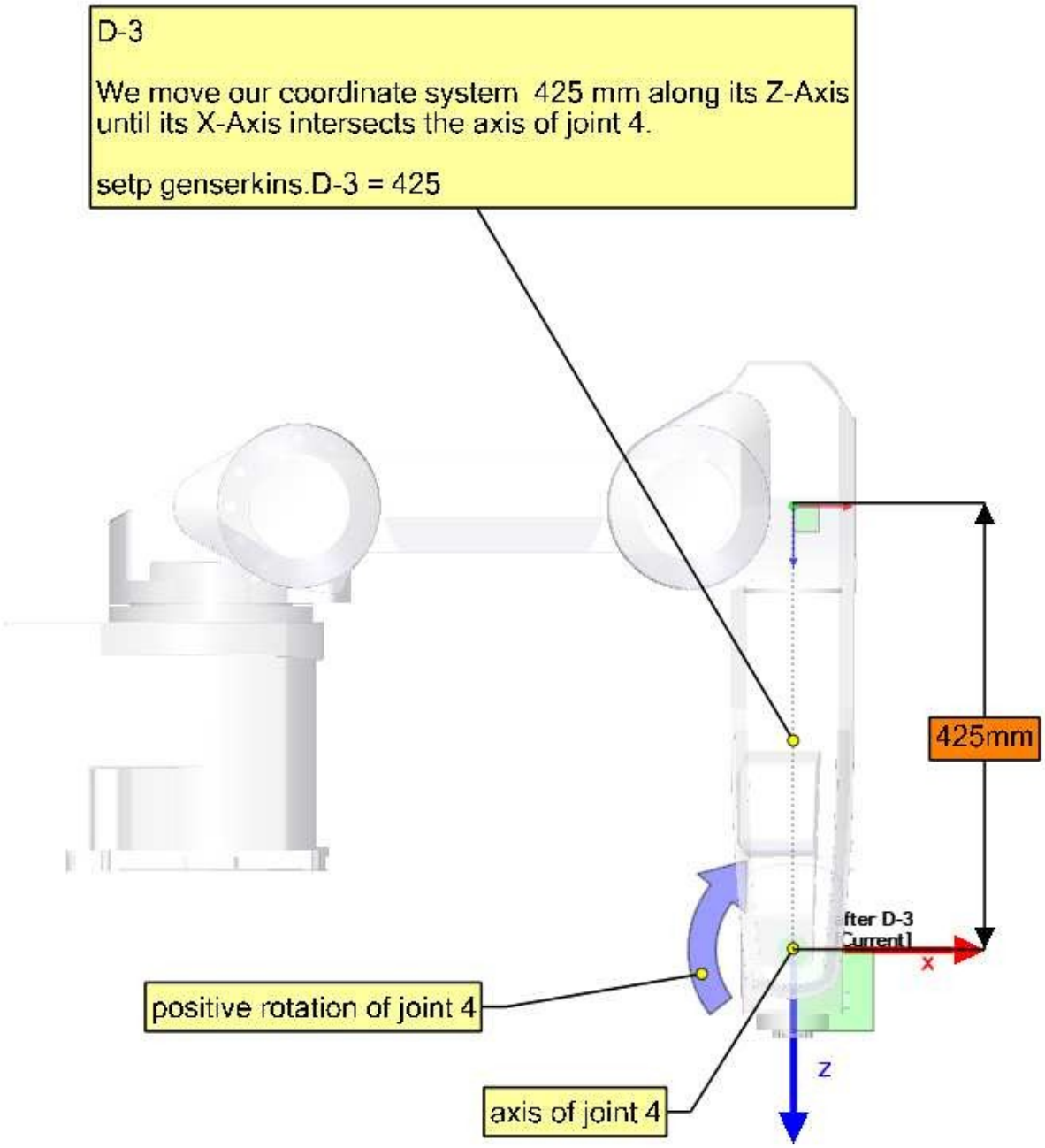
ALPHA-3

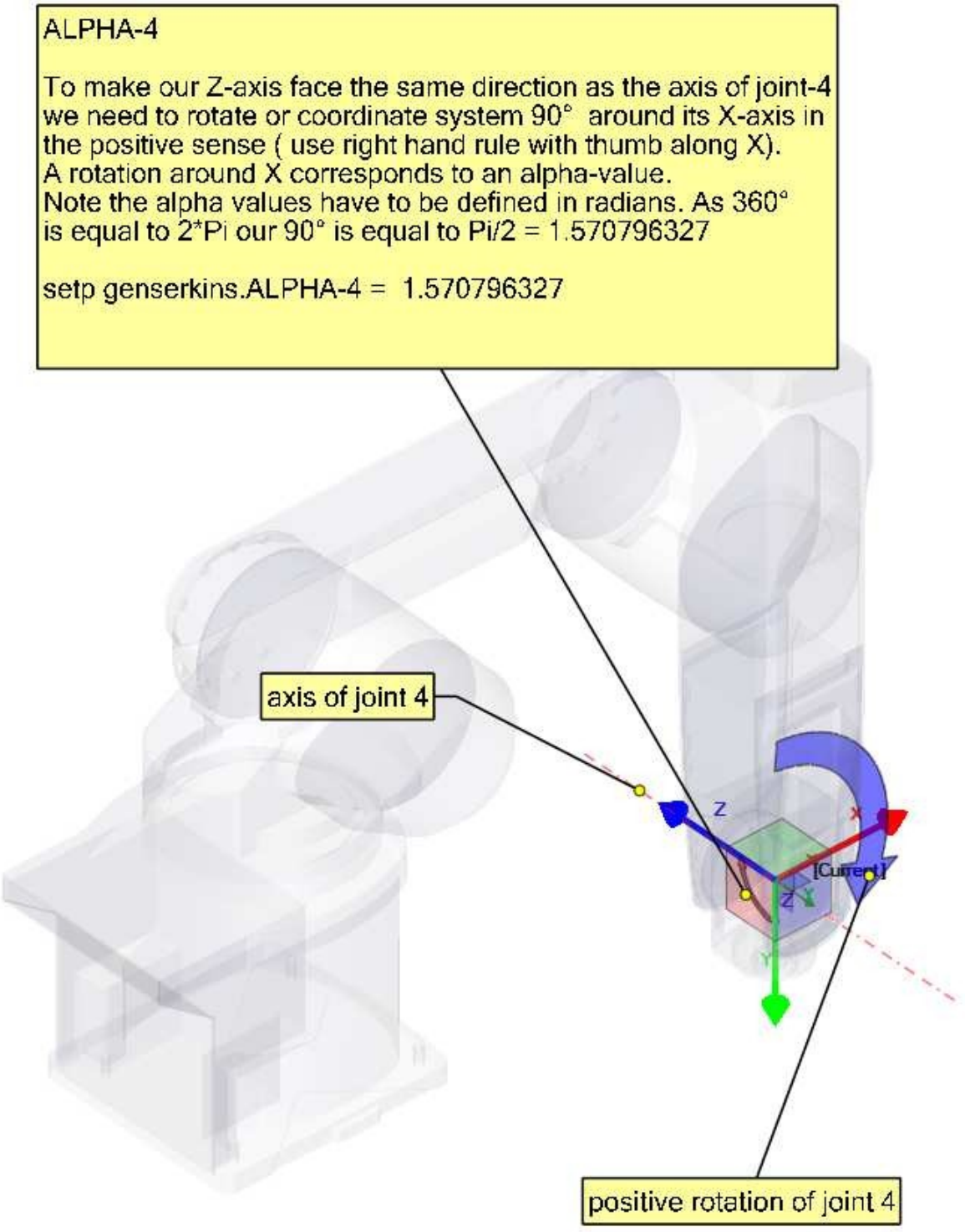
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense (use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

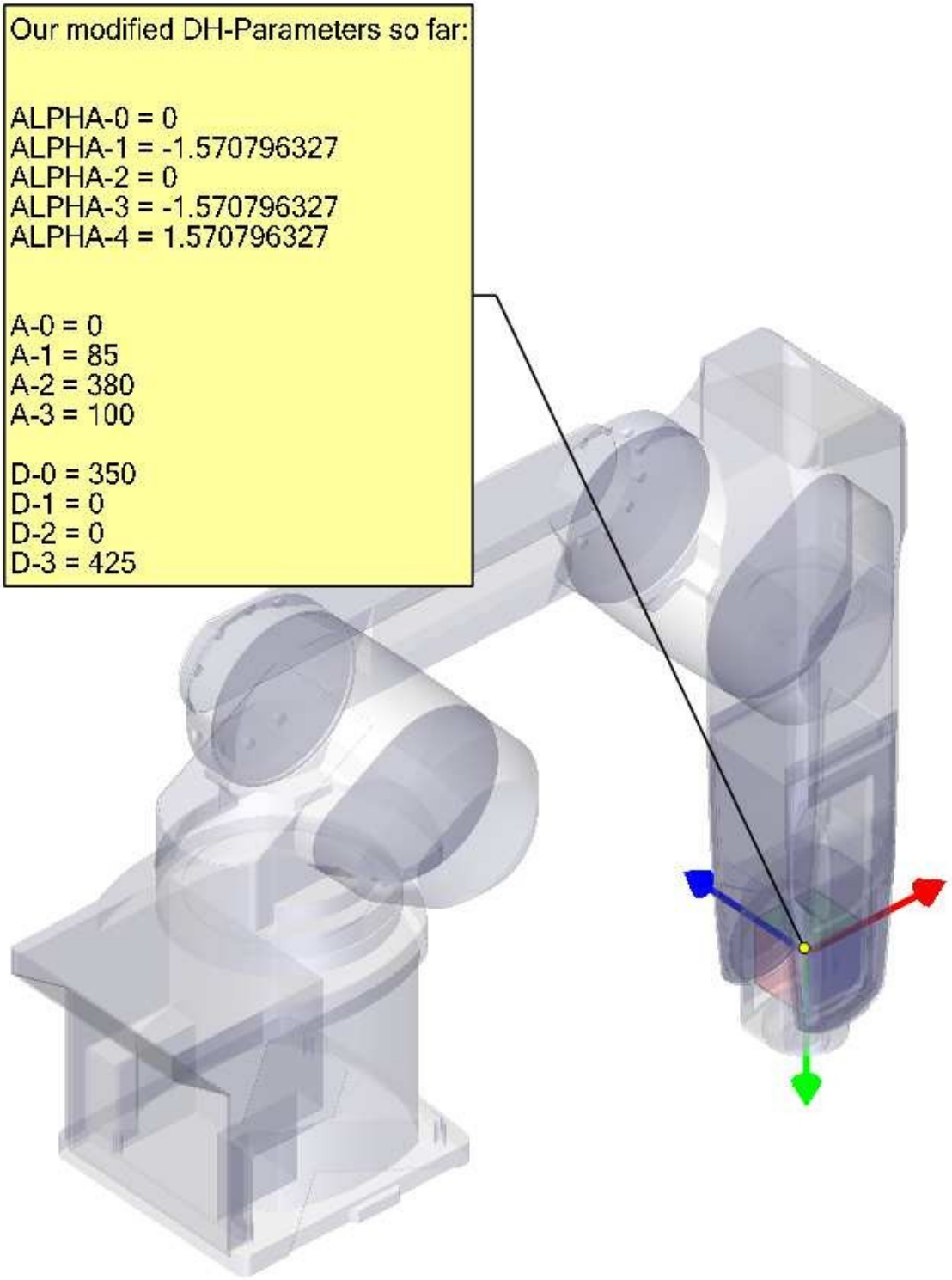
setp genserkins.ALPHA-3 = -1.570796327

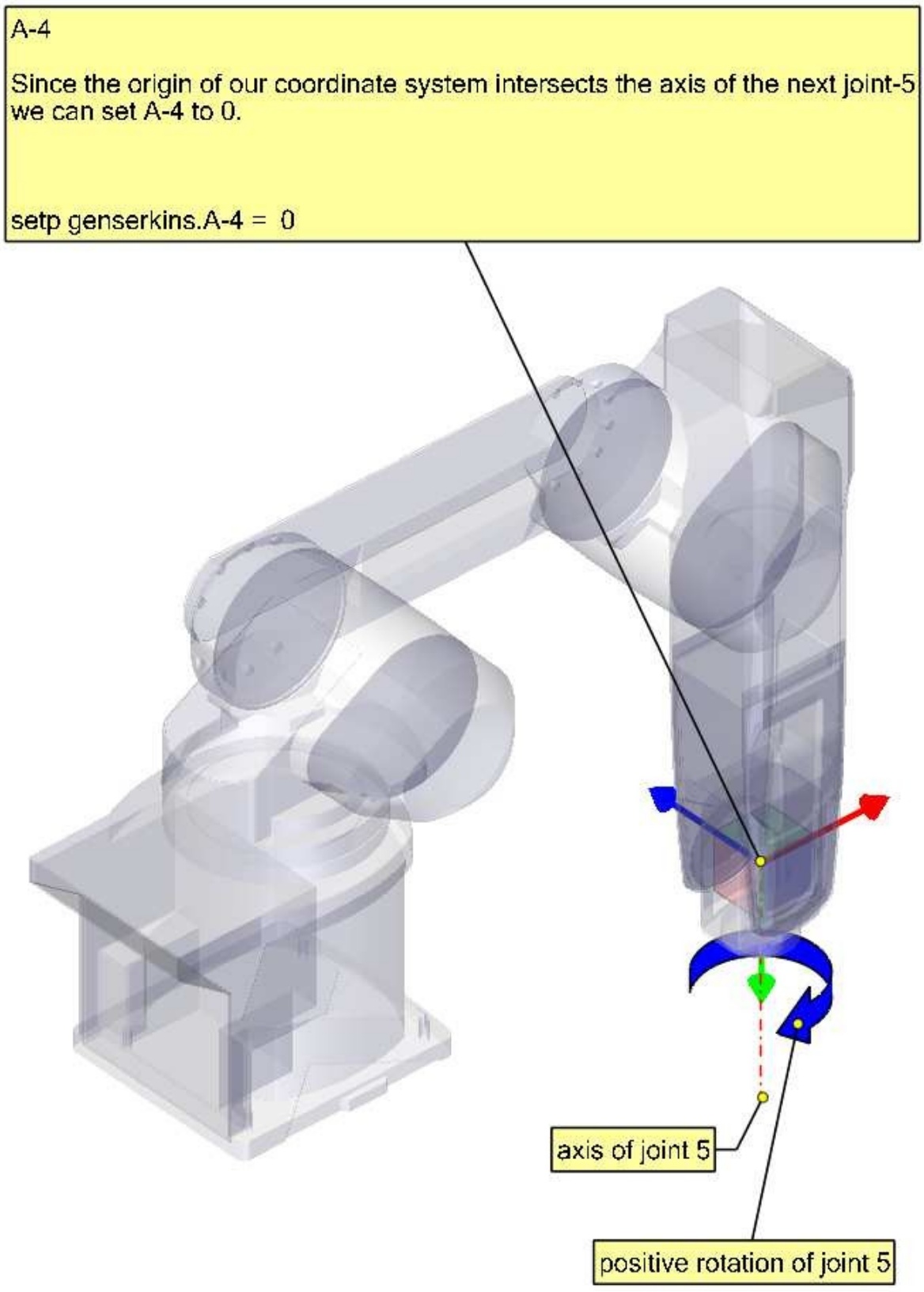

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


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
Z
X
Y
[Current]


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

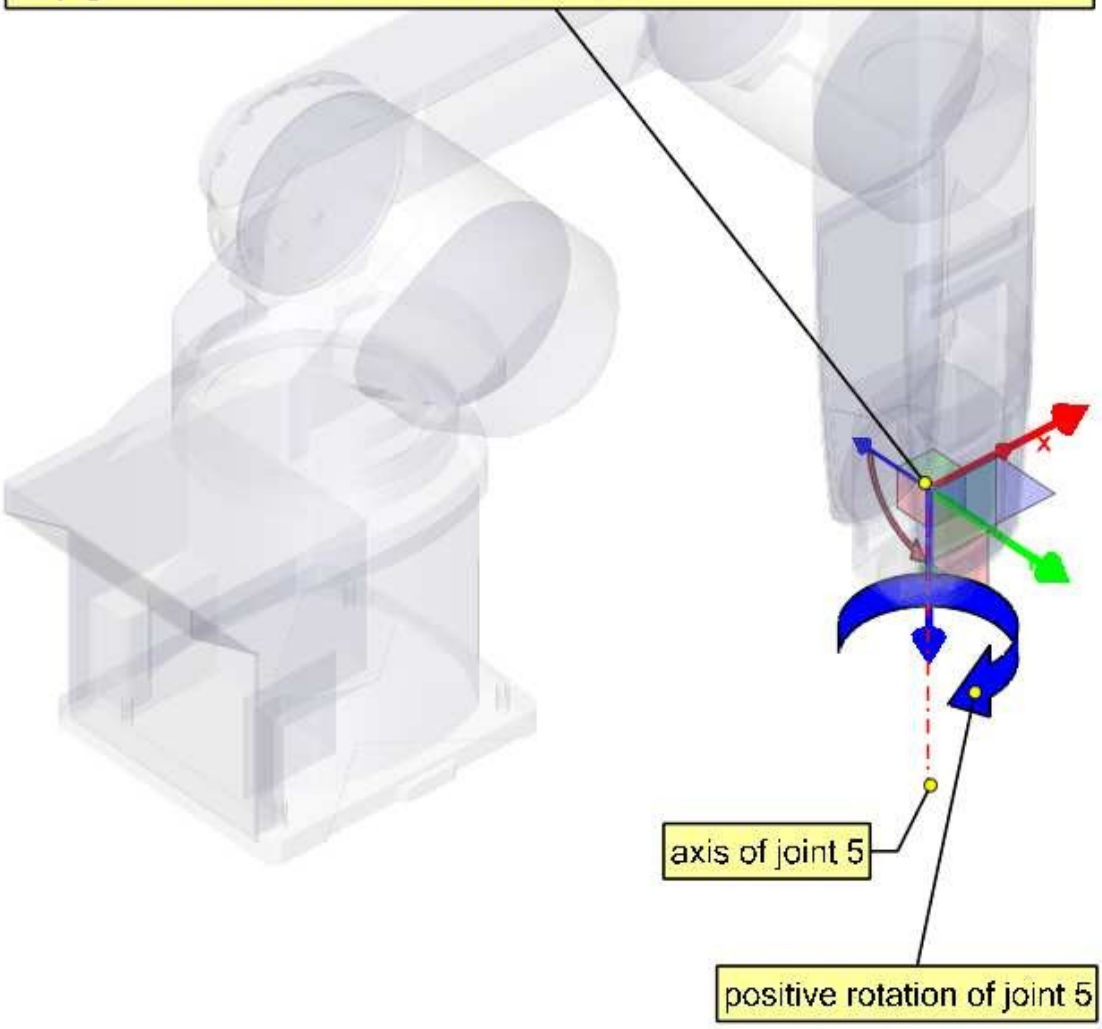
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

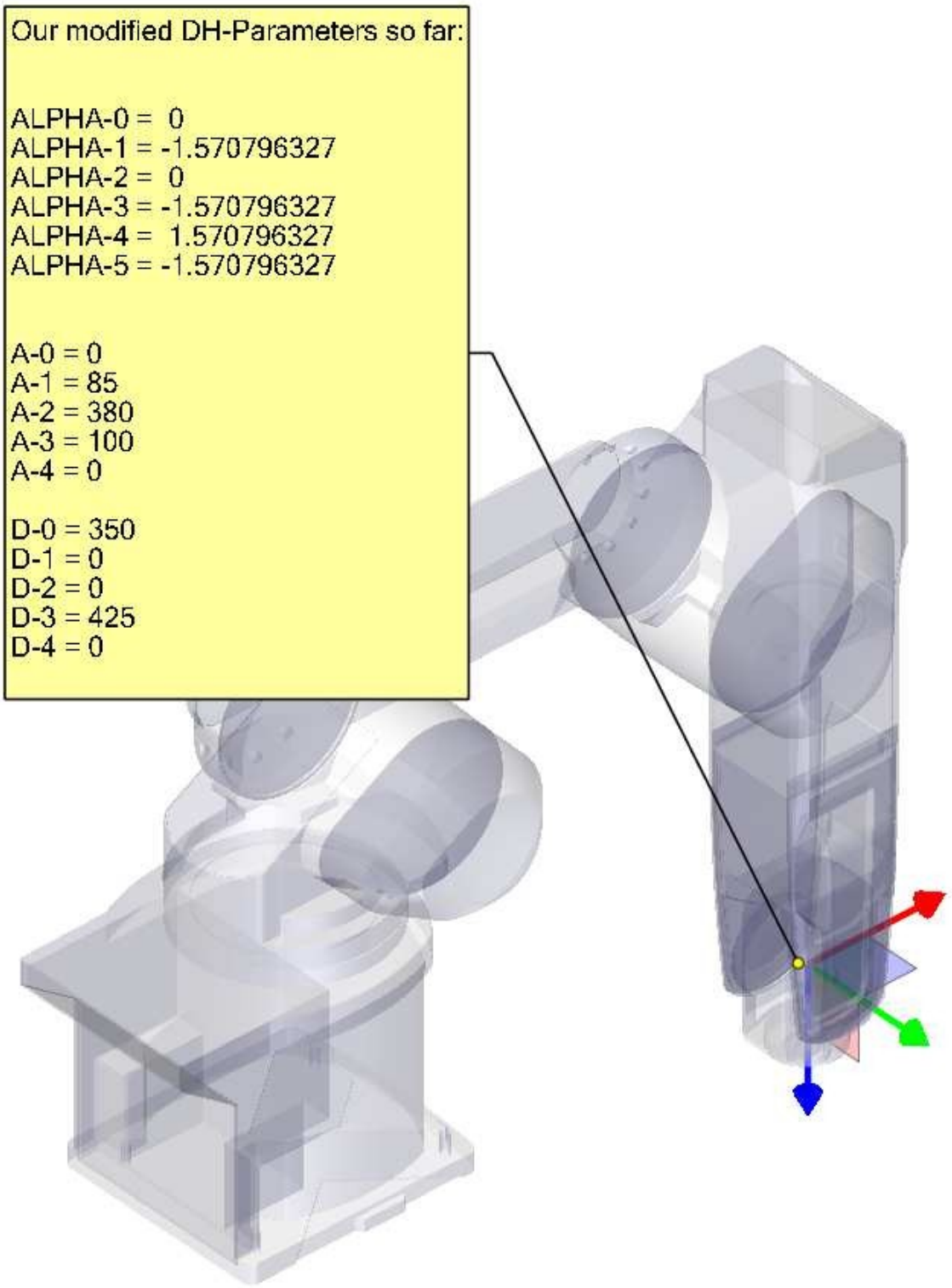
setp genserkins.D-4 = 0

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



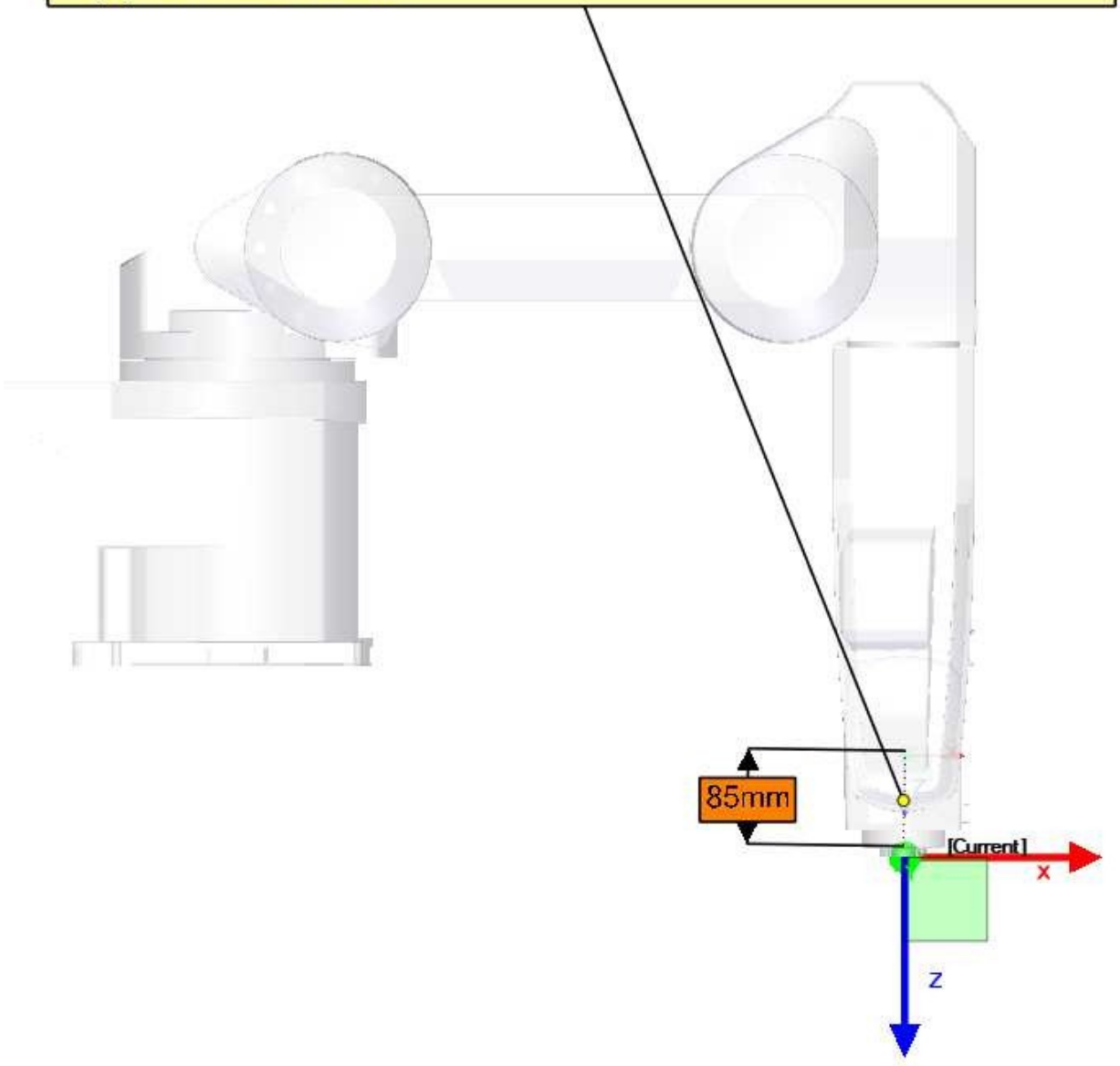

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0

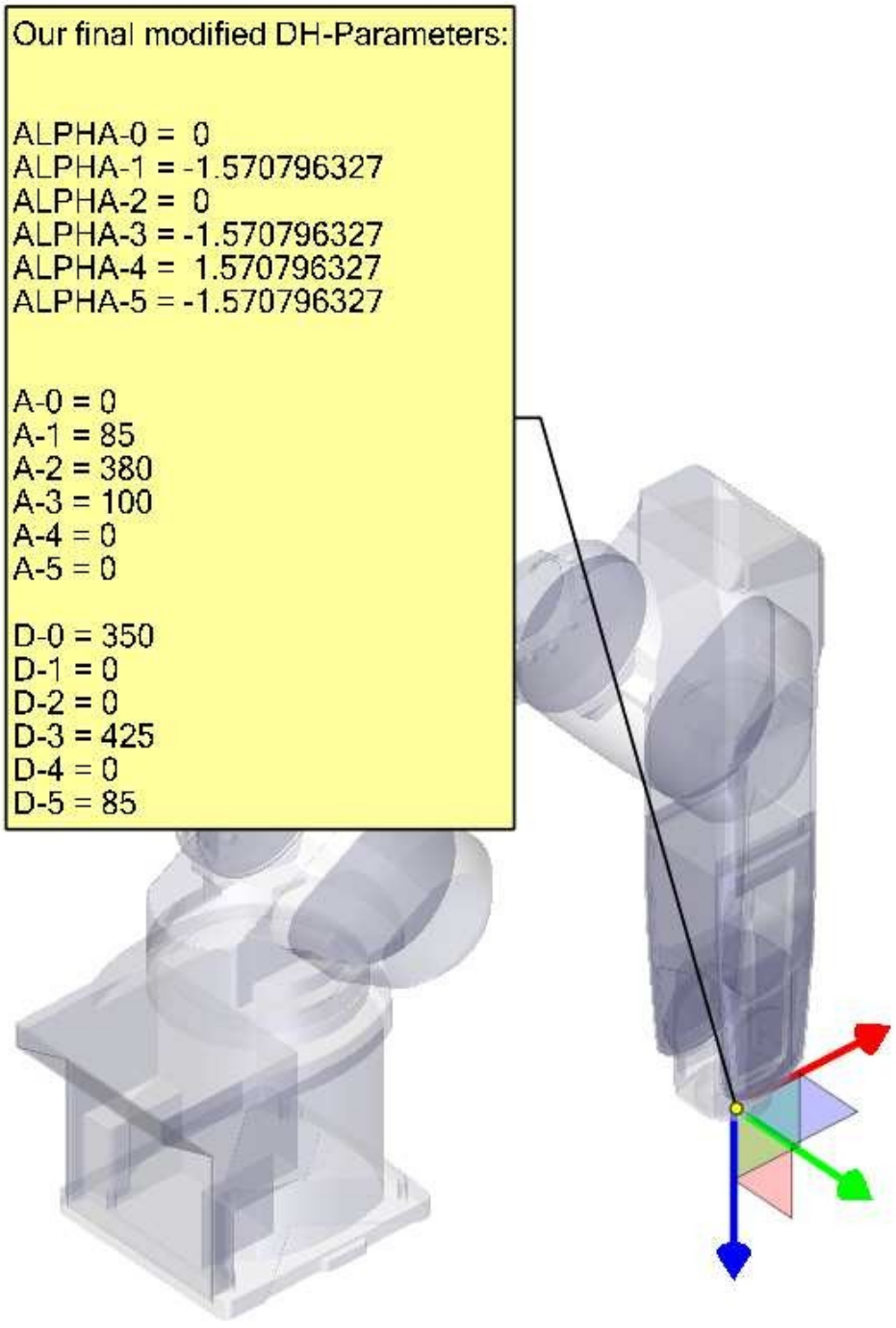
D-5

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




Our final modified DH-Parameters:
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
A-5 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0
D-5 = 85

9.2.9 Danksagungen

Vielen Dank an den Benutzer Aciera für den gesamten Text und die Grafiken für den RV-6SL-Roboter!

9.3 5-Achsen-Kinematik

9.3.1 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.

9.3.2 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.

9.3.3 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:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \quad \text{orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \quad \text{position vector} \qquad (1)$$

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.

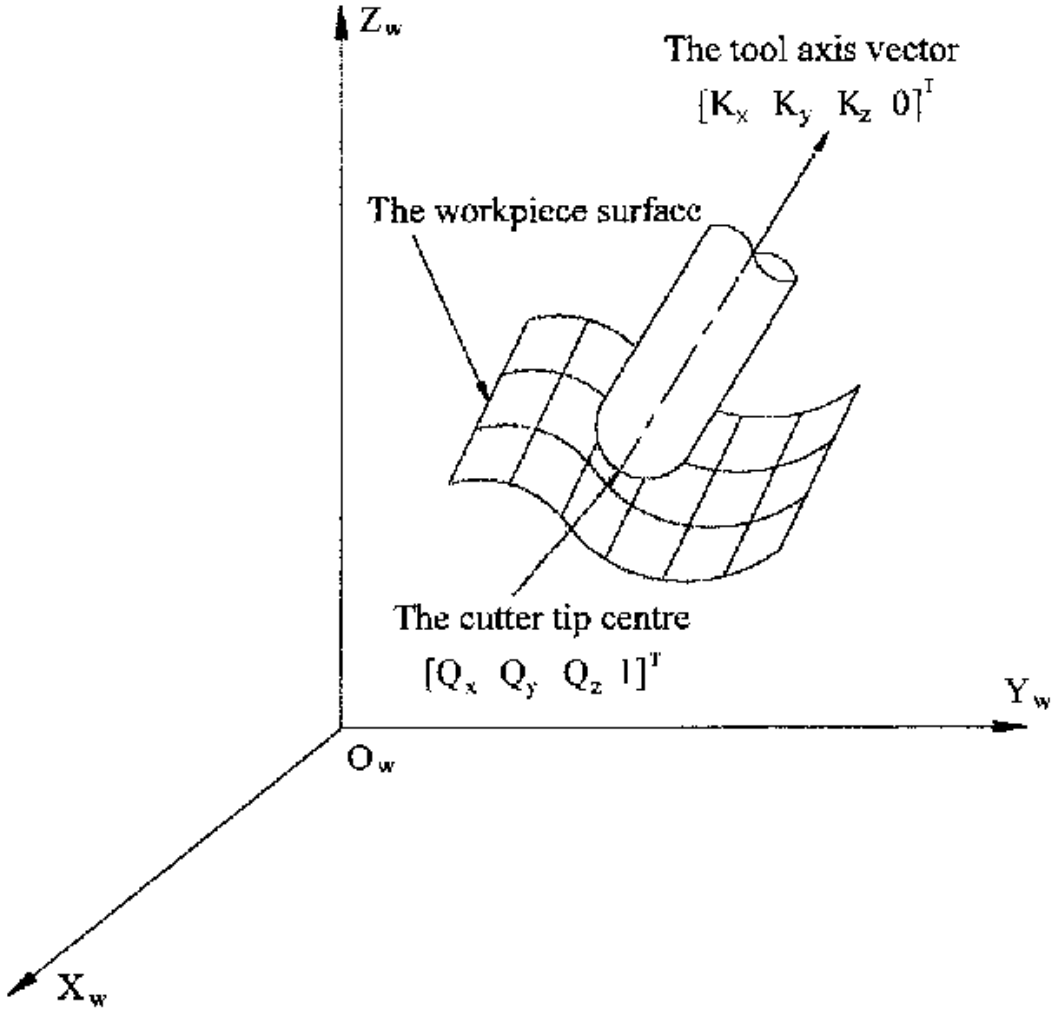


Abbildung 9.2: Standortdaten des Fräsers

9.3.4 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:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

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.

9.3.5 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.

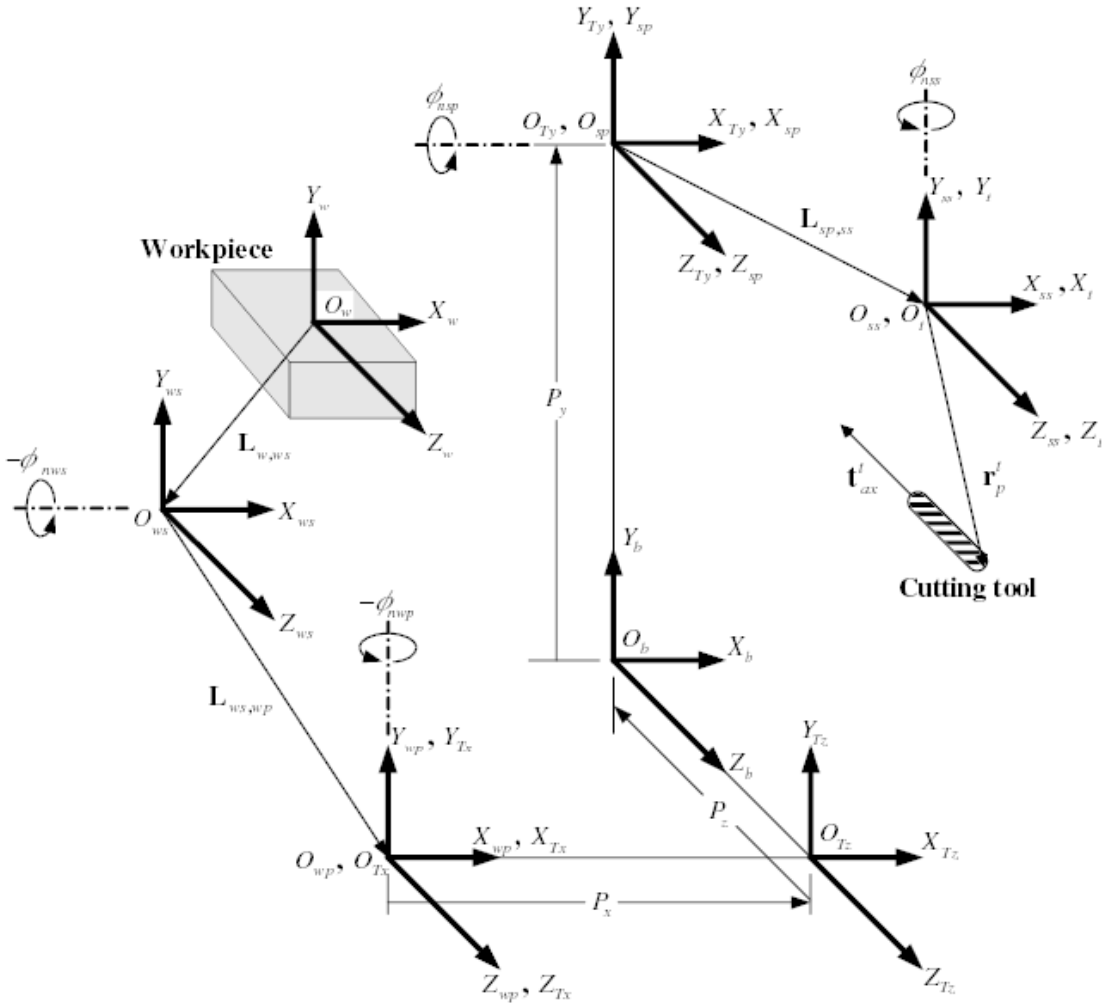


Abbildung 9.3: Allgemeine Konfiguration und Koordinatensysteme

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.

9.3.5.1 Transformationen für eine xyzac-trt-Werkzeugmaschine mit Werkstückversatz

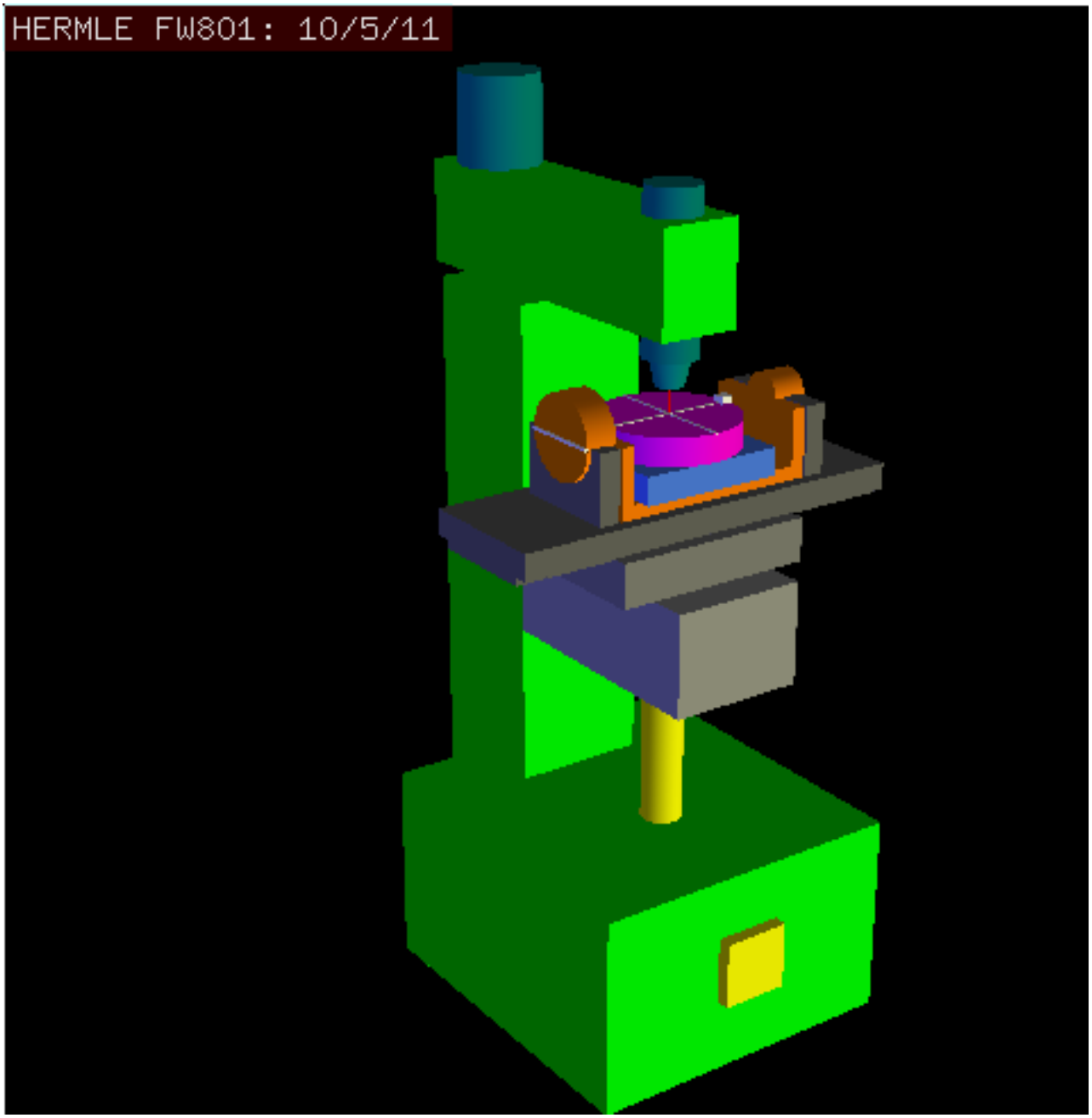


Abbildung 9.4: vismach-Modell von xyzac-trt mit übereinstimmenden Drehachsen

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.

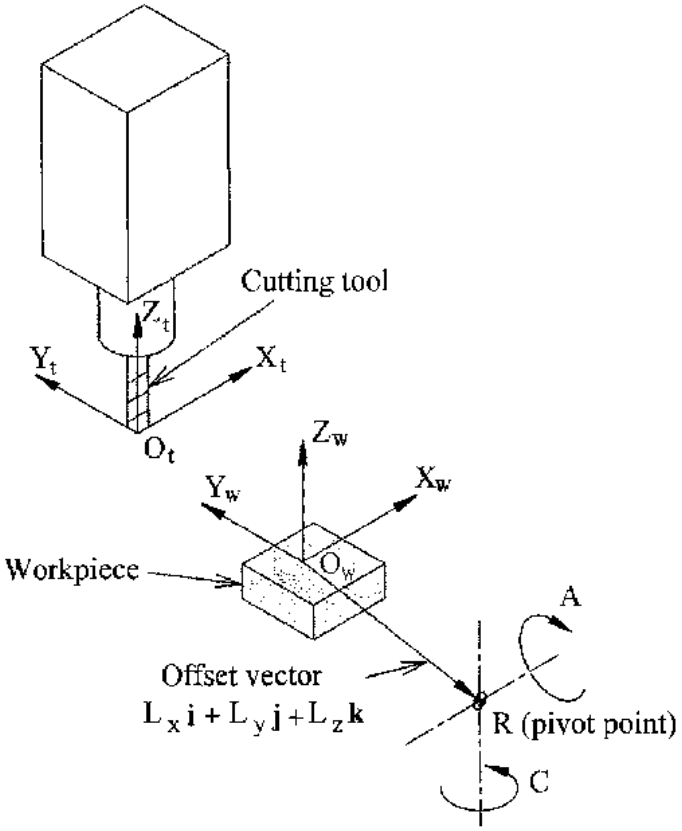


Abbildung 9.5: Kipp-/Drehkonfiguration des Tisches

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.

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:

$$ \begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17) $$

9.3.5.2 Transformationen für eine xyzac-trt-Maschine mit Drehachsenverschiebungen

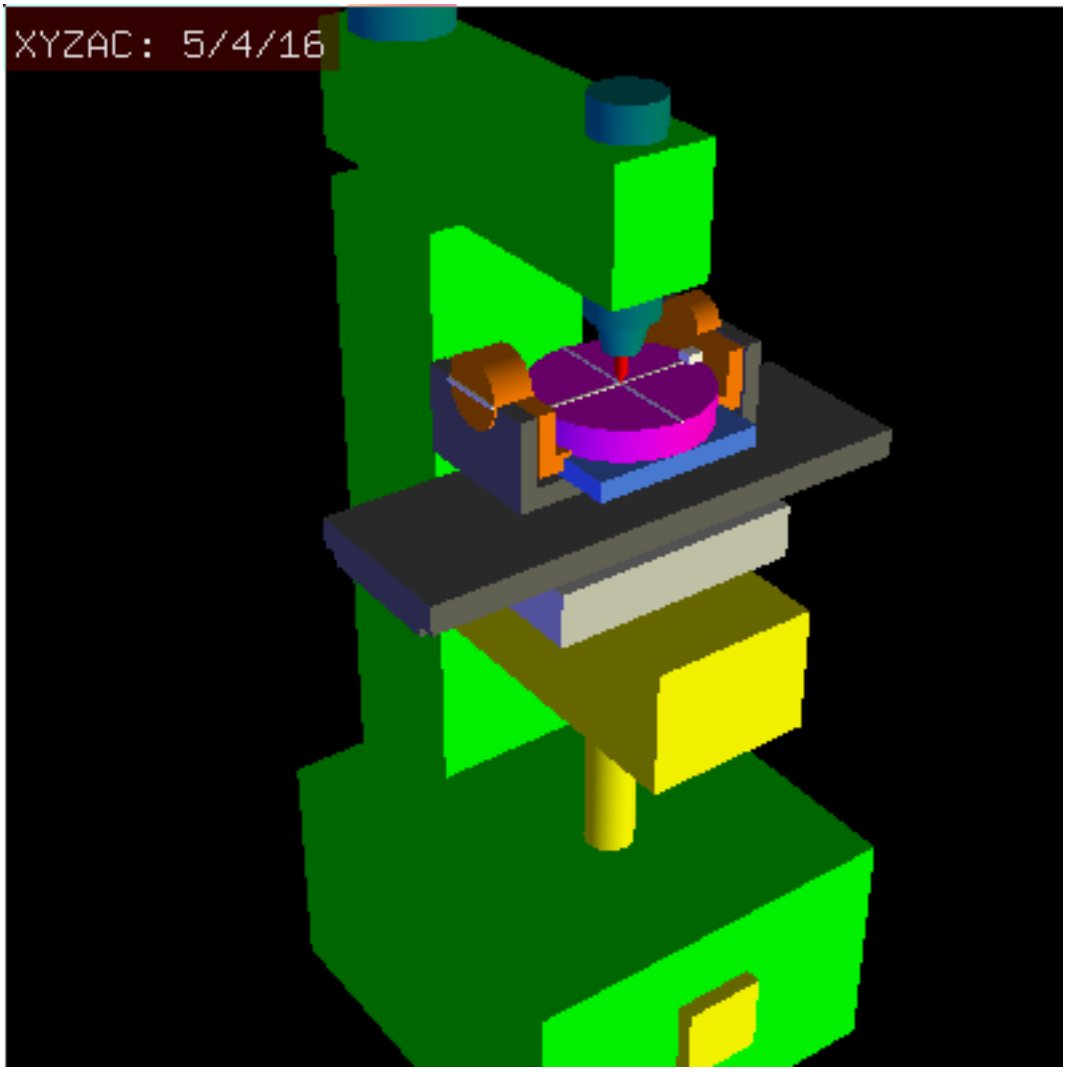


Abbildung 9.6: Vismach-Modell von xyzac-trt mit Rotationsachsenversatz (positiv)

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.

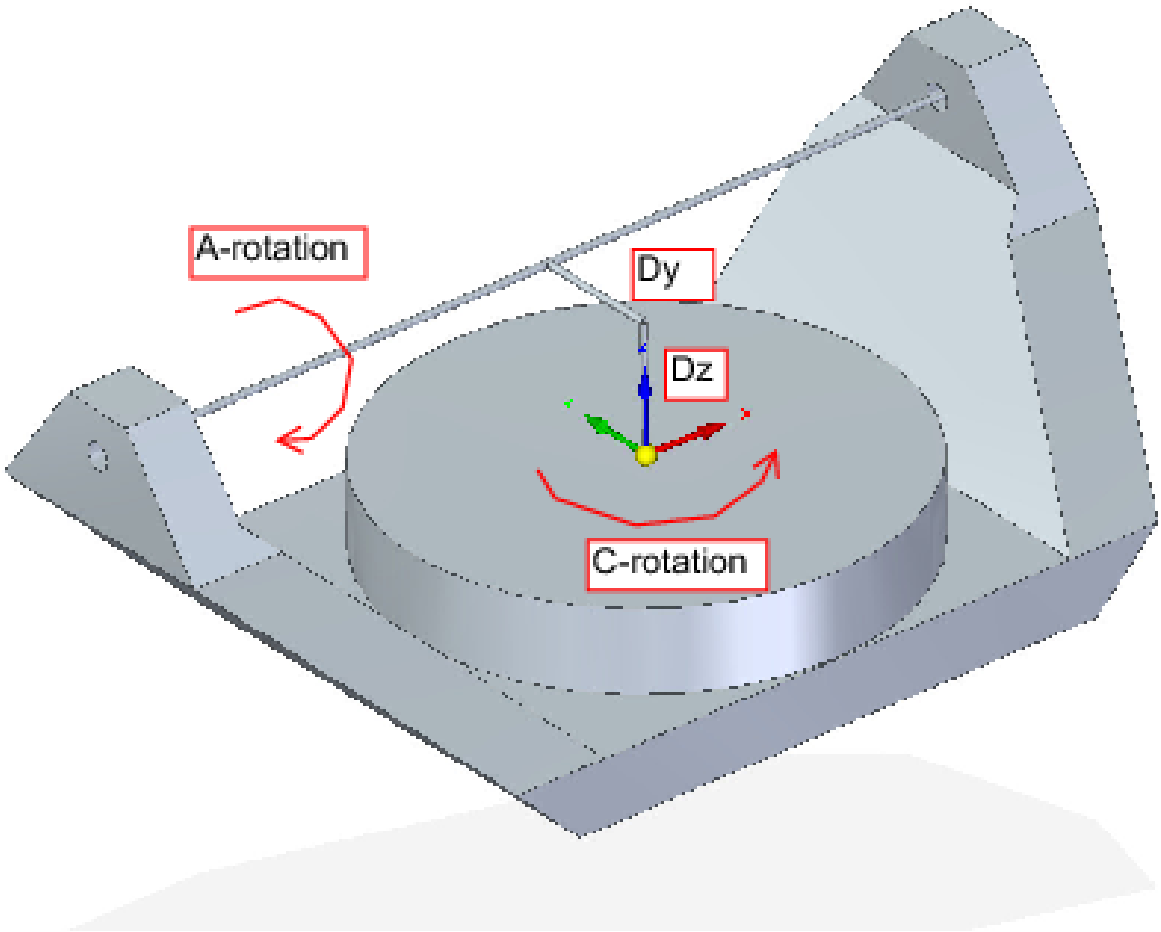


Abbildung 9.7: Kipp-/Drehkonfiguration des Tisches xyzac-trt, mit Achsenversatz

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.

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:

$$ \begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28) $$

9.3.5.3 Transformationen für eine xyzbc-trt-Maschine mit Drehachsenverschiebungen

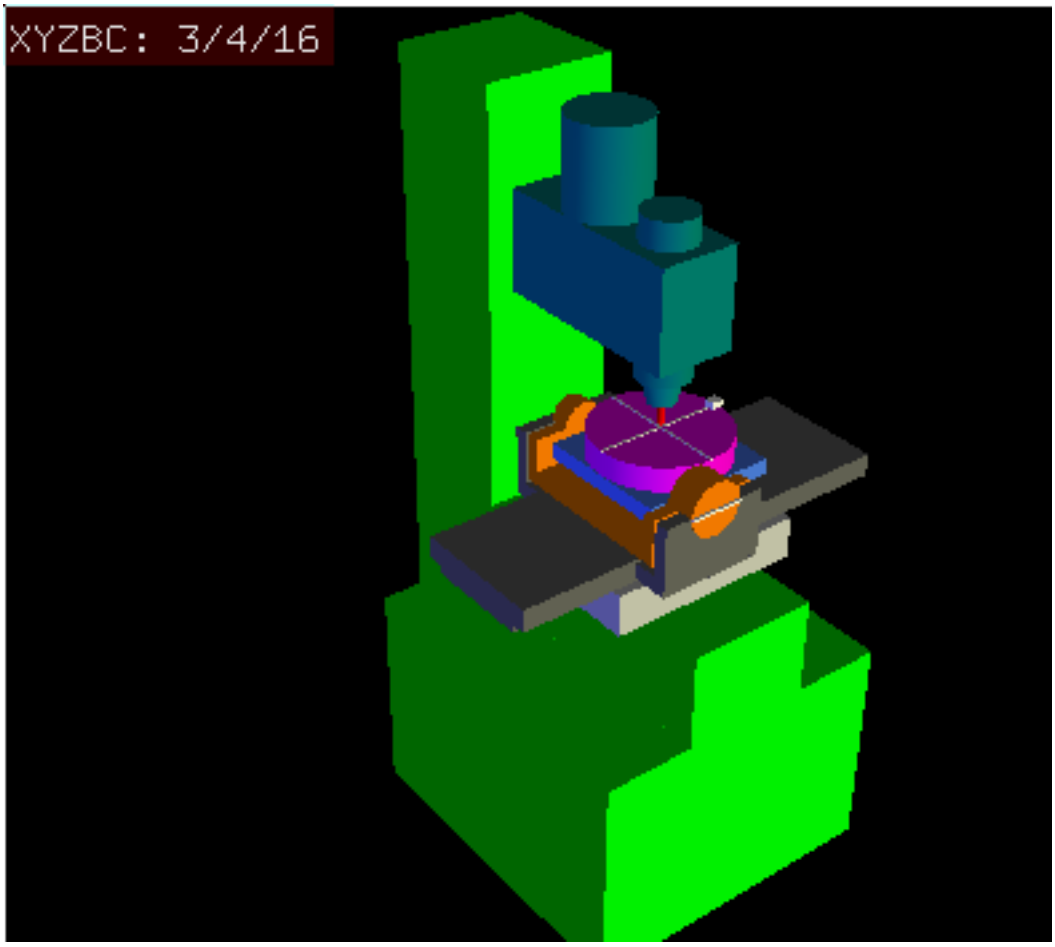


Abbildung 9.8: Vismach-Modell von xyzbc-trt mit Rotationsachsenversatz (negativ)

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.

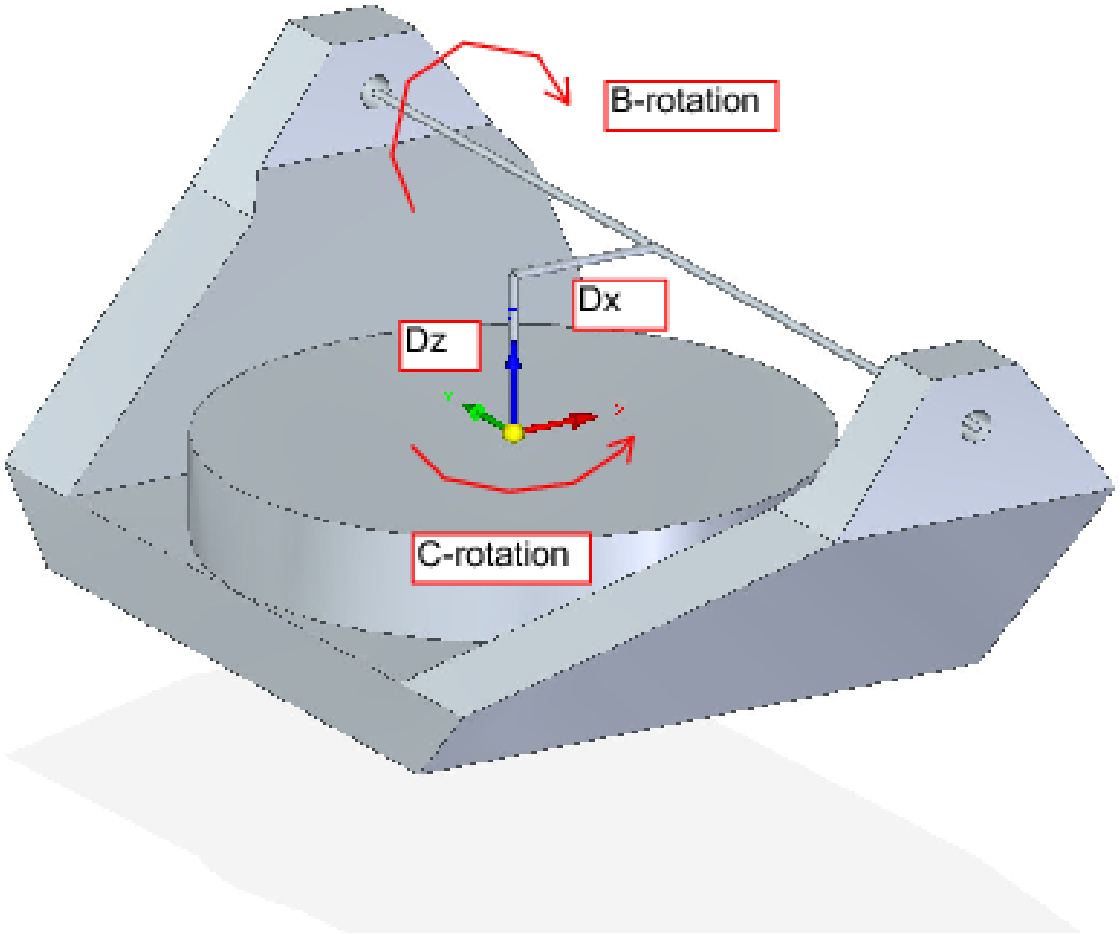


Abbildung 9.9: Kipp-/Drehkonfiguration des Tisches xyzbc-trt, mit Achsenversatz

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.

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:

$$ \begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \qquad (39) $$

9.3.6 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/*).

The example configurations include the required INI files and an examples subdirectory with G-code (NGC) files. These sim configurations invoke a realistic 3-dimensional model using the LinuxCNC vismach facility.

9.3.6.1 Vismach Simulationsmodelle

Vismach is a library of python routines to display a dynamic simulation of a CNC machine on the PC screen. The python script for a particular machine is loaded in HAL and data passed by HAL pin connections. The user-space vismach model is loaded by a HAL command like:

```
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
...
```

See the simulation INI files for details of the HAL connections used for the vismach model.

9.3.6.2 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
```

9.3.7 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**)

Once it is compiled and installed you can reference it in your config setup of your machine. This is done in the INI file of your config directory. For example, the common INI specificaion:

```
[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
```

9.3.8 Abbildungen

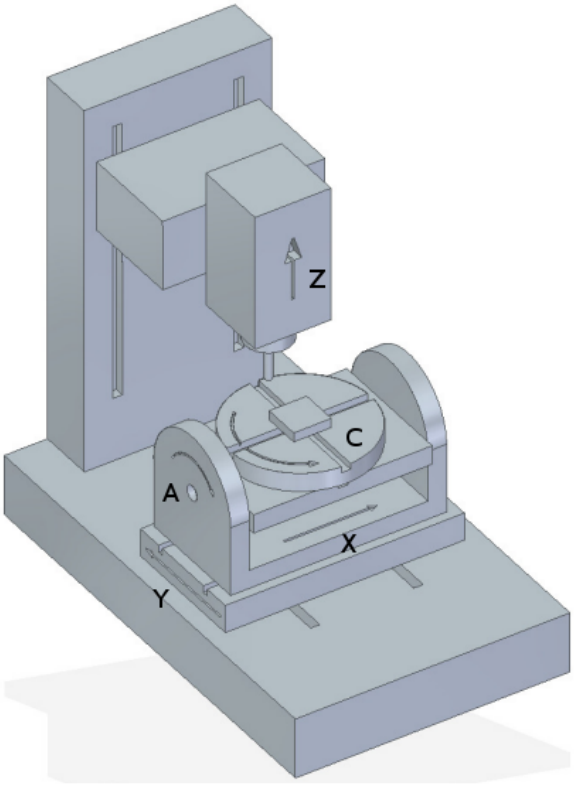


Abbildung 9.10: Kipp-/Drehkonfiguration des Tisches

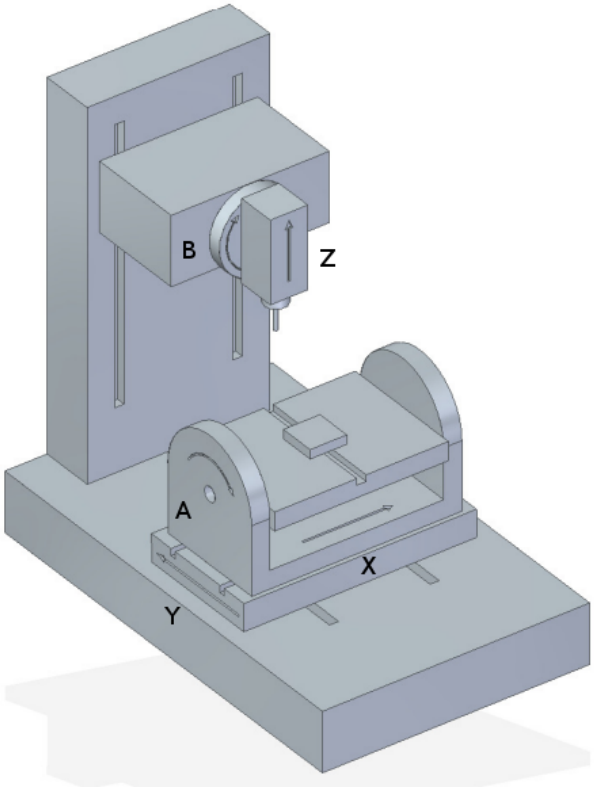


Abbildung 9.11: Spindel-/Tischkippkonfiguration

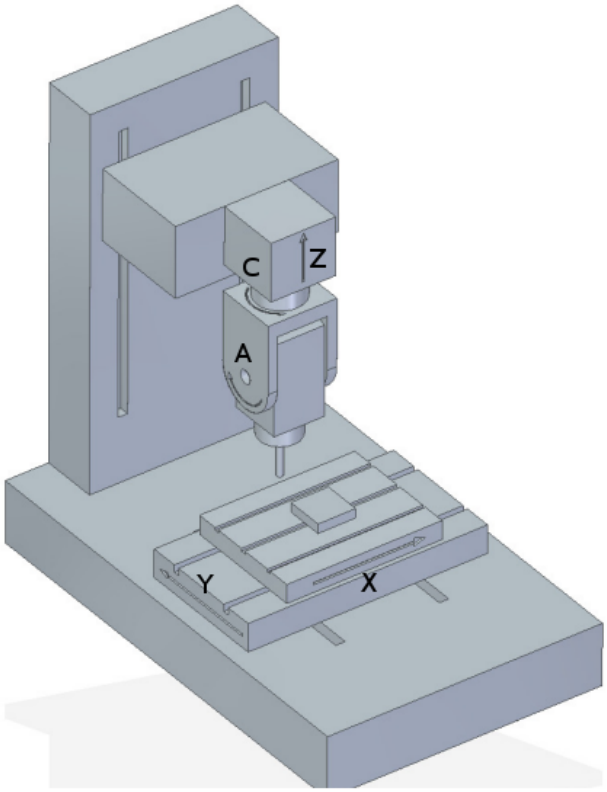


Abbildung 9.12: Kipp-/Drehkonfiguration der Spindel

9.3.9 VERWEISE

1. 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.
2. 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.
3. 3D 6-DOF Serial Arm Robot Kinematics, RJ du Preez, SA-CNC-CLUB, Dec. 5, 2013.
4. 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.

9.4 Schaltbare Kinematik (switchkins)

9.4.1 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.

Die Auswahl des Kinematik-Typs erfolgt über einen Motion-Modul-HAL-Pin, der über ein G-Code-Programm oder über interaktive MDI-Befehle aktualisiert werden kann. Die halui-Bestimmungen für die Aktivierung von MDI-Befehlen können verwendet werden, um die Auswahl des Kinematik-Typs über Hardware-Steuerungen oder ein virtuelles Panel (PyVCP, GladeVCP, etc.) zu ermöglichen.

Wenn eine Kinematik Typ geändert wird, muss der G-Code auch Befehle zu **zwingen Synchronisation** des Interpreters und Bewegung Teile von LinuxCNC. Typischerweise wird ein HAL-Pin *read* Befehl (M66 E0 L0) unmittelbar nach der Änderung des steuernden HAL-Pins verwendet, um die Synchronisation zu erzwingen.

9.4.2 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. **scarakins** (type0:scarakins type1:identity)
7. **5axiskins** (type0:5axiskins type1:identity) (bridgemill)

The xyz[ab]c-trt-kins modules by default use type0==xyz[ab]c-trt-kins for backwards compatibility. The provided sim configs alter the type0/type1 convention by forcing type0==identity kinematics using the module string parameter *sparm* with an INI file setting like:

```
[KINS]
KINEMATICS = xyzac-trt-kins sparm=identityfirst
...
```

9.4.2.1 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
```

Anmerkung
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.

9.4.2.2 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.

9.4.3 HAL-Pins

Kinematics switching is controlled by the motion module input HAL pin **motion.switchkins-type**. The floating point pin value is truncated to integer and used to select one of the provided kinematics types. The zero startup value selects the type0 default kinematics type.

Anmerkung
The motion.switchkins-type input pin is floating point in order to facilitate connections to motion module output pins like motion.analog-out-0n that are controllable by standard M-codes (typically M68EnL0).

Output HAL pins are provided to inform GUIs of the current kinematics type. These pins can also be connected to digital inputs that are read by G-code programs to enable or disable program behavior in accordance with the active kinematics type.

9.4.3.1 HAL Pin Summary

1. **motion.switchkins-type** Input (float)
2. **kinstype.is-0** Output (bit)
3. **kinstype.is-1** Output (bit)
4. **kinstype.is-2** Output (bit)

9.4.4 Anwendung

9.4.4.1 HAL-Verbindungen

Die Switchkins-Funktionalität wird durch den Pin **motion.switchkins-type** aktiviert. Normalerweise wird dieser Pin von einem analogen Ausgangspin wie motion.analog-out-03 gespeist, so dass er durch M68-Befehle gesetzt werden kann. Beispiel:

```
net :kinstype-select <= motion.analog-out-03
net :kinstype-select => motion.switchkins-type
```

9.4.4.2 G-/M-Code-Befehle

Die Auswahl des Kinstype wird verwaltet über G-Code-Sequenzen wie:

```
...
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
...
```

Anmerkung
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.

These G-code command sequences are typically implemented in G-code subroutines as remapped M-codes or with conventional M-code scripts.

Vorgeschlagene Codes (wie in den Sim-Konfigurationen verwendet) sind:

Conventional User M-codes:

1. M128 Kintyp 0 auswählen (Standardkinematik beim Start)
2. M129 Kintyp 1 auswählen (typischerweise Identitätskinematik)
3. M130 Kinstype 2 auswählen (benutzerdefinierte Kinematik)

Remapped M-codes:

1. M428 Kintyp 0 auswählen (Standardkinematik beim Start)
2. M429 Kinstype 1 auswählen (typischerweise Identitätskinematik)
3. M430 Kinstype 2 auswählen (benutzerdefinierte Kinematik)

Anmerkung
Conventional user M-codes (in the range M100-M199) are in modal group 10. Remapped M-codes (in the range M200 to M999) can specify a modalgroup. See the remap documentation for additional information.

9.4.4.3 INI file limit settings

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.

Anmerkung
INI-HAL-Pins werden während eines G-Code-Programms normalerweise nicht erkannt, es sei denn, es wird ein Synchronisationsbefehl (Queue-Buster) ausgegeben. Weitere Informationen 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
```

The relevant INI-HAL pins for an axis coordinate (*L*) are:

```
ini.L.min_limit
ini.L.max_limit
ini.L.max_velocity
ini.L.max_acceleration
```

Anmerkung
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)

Anmerkung
Die Vismach-Simulationskonfigurationen für einen Puma-Roboter demonstrieren die M-Code-Skripte (M128, M129, M130) für diesen Beispielanwendungsfall.

9.4.4.4 Überlegungen zum Offset

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.

9.4.5 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)

9.4.6 Kinematische Bestimmungen des Benutzers

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
```

9.4.7 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.

Anmerkung
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.

9.4.8 Code Anmerkungen

Kinematikmodule, die switchkins-Funktionen bereitstellen, sind mit dem Objekt switchkins.o (switchkins.c) verknüpft, welches das Hauptprogramm des Moduls (rtapi_app_main()) und zugehörige Funktionen bereitstellt. Dieses Hauptprogramm liest die (optionalen) Kommandozeilenparameter des Moduls (Koordinaten, sparm) und übergibt sie an die vom Modul bereitgestellte Funktion switchkinsSetup().

Die Funktion switchkinsSetup() identifiziert die kinstype-spezifischen Setup-Routinen und die Funktionen für die Vorwärts- und Rückwärtsberechnung für jeden Kinstype (0,1,2) und setzt eine Reihe von Konfigurationseinstellungen.

Nach dem Aufruf von switchkinsSetup() prüft rtapi_app_main() die übergebenen Parameter, erstellt eine HAL Komponente und ruft dann die für jeden Kinstype (0,1,2) identifizierte Setup-Routine auf.

Jede Kinstype (0,1,2) Setup-Routine kann (optional) HAL Pins erzeugen und auf Standardwerte setzen. Wenn alle Setup-Routinen abgeschlossen sind, gibt rtapi_app_main() hal_ready() für die Komponente aus, um die Erstellung des Moduls abzuschließen.

9.5 PID-Abstimmung (engl. tuning)

9.5.1 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).

9.5.1.1 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.

[3] Dieser Unterabschnitt ist einem viel umfangreicheren Artikel entnommen, der unter http://en.wikipedia.org/wiki/-PID_controller zu finden ist.

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.

9.5.1.2 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.

9.5.1.3 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-Verfahren Eine weitere Abstimmungsmethode ist formal als ”Ziegler-Nichols-Methode” bekannt, die von John G. Ziegler und Nathaniel B. Nichols eingeführt wurde. footnote:[Eingeführt in dem 1942 veröffentlichten Papier Optimum Settings for Automatic Controllers, DOI 10.1115/1.2899060 sowie verfügbar im Internet Archive. Sie beginnt auf die gleiche Weise wie die zuvor beschriebene Methode: Setzen Sie zunächst die I- und D-Verstärkungen auf Null, erhöhen Sie dann die P-Verstärkung und setzen Sie die Schleife externen Störeinflüssen aus, z. B. durch Stöße auf die Motorachse, um sie aus dem Gleichgewicht zu bringen. Notieren Sie die kritische Verstärkung (K_c) und die Schwingungsdauer des Ausgangs (P_c). Stellen Sie dann die Regler P, I und D wie in der Tabelle angegeben ein:

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.

9.6 Neuzuordnung (engl. remap) für das Erweitern von G-Code

9.6.1 Einführung: Erweiterung des RS274NGC-Interpreters durch Remapping von Codes

9.6.1.1 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.

9.6.1.2 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. Dies 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:

- for simple situations, a built-in glue procedure (`argspec`) covers most common parameter passing requirements
- 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.

9.6.2 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.

9.6.2.1 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 unbelegte M-codes Abschnitt zu finden.

- Informationen zu unbelegten G-Codes finden Sie hier.
- Vorhandene Codes, die neu zugewiesen werden können, sind im Abschnitt neu zuweisbare 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
```

You must also add the required Python file in your configuration folder.

Upgrade einer bestehenden Konfiguration

9.6.2.2 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 here,
- Nicht zugeordnete G-Codes werden hier aufgelistet.
- Vorhandene Codes, die neu zugeordnet werden können, sind in dieser Liste aufgeführt.

9.6.2.3 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.

9.6.2.4 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.

9.6.2.5 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.

9.6.2.6 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]SUBR existiert.

Eine ausführliche Erläuterung der REMAP (engl. für Neuzuordnung)-Parameter finden Sie im folgenden Referenzteil.

9.6.3 Neuzuordnung konfigurieren

9.6.3.1 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
: for these the modal group is fixed and any modalgroup= option is ignored.

argspec=<*argspec*>
: See description of the argspec parameter options. Optional.

ngc=<*ngc_basename*>
Basename of an O-word subroutine file name. Do not specify an .ngc extension. Searched for in the directories specified in the directory specified in `[DISPLAY]PROGRAM_PREFIX`, then in `[RS274NGC]SUB`. Mutually exclusive with `python=`. It is an error to omit both `ngc=` and `python=`.

python=<*Python function name*>
Instead of calling an ngc O-word procedure call a Python function. The function is expected to be defined in the `module_basename.oword` module. Mutually exclusive with `ngc=`.

prolog=<*Python function name*>
Before executing an ngc procedure, call this Python function. The function is expected to be defined in the `module_basename.remap` module. Optional.

epilog=<*Python function name*>
After executing an ngc procedure, call this Python function. The function is expected to be defined in the `module_basename.remap` module. Optional.

The `python`, `prolog` and `epilog` options require the Python Interpreter plugin to be configured, and appropriate Python functions to be defined there so they can be referred to with these options.

The syntax for defining a new code, and redefining an existing code is identical.

9.6.3.2 Useful REMAP option combinations

Note that while many combinations of argspec options are possible, not all of them make sense. The following combinations are useful idioms:

argspec=<*words*> ngc=<*procname*> modalgroup=<*group*>
The recommended way to call an NGC procedure with a standard argspec parameter conversion. Used if argspec is good enough. Note it's not good enough for remapping the Tx and M6/M61 tool change codes.

prolog=<*pythonprolog*> ngc=<*procname*> epilog=<*pythonepilog*> modalgroup=<*group*>
Call a Python prolog function to take any preliminary steps, then call the NGC procedure. When done, call the Python epilog function to do any cleanup or result extraction work which cannot be handled in G-code. The most flexible way of remapping a code to an NGC procedure, since almost all of the Interpreter internal variables, and some internal functions may be accessed from the prolog and epilog handlers. Also, a longer rope to hang yourselves.

python=<*pythonfunction*> modalgroup=<*group*>
Directly call to a Python function without any argument conversion. The most powerful way of remapping a code and going straight to Python. Use this if you don't need an NGC procedure, or NGC is just getting in your way.

argspec=<*words*> python=<*pythonfunction*> modalgroup=<*group*>
Convert the argspec words and pass them to a Python function as keyword argument dictionary. Use it when you're too lazy to investigate words passed on the block yourself.

Note that if all you want to achieve is to call some Python code from G-code, there is the somewhat easier way of calling Python functions like O-word procedures.

9.6.3.3 The argspec parameter

The argument specification (keyword `argspec`) describes required and optional words to be passed to an ngc procedure, as well as optional preconditions for that code to execute.

An argspec consists of 0 or more characters of the class `[@A-KMNP-Za-kmnp-z^>]` . It can by empty (like `argspec=`).

An empty argspec, or no argspec argument at all implies the remapped code does not receive any parameters from the block. It will ignore any extra parameters present.

Note that RS274NGC rules still apply - for instance you may use axis words (eg X,Y,Z) only in the context of a G-code.

Axis words may also only be used if the axis is enabled. If only XYZ are enabled, ABCUVW will not be available to be used in argspec.

Words FST will have the normal functions but will be available as variables in the remapped function. F will set feedrate, S will set spindle RPM, T will trigger the tool prepare function. Words FST should not be used if this behavior is not desired.

Words DEIJKPQR have no predefined function and are recommended for use as argspec parameters.

ABCDEFHIJKPQRSTUVWXYZ
: Defines a required word parameter: an uppercase letter specifies that the corresponding word **must** be present in the current block. The word`s value will be passed as a local named parameter with a corresponding name. If the `@` character is present in the argspec, it will be passed as positional parameter, see below.

abcdefhijkpqrstuvwxyz
: Defines an optional word parameter: a lowercase letter specifies that the corresponding word **may** be present in the current block. If the word is present, the word's value will be passed as a local named parameter. If the `@` character is present in the argspec, it will be passed as positional parameter, see below.

@
: The `@` (at-sign) tells argspec to pass words as positional parameters, in the order defined following the `@` option. Note that when using positional parameter passing, a procedure cannot tell whether a word was present or not, see example below.

Tipp
this helps with packaging existing NGC procedures as remapped codes. Existing procedures do expect positional parameters. With the `@` option, you can avoid rewriting them to refer to local named parameters.

---

^
: The ^ (caret) character specifies that the current spindle speed must be greater than zero (spindle running), otherwise the code fails with an appropriate error message.

>
: The > (greater-than) character specifies that the current feed must be greater than zero, otherwise the code fails with an appropriate error message.

n
: The `n` (greater-than) character specifies to pass the current line number in the `n`local named parameter.

By default, parameters are passed as local named parameter to an NGC procedure. These local parameters appear as *already set* when the procedure starts executing, which is different from existing semantics (local variables start out with value 0.0 and need to be explicitly assigned a value).

Optional word parameters may be tested for presence by the `EXISTS(#<word>)` idiom.

Example for named parameter passing to NGC procedures Assume the code is defined as

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=m400
```

and `m400.ngc` looks as follows:

```
o<m400> sub
(P is required since it's 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
```

- executing `M400` will fail with the message `user-defined M400: missing: P`
- executing `M400 P123` will display `P word=123.000000`
- executing `M400 P123 Q456` will display `P word=123.000000` and `Q word set: 456.000000`

Example for positional parameter passing to NGC procedures Assume the code is defined as

`REMAP=M410 modalgroup=10 argspec=@PQr ngc=m410`

and `m410.ngc` looks as follows:

```
o<m410> sub
(debug, [1]=#1 [2]=#2 [3]=#3)
o<m410> endsub
M2
```

- executing `M410 P10` will display `m410.ngc: [1]=10.000000 [2]=0.000000`
- executing `M410 P10 Q20` will display `m410.ngc: [1]=10.000000 [2]=20.000000`

Anmerkung
you lose the capability to distinguish more than one optional parameter word, and you cannot tell whether an optional parameter was present but had the value 0, or was not present at all.

Simple example for named parameter passing to a Python function It's possible to define new codes *without* any NGC procedure. Here's a simple first example, a more complex one can be found in the next section.

Assume the code is defined as

`REMAP=G88.6 modalgroup=1 argspec=XYZp python=g886`

This instructs the interpreter to execute the Python function `g886` in the `module_basename.remap` module which might look like so:

```
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
```

Try this with out with: g88.6 x1 y2 z3 g88.6 x1 y2 z3 p33 (a comment here)

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):
        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.

9.6.4 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` 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/x` 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
```

Bearbeiten Sie nun `/home/user/xxx/xxx.ini` und fügen Sie 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
```

9.6.5 Codes für den Wechsel des Remapping-Werkzeugs: T, M6, M61

9.6.5.1 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.

Decisions, decisions:

- 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.

Anmerkung
If you hate O-word procedures and love Python, you're free to do it all in Python, in which case you would just have a `python=_<function>_` spec in the REMAP statement. But assuming most folks would be interested in using O-word procedures because they are more familiar with that, we'll do that as the first example.

Der Gesamtansatz für unser erstes Beispiel lautet also:

1. We'd like to do as much as possible with G-code in an O-word procedure for flexibility. That includes all HAL interaction which would normally be handled by `iocontrol` - because we rather would want to do clever things with moves, probes, HAL pin I/O and so forth.
2. We'll try to minimize Python code to the extent needed to keep the interpreter happy, and cause `task` to actually do anything. That will go into the `prolog` and `epilog` Python functions.

9.6.5.2 Verstehen der Rolle von ”iocontrol” mit neu zugeordneten Werkzeugwechselcodes

`iocontrol` bietet zwei HAL-Interaktionssequenzen, die wir verwenden oder nicht verwenden können:

- When the NML message queued by a SELECT_TOOL() canon command is executed, this triggers the ”raise tool-prepare and wait for tool-prepared to become high” HAL sequence in `iocontrol`, besides setting the XXXX pins
- When the NML message queued by the CHANGE_TOOL() canon command is executed, this triggers the ”raise tool-change and wait for tool-changed to become high” HAL sequence in `iocontrol`, besides setting the XXXX pins

What you need to decide is whether the existing `iocontrol` HAL sequences are sufficient to drive your changer. Maybe you need a different interaction sequence - for instance more HAL pins, or maybe a more complex interaction. Depending on the answer, we might continue to use the existing `iocontrol` HAL sequences, or define our own ones.

For the sake of documentation, we’ll disable these `iocontrol` sequences, and roll our own - the result will look and feel like the existing interaction, but now we have complete control over them because they are executed in our own O-word procedure.

So what we’ll do is use some `motion.digital-*` and `motion.analog-*` pins, and the associated `M62 .. M68` commands to do our own HAL interaction in our O-word procedure, and those will effectively replace the `iocontrol` *tool-prepare/tool-prepared* and *tool-change/tool-changed* sequences. So we’ll define our pins replacing existing `iocontrol` pins functionally, and go ahead and make the `iocontrol` interactions a loop. We’ll use the following correspondence in our example:

`iocontrol` pin correspondence in the examples

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

Let us assume you want to redefine the M6 command, and replace it by an O-word procedure, but other than that things *should continue to work*.

So what our O-word procedure would do is to replace the steps outlined here. Looking through these steps you’ll find that NGC code can be used for most of them, but not all. So the stuff NGC can’t handle will be done in Python prolog and epilog functions.

9.6.5.3 Specifying the M6 replacement

To convey the idea, we just replace the built in M6 semantics with our own. Once that works, you may go ahead and place any actions you see fit into the O-word procedure.

Going through the steps, we find:

1. check for T command already executed - **execute in Python prolog**
2. check for cutter compensation being active - **execute in Python prolog**
3. stop the spindle if needed - **can be done in NGC**
4. quill up - **can be done in NGC**

5. if TOOL_CHANGE_AT_G30 was set:
   a. move the A, B and C indexers if applicable - **can be done in NGC**
   b. generate rapid move to the G30 position - **can be done in NGC**
6. send a CHANGE_TOOL Canon command to `task` - **execute in Python epilog**
7. set the numberer parameters 5400-5413 according to the new tool - **execute in Python epilog**
8. signal to `task` to stop calling the interpreter for readahead until tool change complete - **execute in Python epilog**

So we need a prolog, and an epilog. Lets assume our INI file incantation of the M6 remap looks as follows:

```
REMAP=M6   modalgroup=6  prolog=change_prolog ngc=change epilog=change_epilog
```

So the prolog covering steps 1 and 2 would look like so - we decide to pass a few variables to the remap procedure which can be inspected and changed there, or used in a message. Those are: `tool_in_spindle`, `selected_tool` (tool numbers) and their respective tooldata indices `current_pocket` and `selected_pocket`:

Anmerkung
The legacy names **selected_pocket** and **current_pocket** actually reference a sequential tooldata index for tool items loaded from a tool table ([EMCIO]TOOL_TABLE) or via a tooldata database ([EMCIO]DB_PROGRAM)

```
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, e:
        return "M6/change_prolog: %s" % (e)
```

You will find that most prolog functions look very similar: first test that all preconditions for executing the code hold, then prepare the environment - inject variables and/or do any preparatory processing steps which cannot easily be done in NGC code; then hand off to the NGC procedure by returning INTERP_OK.

Our first iteration of the O-word procedure is unexciting - just verify we got parameters right, and signal success by returning a positive value; steps 3-5 would eventually be covered here (see here for the variables referring to INI file settings):

```
O<change> sub
(debug, change: current_tool=#<current_tool>)
(debug, change: selected_pocket=#<selected_pocket>)
;
; insert any G-code which you see fit here, eg:
; G0  #<_ini[setup]tc_x>  #<_ini[setup]tc_y>  #<_ini[setup]tc_z>
;
O<change> endsub [1]
m2
```

Assuming success of `change.ngc`, we need to mop up steps 6-8:

```
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)
```

This replacement M6 is compatible with the built in code, except steps 3-5 need to be filled in with your NGC code.

Again, most epilogs have a common scheme: first, determine whether things went right in the remap procedure, then do any commit and cleanup actions which can't be done in NGC code.

9.6.5.4 Configuring `iocontrol` with a remapped M6

Note that the sequence of operations has changed: we do everything required in the O-word procedure - including any HAL pin setting/reading to get a changer going, and to acknowledge a tool change - likely with `motion.digital-*` and `motion-analog-*` IO pins. When we finally execute the `CHANGE_TOOL()` command, all movements and HAL interactions are already completed.

Normally only now `iocontrol` would do its thing as outlined here. However, we don't need the HAL pin wiggling anymore - all `iocontrol` is left to do is to accept we're done with prepare and change.

This means that the corresponding `iocontrol` pins have no function any more. Therefore, we configure `iocontrol` to immediately acknowledge a change by configuring like so:

```
# loop change signals when remapping M6
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

If you for some reason want to remap `T__x__` (prepare), the corresponding `iocontrol` pins need to be looped as well.

9.6.5.5 Writing the change and prepare O-word procedures

The standard prologs and epilogs found in `ncfiles/remap_lib/python-stdglue/stdglue.py` pass a few *exposed parameters* to the remap procedure.

An *exposed parameter* is a named local variable visible in a remap procedure which corresponds to interpreter-internal variable which is relevant for the current remap. Exposed parameters are set up in the respective prolog, and inspected in the epilog. They can be changed in the remap procedure and the change will be picked up in the epilog. The exposed parameters for remappable built in codes are:

- `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>__

If you have specific needs for extra parameters to be made visible, that can simply be added to the prolog - practically all of the interpreter internals are visible to Python.

9.6.5.6 Making minimal changes to the built in codes, including M6

Remember that normally remapping a code completely disables all internal processing for that code.

However, in some situations it might be sufficient to add a few codes around the existing M6 built in implementation, like a tool length probe, but other than that retain the behavior of the built in M6.

Since this might be a common scenario, the built in behavior of remapped codes has been made available within the remap procedure. The interpreter detects that you are referring to a remapped code within the procedure which is supposed to redefine its behavior. In this case, the built in behavior is used - this currently is enabled for the set: M6, M61,T, S, F. Note that otherwise referring to a code within its own remap procedure would be a error - a remapping recursion.

Slightly twisting a built in would look like so (in the case of M6):

```
REMAP=M6   modalgroup=6  ngc=mychange
```

```
o<mychange> sub
M6 (use built in M6 behavior)
(.. move to tool length switch, probe and set tool length..)
o<mychange> endsub
m2
```

Achtung
when redefining a built in code, **do not specify any leading zeroes in G- or M-codes** - for example, say REMAP=M1 .., not REMAP=M01 ....

See the configs/sim/axis/remap/extend-builtins directory for a complete configuration which is the recommended starting point for own work when extending built in codes.

9.6.5.7 Specifying the T (prepare) replacement

If you're confident with the default implementation, you wouldn't need to do this. But remapping is also a way to work around deficiencies in the current implementation, for instance to not block until the "tool-prepared" pin is set.

What you could do, for instance, is: - in a remapped T, just set the equivalent of the "tool-prepare" pin, but **not** wait for "tool-prepared" here - in the corresponding remapped M6, wait for the "tool-prepared" at the very beginning of the O-word procedure.

Again, the iocontrol tool-prepare/tool-prepared pins would be unused and replaced by motion.* pins, so those would pins must be looped:

```
# loop prepare signals when remapping T
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
```

So, here's the setup for a 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)
```

The minimal ngc prepare procedure again looks like so:

```
o<prepare> sub
; returning a positive value to commit:
o<prepare> endsub [1]
m2
```

And the 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)
```

prepare_prolog und prepare_epilog sind Teil des "Standard Glue", der von "nc_files/remap_lib/python-stdglue/stdglue.py" bereitgestellt wird. Dieses Modul soll die meisten Standard-Remapping-Situationen auf eine einheitliche Weise abdecken.

9.6.5.8 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
```

Achtung
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.

9.6.5.9 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
```

Anmerkung
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
- Gibt 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).

9.6.6 Umschlüsselung anderer bestehender Codes: S, M0, M1, M60

9.6.6.1 Automatic gear selection be remapping S (set spindle speed)

A potential use for a remapped S code would be *automatic gear selection* depending on speed. In the remap procedure one would test for the desired speed attainable given the current gear setting, and change gears appropriately if not.

9.6.6.2 Anpassen des Verhaltens von M0, M1, M60

A use case for remapping M0/M1 would be to customize the behavior of the existing code. For instance, it could be desirable to turn off the spindle, mist and flood during an M0 or M1 program pause, and turn these settings back on when the program is resumed.

For a complete example doing just that, see *configs/sim/axis/remap/extend-builtins/*, which adapts M1 as laid out above.

9.6.7 Creating new G-code cycles

A G-code cycle as used here is meant to behave as follows:

- On first invocation, the associated words are collected and the G-code cycle is executed.
- If subsequent lines just continue parameter words applicable to this code, but no new G-code, the previous G-code is re-executed with the parameters changed accordingly.

An example: Assume you have `G84.3` defined as remapped G-code cycle with the following INI segment (see here for a detailed description of `cycle_prolog` and `cycle_epilog`):

```
[RS274NGC]
# A cycle with an O-word procedure: 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.

9.6.8 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.

9.6.8.1 Python plugin : INI file configuration

`[PYTHON]`

`TOPLEVEL = <Dateiname>`
Dateiname des anfänglichen Python-Skripts, das ausgeführt werden soll beim Start von LinuxCNC. Dieses Skript ist für die Einrichtung des Paket Namenstruktur 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>`
log level of plugin-related actions. Increase this if you suspect problems. Can be very verbose.

`RELOAD_ON_CHANGE = [0|1]`
reload the *TOPLEVEL* script if the file was changed. Handy for debugging but currently incurs some runtime overhead. Turn this off for production configurations.

`PYTHON_TASK = [0|1]`
Start the Python task plug in. Experimental. See xxx.

9.6.8.2 Executing Python statements from the interpreter

For ad-hoc execution of commands the Python *hot comment* has been added. Python output by default goes to stdout, so you need to start LinuxCNC from a terminal window to see results. Example (eg. in the MDI window):

```
;py,print(2*3)
```

Note that the interpreter instance is available here as `self`, so you could also run:

```
;py,print(self.tool_table[0].toolno)
```

The `emcStatus` structure is accessible, too:

```
;py,from emctask import *
;py,print(emcstat.io.aux.estop)
```

9.6.9 Programming Embedded Python in the RS274NGC Interpreter

9.6.9.1 The Python plugin namespace

The namespace is expected to be laid out as follows:

`oword`
Any callables in this module are candidates for Python O-word procedures. Note that the Python `oword` module is checked **before** testing for a NGC procedure with the same name - in effect names in `oword` will hide NGC files of the same basename.

remap
Python callables referenced in an argspec `prolog`,`epilog` or `python` option are expected to be found here.

namedparams
Python funtcions int this module extend or redefine the namespace of predefined named parameters, see adding predefined parameters.

task
Hier werden aufgabenbezogene Abrufe erwartet.

9.6.9.2 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.

9.6.9.3 Die Interpreterfunktionen `__init__` und `__delete__`

If the `TOPLEVEL` module defines a function `__init__`, it will be called once the interpreter is fully configured (INI file read, and state synchronized with the world model).

If the `TOPLEVEL` module defines a function `__delete__`, it will be called once before the interpreter is shutdown and after the persistent parameters have been saved to the `PARAMETER_FILE`.

Note_ at this time, the `__delete__` handler does not work for interpreter instances created by importing the `gcode` module. If you need an equivalent functionality there (which is quite unlikely), please consider the Python `atexit` module.

```
# this would be defined in the TOPLEVEL module

def __init__(self):
    # add any one-time initialization here
    if self.task:
  # this is the milltask instance of interp
  pass
    else:
  # this is a non-milltask instance of interp
        pass

def __delete__(self):
    # add any cleanup/state saving actions here
    if self.task: # as above
  pass
    else:
        pass
```

This function may be used to initialize any Python-side attributes which might be needed later, for instance in remap or oword functions, and save or restore state beyond what `PARAMETER_FILE` provides.

If there are setup or cleanup actions which are to happen only in the milltask Interpreter instance (as opposed to the interpreter instance which sits in the `gcode` Python module and serves preview/progress display purposes but nothing else), this can be tested for by evaluating self.task.

An example use of `__init__` and `__delete__` can be found in *configs/sim/axis/remap/cycle/python/toplevel.py* initialising attributes needed to handle cycles in *ncfiles/remap_lib/python-stdglue/stdglue.py* (and imported into *configs/sim/axis/remap/cycle/python/remap.py*).

9.6.9.4 Calling conventions: NGC to Python

Python code is called from NGC in the following situations:

- during normal program execution:
  - when an O-word call like `O<proc> call` is executed and the name `oword.proc` is defined and callable
  - when a comment like `;py,<Python statement>` is executed
- during execution of a remapped code: any `prolog=`, `python=` and `epilog=` handlers.

Calling O-word Python subroutines

Arguments:

`self`
: the interpreter instance

`*args`
: the list of actual positional parameters. Since the number of actual parameters may vary, it is best to use this style of declaration:

```
# this would be defined in the oword module
def mysub(self, *args):
    print("number of parameters passed:", len(args))
    for a in args:
  print(a)
```

Return values of O-word Python subroutines Just as NGC procedures may return values, so do O-word Python subroutines. They are expected to either return

- no value (no `return` statement or the value `None`),
- a float or int value,
- a string, this means *this is an error message, abort the program*. Works like `(abort, msg)`.

Any other return value type will raise a Python exception.

In a calling NGC environment, the following predefined named parameters are available:

`#__<_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`
: the interpreter instance

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.

”ein Nachrichtentext”
: 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
: the interpreter instance

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
: Geben Sie dies bei Erfolg zurück

”ein Nachrichtentext”
: 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, 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)
```

Warnung
The *yield* feature is fragile. The following restrictions apply to the usage of *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.

9.6.9.5 Aufrufkonventionen: Python zu NGC

NGC-Code wird von Python ausgeführt, wenn

- die Methode `self.execute(<NGC-Code>[,<Zeilennummer>])` ausgeführt wird, oder
- during execution of a remapped code, if a `prolog=` function is defined, the NGC procedure given in `ngc=` is executed immediately thereafter.

The prolog handler does not call the handler, but it prepares its call environment, for instance by setting up predefined local parameters.

Inserting parameters in a prolog, and retrieving them in an epilog Conceptually a prolog and an epilog execute at the same call level like the O-word procedure, that is: after the subroutine call is set up, and before the subroutine endsub or return.

This means that any local variable created in a prolog will be a local variable in the O-word procedure, and any local variables created in the O-word procedure are still accessible when the epilog executes.

The `self.params` array handles reading and setting numbered and named parameters. If a named parameter begins with _ (underscore), it is assumed to be a global parameter; if not, it is local to the calling procedure. Also, numbered parameters in the range 1..30 are treated like local variables; their original values are restored on return/endsub from an O-word procedure.

Here is an example remapped code demonstrating insertion and extraction of parameters into/from the O-word procedure:

```
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
```

`self.params()` returns a list of all variable names currently defined. Since `myname` is local, it goes away after the epilog finishes.

Calling the interpreter from Python You can recursively call the interpreter from Python code as follows:

```
self.execute(<NGC code>[,<line number>])
```

Beispiele:

```
  self.execute("G1 X%f Y%f" % (x,y))
  self.execute("O <myprocedure> call", currentline)
```

You might want to test for the return value being < `INTERP_MIN_ERROR`. If you're using lots of execute() statements, it's probably easier to trap InterpreterException as per below.

Achtung
The parameter insertion/retrieval method described in the previous section does not work in this case. It is good enough for just executing simple NGC commands or a procedure call and advanced introspection into the procedure, and passing of local named parameters is not needed. The recursive call feature is fragile.

Interpreter Exception during execute() if `interpreter.throw_exceptions` is nonzero (default 1), and self.execute() returns an error, the exception `InterpreterException` is raised. InterpreterException has the following attributes:

`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 The canon layer is practically all free functions. Example:

```
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
```

The actual canon functions are declared in `src/emc/nml_intf/canon.hh` and implemented in `src/emc/task/`
The implementation of the Python functions can be found in `src/emc/rs274ncg/canonmodule.cc`.

9.6.9.6 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` Regressionstest.

emccanon
legt die meisten Aufrufe von `src/emc/task/emccanon.cc` offen.

emctask
stellt die Instanz der Klasse `emcStatus` zur Verfügung. Siehe `src/emc/task/taskmodule.cc`. Nicht vorhanden, wenn das Modul `gcode` für Benutzeroberflächen verwendet wird - nur in der milltask-Instanz des Interpreters vorhanden.

9.6.10 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-`

9.6.11 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:

9.6.11.1 T: `prepare_prolog` and `prepare_epilog`

Diese verpacken ein NGC-Verfahren für Tx 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.

Actions of `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.

9.6.11.2 M6: `change_prolog` and `change_epilog`

Diese schließen ein NGC-Verfahren für den M6-Werkzeugwechsel ein.

Actions of `change_prolog`

- Die folgenden drei Schritte sind nur anwendbar, wenn die Komponente ”iocontrol-v2” verwendet wird:
  - Wenn der Parameter 5600 (Fehleranzeige) größer als Null ist, deutet dies auf einen Fehler des Werkzeugwechslers hin, der wie folgt behandelt wird:
  - Wenn der Parameter 5601 (Fehlercode) negativ ist, deutet dies auf einen schwerwiegenden Fehler hin und der Prolog bricht mit einer Fehlermeldung ab.
  - Wenn Parameter 5601 (Fehlercode) größer als Null ist, bedeutet dies einen Soft-Fehler. Es wird eine Informationsmeldung angezeigt und das Prolog wird fortgesetzt.
- 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

Actions of +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.
- Ist der Parameter 5600 (Fehlerindikator) größer als Null, deutet dies auf einen Werkzeugwechslerfehler hin, der wie folgt behandelt wird (nur ”iocontrol-v2”):
  - If parameter 5601 (error code) is negative, this indicates a hard fault and the epilog aborts with an error message.

  - If parameter 5601 (error code) is greater equal zero, this indicates a soft fault. An informational message is displayed and the epilog continues.

- 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.
- Then, the Canon command `CHANGE_TOOL(#<selected_pocket>)` is executed.
- Die neuen Werkzeugparameter (Versatz, Durchmesser usw.) werden eingestellt.

9.6.11.3 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.

These routines are designed to work in conjunction with an argspec=<words> parameter. While this is easy to use, in a realistic scenario you would avoid argspec and do a more thorough investigation of the block manually in order to give better error messages.

Der Vorschlag für argspec lautet wie folgt:

```
REMAP=G<somecode> argspec=xyzabcuvwqplr prolog=cycle_prolog ngc=<ngc procedure> epilog= ↩
    cycle_epilog modalgroup=1
```

This will permit `cycle_prolog` to determine the compatibility of any axis words give in the block, see below.

Actions of `cycle_prolog`

- Ermitteln Sie, ob die vom aktuellen Block übergebenen Wörter die unter Canned Cycle Errors genannten Bedingungen erfüllen.
  - Export the axis words as <x>, #<y> etc; fail if axis words from different groups (XYZ) (UVW) are used together, or any of (ABC) is given.
  - Export *L*- as #<l>; default to 1 if not given.
  - Export *P*- as #<p>; fail if p less than 0.
  - Export *R*- as #<r>; fail if r not given, or less equal 0 if given.
  - Fail if feed rate is zero, or inverse time feed or cutter compensation is on.
- Feststellen, ob dies der erste Aufruf eines Zyklus-G-Codes ist, falls ja:
  - Add the words passed in (as per argspec) into a set of sticky parameters, which is retained across several invocations.
- If not (a continuation line with new parameters) then
  - merge the words passed in into the existing set of sticky parameters.
- Exportieren Sie den Satz der Sticky-Parameter in das NGC-Verfahren.

Actions of `cycle_epilog`

- Determine if the current code was in fact a cycle, if so, then
  - retain the current motion mode so a continuation line without a motion code will execute the same motion code.

9.6.11.4 S (Set Speed) : `setspeed_prolog` and `setspeed_epilog`

TBD

9.6.11.5 F (Set Feed) : `setfeed_prolog` and `setfeed_epilog`

TBD

9.6.11.6 M61 Set tool number : `settool_prolog` and `settool_epilog`

TBD

9.6.12 Remapped code execution

9.6.12.1 NGC procedure call environment during remaps

Normally, an O-word procedure is called with positional parameters. This scheme is very limiting in particular in the presence of optional parameters. Therefore, the calling convention has been extended to use something remotely similar to the Python keyword arguments model.

see LINKTO gcode/main Subroutines: sub, endsub, return, call.

9.6.12.2 Nested remapped codes

Remapped codes may be nested just like procedure calls - that is, a remapped code whose NGC procedure refers to some other remapped code will execute properly.

The maximum nesting level remaps is currently 10.

9.6.12.3 Sequence number during remaps

Sequence numbers are propagated and restored like with O-word calls. See `tests/remap/nested-remaps/w` for the regression test, which shows sequence number tracking during nested remaps three levels deep.

9.6.12.4 Debugging-Flags

Die folgenden Flags sind für das Mapping und die Ausführung in Python relevant:

```
EMC_DEBUG_OWORD             0x00002000  traces execution of O-word subroutines
EMC_DEBUG_REMAP             0x00004000  traces execution of remap-related code
EMC_DEBUG_PYTHON            0x00008000  calls to the Python plug in
EMC_DEBUG_NAMEDPARAM        0x00010000  trace named parameter access
EMC_DEBUG_PYTHON_TASK       0x00040000  trace the task Python plug in
EMC_DEBUG_USER1             0x10000000  user-defined - not interpreted by LinuxCNC
EMC_DEBUG_USER2             0x20000000  user-defined - not interpreted by LinuxCNC
```

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*.

9.6.12.5 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.

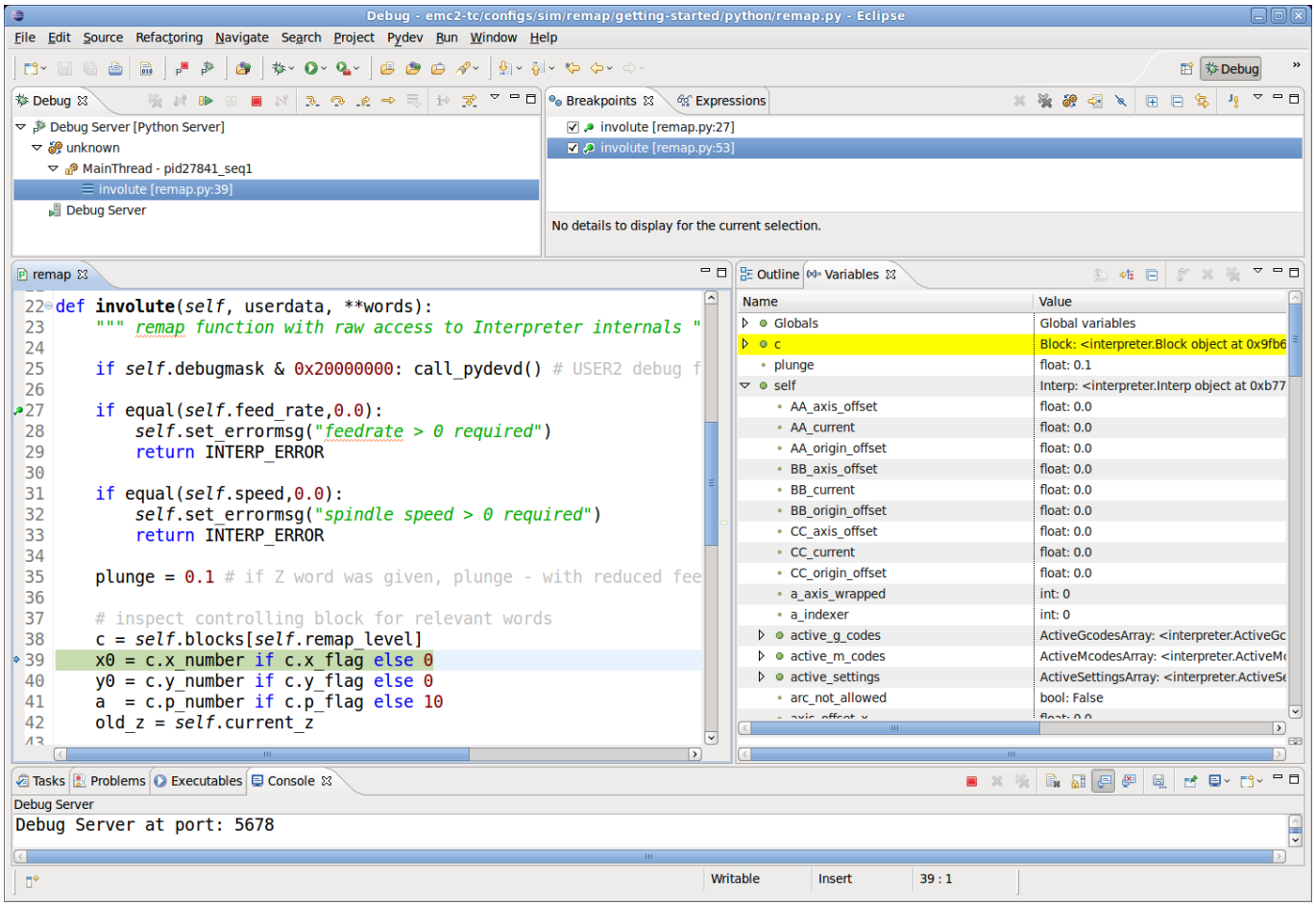
Um diesen Ansatz zu verwenden:

- Installieren Sie Eclipse über das *Ubuntu Software Center* (wählen Sie die erste Option).
- Installieren Sie das PyDev-Plug-in von der Pydev Update Site.
- Setup the LinuxCNC source tree as an Eclipse project.
- Start the Pydev Debug Server in Eclipse.
- Make sure the embedded Python code can find the `pydevd.py` module which comes with that plug in - it's buried somewhere deep under the Eclipse install directory. Set the the `pydevd` variable in `util.py` to reflect this directory location.
- Add `import pydevd` to your Python module - see example `util.py` and `remap.py`.
- Call `pydevd.settrace()` in your module at some point to connect to the Eclipse Python debug server - here you can set breakpoints in your code, inspect variables, step etc as usual.

Achtung
`pydevd.settrace()` will block execution if Eclipse and the Pydev debug server have not been started.

To cover the last two steps: the `o<pydevd>` procedure helps to get into the debugger from MDI mode. See also the `call_pydevd` function in `util.py` and its usage in `remap.involute` to set a breakpoint.

Here's a screen-shot of Eclipse/PyDevd debugging the `involute` procedure from above:



See the Python code in `configs/sim/axis/remap/getting-started/python` for details.

9.6.13 Axis Preview and Remapped code execution

For complete preview of a remapped code's tool path some precautions need to be taken. To understand what is going on, let's review the preview and execution process (this covers the Axis case, but others are similar):

First, note that there are **two** independent interpreter instances involved:

- One instance in the milltask program, which executes a program when you hit the *Start* button, and actually makes the machine move.
- A second instance in the user interface whose primary purpose is to generate the tool path preview. This one *executes* a program once it is loaded, but it doesn't actually cause machine movements.

Now assume that your remap procedure contains a G38 probe operation, for example as part of a tool change with automatic tool length touch off. If the probe fails, that would clearly be an error, so you'd display a message and abort the program.

Now, what about preview of this procedure? At preview time, of course it's not known whether the probe succeeds or fails - but you would likely want to see what the maximum depth of the probe is, and assume it succeeds and continues execution to preview further movements. Also, there is no point in displaying a *probe failed* message and aborting **during preview**.

The way to address this issue is to test in your procedure whether it executes in preview or execution mode. This can be checked for by testing the `#<_task>` predefined named parameter - it will be 1

during actual execution and 0 during preview. See *configs/sim/axis/remap/manual-toolchange-with-tool-length-switch/nc_subroutines/manual_change.ngc* for a complete usage example.

Within Embedded Python, the `task` instance can be checked for by testing *self.task* - this will be 1 in the milltask instance, and 0 in the preview instance(s).

9.6.14 Remappable Codes

9.6.14.1 Existing codes which can be remapped

The current set of **existing** codes open to redefinition is:

- T*x* (Prepare)
- M6 (Change tool)
- M61 (Set tool number)
- M0 (pause a running program temporarily)
- M1 (pause a running program temporarily if the optional stop switch is on)
- M60 (exchange pallet shuttles and then pause a running program temporarily)
- S (set spindle speed)
- F (set feed)

Note that the use of M61 currently requires the use of `iocontrol-v2`.

9.6.14.2 Currently unallocated G-codes:

Currently unallocated G-codes (for remapping) must be selected from the blank areas of the following tables. All the listed G-codes are already defined in the current implementation of LinuxCNC and may **not** be used to remap new G-codes. (Developers who add new G-codes to LinuxCNC are encouraged to also add their new G-codes to these tables.)

Tabelle 9.1: Table of Allocated 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										

Tabelle 9.2: Table of Allocated 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								

Tabelle 9.3: Table of Allocated 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								
29										

Tabelle 9.4: 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										

Tabelle 9.5: 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								

Tabelle 9.5: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
43	G43	G43.1								
44										
45										
46										
47										
48										
49	G40									

Tabelle 9.6: 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						

Tabelle 9.7: 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										

Tabelle 9.8: 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										

Tabelle 9.8: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
75										
76	G76									
77										
78										
79										

Tabelle 9.9: 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									

Tabelle 9.10: 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									

9.6.14.3 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.)

Tabelle 9.11: 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.

9.6.14.4 Vorauslesezeit und Ausführungszeit

FIXME Füge fehlende Informationen hinzu

9.6.14.5 Plugin/Pickle-Hack

FIXME Füge fehlende Informationen hinzu

9.6.14.6 Modul, Methoden, Klassen, usw. Referenz

FIXME Füge fehlende Informationen hinzu

9.6.15 Einführung: Erweiterung der Task-Ausführung

FIXME Füge fehlende Informationen hinzu

9.6.15.1 Warum sollten Sie die Task-Ausführung ändern wollen?

FIXME Füge fehlende Informationen hinzu

9.6.15.2 Ein Diagramm: `task`, `interp`, `iocontrol`, `UI` (??)

FIXME Füge fehlende Informationen hinzu

9.6.16 Modelle der Aufgaben (engl. task) -Ausführung

FIXME Füge fehlende Informationen hinzu

9.6.16.1 Traditionelle Ausführung von `iocontrol/iocontrolv2`

FIXME Füge fehlende Informationen hinzu

9.6.16.2 IO-Verfahren neu definieren

FIXME Füge fehlende Informationen hinzu

9.6.16.3 Python-Prozeduren zur Ausführungszeit

FIXME Füge fehlende Informationen hinzu

9.6.17 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.

9.6.17.1 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. *Modal state and settings* - Refers to state which is *carried over* between executing individual NGC codes - for instance, once the spindle is turned on and the speed is set, it remains at this setting until turned off. The same goes for many codes, like feed, units, motion modes (feed or rapid) and so forth.
4. *Interpreter execution state* - Holds information about the block currently executed, whether we are in a subroutine, interpreter variables, etc. . Most of this state is aggregated in a - fairly unsystematic - `structure _setup` (see interp_internals.hh).

9.6.17.2 Task and Interpreter interaction, Queuing and Read-Ahead

The `task` part of LinuxCNC is responsible for coordinating actual machine commands - movement, HAL interactions and so forth. It does not by itself handle the RS274NGC language. To do so, `task` calls upon the interpreter to parse and execute the next command - either from MDI or the current file.

The interpreter execution generates canonical machine operations, which actually move something. These are, however, not immediately executed but put on a queue. The actual execution of these codes happens in the `task` part of LinuxCNC: canon commands are pulled off that interpreter queue, and executed resulting in actual machine movements.

This means that typically the interpreter is far ahead of the actual execution of commands - the parsing of the program might well be finished before any noticeable movement starts. This behavior is called *read-ahead*.

9.6.17.3 Predicting the machine position

To compute canonical machine operations in advance during read ahead, the interpreter must be able to predict the machine position after each line of G-code, and that is not always possible.

Let's look at a simple example program which does relative moves (G91), and assume the machine starts at x=0,y=0,z=0. Relative moves imply that the outcome of the next move relies on the position of the previous one:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G1 Y20 Z-5
N40 G0 Z30
N50 M2
```

Here the interpreter can clearly predict machine positions for each line:

After N20: x=10 y=-5 z=20; after N30: x=10 y=15 z=15; after N40: x=10 y=15 z=45

and so can parse the whole program and generate canonical operations well in advance.

9.6.17.4 Queue-busters break position prediction

However, complete read ahead is only possible when the interpreter can predict the position impact for **every** line in the program in advance. Let's look at a modified example:

```
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
```

To pre-compute the move in N90, the interpreter would need to know where the machine is after line N80 - and that depends on whether the probe command succeeded or not, which is not known until it's actually executed.

So, some operations are incompatible with further read-ahead. These are called *queue busters*, and they are:

- Reading a HAL pin's value with M66: value of HAL pin not predictable.
- Loading a new tool with M6: tool geometry not predictable.
- Executing a probe with G38.*n*: final position and success/failure not predictable.

9.6.17.5 How queue-busters are dealt with

Whenever the interpreter encounters a queue-buster, it needs to stop read ahead and wait until the relevant result is available. The way this works is:

- When such a code is encountered, the interpreter returns a special return code to `task` (*INTERP_EXECUTE*
- This return code signals to `task` to stop read ahead for now, execute all queued canonical commands built up so far (including the last one, which is the queue buster), and then *synchronize the interpreter state with the world model*. Technically, this means updating internal variables to reflect HAL pin values, reload tool geometries after an M6, and convey results of a probe.

- The interpreter's *synch()* method is called by `task` and does just that - read all the world model *actual* values which are relevant for further execution.
- At this point, `task` goes ahead and calls the interpreter for more read ahead - until either the program ends or another queue-buster is encountered.

9.6.17.6 Word order and execution order

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).

9.6.17.7 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.

9.6.17.8 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-Rückgabew 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.

9.6.17.9 Prozedurausführung

O-word procedures complicate handling of queue busters a bit. A queue buster might be found somewhere in a nested procedure, resulting in a semi-finished procedure call when INTERP_EXECUTE_FINISH is returned. Task makes sure to synchronize the world model, and continue parsing and execution as long as there is still a procedure executing (call_level > 0).

9.6.17.10 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])
```

Um Felder in der globalen emcStatus-Struktur zu sehen, versuchen Sie dies:

```
;py,from emctask import *
;py,print(emcstat.io.tool.pocketPrepped)
;py,print(emcstat.io.tool.toolInSpindle)
;py,print(emcstat.io.tool.toolTable[0])
```

Sie müssen LinuxCNC von einem Terminal-Fenster aus gestartet haben, um die Ergebnisse zu sehen.

9.6.17.11 So funktioniert Tx (Prepare Tool)

Aktion des Interpreters bei einem Tx-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_TO 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, dass die Aufgabe fortgesetzt werden soll. Auch hier ist dieses ”Warten” m.E. nicht unbedingt erforderlich.

Building the prolog and epilog for Tx See the Python functions `prepare_prolog` and `prepare_epilog` in `configs/sim/axis/remap/toolchange/python/toolchange.py`.

9.6.17.12 How M6 (Change tool) works

You need to understand this fully before you can adapt it. It is very relevant to writing a prolog and epilog handler for a remapped M6. Remapping an existing codes means you disable the internal steps taken normally, and replicate them as far as needed for your own purposes.

Even if you are not familiar with C, I suggest you look at the *Interp::convert_tool_change* code in *src/emc/rs274/interp_convert.cc*.

Interpreter action on a M6 command

When the interpreter sees an M6, it:

1. checks whether a T command has already been executed (test *settings->selected_pocket* to be >= 0) and fail with *Need tool prepared -Txx- for toolchange* message if not.
2. check for cutter compensation being active, and fail with *Cannot change tools with cutter radius compensation on* if so.
3. stop the spindle except if the ”TOOL_CHANGE_WITH_SPINDLE_ON” INI option is set.
4. generate a rapid *Z up* move if if the ”TOOL_CHANGE_QUILL_UP” INI option is set.
5. if TOOL_CHANGE_AT_G30 was set:
   a. move the A, B and C indexers if applicable
   b. generate rapid move to the G30 position
6. execute a CHANGE_TOOL canon command,with the selected pocket as parameter. CHANGE_TOOL will:
   a. generate a rapid move to TOOL_CHANGE_POSITION if so set in INI
   b. enqueue an EMC_TOOL_LOAD NML message to `task`.
7. set the numberer parameters 5400-5413 according to the new tool
8. signal to `task` to stop calling the interpreter for readahead by returning INTERP_EXECUTE_FINISH since M6 is a queue buster.

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 `configs/sim/axis/remap/toolchange/python/toolchange.py`.

9.6.17.13 How M61 (Change tool number) works

M61 requires a non-negative `Q`parameter (tool number). If zero, this means *unload tool*, else *set current tool number to Q*.

Building the replacement for M61 An example Python redefinition for M61 can be found in the `set_tool_number` function in `configs/sim/axis/remap/toolchange/python/toolchange.py`.

9.6.18 Status

1. The RELOAD_ON_CHANGE feature is fairly broken. Restart after changing a Python file.
2. M61 (remapped or not) is broken in `iocontrol` and requires `iocontrol-v2` to actually work.

9.6.19 Changes

- the method to return error messages and fail used to be *self.set_errormsg(text)* followed by *return INTERP_ERROR*. This has been replaced by merely returning a string from a Python handler or oword subroutine. This sets the error message and aborts the program. Previously there was no clean way to abort a Python oword subroutine.

9.6.20 Debugging

In the *[EMC]* section of the INI file the *DEBUG* parameter can be changed to get various levels of debug messages when LinuxCNC is started from a terminal.

```
Debug level, 0 means no messages. See src/emc/nml_intf/debugflags.h for others
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
```

9.7 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, posplusoffset-M, fb-minusoffset-M) angelegt, wenn beide Freigabe-Eingangs-Pins (apply-offsets und moveenable) 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 Rückverfolgungsaktivierung 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.

Achtung
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)

9.7.1 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).

Anmerkung
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]] (Standard: 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] (Standard: 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.
```

9.8 Eigenständiger Interpreter

Der eigenständige Interpreter rs274 kann über die Kommandozeile verwendet werden.

9.8.1 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)
```

9.8.2 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
```

9.9 Offsets der externen Achse

External axis offsets are supported during teleop (world) jogs and coordinated (G-code) motion. External axis offsets are enabled on a per-axis basis by INI file settings and controlled dynamically by INI input pins. The INI interface is similar to that used for wheel jogging. This type of interface is typically implemented with a manual-pulse-generator (mpg) connected to an encoder INI component that counts pulses.

9.9.1 INI File Settings

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

Planung maximale Beschleunigung = (1-r) * MAX_ACCELERATIOIN
externe Offset-Beschleunigung = ( r) * MAX_ACCELERATION
```

9.9.2 HAL-Pins

9.9.2.1 Per-Axis Motion HAL Pins

Für jeden Achsenbuchstaben (**L** in xyzabcuvw)

1. **axis. L.eoffset-enable** Eingang (Bit): Aktivieren
2. **axis. L.eoffset-scale** Input(float): Skalierungsfaktor
3. **axis.L.eoffset-counts** Input(s32): Eingabe in das Zählregister
4. **axis.L.eoffset-clear** Input(bit): angeforderten Offset löschen
5. **axis. L.eoffset** Output(float): aktueller externer Offset
6. **axis.L.eoffset-request** Output(float): angeforderter externer Offset

9.9.2.2 Other Motion HAL Pins

1. **motion.eoffset-active** Output(Bit): Externe Offsets ungleich Null angewendet
2. **motion.eoffset-limited** Output(bit): Bewegung gesperrt durch Softlimit

9.9.3 Anwendung

The axis input HAL pins (enable,scale,counts) are similar to the pins used for wheel jogging.

9.9.3.1 Offset-Berechnung

At each servo period, the *axis.L.eoffset-counts* pin is compared to its value in the prior period. The increase or decrease (positive or negative delta) of the *axis.L.eoffset-counts* pin is multiplied by the current *axis.L.eoffset-scale* pin value. This product is accumulated in an internal register and exported to the *axis.L.eoffset-request* HAL pin. The accumulation register is reset to zero at each machine-on.

The requested offset value is used to plan the movement for the offset that is applied to the *L* coordinate and represented by the *axis.L.eoffset* HAL pin. The planned motion respects the allocated velocity and acceleration constraints and may be limited if the net motion (offset plus teleop jogging or coordinated motion) reaches a soft limit for the *L* coordinate.

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.

9.9.3.2 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.

At each startup (machine-on), the internal counts register for each HAL pin *axis.L.eoffset-counts* is zeroed and the corresponding HAL output pin *axis.L.eoffset* is reset to zero.

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.

9.9.3.3 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.eoffse 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.

9.9.3.4 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 *axis.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.

9.9.3.5 Warnung

The use of external offsets can alter machine motion in a significant manner. The control of external offsets with HAL components and connections and any associated user interfaces should be carefully designed and tested before deployment.

9.9.4 Related HAL Components

9.9.4.1 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**).

9.9.5 Testen

Der externe Achsenversatz wird durch Hinzufügen einer *[AXIS_L]*-Einstellung für jede Kandidatenachse aktiviert. Zum Beispiel:

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

For testing, it is convenient to simulate a jog wheel interface using the **sim_pin** GUI. For example, in a terminal:

```
$ sim_pin axis.z.eoffset-enable axis.z.eoffset-scale axis.z.eoffset-counts
```

The use of external offsets is aided by displaying information related to the current offsets: the current eoffset value and the requested eoffset value, the axis pos-cmd, and (for an identity kinematics machine) the corresponding joint motor pos-cmd and motor-offset. The provided sim configuration (see below) demonstrates an example PyVCP panel for the AXIS GUI.

Wenn keine benutzerdefinierte Anzeige vorhanden ist, kann **halshow** als Hilfsanwendung mit einer benutzerdefinierten Überwachungsliste gestartet werden.

Example INI file settings to simulate the HAL pin eoffset connections and display eoffset information for the z axis (for identity kinematics with 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
```

9.9.6 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 den Maschine in den HALFILEs ersetzen. Die mitgelieferten PyVCP-Dateien (.hal und .xml) können ein Ausgangspunkt für anwendungsspezifische Benutzeroberflächen sein.

9.9.6.1 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.

9.9.6.2 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.

9.9.6.3 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.

Anmerkung
Änderungen an der z-Koordinate max-acceleration und max-velocity werden während der Ausführung eines Programms nicht bestätigt.

9.9.6.4 opa.ini (eoffset_per_angle)

The opa.ini configuration uses the INI component eoffset_per_angle (**$ man eoffset_per_angle**) to demonstrate an XZC machine with functional offsets computed from the C coordinate (angle) and applied to the transvers (X) coordinate. Offset computations are based on a specified reference radius typically set by a program (or MDI) M68 command to control a **motion.analog-out-NN** pin.

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.

9.10 Tool Database Interface

Tool data is conventionally described by a tool table file specified by an inifile setting: [EMCIO]TOOL_TABLE=
A tool table file consists of a text line for each available tool describing the tool's parameters, see Tool Table Format.

The tool database interface provides an alternative method for obtaining tool data via a separate program that manages a database of tools.

9.10.1 Interface

9.10.1.1 INI-Datei Einstellungen

Die Einstellungen in der INI-Datei ermöglichen den (optionalen) Betrieb eines vom Benutzer bereitgestellten Werkzeugdatenbankprogramms:

```
[EMCIO]
DB_PROGRAM = db_program [args]
```

When included, **db_program** specifies the path to a user-provided executable program that provides tooldata. Up to 10 space-separated args may be included and passed to the **db_program** at startup.

Anmerkung
INI-Datei-Einstellungen für [EMCIO]TOOL_TABLE werden ignoriert, wenn ein **db_program** angegeben ist.

Anmerkung
The **db_program** may be implemented in any language currently supported in LinuxCNC (e.g., BASH scripts, Python or Tcl scripts, C/C++ programs) as long as it conforms to the interface messages received on stdin and replied on stdout. A **db_program** could manage data from a flat file, a relational database (SQLite for example), or other data sources.

9.10.1.2 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. The **db_program** must respond by writing a single line acknowledgement consisting of a version string (e.g., ”v2.1”). No tools will be available if the version is not compatible with the LinuxCNC database interface version.
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” put data changes caused by G10L1, G10L10, G10L11 G-codes. The tool data line will include all elements of a tool table text line.
- ”l” spindle_load (T*n*M6). The tool data line includes only the *T* and *P* items identifying the relevant tool number and pocket number.
- ”u” spindle_unload (T0M6). The tool data line includes only the *T* and *P* items identifying the relevant tool number and pocket number

Anmerkung
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 T*n *P0* (Platz 0 ist die Spindel). Der Befehl spindle_unload für T0M6 lautet: *u T0 P0*.

Anmerkung
When a RANDOM tool changer is specified using [EMCIO]RANDOM_TOOL_CHANGER=1, a pair of spindle_unload/spindle_load commands are issued at each tool exchange. The pair of commands issued for T*n*M6 (or M61Q*n*) are *u T*u *P*m followed by *l T*n *P0* where *u* is the current tool to be sent to pocket *m* and *n* is the new tool to load in the spindle (pocket *0*). By convention, a tool number of 0 is used to specify an empty tool,

9.10.1.3 Anwendung

Using a **db_program** does not change the way LinuxCNC operates but provides support for new database functionality for tool management.

For example, a **db_program** database application can maintain the operating hours for all tools by tracking each load/unload of a tool. A machine could then have three 6 mm endmills in pockets 111, 112, and 113 with the database application programmed to assign tool number 110 to the 6 mm endmill with the fewest operating hours. Then, when a LinuxCNC program requests tool 110, the database would specify the appropriate pocket based on tool usage history.

Tool data changes made within LinuxCNC (*p*,*u*,*l* commands) are pushed immediately to the **db_program** which is expected to synchronize its source data. By default, LinuxCNC requests for tool data (*g* commands) are made at startup only. A database program may update tool usage data on a continuous basis so long-lived LinuxCNC applications may benefit by refreshing the tool data provided by the **db_program**. The G-code command **G10L0** can be used to request a tool data reload (*g* command) from within G-code programs or by MDI. A reload operation is also typically provided by a Graphical User Interface (GUI) so that on-demand reloads can be requested. For example, a Python GUI application can use:

```
#!/usr/bin/env python3
from  linuxcnc import command
command().load_tool_table()
```

Alternatively, a **db_program** may push its local data changes to synchronize its data with LinuxCNC by using the load_tool_table() interface command. Commands which push changes to LinuxCNC may be rejected if the interpreter is running. The interpreter state can be checked before issuing a load_tool_table() command. Example:

```
#! /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 until interp not idle
   ...
```

If the database application adds or removes tools after initialization, a call to tooldb_tools() must be issued with an updated user_tools list. The updated list of tools will be used on subsequent get commands or load_tool_table() requests.

Anmerkung
Das Entfernen einer Werkzeugnummer sollte nur dann erfolgen, wenn die Werkzeugnummer derzeit nicht in der Spindel geladen ist.

Exporting the environmental variable DB_SHOW enables LinuxCNC prints (to stdout) that show tool data retrieved from the **db_program** at startup and at subsequent reloading of tool data.

Exporting the environmental variable DB_DEBUG enables LinuxCNC prints (to stdout) for additional debugging information about interface activity.

9.10.1.4 Example program

An example **db_program** (implemented as a Python script) is provided with the simulation examples. The program demonsrates the required operations to:

1. acknowledge startup version
2. receive tool data requests: *g* (**get** command)
3. receive tool data updates: *p* (**put** command)
4. receive tool load updates: *l* (**load_spindle** command)
5. receive tool unload updates: *u* (**unload_spindle** command)

9.10.1.5 Python tooldb module

The example program uses a LinuxCNC provided Python module (*tooldb*) that manages the low-level details for communication and version verification. This module uses callback functions specified by the **db_program** to respond to the *g* (get) command and the commands that indicate tool data changes (*p*, *l*, *u*).

The **db_program** uses the *tooldb* module by implementing the following Python code:

```
user_tools = list(...)   # list of available tool numbers

def user_get_tool(toolno):
    # function to respond to 'g' (get) commands
    # called once for each toolno in user_tools
    ...
def user_put_tool(toolno,params):
    # function to respond to 'p' (put) commands
    ...
def user_load_spindle(toolno,params):
    # function to respond to 'l' (put) commands
    ...
def user_unload_spindle(toolno,params):
    # function to respond to 'u' (put) commands
    ...

#------------------------------------------------------------
# Begin:
from tooldb import tooldb_tools     # identify known tools
from tooldb import tooldb_callbacks # identify functions
from tooldb import tooldb_loop      # main loop

tooldb_tools(user_tools)
tooldb_callbacks(user_get_tool,
```

```
                  user_put_tool,
                  user_load_spindle,
                  user_unload_spindle,
                 )
tooldb_loop()
```

Anmerkung
Use of *tooldb* is not required — it is provided as a demonstration of the required interface and as a convenience for implementing Python-based applications that interface with an external database.

9.10.2 Simulationskonfigurationen

Simulation configs using the axis gui:

1. configs/sim/axis/db_demo/**db_ran**.ini (random_toolchanger)
2. configs/sim/axis/db_demo/**db_nonran**.ini (nonrandom_toolchanger)

Each sim config simulates a **db_program** implementing a database with 10 tools numbered 10—19.

The **db_program** is provided by a single script (db.py) and symbolic links to it for alternative uses: db_ran.py and db_nonran.py. By default, the script implements random_toolchanger functionality. Nonrandom toolchanger functions are substituted if the link name includes the text "*nonran*".

The sim configs demonstrate the use of the Python *tooldb* interface module and implement a basic flat-file database that tracks tool time usage for multiple tools having equal diameters. The database rules support selection of the tool having the lowest operating time.

The sim configs use a primary task to monitor and respond to tool updates initiated from within LinuxCNC. A periodic task updates tool time usage at reguar intervals. Separate, concurrent tasks are implemented as threads to demonstrate the code required when changes are initiated by the **db_program** and demonstrate methods for synchronizing LinuxCNC internal tooldata. Examples include:

1. Aktualisierung der Werkzeugparameter
2. addition and removal of tool numbers

A mutual exclusion lock is used to protect data from inconsistencies due to race conditions between LinuxCNC tooldata updates and the database application updates.

9.10.2.1 Anmerkungen

When a **db_program** is used in conjunction with a random tool changer ([EMCIO]RANDOM_TOOLCHANGE
LinuxCNC maintains a file (*db_spindle.tbl* in the configuration directory) that consists of a single tool table line identifying the current tool in the spindle.

Teil II

Anwendung

Kapitel 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.

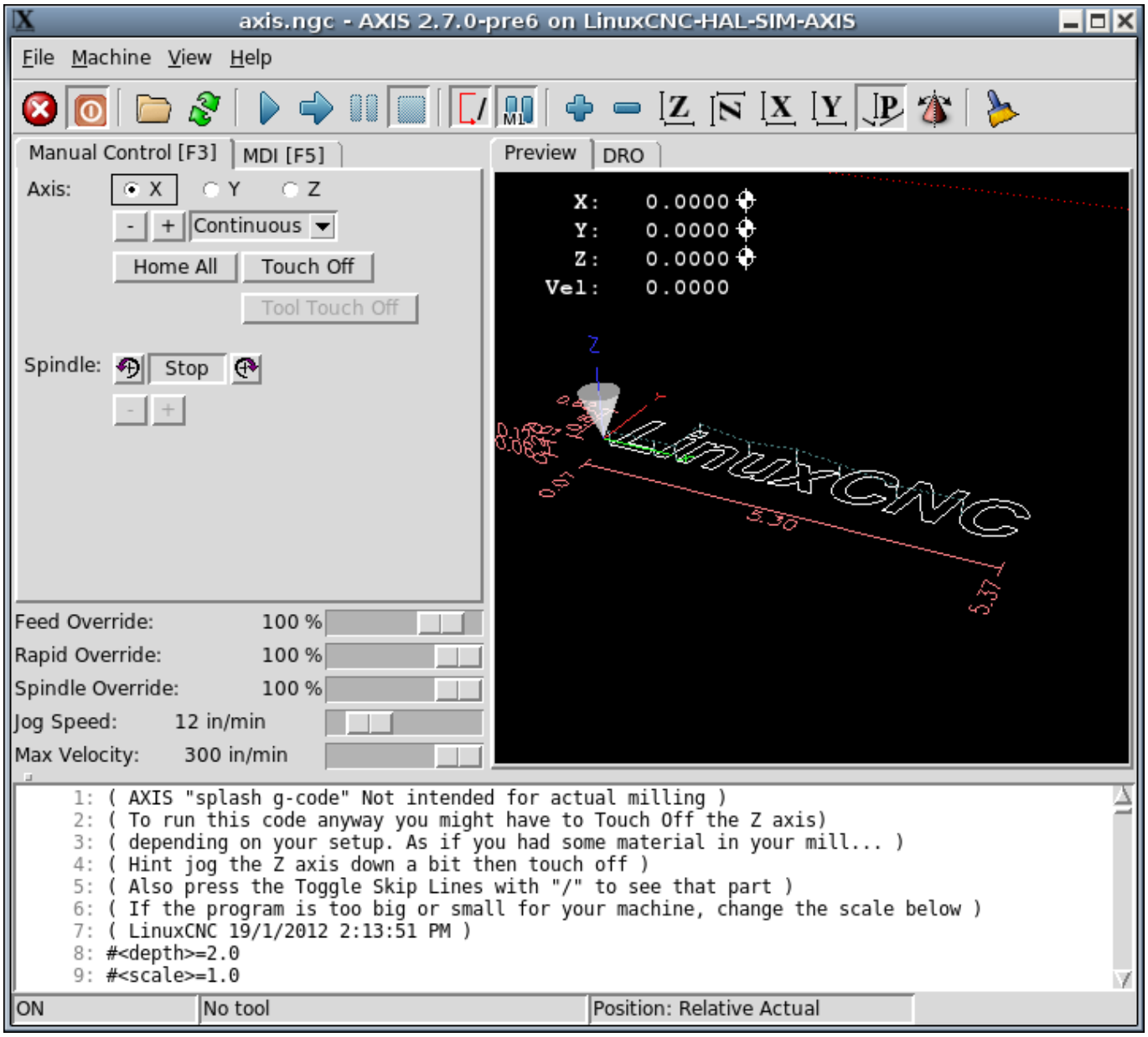


Abbildung 10.1: Das AXIS-Fenster

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 Abbildung 10.1 oben.

10.1.2.1 INI-Einstellungen

Weitere Informationen zur Einstellung der Funktionsweise von AXIS in der INI-Datei, finden Sie im Abschnitt Anzeige 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
```

10.1.2.2 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.

Anmerkung
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.
- Eine Statusleiste, die den Zustand der Maschine anzeigt. In diesem Screenshot ist die Maschine eingeschaltet, hat kein Werkzeug eingesetzt und die angezeigte Position ist ”Relativ” (mit allen Offsets) und ”Aktuell” (zeigt die Feedback-Position).

10.1.3.1 Menüpunkte

Einige Menüelemente sind möglicherweise ausgegraut, je nachdem, wie Sie Ihre INI-Datei konfiguriert haben. Weitere Informationen zur Konfiguration finden Sie im Kapitel INI.

- *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.
- *Eigenschaften* - Die Summe der Eilgang- und Vorschubbewegungen. Berücksichtigt keine Beschleunigung, Überblendung oder den Pfadmodus, sodass die gemeldete Zeit nie weniger als die tatsächliche Laufzeit ist.
- *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.

- *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.

Warnung
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.

- *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”.
- ”Über AXIS” - Wir alle wissen, was das ist.
- *Schnellübersicht* - Zeigt die Tastenkombinationen an.

10.1.3.2 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)
-  Zoom Out
-  Top view
-  Gedrehte Draufsicht
-  Side view
-  Front view
-  Perspektivische Ansicht
-  Umschalten zwischen Ziehen und Drehen [D]
-  Live-Backplot löschen [Strg-K]

10.1.3.3 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.

Abbildung 10.2: 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.

10.1.3.4 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.

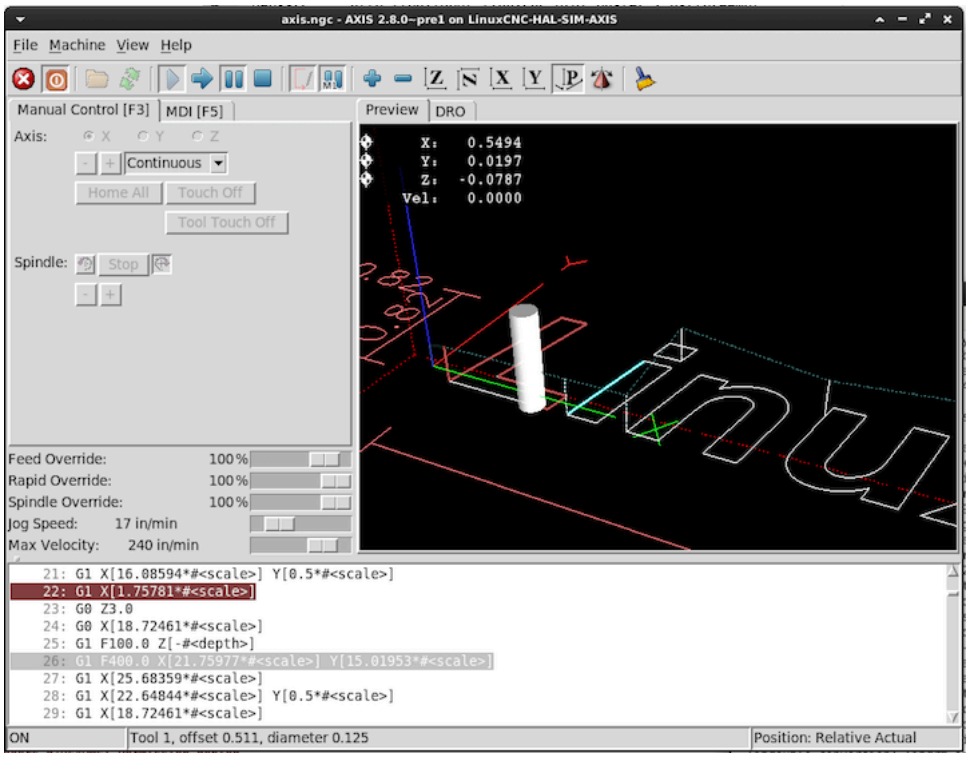


Abbildung 10.3: Aktuelle und ausgewählte Zeilen

10.1.3.5 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.

Wenn Ihre Maschine keine Home-Schalter in der Konfiguration definiert hat, setzt die Schaltfläche ”Home” die aktuelle Position der ausgewählten Achse als absolute Position 0 für diese Achse und setzt das Bit ”is-homed” für diese Achse.

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.

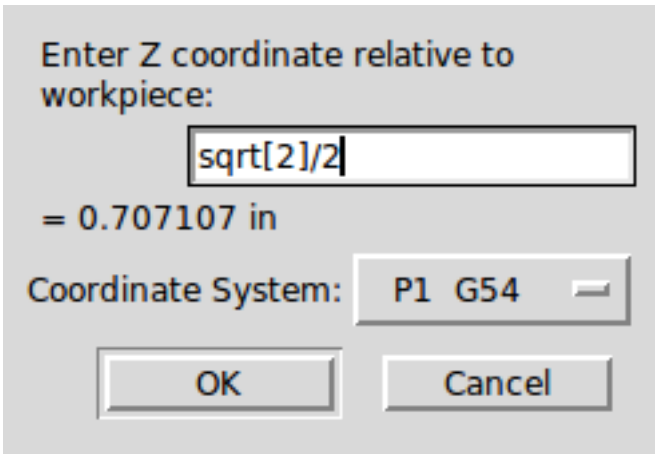


Abbildung 10.4: Touch Off Fenster

Siehe auch die Optionen im Menü Maschine: ”Werkstück berühren” und ”Werkstückhalter berühren”.

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.

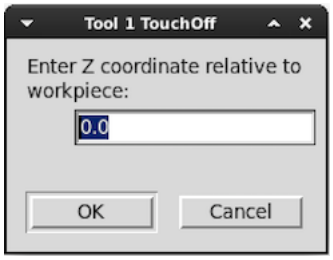


Abbildung 10.5: Werkzeug Touch Off Fenster

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.

10.1.3.6 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.

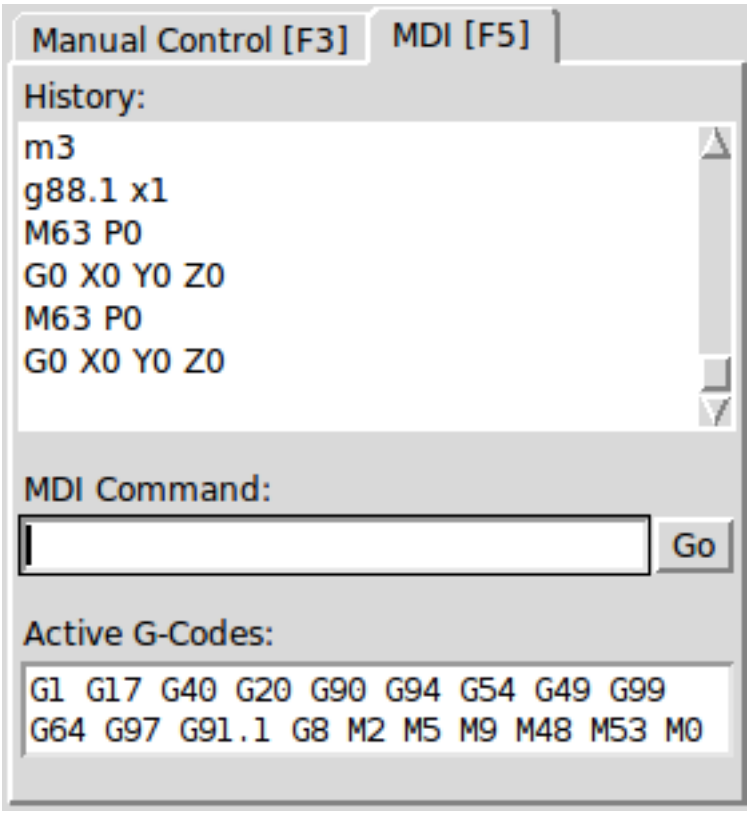


Abbildung 10.6: Die MDI-Registerkarte

- *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.

10.1.3.7 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.

10.1.3.8 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.

10.1.3.9 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.

10.1.3.10 Max. Geschwindigkeit

Durch Verschieben dieses Schiebereglers kann die maximale Geschwindigkeit eingestellt werden. Damit wird die maximale Geschwindigkeit für alle programmierten Bewegungen außer spindelsynchronisierten Bewegungen begrenzt.

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.

10.1.4.1 Vorschub-Neufestsetzung (engl. override)-Tasten

Anmerkung
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:

Tabelle 10.1: Häufigste Tastaturkürzel

Tastenkombination	Ergriffene Maßnahmen	Modus
F1	Notaus ein-/ausschalten	Any
F2	Maschine ein-/ausschalten	Any
`, 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

Tabelle 10.1: (continued)

Tastenkombination	**Ergriffene Maßnahmen**	**Modus**
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	Any
#	Umschalten Relativ/Maschine	Any

10.1.5 Show LinuxCNC Status (`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

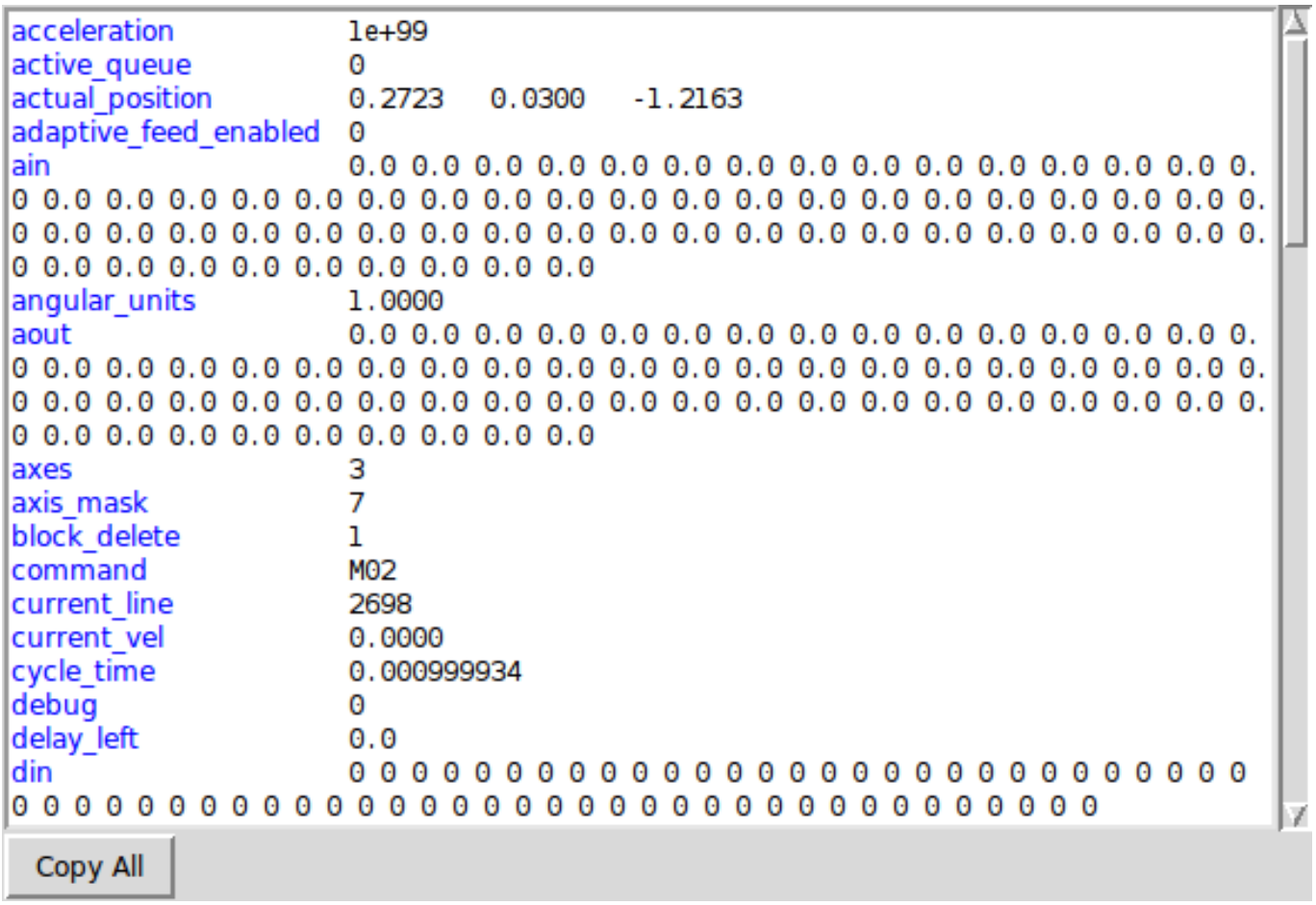


Abbildung 10.7: LinuxCNC-Statusfenster

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`, das die Eingabe von MDI-Befehlen im Textmodus in eine laufende LinuxCNC-Sitzung ermöglicht. Sie können dieses Programm 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 Userspace-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.

Anmerkung
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>.

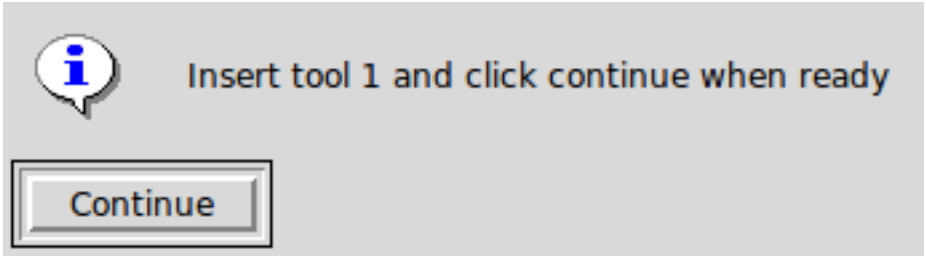


Abbildung 10.8: Fenster für manuellen Werkzeugwechsel

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 Userspace-HAL-Komponenten
- *_togl* stellt ein OpenGL-Widget bereit, das in Tkinter-Anwendungen verwendet werden kann
- *minigl* bietet Zugriff auf die von AXIS verwendete Teilmenge von OpenGL

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 Verwendung 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.

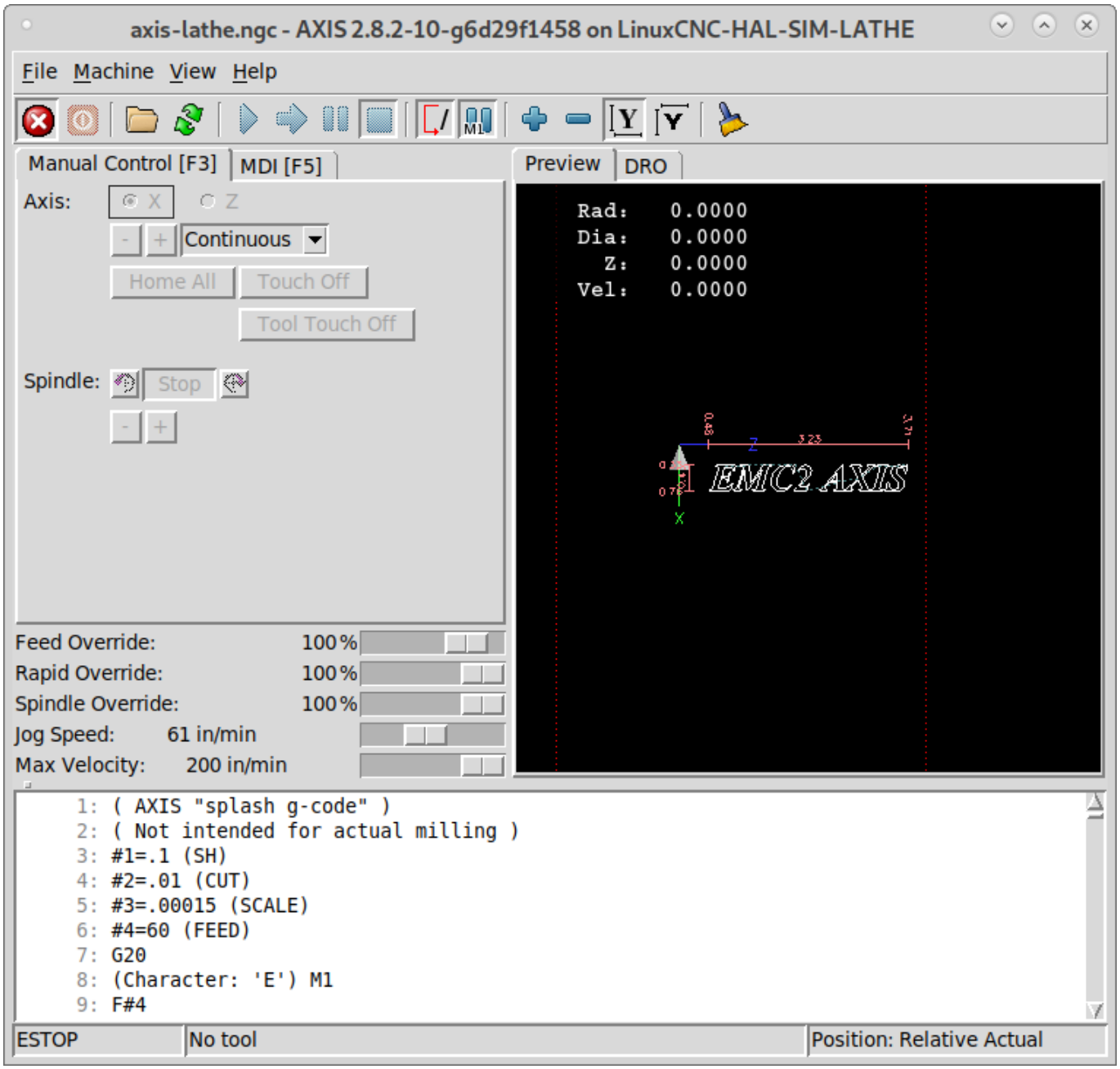


Abbildung 10.9: AXIS-Drehmaschinenmodus

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.

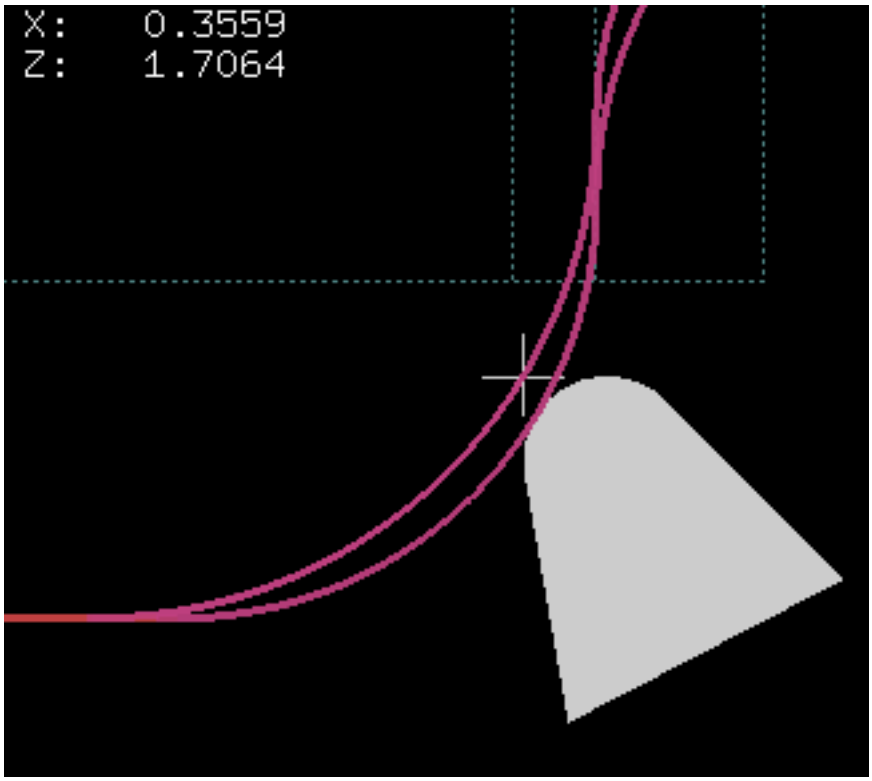


Abbildung 10.10: Drehwerkzeug-Form

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.

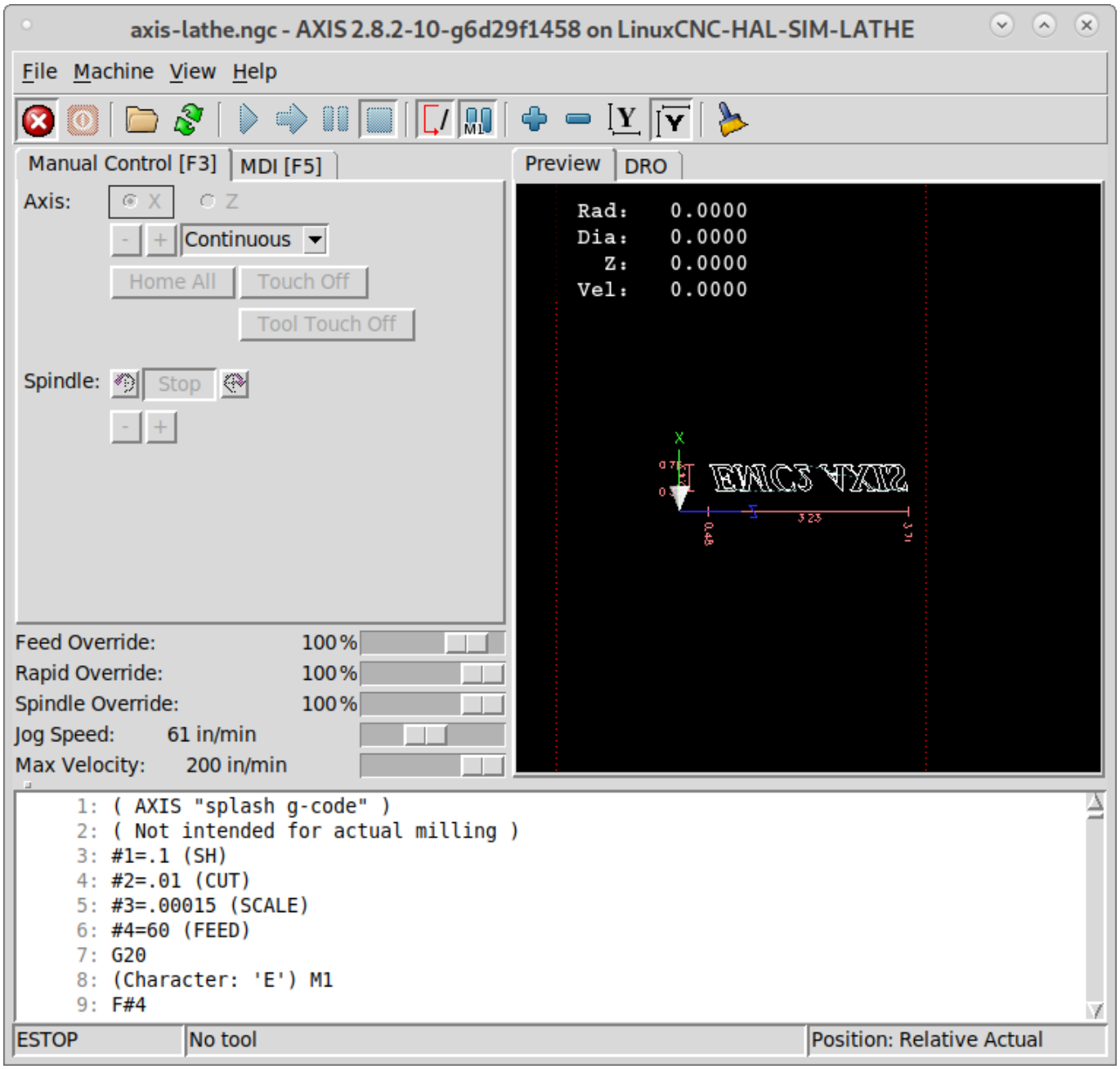


Abbildung 10.11: Lathe Back Tool Shape

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.

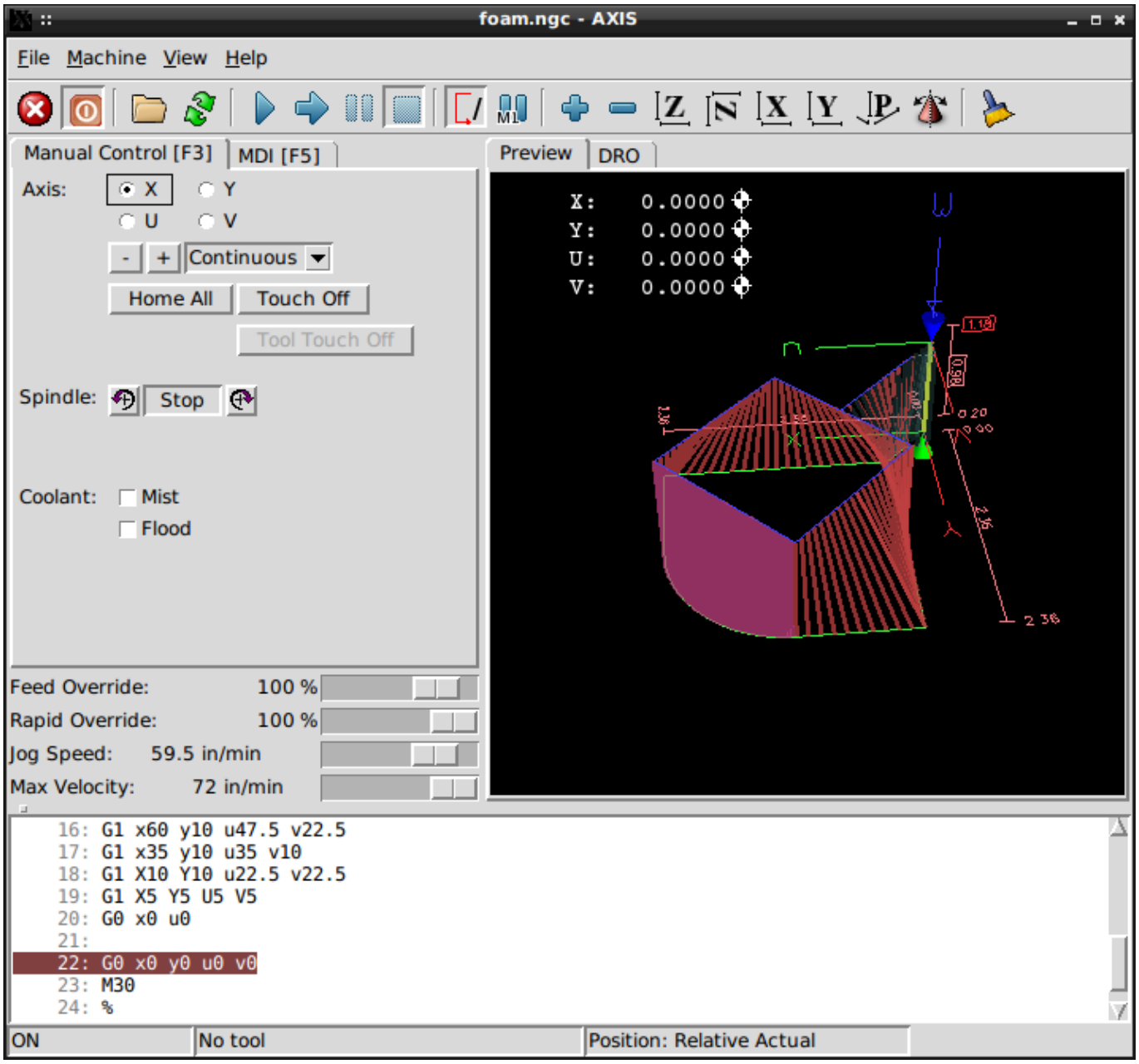


Abbildung 10.12: Modus Schaumstoffschneiden

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.

10.1.12.1 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.

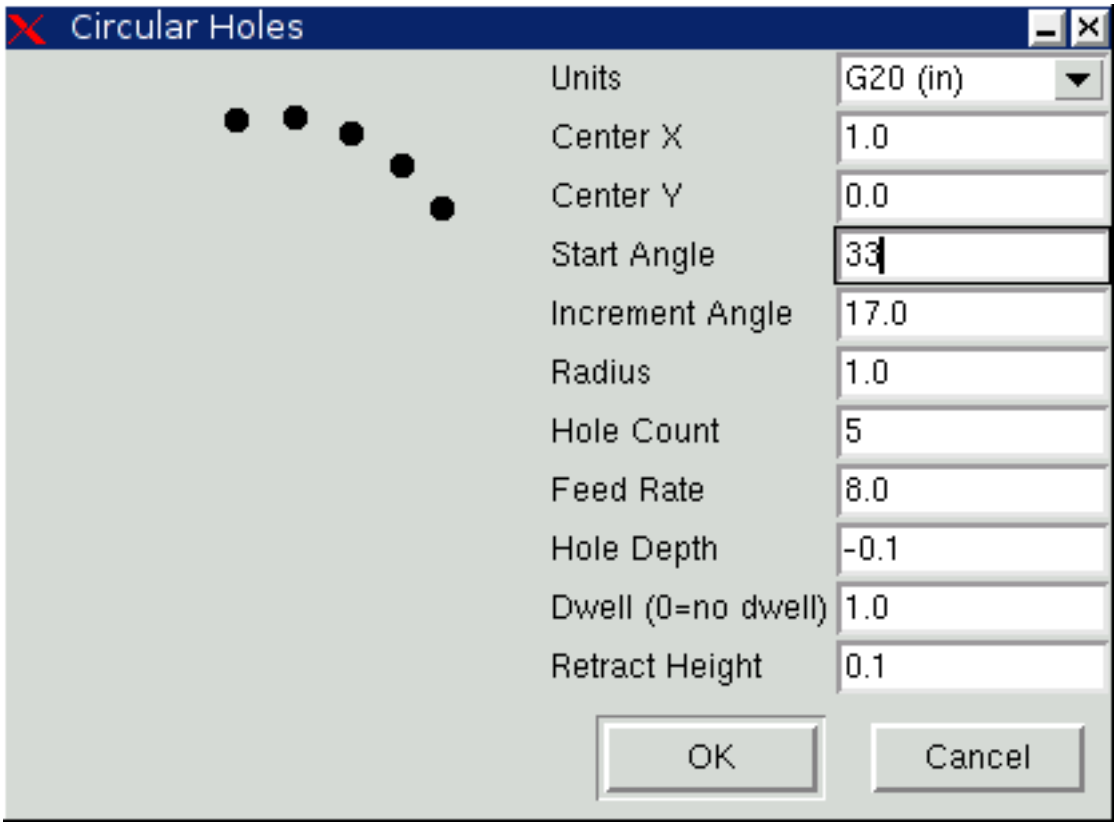


Abbildung 10.13: Kreisförmige Löcher

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.

10.1.12.2 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.

10.1.12.3 Handrad (engl. 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.

10.1.12.4 ~/.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 .”)
```

10.1.12.5 USER_COMMAND_FILE

A configuration-specific Python file may be specified with an INI file setting *[DISPLAY]USER_COMMAND_FI*
Like a `~/.axisrc` file, this file is sourced just before the AXIS GUI is displayed. This file is specific to an INI file configuration not the user's home directory.

10.1.12.6 user_live_update()

The AXIS GUI includes a no-op (placeholder) function named *user_live_update()* that is executed at the conclusion of the update() function of its LivePlotter class. This function may be implemented within a `~/.axisrc` Python script or a *[DISPLAY]USER_COMMAND_FILE* Python script to make custom, periodic actions. The details of what may be accomplished in this function are dependent on the AXIS GUI implementation and subject to change during the development cycle.

10.1.12.7 user_hal_pins()

The AXIS GUI includes a no-op (placeholder) function named *user_hal_pins()*.
It is executed just after the .axisrc file is called and just before any GladeVCP panels / embedded tabs are initialized.
This function may be implemented within a `~/.axisrc` Python script or a *[DISPLAY]USER_COMMAND_FILE* Python script to make custom HAL pins that use the *axisui.* prefix.
Use *comp* as the HAL component instance reference.
HAL comp.ready() is called just after this function returns.

10.1.12.8 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.

10.1.12.9 Virtuelles Bedienfeld (engl. virtual control panel, kurz VCP)

AXIS kann ein benutzerdefiniertes virtuelles Bedienfeld entweder in der rechten Spalte oder in der unteren Zeile anzeigen. Zusätzlich können ein oder mehrere Bedienfelder als eingebettete Registerkarten angezeigt werden. Sie können Schaltflächen, Indikatoren, Datenanzeigen und mehr programmieren. Weitere Informationen finden Sie in den Kapiteln PyVCP und GladeVCP.

10.1.12.10 Vorschau-Steuerung

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.

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
bit  OUT  axisui.jog.x
bit  OUT  axisui.jog.y
bit  OUT  axisui.jog.z
bit  OUT  axisui.jog.a
bit  OUT  axisui.jog.b
bit  OUT  axisui.jog.c
bit  OUT  axisui.jog.u
bit  OUT  axisui.jog.v
bit  OUT  axisui.jog.w
```

AXIS verfügt über einen HAL-Pin zur Anzeige der auf der Registerkarte ”Manuell” ausgewählten Schrittweite.

```
Type  Dir Name
float 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
bit   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
bit   OUT    axisui.error
```

AXIS verfügt über HAL-Eingangspins, um die Popup-Benachrichtigungen nach Fehlern und Informationen zu löschen.

```
Type  Dir    Name
bit   IN     axisui.notifications-clear
bit   IN     axisui.notifications-clear-error
bit   IN     axisui.notifications-clear-info
```

AXIS verfügt über einen HAL-Eingangspin, der die Funktion ”Pause/Resume” deaktiviert/aktiviert.

```
Type  Dir    Name
bit   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.

10.1.14.1 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...')
```

10.1.14.2 Deaktivieren des Schließen-Dialogs

```
# Deaktivieren Sie den "Do you want to close"-Dialog
root_window.tk.call("wm", "protocol",".", "WM_DELETE_WINDOW", "destroy .")
```

10.1.14.3 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')
```

10.1.14.4 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))
```

10.1.14.5 Lesen der INI-Datei

```
# INI-Dateielement lesen
machine = inifile.find(’EMC’,’MACHINE’)
print(’machine name =’,machine)
```

10.1.14.6 Read LinuxCNC Status

```
# LinuxCNC status can be read from s.
print(s.actual_position)
print(s.paused)
```

10.1.14.7 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()
```

10.1.14.8 Erstellen neuer AXISUI HAL-Pins

```
def user_hal_pins():
    comp.newpin(’my-new-in-pin’, hal.HAL_BIT, hal.HAL_IN)
    comp.ready()
```

10.1.14.9 Neue HAL-Komponente und Pins erstellen

```
# Komponente erstellen

mycomp = hal.component('meine_Komponente')
mycomp.newpin('idle-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.newpin('pause-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.ready()

# Pins verbinden

hal.new_sig('idle-led',hal.HAL_BIT)
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)
```

10.1.14.10 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
```

10.1.14.11 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)
```

10.1.14.12 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)
```

10.1.14.13 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
    tool                    = StringVar
    touch_off_system        = StringVar
    trajcoordinates         = StringVar
    tto_g11                 = BooleanVar
    view_type               = IntVar
```

10.1.14.14 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')
```

10.1.14.15 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')
```

10.1.14.16 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)
```

10.1.14.17 Ändern Sie die DRO-Farbe

```
# dro-Bildschirm ändern
root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','-
    background','black')
```

10.1.14.18 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)
```

10.1.14.19 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

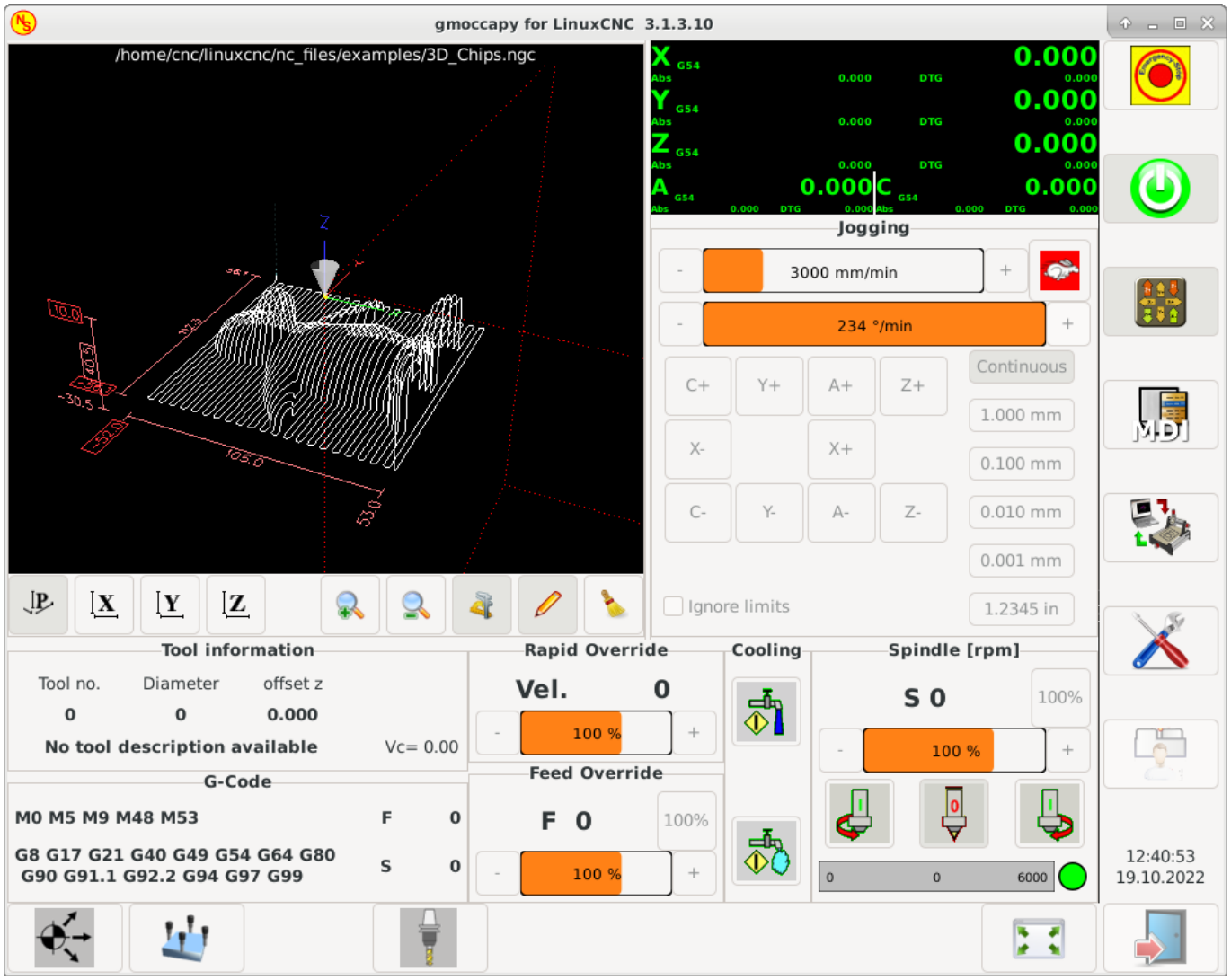
GMOCCAPY ist eine grafische Benutzeroberfläche für LinuxCNC, die für die Verwendung mit einem Touchscreen entwickelt wurde, aber auch auf normalen Bildschirmen mit einer Maus oder Hardware-Tasten und MPG-Rädern verwendet werden kann, da sie HAL-Pins für die häufigsten Anforderungen bereitstellt. Weitere 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 notwendig ist, nicht benötigte Dinge zu übersetzen. Die Dateien befinden sich in **/src/po/gmoccapy**. Du könntest einfach die Datei gmoccapy.pot in etwas wie it.po kopieren und diese Datei mit gtranslator oder poedit übersetzen. Nach dem Neuaufbau erhalten Sie die GUI in der von Ihnen bevorzugten Sprache. Um die Weitergabe der Übersetzung zu erleichtern, ist GMOCCAPY auf Weblate web interface verfügbar. GMOCCAPY ist derzeit in Englisch, Deutsch, Spanisch, Polnisch, Serbisch und Ungarisch verfügbar. Sie können mir gerne helfen, weitere Sprachen einzuführen, sei es lokal oder über das Web. Wenn Sie Hilfe benötigen, zögern Sie nicht, mich unter **nieson@web.de** zu kontaktieren.



10.2.2 Anforderungen

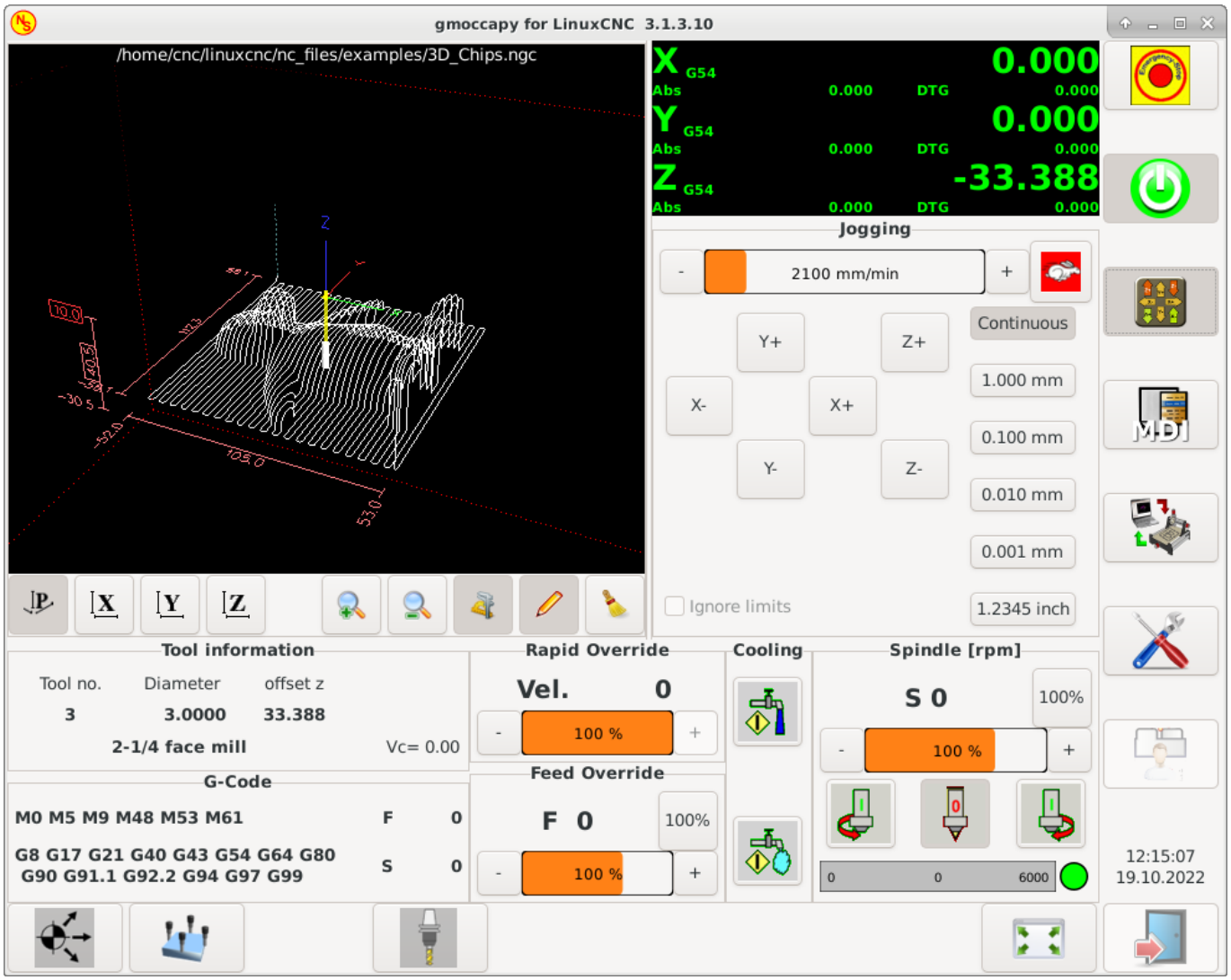
GMOCCAPY 3 wurde auf Debian Jessie, Debian Stretch und MINT 18 mit LinuxCNC Master und 2.8 Release getestet. Es unterstützt vollständig Gelenk / Achse Änderungen von LinuxCNC, so dass es geeignet als GUI für Scara, Roboter oder jede andere Konfiguration mit mehr Gelenke als Achse. Wenn Sie andere Versionen 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.

10.2.3 So erhalten Sie GMOCCAPY

GMOCCAPY 3 is included in the standard distribution of LinuxCNC since release 2.8. So the easiest way to get GMOCCAPY on your controlling PC, is just to download the ISO and install it from the CD/DVD/USB-stick.
This allows you to receive updates with the regular Debian packages.

Sie erhalten einen ähnlichen Bildschirm wie den folgenden (das Design kann je nach Ihrer Konfiguration variieren):



10.2.4 Basiseinstellung

GMOCCAPY 3 unterstützt die folgenden Befehlszeilenoptionen:

- *-user_mode*: If set, the setup button will be disabled, so normal machine operators are not able to edit the settings of the machine.
- *-logo <path to logo file>*: If given, the logo will hide the jog button tab in manual mode, this is only useful for machines with hardware buttons for jogging and increment selection.

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.

Wichtig
Wenn Sie MACROS verwenden wollen, vergessen Sie nicht, den Pfad zu Ihrem Makro- oder Subroutinen-Ordner wie unten beschrieben anzugeben.

Schauen wir uns also die INI-Datei genauer an und was Sie einfügen müssen, um GMOCCAPY auf Ihrem Rechner zu verwenden:

10.2.4.1 Die DISPLAY-Sektion

```
[DISPLAY]
DISPLAY = gmoccapy
PREFERENCE_FILE_PATH = gmoccapy_preferences
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
LATHE = 1
BACK_TOOL_LATHE = 1
PROGRAM_PREFIX = ../../nc_files/
```

- *DISPLAY = gmoccapy* - This tells LinuxCNC to use GMOCCAPY.
- *PREFERENCE_FILE_PATH* - Gives the location and name of the preferences file to be used. In most cases this line will not be needed, it is used by GMOCCAPY to store your settings of the GUI, like themes, DRO units, colors, and keyboard settings, etc., see settings page for more details.

Anmerkung
If no path or file is given, GMOCCAPY will use as default <your_machinename>.pref, if no machine name is given in your INI File it will use gmoccapy.pref. The file will be stored in your config directory, so the settings will not be mixed if you use several configs. If you only want to use one file for several machines, you need to include `PREFERENCE_FILE_PATH` in your INI.

- *MAX_FEED_OVERRIDE = 1.5* - Sets the maximum feed override, in the example given, you will be allowed to override the feed by 150%.

Anmerkung
Wenn kein Wert angegeben wird, so wird er auf 1,0 gesetzt.

- *MIN_SPINDLE_OVERRIDE = 0.5* and *MAX_SPINDLE_OVERRIDE = 1.2* - Will allow you to change the spindle override within a limit from 50% to 120%.

Anmerkung
If no values are given, MIN will be set to 0.1 and MAX to 1.0.

- *LATHE = 1* - Set the screen layout to control a lathe.
- *BACK_TOOL_LATHE = 1* - Is optional and will switch the X axis in a way you need for a back tool lathe. Also the keyboard shortcuts will react in a different way. It is allowed with GMOCCAPY to configure a lathe also with additional axes, so you may use also a XZCW config for a lathe.

Tipp
Siehe auch den Abschnitt speziell zu Drehmaschinen.

- *PROGRAM_PREFIX = ../../nc_files/* - Is the entry to tell LinuxCNC/GMOCCAPY where to look for the NGC files.

Anmerkung
If not specified, GMOCCAPY will look in the following order for NGC files: First `linuxcnc/nc_files` and then the users home directory.

10.2.4.2 Der TRAJ Abschnitt

- *DEFAULT_LINEAR_VELOCITY = 85.0* - Sets the default jog velocity of the machine.

Anmerkung
If not set, half of *MAX_LINEAR_VELOCITY* will be used. If that value is also not given, it will default to 180.

- *MAX_LINEAR_VELOCITY = 230.0* - Sets the maximal velocity of the machine.

Anmerkung
Defaults to 600 if not set.

10.2.4.3 Makro-Buttons

You can add macros to GMOCCAPY, similar to Touchy's way. A macro is nothing else than a NGC file. You are able to execute complete CNC programs in MDI mode by just pushing one button. To do so, you first have to specify the search path for macros:

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

This sets the path to search for macros and other subroutines. Several subroutine paths can be separeted ":".

Then you just have to add a section like this:

Configuration of Five Macros to be Shown in the MDI Button List

```
[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.
- The files need to be placed in a folder specified in your INI file by *SUBROUTINE_PATH* in the RS274NGC section

Der Code zwischen sub und endsub wird durch Betätigen der entsprechenden Makrotaste ausgeführt.

Anmerkung
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.

Anmerkung
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:

```
[MACRO]
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.n
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>

(DBG, 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.

Anmerkung
Wenn Sie ein Makro ohne Bewegung verwenden möchten, beachten Sie auch die Hinweise in bekannte Probleme.

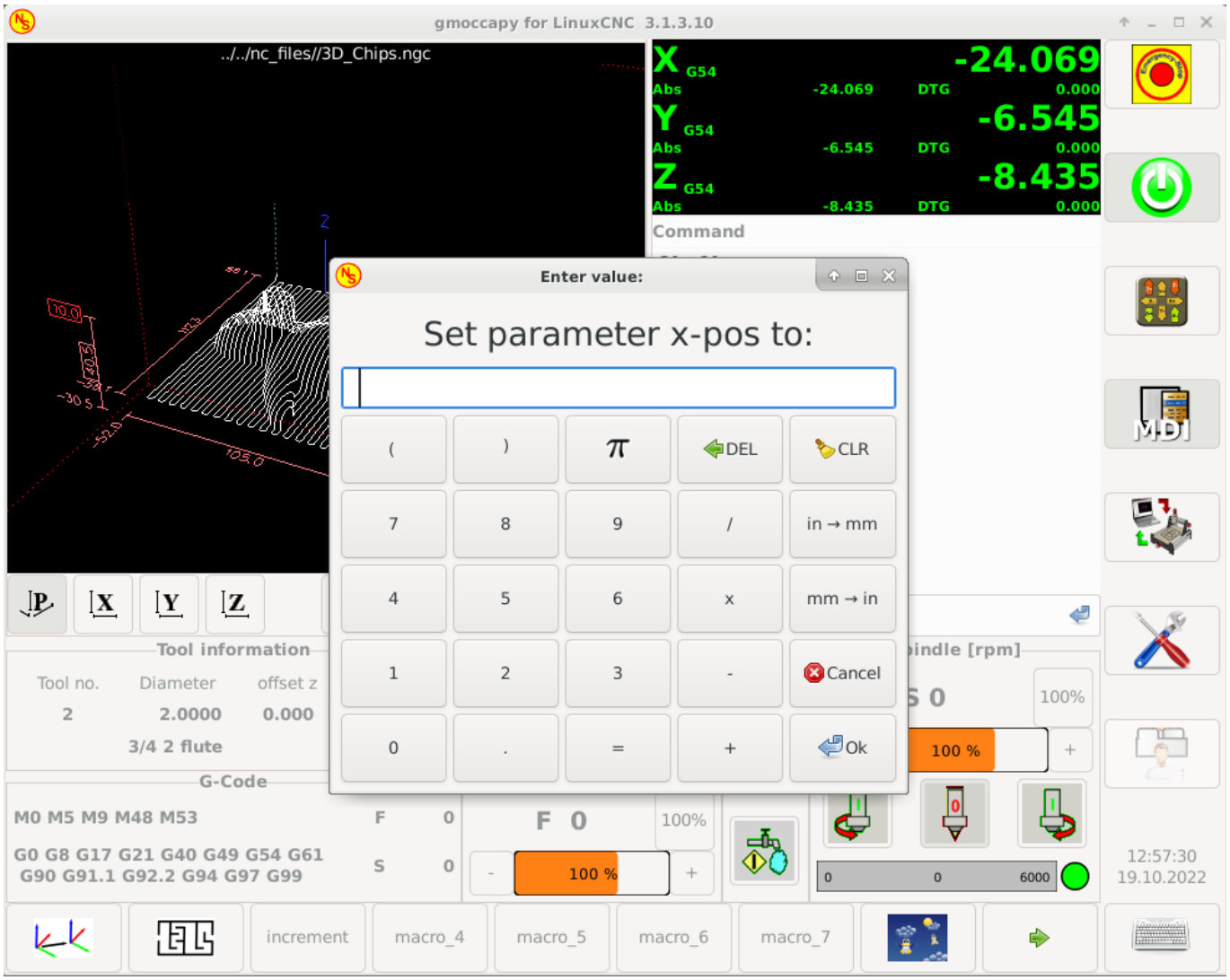


Abbildung 10.14: Makrobeispiel mit dem ”Gehe zu Position”-Makro

10.2.4.4 Embedded Tabs and Panels

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.

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)
  - **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)
  - **box_code_info** (blendet den G-Code-Informationsrahmen aus und fügt hier Ihre Glade-Datei ein)
  - **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)

Anmerkung
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.

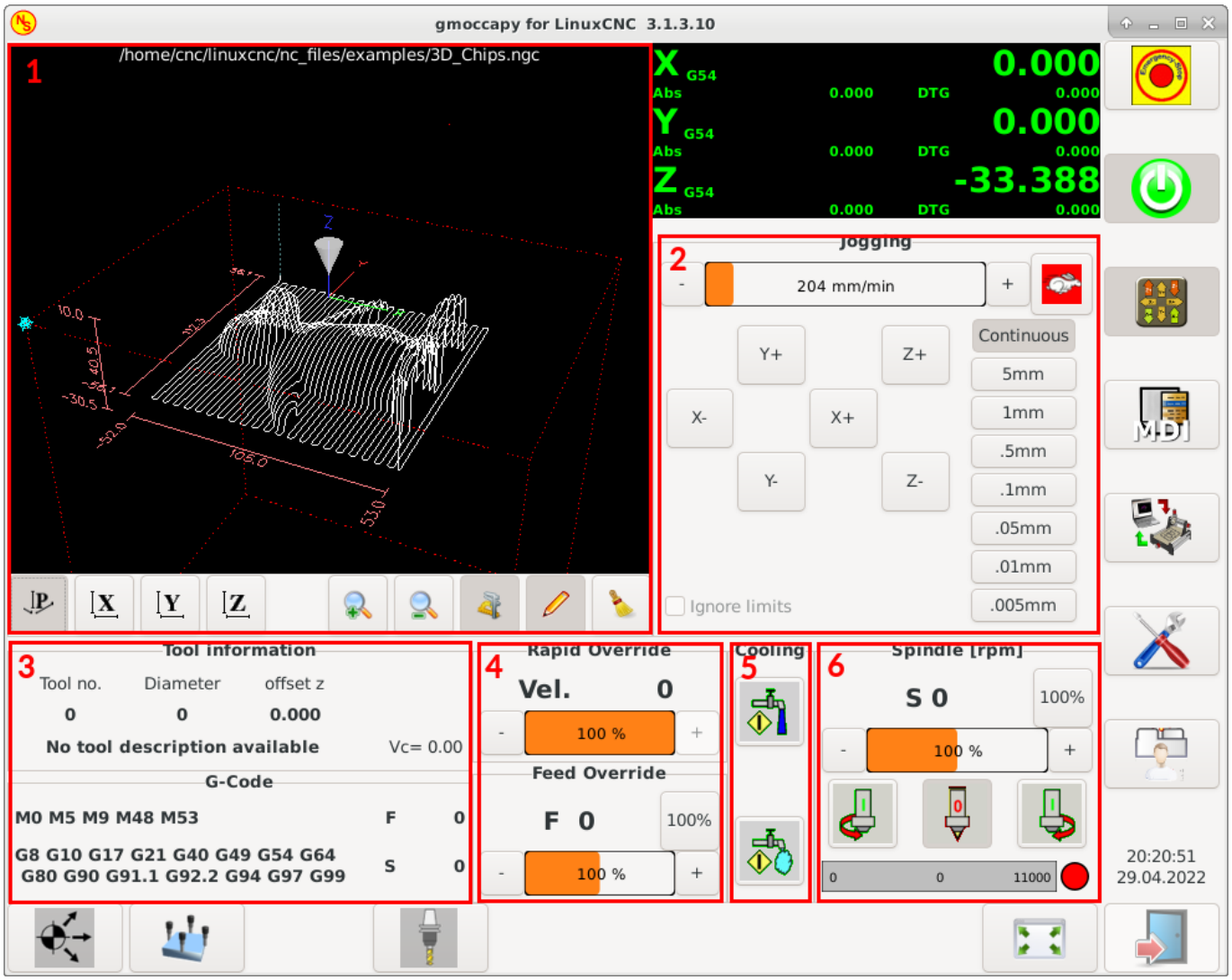


Abbildung 10.15: Eingebettete Registerkartenpositionen

Anmerkung
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:

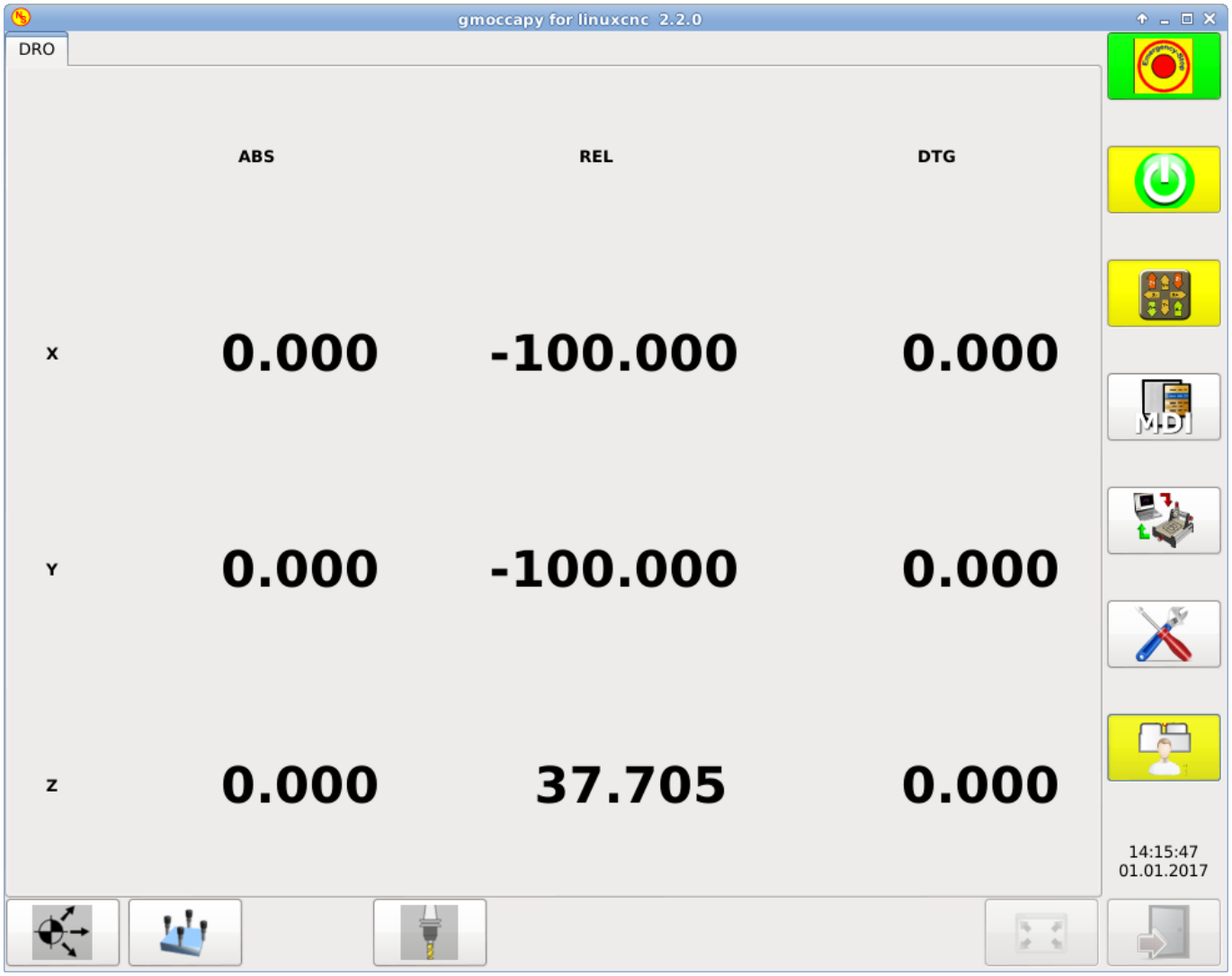


Abbildung 10.16: ntb_preview - in maximierter Ansicht

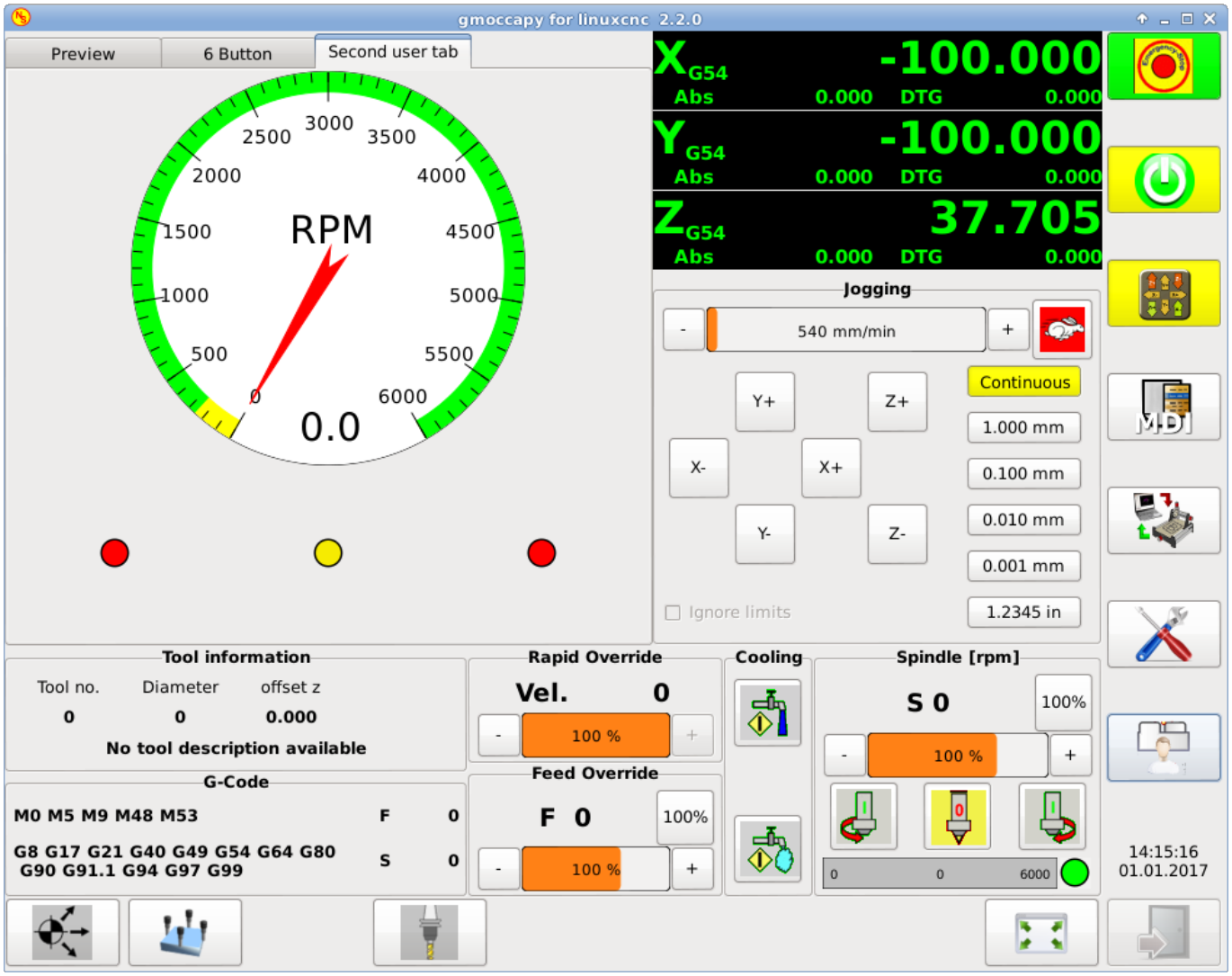


Abbildung 10.17: ntb_preview

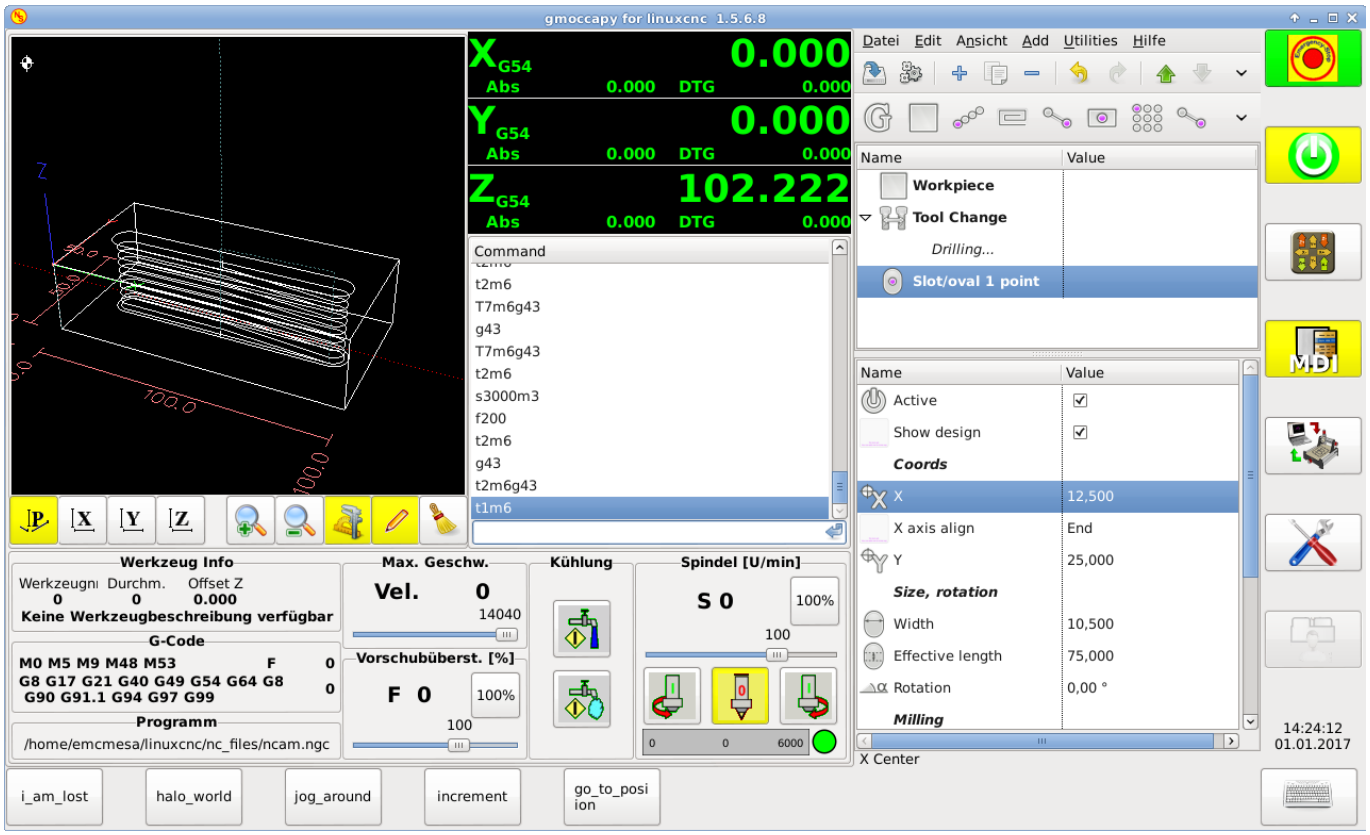


Abbildung 10.18: box_right - und GMOCCAPY im MDI-Modus

10.2.4.5 User Created Messages

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.

For more detailed information of the pins see User Created Message HAL Pins.

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
```

Anmerkung
Derzeit funktioniert die Formatierung nicht.

10.2.4.6 Vorschau Kontrolle

Magic comments can be used to control the G-code preview. On very large programs the preview can take a long time to load. You can control what is shown and what is hidden on the graphics screen by adding the appropriate comments from this list into your gcode:

```
(PREVIEW,hide)
<G-code to be hidden>
(PREVIEW,show)
```

10.2.5 HAL-Pins

GMOCCAPY exports several HAL pins to be able to react to hardware devices. The goal is to get a GUI that may be operated in a tool shop, completely/mostly without mouse or keyboard.

Anmerkung
You will have to do all connections to GMOCCAPY pins in your postgui.hal file. When GMOCCAPY is started, it creates the HAL pins for the GUI then it executes the post-GUI HAL file named in the INI file:

```
[HAL]
POSTGUI_HALFILE=<filename>
```

Typically `<filename>` would be the configs base name + `_postgui.hal`, e.g. `lathe_postgui.hal`, but can be any legal filename.
These commands are executed after the screen is built, guaranteeing the widget's HAL pins are available.
You can have multiple line of `POSTGUI_HALFILE=<filename>` in the INI file. Each will be run one after the other in the order they appear.

10.2.5.1 Right and Bottom Button Lists

The screen has two main button lists, one on the right side an one on the bottom. The right handed buttons will not change during operation, but the bottom button list will change very often. The buttons are count from up to down and from left to right beginning with 0.

Anmerkung
The pin names have changed in GMOCCAPY 2 to order them in a better way.

The pins for the right (vertical) buttons are:

- **gmoccapy.v-button.button-0** *(bit IN)*
- **gmoccapy.v-button.button-1** *(bit IN)*
- **gmoccapy.v-button.button-2** *(bit IN)*
- **gmoccapy.v-button.button-3** *(bit IN)*
- **gmoccapy.v-button.button-4** *(bit IN)*
- **gmoccapy.v-button.button-5** *(bit IN)*
- **gmoccapy.v-button.button-6** *(bit IN)*

For the bottom (horizontal) buttons they are:

- **gmoccapy.h-button.button-0** *(bit IN)*
- **gmoccapy.h-button.button-1** *(bit IN)*
- **gmoccapy.h-button.button-2** *(bit IN)*
- **gmoccapy.h-button.button-3** *(bit IN)*
- **gmoccapy.h-button.button-4** *(bit IN)*
- **gmoccapy.h-button.button-5** *(bit IN)*
- **gmoccapy.h-button.button-6** *(bit IN)*
- **gmoccapy.h-button.button-7** *(bit IN)*
- **gmoccapy.h-button.button-8** *(bit IN)*
- **gmoccapy.h-button.button-9** *(bit IN)*

As the buttons in the bottom list will change according to the mode and other influences, the hardware buttons will activate the displayed functions. So you don’t have to take care about switching functions around in HAL, because that is done completely by GMOCCAPY!

For a three axes XYZ machine the HAL pins will react as shown in the following three tables:

Tabelle 10.2: Functional assignment of horizontal buttons (1)

Pin	Manual Mode	MDI Mode	Auto Mode
gmoccapy.h-button.button-0	open homing button	macro_0 or nothing	open file
gmoccapy.h-button.button-1	open touch off stuff	macro_1 or nothing	reload program
gmoccapy.h-button.button-2		macro_2 or nothing	run
gmoccapy.h-button.button-3	open tool dialogs	macro_3 or nothing	stop
gmoccapy.h-button.button-4		macro_4 or nothing	pause

Tabelle 10.2: (continued)

Pin	Manual Mode	MDI Mode	Auto Mode
gmoccapy.h-button.button-5		macro_5 or nothing	step by step
gmoccapy.h-button.button-6		macro_6 or nothing	run from line if enabled in settings, otherwise Nothing
gmoccapy.h-button.button-7		macro_7 or nothing	optional blocks
gmoccapy.h-button.button-8	full-size preview	macro_8 or switch page to additional macros	full-size preview
gmoccapy.h-button.button-9	exit if machine is off, otherwise no reaction	open keyboard or abort if macro is running	edit code

Tabelle 10.3: Functional assignment of horizontal buttons (2)

Pin	Settings Mode	Homing Mode	Touch off Mode
gmoccapy.h-button.button-0	delete MDI history		edit offsets
gmoccapy.h-button.button-1		home all	touch X
gmoccapy.h-button.button-2			touch Y
gmoccapy.h-button.button-3		home x	touch Z
gmoccapy.h-button.button-4	open classic ladder	home y	
gmoccapy.h-button.button-5	open HAL scope	home z	
gmoccapy.h-button.button-6	open HAL status		zero G92
gmoccapy.h-button.button-7	open HAL meter		
gmoccapy.h-button.button-8	open HAL calibration	unhome all	set selected
gmoccapy.h-button.button-9	open HAL show	back	back

Tabelle 10.4: Functional assignment of horizontal buttons (3)

Pin	Tool Mode	Edit Mode	Select File
gmoccapy.h-button.button-0	delete tool(s)		go to home
gmoccapy.h-button.button-1	new tool	reload file	one directo

Tabelle 10.4: (continued)

Pin	Tool Mode	Edit Mode	Select File
gmoccapy.h-button.button-2	reload tool table	save	
gmoccapy.h-button.button-3	apply changes	save as	move select
gmoccapy.h-button.button-4	change tool by number T? M6		move select
gmoccapy.h-button.button-5	set tool by number without change M61 Q?		jump to dir
gmoccapy.h-button.button-6	change tool to the selected one	new file	
gmoccapy.h-button.button-7			select / EN
gmoccapy.h-button.button-8	touch of tool in Z	show keyboard	
gmoccapy.h-button.button-9	back	back	back

So we have 67 reactions with only 10 HAL pins!

These pins are made available to be able to use the screen without an touch panel, or protect it from excessive use by placing hardware buttons around the panel. They are available in a sample configuration like shown in the image below.

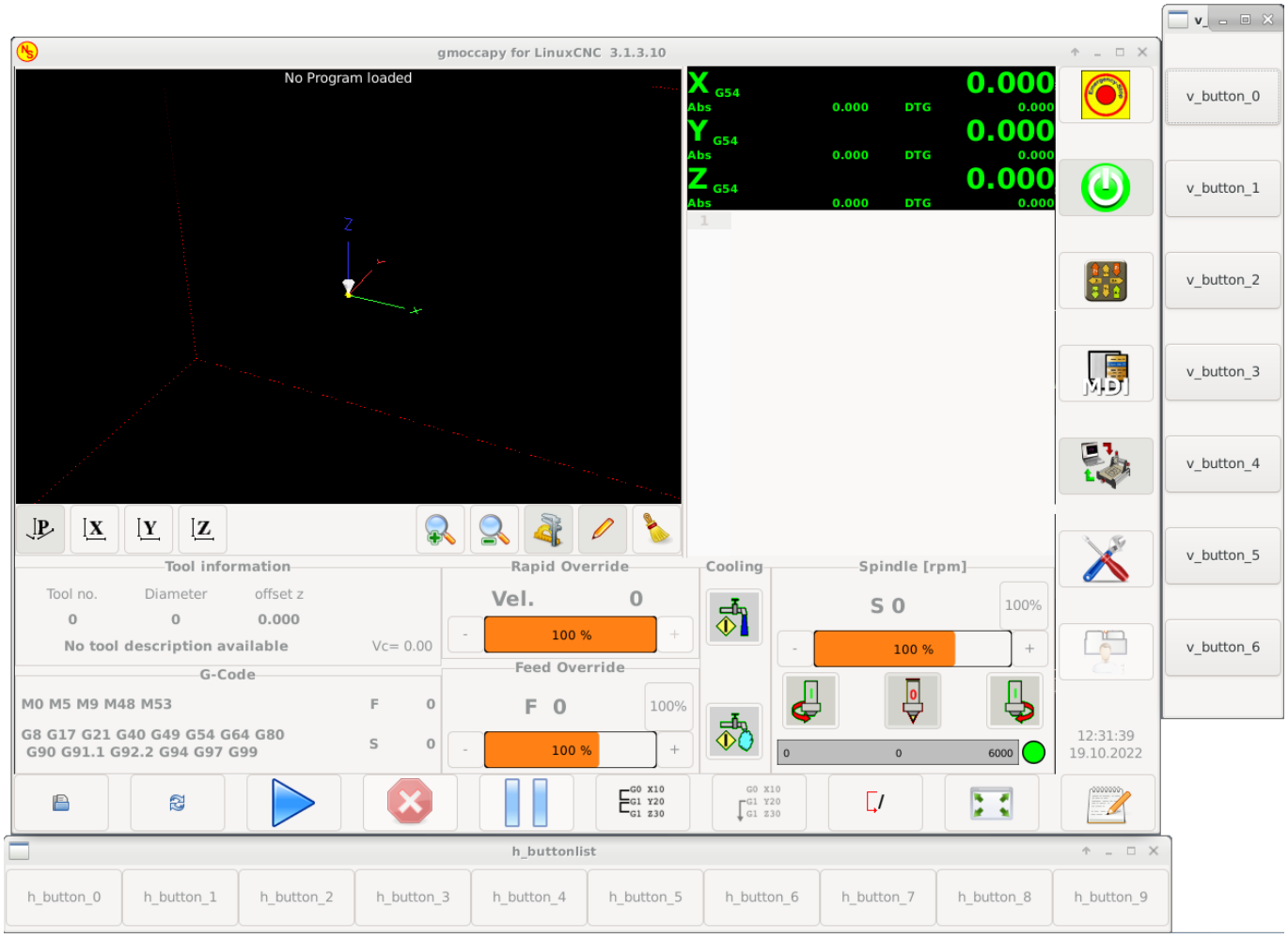


Abbildung 10.19: Sample configuration ”gmoccapy_sim_hardware_button” showing the buttons

10.2.5.2 Geschwindigkeiten und Neufestsetzungen/Übersteuerungen (engl. overrides)

All sliders from GMOCCAPY can be connected to hardware encoders or hardware potentiometers.

Anmerkung
For GMOCCAPY 3 some HAL pin names have changed when new controls have been implemented. Max velocity does not exist any more, it was replaced by rapid override due to the demand of many users.

To connect encoders, the following pins are exported:

- **gmoccapy.jog.jog-velocity.counts** *(s32 IN)* - Jog velocity
- **gmoccapy.jog.jog-velocity.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.feed.feed-override.counts** *(s32 IN)* - feed override
- **gmoccapy.feed.feed-override.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.feed.reset-feed-override** *(bit IN)* - reset the feed override to *0%

- **gmoccapy.spindle.spindle-override.counts** *(s32 IN)* - spindle override
- **gmoccapy.spindle.spindle-override.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.spindle.reset-spindle-override** *(bit IN)* - reset the spindle override to *0%
- **gmoccapy.rapid.rapid-override.counts** *(s32 IN)* - Maximal Velocity of the *chine
- **gmoccapy.rapid.rapid-override.count-enable** *(bit IN)* - Must be True, to enable counts

To connect potentiometers, use the following pins:

- **gmoccapy.jog.jog-velocity.direct-value** *(float IN)* - To adjust the jog velocity slider
- **gmoccapy.jog.jog-velocity.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.feed.feed-override.direct-value** *(float IN)* - To adjust the feed override slider
- **gmoccapy.feed.feed-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.spindle.spindle-override.direct-value** *(float IN)* - To adjust the spindle override slider
- **gmoccapy.spindle.spindle-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs
- **gmoccapy.rapid.rapid-override.direct-value** *(float)* - To adjust the max velocity slider
- **gmoccapy.rapid.rapid-override.analog-enable** *(bit IN)* - Must be True, to allow analog inputs

In addition, GMOCCAPY 3 offers additional HAL pins to control the new slider widgets with momentary switches. The values how fast the increase or decrease will be, must be set in the glade file. In a future release it will be integrated in the settings page.

GESCHWINDIGKEIT

- **gmoccapy.spc_jog_vel.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_jog_vel.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_jog_vel.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_jog_vel.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_jog_vel.scaled-value** *(float OUT)* - Scaled value of the widget .FEED
- **gmoccapy.spc_feed.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_feed.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_feed.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_feed.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_feed.scaled-value** *(float OUT)* - Scaled value of the widget .SPINDLE (engl. für Spindel)
- **gmoccapy.spc_spindle.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_spindle.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_spindle.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)

- **gmoccapy.spc_spindle.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_spindle.scaled-value** *(float OUT)* - Scaled value of the widget .RAPIDS
- **gmoccapy.spc_rapid.increase** *(bit IN)* - As long as True the value of the slider will increase
- **gmoccapy.spc_rapid.decrease** *(bit IN)* - As long as True the value of the slider will decrease
- **gmoccapy.spc_rapid.scale** *(float IN)* - A value to scale the output value (handy to change units/min to units/sec)
- **gmoccapy.spc_rapid.value** *(float OUT)* - Value of the widget
- **gmoccapy.spc_rapid.scaled-value** *(float OUT)* - Scaled value of the widget

The float pins do accept values from 0.0 to 1.0, being the percentage value you want to set the slider value.

Warnung
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.

Wichtig
Please be aware that the jog velocity depends on the turtle button state. It will lead to different slider scales depending on the mode (turtle or rabbit). Please take also a look at jog velocities and turtle-jog HAL pin for more details.

Beispiel 10.1 Setting a slider value

```
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 %
```

10.2.5.3 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.

Anmerkung
Naming of these HAL pins have changed in GMOCCAPY 2.

For the standard XYZ config following HAL pins will be available:

- **gmoccapy.jog.axis.jog-x-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-x-minus** *(bit IN)*

- **gmoccapy.jog.axis.jog-y-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-y-minus** *(bit IN)*
- **gmoccapy.jog.axis.jog-z-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-z-minus** *(bit IN)*

If you use a 4 axes configuration, there will be two additional pins:

- **gmoccapy.jog.jog-<your fourth axis letter >-plus** *(bit IN)*
- **gmoccapy.jog.jog-<your fourth axis letter >-minus** *(bit IN)*

For a C-axis you will see:

- **gmoccapy.jog.axis.jog-c-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-c-minus** *(bit IN)*

10.2.5.4 Jog Velocities and 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 offers this HAL pin to toggle between turtle and rabbit jogging:

- **gmoccapy.jog.turtle-jog** *(bit IN)*

10.2.5.5 Jog Increment HALPins

The jog increments given in thhe INI file like

```
[DISPLAY]
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm
```

are selectable through HAL pins, so a selection hardware switch can be used to select the increment to use. There will be a maximum of 10 HAL pins for the increments given in the INI file. If you give more increments in your INI file, they will be not reachable from the GUI as they will not be displayed.

If you have 6 increments in your INI file like in the example above, you will get **7** pins:

- **gmoccapy.jog.jog-inc-0** *(bit IN)* - This one is fixed and will represent continuous jogging.
- **gmoccapy.jog.jog-inc-1** *(bit IN)* - First increment given in the INI file.
- **gmoccapy.jog.jog-inc-2** *(bit IN)*
- **gmoccapy.jog.jog-inc-3** *(bit IN)*
- **gmoccapy.jog.jog-inc-4** *(bit IN)*
- **gmoccapy.jog.jog-inc-5** *(bit IN)*
- **gmoccapy.jog.jog-inc-6** *(bit IN)*

GMOCCAPY bietet auch einen HAL-Pin zur Ausgabe der gewählten Jog-Schrittweite:

- **gmoccapy.jog.jog-increment** *(float OUT)*

10.2.5.6 Hardware-Entsperr-Pin

To be able to use a key switch to unlock the settings page, the following pin is exported:

- **gmoccapy.unlock-settings** *(bit IN)* - The settings page is unlocked if the pin is high. To use this pin, you need to activate it on the settings page.

10.2.5.7 Fehler/Warnung Pins

- **gmoccapy.error** *(bit 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** *(bit IN)* - Will delete the first error and reset the `gmoccapy.error` pin to false after the last error has been cleared.
- **gmoccapy.warning-confirm** *(bit IN)* - Confirms warning dialog like click on OK

Anmerkung
Messages or user infos will not affect the `gmoccapy.error` pin, but the `gmoccapy.delete-message` pin will delete the last message if no error is shown!

10.2.5.8 Vom Benutzer erstellte HAL-Pins für Nachrichten

GMOCCAPY may be configured to react to external errors, using 3 different user messages:

status

- **gmoccapy.messages.status** *(bit IN)* - Triggers the dialog.

okdialog

- **gmoccapy.messages.okdialog** *(bit IN)* - Triggers the dialog.
- **gmoccapy.messages.okdialog-waiting** *(bit OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bit IN)* - Triggers the dialog.
- **gmoccapy.messages.yesnodialog-waiting** *(bit OUT)* - Will be *1* as long as the dialog is open. Closing the message will reset the this pin.
- **gmoccapy.messages.yesnodialog-response** *(bit 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.

10.2.5.9 Spindel-Feedback-Pins

Es gibt zwei Pins für das Spindelfeedback:

- **gmoccapy.spindle_feedback_bar** *(float IN)* - Pin to show the spindle speed on the spindle bar.
- **gmoccapy.spindle_at_speed_led** *(bit IN)* - Pin to lit the is-at-speed-led.

10.2.5.10 Pins zur Anzeige von Programmfortschrittsinformationen

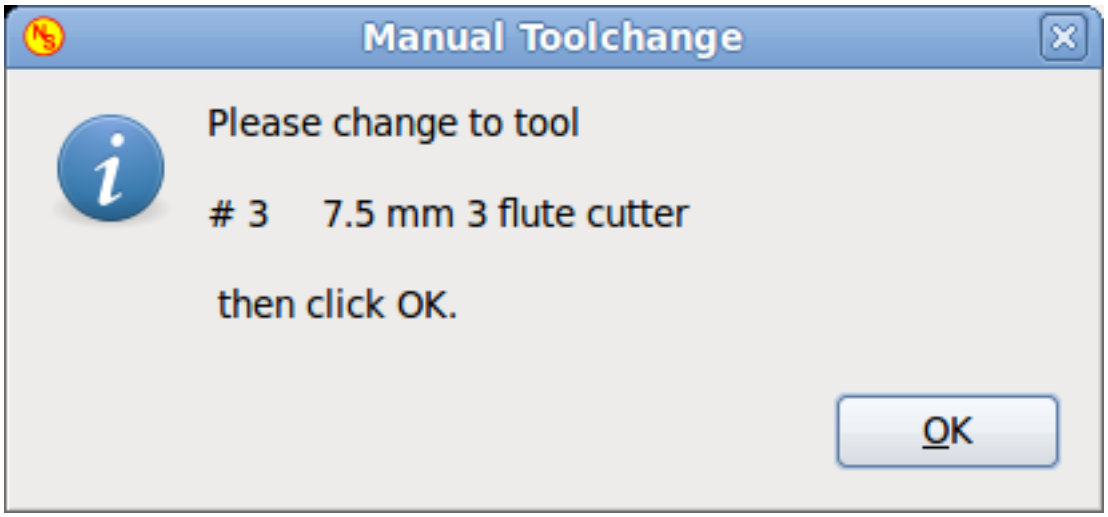
There are three pins giving information about the program progress:

- **gmoccapy.program.length** *(s32 OUT)* - Shows the total number of lines of the program
- **gmoccapy.program.current-line** *(s32 OUT)* - Indicates the current working line of the program
- **gmoccapy.program.progress** *(float OUT)* - Gives the program progress in percentage

The values may not be very accurate if you are working with subroutines or large remap procedures. Also loops will cause different values.

10.2.5.11 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.



- **gmoccapy.toolchange-number** *(s32 IN)* - The number of the tool to be changed
- **gmoccapy.toolchange-change** *(bit IN)* - Indicates that a tool has to be changed
- **gmoccapy.toolchange-changed** *(bit OUT)* - Indicates tool has been changed
- **gmoccapy.toolchange-confirm** *(bit 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
```

Anmerkung
Please take care, that this connections have to be done in the postgui HAL file.

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.

- **gmoccapy.tooloffset-x** *(float IN)*
- **gmoccapy.tooloffset-z** *(float IN)*

Anmerkung
Die Zeile ”tooloffset-x” wird bei einer Fräsmaschine nicht benötigt und wird bei einer Fräsmaschine mit trivialer Kinematik nicht angezeigt.

To display the current offsets, the pins have to be connected like this in the postgui HAL file:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

Wichtig
Please note, that GMOCCAPY takes care of its own to update the offsets, sending an G43 after any tool change, **but not in auto mode!**
So writing a program makes you responsible to include an G43 after each tool change!

10.2.6 Automatische Werkzeugmessung

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.

Wichtig
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.

Tool Measurement in GMOCCAPY is done a little bit different to many other GUIs. You should follow these steps:

1. Touch off your workpiece in X and 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:

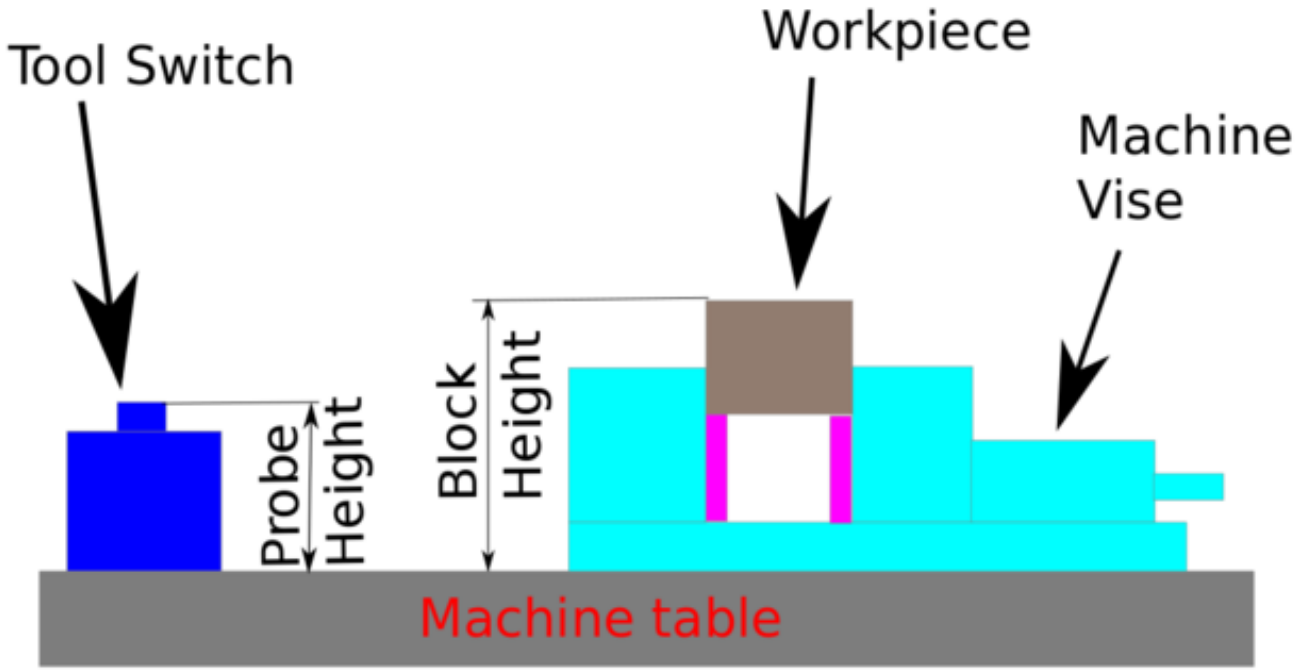


Abbildung 10.20: Werkzeugmessung Daten

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.

Anmerkung
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.

10.2.6.1 Werkzeugmess-Pins

GMOCCAPY offers five pins for tool measurement purposes. These pins are mostly used to be read from a G-code subroutine, so the code can react to different values.

- **gmoccapy.toolmeasurement** *(bit OUT)* - Enable or not tool measurement
- **gmoccapy.blockheight** *(float OUT)* - The measured value of the top face of the workpiece
- **gmoccapy.probeheight** *(floiat OUT)* - The probe switch height
- **gmoccapy.searchvel** *(float OUT)* - The velocity to search for the tool probe switch
- **gmoccapy.probevel** *(float OUT)* - The velocity to probe tool length

10.2.6.2 Änderungen an der INI-Datei für Werkzeugmessungen

Modify your INI file to include the following sections.

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
```

Anmerkung
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) The position of the tool sensor and the start position of the probing movement, all values are absolute coordinates, except MAXPROBE, which must be given in relative movement.

```
[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
```

10.2.6.3 Benötigte Dateien

First make a directory "python" in your config folder. From `<your_linuxcnc-dev_directory>/configs/sim` copy the following files into the just created `config_dir/python` folder:

- `toplevel.py`
- `remap.py`
- `stdglue.py`

From `<your_linuxcnc-dev_directory>/configs/sim/gmoccapy/macros` copy

- `on_abort.ngc`
- `change.ngc`

to the directory specified as `SUBROUTINE_PATH`, see RS274NGC Section.

Open `change.ngc` with a editor and uncomment the following lines (49 and 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

Möglicherweise möchten Sie diese Datei an Ihre Bedürfnisse anpassen.

10.2.6.4 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
```

In your postgui.hal file add the following lines:

```
# 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

To enter the page you will have to click on  and give an unlock code, which is **123** by default. If you want to change it at this time you will have to edit the hidden preference file, see the display section for details.

Die Seite ist in drei Hauptregisterkarten unterteilt:

10.2.7.1 Erscheinungsbild

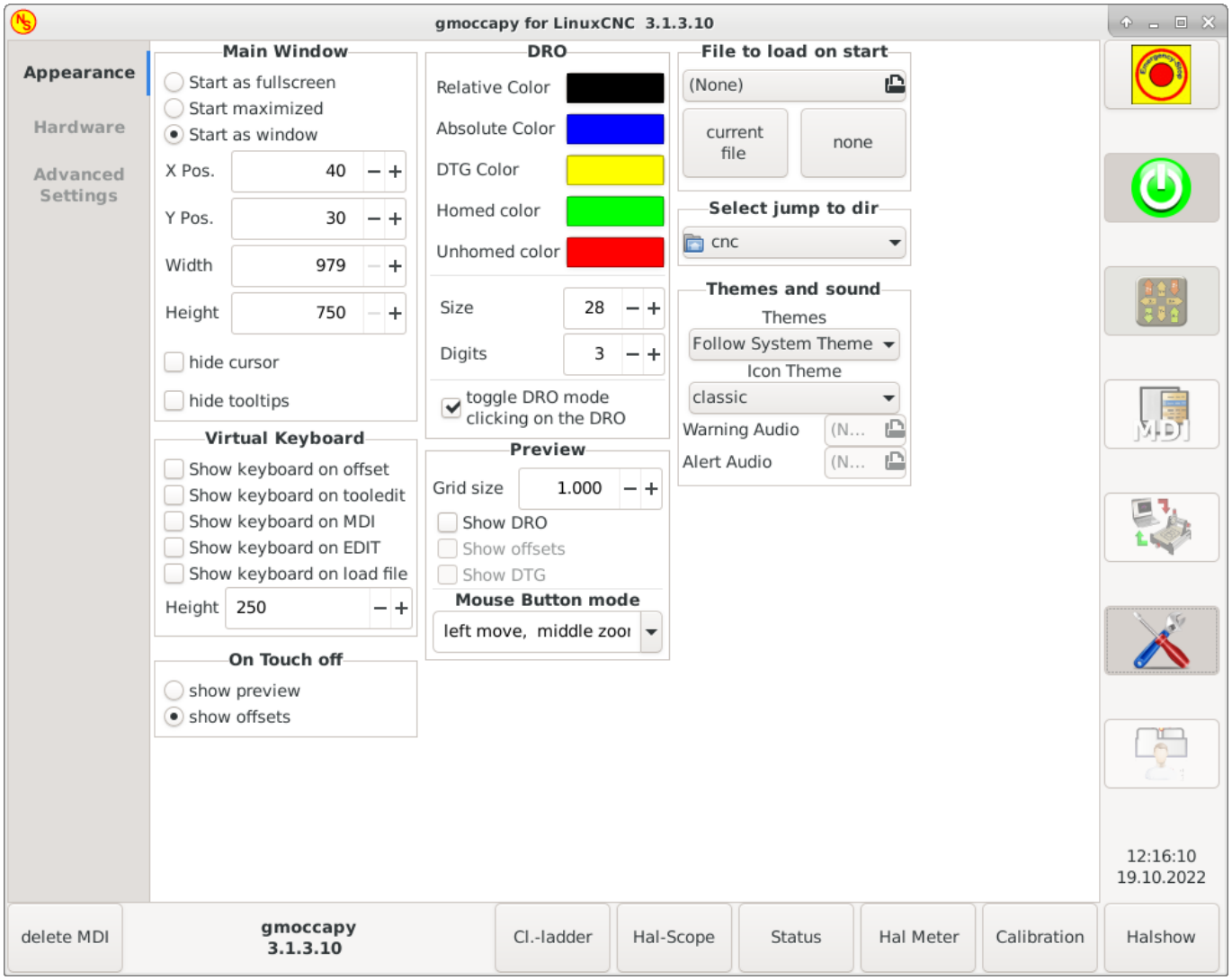


Abbildung 10.21: GMOCCAPY settings page Appearance

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 as full screen*
- *Start maximized*
- *Start as window* - If you select start as window the spinboxes to set the position and size will get active. One time set, the GUI will start every time on the place and with the size selected. Nevertheless the user can change the size and position using the mouse, but that will not have any influence on the settings.
- *hide the cursor* - Does allow to hide the cursor, what is very useful if you use a touch screen.

Tastatur The checkboxes allow the user to select if he wants the on board keyboard to be shown immediately, when entering the MDI Mode, when entering the offset page, the tooledit widget or when open a program in the EDIT mode. The keyboard button on the bottom button list will not be affected by theese settings, so you are able to show or hide the keyboard by pressing the button. The default behavior will be set by the checkboxes.

Standard sind:

Anmerkung
If this section is not sensitive, you have not installed a virtual keyboard, supported ones are *onboard* and *matchbox-keyboard*.

- *Show keyboard on offset*
- *Show keyboard on tooledit*
- *Show keyboard on MDI*
- *Show keyboard on EDIT*
- *Show keyboard on load file*

If the keyboard layout is not correct, i.e. clicking Y gives Z, than the layout has not been set properly, related to your locale settings. For onboard it can be solved with a small batch file with the following content:

```
#!/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.

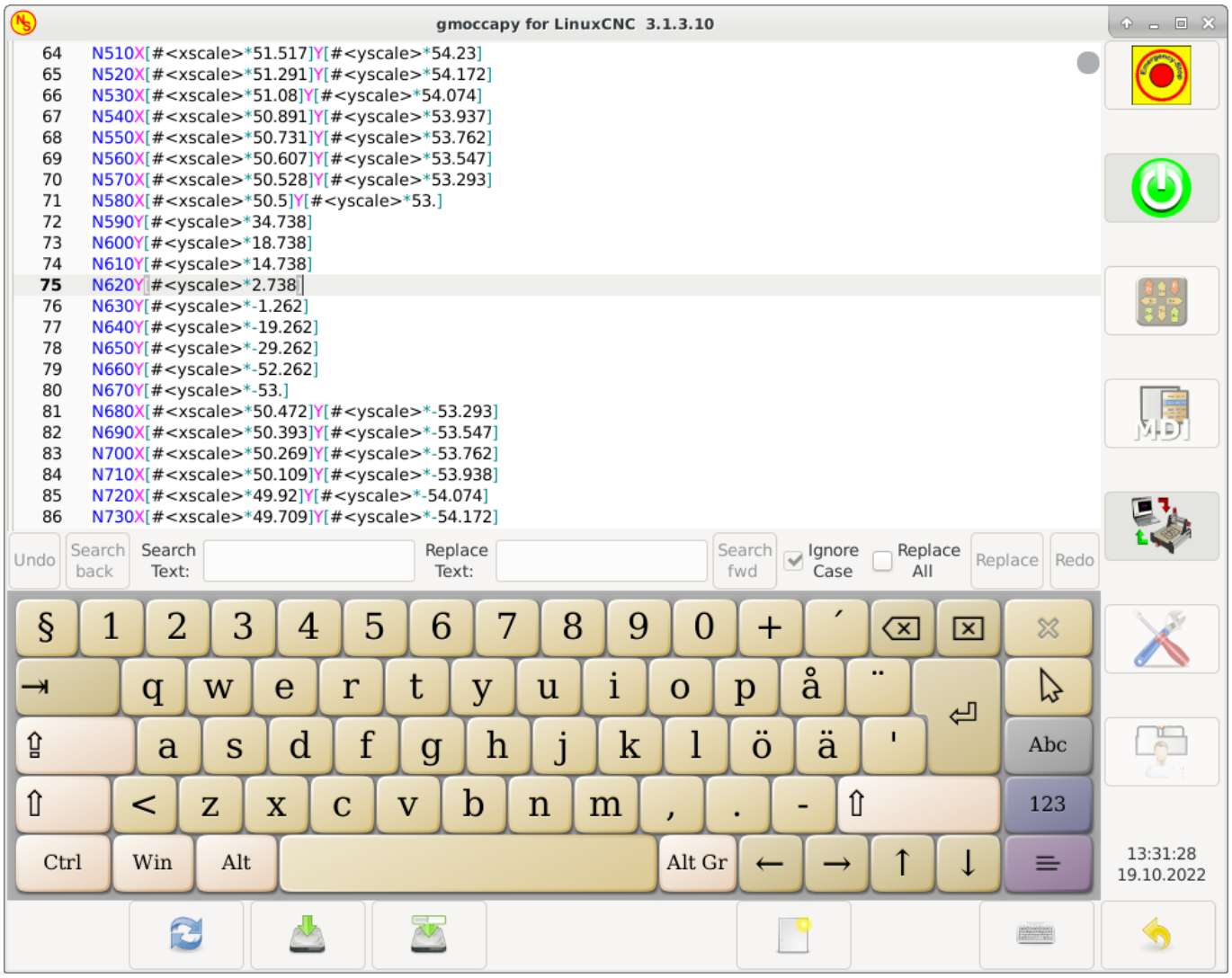


Abbildung 10.22: Gmoccapy with Onboard keyboard in edit mode

On Touch Off This gives the option whether to show the preview tab or the offset page tab when you enter the touch off mode by clicking the corresponding bottom button.

- *show preview*
- *show offsets*

DRO-Optionen 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.

By default, the background colors are:

- Relativer Modus = schwarz
- Absoluter Modus = blau
- Verbleibende Entfernung = gelb

Die Vordergrundfarbe der DRO kann ausgewählt werden mit:

- homed color = green
- unhomed color = red

Anmerkung
You can change through the DRO modes (absolute, relative, distance to go) by clicking the number on the DRO! If you click on the left side letter of the DRO a popup window will allow you to set the value of the axes, making it easier to set the value, as you will not need to go over the touch off bottom button.

Größe
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 (engl. für Ziffern)
Sets the number of digits of the DRO from 1 to 5.

Anmerkung
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.

DRO-Modus umschalten
Wenn diese Option nicht aktiviert ist, führt ein Mausklick auf die Anzeige zu keiner Aktion. +
Standardmäßig ist dieses Kontrollkästchen aktiviert, so dass jeder Klick auf eine Anzeige die Anzeige von aktuell auf relativ zu DTG (distance to go) umschaltet. +
Nichtsdestotrotz öffnet ein Klick auf den Achsenbuchstaben den Popup-Dialog zum Einstellen des Achsenwertes.

Vorschau

- *Grid Size* - Sets the grid size of the preview window. Unfortunately the size **has to be set in inches**, even if your machine units are metric. We do hope to fix that in a future release.

Anmerkung
Das Gitter wird in der perspektivischen Ansicht nicht angezeigt.

- *Show DRO* - Will show the a DRO also in the preview pane, it will be always shown in fullsize preview.
- *Show DTG* - Will show the DTG (direct distance to end point) in the preview pane if Show DRO is active. Otherwise only in full size preview.
- *Show Offsets* - Will show the offsets in the preview pane when Show DRO is active. Otherwise only in full size preview.
- *Mouse Button Mode* - This combobox allows you to select the button behavior of the mouse to rotate, move or zoom within the preview:
  - 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
  Die Standardeinstellung ist links verschieben, Mitte zoomen, rechts drehen.

Mit dem Mausrad können Sie die Vorschau in jedem Modus vergrößern.

Tipp
Wenn Sie ein Element in der Vorschau auswählen, wird das ausgewählte Element als Rotationsmittelpunkt genommen und im Automodus wird die entsprechende Codezeile hervorgehoben.

Beim Starten zu ladende Datei Select the file you want to be loaded on start up. If a file is loaded, it can be set by pressing the current button. To avoid that any program is loaded at start up, just press the None button.

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`.

Zum Verzeichnis springen Sie können hier das Verzeichnis einstellen, in das gesprungen werden soll, wenn Sie im Dateiauswahldialog auf die entsprechende Schaltfläche klicken.

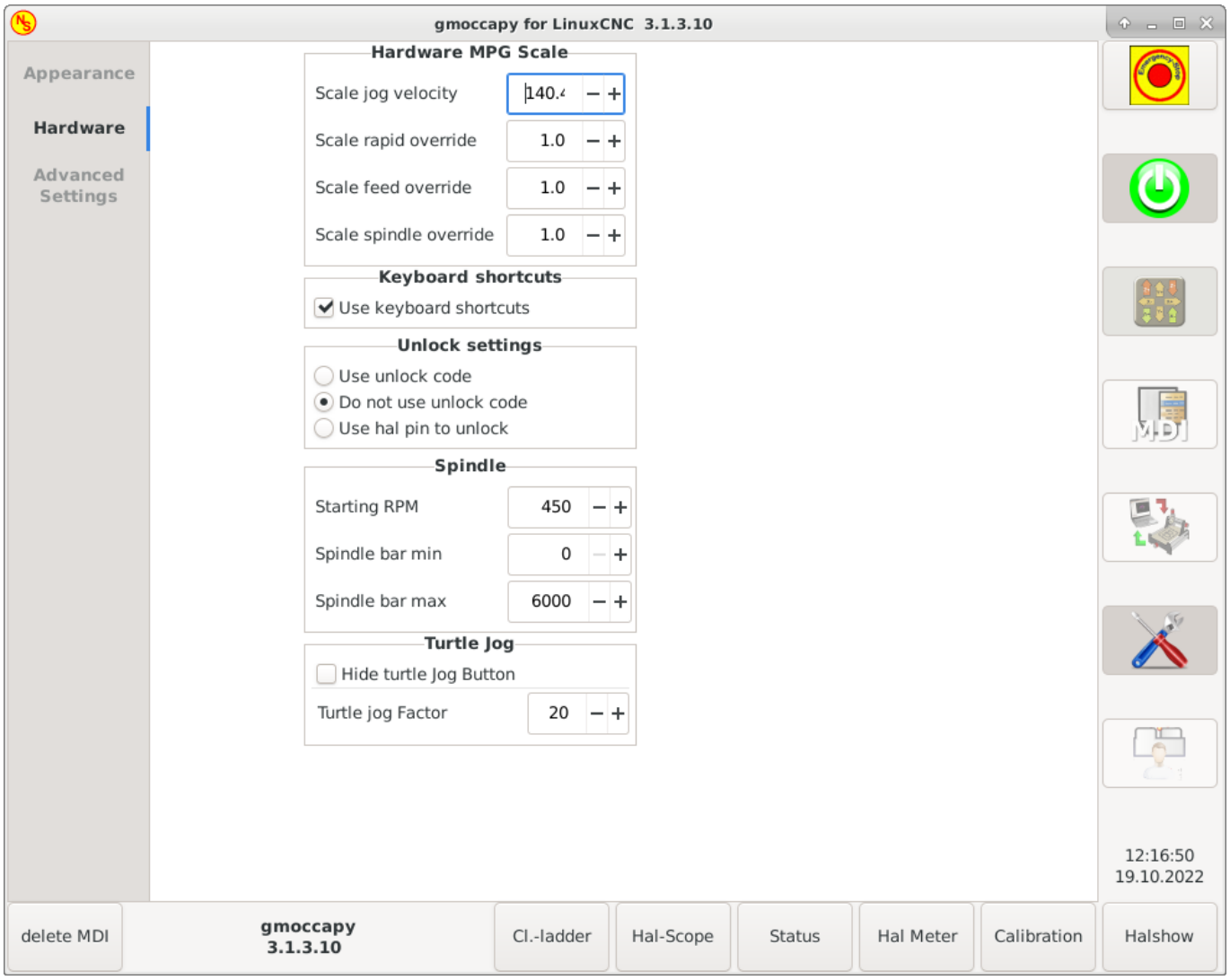
Themen und Klänge Hier kann der Benutzer auswählen, welches Desktop-Thema angewendet werden soll und welche Fehler- und Meldungstöne abgespielt werden sollen. Standardmäßig ist "Systemthema folgen" eingestellt.

It further allows to change the icon theme. Currently there are three themes available:

- classic
- material
- material light

To create custom icon themes, see section Icon Theme for details.

10.2.7.2 Hardware



Hardware MPG Scale For the different HAL pins to connect MPG wheels to, you may select individual scales to be applied. The main reason for this was my own test to solve this through HAL connections, resulting in a very complex HAL file. Imagine a user having an MPG Wheel with 100 ipr and he wants to slow down the max. vel. from 14000 to 2000 mm/min, that needs 12000 impulses, resulting in 120 turns of the wheel! Or an other user having a MPG Wheel with 500 ipr and he wants to set the spindle override witch has limits from 50 to 120 % so he goes from min to max within 70 impulses, meaning not even 1/4 turn.

Standardmäßig werden alle Skalen anhand der Berechnung festgelegt:

```
(MAX - MIN)/100
```

Tastatürkürzel Some users want to jog there machine using the keyboard buttons and there are others that will never allow this. So everybody can select whether to use them or not.
Keyboard shortcuts are disabled by default.

Warnung
It is not recommended to use keyboard jogging, as it represents a serious risk for operator and machine.

Please take care if you use a lathe, then the shortcuts will be different, see the Lathe Specific Section.

Allgemeines

- *F1* - Trigger Estop (will work even if keyboard shortcuts are disabled)
- *F2* - Toggle machine on/off
- *F3* - Manual mode
- *F5* - MDI mode
- *ESC* - Abort

In Manual Mode

- *Arrow_Left* or *NumPad_Left* - Jog X minus
- *Arrow_Right* or *NumPad_Right* - Jog X plus
- *Arrow_up* or *NumPad_Up* - Jog Y plus
- *Arrow_Down* or *NumPad_Down* - Jog Y minus
- *Page_Up* or *NumPad_Page_Up* - Jog Z plus
- *Page_Down* or *NumPad_Page_Down* - Jog Z minus

In Auto Mode

- *R* or *r* - Run program
- *P* or *p* - Pause program
- *S* or *s* - Resume program
- *Control* + *R* or *Control* + *r* - Reload the loaded file

Message handling (see Message behavior and appearance)

- *WINDOWS* - Delete last message
- *Control* + *Space* - Delete all messages

Optionen freischalten Es gibt drei Optionen zum Entsperren der Einstellungsseite:

- *Use unlock code* - The user must give a code to get in
- *Do not use unlock code* - There will be no security check
- *Use HAL pin to unlock* - Hardware pin must be high to unlock the settings, see hardware unlock pin

Default is *use unlock code* (default = **123**)

Spindel

- *Starting RPM* - Sets the rpm to be used if the spindle is started and no S value has been set.

  Anmerkung
  This value will be presetted according to your settings in `[DISPLAY] DEFAULT_SPINDLE_SPEED` of your INI file. If you change the settings on the settings page, that value will be default from that moment, your INI file will not be modified.

- *Spindle bar min* and *Spindle bar max* - Sets the limits of the spindle bar shown in the INFO frame on the main screen.

  Default values are:
  MIN = 0
  MAX = 6000

Anmerkung
It is no error giving wrong values. If you give a maximum of 2000 and your spindle makes 4000 rpm, only the bar level will be wrong on higher speeds than 2000 rpm.

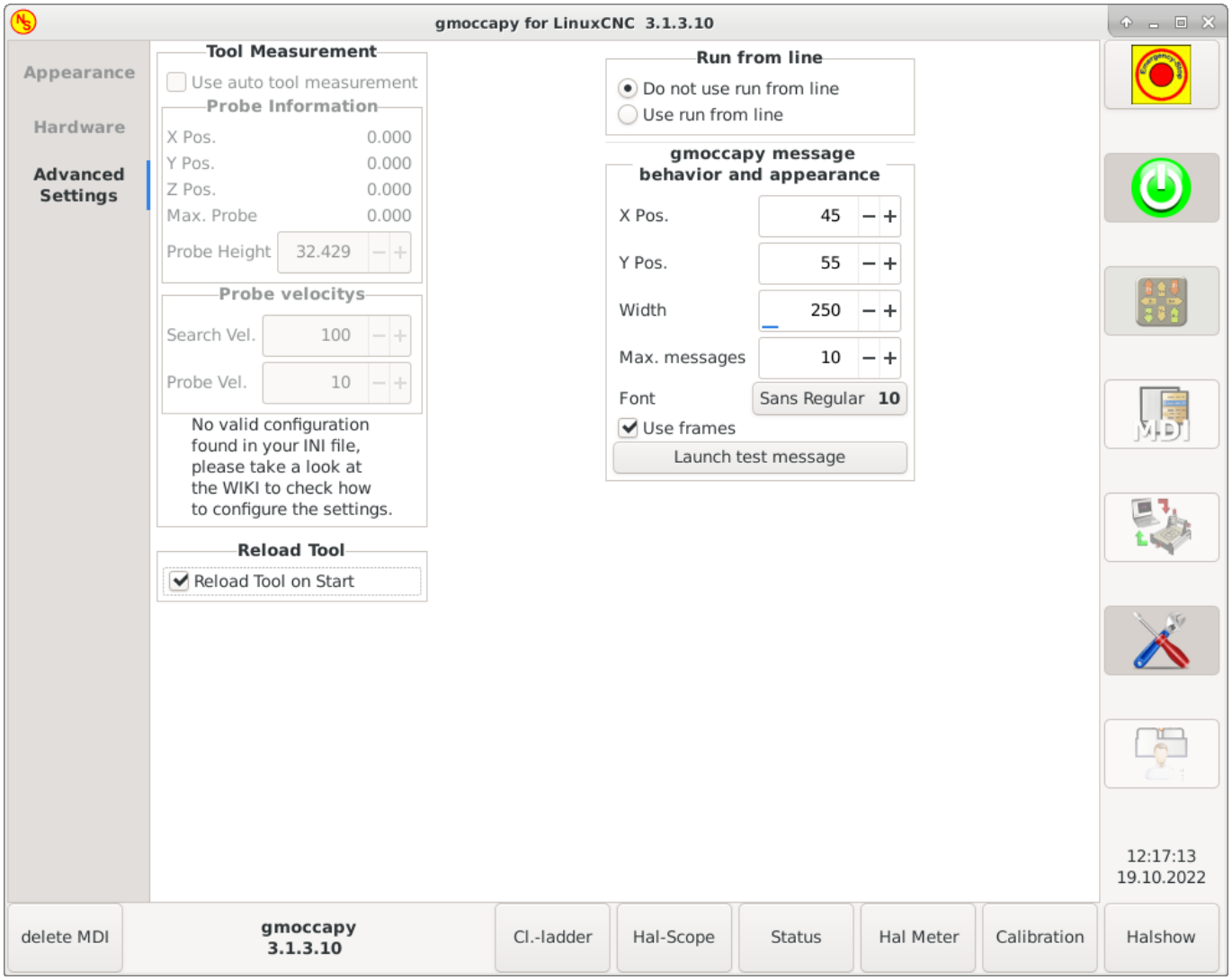
Schildkröten Jog

This settings will have influence on the jog velocities.

- *Hide turtle jog button* - Will hide the button right of the jog velocity slider. If you hide this button, please take care that the ”rabbit mode” is activated, otherwise you will not be able to jog faster than the turtle jog velocity, which is calculated using the turtle jog factor.
- *Turtle jog factor* - Sets the scale to apply for turtle jog mode (button pressed, showing the turtle). If you set a factor of 20, the turtle max. jog velocity will be 1/20 of the max. velocity of the machine.

Anmerkung
This button can be controlled using the Turtle-Jog HAL Pin.

10.2.7.3 Erweiterte Einstellungen



Werkzeugmessung Bitte prüfen Sie Auto Tool Measurement

Anmerkung
Wenn dieser Teil nicht empfindlich ist, haben Sie keine gültige INI-Datei-Konfiguration für die Verwendung der Werkzeugmessung.

- *Use auto tool measurement* - If checked, after each tool change, a tool measurement will be done, the result will be stored in the tool table and a G43 will be executed after the change.

Sonden-Informationen

Die folgenden Informationen werden aus Ihrer INI-Datei entnommen und müssen in absoluten Koordinaten angegeben werden.

- *X Pos.* - The X position of the tool switch.
- *Y Pos.* - The Y position of the tool switch.
- *Z Pos.* - The Z position of the tool switch, we will go as rapid move to this coordinate.
- *Max. Probe* The distance to search for contact, an error will be launched, if no contact is given in this range. The distance has to be given in relative coordinates, beginning the move from Z Pos., so you have to give a negative value to go down!
- *Probe Height* - The height of your probe switch, you can measure it. Just touch off the base where the probe switch is located and set that to zero. Then make a tool change and watch the tool_offset_z value, that is the height you must enter here.

Geschwindigkeiten der Sonde

- *Search Vel.* - The velocity to search for the tool switch. After contact the tool will go up again and then goes towards the probe again with probe vel, so you will get better results.
- *Probe Vel.* - The velocity for the second movement to the switch. It should be slower to get better touch results. (In simulation mode, this is commented out in macros/change.ngc, otherwise the user would have to click twice on the probe button.)

Reload Tool

- *Reload Tool on Start* - Loads the last tool on start after homing.

Option "Von Zeile ausführen" 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.

Warnung
It is not recommend to use run from line, as LinuxCNC will not take care of any previous lines in the code before the starting line. So errors or crashes are fairly likely.

Nachrichtenverhalten und Erscheinungsbild

This will display small pop up windows displaying a message or error text, similar to the ones known from AXIS. You can delete a specific message by clicking on its close button. If you want to delete the last one, just hit the `WINDOWS` key on your keyboard, or delete all messages at once with `Control + Space`.

Sie haben die Möglichkeit, einige Optionen einzustellen:

- *X Pos* - The position of the top left corner of the message in X counted in pixel from the top left corner of the screen.

- *Y Pos* - The position of the top left corner of the message in Y counted in pixel from the top left corner of the screen.
- *Width* - The width of the message box.
- *Max Messages* - The maximum number of messages you want to see at once. If you set this to 10, the 11th message will delete the first one, so you will only see the last 10.
- *Font* - The font and size you want to use to display the messages.
- *Use frames* - If you activate the checkbox, each message will be displayed in a frame, so it is much easier to distinguish the messages. But you will need a little bit more space.
- *Launch test message*-button - It will show a message, so you can see the changes of your settings without the need to generate an error.

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*- The classic GMOCCAPY icons.
- *material* - A modern icon theme inspired by Google's Material Icons that automatically adopts its coloring from the selected desktop theme.
- *material-light* - Derived from material but optimized for light desktop themes.

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

10.2.8.1 Benutzerdefiniertes Symboldesign (eigentlich engl. icon theme)

Creating a custom icon theme is pretty easy. All you need is a text editor and of course the desired icons as pixel or vector graphics. Details about how exactly an icon theme is built can be found at the 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.
- Directories: A comma separated list of all the directories of your icon theme.
  Each directory usually contains all the icons of the theme in a specific size, for example 16x16/actions should contain all icons with the category ”actions” in the size 16x16 pixels as pixel-graphics (e.g. png files). A special case is the directory called ”scalable/actions”, this contains scalable icons not tied to a specific size (e.g. svg files).
  By supplying different sized versions of the icons, we can guarantee a nice looking icon if different sizes and we also have the ability to change the icon according to its size, for example a 64x64 px sized icon may contain more details than its 16x16 px version.

• Für jedes Verzeichnis müssen wir auch einen Abschnitt in die Datei index.theme schreiben:

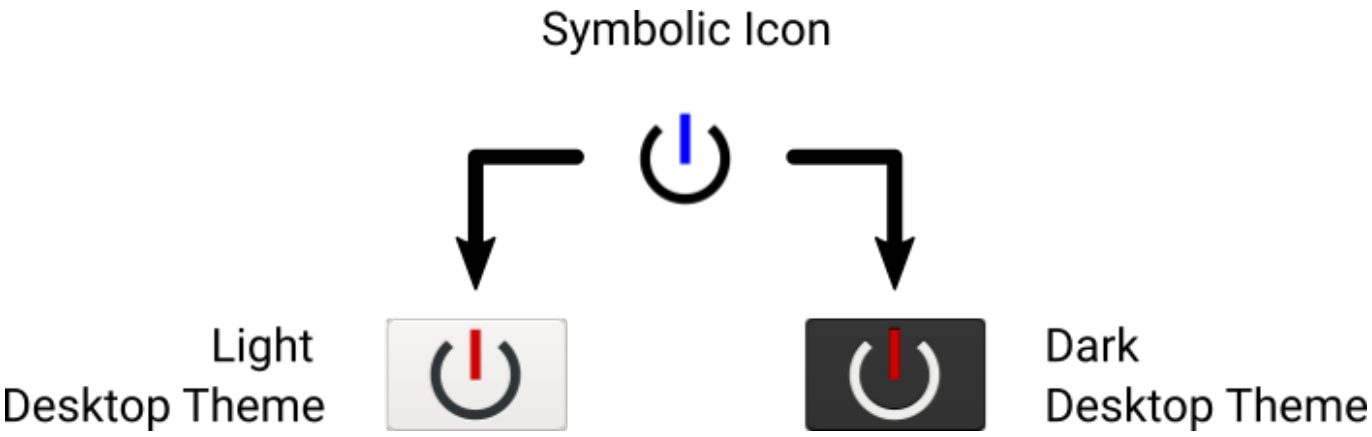
```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

[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.

10.2.8.2 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	Success: this color indicates ”success” (usually something green’ish).
green (engl. für grün)	#00ff00	Warning: this color indicates ”warning” (usually something yellow/orange’ish).
blue (engl. für blau)	#0000ff	Error: this color indicates ”error” (usually something red’ish).

Tipp
Examples of symbolic icons can be found at `linuxcnc/share/gmoccapy/icons/material`.

10.2.9 Drehmaschinen-spezifischer Abschnitt

If in the INI file `LATHE = 1` is given, the GUI will change its appearance to the special needs for a lathe. Mainly the Y axis will be hidden and the jog buttons will be arranged in a different order.

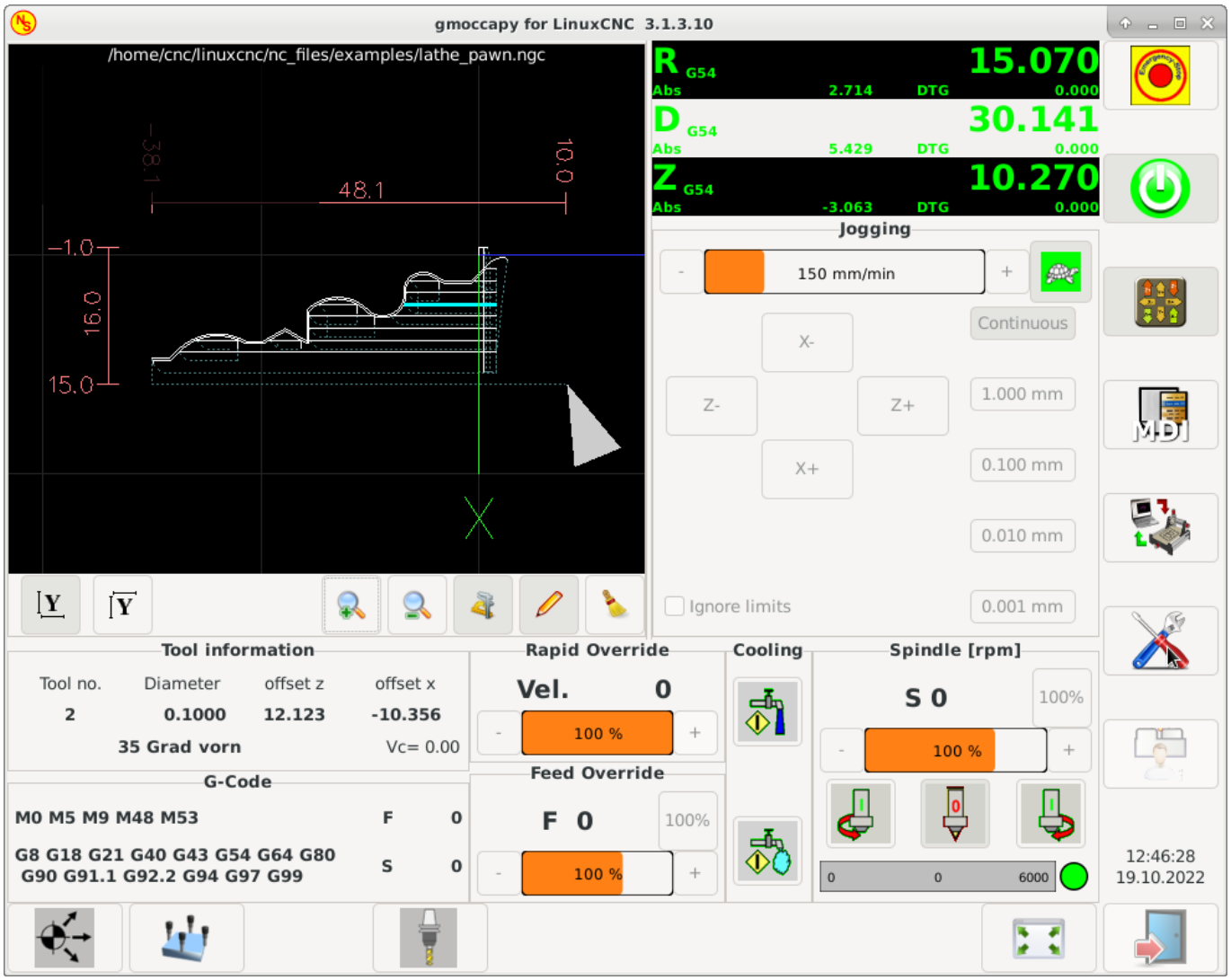


Abbildung 10.23: Normale Drehmaschine

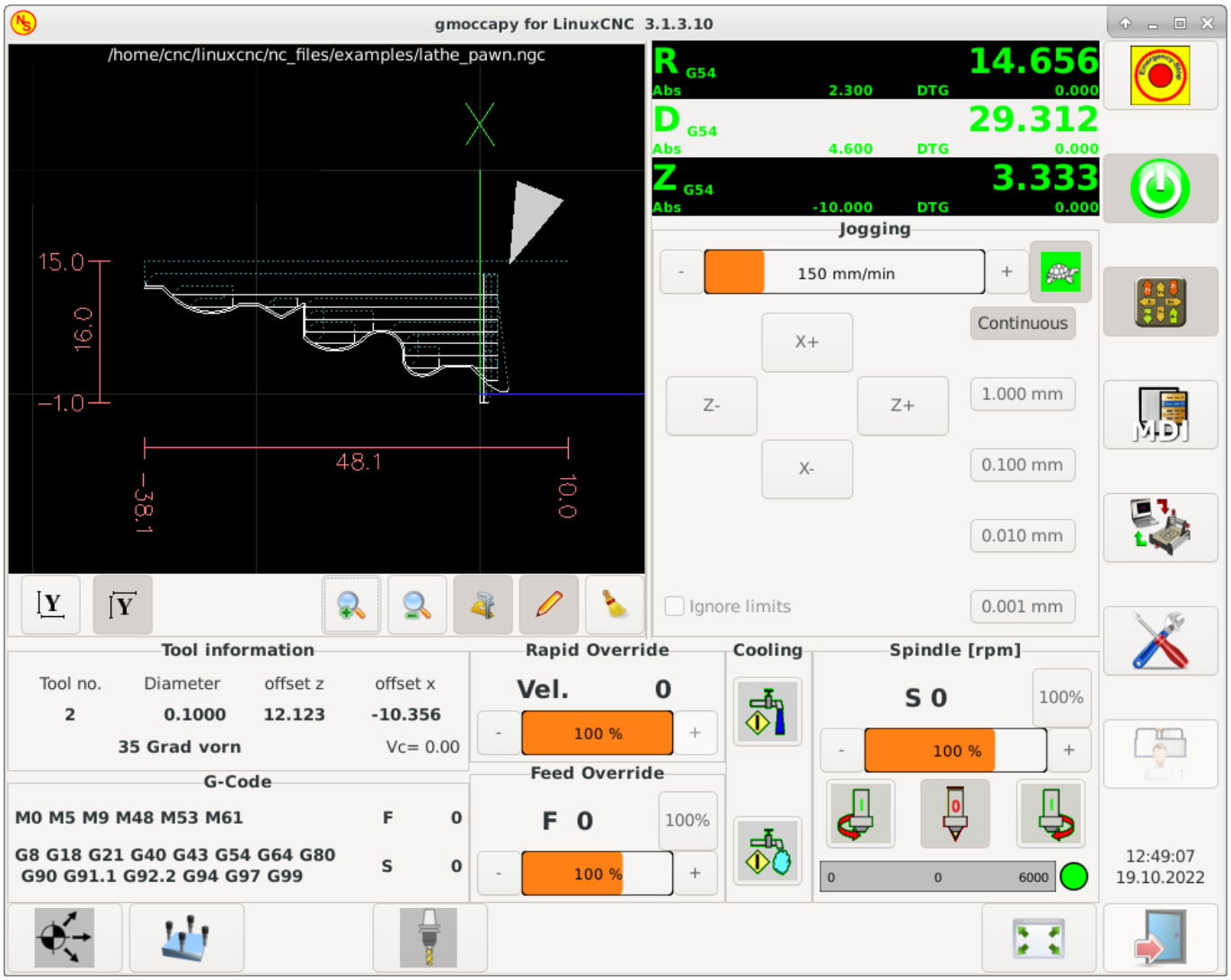


Abbildung 10.24: Back Tool Lathe

As you see the R DRO has a black background and the D DRO is gray. This will change according to the active G-code G7 or G8. The active mode is visible by the black background, meaning in the shown images G8 is active.

The next difference to the standard screen is the location of the jog buttons. X and Z have changed places and Y is gone. You will note that the X+ and X- buttons changes there places according to normal or back tool lathe.

Auch das Verhalten der Tastatur wird sich ändern:

Normale Drehmaschine:

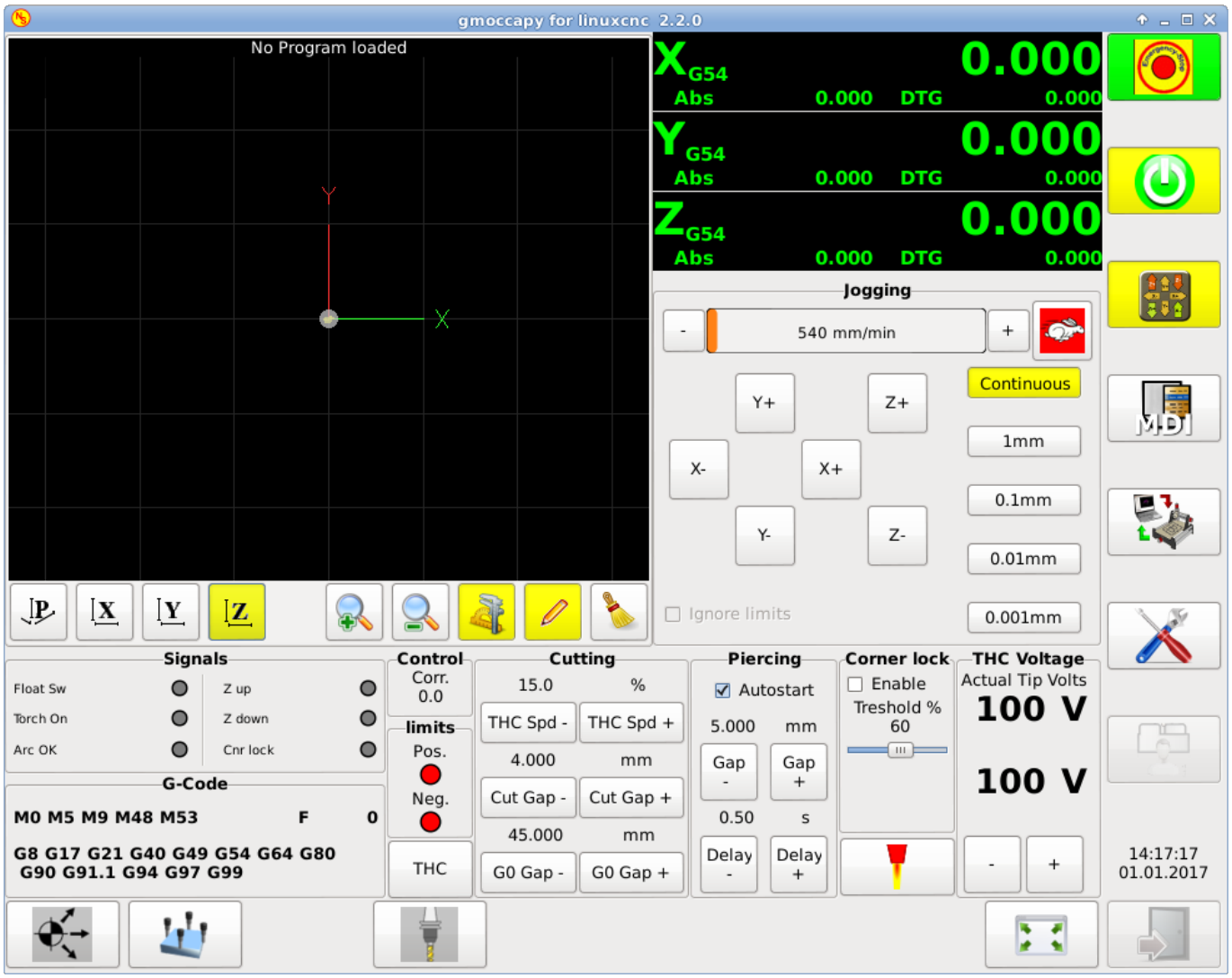
- *Arrow_Left* or *NumPad_Left* - Jog Z minus
- *Arrow_Right* or *NumPad_Right* - Jog Z plus
- *Arrow_up* or *NumPad_Up* - Jog X minus
- *Arrow_Down* or *NumPad_Down* - Jog X plus

Back Tool Lathe:

- *Arrow_Left* or *NumPad_Left* - Jog Z minus
- *Arrow_Right* or *NumPad_Right* - Jog Z plus
- *Arrow_up* or *NumPad_Up* - Jog X plus
- *Arrow_Down* or *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

Below is a series of videos that show GMOCCAPY in action. Unfortunately, these videos don’t show the latest version of GMOCCAPY, but the way to use it will still be the same as in the current version. I will update the videos as soon as possible.

10.2.11.1 Grundlegende Verwendung

https://www.youtube.com/watch?v=O5B-s3uiI6g

10.2.11.2 Simulierte Jog-Wheels

http://youtu.be/ag34SGxt97o

10.2.11.3 Einstellungen Seite

https://www.youtube.com/watch?v=AuwhSHRJoiI

10.2.11.4 Simulierte Hardware-Taste

German: http://www.youtube.com/watch?v=DTqhY-MfzDE
English: http://www.youtube.com/watch?v=ItVWJBK9WFA

10.2.11.5 Benutzer-Registerkarten

http://www.youtube.com/watch?v=rG1zmeqXyZI

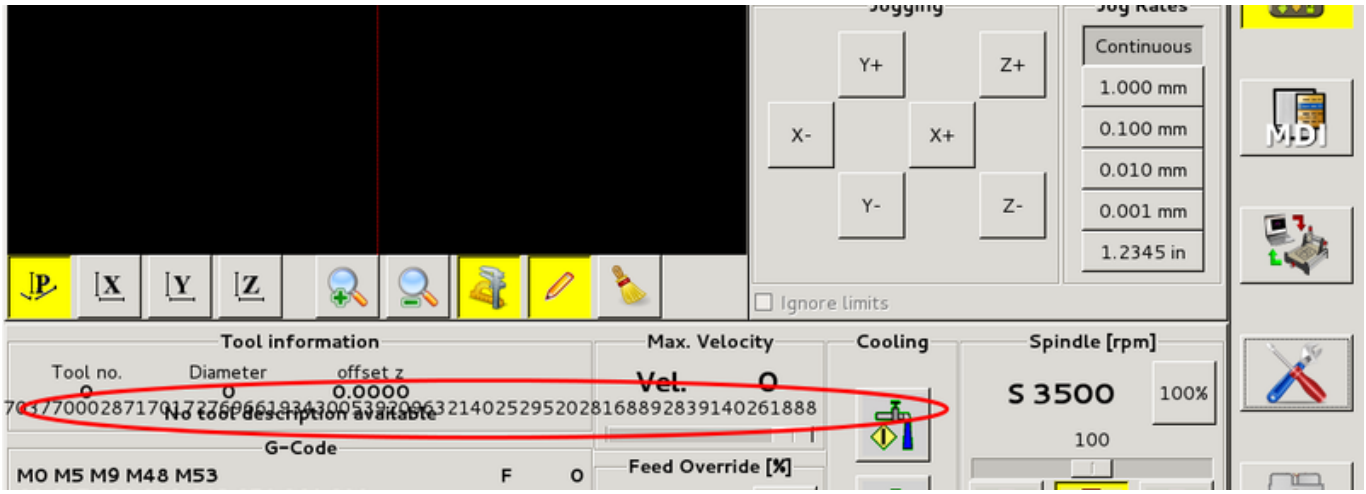
10.2.11.6 Videos zur Werkzeugvermessung

Auto Tool Measurement Simulation: http://youtu.be/rrkMw6rUFdk
Auto Tool Measurement Screen: http://youtu.be/Z2ULDj9dzvk
Auto Tool Measurement Machine: http://youtu.be/1arucCaDdX4

10.2.12 Bekannte Probleme

10.2.12.1 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.

10.2.12.2 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:

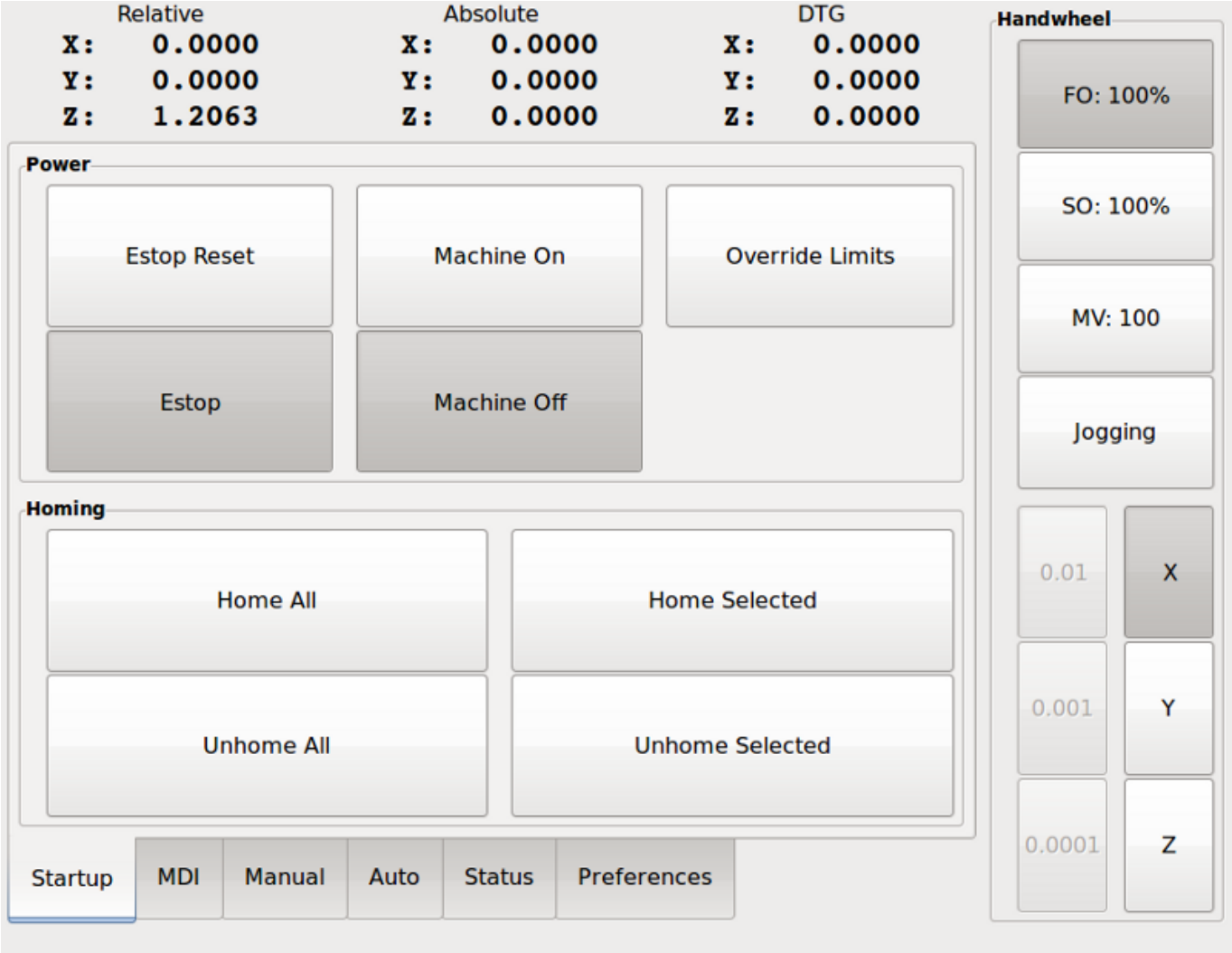
```
 o<zeroxy> sub

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

10.3.1.1 HAL-Verbindungen

Touchy sucht in der INI-Datei unter der Überschrift *[HAL]* nach Einträgen von *POSTGUI_HALFILE=<Datein*
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.

Anmerkung
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.

Anmerkung
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.

10.3.1.2 Empfohlen für jede Einrichtung

- Notaus-Button fest in der Notaus-Kette verdrahtet

10.3.2 Einrichtung

10.3.2.1 Touchy aktivieren

Um Touchy zu verwenden, ändern Sie im Abschnitt *[DISPLAY]* Ihrer INI-Datei die Zeile für die Anzeigeauswahl in *DISPLAY = touchy*.

10.3.2.2 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.

10.3.2.3 Makros

Touchy kann O-Wort-Makros über die MDI-Schnittstelle aufrufen. Um dies zu konfigurieren, fügen Sie im Abschnitt *[TOUCHY]* 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.

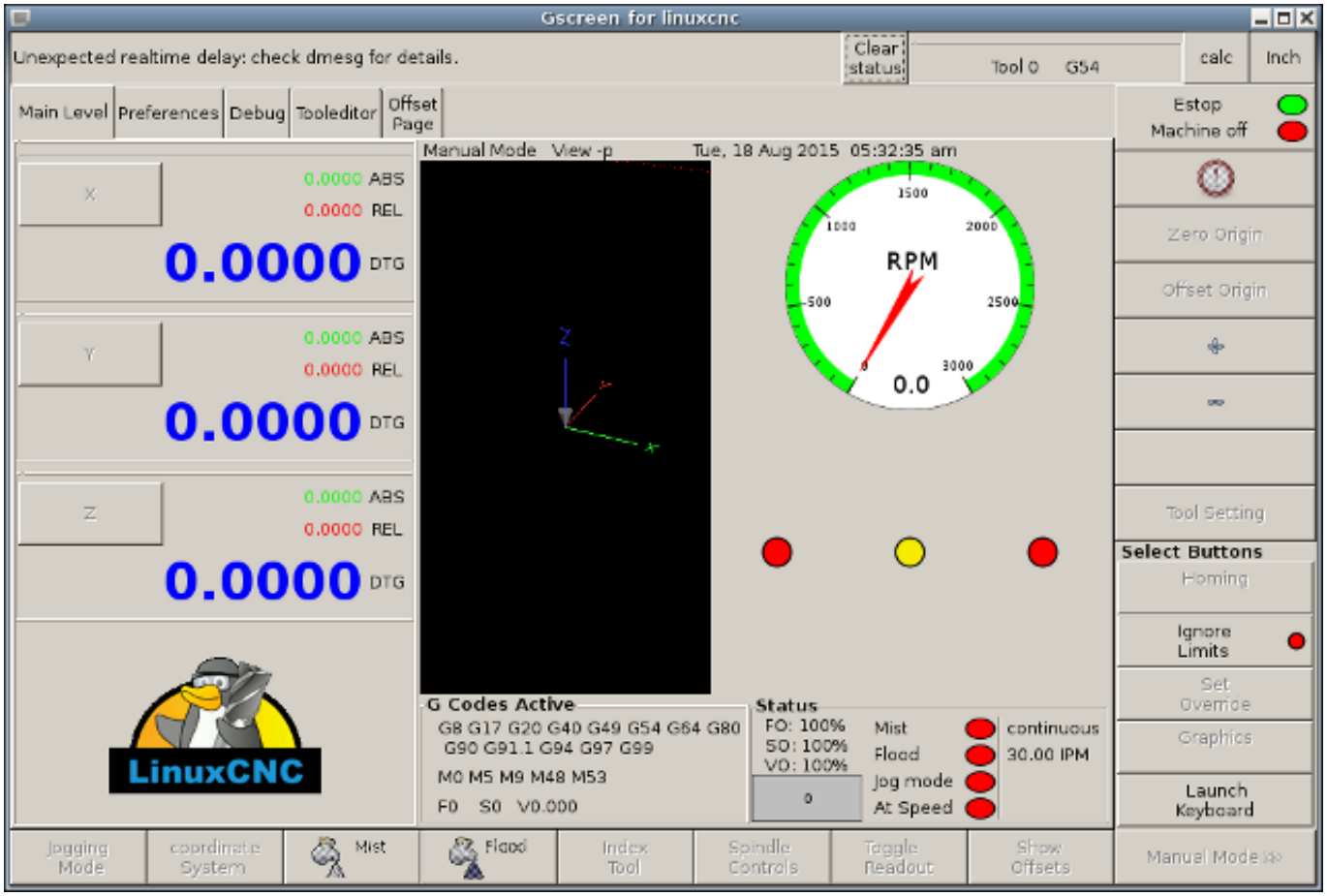


Abbildung 10.25: Gscreen Standardbildschirm

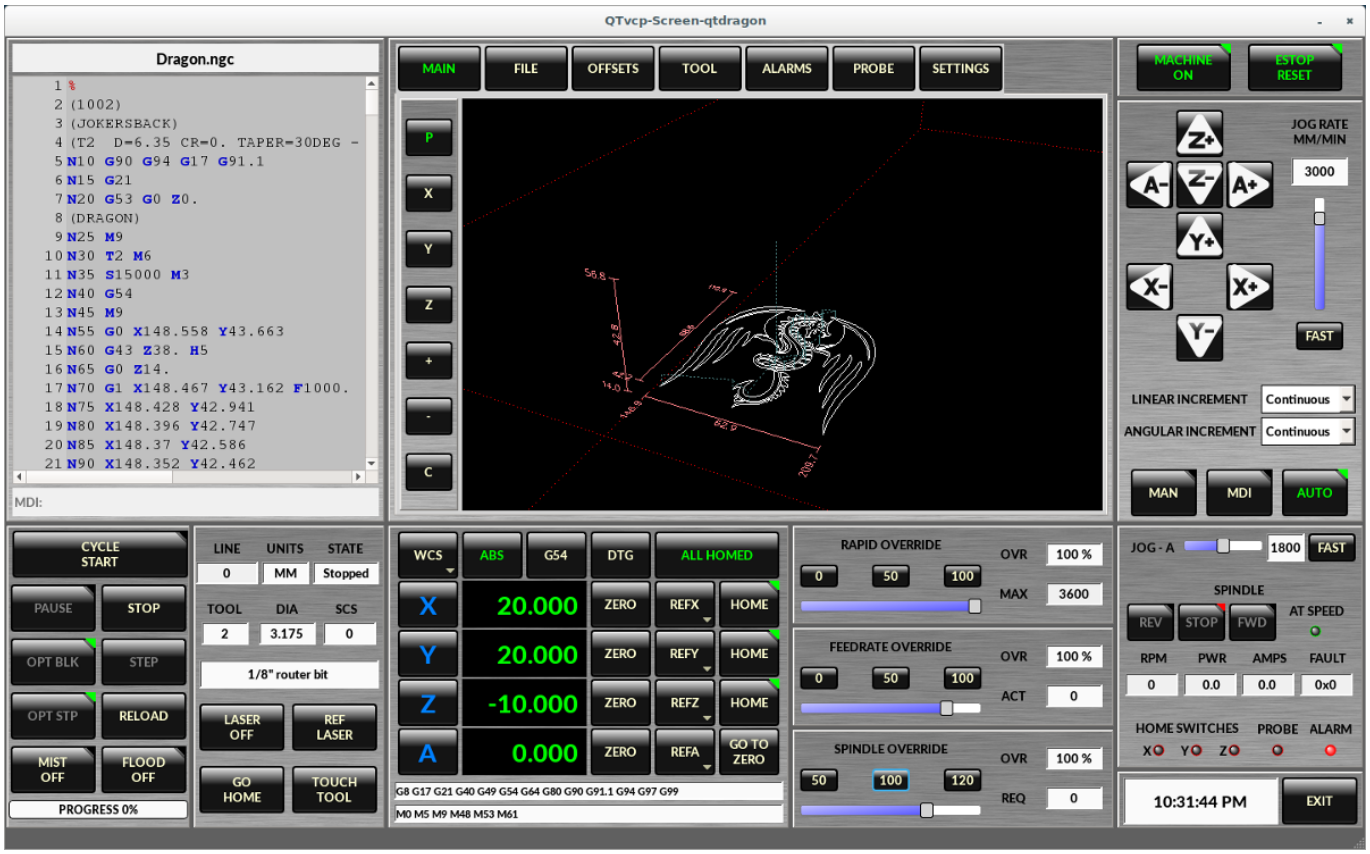


Abbildung 10.26: Gscreen Silverdragon-Bildschirm

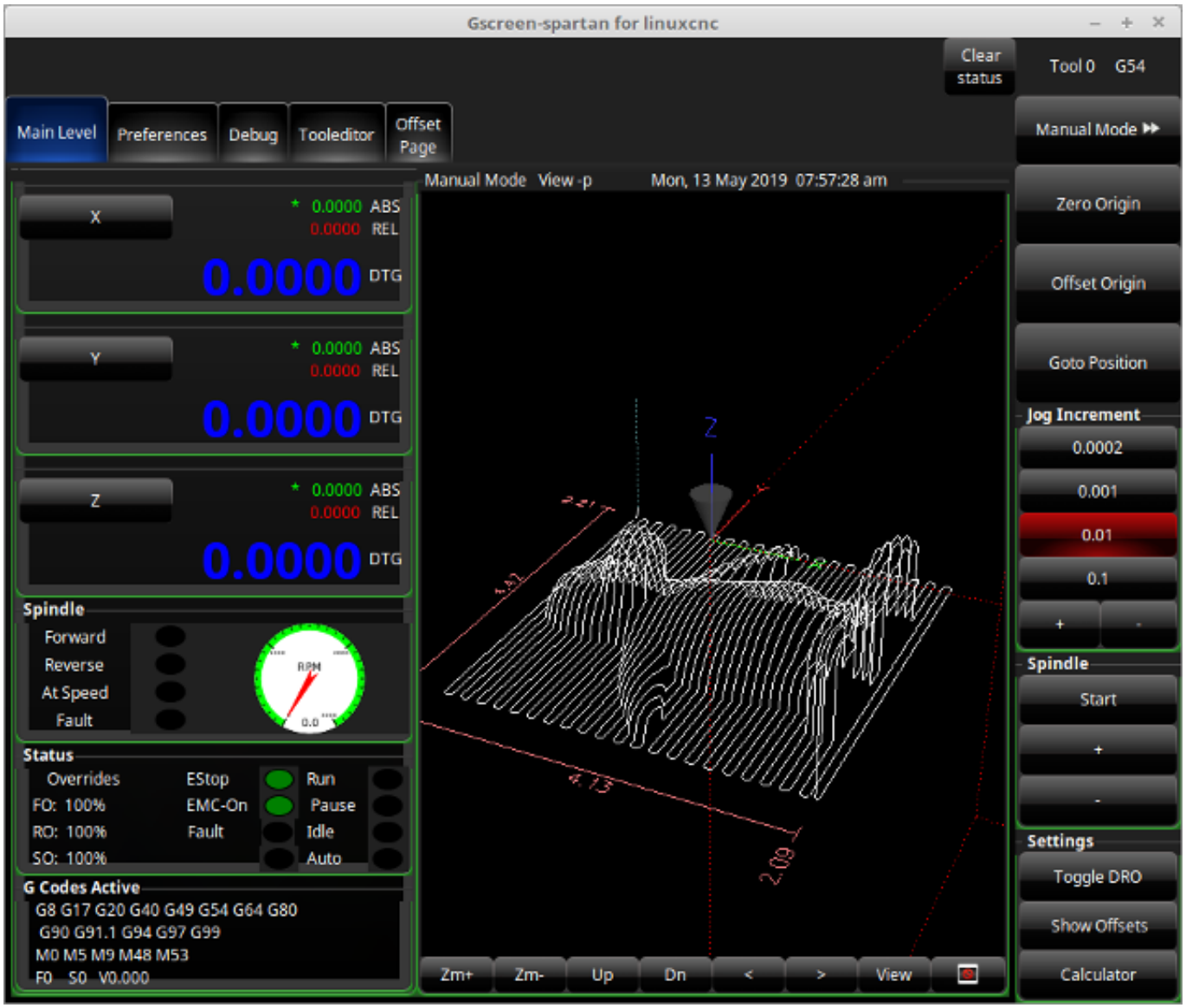


Abbildung 10.27: Gscreen Spartan-Bildschirm

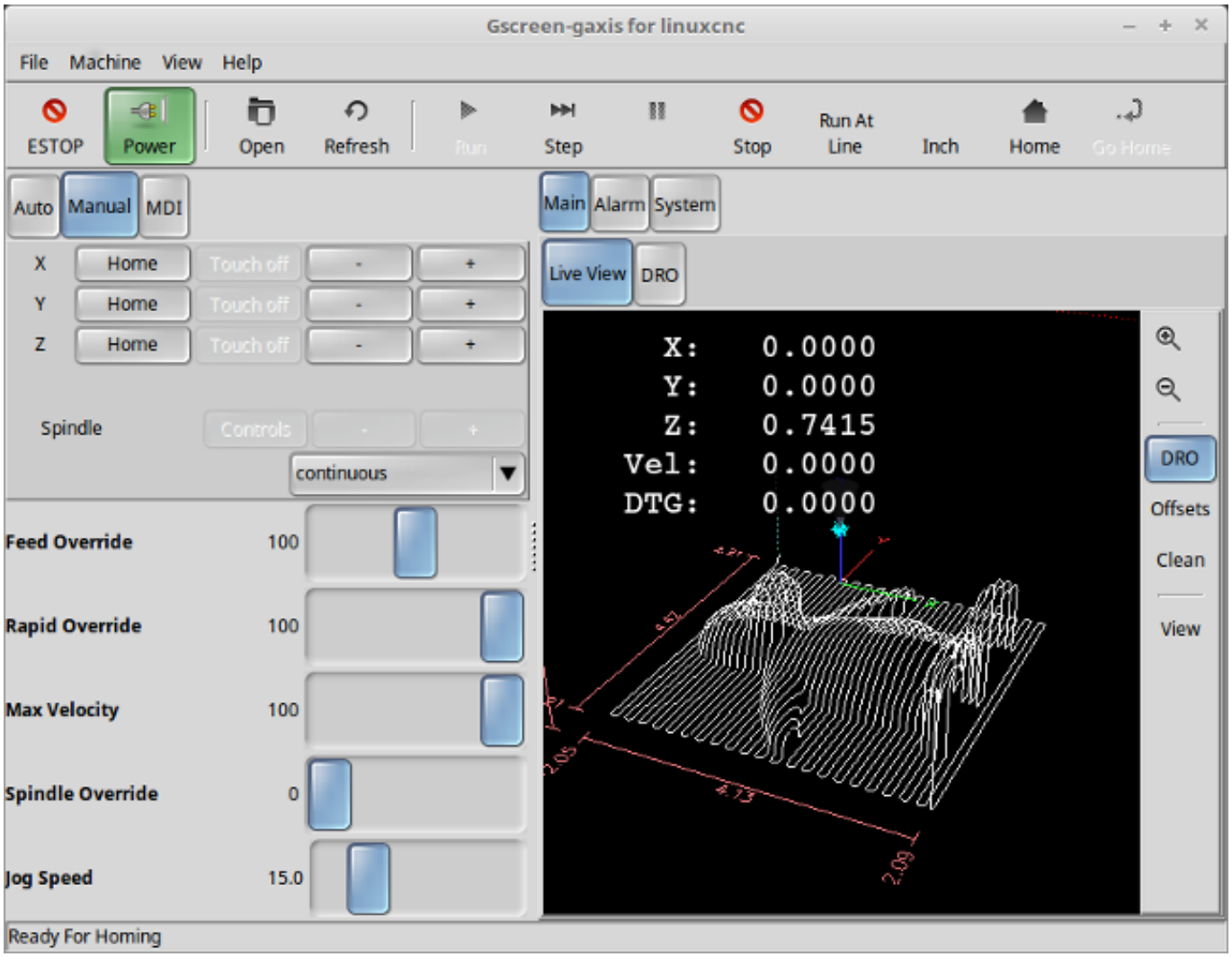


Abbildung 10.28: Gscreen Gaxis-Bildschirm

Abbildung 10.29: 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.

10.4.1.1 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.

10.4.1.2 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!

10.4.2.1 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.

10.4.2.2 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.

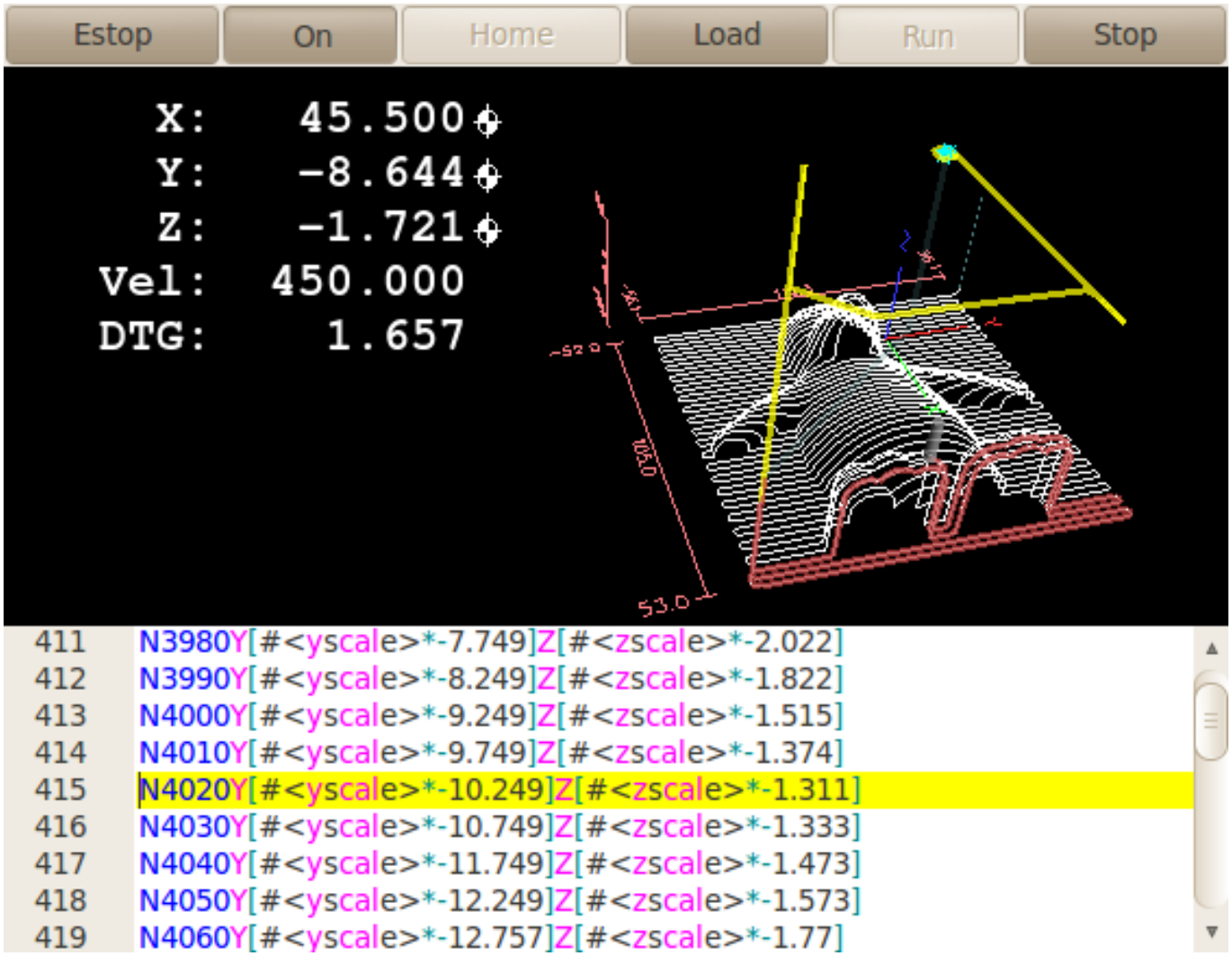
Ok, Sie haben die Glade-Datei geladen und können sie nun bearbeiten. Das erste, was Ihnen auffällt, ist, dass es im Editor nicht so aussieht, wie es angezeigt wird. Gscreen verwendet einige Tricks, wie z. B. das Ausblenden aller Schaltflächenfelder bis auf eines und das Ändern dieses Feldes je nach Modus. 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

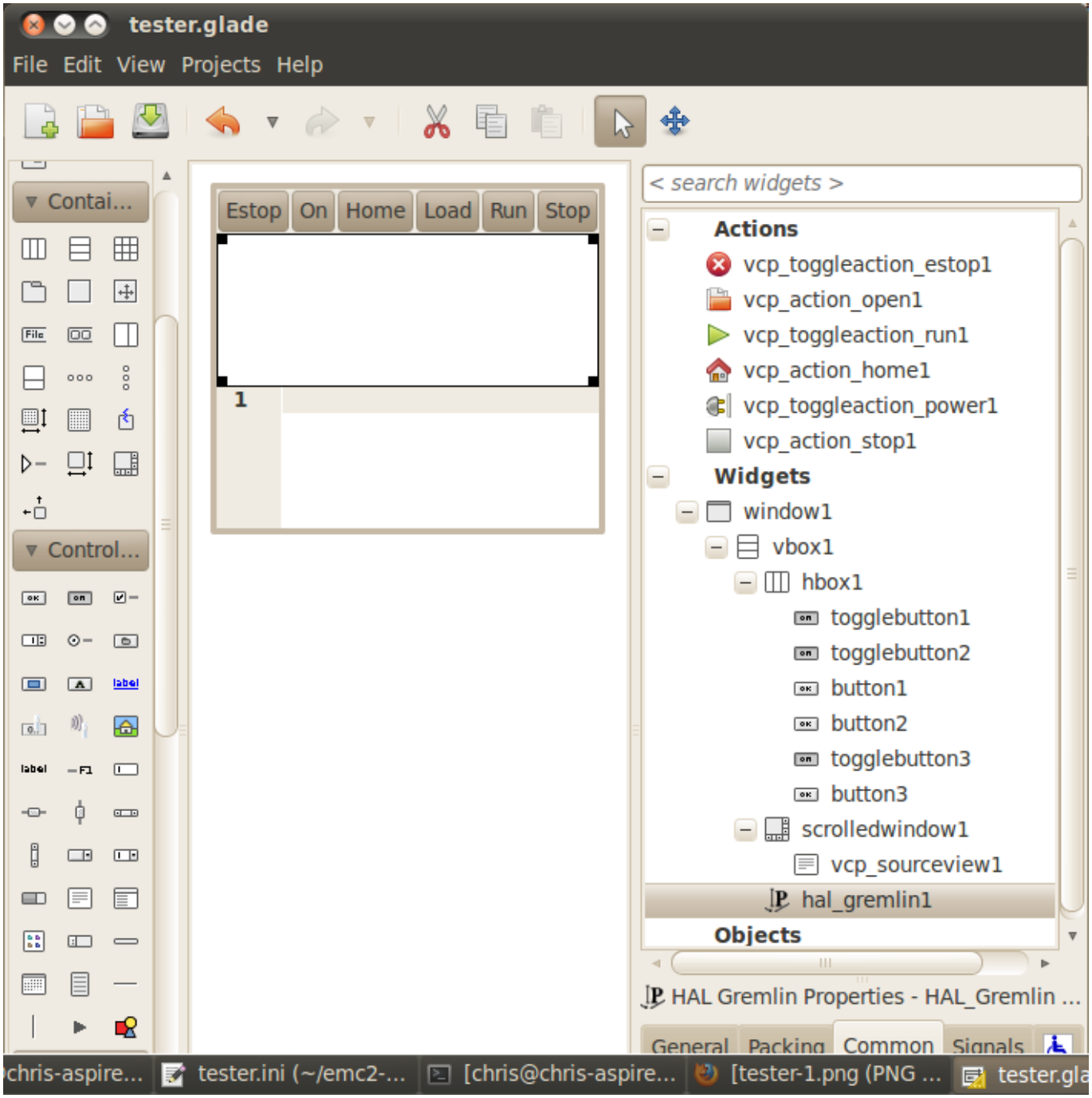


Lassen Sie uns einen einfachen brauchbaren Bildschirm erstellen. Erstellen Sie diesen im Glade-Editor (wenn Sie ein RIP-Paket verwenden, führen Sie ihn von einem Terminal aus, nachdem Sie . scripts/rip-environment verwendet haben).

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(ha 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.

10.4.4.1 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_HAL aufzurufen. Dann fügen wir die Funktion zur Handle-Datei hinzu, um eine Gscreen-Builtin-Funktion zum Starten von Halmeter aufzurufen.

10.4.4.2 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.

10.4.4.3 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 Userspace-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(sel*
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 gedruckt.
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 dem Schalter -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.
  - have three 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
```

10.4.7.1 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 3/4-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.

Anmerkung
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.

10.5.1.1 QtDragon

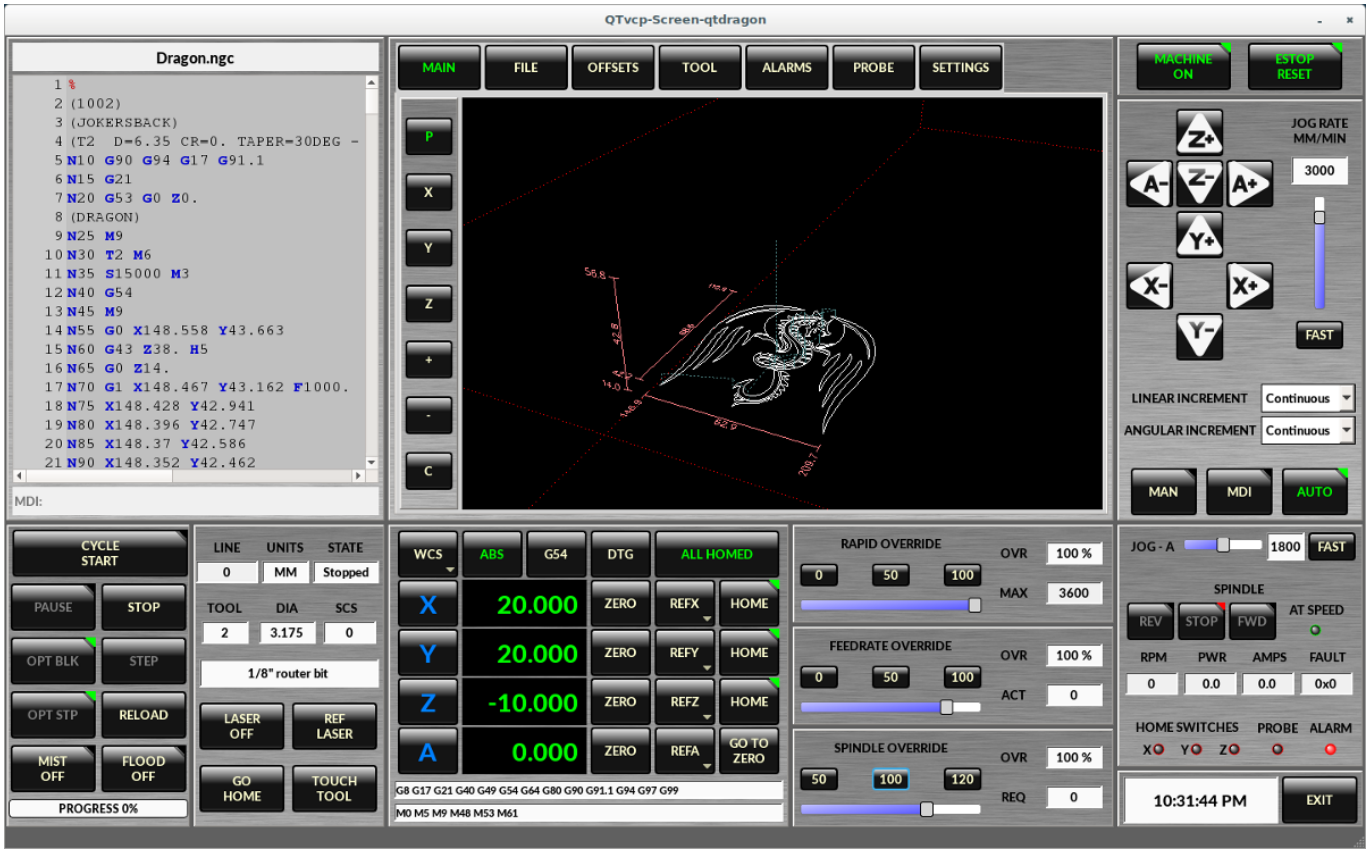


Abbildung 10.30: QtDragon - 3 oder 4 Achsen Muster (1440x860) im Silber-look (eigentlich engl. theme)

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.

10.5.1.2 QtDragon_hd

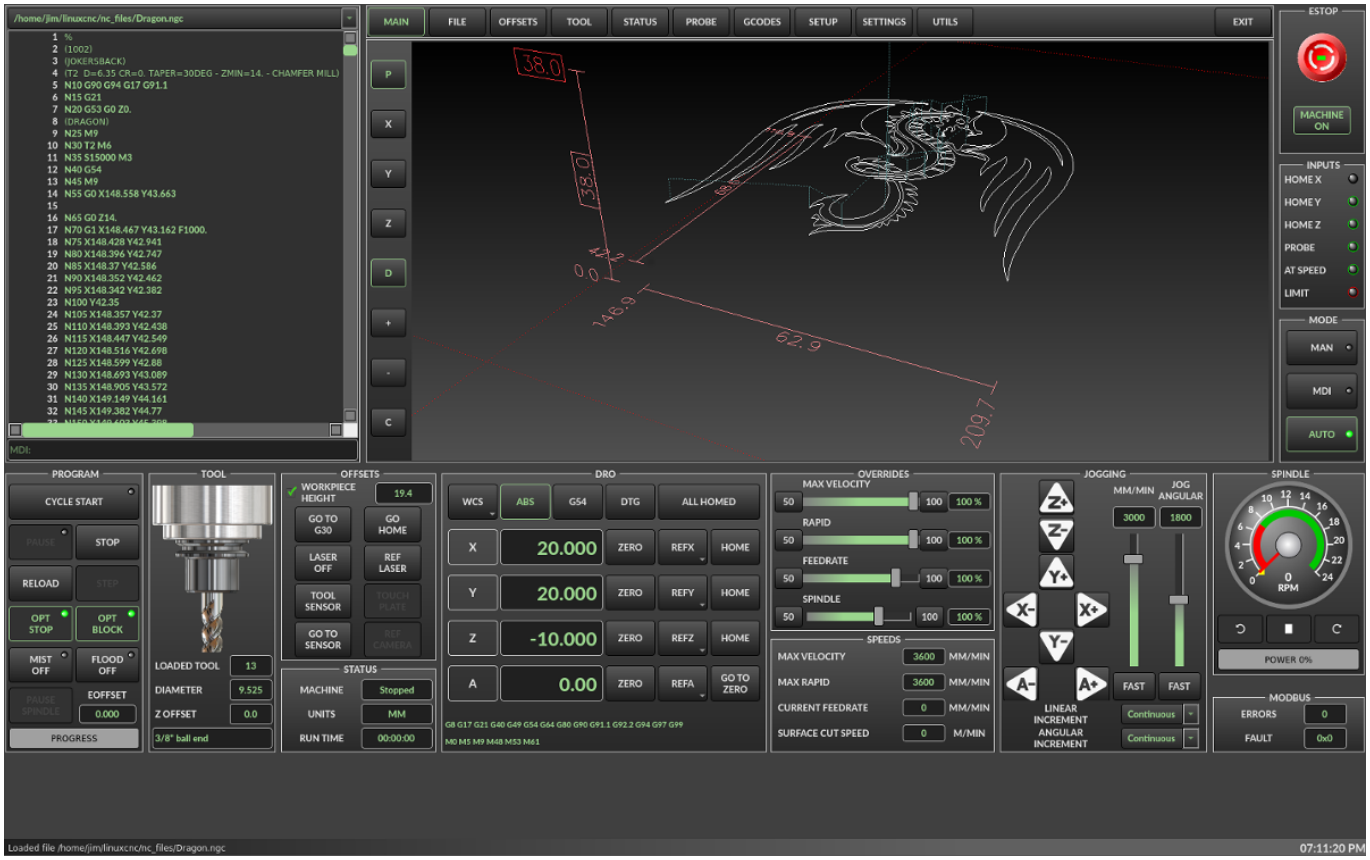


Abbildung 10.31: QtDragon_hd - 3 oder 4 Achsen-Beispiel für größere Monitore (1920x1056) im dunklen Thema

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 has a resolution of 1920x1056 and is not realizable. It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

10.5.2 Erste Schritte

Wenn Ihre Konfiguration derzeit nicht für die Verwendung von QtDragon eingerichtet ist, können Sie sie ändern, indem Sie den Abschnitt [DISPLAY] der INI-Datei bearbeiten. Eine ausführliche Liste der Optionen finden Sie im Abschnitt Display der INI-Datei-Dokumentation.

10.5.2.1 Anzeige (engl. display)

Im Abschnitt *[DISPLAY]* ändern Sie die Zeile *DISPLAY* wie folgt:

- `qtdragon` für eine kleine Version
- `qtdradon_hd` für die große Version.

Sie können `-c-` oder `-v` für die Debug-Ausgabe im Terminal hinzufügen.

```
[DISPLAY]
DISPLAY = qtvcp qtdragon
```

10.5.2.2 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. 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
```

10.5.2.3 Protokollierung (engl. logging)

Sie können angeben, wo der Verlauf/die Protokolle gespeichert werden sollen. Im Abschnitt `[DISPLAY]` hinzufügen:

```
[DISPLAY]
MDI_HISTORY_FILE = mdi_history.dat
MACHINE_LOG_PATH = machine_log.dat
LOG_FILE = qtdragon.log
```

10.5.2.4 Override-Kontrollen

Override-Steuerungen einstellen (1,0 = 100 Prozent):

```
[DISPLAY]
MAX_SPINDLE_0_OVERRIDE = 1.5
MIN_SPINDLE_0_OVERRIDE = .5
MAX_FEED_OVERRIDE       = 1.2
```

10.5.2.5 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
```

10.5.2.6 Jogging-Inkremente

Wählbare Jogging-Inkremente festlegen

```
[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
```

10.5.2.7 Jog-Geschwindigkeit

Festlegen der Jog-Geschwindigkeitssteuerung (in Einheiten pro Minute)

```
[DISPLAY]
MIN_LINEAR_VELOCITY     = 0
MAX_LINEAR_VELOCITY     = 60.00
DEFAULT_LINEAR_VELOCITY = 50.0
```

10.5.2.8 Dialogsystem für Benutzermeldungen

Optional popup custom message dialogs, controlled by HAL pins. MESSAGE_TYPE can be *okdialog* or *yesnodialog*. See qtvcp/library/messages for more information This example shows how to make a dialog that requires the user to select *ok* to acknowledge and hide.

```
[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
```

10.5.2.9 Benutzerdefinierte VCP-Panels einbetten

You can embed QtVCP Virtual Control Panels into the QtDragon or QtDragon_hd screen.
These panels can be either user built or builtin QtVCP Panels.
The `TAB_NAME` entry will used as the title for the new tab.
Tab `TAB_LOCATION` options include: `tabWidget_utilities` and `tabWidget_setup`. See QtVCP/VCP panels for other available builtin panels.

Dieses Beispiel fügt ein eingebautes Panel hinzu; eine grafisch animierte Maschine, die Vismach-Bibliothek nutzend.

```
[DISPLAY]
EMBED_TAB_NAME = Vismach Demo
EMBED_TAB_COMMAND = qtvcp vismach_mill_xyz
EMBED_TAB_LOCATION = tabWidget_utilities
```

10.5.2.10 Vorschau Kontrolle

Magic comments can be used to control the G-code preview.
On very large programs the preview can take a long time to load. You can control what is shown and what is hidden the the graphics screen by adding the appropriate comments from this list into your G-code:

```
(PREVIEW,stop)
(PREVIEW,hide)
(PREVIEW,show)
```

10.5.2.11 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 File (*.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.

Der Abschnitt *[FILTER]* der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine Zeile ”PROGRAM_EXTENSION”. Geben Sie dann 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 von LinuxCNC ausgeführt werden, wenn *Run*. Die folgenden Zeilen fügen Sie Unterstützung für den `image-to-gcode` (engl. für Bild zu G-Code) -Konverter mit LinuxCNC enthalten und die Ausführung von Python-basierte Filter-Programme:

```
[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
```

10.5.2.12 Sonden-/Touchplate-/Lasereinstellungen

QtDragon hat benutzerdefinierte INI-Einträge für die erforderlichen Einstellungen.

```
[TOOLSENSOR]
MAXPROBE = 40
SEARCH_VEL = 200
PROBE_VEL = 50
TOUCH = 29.7

[LASER]
X = 106.9
Y = -16.85
```

QtDragon hat zwei optionale Sondierungs-Tab-Screens zur Verfügung. Kommentieren/unkommentieren Sie, was immer Sie bevorzugen.

Versa probe is a Qtvcp ported version of a popular Gladevcp probing panel. *Basic Probe* is a Qtvcp ported version based on the third party basic probe screen. Both do similar probing routines.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

10.5.2.13 Makro-Buttons

QtDragon hat bis zu zehn praktische Makro-Buttons. In den Beispielkonfigurationen sind sie so beschriftet, dass sie zwischen dem Ursprung des aktuellen Benutzersystems (Nullpunkt) und dem Ursprung des Maschinensystems wechseln. Der Benutzerursprung ist der erste MDI-Befehl in der INI-Liste, der Maschinenursprung der zweite. Diese können auch OWord-Routinen aufrufen, falls gewünscht. Dieses Beispiel zeigt, wie man die Z-Achse zuerst nach oben bewegt. Die Befehle sind durch ein *;* getrennt. Die Beschriftung wird nach dem Komma gesetzt. Die Symbole *\n* fügen einen Zeilenumbruch hinzu.

```
[MDI_COMMAND_LIST]
# for macro buttons
MDI_COMMAND = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND = G53 G0 Z0;G53 G0 X0 Y0,Goto\nMachn\nZero
```

10.5.2.14 Integrierte Beispielkonfigurationen

Die Beispielkonfiguration ”sim/qtvcp_screens/qtdragon/qtdragon_xyza.ini” ist bereits so konfiguriert, dass sie QtDragon als Front-End verwendet.
Es gibt mehrere andere, um verschiedene Maschinenkonfigurationen zu demonstrieren.

10.5.3 Tastenbelegungen

QtDragon ist nicht dafür gedacht, primär eine Tastatur zur Maschinensteuerung zu verwenden. Es fehlen viele Tastenkombinationen, die zum Beispiel AXIS hat - aber Sie können eine Maus verwenden. Es gibt mehrere Tastendrücke für eine bequeme Bedienung 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:

- Klicken Sie auf den MAIN page Button
- The curently selected 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 manuelle Aufforderung zum Werkzeugwechsel benötigen, fügen Sie diese Zeilen in Ihre postgui-Datei ein.

```
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
```

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. (Sie müssen auch die MAX_SPINDLE_POWER in der INI einstellen)

```
qtdragon.spindle-modbus-errors
qtdragon.spindle-amps
qtdragon.spindle-fault
qtdragon.spindle-volts
```

Dieser Bit-Pin ist ein Ausgang für die Spindelsteuerung, um sie anzuhalten: Sie verbinden ihn mit spindle.0.inhibit.

```
qtdragon.spindle-inhibit
```

Dieser Bit-Ausgangspin kann angeschlossen werden, um einen Laser einzuschalten:

```
qtdragon.btn-laser-on
```

Dieser Float-Ausgangspin zeigt die Kameradrehung in Grad an:

```
qtdragon.cam-rotation
```

Diese bit/s32-Pins beziehen sich auf externe Offsets, wenn sie verwendet werden:

```
lqtdragon.eoffset-clear
qtdragon.eoffset-count
qtdragon.eoffset-enable
qtdragon.eoffset-value
```

Diese float-Ausgangspins spiegeln die aktuelle Jograte des Schiebers (in Maschineneinheiten) wider:

```
qtdragon.slider-jogspeed-linear
qtdragon.slider-jogspeed-angular
```

Diese float-Ausgangsstifte geben die aktuellen Schieberegler-Übersteuerungsraten wieder:

```
qtdragon.slider-override-feed
qtdragon.slider-override-maxv
qtdragon.slider-override-rapid
qtdragon.slider-override-spindle
```

Diese Pins 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.

```
qtdragon.versaprobe-blockheight
qtdragon.versaprobe-probeheight
qtdragon.versaprobe-probevel
qtdragon.versaprobe-searchvel
```

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 öffnen, um Sie anzuweisen. Sie müssen den richtigen HAL-Pin in der postgui HAL-Datei anschließen. Zum Beispiel:

```
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 Pausieren der Spindel

QtDragon kann so eingestellt werden, dass die Z-Achse automatisch angehoben und abgesenkt wird, wenn die Spindel angehalten wird. Wenn ein Programm angehalten wird, drücken Sie die Schaltfläche ”Spindelpause”, um die Spindel anzuhalten und in Z anzuheben. Drücken Sie die Schaltfläche erneut, um die Spindel zu starten und abzusenken, und beenden Sie dann das Programm. Der Betrag zum Anheben und Absenken wird auf der Registerkarte ”Einstellungen” unter der Überschrift ”Z Ext Offset” festgelegt. Dazu sind Ergänzungen in der INI und der Datei qtdragon_post_gui erforderlich.

In der INI, unter der Überschrift AXIS_Z.

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

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-count => axis.z.eoffset-counts
net eoffset          qtdragon.eoffset-value <= axis.z.eoffset

# uncomment für 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.

Anmerkung
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 Userspace-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.

10.5.11.1 Verwendung von G-code Ripper für die Z-Ebenen-Kompensation

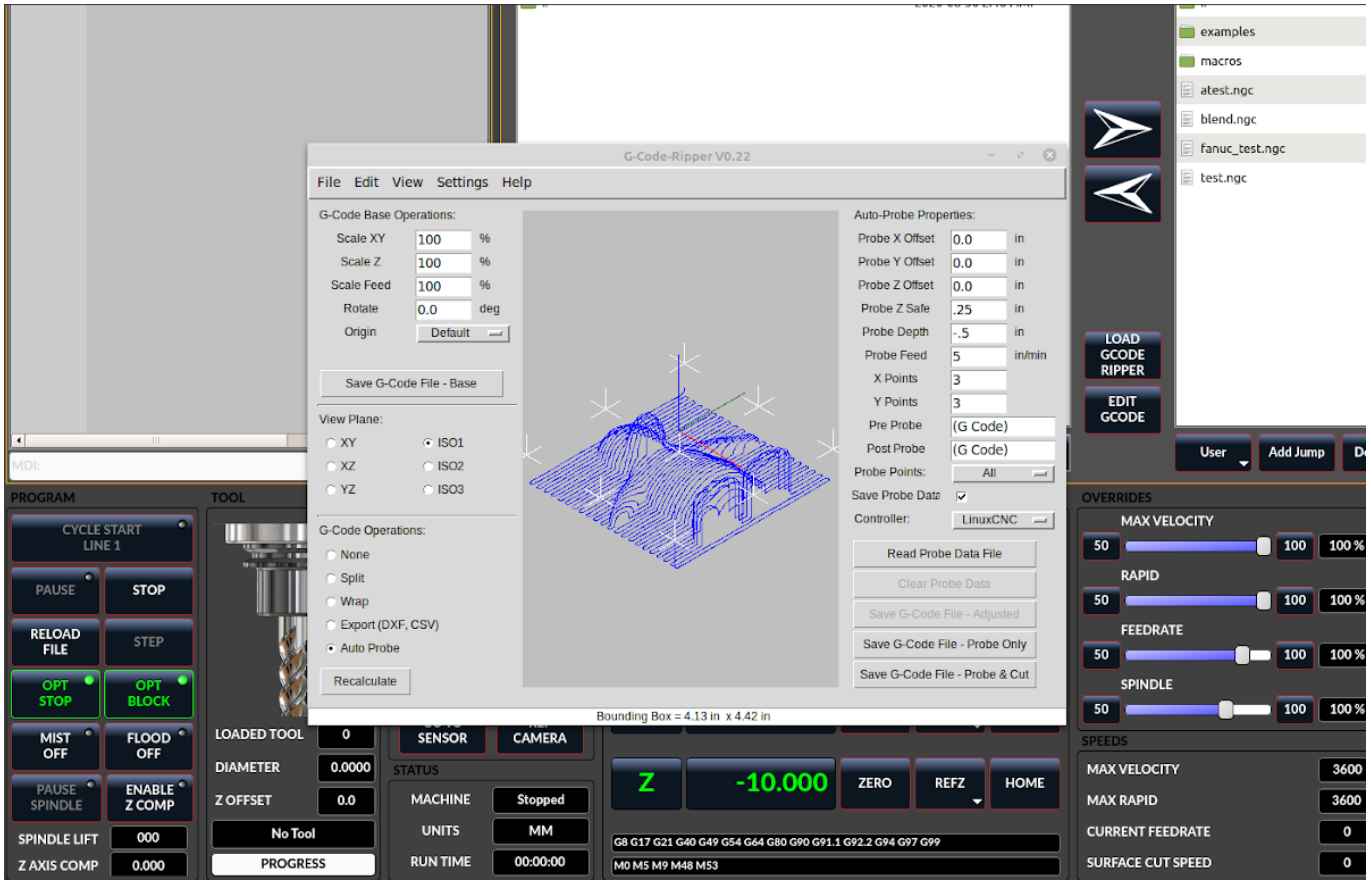


Abbildung 10.32: QtDragon_hd zeigt G-code Ripper

Anmerkung
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
-

- Without changing the offsets, run this program. Make sure the probe tool is installed. When complete, there will be a file in the config directory called *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.

Anmerkung
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
#*****************************************************************************

# load a summing component for adding spindle lift and Z compensation
loadrt scaled_s32_sums
addf scaled-s32-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.method 1
setp z_level_compensation.fade-height 0.0

# 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-spindle-count <= qtdragon.eoffset-spindle-count

# Z level Kompensation
#*****************************************************************************
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

# set up scaled sum component
#*****************************************************************************
net eoffset-spindle-count   scaled-s32-sums.0.in0
net eoffset-zlevel-count    scaled-s32-sums.0.in1       qtdragon.eoffset-value
setp scaled-s32-sums.0.scale0 1000
net eoffset-counts          scaled-s32-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.

Anmerkung
Die Sondierung ist sehr unempfindlich gegenüber Fehlern; überprüfen Sie die Einstellungen vor der Verwendung.

Abbildung 10.33: QtDragon - Grundlegende Sondenoption

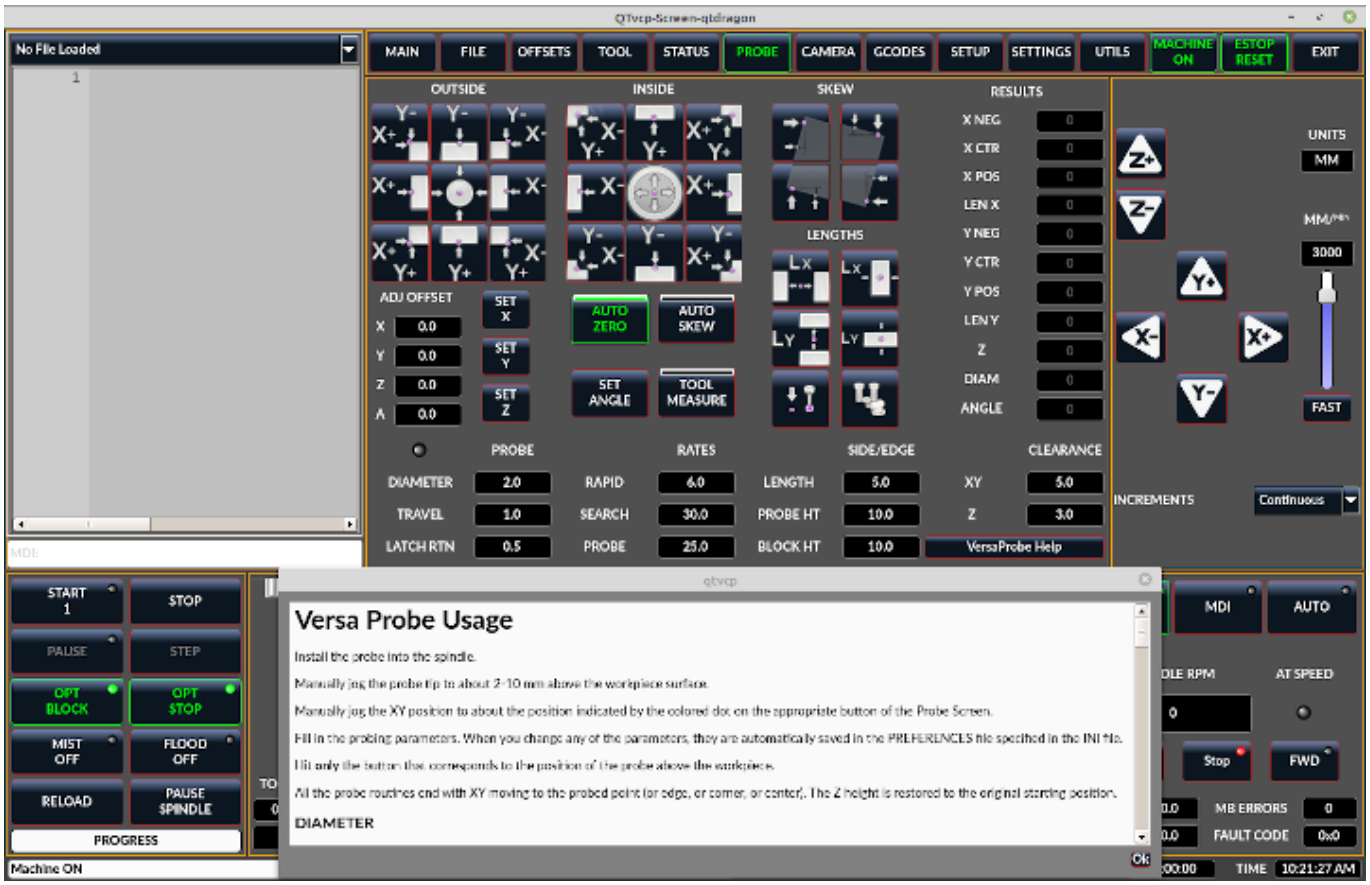


Abbildung 10.34: QtDragon - Versa-Probe-Option

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 und dann das Werkzeug abgesenkt wird, bis es die Platte berührt, wodurch das Messtastersignal ausgelöst wird. Z0 wird auf Tasthöhe - Plattendicke 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, wie weit über dem Tisch der Auslösepunkt des Messtasters liegt (Höhe der Werkzeugeinrichtung) und wie weit über dem Tisch die Oberseite des Werkstücks liegt. 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.

10.5.13 Touch-Platte

Abbildung 10.35: 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.

Anmerkung
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 Werkzeugmessung

QtDragon kann so eingestellt werden, dass es eine integrierte automatische Werkzeugmessung mit dem Versa Probe Widget und Remap-Code durchführt. Um diese Funktion zu nutzen, müssen Sie

einige zusätzliche Einstellungen vornehmen und Sie können den angebotenen HAL-Pin verwenden, um Werte in Ihrer eigenen ngc-Remap-Prozedur zu erhalten.

Wichtig
Vergessen Sie nicht, vor dem ersten Test die Sondenhöhe und die Sondengeschwindigkeiten auf der Seite mit den Sondeneinstellungen einzugeben.

Die Werkzeugmessung in QtDragon erfolgt in den folgenden Schritten:

- Berührung des Werkstücks in X und Y.
- Messen Sie die Höhe Ihres Blocks von der Basis, an der sich Ihr Werkzeugschalter befindet, bis zur Oberseite des Blocks (einschließlich Spannfutter usw.).
- Geben Sie auf der Registerkarte Versa-Sonde den Messwert für die Blockhöhe ein.
- Stellen Sie sicher, dass die Schaltfläche ”Werkzeugmessung verwenden” in der Registerkarte ”Vesa-Sonde” aktiviert ist.
- Gehen Sie in den Automatikmodus und starten Sie Ihr Programm.

Anmerkung
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.

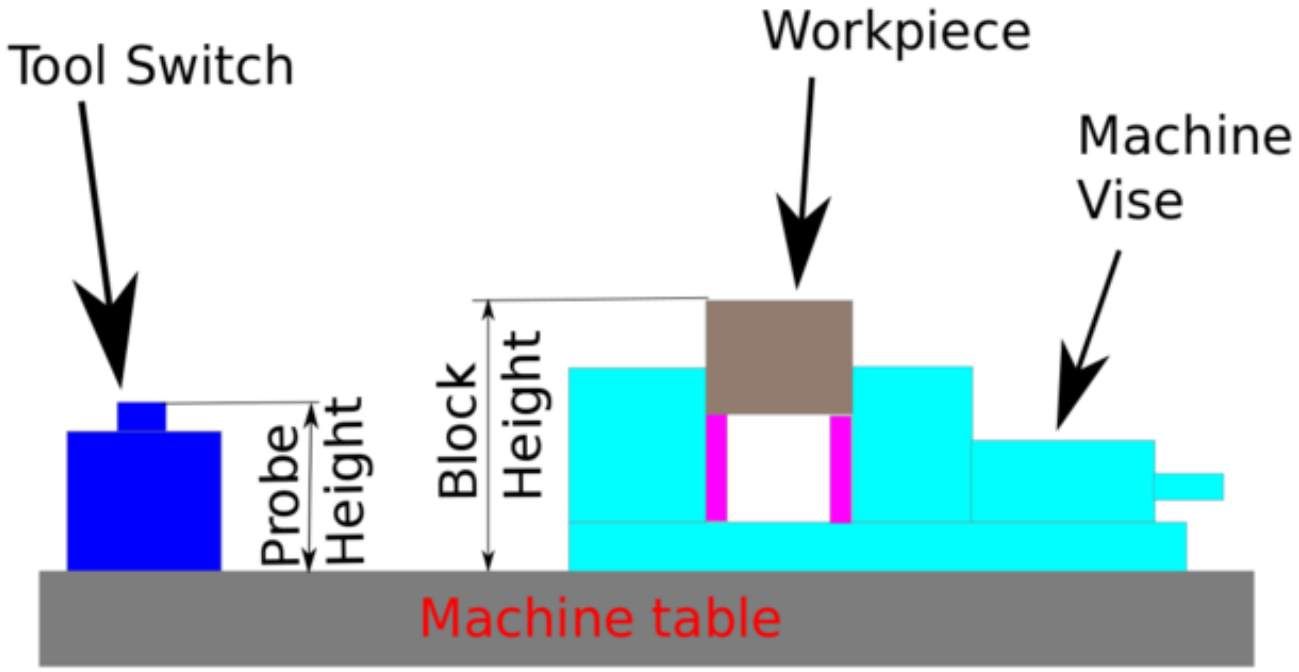


Abbildung 10.36: Automatische Werkzeugvermessung

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.

Anmerkung
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.

10.5.14.1 Werkstückhöhe Antasten

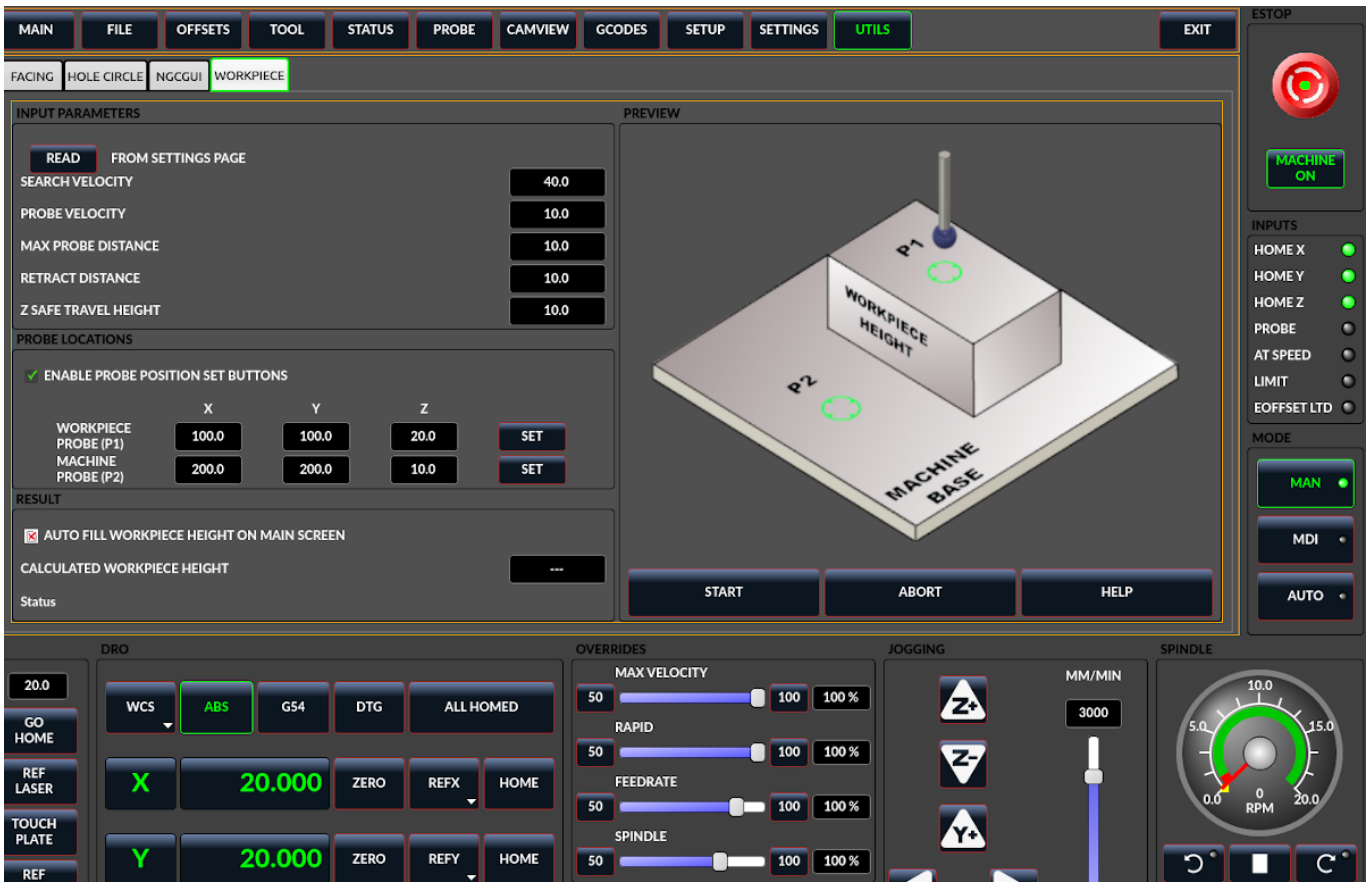


Abbildung 10.37: QtDragon_hd - Werkstück Höhenabtastung

Dieses Programm tastet 2 benutzerdefinierte Positionen in der Z-Achse an und berechnet die Höhendifferenz.

Anmerkung
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

- When checked, the calculated height is automatically transferred to the Workpiece Height field in the main screen.
- 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
- Any 2 points within the machine operating volume can be specified.
- If the first point is higher than the second, the calculated height will be a positive number.
- If the first point is lower than the second, the calculated height will be a negative number.
- Units are irrelevant in this program. The probed values are not saved and only the difference is reported.

Achtung
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.

10.5.14.2 Werkzeugmess-Pins

Versaprobe bietet 5 Pins für die Werkzeugmessung. Die Pins werden von einem Remap-G-Code-Unterprogra
gelesen, so dass der Code auf verschiedene Werte reagieren kann.

- `qtversaprobe.toolmeasurement` (HAL_BIT) Werkzeugmessung aktivieren oder nicht
- `qtversaprobe.blockheight` (HAL_FLOAT) der gemessene Wert der Oberseite des Werkstücks
- `qtversaprobe.probeheight` (HAL_FLOAT) die Höhe des Sondenschalters
- `qtversaprobe.searchvel` (HAL_FLOAT) die Geschwindigkeit, mit der nach dem Schalter für den Werkzeugmesstaster gesucht wird
- `gmoccapy.probevel` (HAL_FLOAT) die Länge des Werkzeugs zwischen Geschwindigkeit und Messtaster

10.5.14.3 Änderungen an der INI-Datei für Werkzeugmessungen

Ändern Sie Ihre INI-Datei so, dass sie Folgendes enthält:

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
```

```
[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

# The remap code for QtVCP's versaprobe's automatic tool probe of Z
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

The abort command file should be in the configuration folder and look something like this:

```
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
```

Die Position des Werkzeugsensors und die Startposition der Antastbewegung, alle Werte sind absolute (G53) Koordinaten, mit Ausnahme von MAXPROBE, das in relativer Bewegung angegeben werden muss. Alle Werte sind in maschineneigenen Einheiten.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE =  -20
```

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
```

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:

```
[PYTHON]
# The path to start a search for user modules
PATH_PREPEND = python
# The start point for all.
TOPLEVEL = python/toplevel.py
```

10.5.14.4 Benötigte Dateien

Sie müssen die folgenden Dateien in Ihr Konfigurationsverzeichnis kopieren

Erstellen Sie zunächst einen Ordner namens *python* im Konfigurationsordner Ihres Rechners.

Wenn Sie eine kompilierte RIP-Version von LinuxCNC verwenden:
Kopieren Sie aus ”IHR-LINUXCNC-DIRECTORY/configs/sim/QtDragon/python” die Dateien ”toplevel.py” und ”remap.py” in den neuen ”python”-Ordner Ihrer Konfiguration.

Wenn Sie eine installierte Version von LinuxCNC verwenden:
Kopieren Sie von */usr/share/doc/linuxcnc/examples/sample-configs/sim/qtvcp_screens/qtdragon/python/* die Dateien *toplevel.py* und *remap.py* in den neuen *python*-Ordner Ihrer Konfiguration.

Alternativ können Sie mit einem Texteditor neue Dateien in Ihrem *python*-Ordner erstellen, die Sie in Ihrem Konfigurationsordner angelegt haben.

Eine mit dem Namen *remap.py*, die mit diesem Text gespeichert wurde:

```
from stdglue import *
```

Eine Datei mit dem Namen *toplevel.py* wird mit diesem Text gespeichert:

```
import remap
```

Erstellen Sie eine symbolischen link oder kopieren Sie die folgenden Dateien in den oben beschriebenen Ordner *python*.

Im Ordner ”~/linuxcnc/nc_files/examples/remap_subroutine/”.

Im Ordner *~/linuxcnc/nc_files/examples/remap_lib/python_stdglue/*

Anmerkung
Diese Dateinamen und Speicherort könnte unterschiedlich sein, je nach installierten Versen Entwicklung (RIP) Version von LinuxCNC. Zum Beispiel *–/linuxcnc/nc_files/macros* ist *–/linuxcnc/nc_files/examples/macros* in installierten Versionen von LinuxCNC. Sie können angepasste Versionen der gleichen Dateien verwenden oder sie anders benennen. Die Einträge in der *[RS274NGC]* Abschnitt diktieren, um LinuxCNC, was und wo zu suchen. Die angegebenen Namen und Speicherorte sollten in beiden Systemen standardmäßig verfügbar sein.

10.5.14.5 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.

Anmerkung
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 und in der INI-Datei angegebenen Werte setzt.

10.5.17 Beschreibung der Registerkarten

Über die Registerkarten kann der Benutzer die am besten geeigneten Informationen/Steuerelemente in den oberen drei Feldern auswählen.
Wenn die Bildschirmtastatur eingeblendet ist und der Benutzer sie ausblenden, aber die aktuelle Registerkarte beibehalten möchte,
kann er dies tun, indem er die aktuelle Registerkarte einblendet.

10.5.17.1 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.

10.5.17.2 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 *Gcode Ripper* laden

10.5.17.3 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.

10.5.17.4 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.

10.5.17.5 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.

10.5.17.6 Registerkarte ”Sonde

Auf dieser Registerkarte werden Optionen für Prüfroutinen angezeigt. Abhängig von den INI-Optionen kann dies sein
VersaProbe- oder BasicProbe-Stil sein. Sie sind funktionell ähnlich.
QtDragon_hd zeigt auch ein kleineres Grafikanzeigefenster an.

10.5.17.7 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.

10.5.17.8 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.

10.5.17.9 Registerkarte "Einstellungen"

Es ist möglich, eine HTML- oder PDF-Datei (.html / .pdf-Endung) mit Einrichtungshinweisen zu laden. Die HTML-/PDF-Dokumente werden auf der Registerkarte "Setup" angezeigt. Einige Programme, wie Fusion 360 und Aspire, erstellen diese Dateien für Sie. Wenn Sie ein G-Code-Programm laden und eine HTML/PDF-Datei mit demselben Namen vorhanden ist, wird diese automatisch geladen. Sie können auch Ihre eigenen HTML-Dokumente mit der mitgelieferten Schaltfläche SetUp Writer schreiben.

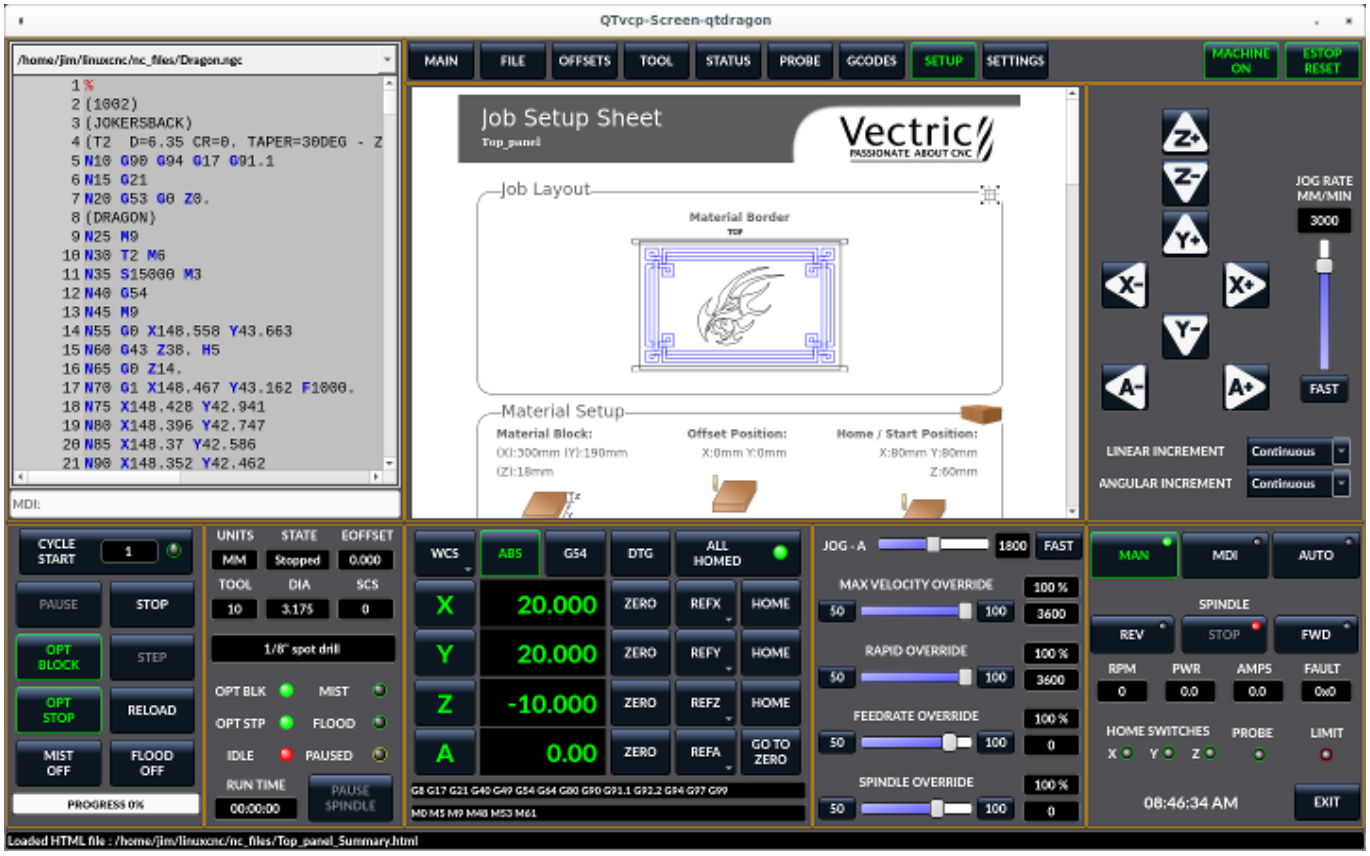


Abbildung 10.38: QtDragon - Beispiel für die Registerkarte Setup

10.5.17.10 Registerkarte ”Einstellungen”

Die Registerkarte ”Einstellungen” dient zum Einstellen der Betriebsoptionen, der Offsets für Messtaster/Tastplatte/Laser/Kamera und zum Laden externer Debugging-Programme.

10.5.17.11 Registerkarte ”Dienstprogramme”

Auf dieser Registerkarte wird eine weitere Auswahl von G-Code-Hilfsprogrammen 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.

10.5.18 Stile

Nearly all aspects of the GUI appearance are configurable via the QtDragon.qss stylesheet file. The file can be edited manually or through the stylesheet dialog widget in the GUI. To call up the dialog, press F12 on the main window. New styles can be applied temporarily and then saved to a new qss file, or overwrite the current qss file.

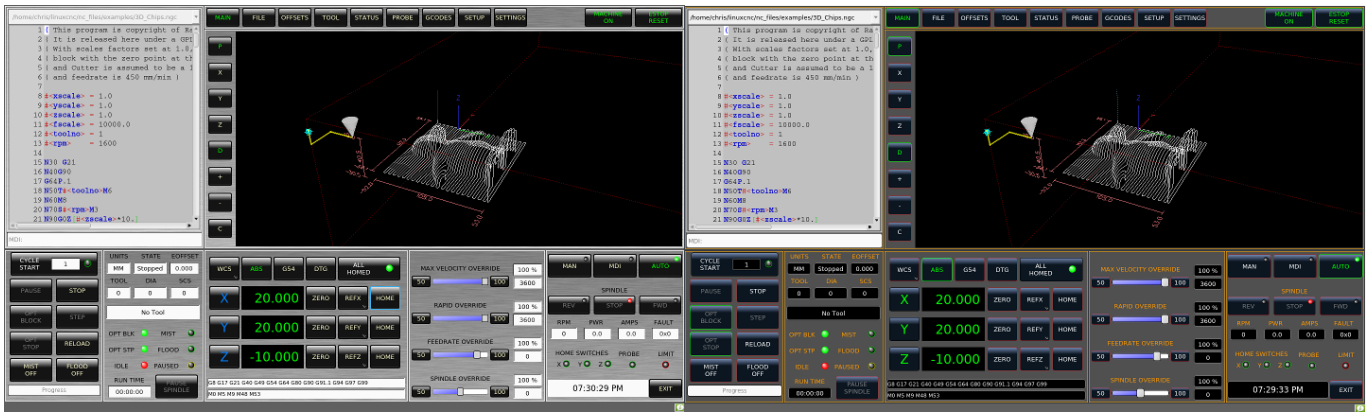


Abbildung 10.39: QtDragon - Zwei Stil-Beispiele

10.5.19 Anpassung

10.5.19.1 Stylesheets

Mit Hilfe von Stylesheets können Sie eine ganze Reihe von Anpassungen vornehmen, aber Sie müssen in der Regel ein wenig über die Widget-Namen wissen. Wenn Sie F12 drücken, wird ein Stylesheet-Editor-Dialog zum Laden/Testen/Speichern von Änderungen angezeigt. Zum Beispiel:

So ändern Sie die DRO-Schriftart (suchen Sie nach diesem Eintrag und ändern Sie den Namen der Schriftart):

```
DROLabel,
StatusLabel#status_rpm {
    border: 1px solid black;
    border-radius: 4px;
    font: 20pt ”Noto Mono”;
}
```

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”;
}
```

10.5.19.2 Qt Designer und Python-Code

Alle Aspekte der grafischen Benutzeroberfläche können mit 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 der erforderlichen Python-Code auf ein Minimum, so dass relativ einfach Änderungen. Die LinuxCNC Website hat eine umfangreiche Dokumentation über die Installation und Verwendung von QtVCP Bibliotheken. QtVCP Overview für weitere Informationen

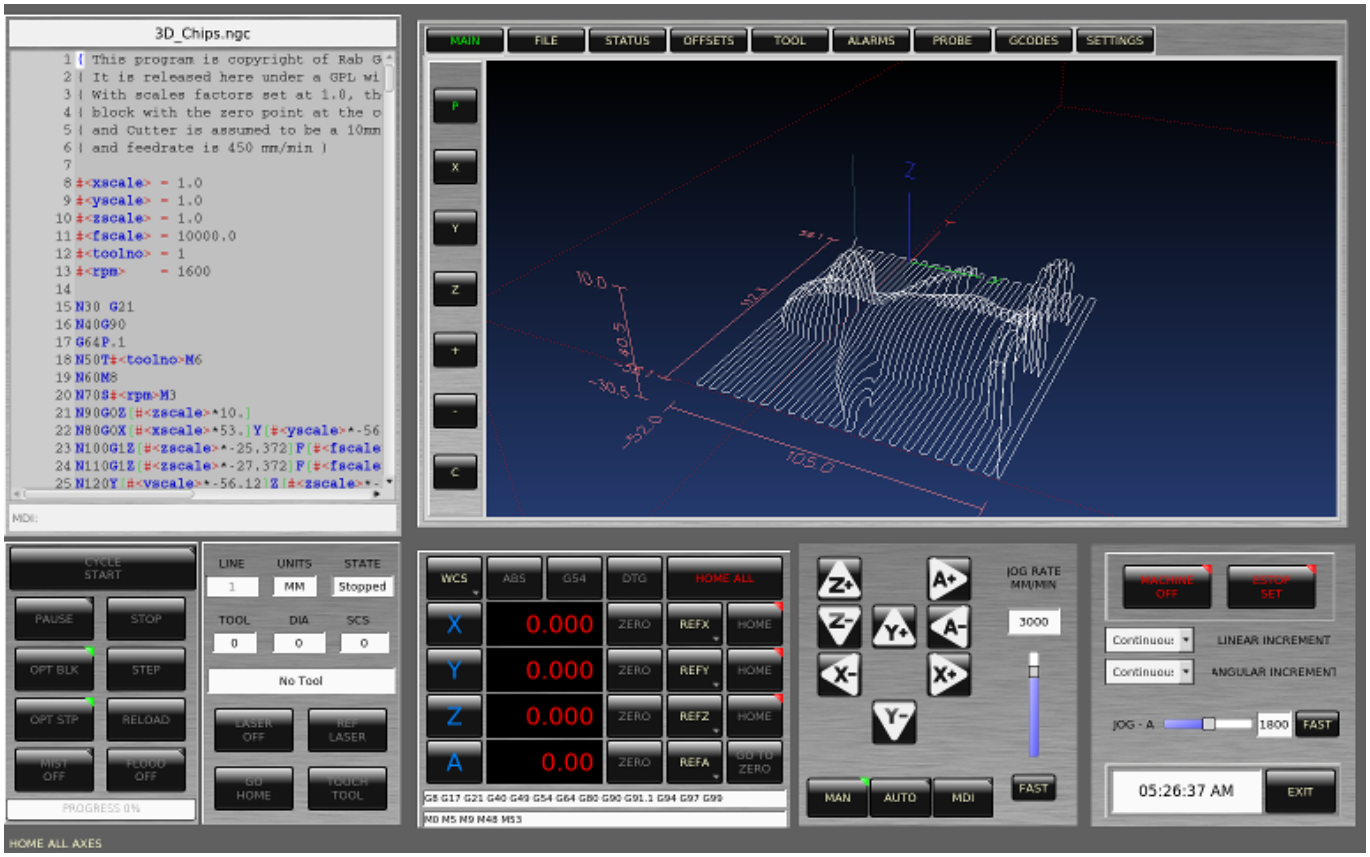


Abbildung 10.40: QtDragon - Angepasster QtDragon

10.6 NGCGUI

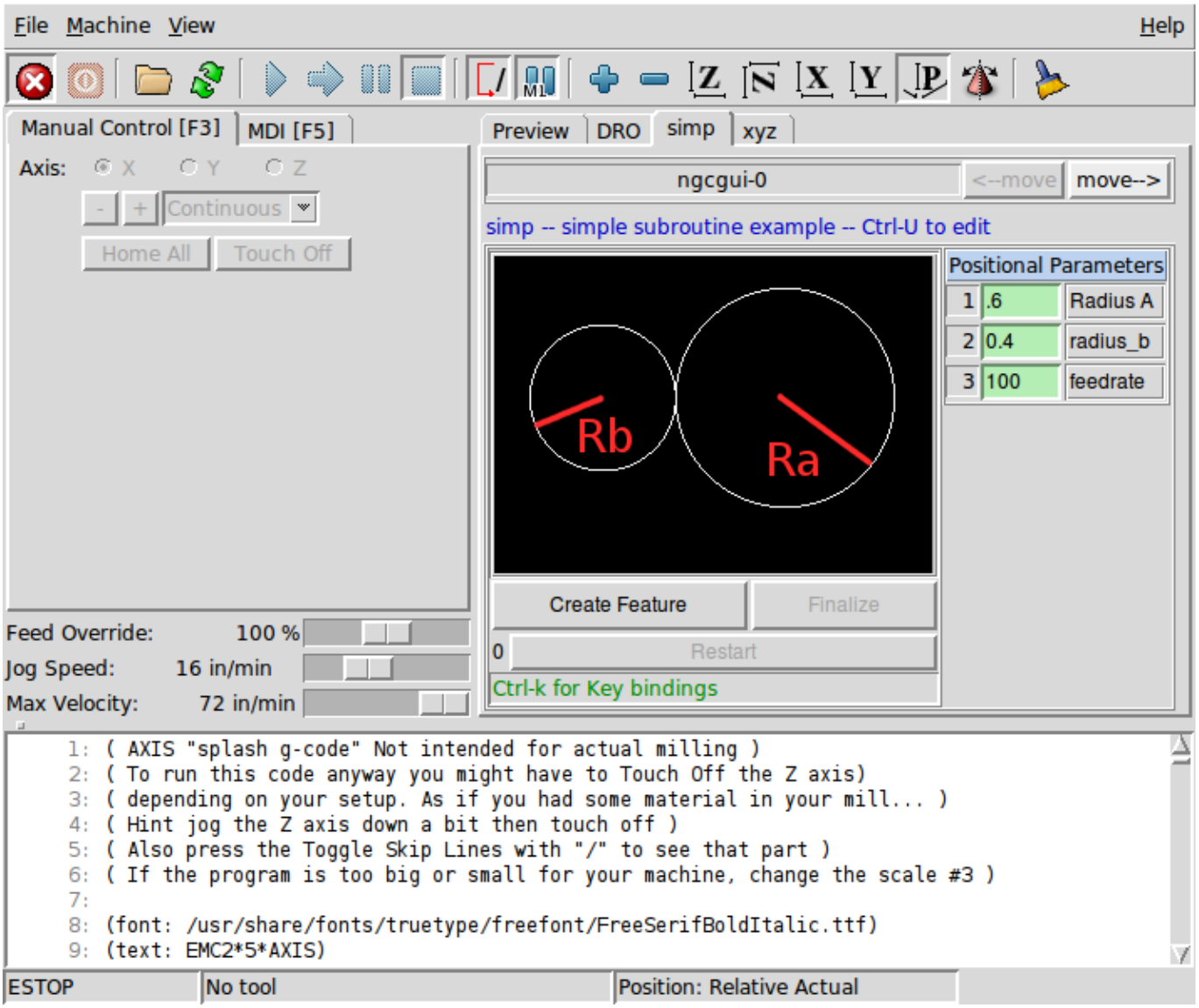


Abbildung 10.41: NGCGUI eingebettet in AXIS

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* is an alternate, Python implementation of 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: http://www.vagrearg.org/content/gcmc)

Anmerkung
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-Unterprogramm
(.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* - machine exerciser constrained to a box shape
  - *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.

Anmerkung
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
```

Anmerkung
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

10.6.4.1 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 Standard: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)
```

Anmerkung
Als eigenständige Anwendung bearbeitet NGCGUI eine einzelne Unterprogrammdatei, die mehrfach aufgerufen werden kann. Mehrere eigenständige NGCGUI-Anwendungen können unabhängig voneinander gestartet werden.

10.6.4.2 Eigenständiges (engl. standalone) PyNGCGUI

Zur Verwendung geben Sie in ein Terminal ein:

```
pyngcgui --help
Usage:
pyngcgui [Options] [<sub_filename>]
Options requiring values:
    [-d | --demo] [0|1|2] (0: DEMO standalone toplevel)
                          (1: DEMO embed new notebook)
                          (2: DEMO embed within existing notebook)
    [-S | --subfile       <sub file name>]
    [-p | --preamble      <preamble file name>]
    [-P | --postamble     <postamble file name>]
    [-i | --ini           <INI file name>]
    [-a | --autofile      <auto file name>]
    [-t | --test          <testno>]
    [-K | --keyboardfile  <glade_file>] (use custom popupkeyboard glade file)
Solo Options:
    [-v | --verbose]
    [-D | --debug]
    [-N | --nom2]         (no m2 terminator (use %))
    [-n | --noauto]       (save but do not automatically send result)
    [-k | --keyboard]     (use default popupkeybaord)
    [-s | --sendtoaxis]   (send generated NGC file to AXIS GUI)
Notes:
      A set of files is comprised of a preamble, subfile, postamble.
      The preamble and postamble are optional.
      One set of files can be specified from cmdline.
      Multiple sets of files can be specified from an INI file.
      If --ini is NOT specified:
         search for a running LinuxCNC and use its INI file.
```

Anmerkung
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

10.6.5.1 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)

Anmerkung
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.

10.6.5.2 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

Anmerkung
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.

Anmerkung
Das Truetype Tracer GUI-Frontend ist derzeit nicht für GladeVCP-Anwendungen verfügbar.

10.6.5.3 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
```

Anmerkung
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.

10.6.5.4 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.

Item:    [DISPLAY]TTT = path_to_truetype-tracer
Example: [DISPLAY]TTT = truetype-tracer
Note:    Optional, if not specified, attempt to use
         /usr/local/bin/truetype-tracer.
         Specify with absolute pathname or as a simple executable
```

```
         name in which case the user PATH environment will used
         to find the program.

Item:    [DISPLAY]TTT_PREAMBLE = preamble_filename
Example: [DISPLAY]TTT_PREAMBLE = in_std.ngc
Note:    Optional, specifies filename for preamble used for ttt
         created subfiles.
```

10.6.5.5 INI File Path Specifications

NGCGUI uses the LinuxCNC search path to find files.

Der Suchpfad beginnt mit dem durch angegebenen Standardverzeichnis:

```
[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/utilitysubs
```

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.

10.6.5.6 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_li`
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
Example: `[DISPLAY]NGCGUI_PREAMBLE = in_std.ngc`
Note: Optional, when specified, the file is prepended to a subfile.
Files created with ”Custom” tab pages use the preamble specified with the page.

[ANZEIGE] NGCGUI_POSTAMBLE = postamble_filename
Example: `[DISPLAY]NGCGUI_POSTAMBLE = bye.ngc`
Note: Optional, when specified, the file is appended to a subfiles.
Files created with ”Custom” tab pages use the postamble specified with the page.

[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, dann wird das Bild in einem separaten Fenster angezeigt und alle Funktionen stehen auf der Registerkarte zur Verfügung.

Die `NGCGUI_OPTIONS` gelten für alle NGCGUI-Registerkarten mit der Ausnahme, dass die Optionen `nonew`, `noremove` und `noiframe` nicht für "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_incl`
Hinweis: Optional, jedes Verzeichnis wird einbezogen, wenn gcmc aufgerufen wird mit der Option: `--include dirname`.

10.6.6 Dateianforderungen für NGCGUI-Kompatibilität

10.6.6.1 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 wird, dann wird dieser beim Start in das Eingabefeld für den Parameter eingetragen.

Wenn comment_text enthalten ist, wird er zur Identifizierung der Eingabe anstelle des Parameternamens 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.

10.6.6.2 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”)
```

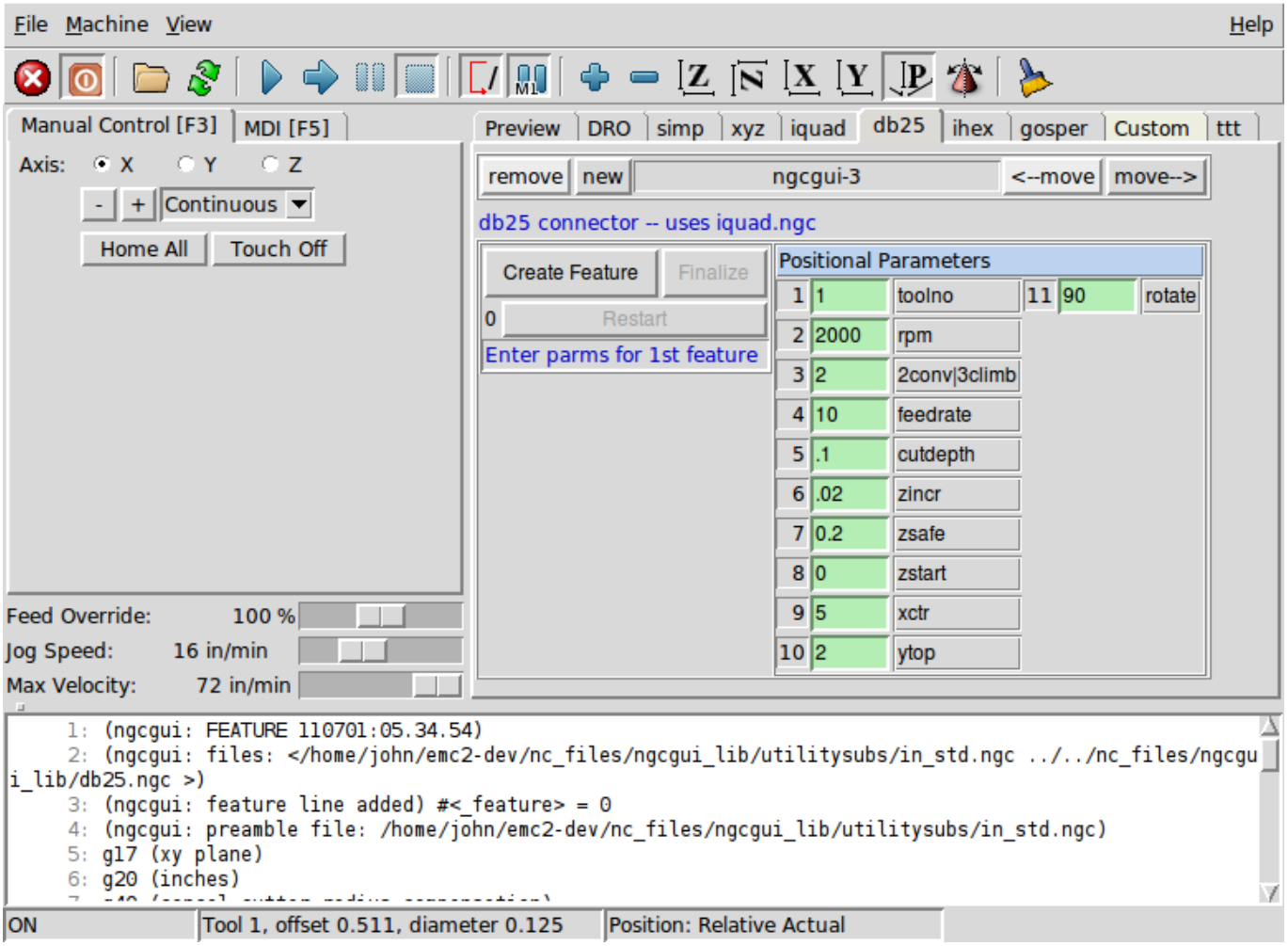
und geben Sie den richtigen Pfad für gcmc include_files in der INI-Datei an, zum Beispiel:

```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

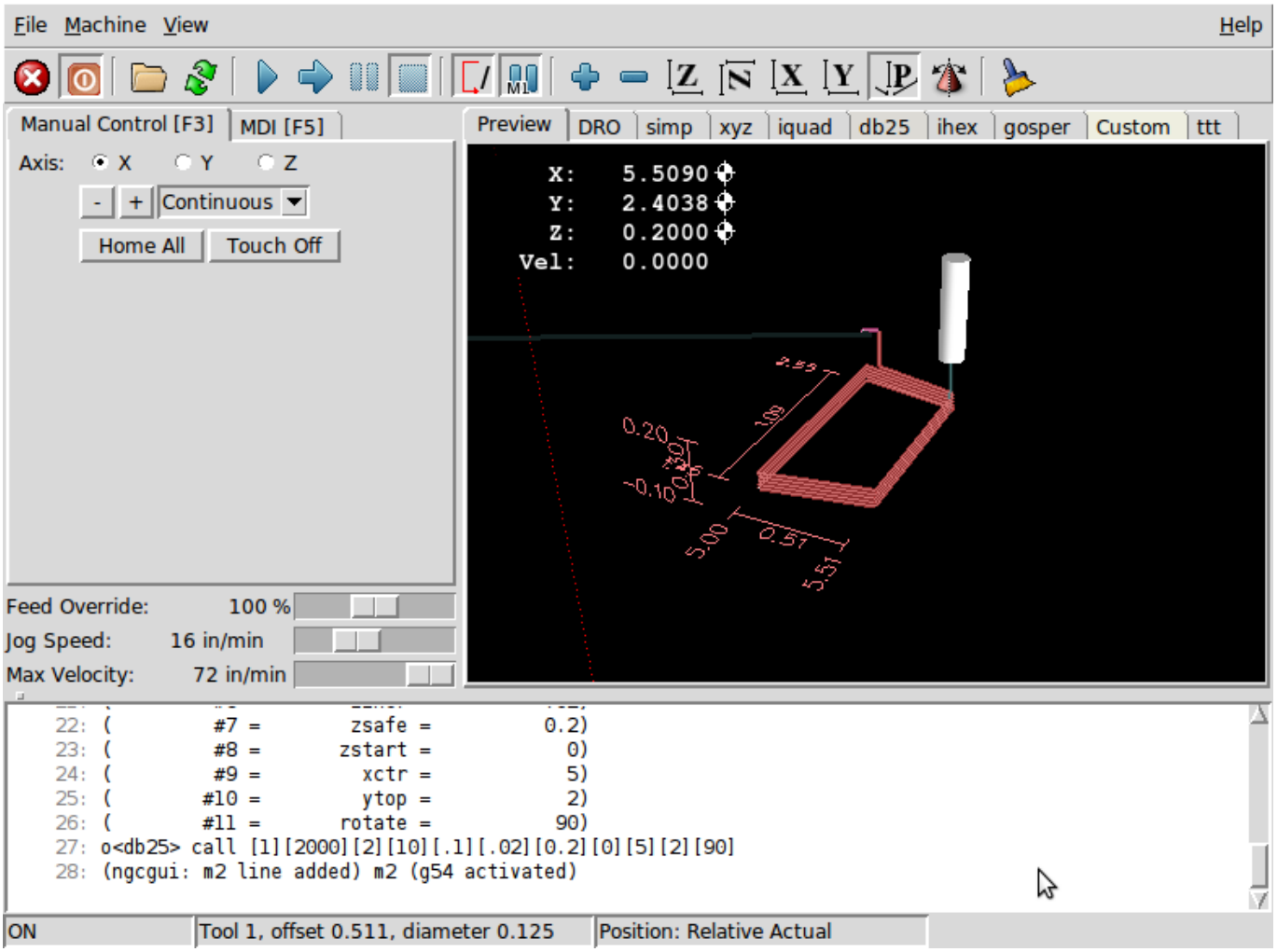
10.6.7 DB25 Beispiel

Das folgende zeigt das DB25-Unterprogramm.

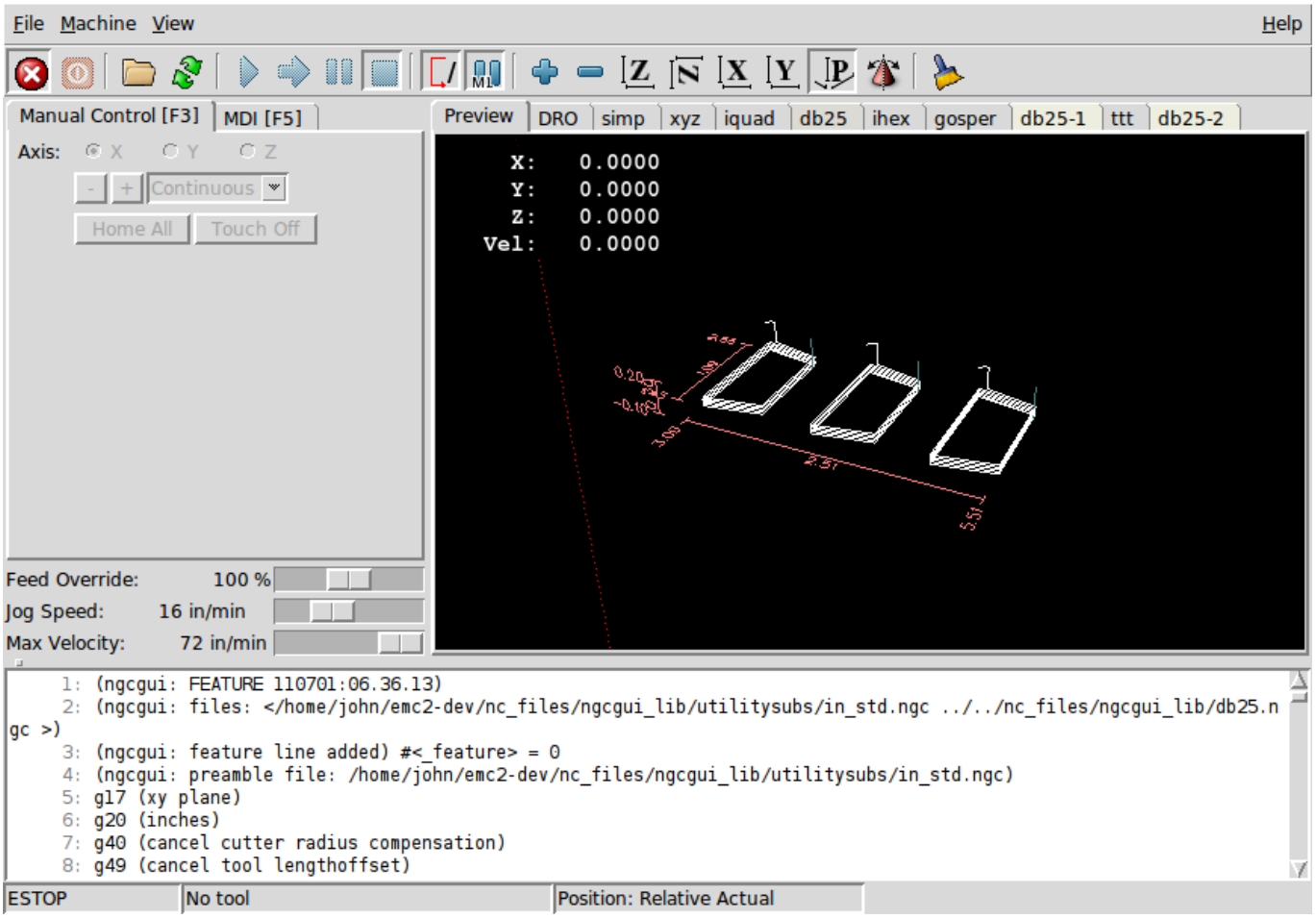
Auf dem ersten Foto sehen Sie, wo Sie die Lücken für jede Variable ausfüllen.



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.

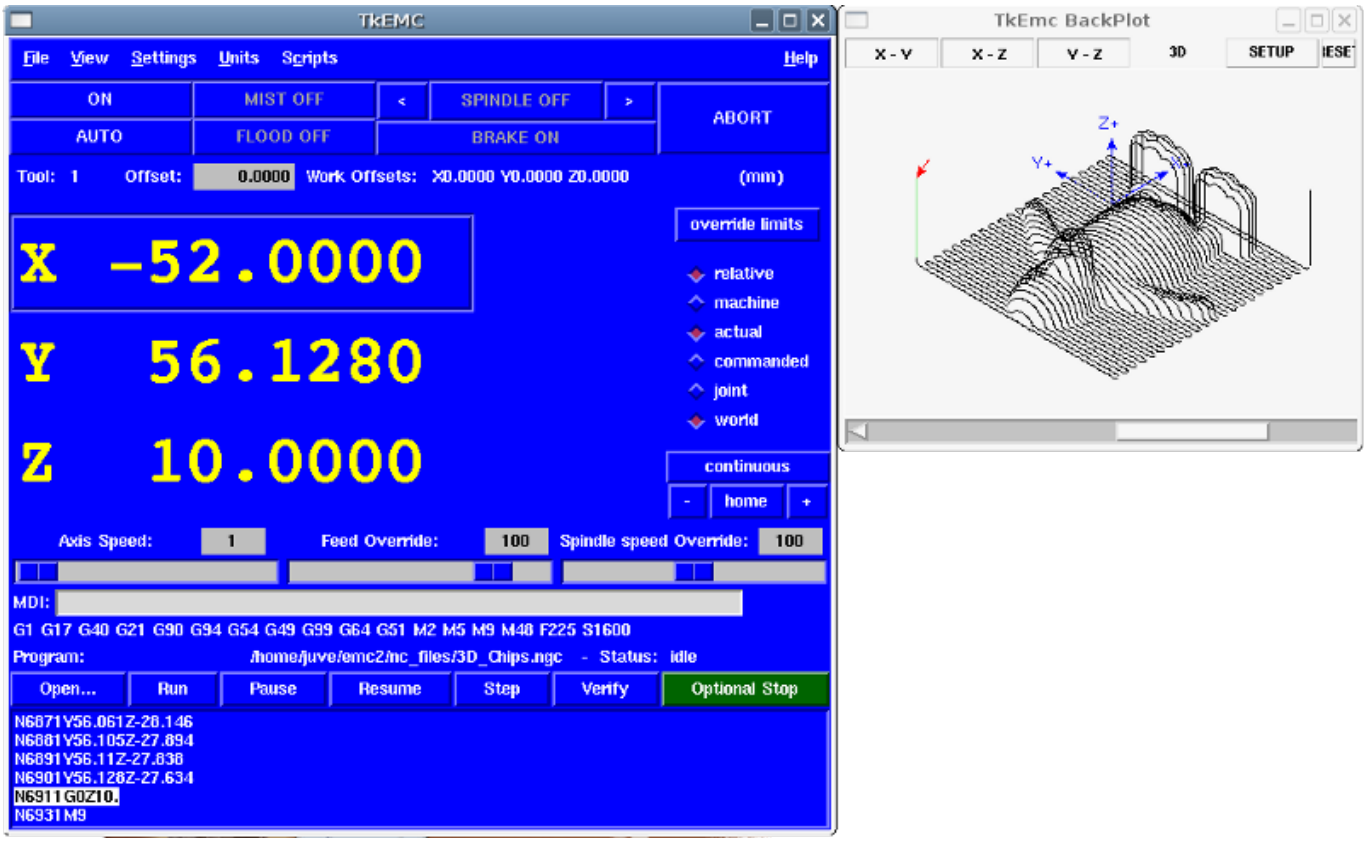


Abbildung 10.42: TkLinuxCNC-Fenster

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.

10.7.2.1 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

10.7.3.1 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
- Abort

dann in der zweiten Zeile:

- Betriebsart: *MANUAL > MDI > AUTO*
- Flutkühlmittel ein-/ausschalten
- Spindelbremse ein-/ausschalten

10.7.3.2 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.

10.7.3.3 Koordinatenanzeigebereich

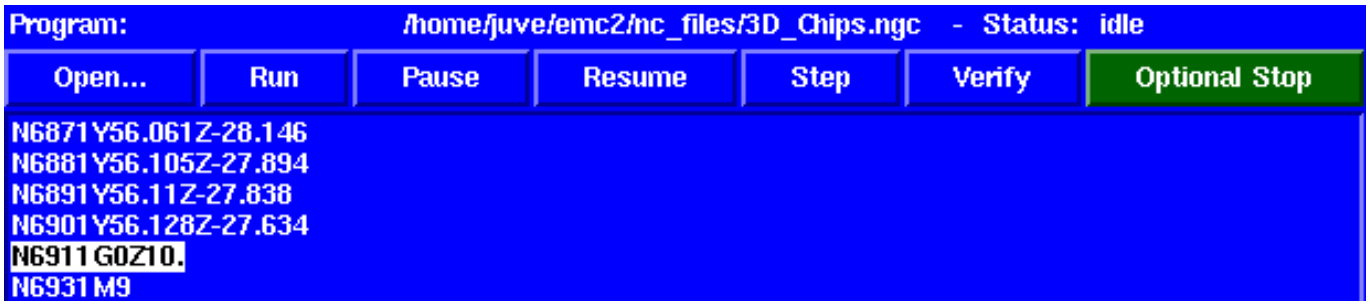
Der Hauptteil der Anzeige zeigt die aktuelle Position des Werkzeugs an. Die Farbe der Positionsanzeige hängt vom Zustand der Achse ab. Wenn die Achse nicht referenziert ist, wird die Achse in gelber Schrift angezeigt. Sobald sie referenziert ist, wird sie in grüner Schrift angezeigt. Wenn es einen Fehler 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.

10.7.3.4 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.

10.7.3.5 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).

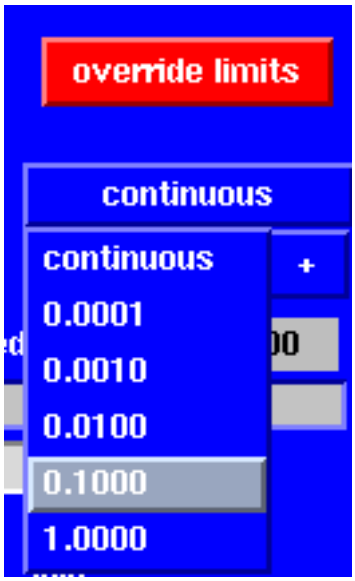


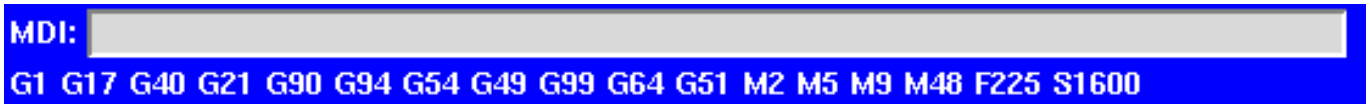
Abbildung 10.43: Beispiel für TkLinuxCNC Override Limits & Jogging Increments

The Spindle groupspindle 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 Kühlmittel 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.

10.7.3.6 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 This shows the *modal codes* that are active in the interpreter. For instance, *G54* indicates that the *G54 offset* is applied to all coordinates that are entered.

10.7.3.7 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.

10.7.3.8 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.

10.7.3.9 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.

Tabelle 10.5: Häufigste Tastaturkürzel

Tastenkombination	Ergriffene Maßnahmen
F1	Notaus ein-/ausschalten
F2	Maschine ein-/ausschalten
`, 1 .. 9, 0	Vorschub-Override von 0% bis 100% einstellen

Tabelle 10.5: (continued)

Tastenkombination	Ergriffene Maßnahmen
X, `	Erste Achse aktivieren
Y, 1	Zweite Achse aktivieren
Z, 2	Dritte Achse aktivieren
A, 3	Vierte Achse aktivieren
Pos1	Send active axis Home
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 Preamble

Except where noted, this guide assumes the user is using the latest version of QtPlasmaC. Version history can be seen by visiting this link which will show the latest available version. The installed QtPlasmaC version is displayed in the title bar. See Update QtPlasmaC for information on updating QtPlasmaC.

10.8.2 License

QtPlasmaC and all of its related software are released under GPLv2.

10.8.3 Einführung

QtPlasmaC is a GUI for plasma cutting which utilises the plasmac component for controlling a plasma table from LinuxCNC v2.9 or later using the Debian Buster or similar distribution.

The QtPlasmaC GUI supports up to five axes and uses the QtVCP infrastructure provided with LinuxCNC.

The standard theme is based on a design by user ”pinder” on the LinuxCNC Forum and the colors are able to be changed by the user.

The QtPlasmaC GUI will run on any hardware that is supported by LinuxCNC provided there are enough hardware I/O pins to fulfill the requirements of a plasma configuration.

There are three available formats:

- 16:9 with a minimum resolution of 1366 x 768
- 9:16 with a minimum resolution of 768 x 1366
- 4:3 with a minimum resolution of 1024 x 768

Screenshot examples of QtPlasmaC are below:

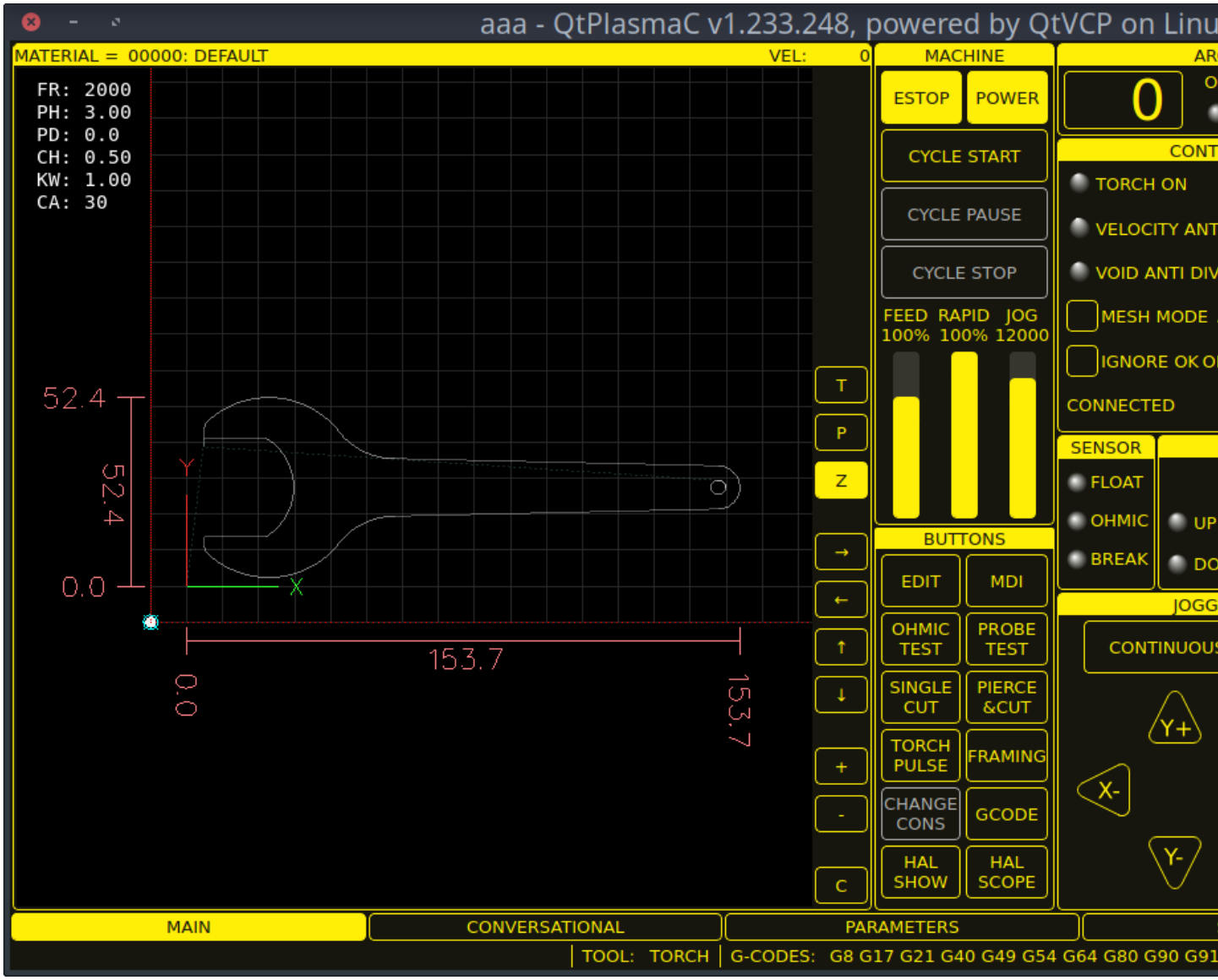
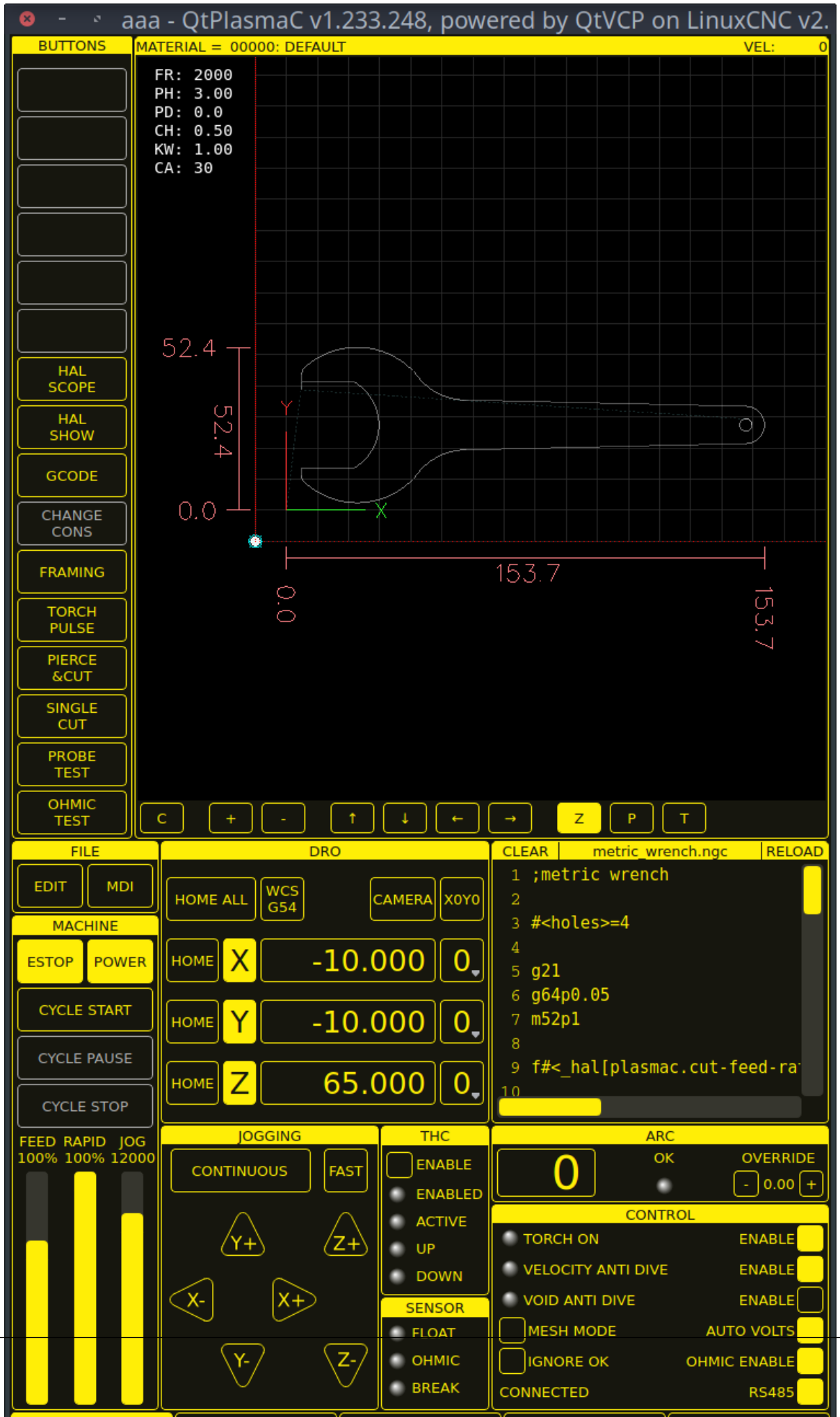


Abbildung 10.44: **16:9**


aaa - QtPlasmaC v1.233.248, powered by QtVCP on LinuxCNC v2.
BUTTONS
MATERIAL = 00000: DEFAULT
VEL: 0
FR: 2000
PH: 3.00
PD: 0.0
CH: 0.50
KW: 1.00
CA: 30
52.4
52.4
0.0
153.7
0.0
153.7
HAL SCOPE
HAL SHOW
GCODE
CHANGE CONS
FRAMING
TORCH PULSE
PIERCE &CUT
SINGLE CUT
PROBE TEST
OHMIC TEST
C
+
-
Z
P
T
FILE
EDIT
MDI
MACHINE
ESTOP
POWER
CYCLE START
CYCLE PAUSE
CYCLE STOP
DRO
HOME ALL
WCS G54
CAMERA
X0Y0
HOME
X
-10.000
0
HOME
Y
-10.000
0
HOME
Z
65.000
0
CLEAR
metric_wrench.ngc
RELOAD
1 ;metric wrench
2
3 #<holes>=4
4
5 g21
6 g64p0.05
7 m52p1
8
9 f#<_hal[plasmac.cut-feed-ra
10
FEED 100%
RAPID 100%
JOG 12000
JOGGING
CONTINUOUS
FAST
Y+
Z+
X-
X+
Y-
Z-
THC
ENABLE
ENABLED
ACTIVE
UP
DOWN
SENSOR
FLOAT
OHMIC
BREAK
ARC
0
OK
OVERRIDE
-
0.00
+
CONTROL
TORCH ON
ENABLE
VELOCITY ANTI DIVE
ENABLE
VOID ANTI DIVE
ENABLE
MESH MODE
AUTO VOLTS
IGNORE OK
OHMIC ENABLE
CONNECTED
RS485

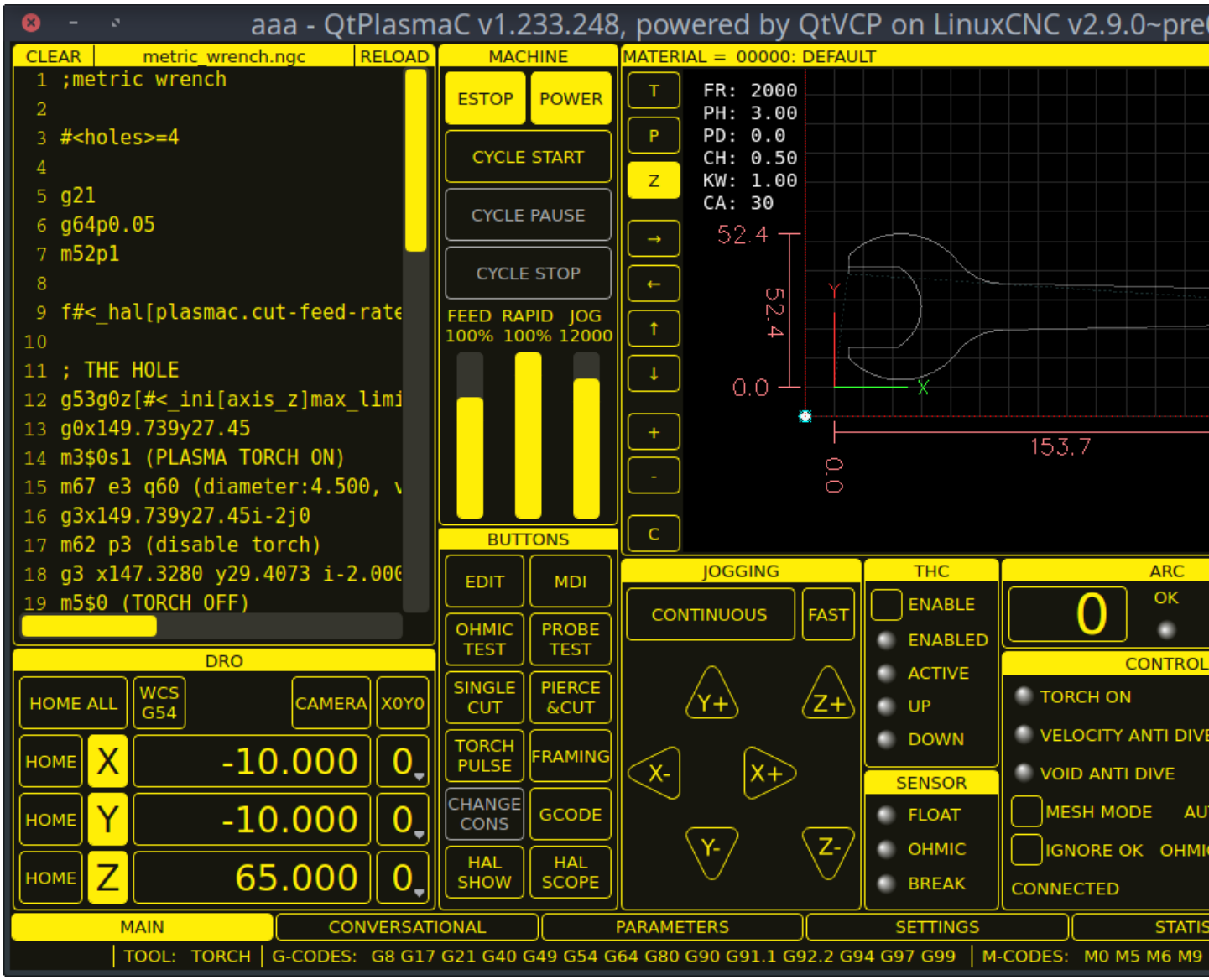


Abbildung 10.46: **4:3**

10.8.4 LinuxCNC installieren

Die bevorzugte Methode zur Installation von LinuxCNC ist über ein ISO-Image, wie unten beschrieben.

Anmerkung
Es ist möglich, LinuxCNC auf einer Vielzahl von Linux-Distributionen zu installieren und auszuführen, was jedoch den Rahmen dieses Benutzerhandbuchs sprengen würde. Wenn der Benutzer möchte eine Linux-Distribution andere als die empfohlenen zu installieren, müssen sie zunächst ihre bevorzugte Linux-Distribution zu installieren und dann installieren LinuxCNC v2.9 oder höher zusammen mit allen erforderlichen Abhängigkeiten.

10.8.4.1 Wenn der Benutzer kein Linux installiert hat

Installationsanweisungen finden Sie unter: link:../getting-started/getting-linuxcnc.html

Die Befolgung dieser Anweisungen wird eine Maschine mit dem aktuellen stabilen Zweig (v2.8) von LinuxCNC auf Debian Buster ergeben.

10.8.4.2 Paket-Installation (Buildbot) Wenn der Benutzer hat Linux mit LinuxCNC v2.8

Eine Paketinstallation (Buildbot) verwendet vorgefertigte Pakete aus dem LinuxCNC Buildbot, Anweisungen für ein Upgrade von 2.8 auf 2.9 sind verfügbar unter: http://buildbot.linuxcnc.org

Befolgen Sie diese Anweisungen unter Verwendung der folgenden Zeilen, um die Maschine auf den letzten LinuxCNC Buildbot-Build-Master-Zweig (v2.9) von LinuxCNC zu aktualisieren. Dies ist möglicherweise nicht immer die neueste Version von Master Branch (v2.9), da der LinuxCNC Buildbot von Zeit zu Zeit aufgrund von Fehlern anhalten kann.

```
deb     http://buildbot.linuxcnc.org/ buster master-rtpreempt
deb-src http://buildbot.linuxcnc.org/ buster master-rtpreempt
```

10.8.4.3 Run In Place Installation, wenn der Benutzer bereits Linux hat mit LinuxCNC v2.8

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 verfügbar unter: link:../code/building-linuxcnc.html

Wenn Sie diese Anweisungen befolgen, installieren Sie den neuesten Master-Zweig (v2.9) von LinuxCNC.

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.

10.8.5.1 Modi

QtPlasmaC erfordert die Auswahl eines der folgenden drei Betriebsmodi:

Mode	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.

Wichtig
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.

Anmerkung
Für die Feinabstimmung von Mode 0 Ark OK siehe Tuning Mode 0 Arc OK im Abschnitt Erweiterte Themen des Handbuchs.

10.8.5.2 Verfügbare I/Os

Anmerkung
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
Arc Voltage	0, 1	Analog input; **optional.** HAL pin name plasmac.arc-voltage-in Connected to the velocity output of an encoder equipped breakout board. This signal is used to read the arc voltage to determine the necessary corrections to maintain the torch distance from the work piece during cutting.
Bogen OK	1, 2	Digital input; **optional.** HAL pin name plasmac.arc-ok-in Connected from the Arc OK output of the plasma power source to an input on the breakout board. This signal is used to determine if the cutting arc has been established and it is ok for the machine to move (sometimes called arc transfer).
Float Switch	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name plasmac.float-switch Connected from a breakout board input to a switch on the floating head. This signal is used to mechanically probe the work piece with the torch and set Z zero at the top of the work piece. If used and no ohmic probe is configured, this is the probing method. If used and an ohmic probe is configured, this is the fallback probing method.
Ohmic Probe	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name `plasmac.ohmic-probe` Connected from to the ohmic probe's output to a breakout board input. This signal is used to probe electronically by completing a circuit using the work piece and the torch consumables and set Z zero at the top of the work piece. If used, this is the primary probing method. If an ohmic probe fails to locate the work piece, and there is no float switch is present, probing will continue until the torch breaks away or the minimum Z limit is reached.

Name	Modi	Beschreibung
Ohmic Probe Enable	0, 1, 2	Digital output; **optional, see info below table:** HAL pin name `plasmac.ohmic-enable` Connected from a breakout board output to an input to control the ohmic probe's power.
Breakaway Switch	0, 1, 2	Digital input; **optional, see info below table:** HAL pin name plasmac.breakaway Connected from a breakout board input to a torch breakaway detection switch. This signal senses if the torch has broken away from its cradle.
Brenner ein (engl. torch on)	0, 1, 2	Digital output; **required.** HAL pin name plasmac.torch-on Connected from a breakout board output to the torch-on input of the plasma power supply. This signal is used to control the plasma power supply and start the arc.
Move Up	2	Digital input; **optional.** HAL pin name plasmac.move-up Connected from the up output of the external THC control to a break out board input. This signal is used to control the Z axis in an upward motion and make necessary corrections to maintain the torch distance from the work piece during cutting.
Move Down	2	Digital input; **optional.** HAL pin name plasmac.move-down Connected from the down output of the external THC control to a break out board input. This signal is used to control the Z axis in a downward motion and make necessary corrections to maintain the torch distance from the work piece during cutting.
Scribe Arming	0, 1, 2	Digital output; **optional.** HAL pin name plasmac.scribe-arm Connected from a breakout board output to the scribe arming circuit. This signal is used to place the scribe into position on the work piece .
Scribe On	0, 1, 2	Digital output; **optional.** HAL pin name plasmac.scribe-on Connected from a breakout board output to the scribe-on circuit. This signal is used to turn the scribing device on.
Laser On	0, 1, 2	Digital output; **optional.** HAL pin name qtplasmac.laser_on This signal is used to turn the alignment laser on.

Only one of either **Float Switch** or **Ohmic Probe** is required. If both are used then **Float Switch** will be a fallback if **Ohmic Probe** is not sensed.

If **Ohmic Probe** is used then **Ohmic Probe Enable** is required to be checked on the QtPlasmaC GUI.

Breakaway Switch is not mandatory because the **Float Switch** is treated the same as a breakaway when not probing. If they are two separate switches, and there are not enough inputs on the breakout board, they could be combined and connected as a **Float Switch**.

Anmerkung
The minimum I/O requirement for a QtPlasmaC configuration to function are: **Arc Voltage** input OR **Arc OK** input, **Float Switch** input, and **Torch On** output. To reiterate, in this case QtPlasmaC will treat the float switch as a breakaway switch when it is not probing.

10.8.5.3 Recommended Settings:

Refer to the Heights Diagram diagram for a visual representation of the terms below.

- **[AXIS_Z] MIN_LIMIT** should be just below top of the slats with allowances for float_switch_travel and over travel tolerance. For example, if the user's float switch takes 4mm (0.157") to activate then set the Z minimum to 5mm (0.2") plus an allowance for overrun (either calculated using the equation below or allow 5mm (0.2") below the lowest slat).
- **[AXIS_Z] MAX_LIMIT** should be the highest the user wants the Z axis to travel (it must not be lower than Z HOME_OFFSET).
- **[AXIS_Z] HOME** should be set to be approximately 5mm-10mm (0.2"-0.4") below the maximum limit.
- **Floating Head** - it is recommended that a floating head be used and that it has enough movement to allow for overrun during probing. Overrun can be calculated using the following formula:

```
o = 0.5 * a * (v / a)^2
```

where: o = overrun, a = acceleration in units/sec2 and v = velocity in units/sec.

Metric example: given a Z axis MAX_ACCELERATION of 600 mm/s^2 and MAX_VELOCITY of 60 mm/s, the overrun would be 3 mm.

Imperial example: given a Z axis MAX_ACCELERATION of 24 in/s^2 and MAX_VELOCITY of 2.4 in/s, the overrun would be 0.12 in.

On machines that will utilize an ohmic probe as the primary method of probing, it is highly recommended to install a switch on the floating head as a backup means of stopping Z motion in the event of ohmic probe failure due to dirty surfaces.

10.8.5.4 Configuring

LinuxCNC provides two configuration wizards which can be used to build a machine configuration. The choice of these wizards is dependent on the hardware used to control the machine.

If the user wishes to use a Run In Place installation then prior to running one of the following commands they will need to run the following command from a terminal:

```
source ~/linuxcnc-dev/scripts/rip-environment
```

If using a Package installation then no additional action is required.

If using a parallel port, use the StepConf wizard (enter the following command into a terminal window):

```
stepconf
```

If using a Mesa Electronics board, use the PnCconf wizard (enter the following command into a terminal window):

```
pncconf
```

If using a Pico Systems board:
This LinuxCNC forum thread may be helpful.

The machine specific settings are not described here, refer to the documentation for the particular configuration wizard that is being used.

There are LinuxCNC forum sections available for these wizards:

StepConf Wizard

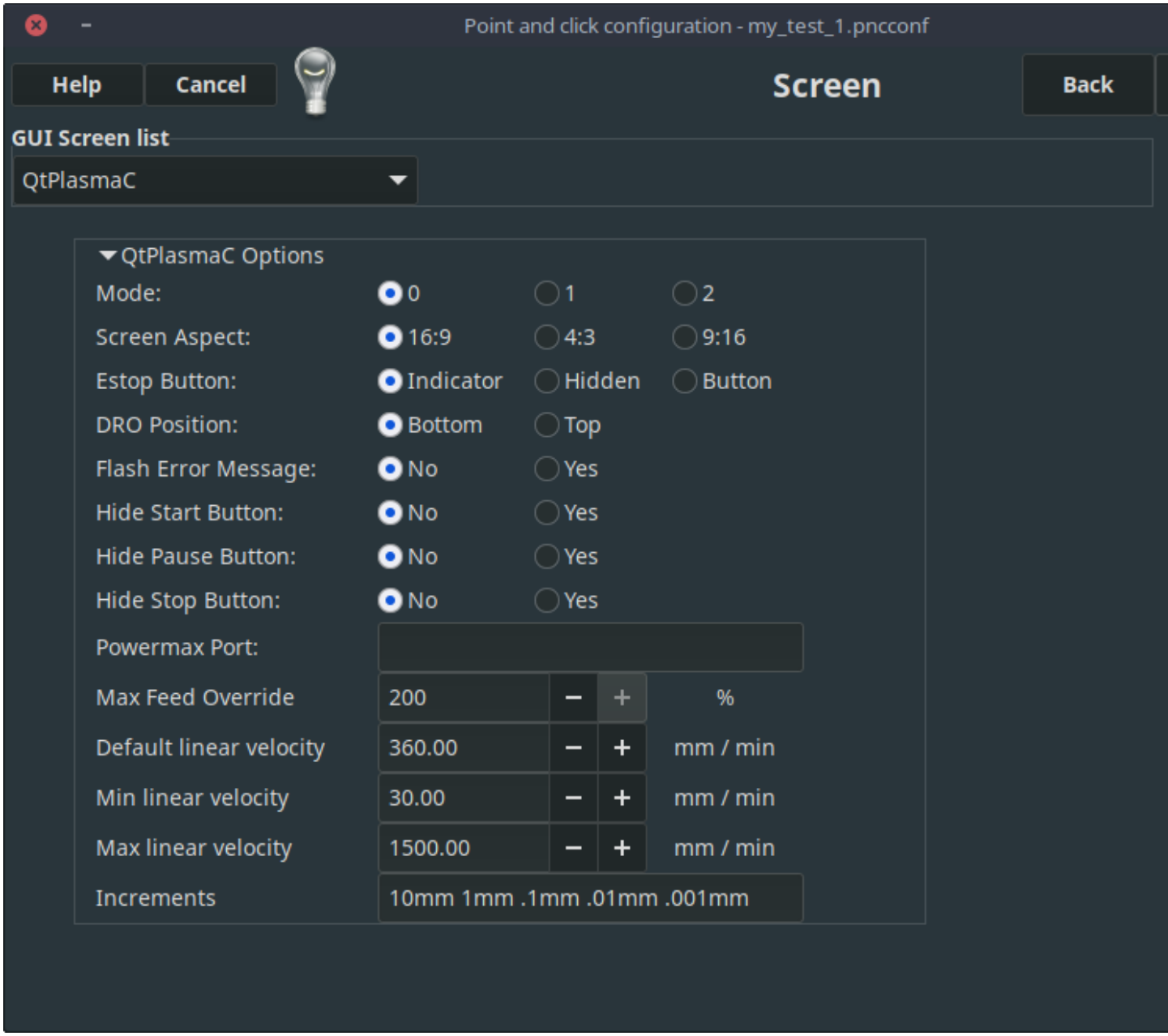
PnCconf Wizard

Füllen Sie die erforderlichen Einträge entsprechend der Konfiguration der Maschinenverdrahtung/Breakout Platine aus.

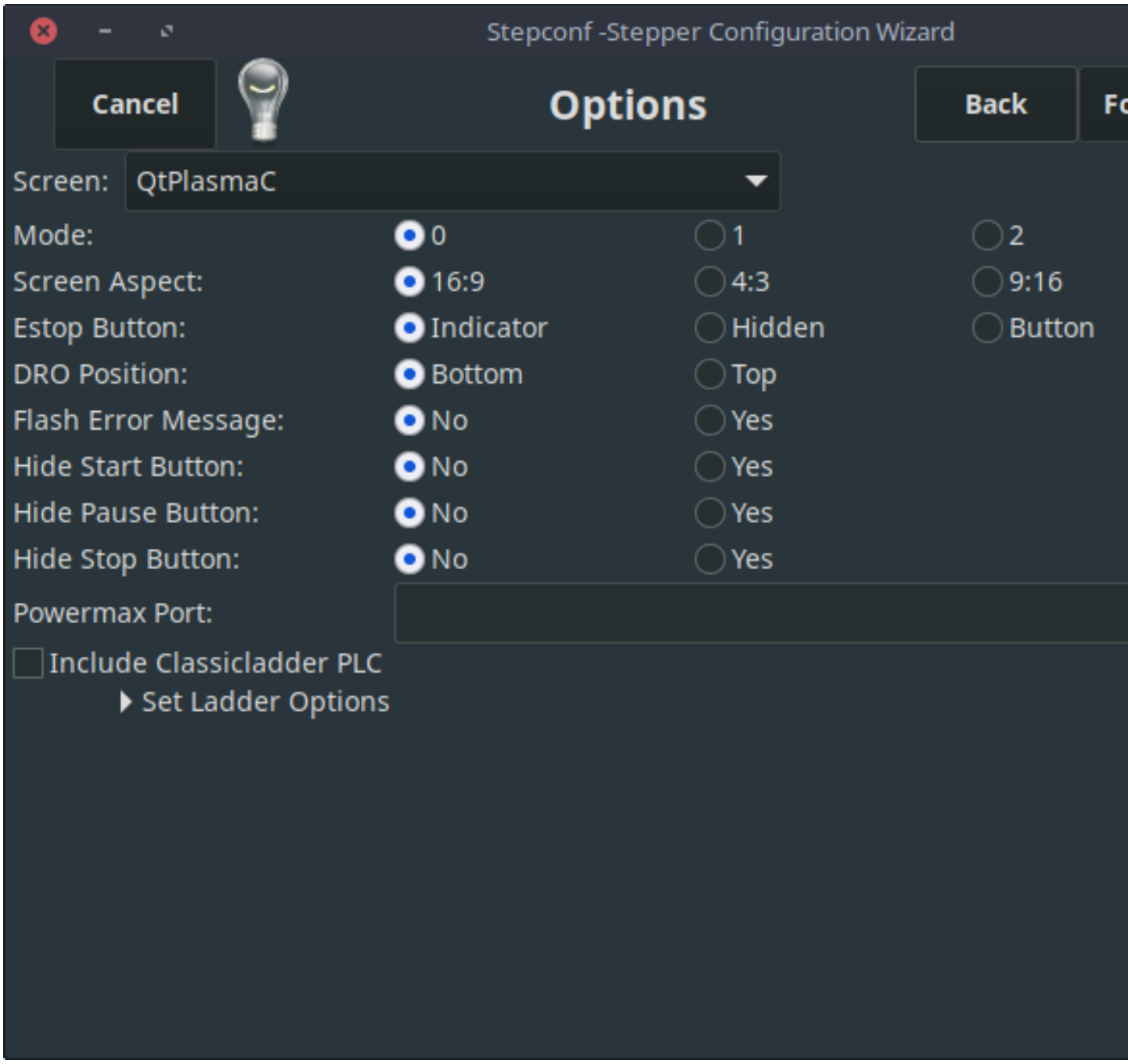
QtPlasmaC fügt den LinuxCNC-Konfigurationsassistenten zwei Seiten für QtPlasmaC-spezifische Parameter hinzu, die beiden Seiten sind QtPlasmaC-Optionen und User Buttons. Füllen Sie jede der Assistenten QtPlasmaC Seite, um die Maschine, die konfiguriert wird und die Benutzer-Button-Anforderungen anzupassen.

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.

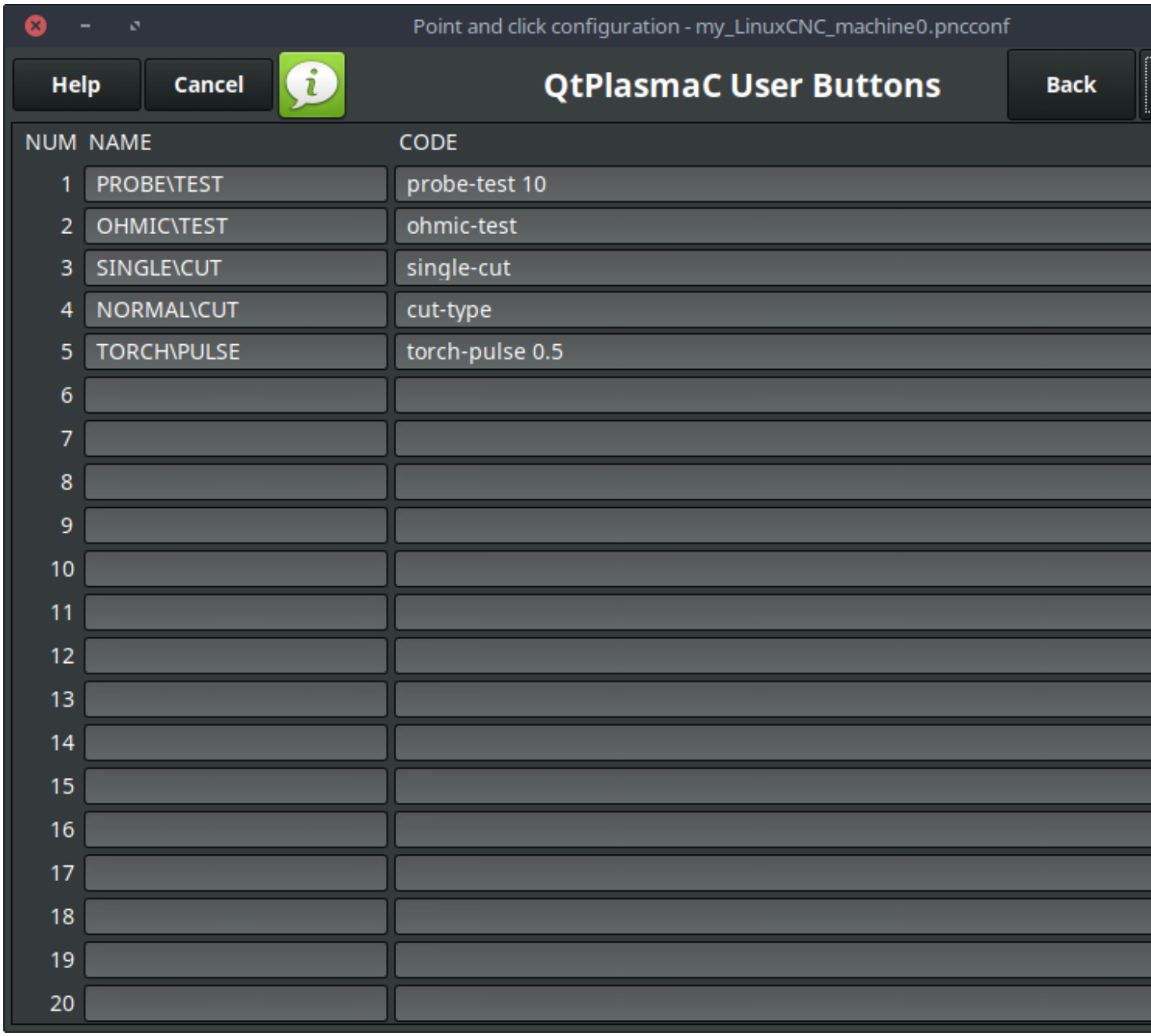
PnCconf QtPlasmaC Options:



StepConf QtPlasmaC-Optionen:

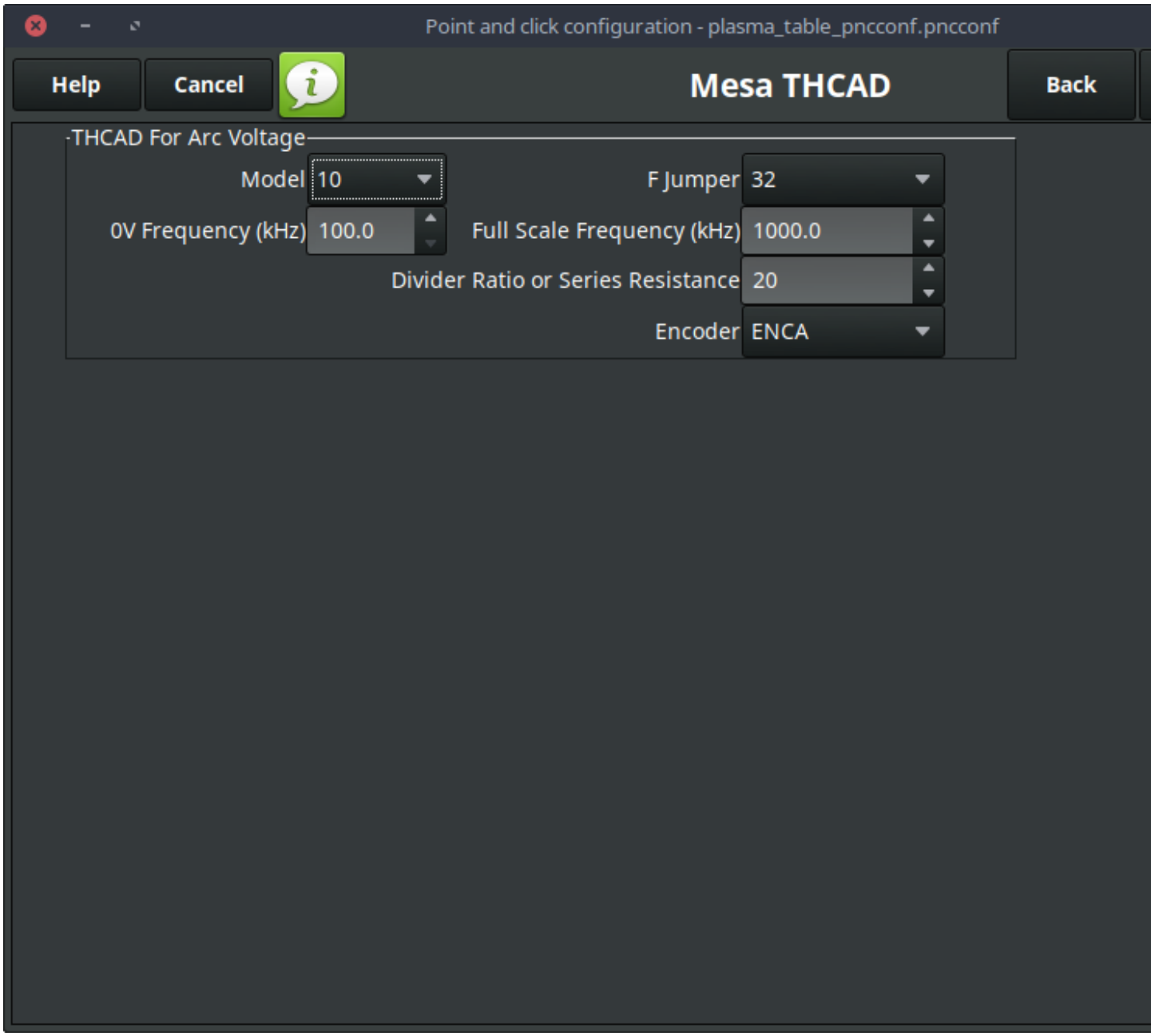


QtPlasmaC Benutzer-Buttons:



QtPlasmaC THCAD:

Der THCAD-Bildschirm wird nur angezeigt, wenn im Kartenbildschirm ein Plasma-Encoder ausgewählt wurde.



Weitere Informationen unter Mesa THCAD.

When the configuration is complete, the wizard will save a copy of the configuration that may be loaded and edited at a later time, a working QtPlasmaC configuration will be created in the following directory: ~/linuxcnc/configs/<*machine_name*>.

Die neu erstellte QtPlasmaC-Konfiguration kann durch Eingabe des folgenden Befehls in einem Terminalfenster ausgeführt werden (**ändern Sie "<Maschinenname>" in den im Konfigurationsassistenten eingegebenen Maschinennamen**):

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.

Wichtig
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.

Anmerkung
Es ist möglich, eine Simulationskonfiguration mit StepConf zu erstellen, aber es ist nicht möglich, Tandemgelenke in der Simulationskonfiguration zu haben.

10.8.5.5 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:

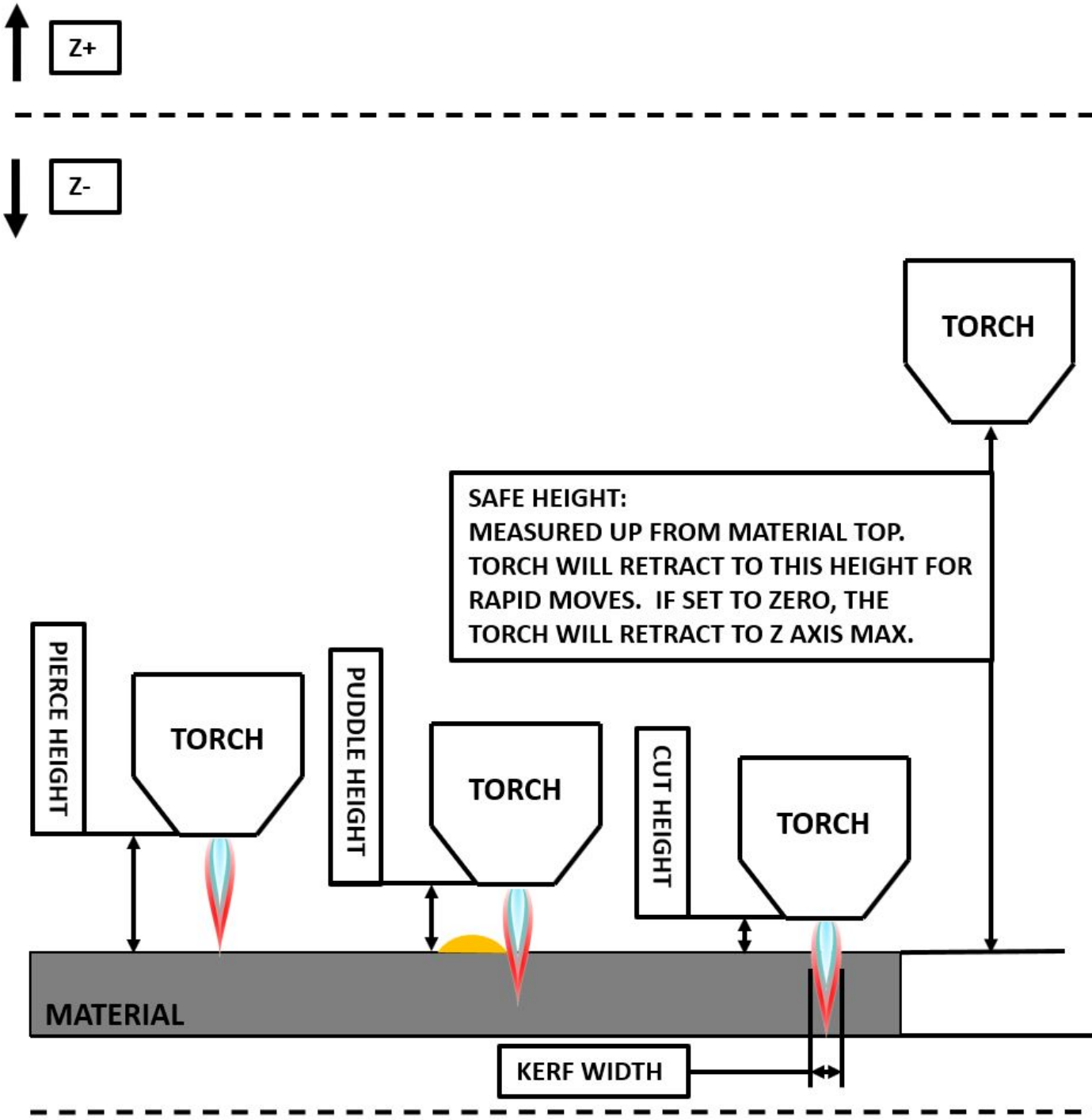
```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

Geben Sie für eine ”run in place”-Installation den folgenden Befehl in ein Terminalfenster ein:

```
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
```

10.8.5.6 Erstmalige Einrichtung

Das folgende Höhendiagramm soll dem Benutzer helfen, die verschiedenen Höhen beim Plasmaschneiden und deren Messung zu veranschaulichen:



Klicken Sie auf die Registerkarte 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.

1. Der Benutzer sollte mit den empfohlenen Z-Achsen Einstellungen vertraut sein.
2. Referenzierung der Z-Achse.
3. 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.
4. Erneute Referenzierfahrt der Z-Achse.

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. Check that the Probe Speed and the Probe Height in the **CONFIGURATION** section of the **PARAMETERS** tab are correct. QtPlasmaC can probe at the full Z axis velocity so long as the machine has enough movement in the float switch to absorb any overrun. If the machine is suitable, the user could set the Probe Height to a value near the Z axis minimum and do all probing at full speed.
2. If the machine is not already homed and in the home position, home the machine.
3. Place some material on the slats under the torch.
4. Press the **PROBE TEST** button.
5. The Z axis will probe down, find the material then move up to the specified **Pierce Height** as set by the currently selected material. The torch will wait in this position for the time set in the *<machine_name>*.prefs file. The default probe test hold time is 10 seconds, this value may be edited in the *<machine_name>*.prefs file. After this the torch will return to the starting height.
6. Measure the distance between the material and the tip of the torch while the torch is waiting at **Pierce Height**.
7. If the measurement is greater than the **Pierce Height** of the currently selected material, then reduce the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the measured value and the specified value. If the measurement is less than **Pierce Height** of the currently selected material, then increase the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the specified value and the measured value.
8. After the adjustments to the ”Float Travel” have been made, repeat the process from #4 above until the measured distance between the material and the torch tip matches the **Pierce Height** of the currently selected material.
9. If the table has a laser or camera for sheet alignment, a scribe, or uses offset probing then the required offsets need to be applied by following the procedure described in Peripheral Offsets.
10. CONGRATULATIONS! The user should now have a working QtPlasmaC Configuration.

Anmerkung
If the amount of time between the torch contacting the material and when the torch moves up and comes to rest at the Pierce Height seems excessive, see the probing section for a possible solution.

Wichtig
IF USING A **Mesa Electronics THCAD** THEN THE **Voltage Scale** VALUE WAS OBTAINED MATHEMATICALLY. IF THE USER INTENDS TO USE CUT VOLTAGES FROM A MANUFACTURE'S CUT CHART THEN IT WOULD BE ADVISABLE TO DO MEASUREMENTS OF ACTUAL VOLTAGES AND FINE TUNE THE **Voltage Scale** AND **Voltage Offset**.

Achtung
PLASMA CUTTING VOLTAGES CAN BE LETHAL, IF THE USER IS NOT EXPERIENCED IN DOING THESE MEASUREMENTS GET SOME QUALIFIED HELP.

10.8.6 Migrating to QtPlasmac From PlasmaC (Axis or Gmoccapy)

There are two methods available to get from a working PlasmaC configuration to a new QtPlasmaC configuration. These methods assume the user is on LinuxCNC v2.9 or later, QtVCP is installed, and all dependency requirements are satisfied.

If there are Qt dependency errors, the user should run the QtVCP install script.

10.8.6.1 Quick Method

A quick method to move to QtPlasmaC from PlasmaC (loaded on top of either Axis or Gmoccapy) is to use the plasmac2qt conversion program which will attempt to create a new QtPlasmaC configuration from an existing PlasmaC INI file. This program will convert the user's parameters, settings, and materials from the previous PlasmaC configuration and create a new QtPlasmaC configuration directory in the ~/linuxcnc/configs directory.

This methods will keep the original PlasmaC config as a backup with _plasmac and a time stamp appended to the directory name.

To run the plasmac2qt conversion program, use the following instructions:

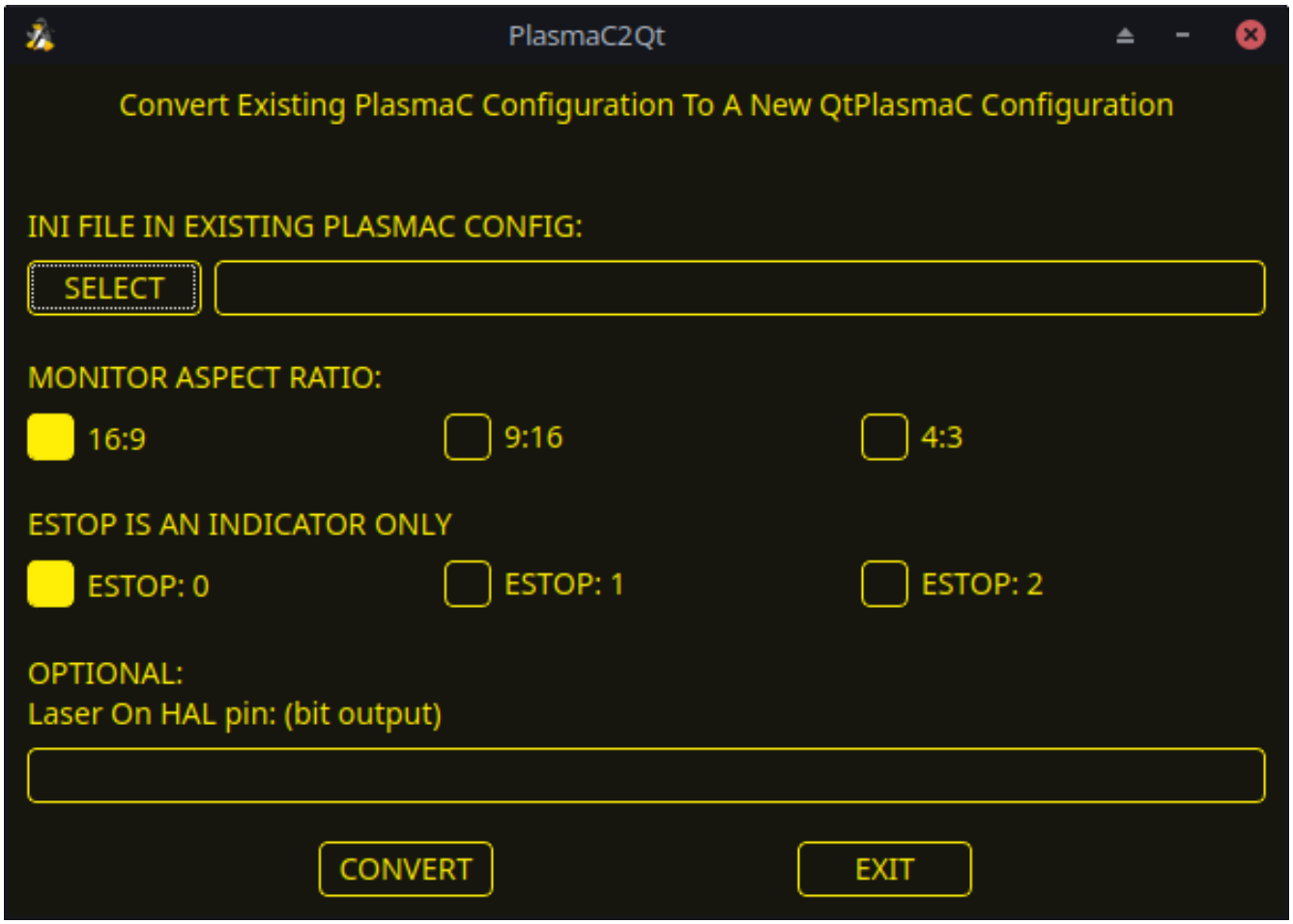
For a package installation (Buildbot) enter the following line in a terminal window:

```
qtplasmac-plasmac2qt
```

For a run in place installation enter the following lines in terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-plasmac2qt
```

The following screen will be displayed:



Mandatory Settings

Field	Beschreibung	Beispiele
INI-DATEI IN VORHANDENER PLASMAC-KONFIGURATION	This is the INI file of the PlasmaC config that requires migrating.	<*machine_name*>.ini
BILDSCHIRM-SEITENVERHÄLTNIS	Dies ist das Seitenverhältnisformat für die GUI.	16:9
ESTOP (engl. für Notaus)	Wählt den gewünschten Estop-Typ anhand der folgenden Kriterien aus: 0 - Estop ist nur ein Indikator. 1 - Estop-Indikator ist ausgeblendet. 2 - Estop ist eine Taste.	ESTOP:1

Optionale Einstellung

This setting is not required unless the machine has a laser for sheet alignment.

Lassen Sie dieses Feld leer, wenn es nicht verwendet/erforderlich ist.

Field	Beschreibung	Beispiele
Laser On HAL-Pins	Schalten Sie ein Laserfadenkreuz für die Bogenausrichtung ein.	**Parallel Port Beispiel:** parport.0.pin-16-out **Mesa 7i96 Beispiel:** hm2_7i96.0.ssr.00.out-00

Nachdem Sie die entsprechenden Eingaben gemacht haben, drücken Sie **CONVERT**.

Anmerkung
Mit dieser Methode werden keine bestehenden Debounce-Komponenten auf die neue dbounce-Komponente umgestellt. Wenn der Benutzer auf die neue dbounce-Komponente wechseln möchte, sollte die Methode New Base Config für die Migration verwendet werden.

10.8.6.2 Neue Basis-Konfigurationsmethode

This method to move to QtPlasmaC from PlasmaC (loaded on top of either Axis or Gmoccapy) is to use a configuration wizard to create a new configuration. This method then allows changing of the base machine configuration at a later date via the configuration wizard provided that the base INI and base HAL files have not been edited.

Bei dieser Methode muss der Benutzer alle HAL-Pins notieren, die in der vorhandenen Konfiguration verwendet werden, damit sie in den Konfigurationsassistenten eingegeben werden können. Alle benutzerdefinierten HAL-Befehle müssen ebenfalls notiert und entweder der Datei custom.hal oder der Datei custom_postgui.hal, die vom Konfigurationsassistenten erstellt wird, manuell hinzugefügt werden.

Nach der Verwendung des Assistenten kann der Benutzer dann ein Konvertierungsprogramm (cfg2prefs) ausführen, um die Parameter, Einstellungen und Materialien der vorherigen PlasmaC-Konfiguration in die neue QtPlasmaC-Konfiguration zu konvertieren. Dieses Werkzeug sollte unmittelbar nach der Erstellung einer neuen QtPlasmaC-Konfiguration verwendet werden.

Vor dem Ausführen dieses Konvertierungsprogramms ist es zwingend erforderlich, dass der Benutzer sowohl eine bestehende PlasmaC-Konfiguration als auch eine neue QtPlasmaC-Konfiguration besitzt. Dieses Programm überschreibt die vorhandenen QtPlasmaC-Voreinstellungen und Materialdateien und sollte mit Vorsicht verwendet werden, wenn es nicht auf einer neuen QtPlasmaC-Konfiguration ausgeführt wird.

Das Programm erstellt eine mit einem Zeitstempel versehene Sicherungskopie der ursprünglichen Einstellungsdatei und der vorhandenen Materialdatei (falls vorhanden).

It will read the existing <machine_name>_config.cfg, <machine_name>_run.cfg, <machine_name>_wizard and plasmac_stats.var files and write them to an existing <*machine_name*>.prefs file. It will also copy the <machine_name>_material.cfg file to the existing QtPlasmaC configuration.

Um das Konvertierungsprogramm cfg2prefs auszuführen, folgen Sie den folgenden Anweisungen:

For a package installation (Buildbot) enter the following line in a terminal window:

```
qtplasmac-cfg2prefs
```

For a run in place installation enter the following lines in terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-cfg2prefs
```

Select the INI file of the old PlasmaC configuration, select the INI file of the new QtPlasmaC configuration, then press **CONVERT**.

10.8.7 Other QtPlasmaC Setup Considerations

10.8.7.1 Lowpass Filter

The plasmac HAL component has a built in lowpass filter that if used is applied to the **plasmac.arc-voltage-in** input pin to filter any noise that could cause erroneous voltage readings. The lowpass filter should only be used after using Halscope to determine the required frequency and whether the amplitude of the noise is large enough to cause any issues. For most plasma machines lowpass is not required and should not be used unless it is required.

The HAL pin assigned to this filter is **plasmac.lowpass-frequency** and is set to 0 (disabled) by default. To apply a lowpass filter to the arc-voltage, the user would edit the following entry in the custom.hal file in the machine's configuration directory to add the appropriate cutoff frequency as measured in Hertz (Hz).

Zum Beispiel:

```
setp plasmac.lowpass-frequency 100
```

The above example would give a cutoff frequency of 100Hz.

10.8.7.2 Contact Bounce

Contact bounce from mechanical relays, switches, or external interference may cause some inconsistent behavior of the following switches:

- Float Switch
- Ohmic Probe
- Breakaway Switch
- Arc OK (for modes 1 & 2)

Due to the fact that the software is capable of sampling rates faster than the contact bounce period, it is possible that the software may see contact bounce as several changes in input states occurring in a very small time period, and incorrectly interpret this as a very quick on-off of the input. One method of mitigating contact bounce is to ”debounce” the input. To summarize debounce, it requires the input state to be stable at the opposite state of the output state for consecutive delay periods before changing the state of the output.

Debounce delay periods can be changed by editing the appropriate debounce value in the custom.hal file in the <machine_name> config directory.

Each increment of delay adds one servo thread cycle to the debounce time. For example: given a servo thread period of 1000000 (measured in nano seconds), a debounce delay of 5 would equate to 5000000ns, or 5ms.

For the Float and Ohmic switches this equates to a 0.001mm (0.00004”) increase in the probed height result.

It is recommended to keep the debounce values as low as possible while still achieving consistent results. Using Halscope to plot the inputs is a good way to establish the correct value.

For QtPlasmaC installations, debounce is achieved by using the HAL dbounce component which is a later alternative to the original debounce component. This new version allows for the loading and naming of individual debounce instances and is compatible with Twopass HAL file processing.

All four signals above have an individual debounce component so the debounce periods can be catered individually to each input. Any changes made to these values in the custom.hal file will not be overwritten by later updates of QtPlasmaC.

The default delay for all four inputs is five servo thread periods. In most cases this value will work quite well. If any of the inputs do not use mechanical switches, it may be possible to either reduce or remove the delay for those inputs.

If debounce is required for other equipment like home or limit switches etc. then more dbounce components may added in any of the HAL files without any regard to the signals listed here.

10.8.7.3 Contact Load

Mechanical relays and switches usually require a minimum current passing through the contacts for reliable operation. This current varies with the material that the contacts in the device are made from.

Depending on the specified minimum contact current and the current drawn by the input device there may be a need to provide a method to increase the current through the contacts.

Most relays using gold contacts will not require any additional current for reliable operation.

There are two different methods available to provide this minimum current if it is required:

1. A 0.1µF film capacitor placed across the contacts.
2. A 1200Ω 1W resistor across the load (see calculations below).

Schematics are shown at contact load schematics.

More information on contact switching load can be seen on page III of Finder Relays General Technical Information

Calculations:

If using a Mesa card, the input resistance of a 7i96 is 4700 Ω (always consult the product manual associated with the revision being used as these values sometimes vary between revisions), giving a contact current of 5.1 mA assuming a supply voltage of 24 V (I = V/R).

As an example, the typical relay used in a Hypertherm Powermax 65 plasma cutter (TE T77S1D10-24) requires a minimum contact load of 100 mA @ 5 VDC which will dissipate 0.5 W (P = I * V). If using a 24 VDC power supply this would then equate to a minimum current of 20.8 mA. Because there is less current drawn by the Mesa input than is required by the relay there needs to be an increase in the current.

The resistance can be calculated using $R = U_s / (I_m - I_i)$ where:

- R = berechneter Widerstand
- U_s = supply voltage
- I_m = minimum current required
- I_i = input current

Using a 7i96 with an input current of 5.1 mA gives a calculated value of 1529 Ω (= 24 V / (.0208 - .0051) A). This could then be rounded down to a commonly available 1500 Ω resistor, giving a small safety margin.

The power dissipation can by calculated using $P = U_s^2 / R_s$ where:

- P = Leistung
- U_s = supply voltage
- R_s = selected resistance

This gives a value of 0.38 W. This could then be rounded up to 1 W, giving a good safety margin. The final selection would be a 1500 Ω 1 W resistor.

10.8.7.4 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.

10.8.7.5 QtPlasmaC Dateien

Nach einer erfolgreichen QtPlasmaC-Installation werden die folgenden Dateien im Konfigurationsverzeichnis angelegt:

Filename	Function
<*machine_name*>.ini	A configuration file for the machine.
<*machine_name*>.hal	A HAL for the machine.
<*machine_name*>.prefs	A configuration file for QtPlasmaC specific parameters and preferences.
custom.hal	A HAL file for user customization.
custom_postgui.hal	A HAL file for user customization which is run after the GUI has initialized.
shutdown.hal	A HAL file which is run during the shutdown sequence.
tool.tbl	A tool table used to store offset information for additional tools (scribe, etc.) used by the QtPlasmaC configuration.
qtplasmac	A link to the directory containing common qtplasmac support files.
backup	A directory for backups of config files.

Anmerkung
<machine_name> is whatever name the user entered into the ”Machine Name” field of the configuration wizard program

Anmerkung
Custom commands are allowed in custom.hal and the custom_postgui.hal files as they are not overwritten during updates.

After running a new configuration for the first time the following files will be created in the configuration directory:

Filename	Funktion
<machine_name>_material.cfg	A file for storing the material settings from the MATERIAL section of the PARAMETERS Tab.
qtvcp.prefs	A file containing the QtVCP preferences.
qtplasmac.qss	This file is used to store the stylesheet for the currently loaded session of QtPlasmaC.

Anmerkung
The configuration files (<*machine_name*>.ini and <*machine_name*>.hal) that are created by configuration wizard are notated to explain the requirements to aid in manual manipulation of these configurations. They may be edited with any text editor.

Anmerkung
The <*machine_name*>.prefs file is plain text and may be edited with any text editor.

10.8.7.6 INI File

QtPlasmaC has some specific <*machine_name*>.ini file variables as follows:

[FILTER] Section

These variables are mandatory.

```
PROGRAM_EXTENSION = .ngc,.nc,.tap G-code File (*.ngc, *.nc, *.tap)
ngc               = qtplasmac_gcode
nc                = qtplasmac_gcode
tap               = qtplasmac_gcode
```

[RS274NGC] Section

These variables are mandatory.

```
RS274NGC_STARTUP_CODE = G21 G40 G49 G80 G90 G92.1 G94 G97 M52P1
SUBROUTINE_PATH       = ./:../../nc_files
USER_M_PATH           = ./:../../nc_files
```

Anmerkung
for a imperial config replace G21 above with G20.

Anmerkung
both the above paths show the minimum requirements.

Wichtig
SEE PATH TOLERANCE FOR RS274NGC_STARTUP_CODE INFORMATION RELATED TO G64.

[HAL] Section

These variables are mandatory.

```
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)
```

Anmerkung
The user could place custom HAL commands in the custom.hal file as this file is not overwritten by QtPlasmaC updates.

[DISPLAY] Section

This variable is mandatory.

```
DISPLAY = qtvcp qtplasmac      (use 16:9 resolution)
        = qtvcp qtplasmac_9x16 (use 9:16 resolution)
        = qtvcp qtplasmac_4x3  (use 4:3 resolution)
```

There are multiple QtVCP options that are described here: QtVCP INI Settings

For example the following would start a 16:9 resolution QtPlasmaC screen in full screen mode:

```
DISPLAY = qtvcp -f qtplasmac
```

[TRAJ] Section

This variable is mandatory.

```
SPINDLES = 3
```

[AXIS_X] Section

These variables are mandatory.

```
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Y] Section

These variables are mandatory.

```
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Z] Section

These variables are mandatory.

```
MIN_LIMIT        = just below the top of the table's slats
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

Anmerkung
QtPlasmaC uses the LinuxCNC External Offsets feature for all Z axis motion, and for moving the X and/or Y axis for a consumable change while paused. For more information on this feature, please read External Axis Offsets in the LinuxCNC documentation.

10.8.8 QtPlasmaC GUI Overview

The following sections will give a general overview of the QtPlasmaC layout.

10.8.8.1 Exiting QtPlasmaC

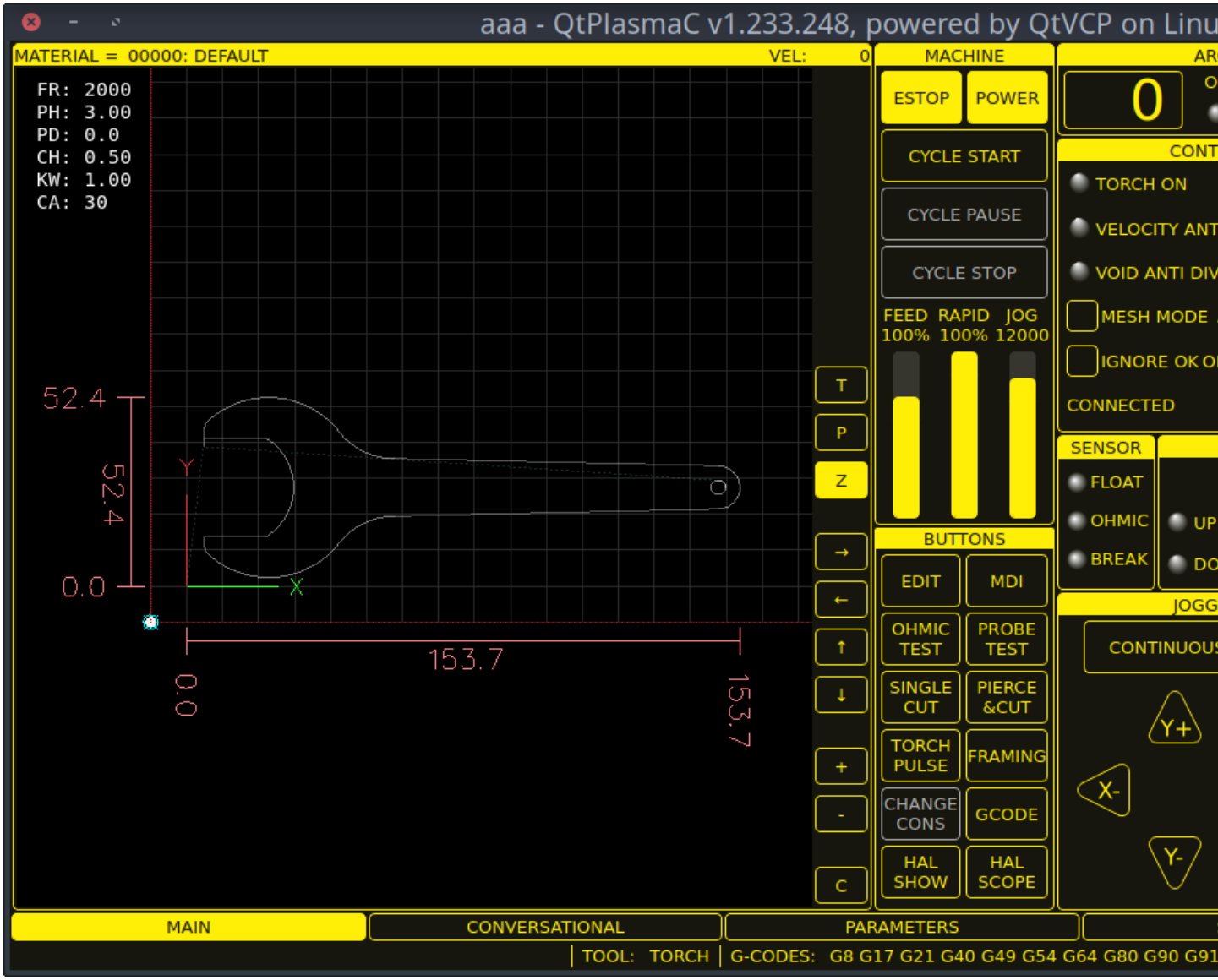
Exiting or shutting down QtPlasmaC is done by either:

1. Click the window shutdown button on the window title bar
2. Long press the **POWER** button on the MAIN Tab.

A shutdown warning can be displayed on every shutdown by checking the **Exit Warning** checkbox on the SETTINGS Tab.

10.8.8.2 HAUPT (engl. main)-Registerkarte (engl. tab)

Screenshot-Beispiel des QtPlasmaC MAIN Tab 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.

VORSCHAU-FENSTER

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.:	Hier wird der tatsächliche Schnittvorschub angezeigt, mit dem sich der Tisch bewegt.
FR:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Feed Rate.

Name	Beschreibung
PH:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Pierce Height.
PD:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Pierce Delay.
CH:	If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Cut Height.
CA:	If ”View Material” is selected on the SETTINGS Tab, and RS485 communications are enabled, this displays the currently selected material’s Cut Amperage.
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.

MASCHINE

Name	Beschreibung
ESTOP (engl. für Notaus)	If ESTOP_TYPE = 0 in the <*machine_name*>.prefs file, this button becomes an indicator of the hardware ESTOP’s status only. If ESTOP_TYPE = 1 in the <*machine_name*>.prefs file, this button will not be visible. If ESTOP_TYPE = 2 in the <*machine_name*>.prefs file, this button will act as a GUI ESTOP. If ESTOP_TYPE is omitted from the <*machine_name*>.prefs file, this button will default to being an indicator of the hardware ESTOP’s status only.
POWER (engl. für Leistung oder Strom)	This button turns the GUI on and allows QtPlasmaC/LinuxCNC to control the hardware. Pressing and holding the **POWER** button for longer than two seconds will bring up a dialog to exit the QtPlasmaC application.
ZYKLUSSTART	Mit dieser Button startet den Zyklus für jede geladene G-Code-Datei.
ZYKLUSPAUSE	This button pauses the cycle for any loaded G-code file. If a cycle is paused, this button will display **CYCLE RESUME** and flash. Pressing **CYCLE RESUME** will resume the cycle.
ZYKLUS STOP (engl. cycle stop)	This button stops any actively running or paused cycle. This includes: - G-code Programs - Torch pulse if the pulse was started during **CYCLE PAUSE** (this will cancel the paused G-code program execution as well) - Probe Test - Framing - Manual Cut
FEED	This slider overrides the feed rate for all feed moves. Any value other than 100% will cause the label to flash. Clicking the label will return the slider to 100%.
RAPID	This slider overrides the rapid rate for all rapid moves. Any value other than 100% will cause the label to flash. Clicking the label will return the slider to 100%.

Name	Beschreibung
JOG	This slider sets the jog rate. Clicking the label will return the slider to the default linear velocity as set in the <machine_name).ini file.

BUTTONS

The Button Panel contains buttons useful for the operation of the machine.

The **EDIT** and **MDI** buttons are permanent, all other buttons are user programmable in the *<machine_name>*.prefs file.

See custom user buttons for detailed information on custom user buttons.

Name	Beschreibung
EDIT	This button opens a G-code editor for the currently loaded program.
MDI	This button places QtPlasmaC into Manual Data Input (MDI) mode which will display the MDI HISTORY and an entry box over top of the G-code window. Once pressed, this button will display ”MDI CLOSE”. Pressing **MDI CLOSE** will close the MDI. Please see the MDI section for additional MDI information.
OHMIC TEST	This button will enable the Ohmic Probe Enable output signal and if the Ohmic Probe input is sensed, the LED indicator in the SENSOR Panel will light. The main purpose of this is to allow a quick test for a shorted torch tip.
PROBE TEST	This button will initiate a Probe Test.
SINGLE CUT	This button will show the dialog box to start an automatic Single Cut.
NORMAL CUT	This button will toggle between Cut Types (NORMAL CUT and PIERCE ONLY).
TORCH PULSE	This button will initiate a Torch Pulse.

ARC

Name	Modi	Beschreibung
Arc Voltage	0, 1	Displays the actual arc voltage.
OK	0, 1, 2	Indicates the status of the Arc OK signal.
+	0, 1	Each press of this button will raise the target voltage by the THC Threshold voltage (The distance changed will be Height Per Volt * THC Threshold voltage).
-	0, 1	Each press of this button will lower the target voltage by the THC Threshold voltage (The distance changed will be Height Per Volt * THC Threshold voltage).
OVERRIDE	0, 1	Clicking this label will return any voltage override to 0.00.

CONTROL

Name	Modi	Beschreibung
TORCH ON	0, 1, 2	Indicates the status of the Torch On output signal.

Name	Modi	Beschreibung
TORCH ON ENABLE	0, 1, 2	This box toggles between Enabling and Disabling the torch. This box defaults to unfilled (disabled) when QtPlasmaC is first run. This box must be filled to change it to ”Torch Enabled” before material cutting can commence. If this box is not filled, then running a loaded program will cause the machine to run the cycle without firing the torch. This is sometimes referred to as a ”dry run”.
VELOCITY ANTI DIVE	0, 1, 2	Indicates that the THC is locked at the current height due to the cut velocity falling below the Velocity Anti Dive (VAD) Threshold percentage set on the PARAMETERS Tab.
VELOCITY ANTI DIVE ENABLE	0, 1, 2	This box toggles between Enabling and Disabling VELOCITY ANTI DIVE.
VOID ANTI DIVE	0, 1	Indicates that the THC is locked due to a void being sensed.
VOID ANTI DIVE ENABLE	0, 1	This box toggles between Enabling and Disabling VOID ANTI DIVE.
MESH MODE	0, 1, 2	This box will enable or disable Mesh Mode for the cutting of expanded metal. This check box may be enabled or disabled at any time during normal cutting. Mesh mode: - Will require an Arc OK signal to start machine motion. - Will disable the THC. - Will not stop machine motion if the Arc OK signal is lost. - Will automatically select CPA mode if PowerMax communications are being used. For more information see Mesh Mode (expanded metal).
AUTO VOLTS	0, 1	Mit diesem Feld wird Auto Volts aktiviert oder deaktiviert.
IGNORE OK	0, 1, 2	This box will determine if QtPlasmaC ignores the Arc OK signal. This check box may be enabled or disabled at any time during normal cutting. Additionally this mode may be enabled or disabled via proper M codes in a running program. Ignore Arc OK mode: - Will not require an Arc OK signal be received before starting machine motion after the ”Torch On” signal is given. - Will disable the THC. - Will not stop machine motion if the Arc OK signal is lost. For more information see Ignore Arc Ok.
OHMIC PROBE	0, 1, 2	This box enables or disables the ohmic probe input. If the Ohmic Probe input is disabled, the Ohmic Probe LED will still show the status of the probe input, but the Ohmic Probe results will be ignored.
RS485	0, 1, 2	This box will enable or disable the communications to a PowerMax. This button is only visible if a PM_PORT is configured in the `[POWERMAX]` section of the *<machine_name>*.prefs file.

Name	Modi	Beschreibung
Status	0, 1, 2	When PowerMax communications are enabled, this will display one of the following: **CONNECTING**, **CONNECTED**, **COMMS ERROR**, or a **Fault Code**. For more information, see the PowerMax Communications section.

SENSOR

Name	Beschreibung
FLOAT	Indicates that the float switch is activated.
OHMIC	Indicates that the probe has sensed the material.
BREAK	Indicates that the torch breakaway sensor is activated.

THC

Name	Beschreibung
ENABLE (AKTIVIEREN)	This box determines whether the THC will be enabled or disabled during a cut.
ENABLED	This LED indicates whether the THC is enabled or disabled.
ACTIVE	This LED indicates that the THC is actively controlling the Z axis.
UP	This LED indicates that the THC is commanding the Z axis to raise.
DOWN	This LED indicates that the THC is commanding the Z axis to lower.

JOGGING

Anmerkung
During Paused Motion, this section will become CUT RECOVERY

Name	Beschreibung
CONTINUOUS	This drop down button will change the jog increment. Options are determined by the values in the **[DISPLAY]** section of the *<machine_name>*.ini file and begin with the label ”INCREMENTS =”.
FAST	This button will toggle between FAST which is the default linear velocity in the *<machine_name>*.ini file or SLOW which is 10% of the default value.
Y+	This button moves the Y axis in the positive direction.
Y-	This button moves the Y axis in the negative direction.
X+	This button moves the X axis in the positive direction.
X-	This button moves the X axis in the negative direction.
Z+	This button moves the Z axis in the positive direction.
Z-	This button moves the Z axis in the negative direction.

CUT RECOVERY

Anmerkung
During Paused Motion, this section will be shown on top of the JOGGING panel. The following section will cover each button encountered in this panel. Please see CUT RECOVERY for a detailed description of the cut recovery functionality.

Name	Beschreibung
PAUSED MOTION FEED SLIDER	In the event of a paused program, this interface allows X/Y motion to follow the programmed path in the reverse or forward direction. This slider's range is from 1%-100% of the Cut Feed Rate for the currently selected material.
FEED	This displays the paused motion feed rate.
REV	In the event of a paused program, this button will move the machine in reverse along the programmed path until it reaches the last M3 command that was either executed or that QtPlasmaC was attempting to execute before the program became paused.
FWD	In the event of a paused program, this button will move the machine forward along the programmed path indefinitely until the program's end, skipping over M3 commands.
CANCEL MOVE	This button will cancel any Cut Recovery movement that was made, and return the torch to the position the Cut Recovery movement was initiated. Note that if FWD or REV were used to move the torch, CANCEL will not return to the position of the torch when the pause occurred.
MOVE x.xxx	This displays the amount of travel that will be incurred with each press of an arrow key, in the direction the arrow key was pressed. This value displayed below MOVE represents the Kerf Width of the currently selected material.
DIRECTIONAL ARROWS	These buttons will move the torch in the direction indicated by a distance of one Kerf Width (of the currently selected material) per press.

G-CODE WINDOW

Name	Beschreibung
CLEAR	This button will clear the currently opened program. The torch (T0) will be selected if it was not the active tool.
OPEN	This button will open a FILE OPEN panel over the PREVIEW WINDOW.
RELOAD	This button will reload the currently loaded G-code File.

DRO

Name	Beschreibung
HOME ALL	This button will home all of the axes in the order set by HOME_SEQUENCE in the <*machine_name*>.ini file.
WCS G54	This drop down button will change the current work offset.
CAMERA	This button will display a CAMVIEW panel on top of the PREVIEW WINDOW and will allow the user to set an origin with or without rotation. See the CAMERA section for detailed instructions.
LASER	This button will allow the user to use a laser to set an origin with or without rotation. See the LASER section for detailed instructions.
X0 Y0	This button will set the current position to X0 Y0.
HOME [AXIS]	This button will home the corresponding axis.
0 [AXIS]	This drop down button will display the following options: **Zero** - zeros the axis. **Set** - launches a dialog box to manually input the axis' coordinate. **Divide By 2** - divides the currently displayed coordinate in the DRO by two. **Set To Last** - sets the axis to the previously set coordinate.

10.8.8.3 Preview Views

The QtPlasmaC preview screen has the ability to be switched between different views and displays, as well as zooming in and out, and panning horizontally and vertically.

When QtPlasmaC is first started, the Z (top down) view will be selected as the default view for a loaded G-code file, but the full table view will be displayed.

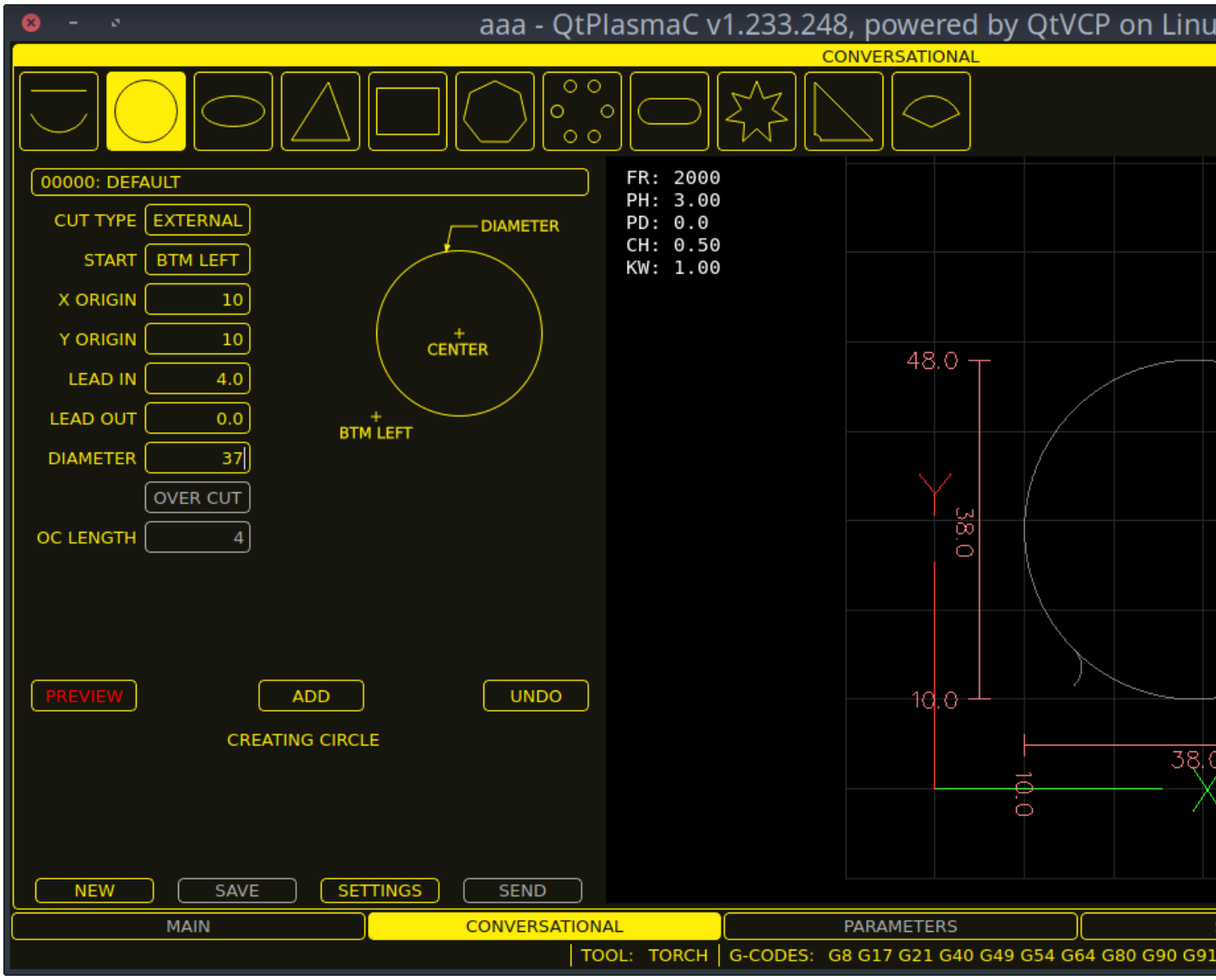
When a G-code file is loaded, the display will change to the selected view.

Whenever there is no G-code file loaded, the full table will automatically be displayed irrespective of which view is currently selected (the highlighted button representing the currently selected view will not change).

If a full table is displayed due to no G-code file being loaded and the user wishes to change the view orientation, then pressing either Z or P will change the display to the newly selected view. If the user then wishes to display the full table while maintaining the currently selected view as the default view for a loaded G-code file, then pressing CLEAR will achieve this and allow the selected view orientation to prevail the next time a G-code file is loaded.

10.8.8.4 CONVERSATIONAL Tab

Screenshot example of the QtPlasmaC CONVERSATIONAL Tab in **16:9** aspect ratio:



The CONVERSATIONAL Tab enables the user to quickly program various simple shapes for quick cutting without the need for CAM software.

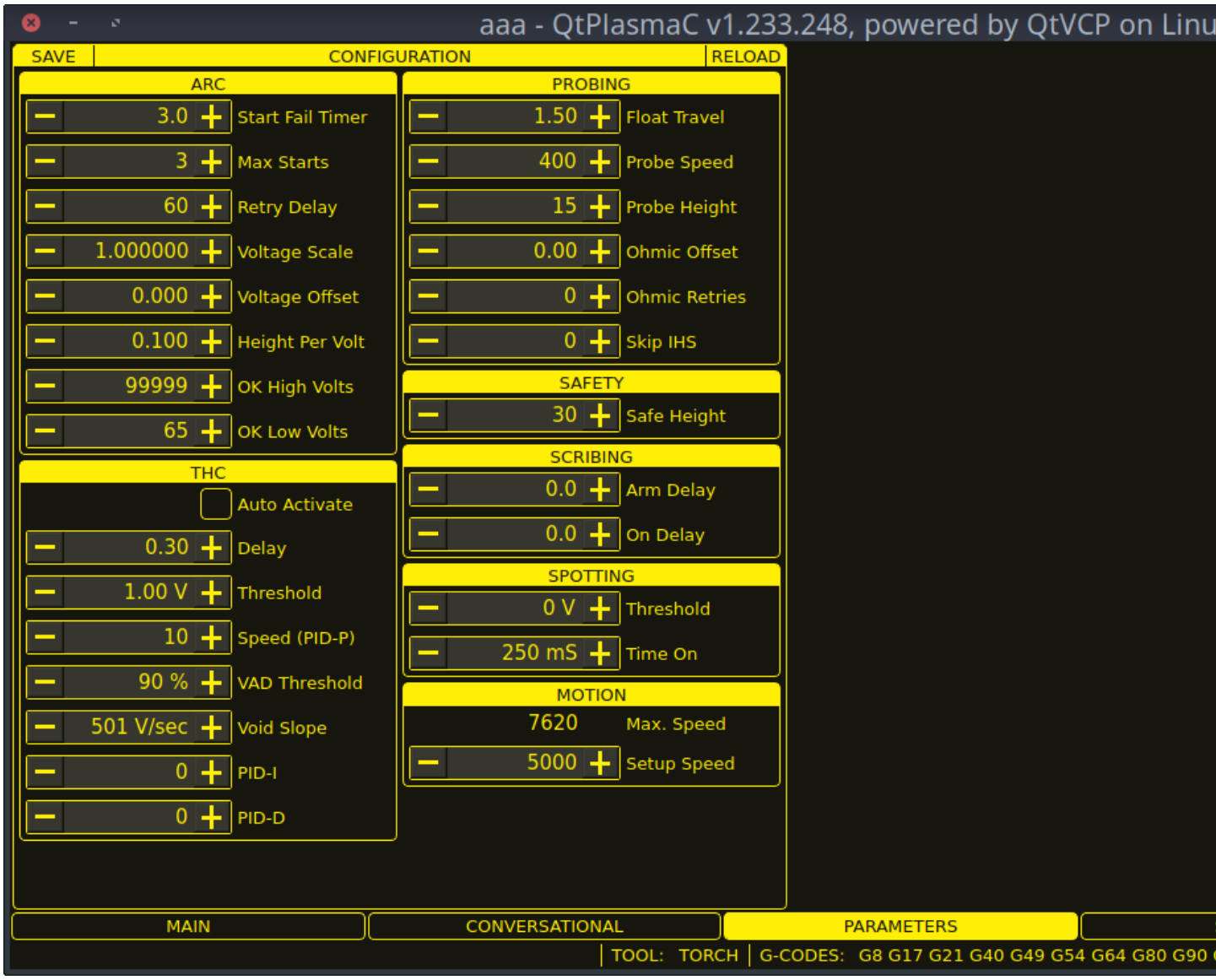
See Conversational Shape Library for detailed information on the Conversational feature.

It is possible to disable this tab so the conversational feature cannot be used by an operator. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.conv_disable 1
```

10.8.8.5 PARAMETERS Tab

Screenshot example of the QtPlasmaC PARAMETERS Tab in **16:9** aspect ratio:



Einige Funktionen/Merkmale werden nur für bestimmte Modi verwendet und werden nicht angezeigt, wenn sie für den gewählten QtPlasmaC-Modus nicht erforderlich sind.

This tab is used to display plasma configuration parameters that are modified infrequently.

It is possible to disable this tab so machine settings cannot be modified by unauthorized personnel. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.param_disable 1
```

CONFIGURATION - ARC

Name	Modi	Beschreibung
Start Fail Timer	0, 1, 2	This sets the amount of time (in seconds) QtPlasmaC will wait between commanding a ”Torch On” and receiving an Arc OK signal before timing out and displaying an error message.
Max. Starts	0, 1, 2	This sets the number of times QtPlasmaC will attempt to start the arc.
Retry Delay	0, 1, 2	This sets the time (in seconds) between an arc failure and another arc start attempt.

Name	Modi	Beschreibung
Voltage Scale	0, 1	This sets the arc voltage input scale and is used to display the correct arc voltage. For initial setup, see Calibration Values.
Voltage Offset	0, 1	This sets the arc voltage offset and is used to display zero volts when there is zero arc voltage input. For initial setup, see Calibration Values.
Height Per Volt	0, 1, 2	This sets the distance the torch would need to move to change the arc voltage by one volt. Used for manual height manipulation only.
OK High Volts	0	This sets the voltage threshold below which Arc OK signal is valid.
OK Low Volts	0	This sets the voltage threshold above which the Arc OK signal is valid.

Anmerkung
When setting the OK Low Volts and OK High Volts in Mode 0, the cut voltage of a stable arc must be greater than the OK Low Volts value but lower than the OK High Volts value for QtPlasmaC to receive a valid Arc OK signal. To further clarify, to have a valid Arc OK, the arc voltage must fall between the two limits.

CONFIGURATION - PROBING

Name	Beschreibung
Float Travel	This sets the amount of travel the float switch moves before completing the float switch circuit. This distance can be measured by using the Probe Test button, and the method described in Initial Setup.
Sonden-Geschwindigkeit	This sets the speed at which the torch will probe to find the material after it moves to the Probe Height.
Probe Height	This sets the height above the Z axis minimum limit that Probe Speed begins. Refer to the Heights Diagram diagram for a visual representation.
Ohmic Offset	This sets the distance above the material the torch will should go after a successful ohmic probe. It is mainly used to compensate for high probing speeds.
Ohmic Retries	This sets the number of times QtPlasmaC will retry a failed ohmic probe before falling back to the float switch for material detection.
Skip IHS	This sets the distance threshold used to determine if an Initial Height Sense (probe) can be skipped for the current cut, see IHS Skip.

Anmerkung
If the amount of time between the torch contacting the material and when the torch moves up and comes to rest at the Pierce Height seems excessive, see the probing section for a possible solution.

CONFIGURATION - SAFETY

Name	Beschreibung
Safe Height	This sets the height above the material that the torch will retract to before executing rapid moves. If set to Zero then Z axis maximum height will be used for the safe height. Refer to the Heights Diagram diagram for a visual representation.

CONFIGURATION - SCRIBING

Name	Beschreibung
Arm Delay	This sets the delay (in seconds) from the time the scribe command is received to the activation of the scribe. This allows the scribe to reach surface of the material before activating the scribe.
On Delay	This sets the delay (in seconds) to allow the scribe mechanism to start before beginning motion.

CONFIGURATION - SPOTTING

Name	Beschreibung
Threshold	This sets the arc voltage at which the delay timer will begin. 0 V starts the delay when the torch on signal is activated.
Time On	This sets the length of time (in milliseconds) the torch is on after threshold voltage is reached.

CONFIGURATION - MOTION

Name	Beschreibung
Max. Geschwindigkeit	Displays the maximum velocity the Z axis is capable of (this is controlled by the <*machine_name*>.ini file).
Setup Speed	The Z axis velocity for setup moves (movements to Probe Height, Pierce Height, Cut Height, etc.).

Anmerkung
Setup Speed has no effect on THC speed which is capable of the velocity displayed in the Max. Speed field.

CONFIGURATION - THC

Two methods of THC activation are available and are selected with the **Auto Activation** checkbutton. Both methods begin their calculations when the current velocity of the torch matches the cut feed rate specified for the selected material:

1. Delay Activation (the default) is selected when **Auto Activation** is unchecked. This method uses a time delay set with the **Delay** parameter.
2. Auto Activation is selected when **Auto Activation** is checked. This method determines that the arc voltage is stable by using the **Sample Counts** and **Sample Threshold** parameters.

Name	Modi	Beschreibung
Delay	0, 1, 2	This sets the delay (in seconds) measured from the time the Arc OK signal is received until Torch Height Controller (THC) activates. This is only available when Auto THC is not enabled.
Sample Counts	0, 1	This sets the number of consecutive arc voltage readings within THC Sample Threshold required to activate the Torch Height Controller (THC). This is only available when Auto THC is enabled.
Sample Threshold	0, 1	This sets the maximum voltage deviation allowed for THC Sample Counts. This is only available when Auto THC is enabled.
Threshold	0, 1	This sets the voltage variation allowed from the target voltage before for THC makes movements to correct the torch height.
Speed (PID-P)	0, 1, 2	This sets the Proportional gain for the THC PID loop. This roughly equates to how quickly the THC attempts to correct changes in height.

Name	Modi	Beschreibung
VAD Threshold	0, 1, 2	(Velocity Anti Dive) This sets the percentage of the current cut feed rate the machine can slow to before locking the THC to prevent torch dive.
Void Slope	0, 1	(Void Anti Dive) This sets the size of the change in cut voltage per seconds necessary to lock the THC to prevent torch dive (higher values need greater voltage change to lock THC).
PID-I	0, 1	This sets the Integral gain for the THC PID loop. Integral gain is associated with the sum of errors in the system over time and is not always needed.
PID-D	0, 1	This sets the Derivative gain for the THC PID loop. Derivative gain works to dampen the system and reduce over correction oscillations and is not always needed.

Anmerkung
PID loop tuning is a complicated process and is outside the scope of this User Guide. There are many sources of information available to assist with understanding and tuning PID loops. If the THC is not making corrections fast enough, it is recommended to increase the P gain in small increments until the system operates favorably. Large P gain adjustments can result in over correction and oscillations.

SAVE & RELOAD Buttons

The **SAVE** button will save the currently displayed parameters to the <*machine_name*>.prefs file.

The **RELOAD** button will reload all the parameters from the <*machine_name*>.prefs file.

MATERIAL

This section shows the parameters which are active for the current cut.

Name	Beschreibung
Material	The top drop down menu is used to manually select the current material cut parameters. If there are no materials in the material file then only the default material will be displayed.
Kerf Width	This sets the kerf width for the currently selected material. Refer to the Heights Diagram diagram for a visual representation.
Pierce Height	This sets the pierce height for the currently selected material. Refer to the Heights Diagram diagram for a visual representation.
Pierce Delay	This sets the pierce delay (in seconds) for the currently selected material.
Cut Height	This sets the cut height for the currently selected material. Refer to the Heights Diagram diagram for a visual representation.
Cut Feed Rate	This sets the cut feed rate for the currently selected material.
Cut Amps	This sets the cut amperage for the currently selected material. This is a visual indicator to the operator only, unless PowerMax communications are being used.
Cut Volts	This sets the cut voltage for the currently selected material.
Puddle Height	Expressed as a percentage of Pierce Height, this sets the Puddle Jump height for the currently selected material. Typically used for thicker materials, Puddle Jump allows the torch to have an intermediate step between Pierce Height and Cut Height. If set, the torch will proceed from Pierce Height to P-Jump Height for a period of time (P-Jump Delay) before proceeding to Cut Height to effectively ”jump” over the molten puddle. Refer to the Heights Diagram diagram for a visual representation.
Puddle Delay	This sets the amount of time (in seconds) the torch will stay at the P-Jump Height before proceeding to Cut Height.

Name	Beschreibung
Pause At End	This sets the amount of time (in seconds) the torch will stay on at the end of the cut before proceeding with the M5 command to turn off and raise the torch. For more information see Pause At End Of Cut.
Gas Pressure	This sets the gas pressure for the currently selected material. This setting is only valid if PowerMax communications are being used. 0 = Use the PowerMax's automatic pressure mode.
Cut Mode	This sets the cut mode for the currently selected material. This setting is only valid if PowerMax communications are being used. 1 = Normal 2 = CPA (Constant Pilot Arc) 3 = Gouge/Mark

Anmerkung
SEE THE THICK MATERIALS SECTION FOR MORE INFORMATION ON PUDDLE JUMP

SAVE, RELOAD, NEW, & DELETE Buttons

The **SAVE** button will save the current material set to the <machine_name>_material.cfg file.

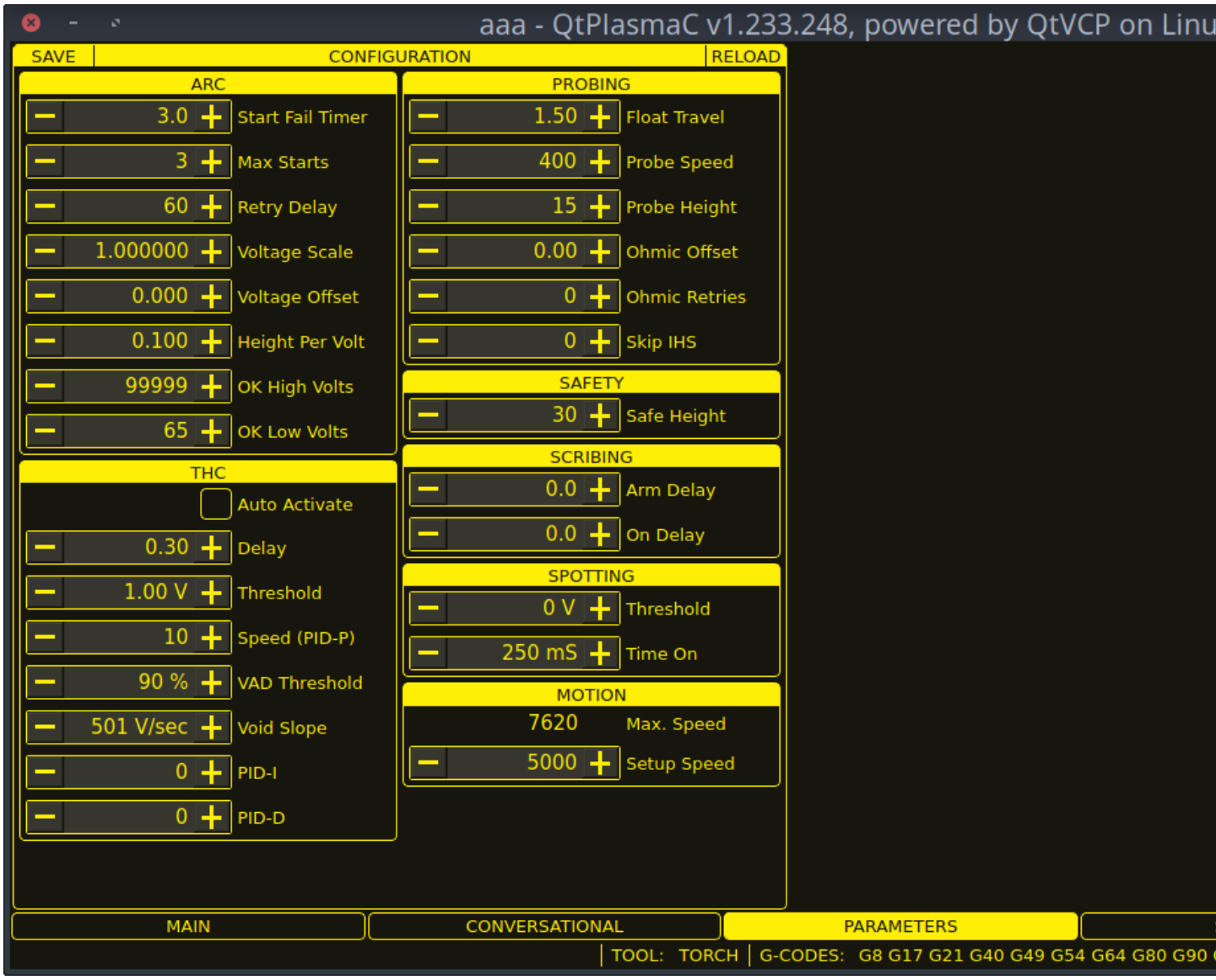
The **RELOAD** button will reload the material set from the <machine_name>_material.cfg file.

The **NEW** button will allow a new material to be added to the material file. The user will be prompted for a material number and a material name, all other parameters will be read from the currently selected material. Once entered, QtPlasmaC will reload the material file and display the new material. The Cut Parameters for the new material will then need to be adjusted and saved.

The **DELETE** this button is used to delete a material. After pressing it, the user will be prompted for a material number to be deleted, and prompted again to ensure the user is sure. After deletion, the material file will be reloaded and the drop down list will display the default material.

10.8.8.6 SETTINGS Tab

Screenshot example of the QtPlasmaC SETTINGS Tab in **16:9** aspect ratio:



This tab is used to display GUI configuration parameters, button text, and shutdown text that are modified infrequently as well as some utility buttons.

It is possible to disable this tab so machine settings cannot be modified by unauthorized personnel. This may be achieved either by wiring the pin to a physical key-switch or similar or it may also be set in a HAL file using the following command:

```
setp qtplasmac.settings_disable 1
```

GUI SETTINGS

This section shows parameters that effect the GUI appearance and GUI behaviors.

To return any of the color changes to their default values, see the Returning To The Default Styling section.

Name	Beschreibung
Foreground	This button allows the user to change the color of the GUI Foreground.
Highlight	This button allows the user to change the color of the GUI Highlight.
LED	This button allows the user to change the color of the GUI LED.
Background	This button allows the user to change the color of the GUI Background.

Name	Beschreibung
Alt Background	This button allows the user to change the color of the GUI Alternate Background.
Frames	This button allows the user to change the color of the GUI Frames.
Estop	This button allows the user to change the color of the GUI Estop.
Disabled	This button allows the user to change the color of the GUI's Disabled features.
Vorschau	This button allows the user to change the color of the GUI Preview Window Background.
Soft Keyboard	This radio button allows the user to enable or disable the soft touchscreen keyboard. If the "onboard" virtual keyboard is installed then the custom layouts will be enabled.
KB Shortcuts	This radio button allows the user to enable or disable Keyboard Shortcuts within the GUI (such as keyboard jogging). In addition to the standard jog keys, a list of the additional shortcuts is available in the keyboard shortcuts section.
View Material	This radio button allows the user to enable or disable the addition of a visual reference showing key material cut settings to the Preview Windows of the MAIN and CONVERSATIONAL tabs. Examples are: Feed Rate, Pierce Height, Pierce Delay, and Cut Height. Cut Amps will be shown if PowerMax communications are enabled.
Exit Warning	This radio button allows the user to enable or disable whether a warning will always be displayed during shutdown. It is possible to add a custom message to the warning by editing the EXIT WARNING MESSAGE in the **[GUI_OPTIONS]** section of the *<machine_name>*.prefs file. The custom message can be made multi-line by adding a "\" between lines.
Optional Stop	This radio button allows the user to enable or disable whether or not a running program will pause at an **M1** command.
Run From Line	This radio button allows the user to enable or disable Run From Line. If enabled, the user can click on a line of G-code and have the program start from that line.
Grenzwerte überschreiten	This radio button allows the user to temporarily Override the input from a Limit Switch in the event the limit switch becomes tripped during operation. This button can only be clicked when a limit switch is tripped.
Override Jog	This radio button will also allow jogging while jogging is inhibited due to a float switch, breakaway switch, or ohmic probe activation. This button can only be clicked when a jog is inhibited.
Optional Block	This radio button allows the user to enable or disable whether or not lines starting with "/" will be skipped if present in a running program.
Grid Size	This allows a user to change the size of the grid in the Preview Window on the MAIN Tab. Grid size of 0.0 will disable the grid.
Cone Size	This allows a user to change the size of the cone (which represents the current tool) in the Preview Window on the MAIN Tab.
Table Zoom	This allows a user to change the default zoom level for the top down full table view in the Preview Window on the MAIN Tab.

USER BUTTON ENTRIES USERBUTTON

This section shows the text that appears on the Custom User Buttons as well as the code associated with the user button. User buttons may be changed and the new settings used without restarting LinuxCNC.

The text and/or code may be edited at any time and will be loaded ready for use if the **SAVE** button is clicked.

Deleting the **Name** and **Code** text will cause that user button to be hidden if the **SAVE** button is

clicked.

To return all the **Name** and **Code** text to their last saved values press the **RELOAD** button.

Name	Code
The text that is displayed on the button	The code that is run when the button is pressed.

Anmerkung
There are 20 user buttons available but not all may be displayed depending on the window size.

EXIT WARNING MESSAGE

This section shows the text that appears on the shutdown dialog if the **Exit Warning** is enabled .

The text may be edited at any time and will be loaded ready for use if the **SAVE** button is clicked.

To return the **EXIT WARNING MESSAGE** text to the last saved value press the **RELOAD** button.

UTILITIES

Some standard LinuxCNC utilities are provided as an aid in the diagnosis of issues that may arise:

- Halshow
- Halscope
- Halmeter
- Calibration
- Status

In addition the following two QtPlasmaC specific utilities are provided:

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.

The **BACKUP CONFIG** button will create a complete machine configuration backup for archiving or to aid in fault diagnosis. A compressed backup of the machine configuration will be saved in the user's Linux home directory. The file name will be <machine_name>_<version>_<date>_<time>.tar.gz where <machine_name> is the machine name entered in the configuration wizard, <version> is the current QtPlasmaC version the user is on, <date> is the current date (YY-MM-DD), and <time> is the current time (HH-MM-SS).

Prior to the backup being made, the machine log will be saved to a file in the configuration directory named machine_log_<date>_<time>.txt where <date> and <time> are as described above. This file along with up to five previous machine logs will also be included in the backup.

These files are not required by QtPlasmaC and are safe to delete at any time.

10.8.8.7 STATISTICS Tab

The STATISTICS Tab provides statistics to allow for the tracking of consumable wear and job run times.

These statistics are shown for the current job as well as the running total.

Previous job statistics are reset once the next program is run.

The total values may be reset either individually by clicking the corresponding ”RESET” button, or they may all be reset together by clicking ”RESET ALL”.

The **RS485 PMX STATISTICS** panel will be only be displayed if the user has Hypertherm PowerMax communications and a valid RS485 connection to the PowerMax is established. This panel will show the **ARC ON TIME** for the PowerMax in hh:mm:ss format.

The **MACHINE LOG** is also displayed on the STATISTICS Tab, this log will display any errors and/or important information that occurs during the current LinuxCNC session. If the user makes a backup of the configuration from the SETTINGS Tab then the machine log is also included in the backup.

10.8.9 Using QtPlasmaC

Once QtPlasmaC is successfully installed, no Z axis motion is required to be part of the G-code cut program. In fact, if any Z axis references are present in the cut program, the standard QtPlasmaC configuration will remove them during the program loading process.

For reliable use of QtPlasmaC the user should **NOT** use any Z axis offsets other than the coordinate system offsets (G54-G59.3).

QtPlasmaC fügt am Anfang jedes G-Code-Programms automatisch eine Zeile G-Code ein, um die Z-Achse auf die richtige Höhe zu bringen.

Versionsinformationen - QtPlasmaC zeigt im Titel des Hauptfensters Informationen zur Versionierung an. Die Informationen werden wie folgt angezeigt ”QtPlasmaC vN.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.

10.8.9.1 Einheitensysteme

All settings and parameters in QtPlasmaC are required to be in the same units as specified in the INI file, being either metric or 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.

10.8.9.2 Präambel und Postambel Codes

The following stanzas are the minimum recommended codes to include in the preamble and postamble of any G-code file to be run by QtPlasmaC:

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.

10.8.9.3 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 $0**, um einen Schneidauftrag zu beenden. Beachten Sie, dass dies nur für Schneideaufträge gilt, für Ritz- und Tuschieraufträge ist die Werkzeugkennung ***$n*** obligatorisch.

10.8.9.4 Koordinaten

Siehe empfohlene Z-Achse Einstellungen.

Each time LinuxCNC (QtPlasmaC) is started Joint homing is required. This allows LinuxCNC (QtPlasmaC) to establish the known coordinates of each axis and set the soft limits to the values specified in the <*machine_name*>.ini file in order to prevent the machine from crashing into a hard stop during normal use.

If the machine does not have home switches then the user needs to ensure that all axes are at the home coordinates specified in the <*machine_name*>.ini file before homing.

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 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.

If the user intends to place the material in the exact same place on the table every time, the user could jog the X and Y axes to the machine to the corresponding X0 Y0 position as established by the CAM software and then **Touch Off** both axes with a zero offset.

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.

10.8.9.5 Cut Feed Rate

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.

10.8.9.6 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 file trägt den Namen **<Maschinen_name>_material.cfg**.

QtPlasmaC erfordert nicht die Verwendung einer Materialdatei. Stattdessen kann der Benutzer die Schnittparameter manuell im Abschnitt MATERIAL des PARAMETERS Tab 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

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

Anmerkung
Die Materialnummern 1000000 und höher sind für temporäre Materialien reserviert.

Warnung
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 PARAMETERS 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 PARAMETERS Tab, um die Materialdatei neu zu laden.

10.8.9.7 Manuelles Materialhandling

Bei der manuellen Materialhandhabung würde der Benutzer das Material manuell aus der Materialliste im Abschnitt MATERIAL des PARAMETERS Tab auswählen, bevor er das G-Code-Programm startet. Zusätzlich zur Auswahl von Materialien aus der Materialliste im Abschnitt MATERIAL des PARAMETERS Tab 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
```

Anmerkung
Bei der manuellen Materialhandhabung kann der Benutzer nur ein Material für die gesamte Arbeit verwenden.

10.8.9.8 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 PARAMETERS Tab 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 MAIN Tab angezeigt, während das Programm geladen wird. Der folgende Code ist der Mindestcode, 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
```

10.8.9.9 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:

- Add new materials to the <*machine_name*>_material.cfg file.
- Edit existing materials in the <*machine_name*>_material.cfg file.
- 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.

Tipp
It is possible to use temporary materials only and have an empty <machine_name>_material.cfg file. This negates the need to keep the QtPlasmaC materials file updated with the CAM tool file.

- Der gesamte Kommentar muss in Klammern gesetzt werden.
- The beginning of the magic comment must be: **(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	Creates a temporary default material. Material information added with this option will be discarded by a LinuxCNC restart or materials reload. They may also be overwritten by a new G-code file that has temporary materials.
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
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	Sets the puddle jump height.
jd	Sets the puddle jump delay.

Ein vollständiges Beispiel:

```
(o=0, nu=2, na=5mm Baustahl (engl. mild steel) 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw ↩
    =0.5, th=1, ca=45, cv=110, pe=0.1, gp=5, 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.

10.8.9.10 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 tool tables are complete and the conversion is fully automatic. The SheetCam tool file must be in the SheetCam .tools format.

Fusion 360 tool tables do not have all of the required fields so the user will be prompted for missing parameters. The Fusion 360 tool file must be in the JSON format of Fusion 360.

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
```

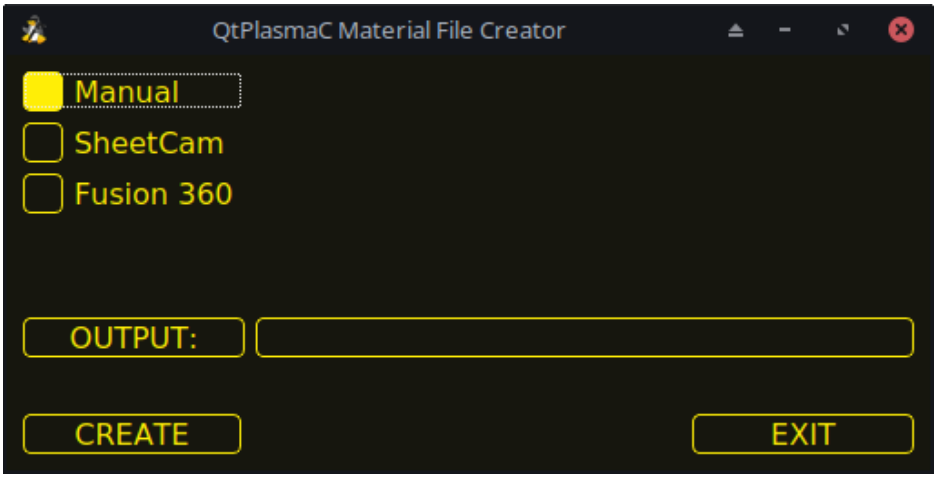
Geben Sie für eine ”run in place”-Installation die folgenden beiden Befehle in ein Terminalfenster ein:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

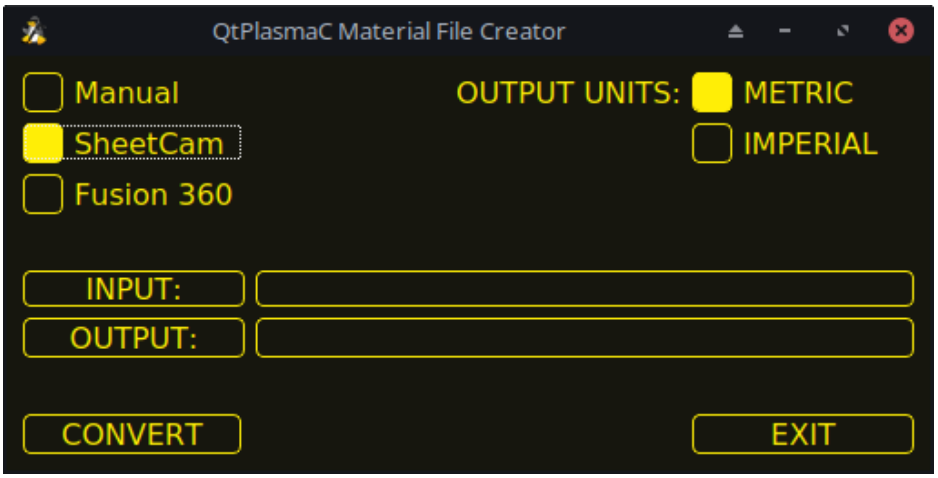
Daraufhin wird das Hauptdialogfeld Materialkonverter mit der Standardeinstellung Manuell angezeigt.

Wählen Sie eines aus:

- **Manuell** - um manuell eine neue Materialdatei zu erstellen.

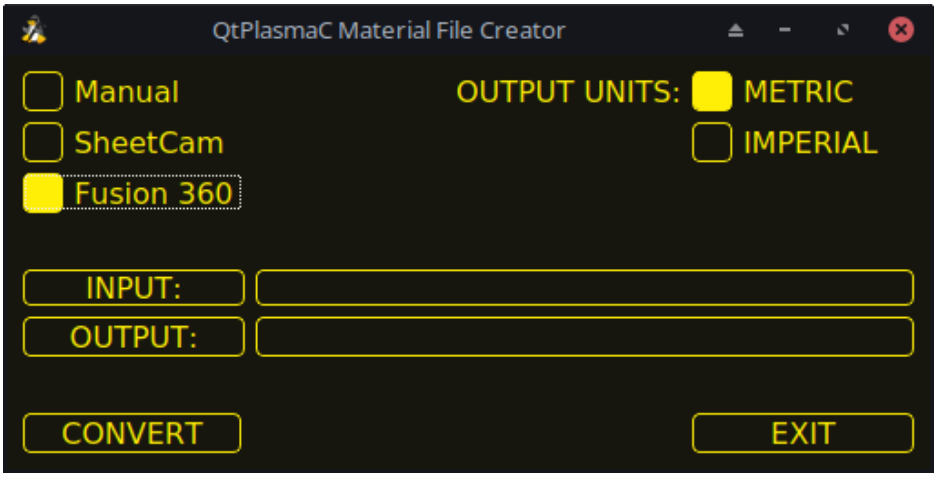


- **SheetCam** - um eine SheetCam-Werkzeugdatei zu konvertieren.



Wählen Sie nur für SheetCam, ob der Benutzer eine metrische oder imperiale Ausgabedatei benötigt.

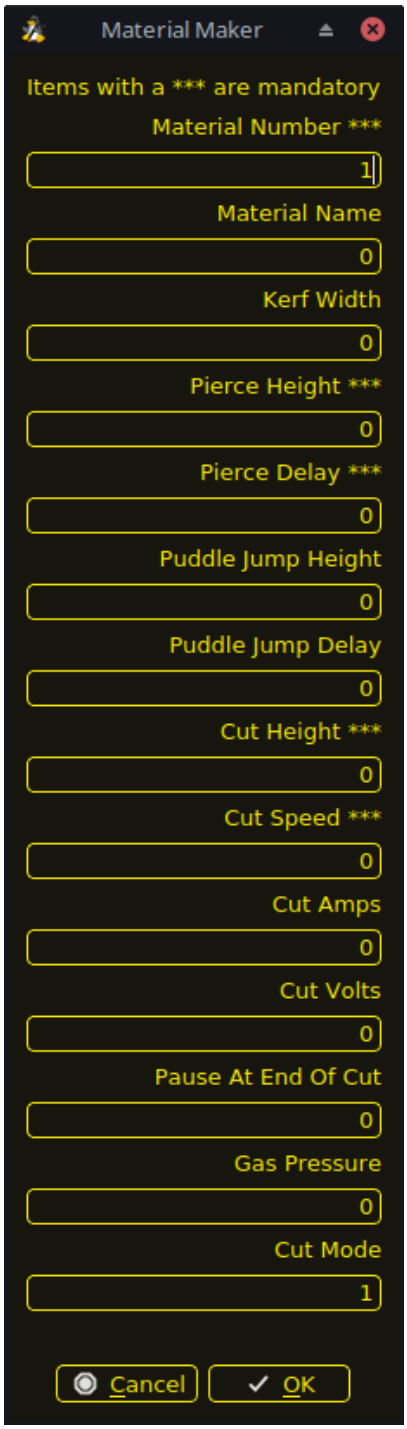
- **Fusion 360** - zum Konvertieren einer Fusion 360 Werkzeugdatei.



Um zu konvertieren:

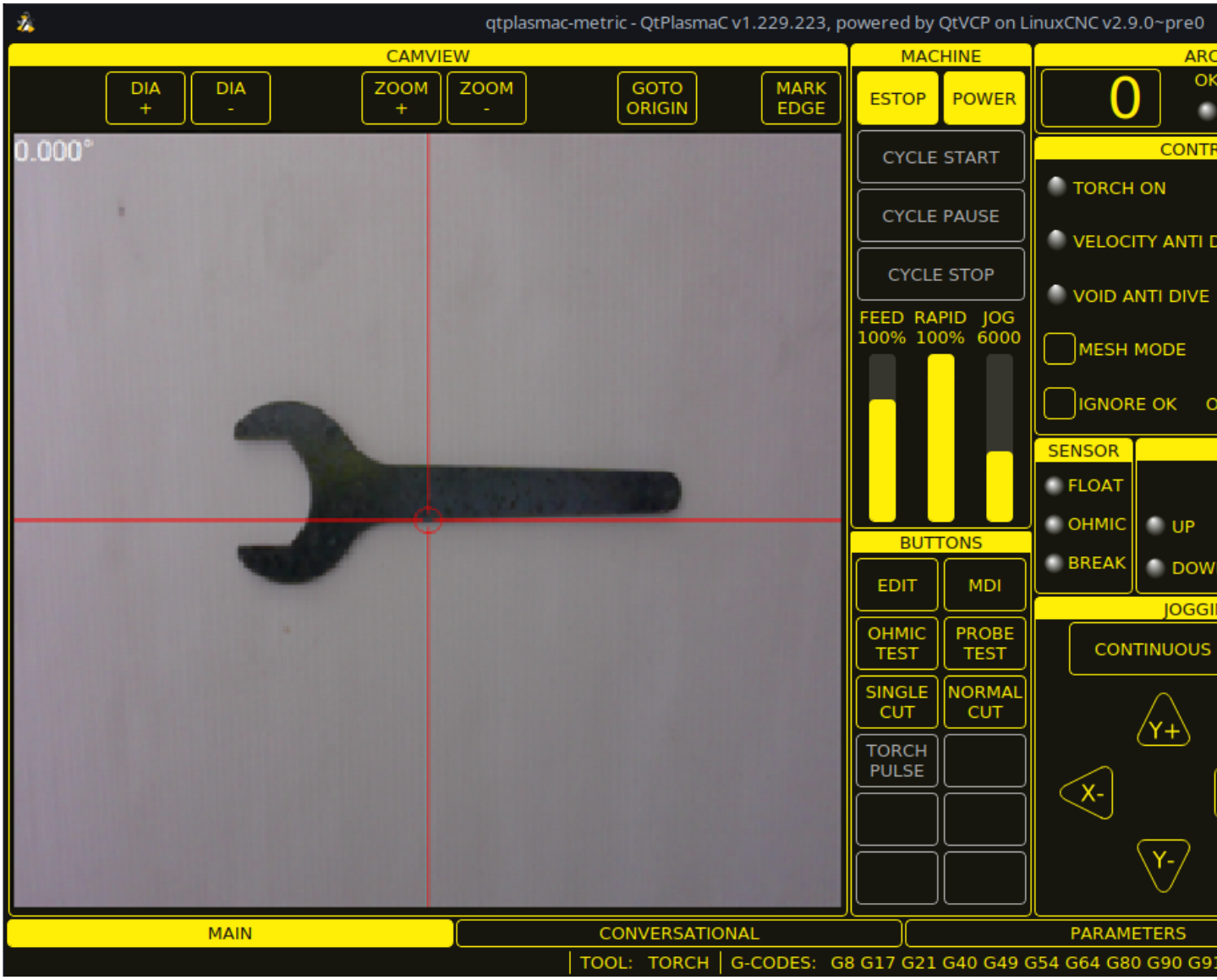
1. Wählen Sie die zu konvertierende Eingabedatei, drücken Sie **INPUT**, um eine Dateiauswahl aufzurufen, oder geben Sie die Datei direkt in das Eingabefeld ein.
2. Wählen Sie die Ausgabedatei, in die geschrieben werden soll, drücken Sie **OUTPUT**, um eine Dateiauswahl aufzurufen, oder geben Sie die 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.



Anmerkung
Wenn der Benutzer ~/linuxcnc/configs/<Maschinenname>_material.cfg auswählt und die Datei bereits existiert, wird sie überschrieben.

10.8.9.11 CAMERA



QtPlasmaC hat die Möglichkeit, eine USB-Kamera zu verwenden, um den Ursprung mit oder ohne Rotationskompensation festzulegen. Die Schaltfläche CAMERA wird aktiviert, nachdem das Gerät referenziert 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.

To modify the offsets manually, the user could edit either or both the following lines in the **[CAMERA_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
```

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.

10.8.9.12 LASER

QtPlasmaC hat die Möglichkeit, einen Laser zum Setzen des Ursprungs mit oder ohne Rotationskompensation zu verwenden. Die LASER-Schaltfläche wird aktiviert, nachdem die Maschine referenziert 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.

To modify the offsets manually, the user could edit either or both the following lines in the **[LASER_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
```

where *n.n* is distance from the center line of the torch to the laser's cross hairs.

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 eingestellt.

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 mSek. 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.

10.8.9.13 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.

The provided G-code filter program will test for the existence of a `G64 P__n__` command prior to the first motion command. If no G64 command is found it will insert a `G64 P0.1` command which sets the path tolerance to 0.1 mm. For a imperial config the command will be `G64 P0.004`.

Für metrisch:

```
G64 P0.1
```

Für imperial:

```
G64 P0.004
```

10.8.9.14 Angehaltene Bewegung

QtPlasmaC ermöglicht die Neupositionierung der X- und Y-Achse entlang des aktuellen Schnittpfades, während das G-Code-Programm pausiert.

Um diese Funktion nutzen zu können, muss die adaptive Vorschubsteuerung (M52) von LinuxCNC eingeschaltet sein (P1).

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
```

10.8.9.15 Pause am Ende von Cut

This feature can be used to allow the arc to ”catch up” to the torch position to fully finish the cut. It is usually required for thicker materials and is especially useful when cutting stainless steel.

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 Registerkarte PARAMETERS eingestellt wurde, fährt QtPlasmaC mit dem Befehl M5 fort, um den Brenner auszuschalten und anzuheben.

10.8.9.16 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.
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.

10.8.9.17 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 auf den im Befehl angegebenen Prozentsatz reduziert.

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% von **CutFeedRate** gesetzt.
- Mit **M67 E3 Q40** würde die Geschwindigkeit auf 40% von **CutFeedRate** gesetzt.

- Mit **M67 E3 Q60** würde die Geschwindigkeit auf 60% von **CutFeedRate** gesetzt.
- Mit **M67 E3 Q100** würde die Geschwindigkeit auf 100% von **CutFeedRate** gesetzt.

Der zulässige Mindestprozentsatz beträgt 10 %, darunter liegende Werte werden auf 10 % gesetzt.

Der maximal zulässige Prozentsatz beträgt 100%, darüber liegende Werte 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.

Wichtig
G-CODE THC UND **VELOCITY BASED THC** KÖNNEN NICHT VERWENDET WERDEN, WENN **CUTTER COMPENSATION** IN KRAFT IST; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.

Warnung
Wenn Cut Feed Rate im Abschnitt MATERIAL des PARAMETERS Tab auf Null gesetzt ist, dann verwendet QtPlasmaC **motion.requested-velocity** (wie durch einen Standard Feedrate-Aufruf im G-Code eingestellt) für die THC-Berechnungen. Dies wird nicht empfohlen, da es kein zuverlässiger Weg ist, geschwindigkeitsbasierte THC zu implementieren.

Anmerkung
Alle Verweise auf CutFeedRate beziehen sich auf den Wert **Cut Feed Rate**, der im Abschnitt MATERIAL zum PARAMETER Tab angezeigt wird.

10.8.9.18 THC (Brennerhöhensteuerung, engl. torch height controller)

Die THC kann über den THC-Rahmen des Haupt-Registrierkarte (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-Registrierkarte abgelaufen ist. Dies dient dazu, dass sich die Lichtbogenspannung stabilisieren kann.

QtPlasmaC verwendet eine Steuerspannung, die vom Zustand der Checkbox **AUTO VOLTS** auf dem 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 MAIN Tab 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.

Geschwindigkeitsbasiertes 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 PARAMETERS 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 MAIN Tab 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 Velocity Reduction die geschwindigkeitsbasierte THC auf folgende Weise beeinflusst:

1. If Velocity Reduction is invoked in the middle of the cut, the THC will be locked.
2. The THC will remain locked until the velocity reduction is canceled by returning it to a value that is above the **VAD Threshold**, and the torch actually reaches 99.9% of the **CutFeedRate**.

10.8.9.19 Fräserkompensation

LinuxCNC (QtPlasmaC) has the ability to automatically adjust the cut path of the current program by the amount specified in Kerf Width of the selected material's Cut Parameters. This is helpful if the G-code is programmed to the nominal cut path and the user will be running the program on different thickness materials to help ensure consistently sized parts.

To use cutter compensation the user will need to use G41.1, G42.1 and G40 with the kerf width HAL pin:

- **G41.1 D#<_hal[plasmac.kerf-width]>** : offsets torch to the left of the programmed path
- **G42.1 D#<_hal[plasmac.kerf-width]>** : offsets torch to the right of the programmed path
- **G40** turns the cutter compensation off

Wichtig
IF **CUTTER COMPENSATION** IS IN EFFECT **G-CODE THC**, **VELOCITY BASED THC** AND **OVER CUT** ARE NOT ABLE TO BE USED; AN ERROR MESSAGE WILL BE DISPLAYED.

10.8.9.20 Initial Height Sense (IHS) Skip

Initial Height Sense may be skipped in one of two different ways:

1. If the THC is disabled, or the THC is enabled but not active, then the IHS skip will occur if the start of the cut is less than **Skip IHS** distance from the last successful probe.
2. If the THC is enabled and active, then the IHS skip will occur if the start of the cut is less than **Skip IHS** distance from the end of the last cut.

A value of zero for **Skip IHS** will disable all IHS skipping.

Any errors encountered during a cut will disable IHS skipping for the next cut if **Skip IHS** is enabled.

10.8.9.21 Sondieren

Probing may be done with either ohmic sensing or a float switch. It is also possible to combine the two methods, in which case the float switch will provide a fallback to ohmic probing. An alternative to ohmic probing is Offset Probing

If the machine's torch does not support ohmic probing, the user could have a separate probe next to the torch. In this case the user would extend the probe below the torch. The probe must NOT extend more than the minimum Cut Height below the torch and the Z axis offset distance needs to be entered as the **Ohmic Offset** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab.

Probing setup is done in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab.

QtPlasmaC can probe at the full Z axis velocity so long as the machine has enough movement in the float switch to absorb any overrun. If the machine's float switch travel is suitable, the user could set the Probe Height to near the Z axis MINIMUM_LIMIT and do all probing at full speed.

Some float switches can exhibit a large switching hysteresis which shows up in the probing sequence as an excessive time to complete the final probe up.

- This time may be decreased by changing the speed of the final probe up.
- This speed defaults to 0.001 mm (0.000039") per servo cycle.
- It is possible to increase this speed by up to a factor of 10 by adding the following line to the custom.hal file:

```
setp plasmac.probe-final-speed n
```

where *n* is a value from 1-10. It is recommended to keep this value as low as possible.

Using this feature will change the final height slightly and will require thorough probe testing to confirm the final height.

This speed value affects ALL probing so if the user uses ohmic probing and the user changes this speed value then the user will need to probe test to set the require offset to compensate for this speed change as well as the float travel.

The reliability of this feature will only be as good as the repeatability of the float switch.

Anmerkung
Probe Height refers to the height above the Z axis MINIMUM_LIMIT.

10.8.9.22 Offset Probing

Offset Probing is the use of a probe that is offset from the torch. This method is an alternative to Ohmic Probing and uses the `plasmac.ohmic-enable` output pin to operate a solenoid for extending and retracting the probe. The `plasmac.ohmic-probe` input pin is used to detect the material and the **Ohmic Offset** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab is used to set the correct measured height.

The probe could be a mechanically deployed probe, a permanently mounted proximity sensor or even simply a stiff piece of wire extending about 0.5 mm (0.2") below the torch tip. If the probe is mechanically deployed then it needs to extend/retract rather quickly to avoid excessive probing times and would commonly be pneumatically operated.

To use this feature, the user must set the probe's offset from the torch center by following the procedure described in Peripheral Offsets.

To modify the offsets manually, the user could edit either or both the following lines in the **[OFFSET_PROBING]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
Delay = t.t
```

where *n.n* is the offset of the probe from the torch center in machine units for the X and Y axes and *t.t* is the time in seconds to allow for any mechanical deployment of the probe if required.

Each of these parameters is optional and also may appear in any order. If a parameter is not detected then the default is 0.0. There can be no space after the X or Z, lower case is permissible.

When this variable appears in the INI file with either X or Y not equal to zero then QtPlasmaC will do **all** Ohmic Probing as Offset Probing. When a probe sequence has begun, the `plasmac.ohmic-enable` pin will be set True causing the probe to extend. When the material is detected the `plasmac.ohmic-enable` pin will be reset to false causing the probe to retract.

The probe will begin moving to the offset position simultaneously with the Z axis moving down to the Probe Height, probing will not commence unless the deployment timer has completed. It is required that the **Probe Height** in the PROBING frame of the CONFIGURATION section of the PARAMETERS Tab is above the top of the material to ensure that the probe is fully offset to the correct X/Y position before the final vertical probe down movement.

Wichtig
PROBE HEIGHT NEEDS TO BE SET ABOVE THE TOP OF THE MATERIAL FOR OFFSET PROBING

10.8.9.23 Cut Types

QtPlasmaC allows two different cut modes:

1. **NORMAL CUT** - runs the loaded G-code program to pierce then cut.
2. **PIERCE ONLY** - only pierces the material at each cut start position, useful prior to a **NORMAL CUT** on thick materials

There are two ways of enabling this feature:

1. Utilize the default custom user button to toggle between the cut types.

2. Adding the following line to the G-code program before the first cut to enable **Pierce Only** mode for the current file:

```
#<pierce-only> = 1
```

If using a custom user button is utilized then QtPlasmaC will automatically reload the file when the cut type is toggled.

10.8.9.24 Hole Cutting - Intro

It is recommended that any holes to be cut have a diameter no less than one and a half times the thickness of the material to be cut.

It is also recommended that holes with a diameter of less than 32 mm (1.26”) are cut at 60% of the feed rate used for profile cuts. This should also lock out THC due to velocity constraints.

QtPlasmaC can utilize G-code commands usually set by a CAM Post Processor (PP) to aid in hole cutting or if the user does not have a PP or the user’s PP does not support these methods then QtPlasmaC can automatically adapt the G-code to suit. This automatic mode is disabled by default.

Es gibt drei Methoden zur Verbesserung der Qualität von kleinen Löchern:

1. **Velocity Reduction** - Reducing the velocity to approximately 60% of the **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.

Anmerkung
Wenn sowohl **Arc Dwell** als auch **Over Cut** gleichzeitig aktiv sind, hat **Over Cut** Vorrang.

Wichtig
DIE FUNKTION **OVER CUT** KANN NICHT VERWENDET WERDEN, WENN DIE SCHNEIDEKOMPENSATION AKTIVIERT IST; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.

10.8.9.25 Löcher schneiden

G-Code-Befehle können entweder von einem CAM-Postprozessor (PP) oder durch Handcodierung erstellt werden.

Hole Cutting Velocity Reduction

If cutting a hole requires a reduced velocity then the user would use the following command to set the velocity: **M67 E3 Qnn** where nn is the percentage of the velocity desired. For example, **M67 E3 Q60** would set the velocity to 60% of the current material’s **CutFeedRate**.

See the Velocity Based THC section.

Sample code:

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
M67 E3 Q60 (reduce feed rate to 60%)
G3 I10 (the hole)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

Arc Dwell (Pause At End)

This method can be invoked by setting the Pause At End parameter in the MATERIAL frame of the PARAMETERS Tab.

Over cut

The torch can be turned off at the end of the hole by setting the **motion.digital-out-03** pin with the M-Codes **M62 (Synchronized with Motion)** or **M64 (Immediate)**. After turning the torch off it is necessary to allow the torch to be turned on again before beginning the next cut by resetting the **motion.digital-out-03** pin with the M-Codes **M63** or **M65**, this will be done automatically by the QtPlasmaC G-code parser if it reaches an M5 command without seeing a **M63 P3** or **M65 P3**.

After the torch is turned off the hole path will be followed for a default length of 4mm (0.157”). This distance may be specified by adding **#<oclength> = *n*** to the G-code file.

- **M62 P3** will turn the torch off (Synchronized with Motion)
- **M63 P3** will allow the torch to be turned on (Synchronized with Motion)
- **M64 P3** will turn the torch off (Immediately)
- **M65 P3** will allow the torch to be turned on (Immediately)

It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- **M62 und M63** (Synchronized with Motion) – Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2 (THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn kein nachfolgender Bewegungsbefehl vorhanden ist, 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** (unmittelbar) - Diese Befehle erfolgen sofort, wenn sie vom Motion Controller empfangen werden. Da diese nicht mit der Bewegung synchronisiert sind, unterbrechen sie die Mischung. Das heißt, wenn diese Codes in der Mitte aktiver Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

Sample code:

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
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 4mm)
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)
```

10.8.9.26 Hole Cutting - Automatic

QtPlasmaC has the ability to automatically modify the G-code to reduce the velocity and/or apply **Over cut** which can be useful when cutting holes.

For valid hole sensing it is required that all values in the G2 or G3 G-code line are explicit, an error dialog will be displayed if any values are mathematically calculated.

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** - Causes QtPlasmaC to reduce the speed of holes less than 32 mm (1.26”) to 60% of **CutFeedRate**.
- **#<holes> = 2** - Bewirkt, dass QtPlasmaC Over cut das Loch zusätzlich zu den Geschwindigkeitsänderungen in Einstellung 1.
- **#<holes> = 3** - Causes QtPlasmaC to reduce the speed of holes less than 32 mm (1.26”) and arcs less than 16 mm (0.63”) to 60% of **CutFeedRate**.
- **#<holes> = 4** - Bewirkt, dass QtPlasmaC Over cut das Loch zusätzlich zur Geschwindigkeitsänderung in Einstellung 3.

The default hole size for QtPlasmaC hole sensing is 32 mm (1.26”). It is possible to change this value with the following command in a G-code file:

- **#<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** - to set the percentage (nn) of the current feed rate required.

Over cut

If Hole Sensing modes 2 or 4 are active, QtPlasmaC will over cut the hole in addition to the velocity changes associated with modes 1 and 3.

The default over cut length for QtPlasmaC hole sensing is 4mm (0.157”). It is possible to change this value with the following command in a G-code file:

- **#<oclength> = nn** to specify an over cut length (nn) in the same units system as the rest of the G-code file.

Arc Dwell (Pause At End)

This feature can be used in addition to setting the desired hole sensing mode via the appropriate G-code parameter by setting the Pause At End parameter in the MATERIAL frame of the PARAMETERS Tab.

Sample code:

```
G21 (metric)
G64 P0.005
M52 P1 (allow paused motion)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
#<holes> = 2 (over cut for holes)
#<oclength> = 6.5 (optional, 6.5mm 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)
```

Anmerkung
It is OK to have multiple and mixed hole commands in a single G-code file.

10.8.9.27 Single Cut

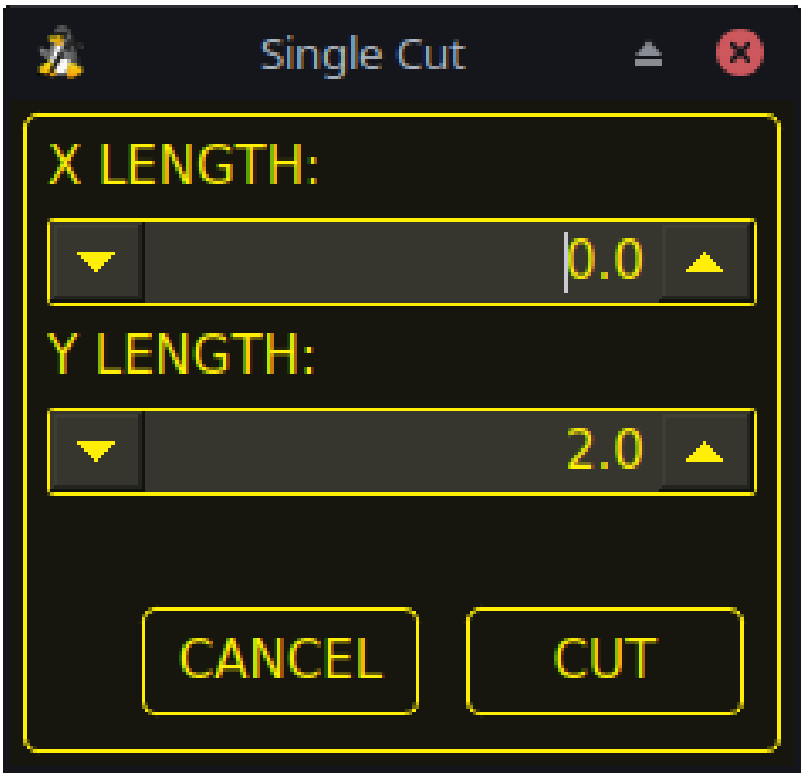
A single cut is a single unidirectional cutting move often used to cut a sheet into smaller pieces prior to running a G-code program.

The machine needs to be homed before commencing a single cut.

A single cut will commence from the machine's current X/Y position.

Automatic Single Cut

This is the preferred method. The parameters for this method are entered in the following dialog box that is displayed after pressing a user button which has been coded to run single cut:



1. Jog to the required X/Y start position.
2. Set required appropriate material, or edit the Feed Rate for the default material in the PARAMETERS Tab.
3. Press the assigned single cut user button.
4. Enter the length of the cut along the X and/or Y axes.
5. Press the **CUT** button and the cut will commence.

Pendant Single Cut

If the machine is equipped with a pendant that can start and stop the spindle plus jog the X and Y axes, the user can manually perform a single cut.

1. Jog to the required X/Y start position.
2. Set the required feed rate with the Jog Speed slider.
3. Start the cut process by starting the spindle.
4. After probing the torch will fire.
5. When the Arc OK is received the machine can be jogged along the cut line using the jog buttons.
6. When the cut is complete stop the spindle.
7. The torch will turn off and the Z axis will return to the starting position.

Manual Single Cut

Manual single cut requires that either keyboard shortcuts are enabled in the GUI SETTINGS section of the SETTINGS Tab, or a custom user button is specified as a manual cut button.

If the user is using a custom user button then substitute **F9** with **User Button** in the following description.

1. Jog to the required X/Y start position.
2. Start the procedure by pressing **F9**. The jog speed will be automatically set to the feed rate of the currently selected material. The jog label will blink to indicate that the jog speed is temporarily being overridden (jog speed manipulation will be disabled while a manual cut is active). **CYCLE START** will change to **MANUAL CUT** and blink.
3. After probing the torch will fire.
4. When the Arc OK is received the machine can be jogged along the cut line using the jog keys.
5. The Z height will remain locked at the cut height for the duration of the manual cut, regardless of the Torch Height Controller **ENABLE** status.
6. When the cut is complete press **F9** or **Esc** or the **CYCLE STOP** button.
7. The torch will turn off and the Z axis will return to the starting position.
8. The jog speed will automatically be returned to the value it was prior to initiating the manual cut process, the label will stop blinking and the jog speed manipulation will be enabled. **MANUAL CUT** will stop blinking and revert to **CYCLE START**.

Anmerkung
If the torch flames out during cutting, the user must still press **F9** or **Esc** or the **CYCLE STOP** button to end the cut. This clears the Z offsets and returns the torch to the starting position.

10.8.9.28 Thick Materials

Cutting thick materials can be problematic in that the large amount of molten metal caused by piercing can shorten the life of consumables and also may cause a puddle high enough that the torch may hit the puddle while moving to cut height.

The are two functions built into QtPlasmaC to help alleviate these issues.

Pierce Only

Pierce Only mode converts the loaded G-code program and then runs the program to pierce the material at the start position of each cut. Scribe and Spotting commands will be ignored and no pierce will take place in those locations.

This mode is useful for thick materials which may produce enough dross on the material surface from piercing to interfere with the torch while cutting. The entire sheet can pierced and then cleaned off prior to cutting.

It is possible to use near-end-of-life consumables for piercing and then they can be swapped out for good consumables to be used while cutting.

Pierce Only is one of two different cut types

Puddle Jump

Puddle Jump is the height that the torch will move to after piercing and prior to moving to **Cut Height** and is expressed as a percentage of **Pierce Height**. This allows the torch to clear any puddle of molten material tht may be caused by piercing. The maximum allowable height is 200% of the **Pierce Height**

Setting for **Puddle Jump** are described in cut parameters

The recommended option is to use **Pierce Only** due to it being able to utilise near end of life consumables.

Wichtig
PUDDLE JUMP IS DISABLED DURING CUT RECOVERY

10.8.9.29 Mesh Mode (Expanded Metal Cutting)

QtPlasmaC is capable of cutting of expand (mesh) metal provided the machine has a pilot arc torch and it is capable of Constant Pilot Arc (CPA) mode.

Mesh Mode disables the THC and also ignores a lost Arc OK signal during a cut. It can be selected by checking the **Mesh Mode** check button in the CONTROL section of the MAIN Tab.

If the machine has RS485 communications enabled with a Hypertherm PowerMax plasma cutter, selecting **Mesh Mode** will automatically override the **Cut Mode** for the currently selected material and set it to cut mode 2 (CPA). When **Mesh Mode** is disabled, the **Cut Mode** will be return to the default cut mode for the currently selected material.

It is also possible to start a **Mesh Mode** cut without receiving an Arc OK signal by checking the **Ignore Arc OK** check button in the CONTROL section of the MAIN Tab.

Both **Mesh Mode** and **Ignore Arc OK** can be enabled/disabled at any time during a job.

10.8.9.30 Ignore Arc OK

Ignore Arc OK mode disables the THC, will begin a cut without requiring an Arc OK signal, and will ignore a lost Arc OK signal during a cut.

This mode can be selected by:

1. Checking the **Ignore Arc OK** check button in the CONTROL section of the MAIN Tab.
2. Setting HAL pin **motion.digital-out-01** to 1 via G-code.
   - **M62 P1** will enable **Ignore Arc OK** (Synchronized with Motion)
   - **M63 P1** will disable **Ignore Arc OK** (Synchronized with Motion)
   - **M64 P1** will enable **Ignore Arc OK** (Immediately)
   - **M65 P1** will disable **Ignore Arc OK** (Immediately)

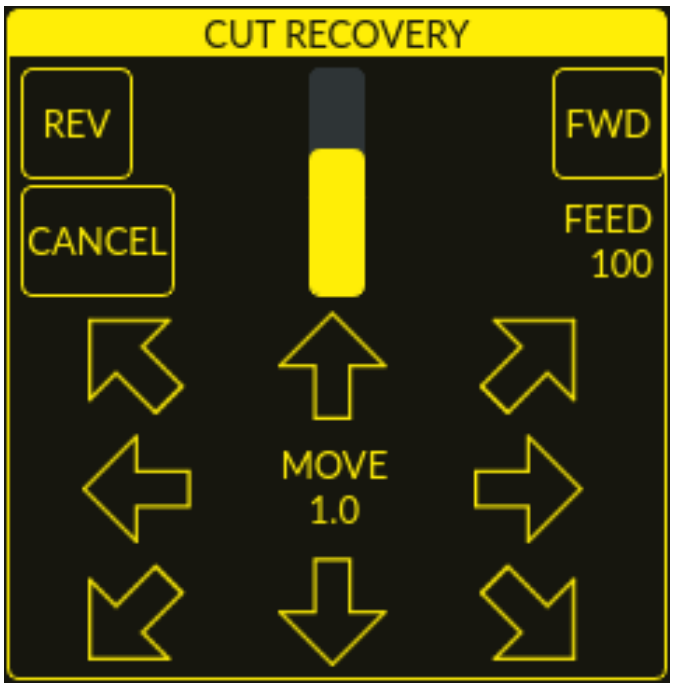
It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- **M62 und M63** (Synchronized with Motion) – Die tatsächliche Änderung des angegebenen Ausgangs (z. B. P2 (THC)) erfolgt zu Beginn des nächsten Bewegungsbefehls. Wenn kein nachfolgender Bewegungsbefehl vorhanden ist, 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** (unmittelbar) - Diese Befehle erfolgen sofort, wenn sie vom Motion Controller empfangen werden. Da diese nicht mit der Bewegung synchronisiert sind, unterbrechen sie die Mischung. Das heißt, wenn diese Codes in der Mitte aktiver Bewegungscodes verwendet werden, wird die Bewegung angehalten, um diese Befehle zu aktivieren.

This mode may also be used in conjunction with **Mesh Mode** if the user doesn't require an Arc OK signal to begin the cut.

Both **Mesh Mode** and **Ignore Arc OK** can be enabled/disabled at any time during a job.

10.8.9.31 Cut Recovery



This feature will produce a CUT RECOVERY panel that will allow the torch to be moved away from the cut path during a paused motion event in order to position the torch over a scrap portion of the material being cut so that the cut restarts with a minimized arc-divot. The CUT RECOVERY panel will display automatically over top of the JOGGING panel when motion is paused.

It is preferable to make torch position adjustments from the point at which paused motion occurred, however if moving along the cut path is necessary prior to setting the new start point, the user may use the paused motion controls (**REV**, **FWD**, and a **JOG-SPEED** slider) at the top of the CUT RECOVERY panel. Once the user is satisfied with the positioning of the torch along the cut path, moving off the cut path is achieved by pressing the **DIRECTION** buttons. Each press of the **DIRECTION** button will move the torch a distance equivalent to the **Kerf Width** parameter of the currently selected material.

The moment the torch has been moved off the cut path, the paused motion controls (**REV**, **FWD**, and a **JOG-SPEED** slider) at the top of the CUT RECOVERY panel will become disabled.

Once the torch position is satisfactory, press **CYCLE RESUME** and the cut will resume from the new position and travel the shortest distance to the original paused motion location. The CUT RECOVERY panel will close and the JOGGING panel will display when the torch returns to the original paused motion location.

Durch Drücken von **CANCEL MOVE** bewegt sich die Taschenlampe wieder dorthin, wo sie positioniert war, bevor die Richtungstasten verwendet wurden, um den Brenner zu versetzen. Es wird keine **REV** oder **FWD** Bewegung zurückgesetzt.

Pressing **CYCLE STOP** will cause the torch to move back to where it was positioned before the direction keys were used to offset the torch and the CUT RECOVERY panel overlay will return to the JOGGING panel. It will not reset any **REV** or **FWD** motion.

If an alignment laser has been set up then it is possible to use the laser during cut recovery for very accurate positioning of the new start coordinates. If either the X axis offset or Y axis offset for the laser would cause the machine to move out of bounds then an error message will be displayed.

To use a laser for cut recovery when paused during a cut:

1. Klicken Sie auf die Schaltfläche **LASER**.

2. **LASER** button will change to disabled, the HAL pin named qtplasmac.laser_on will be turned on and the X and Y axis will offset so that the laser cross hairs will indicate the starting coordinates of the cut when it is resumed.
3. Continue the cut recovery as described above.

If a laser offset is in effect when **CANCEL MOVE** is pressed then this offset will also be cleared.

Anmerkung
Cut recovery movements will be limited to a radius of 10mm (0.4”) from either the point the program was paused, or from the last point on the cut path if paused motion was used.

Wichtig
PUDDLE JUMP IS DISABLED DURING CUT RECOVERY

10.8.9.32 Run From Line

If the user has the Run From Line option enabled in the GUI SETTINGS section of the SETTINGS Tab they will have the ability to start from any line in a G-code program via the following methods:

1. Clicking any line in the Preview Window
2. Clicking any line in the G-code Window

Note that the Run From Line function will run from the beginning of the selected line.

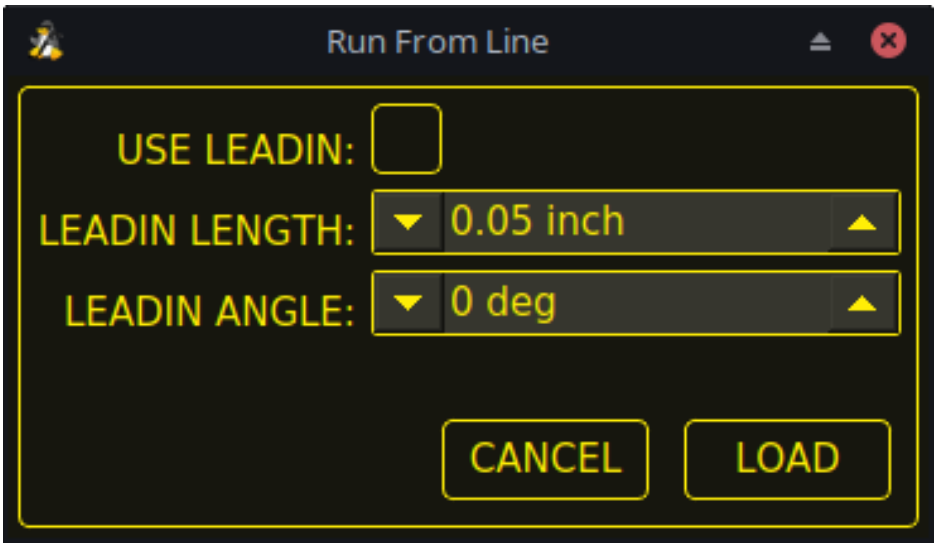
It is important to note that G-code programs can be run from any selected line using this method, however a leadin may not be possible depending on the line selected. In this case, an error message will be displayed to let the user know the leadin calculation was not possible.

Once the user has selected the starting place, the **CYCLE START** button will blink **”SELECTED nn”** where nn is the corresponding line number selected. Clicking this button will bring up the following Run From Line dialog box:

It is not possible to use Run From Line from within a subroutine. If the user selects a line within a subroutine and clicks **”SELECTED nn”** then an error message will be displayed that includes the O-Code name of the subroutine.

It is not possible to use Run From Line if previous G-code has set cutter compensation active. If the user selects a line while cutter compensation is active and clicks **”SELECTED nn”** then an error message will be displayed.

It is possible to select a new line while Run From Line is active.



Name	Beschreibung
USE LEADIN	This radio button will allow the user to start the selected line with a leadin.
LEADIN LENGTH	If USE LEADIN is selected, this will set the length of the lead in the machine units.
LEADIN ANGLE	If USE LEADIN is selected, this will set the angle of approach for the leadin. The angle is measured such that positive increases in value move the leadin counter-clockwise: 0 Degrees = 3 o'clock position 90 Degrees = 12 o'clock position 180 Degrees = 9 o'clock position 270 Degrees = 6 o'clock position
CANCEL	This button will cancel the Run From Line dialog box and any selections.
LOAD	This button will load a temporary "rfl.ngc" program with any selected leadin parameters applied. If the leadin cannot be calculated for the selected line, the following error message will be displayed: "Unable to calculate a leadin for this cut Program will run from selected line with no leadin applied"

After pressing **LOAD**, the blinking "SELECTED nn" button will change to **RUN FROM LINE CYCLE START** button. Click this button to start the program from the beginning of the selected line.

Run From Line selections may be canceled in the following ways:

1. Click the background of the preview window - this method will cancel a selection of either a cut line in the preview window, or a G-code line in the G-code window.
2. Click the text of the first line of the G-code program in the G-code display - this method will cancel a selection of either a cut line in the preview window, or a G-code line in the G-code window.
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.

10.8.9.33 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 **<Maschinen_name>** Konfigurationsverzeichnis. Dies wird getan, nachdem der Ritzer kann auf das Werkstück zu bewegen, um den entsprechenden Offset zu bestimmen.

The plasma torch offsets for X and Y will always be zero. The tools are selected by the **T*n* M6** command followed by a **G43 H0** command which is required to apply the offsets. The tool is then started with a **M3 $*n* S1** command. For *n*, use 0 for torch cutting or 1 for scribing.

Um den Ritzvorgang zu stoppen, verwenden Sie den G-Code-Befehl **M5 $1**.

If the user has not yet assigned the HAL pins for the scribe in the configuration wizard then they may do so by using the appropriate configuration wizard or by manually editing the HAL file, see modifying QtPlasmaC.

There are two HAL output pins used to operate the scribe, the first pin is used to arm the scribe which moves the scribe to the surface of the material. After the Arm Delay has elapsed, the second pin is used to start the scribe. After the On Delay has elapsed, motion will begin.

Using QtPlasmaC after enabling the scribe requires the selection of either the torch or the scribe in each G-code file as a LinuxCNC tool.

The first step is to set the offsets for the scribe by following the procedure described in Peripheral Offsets.

The final step is to set the scribe delays required:

1. **Arm Delay** - allows time for the scribe to descend to the surface of the material.
2. **On Delay** - allows time for the scribe to start before motion begins.

Save the parameters in the Config tab.

After the above directions are completed, the scribe may be tested manually by issuing a **M3 $1 S1** command in the MDI input. The user may find it helpful to use this method to scribe a small divot and then try to pulse the torch in the same location to align the offsets between the scribe and the torch.

To use the scribe from G-code:

```
...
M52 P1 (paused motion on)
F#<_hal[plasmac.cut-feed-rate]>
T1 M6 (select scribe)
G43 H0 (apply offsets for current tool)
M3 $1 S1 (start the scribe)
.
M5 $1 (stop the scribe)
.
T0 M6 (select torch)
G43 H0 (apply offsets for current tool)
G0 X0 Y0 (parking position)
M5 $-1 (end all)
```

It is a good idea to switch back to the torch at the end of the program before the final rapid parking move so the machine is always in the same state at idle.

The user can switch between the torch and the scribe any number of times during a program by using the appropriate G-codes.

Issuing **M3 S1** (without $*n*) will cause the machine to behave as if an **M3 $0 S1** had been issued and issuing **M5** (without $*n*) will cause the machine to behave as if an **M5 $0** had been issued. This will control the torch firing by default in order to provide backward compatibility for previous G-code files.

Warnung
If there is an existing manual tool change parameter set in the <*machine_name*>.hal file then QtPlasmaC will convert it to an automatic tool change.

10.8.9.34 Spotting

To achieve spotting to mark the material prior to drilling etc., QtPlasmaC can pulse the torch for a short duration to mark the spot to drill.

Spotting can be configured by following these steps:

1. Set the arc voltage **Threshold** in the Spotting section of the PARAMETERS Tab. Setting the voltage threshold to zero will cause the delay timer to begin immediately upon starting the torch. Setting the voltage threshold above zero will cause the delay timer to begin when the arc voltage reaches the threshold voltage.
2. Set the **Time On** in the Spotting section of the PARAMETERS Tab. When the **Time On** timer has elapsed, the torch will turn off. Times are adjustable from 0 to 9999 milliseconds.

The torch is then turned on in G-code with the **M3 $2 S1** command which selects the plasma torch as a spotting tool.

To turn the torch off, use the G-code command **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.

An example G-code is:

```
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
```

Anmerkung
Die **hohe Vorschubgeschwindigkeit** von 99999 soll sicherstellen, dass die Bewegung bei der höchsten Vorschubgeschwindigkeit der Maschine erfolgt.

Wichtig
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.

10.8.9.35 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:

Abbildung 10.47: Number keypad - used for the CONVERSATIONAL Tab and the PARAMETERS Tab

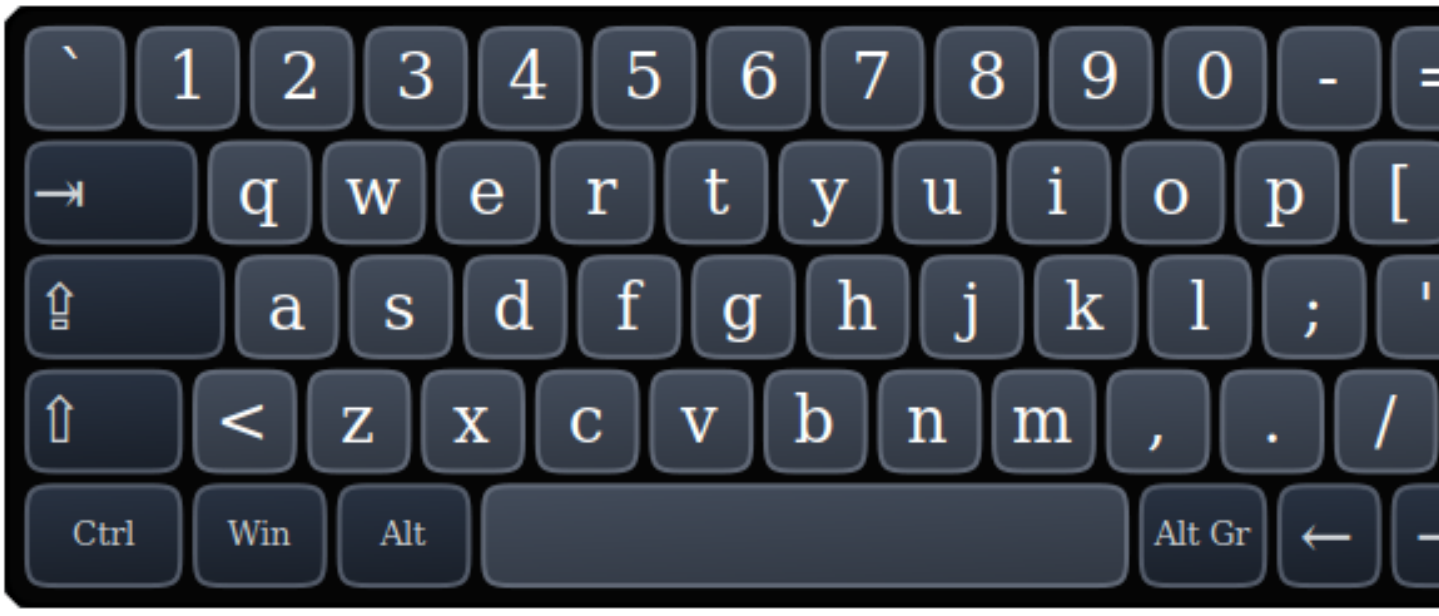


Abbildung 10.48: Alpha-numeric keypad - used for G-code editing and file management.

If the virtual keyboard has been repositioned and on the next opening of a virtual keyboard it is not visible then clicking twice on the onboard icon in the system tray will reposition the virtual keyboard so the move handle is visible.

10.8.9.36 Tastatürkürzel

Nachfolgend finden Sie eine Liste aller verfügbaren Tastaturkürzel in QtPlasmaC.

Anmerkung
Alle Tastaturkürzel sind standardmäßig deaktiviert.

In order to utilize them, **KB Shortcuts** must be enabled in the **GUI SETTINGS** section of the SETTINGS Tab.

Keyboard Shortcut	**Action**
Esc	Aborts current automated motion (example: a running program, a probe test, etc.) as well as an active torch pulse (behaves the same as clicking CYCLE STOP).
F1	Toggles the GUI E-STOP button (if the GUI E-STOP button is enabled).
F2	Toggles the GUI power button.
F9	Toggles the ”Cutting” command, used to begin or end a manual cut.
F12	Show stylesheet editor.
ALT+RETURN	Places QtPlasmaC into Manual Data Input (MDI) mode. Note that ALT + ENTER will achieve the same result. In addition, pressing RETURN (or ENTER) with no entry in the MDI will close the MDI window.
`, 1-9, 0	Changes jog speed to 0%, 10%-90%, 100% of the value present in the DEFAULT_LINEAR_VELOCITY variable in the **[DISPLAY]** section of the *<machine_name>*.ini file.
SHIFT+`, 1-9, 0	Changes rapid speed to 0%, 10%-90%, 100%.
CTRL+1-9, 0	Changes feed rate to 10%-90%, 100%.
CTRL+HOME	Homes all axes if they are not yet homed and have a homing sequence set in the *<machine_name>*.ini file. If they are already homed, they will no longer be homed.
CTRL+R	Cycle Start if the program is not already running. Cycle Resume if the program is paused.
END	Touches off X and Y to 0.
DEL	Allows the user to use a laser to set an origin with or without rotation. See the LASER section for detailed instructions.
SPACE BAR	Pauses motion.
CTRL+SPACE BAR	Clears notifications.
O	Opens a new program.
L	Loads the previously opened program if no program is loaded. Reloads the current program if there is a program loaded.
→	Jogs the X axis positive.
←	Jogs the X axis negative.
↑	Jogs the Y axis positive.
↓	Jogs the Y axis negative.
PAGE UP	Jogs the Z axis positive.
PAGE DOWN	Jogs the Z axis negative.
[	Jogs the A axis positive.
]	Jogs the A axis negative.

Keyboard Shortcut	Action
.	Jogs the B axis positive.
,	Jogs the B axis negative.
SHIFT (+ Jog Key)	The shift key is used with any jog key to invoke a rapid jog.
+ (+Jog Key)	The plus key can be used with any jog key to invoke a rapid jog (behaves the same as SHIFT).
- (+Jog Key)	The minus key can be used with any jog key to invoke a slow jog (10% of the displayed jog speed) If SLOW jogging is already active, the axis will jog at the displayed jog speed.

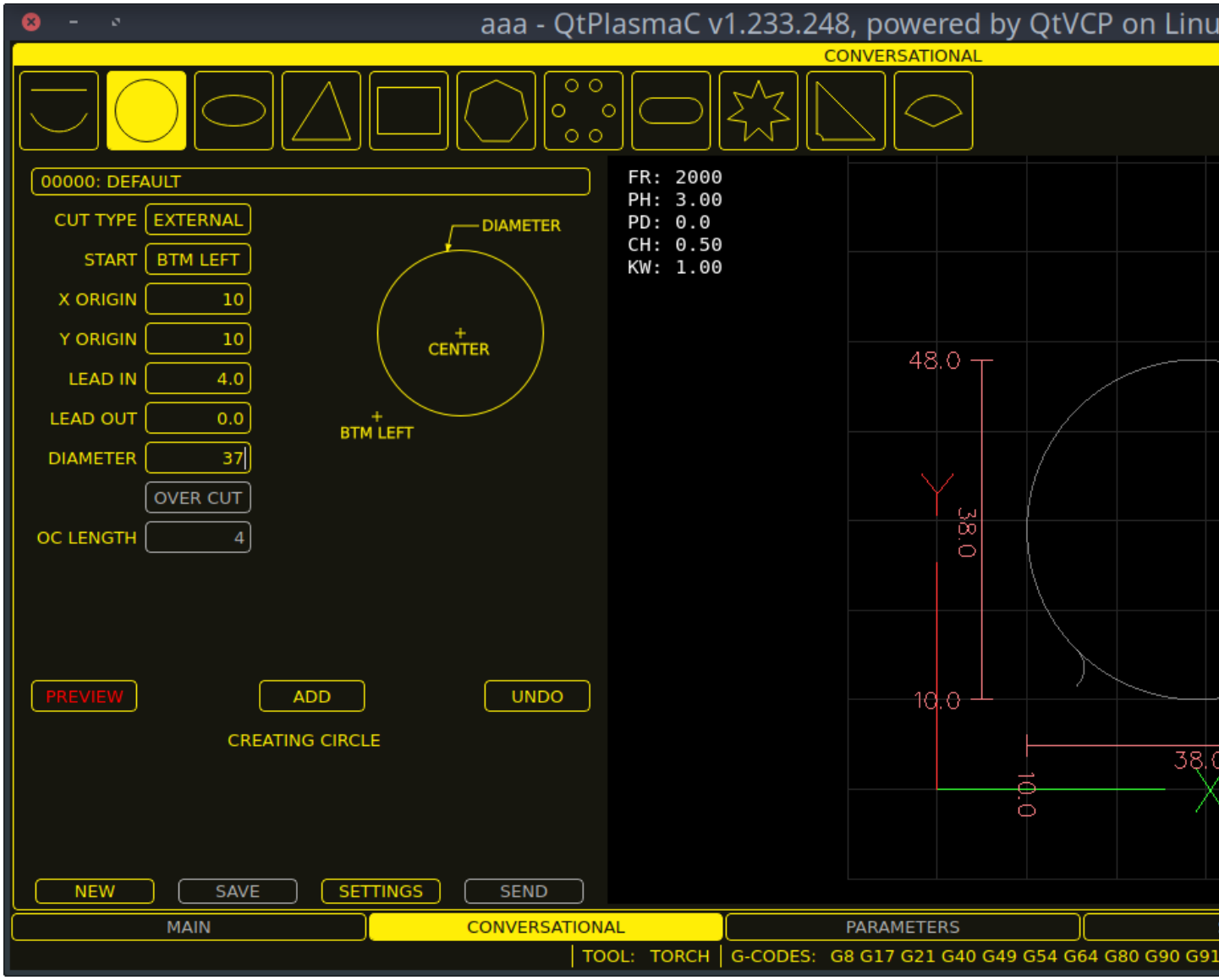
10.8.9.37 MDI

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: Abschnitt 12.7.2.21[MDILine Widget]

Anmerkung
M3, M4, and M5 are not allowed in the QtPlasmaC MDI.

In addition, pressing RETURN (or ENTER) with no entry in the MDI will close the MDI window.

10.8.10 Conversational Shape Library



The **Conversational Shape Library** consists of several basic shapes and functions to assist the user with generating quick G-code at the machine to cut simple shapes quickly. This feature is found on the CONVERSATIONAL Tab.

Anmerkung
The Conversational Library is not meant to be a CAD/CAM replacement as there are limitations to what can be achieved.

Blank entries in the shape input boxes will use the current setting at the time the G-code was generated. For example, if **X start** was left blank then the current X axis position would be used.

All leadins and leadouts are arcs except for **Circles** and **Stars**:

Circles:

- If the circle is external then any leadin or leadout will be an arc.

- If the circle is internal and a **small hole** then any leadin will be perpendicular and there will be no lead out.
- If the circle is internal and not a **small hole** then any leadin and leadout will be an arc. If the leadin has a length greater than half the radius then the leadin will revert to perpendicular and there will be no leadout. If the leadout has a length greater than half the radius then there will be no leadout.

Stars:

- The leadin is at the same angle as the first cut and the leadout is at the same angle as the last cut.

Anmerkung
A **small hole** is a circle that is smaller than the SMALL HOLE DIAMETER specified in the CONVERSATIONAL SETTINGS page.

Anmerkung
The holes in a BOLT CIRCLE shape will also abide by the above rules.

The cut order will occur in the same order as the shape was built.

Pressing **Return** on the keyboard while editing parameters will automatically show the preview of the shape if there are enough parameters entered to create the shape. Clicking any of the available check boxes will do the same.

The general functions are as follows:

Name	Beschreibung
Material Drop Down	Allows the user to select the desired material for cutting. If "VIEW MATERIAL" is selected on the SETTINGS Tab, a visual reference showing key material cut settings will be displayed on the Conversational Preview Window. Examples are: Feed Rate, Pierce Height, Pierce Delay, Cut Height, and Kerf Width (for Conversational only). Cut Amps will be shown if PowerMax communications are enabled.
NEW	Removes the current G-code file and load a blank G-code file.
SAVE	Opens a dialog box allowing the current shape to be saved as a G-code file.
SETTINGS	Allows the changing of the global settings.
SEND	Loads the current shape into LinuxCNC (QtPlasmaC). If the last edit was not added then it will be discarded.
PREVIEW	Displays a preview of the current shape provided the required information is present.
CONTINUE	This button is used for lines and arcs only. Allows another segment to be added to the current segment/segments.
ADD	Stores the current shape into the current job.
UNDO	Reverts to the previously stored state.
RELOAD	Reloads the original G-code file or a blank file if none was loaded.

If there is a G-code file loaded in LinuxCNC (QtPlasmaC) when the CONVERSATIONAL Tab is selected, that code will be imported into the conversational as the first shape of the job. If this code is not required then it can be removed by pressing the **NEW** button.

If there is an added shape that is unsaved or unsent then it is not possible to switch tabs in the GUI.

To re-enable switching tabs it is necessary to either **SAVE** the shape, **SEND** the shape, or press **NEW** to remove the shape.

Wenn **NEU** gedrückt wird, um eine hinzugefügte Form zu entfernen, die nicht gespeichert oder gesendet wurde, wird ein Warndialog angezeigt.

Anmerkung
Alle Entfernungen sind in Maschineneinheiten relativ zum aktuellen Benutzerkoordinatensystem und alle Winkel sind in Grad angegeben.

10.8.10.1 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))**
- **Leadin length**
- **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 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:

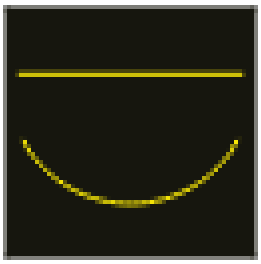
```
G21\nG40\nG49\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

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.

10.8.10.2 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:

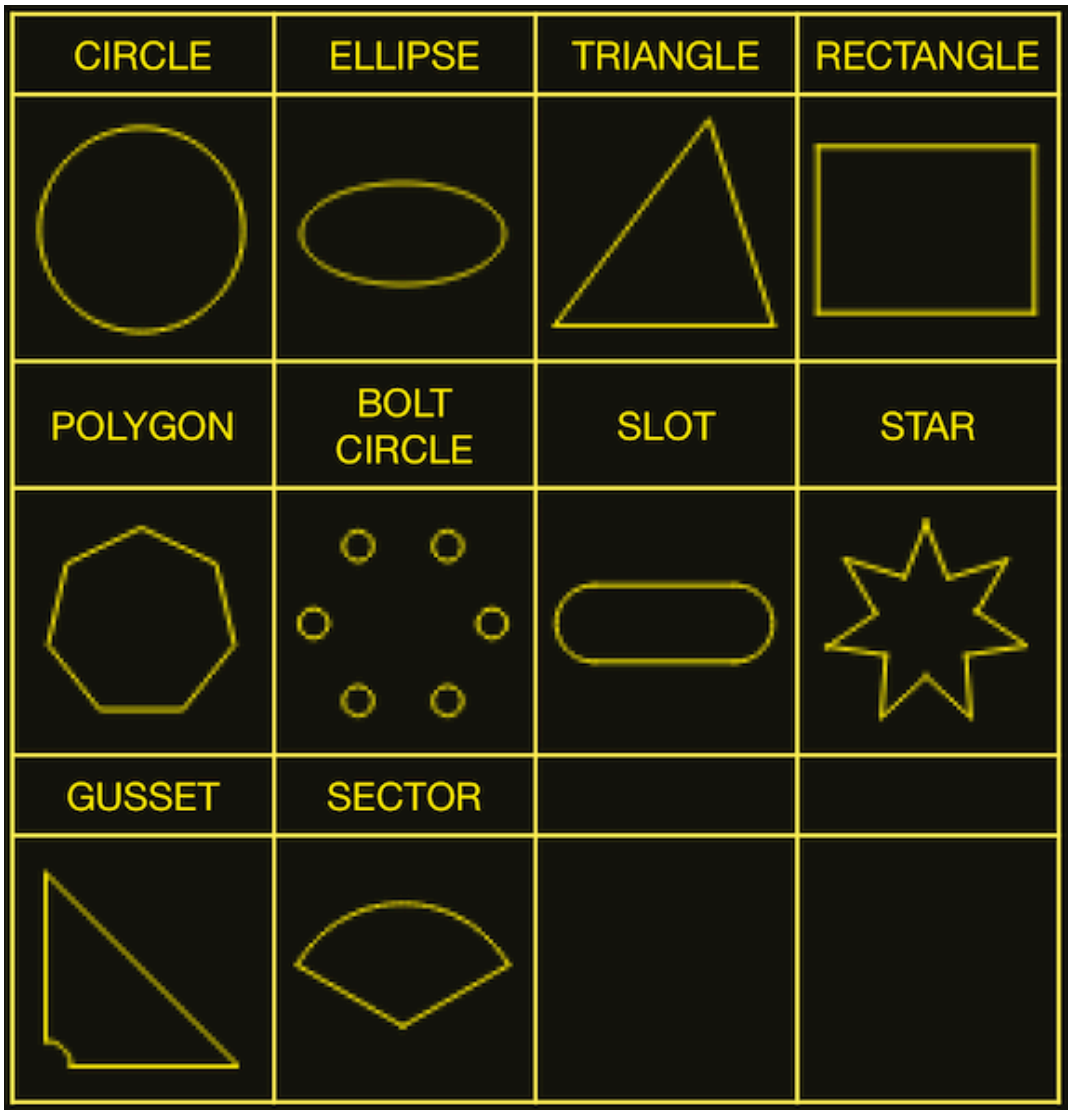
1. Wählen Sie das Symbol **Linien und Bögen** aus.
2. Wählen Sie den Typ der zu erstellenden Linie oder des Bogens.
3. Choose the material from the MATERIAL drop down. If no material is chosen, the default material (00000) will be used.
4. Geben Sie die gewünschten Parameter ein.
5. Drücken Sie **PREVIEW**, 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.

Anmerkung
In diesem Stadium gibt es keine automatische Option für die Erstellung eines Leadin/Leadout, wenn die Form geschlossen ist.

10.8.10.3 Conversational Single Shape

Die folgenden Formen sind für die Erstellung verfügbar:



So erstellen Sie eine Form:

1. Select the corresponding icon for the shape to create. The available parameters will be displayed.
2. Choose the material from the MATERIAL drop down. If no material is chosen, the default material (00000) will be used.
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.

For **CIRCLE**, the **OVER CUT** button will become valid when a CUT TYPE of INTERNAL is selected and the value entered in the DIAMETER field is less than the Small Hole Diameter parameter in the Conversational SETTINGS section.

For **BOLT CIRCLE** the **OVER CUT** button will become valid if the value entered in the HOLE DIA field is less than the SMALL HOLES DIAMETER parameter in the Conversational SETTINGS section.

For the following shapes, KERF OFFSET will become active once a LEAD IN is specified:

1. TRIANGLE
2. RECTANGLE
3. POLYGON
4. SLOT
5. STAR
6. GUSSET

10.8.10.4 Conversational Group Of Shapes

Multiple shapes can be added together to create a complex group.

The cut order of the group is determined by the order in which the individual shapes are added to the group.

Once a shape is added to the group it cannot be edited or removed.

Groups cannot have shapes removed, only added to.

To create a group of shapes:

1. Create the first shape as in **Single Shape**.
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.

10.8.10.5 Conversational Block

Die Funktion ”Conversational Block” ermöglicht die Durchführung von Blockoperationen mit der aktuellen Form oder einer Gruppe von Formen, die im CONVERSATIONAL Registrierkarte 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.

Blockoperationen:

- Rotate
- 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.

   COLUMNS & ROWS
   specifies the number of duplicates of the original shape arranged in columns and rows as well as the offset distance from the original shape.

   ORIGIN
   offset the result from the origin coordinates.

   ANGLE
   rotate the result.

   SCALE
   scale the result.

   ROTATION
   rotate the shape within the result.

   MIRROR
   mirror the shape about its X coordinates within the result.

   FLIP
   flip the shape about its Y coordinates within the result.

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.

10.8.10.6 Conversational Saving A Job

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 Tab verlassen hat, kann der Benutzer die G-Code Datei immer noch von der GUI aus speichern. Alternativ kann der Benutzer auf den CONVERSATIONAL Tab klicken, wodurch der Job neu geladen wird, woraufhin er die Schaltfläche **SAVE** drücken kann.

10.8.11 Fehlermeldungen

10.8.11.1 Fehlerprotokollierung

All errors are logged into the machine log which is able to be viewed in the STATISTICS Tab. The log file is saved into the configuration directory when QtPlasmaC is shutdown. The five last logfiles are kept, after which the oldest logfile is deleted each time a new log file is created. These saved log files may be viewed with any text editor.

10.8.11.2 Error Message Display

By default, QtPlasmaC will display error messages via a Operator Error popup window. In addition, QtPlasmaC will alert the user that an error has been sent to the machine log by displaying the message **"ERROR SENT TO MACHINE LOG"** in the lower left portion of the status bar.

The user may opt to disable the Operator Error popup window, and view the error messages by going to the STATISTICS Tab by changing the following preference to **False** in the **[SCREEN_OPTIONS]** of the <*machine_name*>.prefs file in the <machine_name> directory:

```
desktop_notify
```

Anmerkung
<*machine_name*>.prefs must be edited with QtPlasmaC closed or any changes will be overwritten on exit.

Additionally, it is possible for **ERROR SENT TO MACHINE LOG** to flash to get the user's attention by adding or editing the following line in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Flash error = True
```

10.8.11.3 Critical Errors

There are a number of error messages printed by QtPlasmaC to inform the user of faults as they occur. The messages can be split into two groups, **Critical** and **Warning**.

Critical Errors will cause the running program to pause, and the operator will need to clear the cause of the error before proceeding.

If the error was received during cutting then forward or reverse motion is allowed while the machine is paused to enable the user to reposition the machine prior to resuming the cut.

When the error is cleared the program may be resumed.

These errors indicate the corresponding sensor was activated during cutting:

- **breakaway switch activated program is paused**
- **float switch activated program is paused**
- **ohmic probe activated program is paused**

These errors indicate the corresponding sensor was activated before probing commenced:

- **ohmic probe detected before probing program is paused**
- **float switch detected before probing program is paused**
- **breakaway switch detected before probing program is paused**

The Arc OK signal was lost during cutting motion, before the **M5** command was reached:

- **valid arc lost program is paused**

The Z axis reached the bottom limit before the work piece was detected:

- **bottom limit reached while probing down program is paused**

The work piece is too high for any safe rapid removes:

- **material too high for safe traverse program is paused**

One of these values in MATERIAL section of the PARAMETERS Tab is invalid (For example: if they are set to zero):

- **invalid pierce height or invalid cut height or invalid cut volts program is paused**

No arc has been detected after attempting to start the number of times indicated by **Max Starts** in the ARC frame of the CONFIGURATION section of the PARAMETERS Tab:

- **no arc detected after <*n*>d start attempts program is paused**
- **no arc detected after <*n*>d start attempts manual cut is stopped**

THC has caused the bottom limit to be reached while cutting:

- **bottom limit reached while THC moving down program is paused**

THC has caused the top limit to be reached while cutting:

- **top limit reached while THC moving up program is paused**

These errors indicate move to pierce height would exceed the Z Axis MAX_LIMIT for the corresponding probe method:

- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during float switch probing**
- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during ohmic probing**

These errors indicate the move to pierce height would exceed the Z axis maximum safe height for the corresponding probe method:

- **pierce height would exceed Z axis maximum safe height condition found while float switch probing**
- **pierce height would exceed Z axis maximum safe height condition found while ohmic probing**

10.8.11.4 Warning Errors

Warning errors will not pause a running program and are informational only.

These errors indicate the corresponding sensor was activated before a probe test commenced:

- **ohmic probe detected before probing probe test aborted**
- **float switch detected before probing probe test aborted**
- **breakaway switch detected before probing probe test aborted**

This indicates that the corresponding sensor was activated during a consumable change:

- **breakaway, float, or ohmic activated during consumable change, motion is paused WARNING: MOTION WILL RESUME IMMEDIATELY UPON RESOLVING THIS CONDITION!**

Warnung
CONSUMABLE CHANGE MOTION WILL RESUME IMMEDIATELY UPON RESOLVING THE CORRESPONDING SENSOR ACTIVATION.

This indicates that the corresponding sensor was activated during probe testing:

- **breakaway switch detected during probe test**

This indicates that probe contact was lost before probing up to find the zero point:

- **probe trip error while probing**

This indicates that the bottom limit was reached during a probe test:

- **bottom limit reached while probe testing**

This indicates that the move to pierce height would exceed the Z Axis MAX_LIMIT during the corresponding probe method:

- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during float switch probe testing**
- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during ohmic probe testing**

Dies zeigt an, dass die sichere Höhe reduziert wurde, weil THC die Z-Achse während des Schneidens anhebt:

- **sichere Verfahrhöhe wurde reduziert**.

This indicates that the value for the Arc Voltage was invalid (NAN or INF) when QtPlasmaC launched.

- **invalid arc-voltage-in**

10.8.12 Updating QtPlasmaC

10.8.12.1 Standard Update

QtPlasmaC update notices are posted at: https://forum.linuxcnc.org/plasmac/37233-plasmac-updates

Users are strongly encouraged to create a Username and subscribe to the above thread to receive update notices.

For a standard ISO installation, LinuxCNC will only be updated when a new minor release has been released. QtPlasmaC will then automatically update its configuration the first time it is run after a LinuxCNC update.

LinuxCNC is normally updated by entering the following commands into a terminal window (one at a time):

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

10.8.12.2 Continuous Update

Enhancements and bug fixes will not be available on a standard installation until a new minor release of LinuxCNC has been released. If the user wishes to update whenever a new QtPlasmaC version has been pushed, they could use the LinuxCNC Buildbot repository rather than the standard LinuxCNC repository by following the instructions at: http://buildbot.linuxcnc.org/

10.8.13 Modify An Existing QtPlasmaC Configuration

There are two ways to modify an existing QtPlasmaC configuration:

1. Running the appropriate configuration wizard and loading the conf file saved by the wizard.
2. Manually edit the INI and/or the HAL file of the configuration.

Wichtig
Any manual modification to the <*machine_name*>.ini and <*machine_name*>.hal files will not be registered in PnCconf or StepConf.

Anmerkung
If unsure of the HAL pin's full name, the user may start LinuxCNC and run **HalShow** for a full listing of all HAL pins.

10.8.14 Customizing QtPlasmaC GUI

Styling of the QtPlasmaC GUI is done with Qt stylesheets and some customization may be achieved by the use of a custom stylesheet. This allows the user to change some GUI items such as color, border, size, etc. It cannot change the layout of the GUI.

Information on Qt stylesheets is available here.

There are two methods available to apply custom styles:

1. Add A Custom Style: use this for minor style changes.
2. Create A New Style use this for a complete style change.

10.8.14.1 Add A Custom Style

Adding style changes to the default stylesheet is achieved by creating a file in the <machine_name> configuration directory. This file MUST be named qtplasmac_custom.qss. Any required style changes are then added to this file.

For example the user may want the arc voltage display in red, a green Torch On LED of a larger size and a larger Torch Enable button. This would be done with the following code in qtplasmac_custom.qss:

```
#arc_voltage {
    color: #ff0000 }

#led_torch_on {
    qproperty-diameter: 30;
    qproperty-color: green }

#torch_enable::indicator {
    width: 30;
    height: 30}
```

10.8.14.2 Create A New Style

Custom stylesheets are enable by a setting in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file giving the filename of the stylesheet.

```
Custom style = the_cool_style.qss
```

The filename may be any valid filename. The standard extension name is .qss but this is not mandatory.

There are some constraints on the custom stylesheet for QtPlasmaC, e.g., the jog buttons, cut-recovery buttons, and the conversational shape buttons are image files and are not able to be custom styled.

The custom style file requires a header in the following format:

```
/*****************************
Custom Stylesheet Header

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

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

The colors may be expressed in any valid stylesheet format.

The above colors are used for the following widgets. So any custom styling will need to take these into account. The colors shown below are the defaults used in QtPlasmaC along with the color name from the SETTINGS Tab.

Farbe	Parameter	Affects
color1 (#ffee06)	Foreground	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)	Background	background of latching user buttons background of camera/laser buttons background of G-code editor active line background of conversational shape buttons
color3 (#ffee06)	Highlight	background of active latching user buttons background of active camera/laser buttons foreground of G-code editor cursor
color4 (#36362e)	Alt Background	Hintergrund der aktiven Zeile der G-Code-Anzeige

10.8.14.3 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. Open <*machine_name*>.prefs file.
5. Delete the **[COLOR_OPTIONS]** section.
6. Delete the Custom style line from the **[GUI_OPTIONS]** section.
7. Save the file.

Beim nächsten Laden von QtPlasmaC wird das gesamte benutzerdefinierte Styling entfernt und das Standard-Styling wird wiederhergestellt.

Below is an example of the section to be deleted from <*machine_name*>.prefs:

```
[COLOR_OPTIONS]
Foreground = #ffee06
Highlight = #ffee06
LED = #ffee06
Background = #16160e
Background Alt = #36362e
Frames = #ffee06
Estop = #ff0000
Disabled = #b0b0b0
Preview = #000000
```

10.8.14.4 Custom Python Code

It is possible to add custom python code to change some existing functions or to add new ones. Custom code can be added in two different way, a user command file or a user periodic file.

A user command file is specified in the DISPLAY section of the <*machine_name*>.ini file and contains python code that is processed during startup.

```
n USER_COMMAND_FILE = my_custom_code.py
```

A user periodic file must be named user_periodic.py and must be loated in the machines config directory. This file is processed every cycle (usually 100 ms) and is used for functions that require regular updating.

10.8.15 QtPlasmaC Fortgeschrittene Themen

10.8.15.1 Benutzerdefinierte Buttons

The QtPlasmaC GUI offers user buttons that can be customized by adding commands in the USER BUTTON ENTRIES section of the SETTINGS Tab in the <*machine_name*>.prefs file.

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.

All <*machine_name*>.prefs file settings for the buttons are found in the **[BUTTONS]** section.

Button-Namen Der Text, der auf einem Button erscheint, wird auf folgende Weise festgelegt:

```
n Name = HAL Show
```

Where *n* is the button number and **HAL Show** is the text.

For text on multiple lines, split the text with a \ (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 scripts
3. G-code Befehle
4. Umschalten eines HAL-Pins
5. Toggle the alignment laser HAL pin
6. Pulse a HAL pin
7. Probe test
8. Ohmic Test
9. Cut Type
10. Change consumables

11. Ein G-Code-Programm laden
12. Pulse the torch on
13. Single unidirectional cut
14. Framing a job
15. Begin/End a manual cut
16. Anzeige/Verstecken eines Offsets-Viewers
17. Load the latest modified NGC file found in a directory

External Commands

To run an external command, the command is preceded by a % character.

```
n Code = %halshow
```

External Python Scripts

To run an external python script, the script name is preceded by a % character and it also requires a .py extension. It is valid to use the ~ character as a shortcut for the users home directory.

```
n Code = %my_python_script.py
```

G-code

To run G-code, just enter the code to be run.

```
n Code = G0 X100
```

To run an existing subroutine.

```
n Code = o<the_subroutine> call
```

<*machine_name*>.ini file variables can be entered by using {} (a space must be placed after the })

```
n Code = G0 X{JOINT_0 HOME} Y1
n Code = G53 G0 Z[{AXIS_Z MAX_LIMIT} - 1.001]
```

Multiple codes can be run by separating the codes with a "\" (backslash) character. The exception is the special commands which are required to be a single command per button.

```
n Code = G0 X0 Y0 \ G1 X5 \ G1 Y5
```

External commands and G-code may be mixed on the same button.

```
n Code = %halshow \ g0x.5y.5 \ %halmeter
```

Toggle HAL Pin

The following code will allow the user to use a button to invert the current state of a HAL bit pin:

```
n Code = toggle-halpin the-hal-pin-name
```

This code is required to be used as a single command and may only control one HAL bit pin per button.

The button colors will follow the state of the HAL pin.

After setting the code, upon clicking, the button will invert colors and the HAL pin will invert pin state. The button will stay "latched" until the button is clicked again, which will return the button to the original colors and the HAL pin to the original pin state.

There are three External HAL Pins that are available to toggle as an output, the pin names are `qtplasmac.ext_out_0`, `qtplasmac.ext_out_1`, and `qtplasmac.ext_out_2`. HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmac GUI has loaded.

It is possible for the user to mark a toggle-halpin button's associated HAL pin as being required to be turned "ON" before **CYCLE START** can be pressed by adding "runcritical" after the HAL pin in the button code.

```
n Code = toggle-halpin the-hal-pin-name runcritical
```

Toggle Alignment Laser HAL Pin

The following code will allow the user to use a button to invert the current state of the alignment laser HAL bit pin:

```
n Code = toggle-laser
```

This code is also able to be used as a multiple command with G-code or external commands but may control only the alignment laser HAL bit pin.

The button colors will follow the state of the alignment laser HAL pin.

After setting the code, upon clicking, the button will invert colors and the alignment laser HAL pin will invert pin state. The button will stay "latched" until the button is clicked again, which will return the button to the original colors and the alignment laser HAL pin to the original pin state.

The following code would allow the user to use a button to invert the current state of the alignment laser HAL bit pin and then move the X and Y axes to the offset for the alignment laser as specified in the <*machine_name*>.prefs file:

```
n Code = G0 {QTPLASMAC LASER_TOUCHOFF} \ toggle-laser
```

The position of the "toggle-laser" command is not important as it is always the first command actioned regardless of position.

Pulse HAL Pin

The following code will allow the user to use a button to pulse a HAL bit pin for a duration of 0.5 seconds:

```
n Code = pulse-halpin the-hal-pin-name 0.5
```

This code is required to be used as a single command and may only control one HAL bit pin per button.

The pulse duration is specified in seconds, if the pulse duration is not specified then it will default to one second.

The button colors will follow the state of the HAL pin.

After setting the code, upon clicking the button, the button will invert colors, the HAL pin will invert pin state, and the time remaining will be displayed on the button. The button color and the pin state will stay inverted until the pulse duration timer has completed, which will return the button to the original colors, the HAL pin to the original pin state, and the original button name.

An active pulse can be canceled by clicking the button again.

There are three External HAL Pins that are available to pulse as an output, the pin names are qtplasmac.ext_out_0, qtplasmac.ext_out_1, and qtplasmac.ext_out_2. HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmac GUI has loaded.

Sonden-Test

QtPlasmaC will begin a probe and when the material is detected, the Z axis will rise to the Pierce Height currently displayed in the MATERIAL section of the PARAMETERS Tab. If the user has "View

Material” selected in the GUI SETTINGS section of the SETTINGS Tab, this value will be displayed in the top left corner of the PREVIEW Window next to **PH:**.

QtPlasmaC will then wait in this state for the time specified (rounded to no decimal places) before returning the Z axis to the starting position. An example of a 6 second delay is below. If there is no time specified then the probe time will default to 10 seconds.

```
n Code = probe-test 6
```

Anmerkung
Enabling a user button as a Probe Test button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmac GUI has loaded.

Ohmscher Test

QtPlasmaC will enable the Ohmic Probe Enable output signal and if the Ohmic Probe input is sensed, the LED indicator in the SENSOR Panel will light. The main purpose of this is to allow a quick test for a shorted torch tip.

```
n Code = ohmic-test
```

Anmerkung
Enabling a user button as an Ohmic Test button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmac GUI has loaded.

Cut Type

This button if selected will toggle between the two cut types, Pierce and Cut (default cutting mode) or Pierce Only.

```
n Code = cut-type
```

Verbrauchsmaterialien wechseln

Pressing this button moves the torch to the specified coordinates when the machine is paused to allow the user easy access to change the torch consumables.

Valid entries are Xnnn Ynnn Fnnn. Feed Rate (F) is mandatory and at least one of the X or Y coordinates are required.

The X and Y coordinates are in absolute machine coordinates. If X or Y are missing then the current coordinate for that axis will be used.

There are three methods to return to the previous coordinates:

1. Press the **Change Consumables** button again - the torch will return to the original coordinates and the machine will wait in this position for the user to resume the program.
2. Press **CYCLE RESUME** - the torch to return to the original coordinates and the program will resume.
3. Press **CYCLE STOP** - the torch to return to the original coordinates and the program will abort.

```
n Code = change-consumables X10 Y10 F1000
```

Anmerkung
Enabling a user button as a Change Consumables button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmac GUI has loaded.

Load

Loading a G-code program from the directory specified by the **PROGRAM_PREFIX** variable in the <*machine_name*>.ini file (usually ~/linuxcnc/nc_files) is possible by using the following format:

```
n Code = load G-code.ngc
```

If the user's G-code file is located in a sub-directory of the **PROGRAM_PREFIX** directory, it would be accessed by adding the sub-directory name to the beginning of the G-code file name. Example for a sub-directory named **plasma**:

```
n Code = load plasma/G-code.ngc
```

Note that the first "/" is not necessary as it will be added automatically.

Brenner-Puls

Pulse the torch on for a predetermined time. The time must be specified in seconds using up to one decimal place. The maximum allowable time is 3 seconds, anything specified above that value will be limited to 3 seconds. An example of a 0.5 second pulse is below. It there is no time specified then it will default to 1 second. Pulse times with more than one decimal place will be rounded to one decimal place.

Pressing the button again during the countdown will cause the torch to be turned off, as will pressing Esc if keyboard shortcuts are enabled in the SETTINGS Tab.

If the button is released before the countdown is complete then the torch will turn off at countdown completion, holding the button on until after the countdown has completed will cause the torch to remain on until the button has been released.

```
n Code = torch-pulse 0.5
```

Anmerkung
Enabling a user button as a Torch Pulse button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmac GUI has loaded.

Single Cut

Run a single unidirectional cut. This utilises the automatic Single Cut feature.

```
n Code = single-cut
```

Framing

Framing is the ability to move the torch around the perimeter of a rectangle that encompasses the bounds of the current job.

The laser enable HAL pin (qtplasmac.laser_on) will be turned on during the framing moves and any X/Y offsets for the laser pointer in the <*machine_name*>.prefs file will also be applied to the X/Y motion. After the framing motion is completed, the torch will move to the X0 Y0 position to clear any applied laser offsets and qtplasmac.laser_on will be turned off.

Upon starting a Framing cycle, it is important to note that by default the Z axis will be moved to a height of [AXIS_Z]MAX_LIMIT - 5mm (0.2”) before X/Y motion begins.

The velocity for the XY movements of the Framing motion can be specified so that Framing motion always occurs at a set velocity. This can be achieved by adding the feed rate (F) as the as the last portion of the button code. If the feed rate is omitted from the button code, framing motion velocity will default to the feed rate for the currently selected material.

The following GUI buttons and Keyboard Shortcuts (if enabled in the SETTINGS Tab) are valid during Framing motion:

1. Pressing **CYCLE STOP** or the ESC keyboard shortcut - Stops Framing motion.
2. Pressing **CYCLE PAUSE** or the SPACE BAR keyboard shortcut- Pauses Framing motion.
3. Pressing **CYCLE RESUME** or the CTRL+r keyboard shortcut- Resumes paused Framing motion.
4. Changing the **FEED SLIDER** or any of the CTRL+0-9 keyboard shortcuts - Slows the feed rate.

Anmerkung
IF THE FEED RATE IS CHANGED FOR THE FRAMING MOTION, IT WILL BE NECESSARY TO RETURN THE FEED SLIDER TO 100% BEFORE PRESSING CYCLE START AND CUTTING THE LOADED JOB.

```
n Code = framing
```

It is possible for the user to omit the initial default Z movement and run the framing sequence at the current Z height by adding ”usecurrentzheight” after ”framing”.

```
n Code = framing usecurrentzheight
```

To specify a feed rate:

```
n Code = framing F100
```

oder:

```
n Code = framing usecurrentzheight F100
```

Enabling a user button as a framing button will add an external HAL pin that may be connected from a pendant etc. HAL connections to this HAL pin needs to be specified in a postgui HAL file as the HAL pin is not available until the QtPlasmac GUI has loaded.

Manual Cut

Manual Cut functions identically to the **F9** button to begin or end a manual cut.

```
n Code = manual-cut
```

Offset Viewer

This allows the showing/hiding of an offset viewing screen that displays all machine offsets. All relative offsets can be edited and the G54 ~ G59.3 work system coordinates are able to be given custom names.

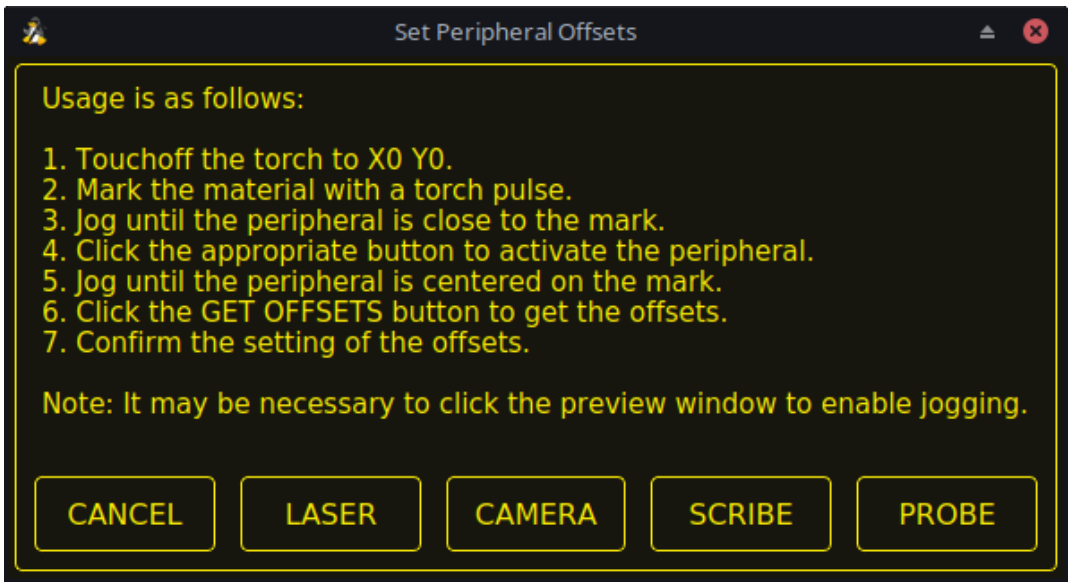
```
n Code = offsets-view
```

Load Latest File

this allows the loading of the last modified file in a directory. The directory name is optional and if omitted will default to the last directory a file was loaded from.

```
n Code = latest-file /home/me/linuxcnc/nc_files/qtplasmac-test
```

10.8.15.2 Peripheral Offsets (Laser, Camera, Scribe, Offset Probe)



Use the following sequence to set the offsets for a laser, camera, scribe, or offset probe:

1. Place a piece of scrap material under the torch.
2. Die Maschine muss referenziert und im Leerlauf sein, bevor Sie fortfahren.
3. Open the SETTINGS Tab.
4. Click the SET OFFSETS button which opens the Set Peripheral Offsets dialog.
5. Click the X0Y0 button to set the torch position to zero.
6. Make a mark on the material by one of:
   a. Jog the torch down to pierce height then pulse the torch on to make a dimple in the material.
   b. Place marking dye on the torch shield then jog the torch down to mark the material.
7. Raise the Z axis so the torch and peripheral are clear of the material
8. Jog the X/Y axes so that the peripheral is close to the mark from the torch.
9. Click the appropriate button to activate the peripheral.
10. Jog the X/Y axes so that the peripheral is centered in the mark from the torch.
11. Click the GET OFFSETS button to get the offsets and a confirmation dialog will open.
12. Click SET OFFSETS and the offsets will now be saved.

Canceling may be done at any stage by pressing the CANCEL button which will close the dialog and no changes will be saved.

Anmerkung
It may be necessary to click the preview window to enable jogging. By following the above procedure the offsets are available for use immediately and no restart of LinuxCNC is required.

10.8.15.3 Keep Z Motion

By default, QtPlasmaC will remove all Z motion from a loaded G-code file and add an initial Z movement to bring the torch near the top of travel at the beginning of the file. If the user wishes to use their table with a marker, a drag knife, diamond scribe, etc. mounted in the torch holder, QtPlasmaC has the ability to retain the Z movements when executing a program by adding the following command in a G-code file:

```
#<keep-z-motion> = 1
```

Omitting this command, or setting this value to anything but 1 will cause QtPlasmaC to revert to the default behavior of stripping all Z motion from a loaded G-code file and making an initial Z movement to bring the torch near the top of travel at the beginning of the file.

10.8.15.4 Externe HAL-Pins

QtPlasmaC erstellt einige HAL-Pins, die für den Anschluss eines externen Tasters oder einer Fernbedienung usw. verwendet werden können.

HAL connections to these HAL pins need to be specified in a postgui HAL file as the HAL pins are not available until the QtPlasmac GUI has loaded.

Die folgenden HAL-Bit-Pins werden immer erzeugt. Der HAL-Pin hat das gleiche Verhalten wie der zugehörige QtPlasmaC GUI-Button.

User Button Function	**HAL Pin**	**GUI Function**
Maschinenleistung umschalten	`qtplasmac.ext_power`	POWER (engl. für Leistung oder Strom)
Run the loaded G-code program	`qtplasmac.ext_run`	ZYKLUSSTART
Pause/Resume the loaded G-code program	`qtplasmac.ext_pause`	ZYKLUSPAUSE
Abort the loaded G-code program	`qtplasmac.ext_abort`	ZYKLUS STOP (engl. cycle stop)
Touchoff X & Y axes to zero	`qtplasmac.ext_touchoff`	TOUCHOFF X0Y0
Use a laser to set an origin with or without rotation	`qtplasmac.ext_laser_touchoff`	LASER
Run/Pause/Resume the loaded G-code program	`qtplasmac.ext_run_pause`	CYCLE START, CYCLE PAUSE, CYCLE RESUME in sequence
Torch height override plus	`qtplasmac.ext_height_ovr_plus`	OVERRIDE +
Torch height override 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`	n/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
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
Toggle mesh mode	`qtplasmac.ext_mesh_mode`	MESH MODE
Toggle arc ignore OK	`qtplasmac.ext_ignore_arc_ok`	IGNORE OK

User Button Function	HAL Pin	GUI Function
Vorwärts entlang des programmierten Pfades	qtplasmac.ext_cutrec_fwd	CUT RECOVERY FWD
Rückwärts entlang des programmierten Pfads	qtplasmac.ext_cutrec_rev	CUT RECOVERY REV
Cancel any Cut Recovery movement	qtplasmac.ext_cutrec_cancel	CUT RECOVERY CANCEL MOVE
Move up	qtplasmac.ext_cutrec_n	CUT RECOVERY arrow up
Move down	qtplasmac.ext_cutrec_s	CUT RECOVERY arrow down
Move right	qtplasmac.ext_cutrec_e	CUT RECOVERY arrow right
Move left	qtplasmac.ext_cutrec_w	CUT RECOVERY arrow left
Move up-right	qtplasmac.ext_cutrec_ne	CUT RECOVERY arrow up-right
Move up-left	qtplasmac.ext_cutrec_nw	CUT RECOVERY arrow up-left
Move down-right	qtplasmac.ext_cutrec_se	CUT RECOVERY arrow down-right
Move down-left	qtplasmac.ext_cutrec_sw	CUT RECOVERY arrow down-left

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.

User Button Function	HAL Pin
Sonden-Test	qtplasmac.ext_probe
Brenner-Puls	qtplasmac.ext_pulse
Ohmscher Test	qtplasmac.ext_ohmic
Verbrauchsmaterialien wechseln	qtplasmac.ext_consumables
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

10.8.15.5 Programm-Buttons ausblenden

If the user has external buttons and/or a pendant that emulates any of the program buttons, CYCLE START, CYCLE PAUSE, or CYCLE STOP then it is possible to hide any or all of these GUI program buttons by adding the following code to the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Hide run = True
```

```
Hide pause = True
Hide abort = True
```

For the 16:9 or 4:3 GUIs, the hiding of each of these GUI buttons will expose two more custom user buttons in the GUI.

10.8.15.6 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.

There are two settings in the PARAMETERS Tab for setting the range, these are:

- **OK High Volts** which is the upper value of the voltage range. The default is 250 V.
- **OK Low Volts** which is the lower value of the voltage range. The default is 60 V.

Both of these values may be changed by direct entry or by the use of the increment/decrement buttons.

There are also two HAL pins that have been provided to allow the user to tune the set point. These HAL pins are:

- `plasmac.arc-ok-counts` which is the number of consecutive readings within the threshold that are required to set the Arc OK signal. The default is 10.
- `plasmac.arc-ok-threshold` which is the maximum voltage deviation that is allowed for a valid voltage to set the Arc OK signal. The default is 10.

The following example would set the number of valid consecutive readings required to 6:

```
setp plasmac.arc-ok-counts 6
```

These settings if used should be in the custom.hal file of the configuration.

10.8.15.7 Lost Arc Delay

Some plasma power sources/machine configurations may lose the Arc OK signal either momentarily during a cut, or permanently near the end of a cut causing QtPlasmaC to pause the program and report a ”valid arc lost” error.

There is a HAL pin named `plasmac.arc-lost-delay` that may be used to set a delay (in seconds) that will prevent a paused program/error if the lost Arc OK signal is regained, or the **M5** command is reached before the set delay period expires.

It is important to note that the THC will be disabled and locked at the cutting height at the time the Arc OK signal was lost.

The following code would set a delay of 0.1 seconds:

```
setp plasmac.arc-lost-delay 0.1
```

It is recommended that the user set this pin in the custom.hal file.

This setting should only be used if the user experiences the above symptoms. It should also be noted that the user could use the appropriate Ignore Arc OK G-code commands to achieve a similar result.

10.8.15.8 Zero Window

Small fluctuations in the arc voltage displayed while the machine is at idle are possible depending on many different variables (electrical noise, incorrect THCAD tuning, etc.).

After all contributing factors have been mitigated, if a small fluctuation still exists it is possible to eliminate it by widening the voltage window for which QtPlasmaC will display 0 V.

The pin for adjusting this value is named plasmac.zero-window and the default value is set to 0.1. To change this value, add the pin and the required value to the custom.hal file.

The following example would set the voltage window to be displayed as 0 V from -5 V to +5 V:

```
setp plasmac.zero-window 5
```

10.8.15.9 Tuning Void Sensing

In addition to the **Void Slope** setting in the PARAMETERS Tab there are two HAL pins to aid in the fine tuning of void anti-dive. These HAL pins are:

- **plasmac.void-on-cycles** which is the number of times the slope rate needs to be exceeded to activate void anti-dive. The default is 2.
- **plasmac.void-off-cycles** which is the number of cycles without the slope rate being exceeded to deactivate void anti-dive. The default is 10.

The following example would set the number of on cycles required to 3:

```
setp plasmac.void-on-cycles 3
```

The objective is to have as low a value of Void Slope as possible without any false triggering then adjust on and off cycles to ensure clean activation and deactivation of void anti-dive. In most cases it should not be necessary to change on and off cycles from the default value.

These settings if used should be in the custom.hal file of the configuration.

10.8.15.10 Max Offset

Max Offset is the distance (in millimeters) away from the Z MAX_LIMIT that QtPlasmaC will allow the Z axis to travel while under machine control.

The pin for adjusting this value is named plasmac.max-offset and the default value (in millimeters) is set to 5. To change this value, add the pin and the required value to the custom.hal file. It is not recommended to use values less than 5mm as offset overrun may cause unforeseen issues.

The following example would set the distance from Z MAX_LIMIT to 10 mm:

```
setp plasmac.max-offset 10
```

10.8.15.11 Enable Tabs During Automated Motion

By default, all tabs except the MAIN Tab are disabled during automated motion. It is possible for every tab but the CONVERSATIONAL Tab to be enabled during automated motion by setting the following HAL pin True:

```
setp qtplasmac.tabs_always_enabled 1
```

Warnung
It is the responsibility of the operator to ensure that the machine is equipped with a suitable, working hardware ESTOP. If using only a touchscreen to navigate the QtPlasmaC GUI, there is no way to stop automated machine motion on any tab but the MAIN tab.

10.8.15.12 Override Jog Inhibit Via Z+ Jog

It is possible to override the jog inhibit by using the GUI or keyboard to jog in the Z+ direction rather than checking the Override Jog box on the SETTINGS Tab.

This is done by changing the following preference to **True** in the **[GUI_OPTIONS]** of the <*machine_name*>.prefs file in the <machine_name> folder:

```
Aufhebung der Jog-Sperre über Z+
```

10.8.15.13 QtPlasmaC State Outputs

The plasmac HAL component has a HAL pin named **plasmac.state-out** which can be used to interface with user-coded components to provide the current state of the component.

The different states QtPlasmaC could encounter are as follows:

Zustand	Name	Beschreibung
0	IDLE	idle and waiting for a start command
1	PROBE_HEIGHT	move down to probe height
2	PROBE_DOWN	probe down until material sensed
3	PROBE_UP	probe up until material not sensed, this sets the zero height
4	ZERO_HEIGHT	not used at present
5	PIERCE_HEIGHT	move up to pierce height
6	TORCH_ON	turn the torch on
7	ARC_OK	wait until arc ok detected
8	PIERCE_DELAY	wait for pierce delay time
9	PUDDLE_JUMP	xy motion begins, move to puddle jump height
10	CUT_HEIGHT	move to cut height
11	CUT_MODE_01	cutting in either mode 0 or mode 1
12	CUT_MODE_2	cutting in mode 2
13	PAUSE_AT_END	pause motion at end of cut
14	SAFE_HEIGHT	move to safe height
15	MAX_HEIGHT	move to maximum height
16	END_CUT	end the current cut
17	END_JOB	end the current job
18	TORCHPULSE	a torch pulse is active
19	PAUSED_MOTION	cut recovery motion is active while paused
20	OHMIC_TEST	an ohmic test is active
21	PROBE_TEST	a probe test is active
22	SCRIBING	a scribing job is active
23	CONSUMABLE_CHANGE_ON	move to consumable change coordinates
24	CONSUMABLE_CHANGE_OFF	return from consumable change coordinates
25	CUT_RECOVERY_ON	cut recovery is active
26	CUT_RECOVERY_OFF	cut recovery is deactivated

The DEBUG state is for testing purposes only and will not normally be encountered.

10.8.15.14 QtPlasmaC Debug Print

The plasmac HAL component has a HAL pin named **plasmac.debug-print** which if set to 1 (true) will print to terminal every state change as a debug aid.

10.8.15.15 Hypertherm PowerMax Communications

Communications can be established with a Hypertherm PowerMax plasma cutter that has a RS485 port. This feature enables the setting of **Cut Mode**, **Cutting Amperage** and **Gas Pressure** automatically from the **Cut Parameters** of the material file. In addition, the user will be able to view the PowerMax's **Arc On Time** in hh:mm:ss format on the STATISTICS Tab.

If **Gas Pressure** is set to Zero then the PowerMax will automatically calculate the required pressure from the **Cut Mode**, **Cut Current**, torch type, and torch length.

Changing the cutting mode will set the gas pressure to zero causing the machine to use its automatic gas pressure mode.

The maximum and minimum values of these parameters are read from the plasma cutter and the related spin-buttons in the Cut Parameters are then limited by these values. Gas pressure cannot be changed from zero until communications have been established.

This feature is enabled by setting the correct port name for the PM_PORT option in the **[POWERMAX]** section of the <*machine_name*>.prefs file. If the PM_PORT option is not set in the <*machine_name*>.prefs file then the widgets associated with this feature will not be visible.

Example showing enabling the Hypertherm PowerMax Communications on USB0:

```
[POWERMAX]
Port = /dev/ttyusb0
```

If the user is unsure of the name of the port, there is a python script in the configuration directory that will show all available ports and can also be used to test communications with the plasma unit prior to enabling this feature in the QtPlasmaC GUI.

To use the test script follow these instructions:

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
```

The gas pressure units display (psi or bar) is determined by the data received during initial setup of the communication link and is then shown next to the Gas Pressure setting in the MATERIAL section of the PARAMETERS Tab.

The PowerMax machine will go into remote mode after communications have been established and may only be controlled remotely (via the QtPlasmaC GUI) at this point. The connection can be validated by observing the PowerMax display.

To switch the PowerMax back to local mode the user can either:

1. Disable PowerMax Comms from the MAIN Tab
2. Close LinuxCNC which will put the PowerMax into local mode during shutdown.
3. Turn the PowerMax off for 30 seconds and then power it back on.

Tipp
If PowerMax communications is active then selecting Mesh Mode will automatically select CPA mode on the PowerMax unit.

Anmerkung
To use the PowerMax communications feature it is necessary to have the python pyserial module installed.
If pyserial is not installed an error message will be displayed.

To install pyserial, enter the following command into a terminal window:

```
sudo apt install python-serial
```

A typical connection diagram is shown in the appendix of this document as well as confirmed working interfaces.

10.8.16 Internationalisation

It is possible to create translation files for QtPlasmaC to display in the language of the current locale.

To create and or edit a translation file requires that LinuxCNC has been installed as run in place.

The following assumes that the linuxcnc git directory is ~/linuxcnc-dev.

The **$** indicates a terminal prompt.

All language files are kept in ~/linuxcnc-dev/share/screens/qtplasmac/languages.

The qtplasmac.py file is a python version of the GUI file used for translation purposes.

The .ts files are the translation source files for the translations. These are the files that require creating/editing for each language.

The .qm files are the compiled translation files used by pyqt.

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 pyqt5-dev-tools installieren:

```
$ sudo apt install 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
```

The user can either create a new translation source file for a non-existing language translation or modify an existing translation source file due to changes being made to some text in a QtPlasmaC source file. If modifying an existing translation that has had no source file changes then this step is not required.

Create or edit a .ts file:

```
$ langfile xx
```

Anmerkung
this command is a script which runs the following: $ pylupdate5 **.py ..**/.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

The editing of the translation is done with the linguist application:

```
$ linguist
```

Then open the required translation file.

It is not necessary to provide a translation for every text string, if no translation is specified for a string then the original string will be used in the application. The user needs to be careful with the length of strings that appear on widgets as space is limited. If possible try to make the translation no longer than the original.

When editing is complete save the file:

```
File > Save
```

Then create the .qm file:

```
File > Release
```

Then create links to the compiled .qm file for the other QtPlasmaC GUIs.

```
$ ln -s qtplasmac_en.qm ../../qtplasmac_4x3/languages/
$ ln -s qtplasmac_en.qm ../../qtplasmac_9x16/languages/
```

QtPlasmaC will be translated to the language of the current locale on the next start so long as a .qm file exists in that language.

10.8.17 Appendix

10.8.17.1 Beispielkonfigurationen

There are example configuration files which use the QtPlasmaC GUI to simulate plasma cutting machines.

They can be found in the LinuxCNC chooser under: Sample Configurations -> by_machine -> qtplasmac

Three versions are available in both metric and imperial units:

1. qtplasmac_l - 16:9 format, minimum resolution 1366x768
2. qtplasmac_p - 9:16 format, minimum resolution 786x1366
3. qtplasmac_s - 4:3 format, minimum resolution 1024x768

Each sample configuration includes a popup control panel to simulate various inputs to the GUI such as:

1. ARC VOLTAGE
2. OHMIC SENSE
3. FLOAT SWITCH
4. BREAKAWAY SWITCH
5. ESTOP (engl. für Notaus)

10.8.17.2 NGC Samples

There are some sample G-code files in the ~/linuxcnc/nc_files/examples/plasmac directory.

10.8.17.3 QtPlasmaC Specific G-codes

Beschreibung	**Code**
Begin cut	M3 $0 S1
End cut	M5 $0
Begin scribe	M3 $1 S1
End scribe	M5 $1
Begin center spot	M3 $2 S1
End center spot	M5 $2
End all the above	M5 $-1
Select a material	M190 P*n* *n* denotes the material number.
Wait for material change confirmation	M66 P3 L3 Qn *n* is delay time (in seconds). This value may need to be increased for very large material files.
Set feed rate from material	F#<_hal[plasmac.cut-feed-rate]>
Enable Ignore Arc OK	M62 P1 (synchronized with motion) M64 P1 (immediate)
Disable Ignore Arc OK	M63 P1 (synchronized with motion) M65 P1 (immediate)
Disable THC	M62 P2 (synchronized with motion) M64 P2 (immediate)
Enable THC	M63 P2 (synchronized with motion) M65 P2 (immediate)
Disable Torch	M62 P3 (synchronized with motion) M64 P3 (immediate)
Enable Torch	M63 P3 (synchronized with motion) M65 P3 (immediate)
Set velocity to a percentage of feed rate	M67 E3 Qn (synchronized with motion) M68 E3 Qn (immediate) *n* is the percentage to set 10 is the minimum, below this will be set to 100% 100 is the maximum, above this will be set to 100% **It is recommended to have M68 E3 Q0 in both the preamble and postamble**
Cutter compensation - left of path	G41.1 D#<_hal[plasmac.kerf-width]>

Beschreibung	Code
Cutter compensation - right of path	G42.1 D#<_hal[plasmac.kerf-width]>
Cutter compensation off	G40 **Note that M62 through M68 are invalid while cutter compensation is on**
Cut holes at 60% feed rate	#<holes> = 1 for holes less than 32 mm (1.26”) diameter
Cut holes at 60% feed rate, turn torch off at hole end, continue hole path for over cut	#<holes> = 2 for holes less than 32 mm (1.26”) diameter over cut length = 4 mm (0.157”)
Cut holes and arcs at 60% feed rate	#<holes> = 3 for holes less than 32 mm (1.26”) diameter for arcs less than 16 mm (0.63”) radius
Cut holes and arcs at 60% feed rate, turn torch off at hole end, continue hole path for over cut	#<holes> = 4 for holes less than 32 mm (1.26”) diameter for arcs less than 16 mm (0.63”) radius over cut length = 4 mm (0.157”)
Specify hole diameter for #<holes> = 1-4	#<h_diameter> = *n* (*n* is the diameter, use the same units system as the rest of the G-code file)
Specify hole velocity for #<holes>=1-4	#<h_velocity> = *n* (*n* is the percentage, set the percentage of the current feed rate)
Specify over cut length	#<oclength> = *n* (*n* is the length, use the same units system as the rest of the G-code file)
Specify pierce-only mode	#<pierce-only> = *n* (*n* is the mode, 0=normal cut mode, 1=pierce only mode)
Create or edit materials. options: 0 - Create temporary default 1 - Add if not existing 2 - Overwrite if existing else add new	mandatory parameters: (o=<option>, nu=<nn>, na=<ll>, ph=<nn>, pd=<nn>, ch=<nn>, fr=<nn>) optional parameters: (kw=<nn>, th=<nn>, ca=<nn>, cv=<nn>, pe=<nn>, gp=<nn>, cm=<nn>, jh=<nn>, jd=<nn>)
Keep Z Motion	#<keep-z-motion> = 1

10.8.17.4 QtPlasmaC G-code Examples

Beschreibung	Beispiel
Select material and do a normal cut	M190 P3 M66 P3 L3 Q1 F#<_hal[plasmac.cut-feed-rate]> M3 $0 S1 . . M5 $0
Set velocity to 100% of CutFeedRate	M67 E3 Q0 or M67 E3 Q100
Set velocity to 60% of CutFeedRate	M67 E3 Q60
Geschwindigkeit auf 40% der CutFeedRate setzen	M67 E3 Q40

Beschreibung	**Beispiel**
Schneiden Sie ein Loch mit 60% reduzierter Geschwindigkeit mit der Geschwindigkeitseinstellung	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) 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 (allow paused motion) #<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)
Schneiden Sie ein Loch mit Überschnitt mit Brenner deaktivieren	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) 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 4mm) 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 (allow paused motion) #<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)

Beschreibung	**Beispiel**
Schneiden Sie ein Loch mit 6,5 mm Überschnitt mit dem Befehl #<holes>	G21 (metric) G64 P0.05 M52 P1 (allow paused motion) #<holes> = 2 (over cut for holes) <oclength> = 6.5 (6.5mm 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)
Wählen Sie Ritzer/Schreiber (engl. scribe) und wählen Sie den Brenner am Ende des Ritzens	. . M52 P1 (paused motion on) 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)
Hole center spotting.	(Requires a small motion command or nothing happens) 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
Create temporary default material	(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000)
Edit material, if not existing create a new one	(o=2, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=1.0)

10.8.17.5 Mesa THCAD

The Mesa THCAD is a common way of obtaining the arc voltage from a plasma cutter and is also useful for ohmic sensing of the material during probing. The THCAD may be used for parallel port configurations as well as configurations using Mesa Electronics hardware. The THCAD is available in three different models, THCAD-5, THCAD-10, and THCAD-300.

There is a mode jumper on each THCAD card which should be set to **UNIPOLAR**

There is a frequency divider jumper on each THCAD card which should be set according to the hardware type:

Input Device	Recommended Setting
Parallel Port with very low latency	F/32
Parallel Port recommended starting point	F/64
Parallel Port with higher latency, or when cutting thick material	F/128
Mesa Card	F/32

This value is required to be entered into PNcConf during installation.

Anmerkung
If using a parallel port it may be necessary for the user to adjust the jumper setting and the subsequent scaling values on the Parameters Tab to achieve optimal results. Symptoms may include random torch raises or dives during otherwise stable cutting. Halscope plots may be useful in diagnosing these issues.

Located on the rear of the THCAD is a calibration sticker showing:

```
   THCAD-nnn

0V  121.1 kHz
5V  925.3 kHz
```

or similar values, these values are required to be entered into PNcConf during installation.

PNcConf has entries for all required THCAD parameters and will calculate and configure any required settings. The calculations used are as follows:

Voltage Scale

```
vs = r / ((f - z) / d / v)
```

Voltage Offset

```
vo = z / d
```

r = divider ratio (see below).

f = full scale value from calibration sticker.

z = 0 V value from calibration sticker.

d = value from jumper above.

v = full scale voltage of THCAD

Divider Ratio *THCAD-5 or THCAD-10*

If connecting to a plasma CNC port then the divider ratio is selected from the plasma machine. A common ratio used is 20:1

If connecting to the plasma machines full arc voltage then a common setup for a THCAD-10 is to use a 1 MΩ; resistor from arc negative to THCAD negative and a 1 MΩ; resistor from arc positive to THCAD positive. The divider ratio is obtained by:

```
r = (total_resistance + 100000) / 100000
```

THCAD-300

```
r = 1
```

Wichtig
IF THE USER IS USING A HF START PLASMA POWER SUPPLY THEN EACH OF THESE RESISTANCES SHOULD BE MADE UP OF SEVERAL HIGH VOLTAGE RESISTORS.

Achtung
IF THE USER IS USING A HF START PLASMA POWER SUPPLY THEN OHMIC SENSING IS NOT RECOMMENDED.

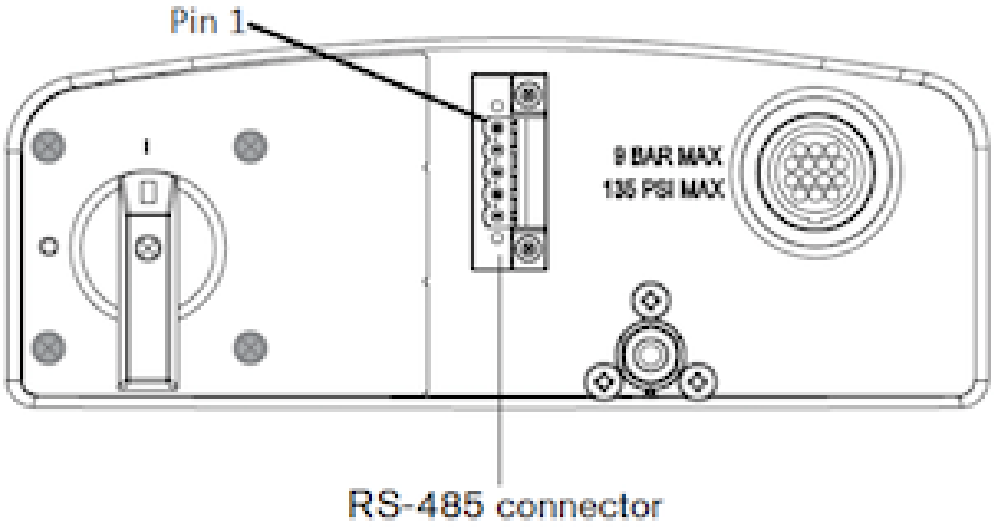
Anmerkung
These values can be calculated by using this online calculator.

Anmerkung
There is a lowpass filter available which may be useful if using a THCAD and there is a lot of noise on the returned arc voltage.

10.8.17.6 RS485 Connections

Hypertherm RS485 Wiring Diagram (wire colors inside the Hypertherm in parentheses):

Connection at Machine Pin #	Connection at Breakout Board
1 - Tx+ (Red)	->RXD+
2 - Tx- (Black)	->RXD-
3 - Rx+ (Brown)	->T/R+
4 - Rx- (White)	->T/R-
5 - GND (Green)	->GND



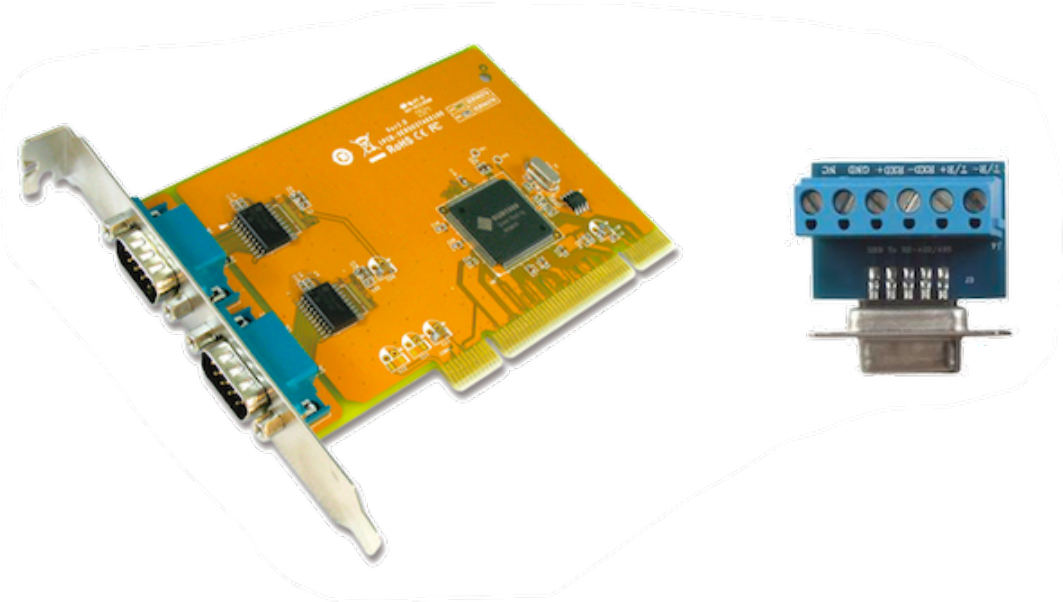
RS485 interfaces that are known to work:

DTECH DT-5019 USB to RS-485 converter adapter:

The following is necessary to convert a motherboard Serial connection or Serial card (RS232) to RS485:

DTECH RS-232 to RS-485 converter:

Serial card example (Sunnix SER5037A PCI Card shown with Breakout Board):

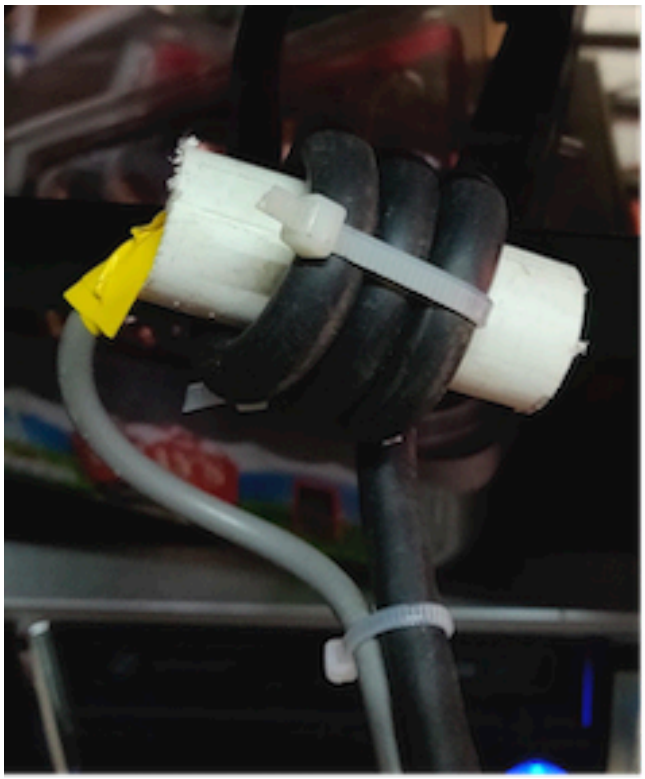


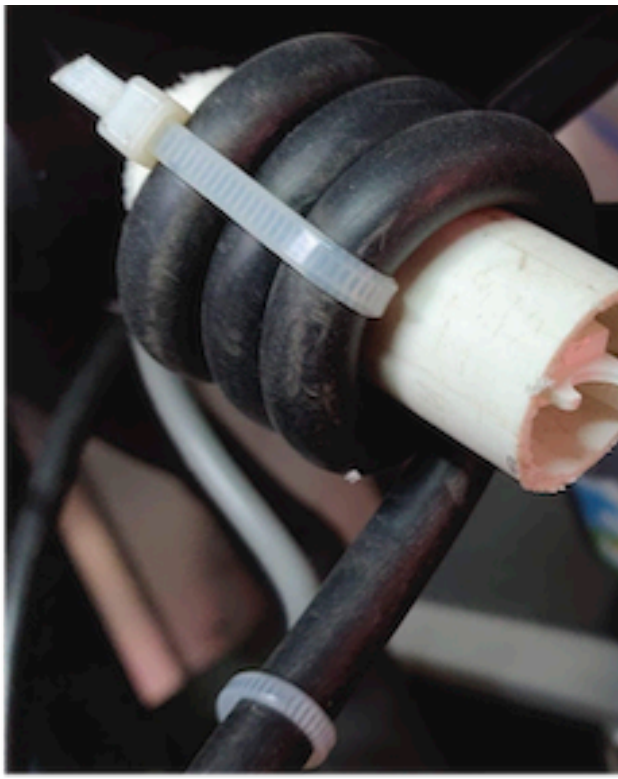
10.8.17.7 Arc OK With A Reed Relay

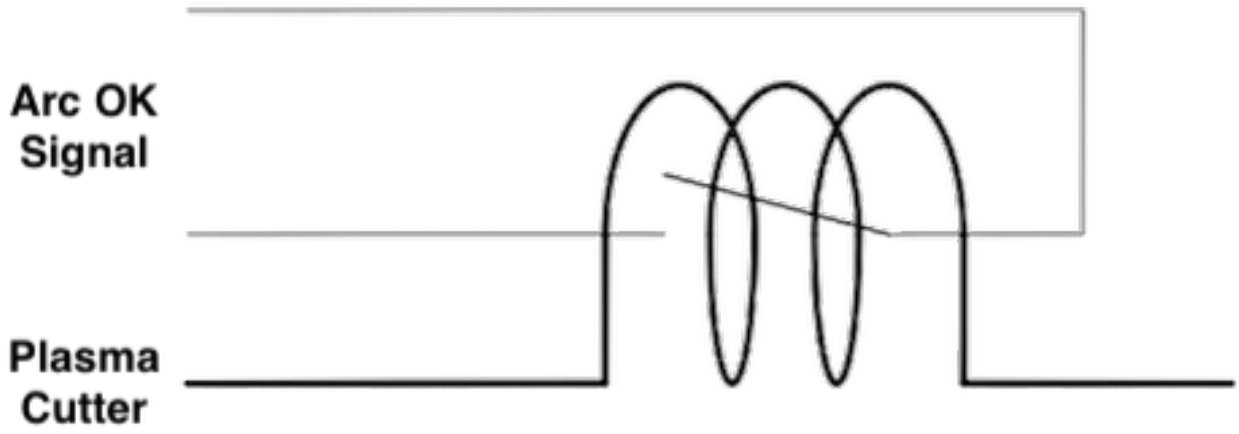
An effective and very reliable method of obtaining an Arc OK signal from a plasma power supply without a CNC port is to mount a reed relay inside a non-conductive tube and wrap and secure three turns of the work lead around the tube.

This assembly will now act as a relay that will switch on when current is flowing through the work lead which only occurs when a cutting arc has been established.

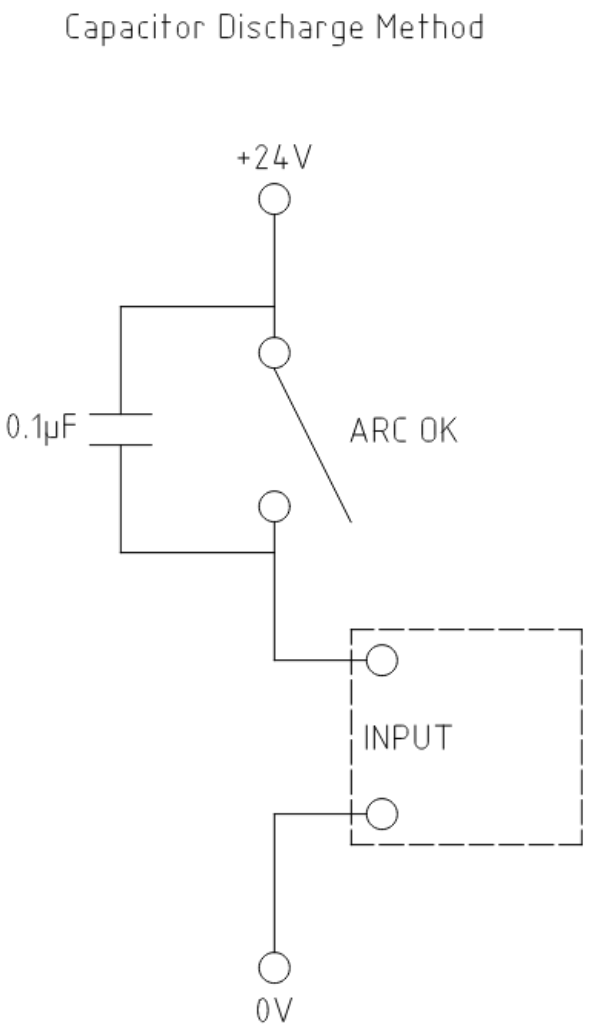
This will require that QtPlasmaC be operated in Mode 1 rather than Mode 0. See the QtPlasmaC Modes sections for more information.

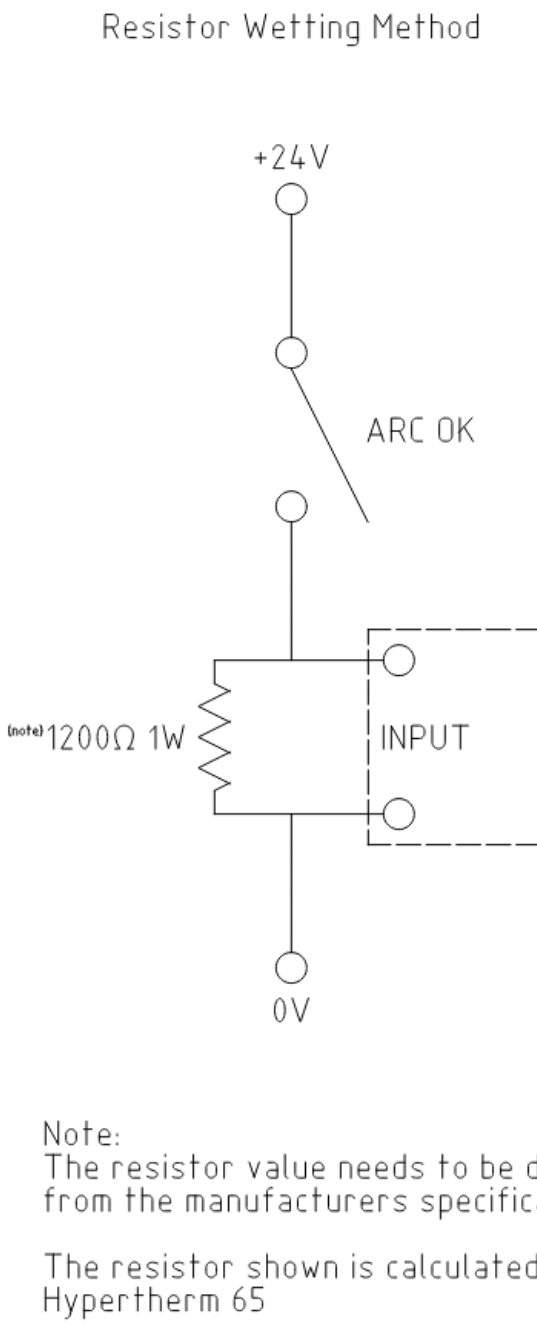





Arc OK
Signal
Plasma
Cutter

10.8.17.8 Contact Load Schematics





A full description is at Contact Load

10.8.18 Bekannte Probleme

10.8.18.1 Keyboard Jogging

There is a known issue with some combinations of hardware and keyboards that may affect the autorepeat feature of the keyboard and will then affect keyboard jogging by intermittent stopping and starting during jogging. This issue can be prevented by disabling the Operating System's autorepeat feature for all keys. QtPlasmaC uses this disabling feature by default for all keys only when the MAIN

Tab is visible, with the following exceptions when autorepeat is allowed with the MAIN Tab visible: G-code editor is active, MDI is active. When QtPlasmaC is shut down, the Operating System's autorepeat feature will be enabled for all keys.

If the user wishes to prevent QtPlasmaC from changing the Operating System's autorepeat settings, enter the following in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Autorepeat all == True
```

Dieses Problem betrifft nicht das Joggen mit den GUI-Jog-Tasten.

Anmerkung
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.

10.8.19 Unterstützung

Online-Hilfe und -Unterstützung finden Sie unter PlasmaC section des LinuxCNC Forum.

The user can create a compressed file containing the complete machine configuration to aid in fault diagnosis by pressing following the directions in the backup section. The resulting file is suitable for attaching to a post on the LinuxCNC Forum to help the community diagnose specific issues.

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.

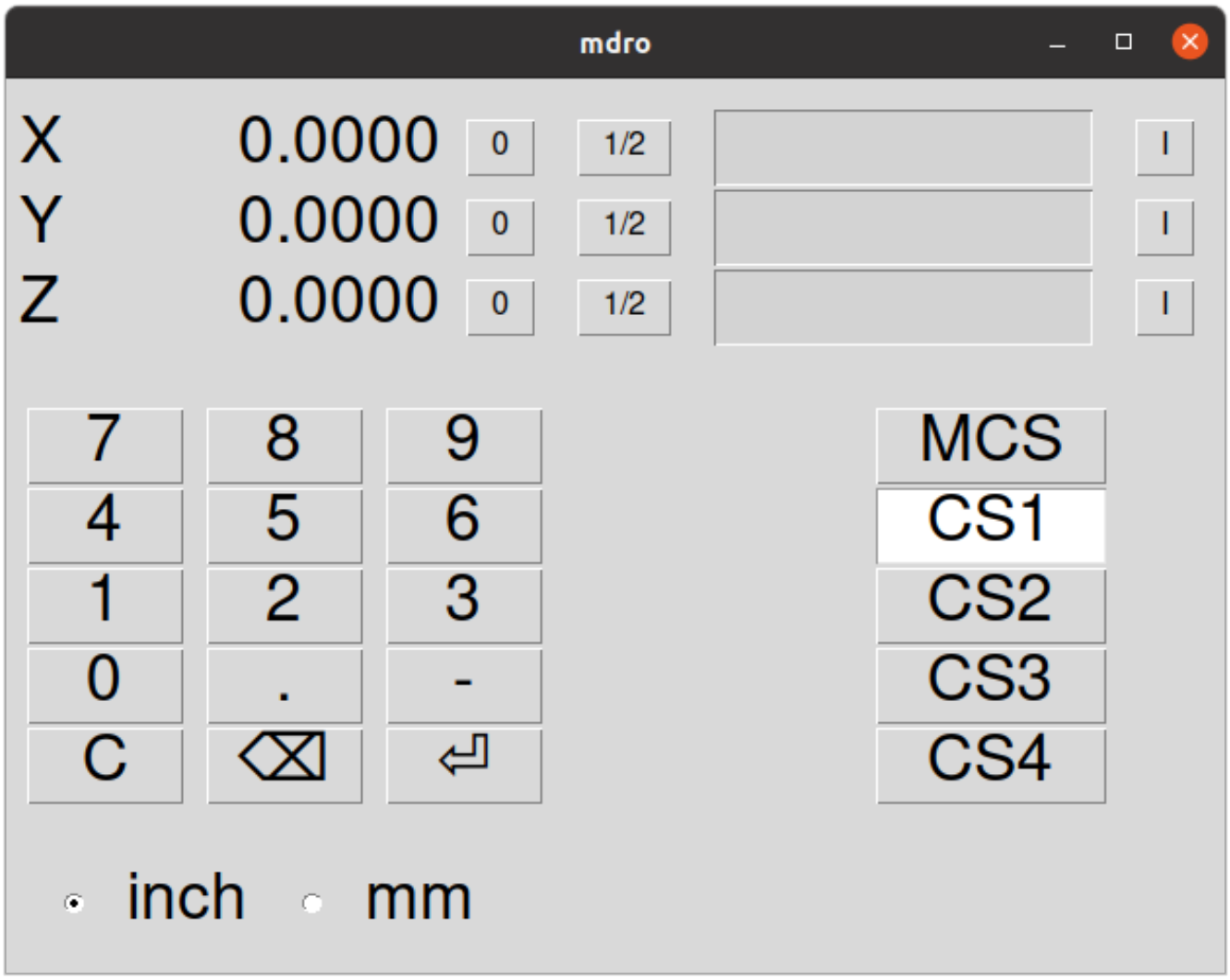


Abbildung 10.49: MDRO Fenster

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 Abbildung 10.49 oben.

10.9.2.1 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.

10.9.2.2 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.

10.9.2.3 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
```

Kapitel 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.

11.1.2.1 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

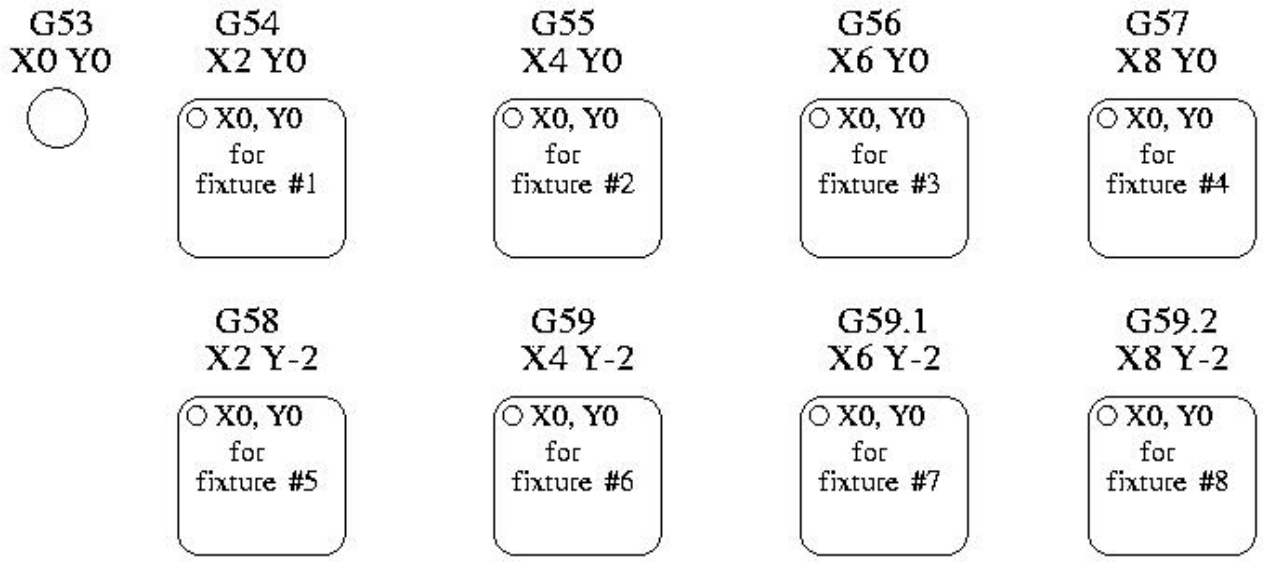


Abbildung 11.1: Beispiel für Koordinatensysteme

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 Koordinatensyste 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:

Tabelle 11.1: 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.

```
G54 verwendet die Parameter des Koordinatensystems 1(((G54)))
G55 verwendet die Parameter des Koordinatensystems 2(((G55)))
G56 verwendet die Parameter des Koordinatensystems 3(((G56)))
G57 verwendet Parameter des Koordinatensystems 4(((G57)))
G58 verwendet Parameter des Koordinatensystems 5(((G58)))
G59 verwendet Parameter des Koordinatensystems 6(((G59)))
G59.1 verwendet Parameter des Koordinatensystems 7(((G59.1)))
G59.2 verwendet Parameter des Koordinatensystems 8(((G59.2)))
G59.3 verwendet die Parameter des Koordinatensystems 9(((G59.3)))
```

11.1.3.1 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.

11.1.3.2 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).

Anmerkung
Wir geben hier nur einen kurzen Überblick, eine vollständige Beschreibung finden Sie in den G-Code-Abschnitten.

11.1.4 Lokale und globale Offsets

11.1.4.1 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.

Achtung
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.

Achtung
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.

11.1.5.1 Die G92-Befehle

G92 wird typischerweise auf zwei konzeptionell unterschiedliche Arten verwendet: als ”globaler Koordinaten 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.

Anmerkung
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.

Anmerkung
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.

Anmerkung
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.

11.1.5.2 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.

11.1.5.3 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.

11.1.5.4 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

11.1.6.1 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.

11.1.6.2 Beispielprogramm mit G52-Offsets

(muss noch geschrieben werden)

11.2 Werkzeugkorrektur

11.2.1 Touch Off((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 *T*n *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 *T*n *M6* geladen ist.

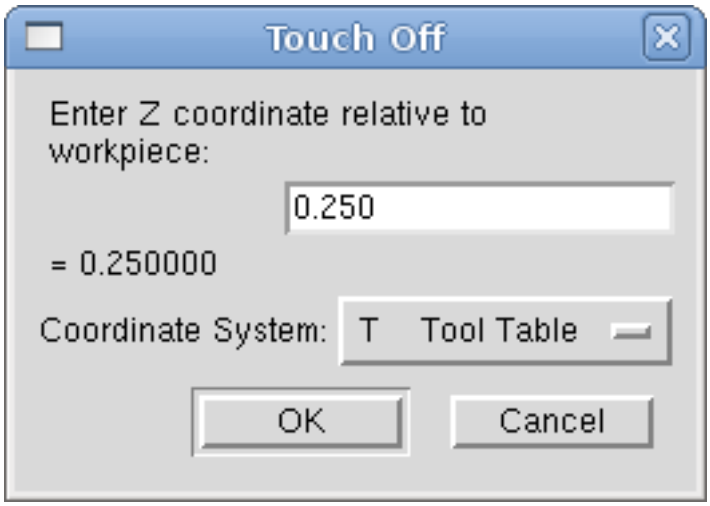


Abbildung 11.2: Touch-Off-Werkzeugtabelle

11.2.1.1 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)

Anmerkung
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.

11.2.2.1 Werkzeugtabellen-Format

Tabelle 11.2: Werkzeugtabellen-Format

T#	P#	X	Y	Z	A	B	C	U	V	W	Durch...	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 unsignierte ganze Zahlen 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 Nummer wird nur verwendet, wenn die Fräserkompensation mit diesem Werkzeug aktiviert ist. Wenn der programmierte Weg 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 der programmierte Weg während der Kompensation der Weg eines Werkzeugs ist, dessen Durchmesser nominal ist, sollte dies eine kleine Zahl (positiv oder negativ, aber nahe Null) sein, die nur die Differenz zwischen dem gemessenen Durchmesser des Werkzeugs und dem Nenndurchmesser darstellt. Wenn die Fräserkompensation nicht mit einem Werkzeug verwendet wird, spielt es keine Rolle, welche Nummer in dieser Spalte enthalten ist.

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.

Anmerkung
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.

11.2.2.2 Werkzeug-E/A (engl. tool I/O)

Das durch **[EMCIO]EMCIO = io** spezifizierte Userspace-Programm wird üblicherweise für die Werkzeugwe Verwaltung (und andere E/A (I/O)-Funktionen zum Aktivieren von LinuxCNC und die Steuerung von Kühlmittel-/Schmiermittel-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**

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

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

---

Anmerkung
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.

11.2.2.3 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 Userspace-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.

Anmerkung
Bei Verwendung von [EMCIO]RANDOM_TOOLCHANGER = 0 (Standardeinstellung) ist die Taschennummer *P* ein Parameter der Werkzeugdaten, die aus der tooldata-Quelle ([EMCIO]TOOL_TABLE oder [EMCIO]DB_PROGRAM) abgerufen werden. In vielen Anwendungen ist es behoben, kann aber durch Änderungen am [EMCIO]TOOL_TABLE oder programmgesteuert geändert werden, wenn das [EMCIO]DB_PROGRAM verwendet wird. LinuxCNC pusht Aktualisierungen an die Datenquelle ([EMCIO]TOOL_TABLE oder [EMCIO]DB_PROGRAM) für die G-codes G10L1, G10L10, G10L11, M61. LinuxCNC kann Werkzeugdaten-Aktualisierungen aus der Datenquelle über UI-Befehle (Benutzeroberfläche) (Python-Beispiel: linuxcnc.command().load_tool_table()) oder über den G-Code `G10L0` abrufen.

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

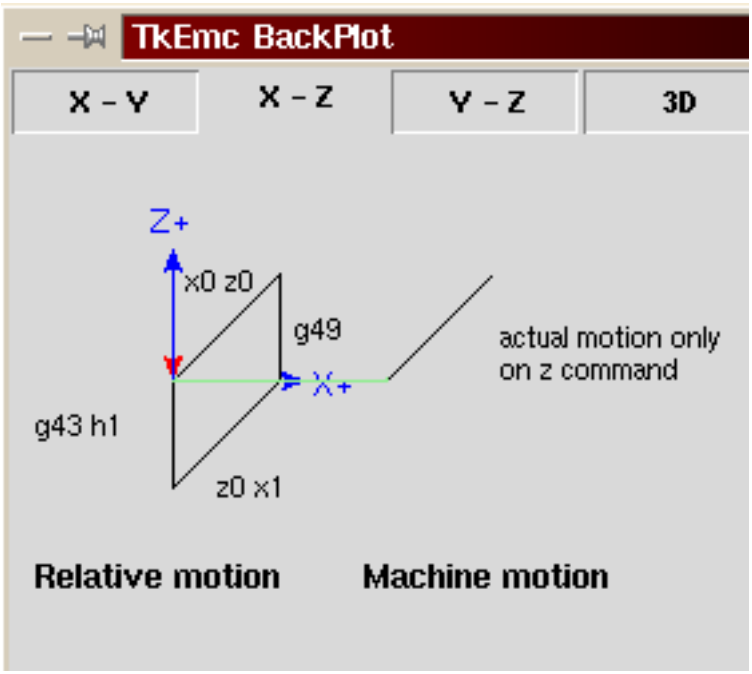
Die Werkzeuglängenkorrekturen werden als positive Zahlen in der Werkzeugtabelle angegeben. Eine Werkzeugkorrektur wird mit G43 H*n* programmiert, wobei *n* die Indexnummer des gewünschten Werkzeugs in der Werkzeugtabelle ist. Es ist vorgesehen, dass alle Einträge in der Werkzeugtabelle positiv sind. 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 Fräserradiuskompensation

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.

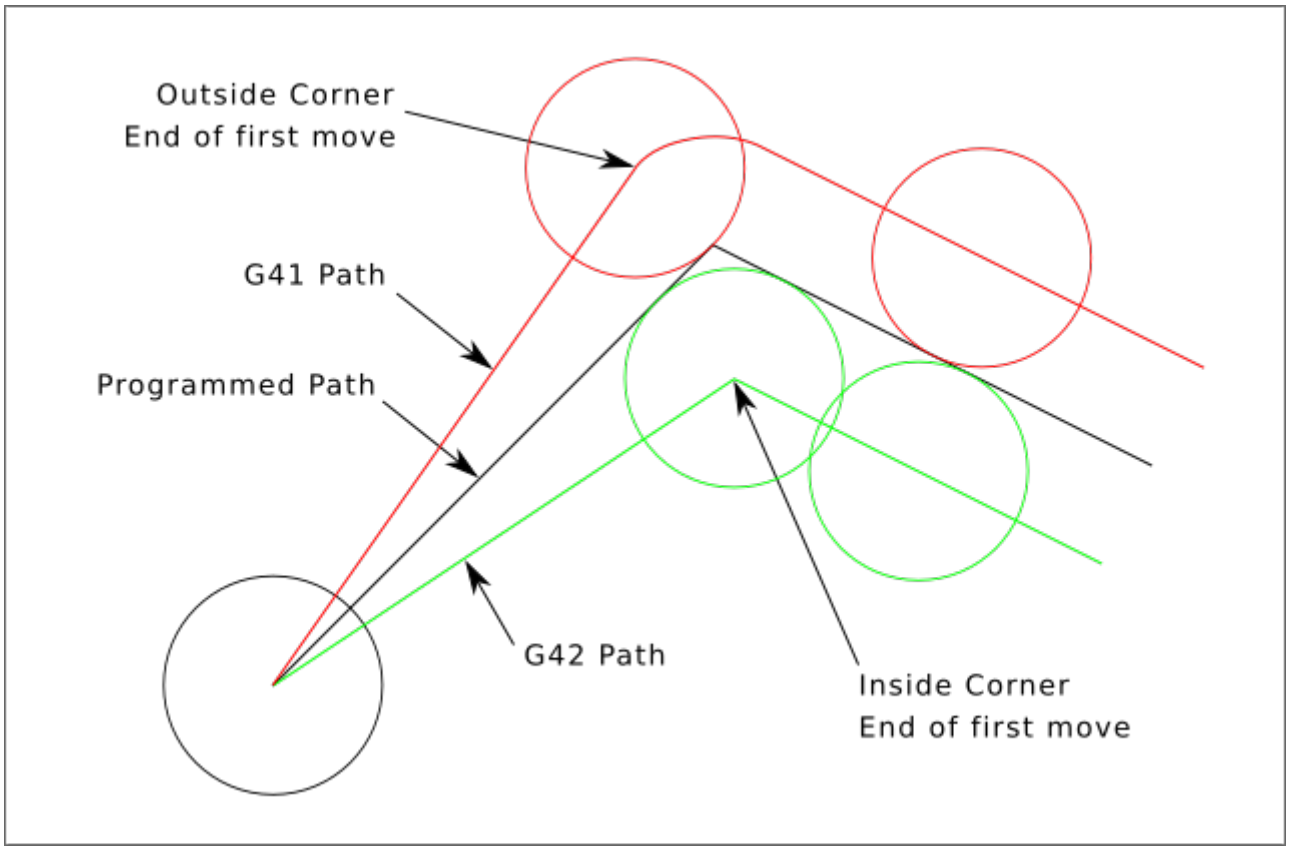


Abbildung 11.3: Ausgleich am Endpunkt (engl. Compensation End Point)

11.2.4.1 Übersicht

Die Fräserkompensation verwendet die Daten aus der Werkzeugtabelle, um den benötigten Versatz zu bestimmen. Die Daten können zur Laufzeit mit G10 L1 eingestellt werden.

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.

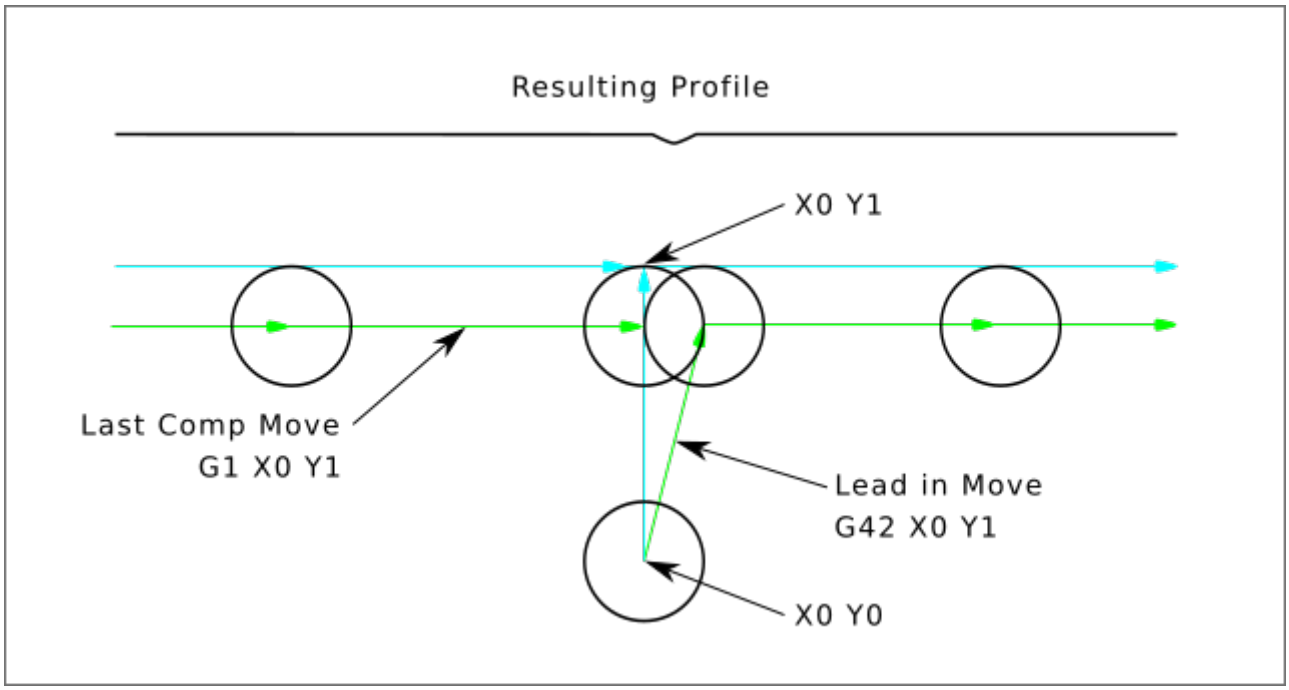


Abbildung 11.4: Eingangs-Bewegung (engl. entry move)

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 können programmiert werden, während die Kompensation eingeschaltet ist.

Starten Sie ein Programm mit G40, um sicherzustellen, dass die Kompensation ausgeschaltet ist.

11.2.4.2 Beispiele

```
G-Code
F25 ( Set Feed Rate )
G40 ( Cancel Comp )
G10 L1 P1 R0.25 Z1 ( Set Tool Table )
T1 M6 ( Load Tool )
G42 ( Start Comp Right )
G1 X1 Y1 (Lead In Move)
X5 ( Cut Path )
Y5
X1
Y1
G40 ( Cancel Comp )
G0 X0 Y0 ( Exit Move )
M2 ( End Program )
```

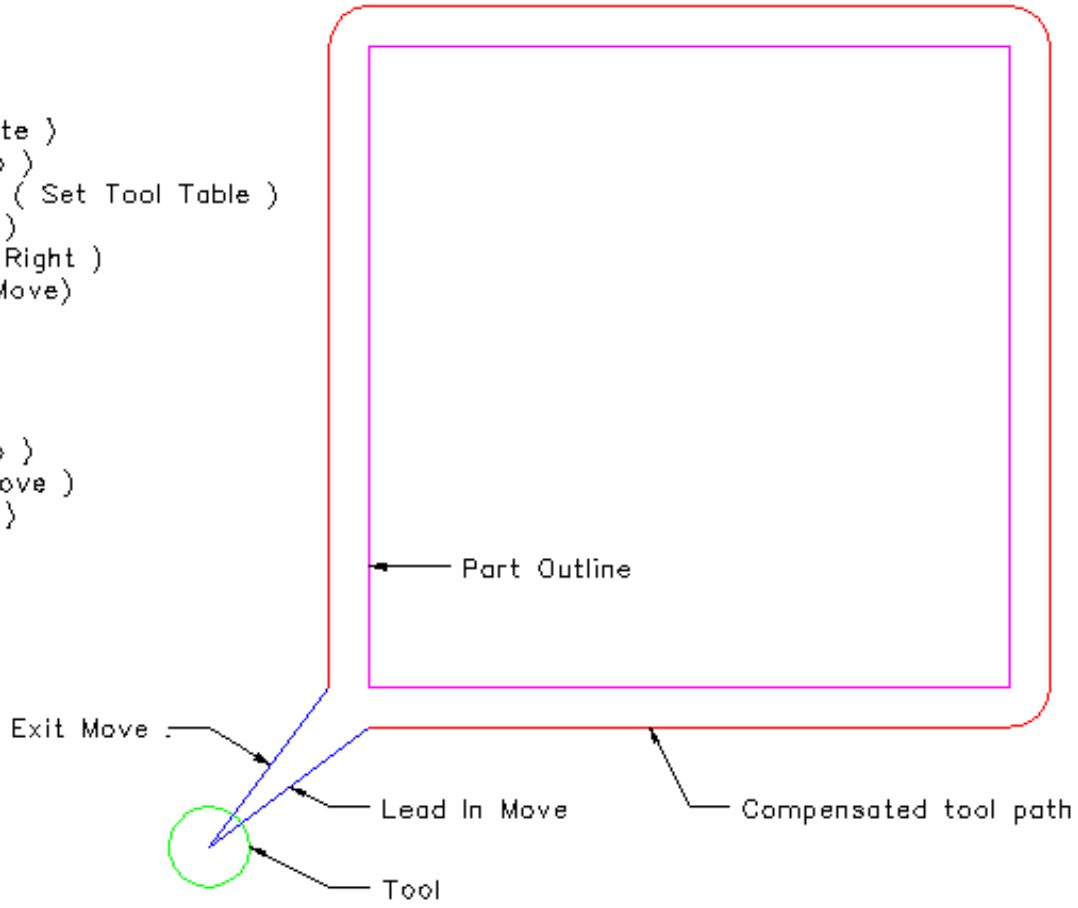


Abbildung 11.5: Äußeres Profil

```
G20 ( Inch Mode )
F30 ( Set Feed Rate )
G10 L1 P1 R.25 Z1 ( Set Tool Table )
T1 M6 ( Load the Tool )
G0 Z0 ( Move to safe Z height )
G41 ( Start Cutter Comp Left )
X4 Y3 ( Rapid to start point )
G1 X5 Z-1 ( Move to cut height )
G3 X6 Y4 J1 ( Arc into cut path )
G1 Y6 ( Cut Profile )
X2
Y2
X6
Y4
G3 X5 Y5 I-1 ( Arc out of cut path )
G0 Z0 ( Move cutter to safe Z height )
G40 ( Stop Cutter Comp )
G0 X1 Y1 ( Move to safe position )
T0 M6 ( Remove Tool )
M2 ( End Program )
```

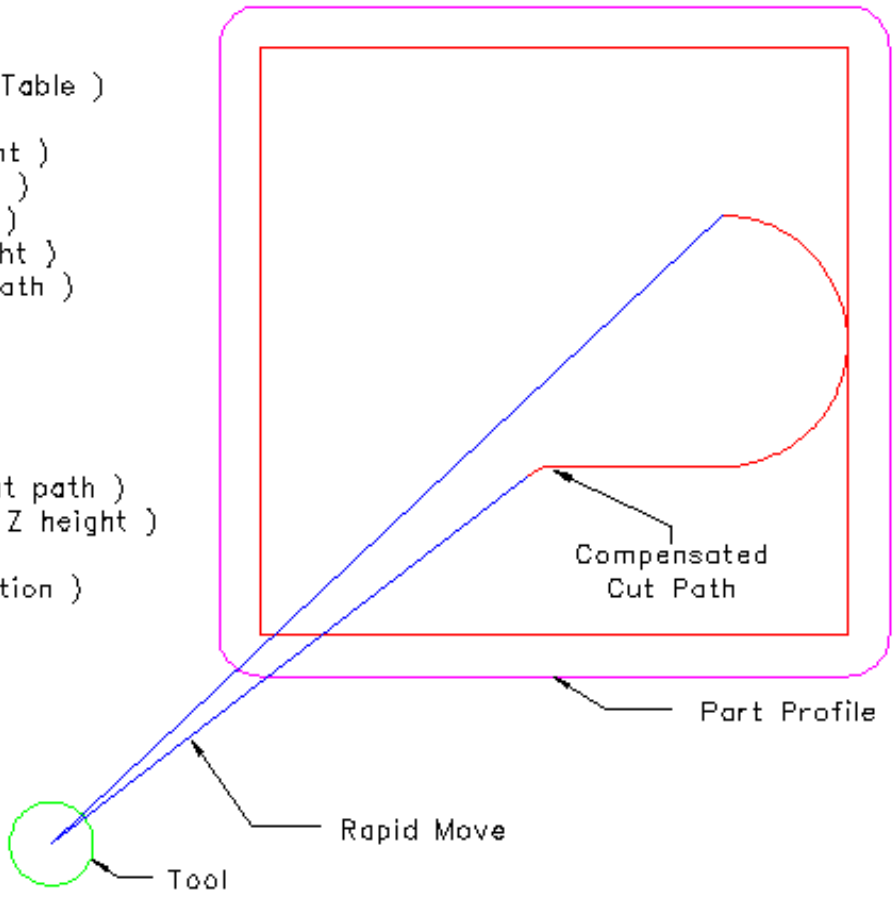


Abbildung 11.6: Innenprofil

11.3 GUI zur Werkzeug-Bearbeitung

11.3.1 Übersicht

Anmerkung
The tooledit elements described here are available since version 2.5.1 and later. In version 2.5.0, the graphical interface interface does not allow these adjustments.

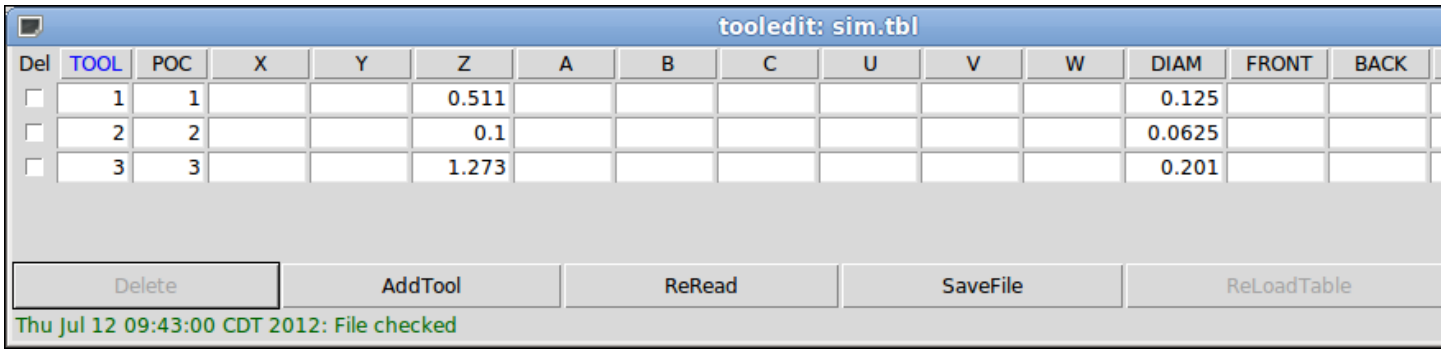


Abbildung 11.7: Tool Edit GUI - Überblick

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.

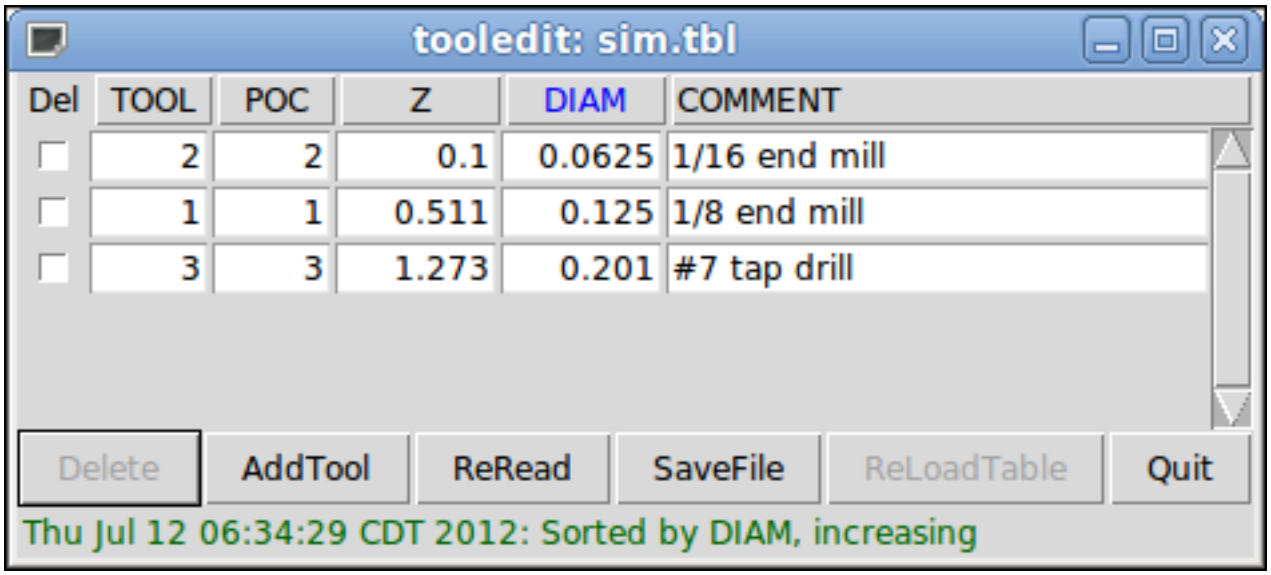


Abbildung 11.8: Tool Edit GUI - Spaltensortierung

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
```

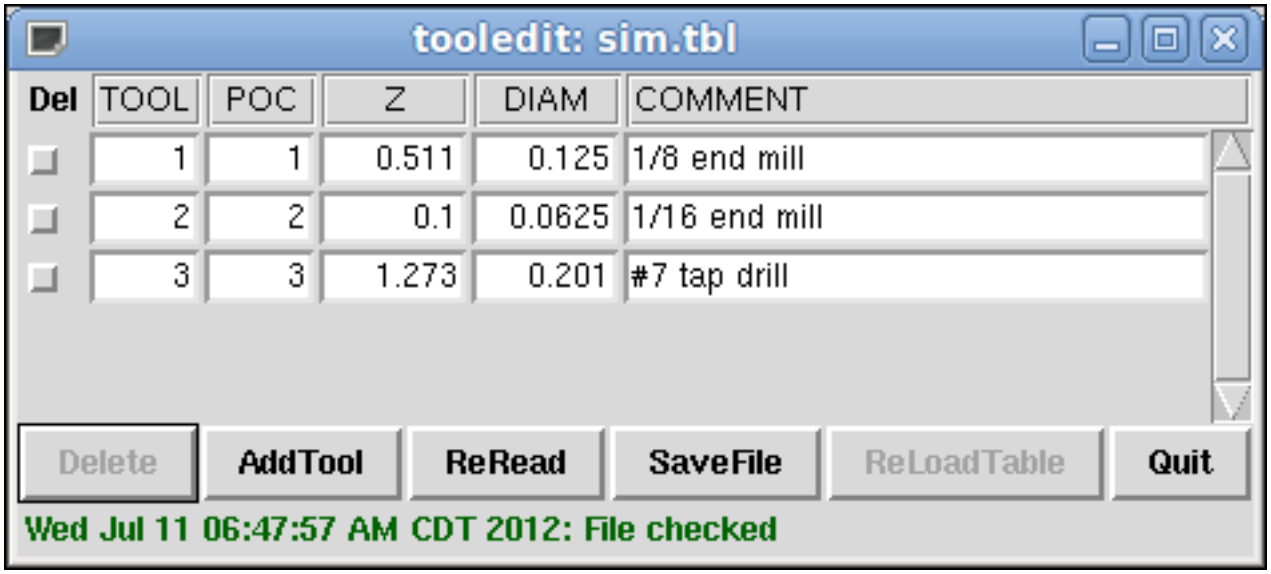


Abbildung 11.9: Tool Edit GUI - Spaltenauswahl Beispiel

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 G-Code Übersicht

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.

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).

- ein optionales Blocklöschzeichen, das ein Schrägstrich / ist.
- eine optionale Zeilennummer.
- eine beliebige Anzahl von Wörtern, Parametereinstellungen und Kommentaren.
- 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.

11.4.2.1 /: Block löschen (engl. block delete)

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 

11.4.2.2 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.

11.4.2.3 Wort

Ein Wort ist ein Buchstabe außer N, gefolgt von einer Gleitkommazahl.

Wörter können mit jedem der in der folgenden Tabelle aufgeführten Buchstaben beginnen. Die Tabelle enthält der Vollständigkeit halber auch N, obwohl Zeilennummern, 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.

Tabelle 11.3: Wörter und ihre Bedeutungen

Buchstabe	Bedeutung
A	A-Achse der Maschine
B	B-Achse der Maschine
C	C-Achse der Maschine
D	Werkzeugradius-Korrekturnummer
F	Vorschubgeschwindigkeit
G	Allgemeine Funktion (siehe Tabelle <cap:modal-groups,Modale Gruppen>>)
H	Werkzeuglängen-Offset-Index
I	X-Versatz für Bögen und G87-Festzyklen
J	Y-Versatz für Bögen und G87-Festzyklen
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 <cap:modal-groups,Modal Groups>>)
N	Zeilennummer
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

11.4.2.4 Nummern(Nummern)

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.

11.4.3 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: gcode:binary-operators,Boolesche Operatoren>> (*AND*, *OR*, *XOR*, und die Vergleichsoperatoren *EQ*,*NE*,*GT*,*GE*,*LT*,*LE*), sowie die *MOD*, *ROUND*, *FUP* und *FIX* <gcode:functions,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.

11.4.3.1 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.
- *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.
- *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.

[1] Der Bereich der persistenten Parameter kann sich mit fortschreitender Entwicklung ändern. Dieser Bereich liegt derzeit zwischen 5161 und 5390. Er ist 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.

- *5210* - 1, wenn ”G52”- oder ”G92”-Offset derzeit angewendet wird, sonst 0. Standardmäßig flüchtig; beständig, wenn *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei.
- *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* - Werkzeug-Offsets für X, Y, Z, A, B, C, U, V & W. Flüchtig.
- *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.
- *5600* - Werkzeugwechsler-Fehleranzeige. Wird mit der Komponente iocontrol-v2 verwendet. 1: Werkzeugwechsler gestört, 0: normal. Flüchtig.
- *5601* - Fehlercode des Werkzeugwechslers. Wird mit der Komponente iocontrol-v2 verwendet. Gibt den Wert des HAL-Pins *toolchanger-reason* wieder, wenn ein Fehler aufgetreten ist. 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.

Tabelle 11.4: Parameter-Dateiformat

Parameter-Nummer	Parameter-Wert
5161	0.0
5162	0.0

11.4.3.2 Unterprogramm-Parameter

- *1-30* Lokale Parameter für den Aufruf von Unterroutinen. Diese Parameter sind lokal für das Unterprogramm. Flüchtig. Siehe auch das Kapitel zu O-Codes.

11.4.3.3 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.

11.4.3.4 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

- #<_plane> - gibt den Wert zurück, der die aktuelle Ebene bezeichnet:

Ebene	Rückgabewert
G17	170
G18	180
G19	190
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
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.

11.4.3.5 Systemparameter

- *#<_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.
- *#<_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-Wortes *return* oder *endsub*. Standardwert 0, wenn kein Ausdruck nach *return* oder *endsub*. Wird beim Programmstart auf 0 initialisiert.
- *#<_value_returned>* - 1.0 wenn das letzte O-Wort *return* oder *endsub* einen Wert zurückgegeben hat, sonst 0. Wird durch den nächsten O-Wort-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-Wort-Prozeduren. Für die Fehlersuche.
- *#<_remap_level>* - aktuelle Ebene des Remap-Stapels. Jeder Remap in einem Block erhöht die Remap-Ebene um eins. Zur Fehlersuche.

11.4.4 HAL-Pins und INI-Werte

Wenn dies in der <sub:ini:sec:rs274ngc, 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. Bei den Namen wird nicht zwischen Groß- und Kleinschreibung unterschieden - sie werden vor der Konsultation der INI-Datei in Großbuchstaben umgewandelt.

- *#<_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.

11.4.5 Ausdrücke (engl. expression)

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]]]*.

11.4.6 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.

Tabelle 11.5: Vorrang der Operatoren

Operatoren	**Vorrang**
**	*höchste*
* / MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	*niedrigste*

11.4.7 Gleichheit und Gleitkommawerte

Die Sprache RS274/NGC unterstützt nur Fließkommazahlen mit endlicher 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 0,0001 ist (dieser Wert ist als *TOLERANCE_EQUAL* in src/emc/rs274ngc/interp_internal.hh definiert).

11.4.8 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.

Tabelle 11.6: G-Code-Funktionen

Funktionsname	**Funktionsergebnis**
ATAN[arg]/[arg]	Four quadrant inverse tangent
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
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
```

11.4.9 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 <gcode:modal-groups,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.

11.4.10 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.

11.4.11 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.

11.4.12 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.

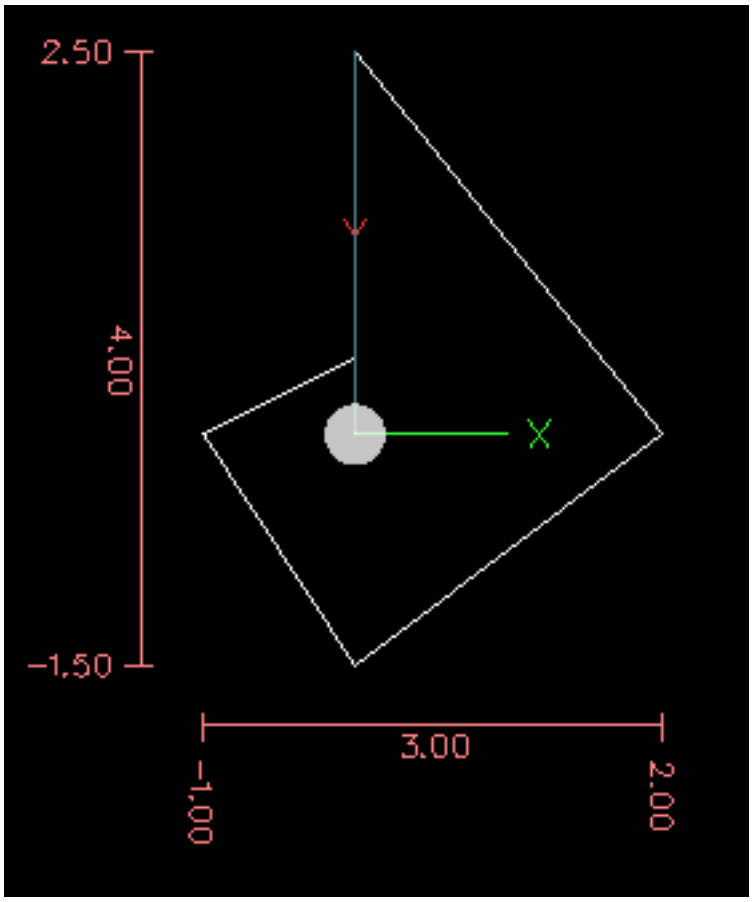


Abbildung 11.10: Polare Spirale

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.

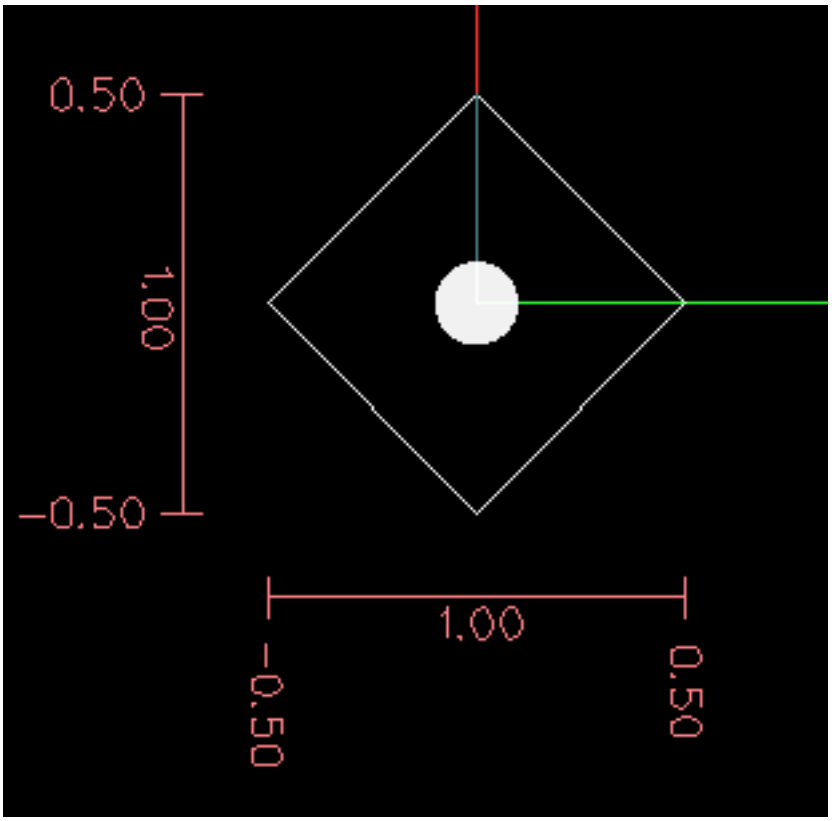


Abbildung 11.11: Polares Quadrat

Es ist ein Fehler, wenn:

- Eine inkrementelle Bewegung wird am Ursprung gestartet
- Eine Mischung aus Polar und X- oder Y-Wörtern wird verwendet

11.4.13 Modale Gruppen

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.

Tabelle 11.7: G-code Modal Groups

Bedeutung der Modalgruppe	**Member-Wörter**
Nicht-modale Codes (Gruppe 0)	G4, G10 G28, G30, G52, G53, G92, G92.1, G92.2, G92.3,
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

Tabelle 11.7: (continued)

Bedeutung der Modalgruppe	**Member-Wörter**
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, G49
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

Tabelle 11.8: M-code Modal Groups

Bedeutung der Modalgruppe	**Member-Wörter**
Anhalten (Gruppe 4)	M0, M1, M2, M30, M60
Ein/Aus E/A (*Gruppe 5*)	FIXME M6 T*n*
Werkzeugwechsel (*Gruppe 6*)	M6 T*n*
Spindel (Gruppe 7)	M3, M4, M5
Kühlmittel (Gruppe 8)	(M7 und M8 können beide eingeschaltet sein), M9
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.

11.4.14 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.

Anmerkung
Inline-Kommentare zu O-Wörtern sollten nicht verwendet werden, siehe den Abschnitt O-Code Kommentare für weitere Informationen.

11.4.15 Meldungen

- *(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)
```

11.4.16 Prüfpunkt-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.

11.4.17 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.ng* und *nc_files/ngcgui_lib/rectange_probe.ngc*.

11.4.18 Debug-Meldungen

- *(DEBUG,)* - zeigt eine Meldung wie *(MSG,)* an, mit dem Zusatz einer besonderen Behandlung von Kommentarparametern, wie unten beschrieben.

11.4.19 Meldungen drucken

- *(PRINT,)* - Meldungen werden auf *stderr* ausgegeben, wobei Kommentarparameter wie unten beschrieben besonders behandelt werden.

11.4.20 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.

11.4.21 Dateianforderungen

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.

Warnung
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.

Anmerkung
Die Datei muss mit einem Texteditor wie Gedit erstellt werden und nicht mit einem Textverarbeitungsprogramm wie Open Office Word Processor.

11.4.22 Dateigröße((Einzelgröße)

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.

11.4.23 G-Code Reihenfolge der Ausführung

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-Wort-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, G94).
- Bewegung ausführen (G0 bis G3, G33, G38.n, G73, G76, G80 bis G89), eventuell modifiziert durch G53.
- Stopp (M0, M1, M2, M30, M60).

11.4.24 Bewährte Verfahren für den G-Code

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.

11.4.25 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.

11.4.26 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-Kurzübersicht

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	NURBS, Kontrollpunkt hinzufügen
G7	Durchmesser-Modus (Drehmaschine)
G8	Radius-Modus (Drehmaschine)

Code	Beschreibung
G10 L0	Werkzeug-Tabellendaten neu laden
G10 L1	Werkzeugtabelleneintrag festlegen
G10 L10	Bestimme Werkzeugtabelle, Berechnet, Werkstück
G10 L11	Bestimme Werkzeugtabelle, Berechnet, Spannmittel
G10 L2	Festlegung des Koordinatensystem-Ursprungs
G10 L20	Ursprungseinstellung des Koordinatensystems berechnet
G17 - G19.1	Ebene auswählen
G20 G21	Maßeinheiten festlegen
G28 - G28.1	Zur vordefinierten Position gehen
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
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
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

```
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.

11.5.3.1 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.

Warnung
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.

11.5.5.1 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

11.5.5.2 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)
```

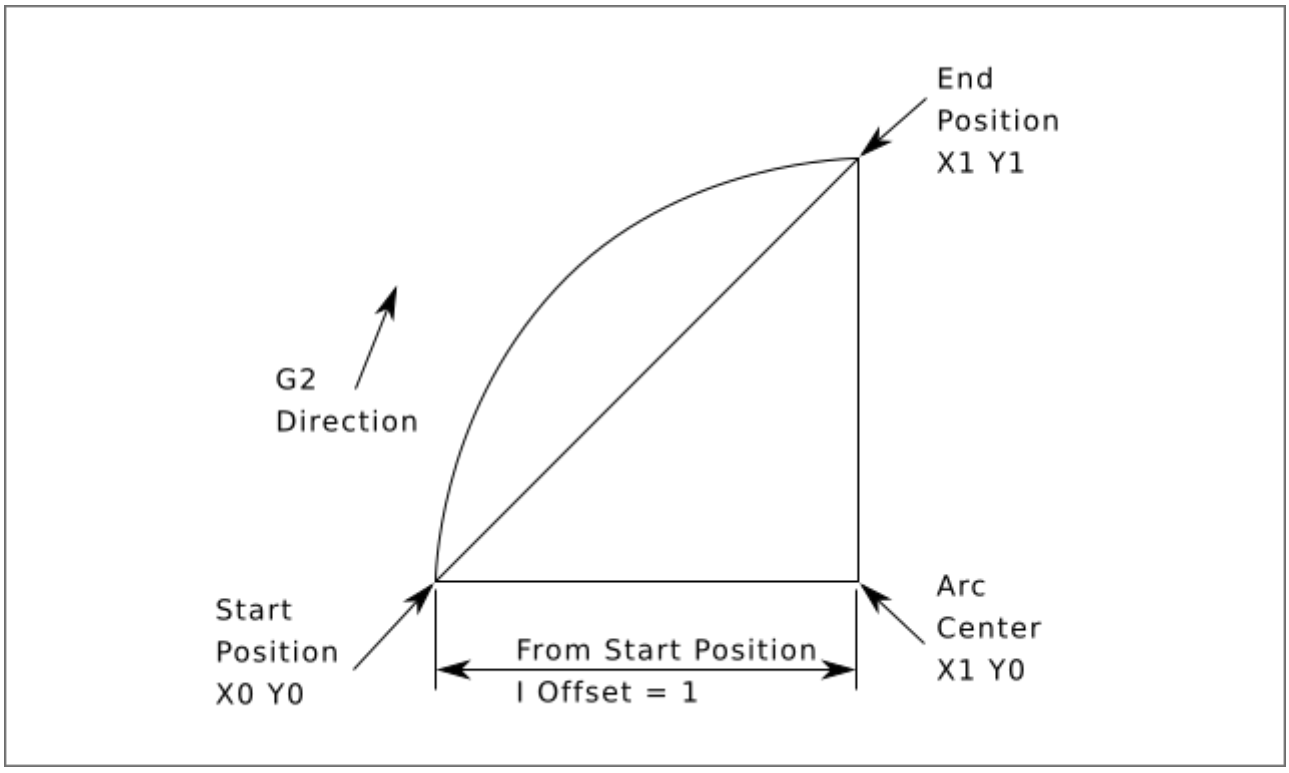


Abbildung 11.12: G2-Beispiel

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)
```

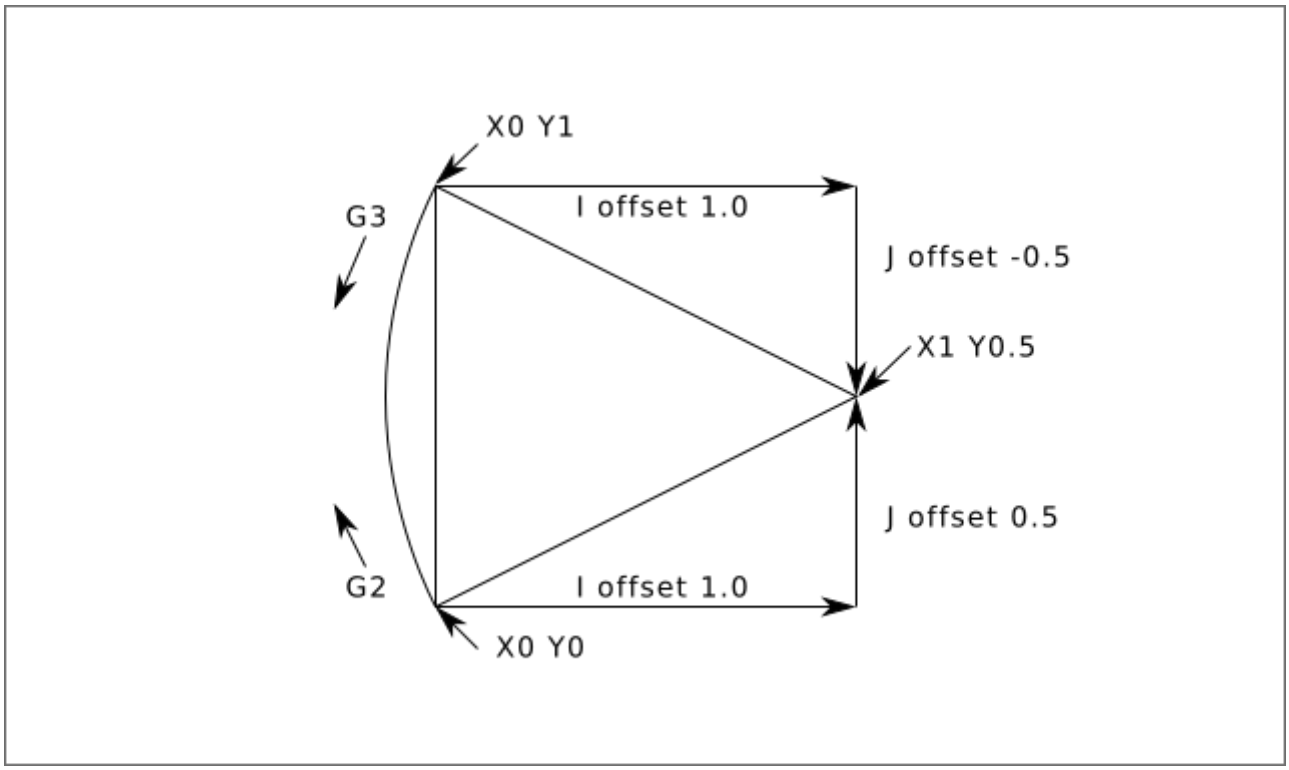


Abbildung 11.13: G2-G3 Beispiel

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.

11.5.5.3 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 Verweilzeit (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
```

Warnung
G5.2, G5.3 ist experimentell und nicht vollständig getestet.

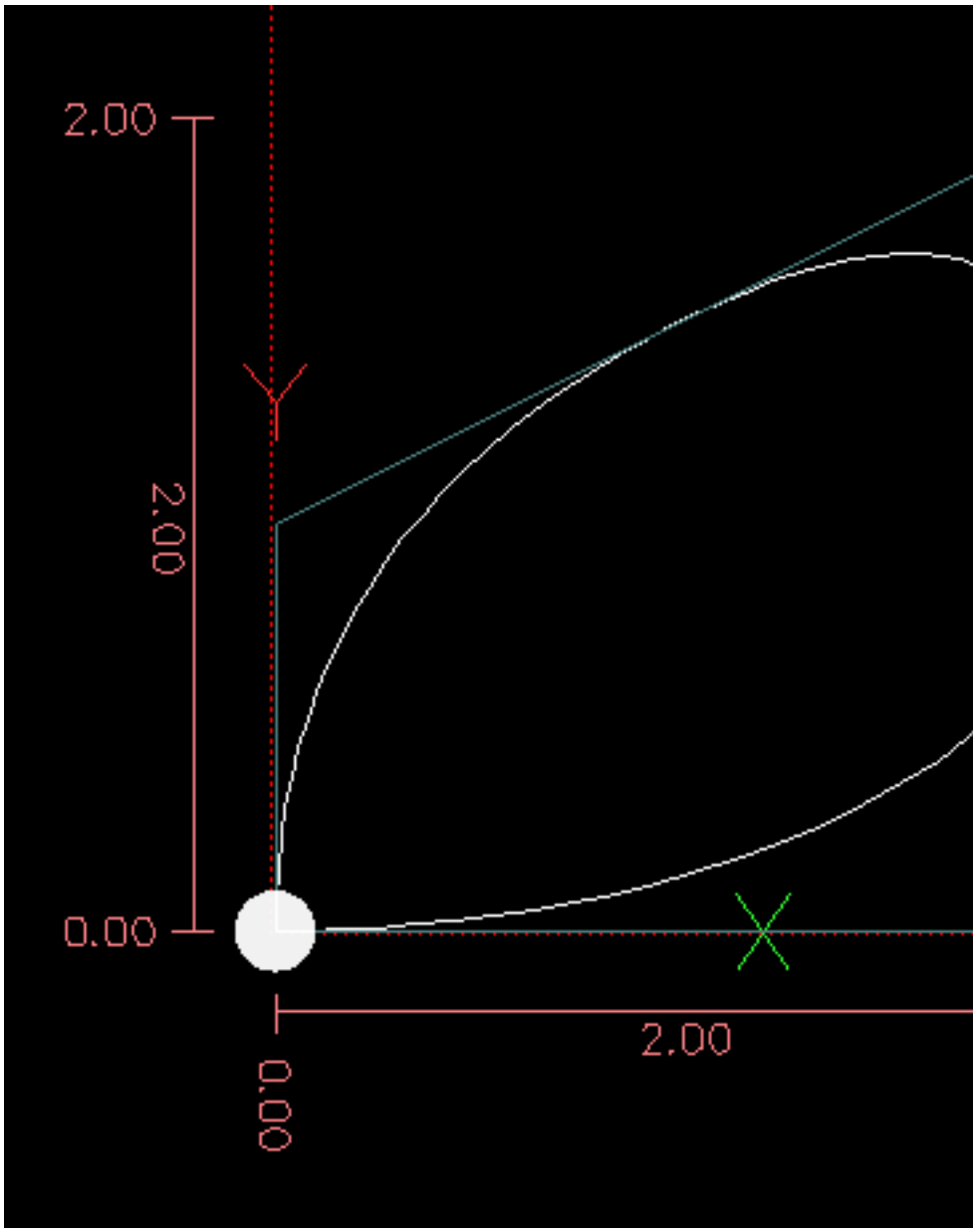
G5.2 dient zum Öffnen des Datenblocks, der ein NURBS definiert, und G5.3 zum Schließen des Datenblocks. In den Zeilen zwischen diesen beiden Codes werden die Kurvenkontrollpunkte mit ihren zugehörigen ”Gewichten” (P) und dem Parameter (L), der die Reihenfolge der Kurve bestimmt, definiert.

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:

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS

11.5.10 G7-Drehdurchmesser-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-Drehradius-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 Werkzeugtabellendaten neu laden

```
G10 L0
```

G10 L0 Alle Daten der Werkzeugtabelle neu laden. Erfordert, dass kein aktuelles Werkzeug in der Spindel geladen ist.

Anmerkung
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 einstellen

```
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 setzen

```
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.

Tabelle 11.9: 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 Bestimme Werkzeugtabelle

```
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 <gcode:g38,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 einstellen

```
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 einstellen

```
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 G17 - G19.1 Ebenenauswahl

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.

11.5.19 G20, G21 Einheiten (engl. units)

- *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.

11.5.20 G28, G28.1 Gehe zu/Bestimme Vordefinierte Position

Warnung
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 <sub:numbered-parameters,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 *gcode:g0,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

11.5.21 G30, G30.1 Gehe zu/Bestimme Vordefinierte Position

Warnung
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 <sub:numbered-parameters,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.

Anmerkung
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 *gcode:g0,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 #5i83 angegebenen Z-Position)
```

Es ist ein Fehler, wenn :

- Fräserausgleich (engl. cutter compensation) aktiviert ist

11.5.22 G33 Spindelsynchronisierte Bewegung

```
G33 X- Y- Z- K- $-
```

- *K* - Weg pro Umdrehung

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.

Alle Achsenwörter sind optional, außer dass mindestens eines verwendet werden muss.

Anmerkung
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.

Technische Informationen

Zu Beginn eines jeden G33-Durchlaufs nutzt LinuxCNC die Spindeldrehzahl- und die Maschinen-Beschleunigungs-Begrenzungen, um zu berechnen, wie lange es dauern wird Z nach dem die Index-Impuls zu beschleunigen, und bestimmt, wieviel Grad die Spindel sich während dieser Zeit zu drehen wird. Dann addiert es diesen Winkel zur Indexposition und berechnet die Z-Position unter Verwendung des korrigierten Spindelwinkels. Das bedeutet, dass Z die korrekte Position erreicht, sobald es die Beschleunigung auf die richtige Geschwindigkeit beendet hat, und sofort mit dem Schneiden eines guten Gewindes beginnen kann.

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.
- Die Spindel dreht sich nicht, wenn dieser Befehl ausgeführt wird.
- Die angeforderte lineare Bewegung überschreitet die Geschwindigkeitsgrenzen der Maschine aufgrund der Spindeldrehzahl.

11.5.23 G33.1 Starres Gewindeschneiden

```
G33.1 X- Y- Z- K- I- $-
```

- *K* - Weg pro Umdrehung
- I” - optionaler Multiplikator der Spindeldrehzahl für schnelleren Rücklauf

- *$* - optionaler Spindelselektor

Warnung
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.

Es ist ein Fehler, wenn:

- Alle Achsenwörter werden weggelassen.
- Die Spindel dreht sich nicht, wenn dieser Befehl ausgeführt wird.
- Die angeforderte lineare Bewegung überschreitet die Geschwindigkeitsgrenzen der Maschine aufgrund der Spindeldrehzahl.

11.5.24 G38.n Gerade Sonde

```
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

Wichtig
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.

Tabelle 11.10: 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.

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

11.5.25 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.

11.5.26 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.

Anmerkung
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.

11.5.27 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.

11.5.28 G43 Werkzeuglängen-Offset((G43 Werkzeuglängen-Offset)

```
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.

Anmerkung
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”)

11.5.29 G43.1 Dynamischer Werkzeuglängen-Offset

```
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 (aktuelle Werkzeugkorrektur um 0.250 korrigieren, DRO zeigt jetzt Z1.250 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

Anmerkung
G43.1 schreibt nicht in die Werkzeugtabelle.

11.5.30 G43.2 Zusätzlicher Werkzeuglängenversatz anwenden

```
G43.2 H- Achsen-
```

- *H* - Werkzeugnummer
- *G43.2* - wendet einen zusätzlichen simultanen Werkzeug-Offset an.

G43.2 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)
```

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.

Anmerkung
G43.2 schreibt nicht in die Werkzeugtabelle.

11.5.31 G49 Werkzeuglängenkorrektur abbrechen

- *G49* - hebt die Werkzeuglängenkompensation auf

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.

11.5.32 G52 Offset des lokalen Koordinatensystems

```
G52-Achsen
```

G52 wird in einem Werkstüc-Programm als temporärer ”lokaler Koordinatensystem-Offset” innerhalb des Werkstückkoordinatensystems verwendet. Weitere Informationen zu G52 finden Sie im Abschnitt Lokale und globale Offsets.

11.5.33 G53 Bewegung 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.

11.5.34 G54-G59.3 Auswahl des Koordinatensystems

- *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.

Tabelle 11.11: 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.

11.5.35 G61 Genauer Pfadmodus

- 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.

11.5.36 G61.1 Exakter Stoppmodus

- G61.1’ - Exakter Stoppmodus, die Bewegung wird am Ende jedes programmierten Segments angehalten.

11.5.37 G64 Pfad-Übergänge

```
G64 <P- <Q->>
```

- *P* - Toleranz für Bewegungs-Übergänge
- *Q* - naive Nockentoleranz
- *G64* - bestmögliche Geschwindigkeit. Ohne P 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- >’ mit Toleranz überlagern. Auf diese Weise können Sie Ihr System feinabstimmen, um den besten Kompromiss zwischen Geschwindigkeit und Genauigkeit zu finden. Die P- Toleranz bedeutet, dass die tatsächliche Bahn nicht weiter als P- vom programmierten Endpunkt entfernt sein darf. Die Geschwindigkeit wird bei Bedarf reduziert, um die Bahn zu halten. Wenn Sie G64 P- Q- aktivieren, wird außerdem der *Naive CAM-Detektor* eingeschaltet; wenn es eine Reihe von linearen XYZ-Vorschubbewegungen mit der gleichen feed rate gibt, die weniger als Q- davon entfernt sind,

kollinear zu sein, werden sie zu einer einzigen linearen Bewegung zusammengefasst. Bei G2/G3-Bewegungen in der G17 (XY)-Ebene, bei denen die maximale Abweichung eines Bogens von einer geraden Linie kleiner ist als die G64 P-Toleranz, wird der Bogen in zwei Linien unterteilt (vom Bogenanfang zum Mittelpunkt und vom Mittelpunkt zum Ende). Diese Linien werden dann dem naiven Nockenalgorithmus für Linien unterzogen. Auf diese Weise profitieren sowohl Linien-Bogen-, Bogen-Bogen- und Bogen-Linien-Fälle als auch Linien-Linien von dem ”naiven Nocken-Detektor”. Dies verbessert die Konturierungsleistung durch Vereinfachung des Pfades. Es ist in Ordnung, für den Modus zu programmieren, der bereits aktiv ist. Siehe auch den Abschnitt Trajectory Control für weitere Informationen über diese Modi. Wenn Q nicht angegeben wird, verhält es sich wie zuvor und verwendet den Wert von P-.

G64 P- Beispielzeile

```
G64 P0.015 (stellt die Bahnverfolgung so ein, dass sie innerhalb von 0,015 der ↩
    tatsächlichen Bahn liegt)
```

Es empfiehlt sich, in die Präambel jeder G-Code-Datei eine Pfadsteuerungsangabe aufzunehmen.

11.5.38 G70 Drehmaschinen-Finishing-Zyklus

```
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 Abschnitt 11.5.5 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 Abschnitt 11.5.39 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.
- Abschnitt 11.5.18 wurde nicht zur Auswahl der ZX-Ebene verwendet.

11.5.39 G71 G72 Schruppzyklen auf der Drehmaschine

```
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 Abschnitt 11.5.3 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.
- Abschnitt 11.5.18 wurde nicht zur Auswahl der ZX-Ebene verwendet.
- Abschnitt 11.5.26 ist aktiv.

11.5.40 G73 Bohrzyklus mit Spanbrecher

```
G73 X- Y- Z- R- Q- <L->
```

- *R* - Rückzugsposition entlang der Z-Achse.
- *Q* - Delta-Inkrement entlang der Z-Achse.
- *L* - wiederholen

Der ”G73”-Zyklus ist Bohren oder Fräsen mit Spanbruch. Dieser Zyklus nimmt eine Q-Zahl, die ein ”Delta”-Inkrement entlang der Z-Achse darstellt.

- 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.
- Schnelles Aufsteigen (engl. rapid up) um 0,010 Zoll oder 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

11.5.41 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)).

Warnung
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.

11.5.42 G76 Gewindeschneidzyklus

```
G76 P- Z- I- J- R- K- Q- H- E- L- $-
```

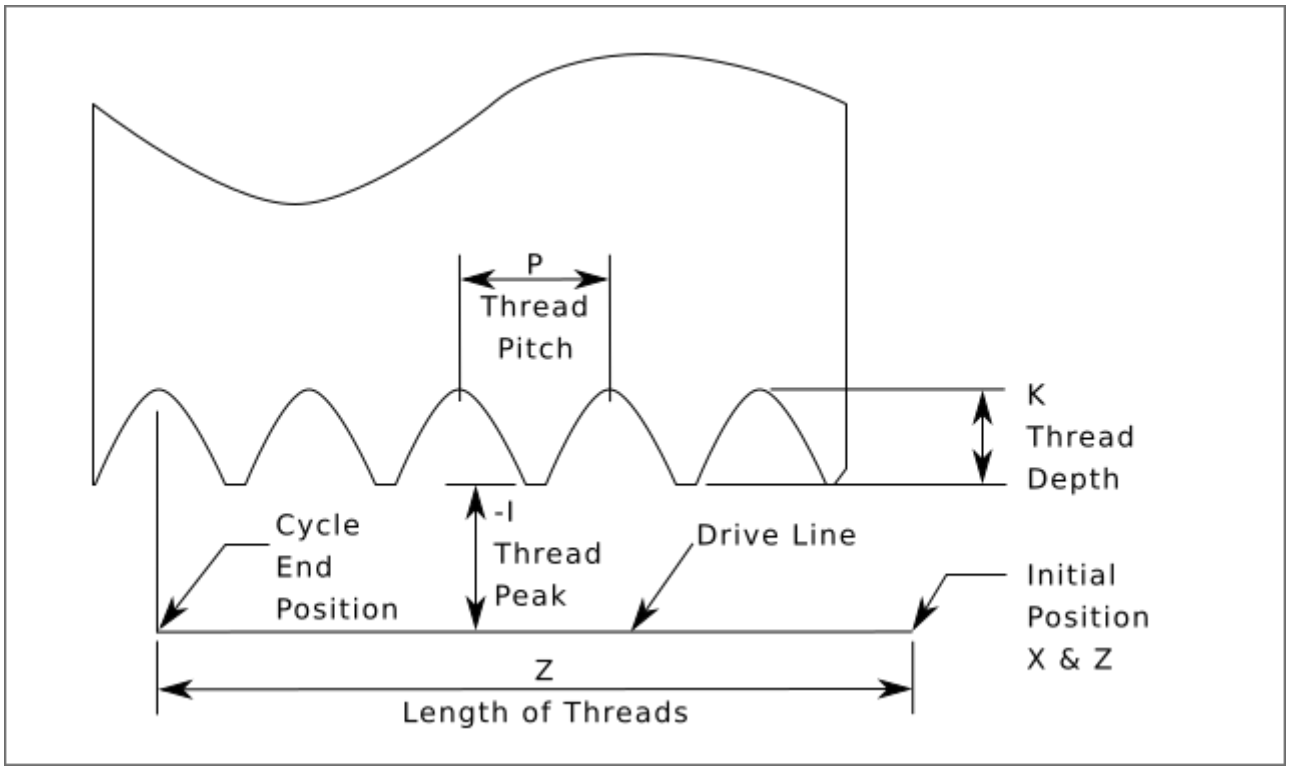


Abbildung 11.14: G76 Gewindeschneiden

- *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.

Anmerkung
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)

Warnung
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*.

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.

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.

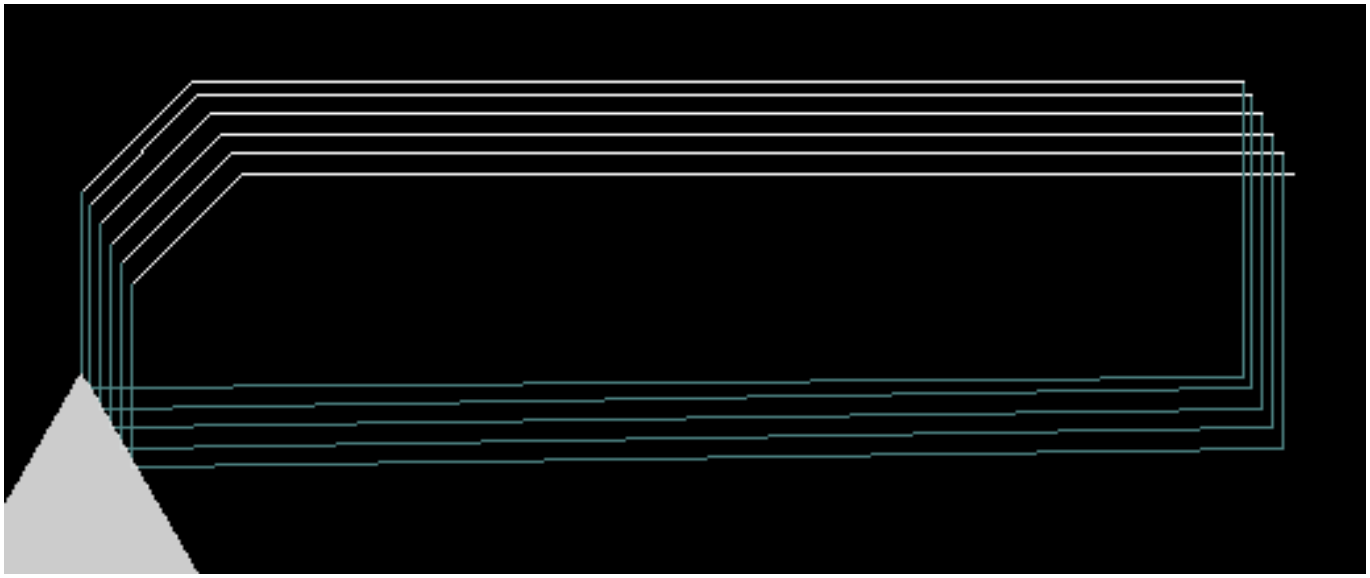


Abbildung 11.15: G76 Beispiel

11.5.43 G80-G89 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.

11.5.43.1 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.

11.5.43.2 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.

11.5.43.3 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.

11.5.43.4 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.

11.5.43.5 Festzyklusfehler

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.

11.5.43.6 Vorläufige und zwischenzeitliche Bewegung

Die Vorbewegung ist eine Reihe von Bewegungen, die für alle feststehenden Fräszyklen gelten. 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.

11.5.43.7 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.

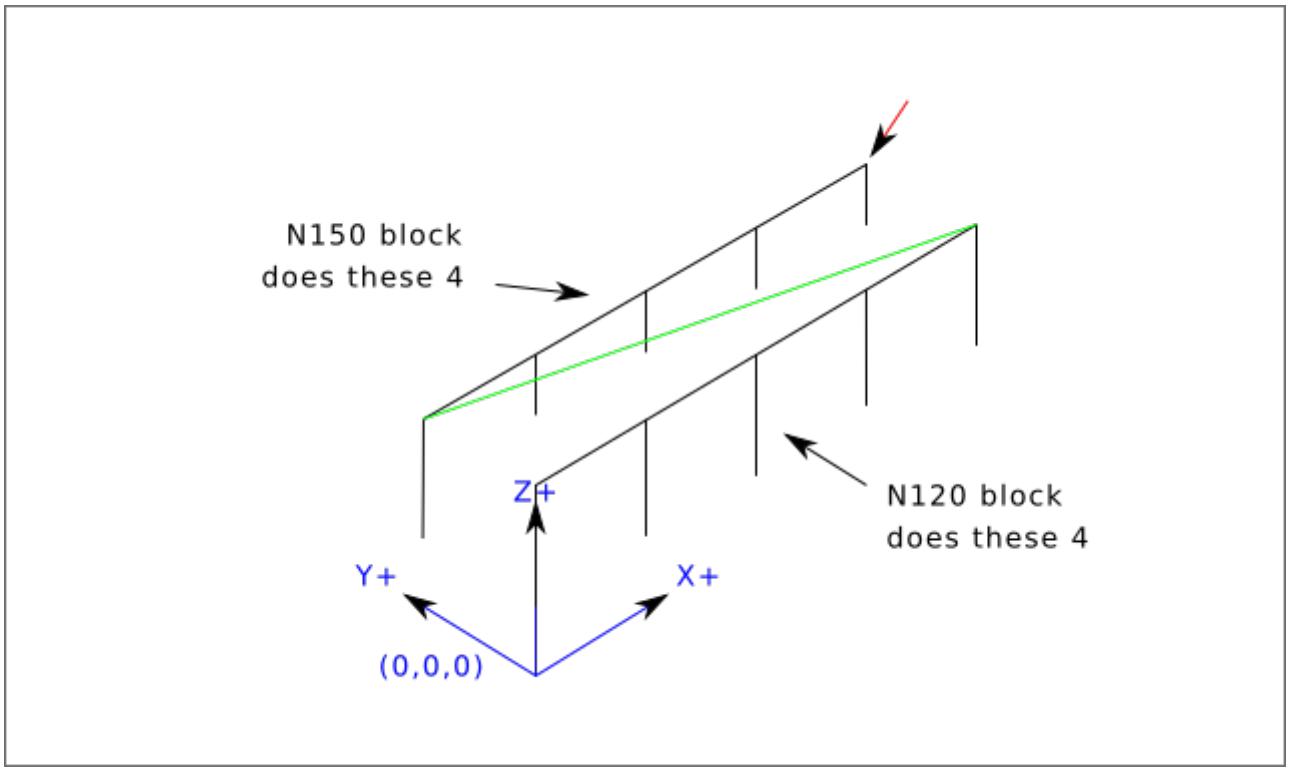
Die G81 Example 1 demonstriert, wie ein Festzyklus verwendet werden kann, um 8 Löcher mit zehn Zeilen G-Code im Festzyklusmodus zu erzeugen. Das folgende Programm erzeugt den gleichen Satz von 8 Löchern mit fünf Zeilen für den Festzyklus. Es folgt nicht genau dem gleichen Pfad und bohrt auch nicht in der gleichen Reihenfolge wie das frühere Beispiel. Aber die Wirtschaftlichkeit eines guten Festzyklus beim Programmieren sollte offensichtlich sein.

Anmerkung
Zeilennummern sind nicht erforderlich, dienen aber der Verdeutlichung dieser Beispiele.

Acht Löcher

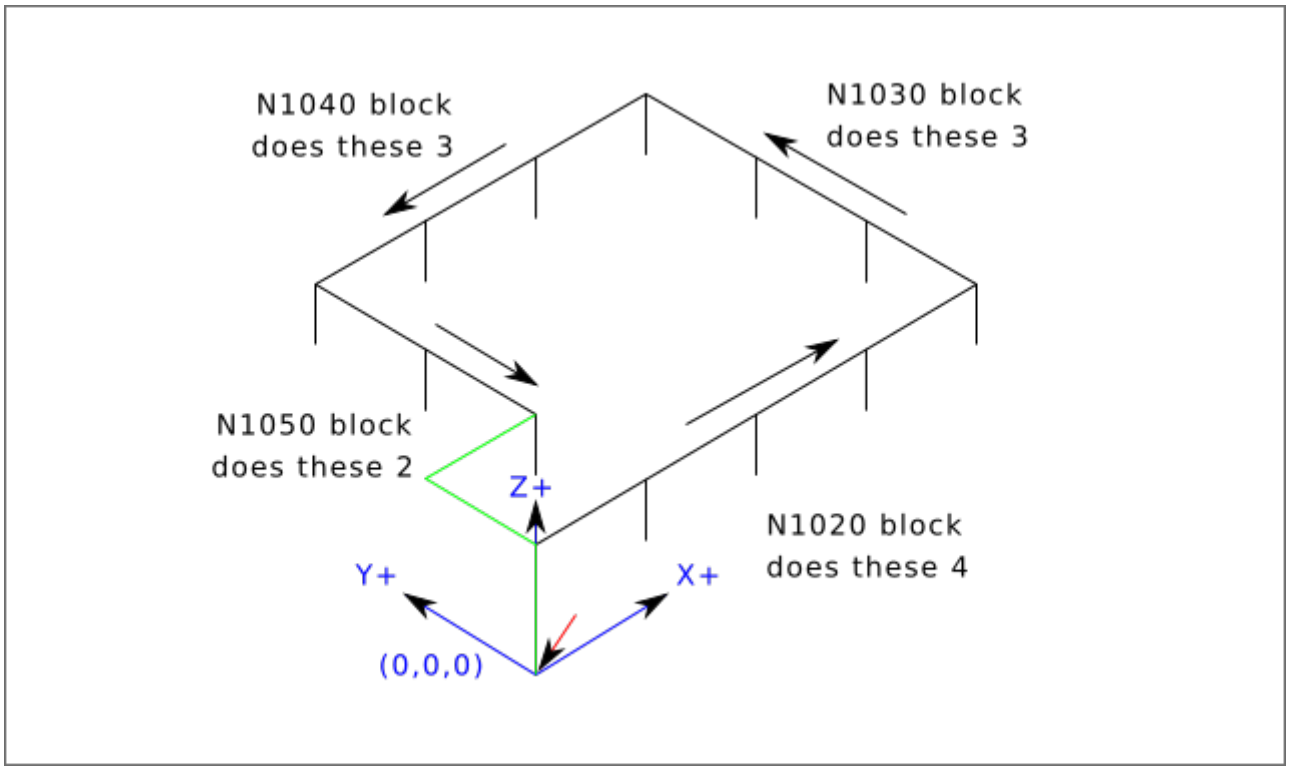
```
N100 G90 G0 X0 Y0 Z0 (Koordinate zum Referenzpunkt setzen)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4 (Festbohrzyklus)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0,5 R1 L4(Festbohrzyklus)
N160 G80 (Ausschalten des Festzyklus)
N170 M2 (Programmende)
```

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.

11.5.44 G80 Festzyklus (engl. canned cycle) 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)
```

Anmerkung
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.

11.5.45 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.

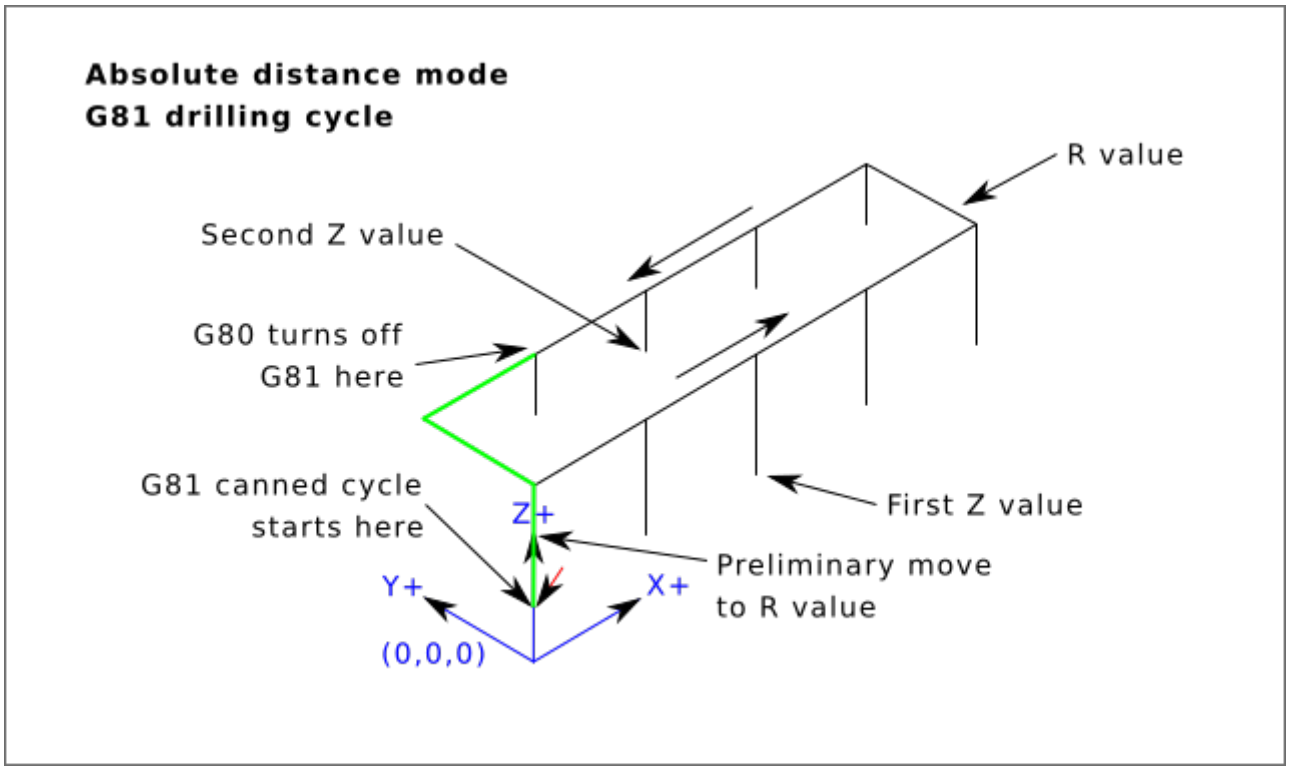


Abbildung 11.16: G81 Zyklus

Beispiel 1 – Absolute Position G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

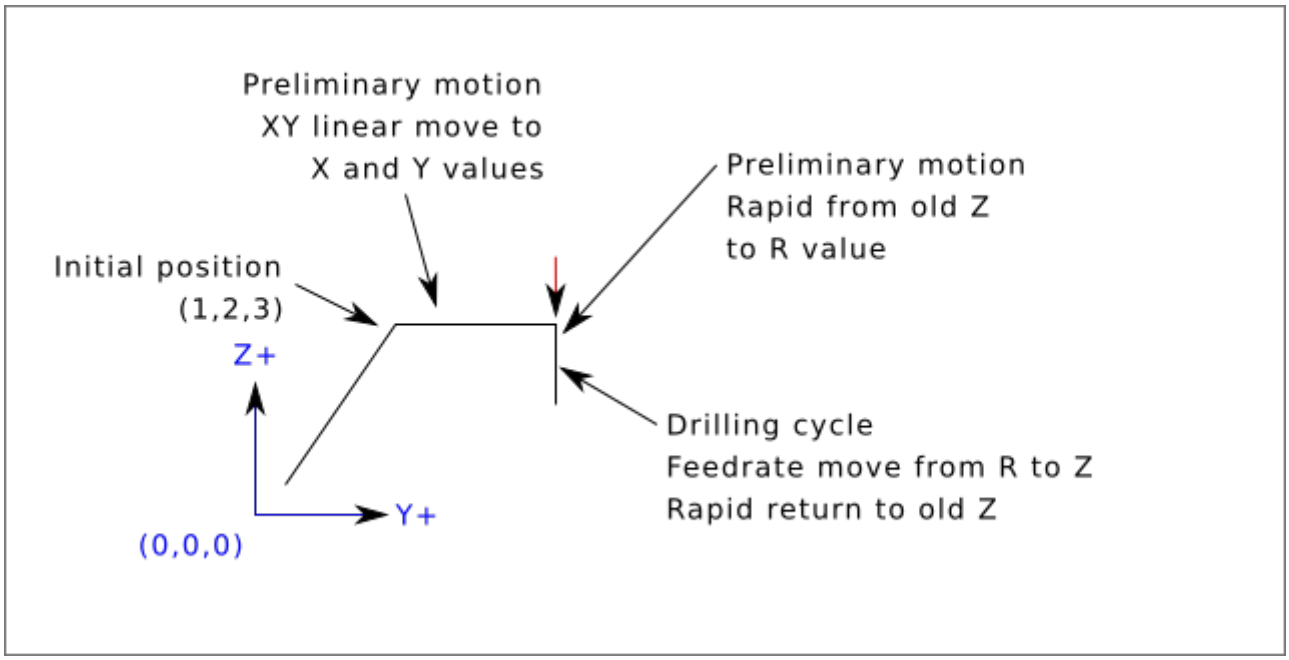
Angenommen, die aktuelle Position ist (X1, Y2, Z3) und die vorangehende Zeile des NC-Codes wird interpretiert.

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)

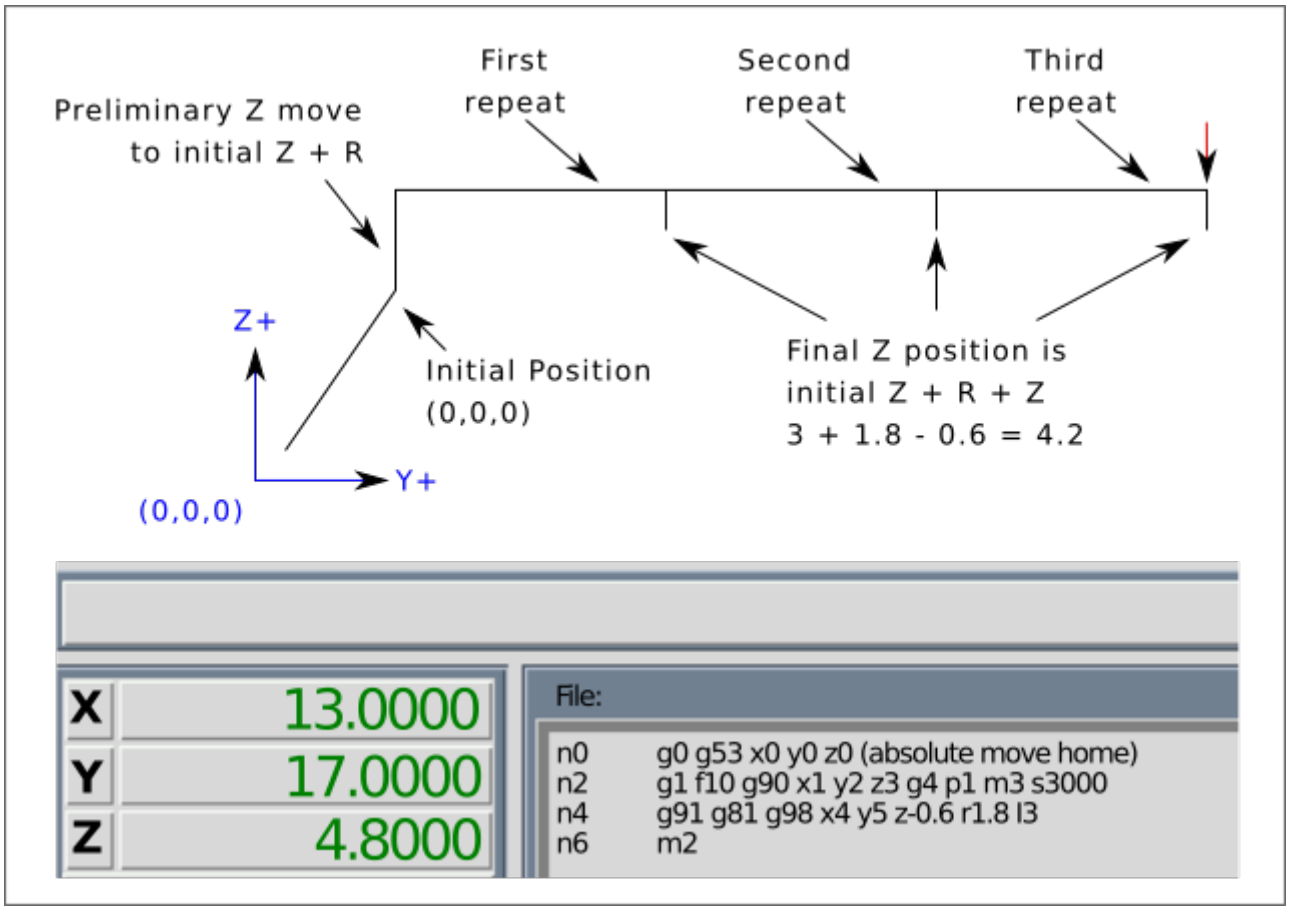
Die zweite Wiederholung besteht aus 3 Zügen. Die X-Position wird auf 9 (=5+4) und die Y-Position auf 12 (=7+5) zurückgesetzt.

- 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)

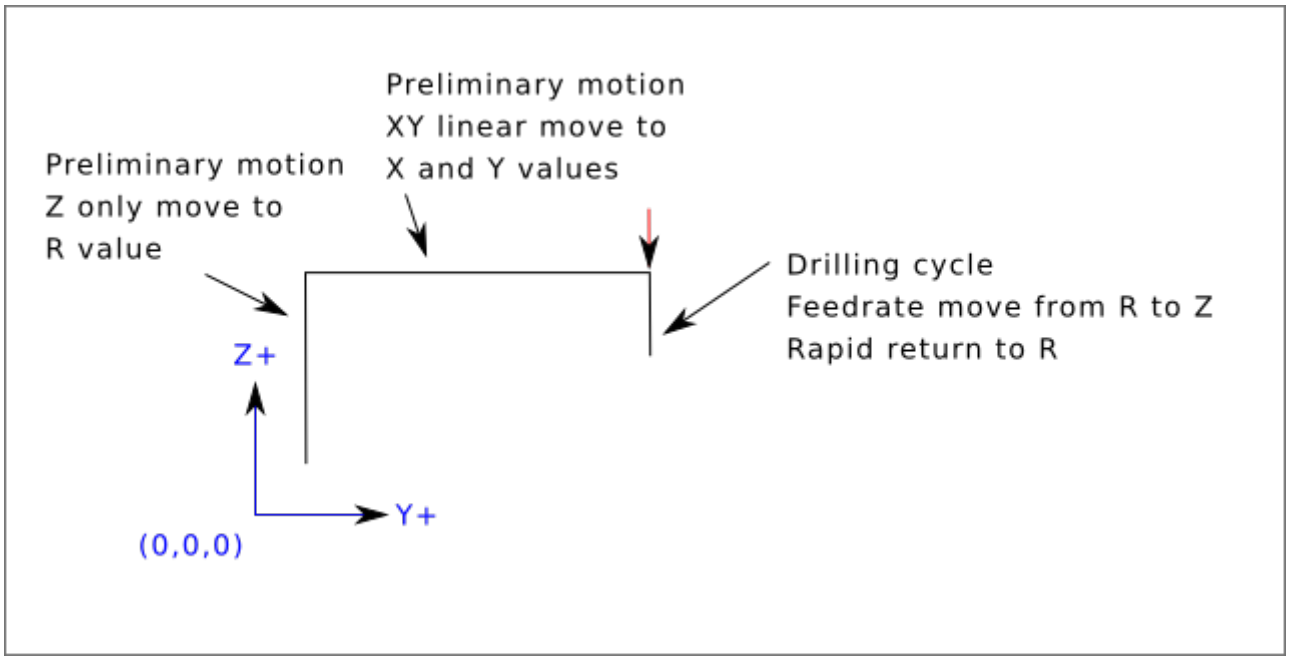


Beispiel 3 - Relative Position G81

```
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.

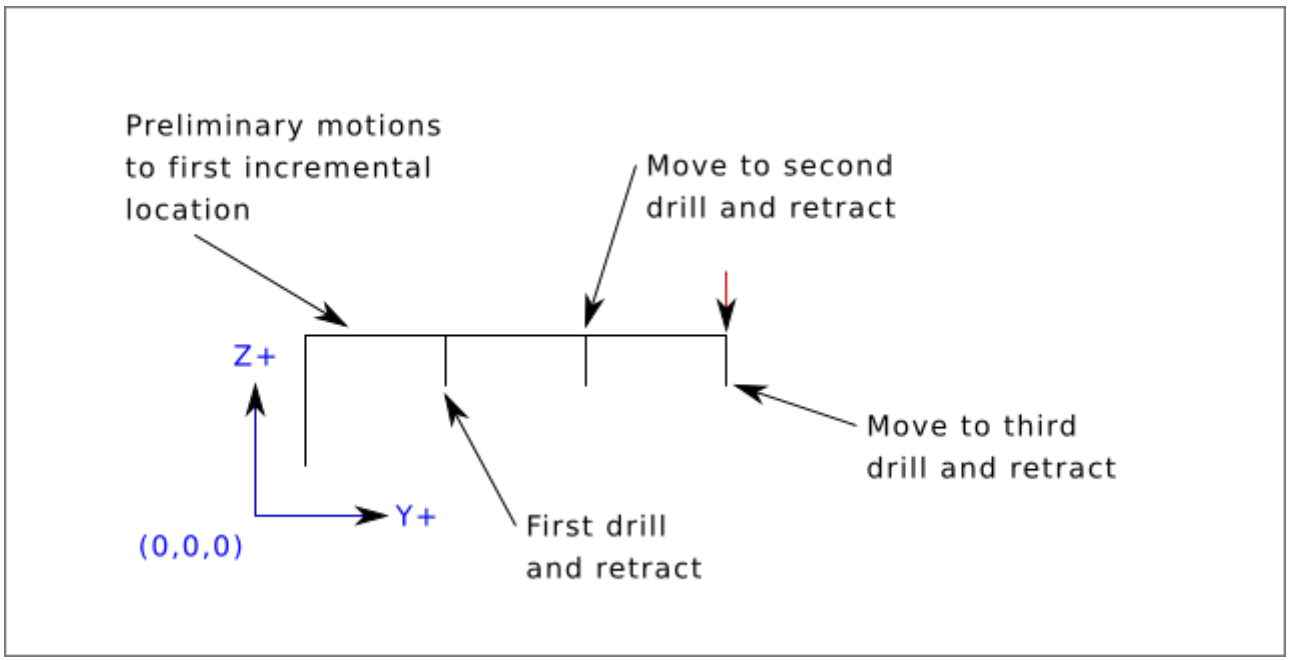
Da OLD_Z unter dem R-Wert liegt, fügt es der Bewegung nichts hinzu, aber da der Anfangswert von Z kleiner als der in R angegebene Wert ist, wird es während der vorbereitenden Bewegungen eine anfängliche Z-Bewegung geben.



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.

11.5.46 G82 Bohrzyklus, Verweilzeit

```
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.

11.5.47 G83 Tiefbohrzyklus mit Spanbruch und Entspänen (engl. peck drilling cycle)

```
G83 (X- Y- Z-) or (U- V- W-) R- L- Q-
```

Der *G83*-Zyklus (oft als Tieflochbohren bezeichnet) ist für das Tiefbohren oder Fräsen mit Spanbruch vorgesehen. Die Rückzüge in diesem Zyklus befreien das Loch von Spänen und schneiden alle langen Spänen ab (die beim Bohren in Aluminium üblich sind). Dieser Zyklus nimmt eine Q-Zahl an, die ein ”Delta”-Inkrement entlang der Z-Achse darstellt. Der Rückzug vor der Endtiefe erfolgt immer in die Rückzugsebene, auch wenn G98 aktiviert ist. Der endgültige Rückzug richtet sich nach den geltenden G98/99. G83 funktioniert wie G81 mit dem Zusatz, dass während des Bohrvorgangs zurückgezogen wird.

- 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.
- Bewegen Sie sich im Eiltempo zurück zum aktuellen Bohrungsgrund, abzüglich 0,254 mm oder 0,010 Zoll.
- 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.

11.5.48 G84 Gewindebohrzyklus (engl. right-hand tapping cycle, dwell)

```
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)).

Warnung
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.

11.5.49 G85 Bohrzyklus, Vorschub aus

```
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.

11.5.50 G86 Bohrzyklus, Spindelstopp, Eilgang

```
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.

11.5.51 G87 Rückwärtsbohrzyklus

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

11.5.52 G88 Bohrzyklus, Spindelanschlag, manueller Ausgang

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

11.5.53 G89 Bohrzyklus, Verweilzeit, mit Vorschubgeschwindigkeit zurück

```
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.

11.5.54 G90, G91 Distanzmodus

- *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.

11.5.55 G90.1, G91.1 Bogenabstandsmodus

- *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.

11.5.56 G92 Koordinatensystem-Offset

```
G92 Achsen
```

Warnung
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).

Anmerkung
Der Befehl *G52* kann auch verwendet werden, um diesen Offset zu ändern; siehe den Abschnitt Offsets 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.

11.5.57 G92.1, G92.2 G92 Offsets zurücksetzen

- *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.

Anmerkung
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.

11.5.58 G92.3 Wiederherstellung von G92-Offsets

- *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.

11.5.59 G93, G94, G95 Vorschubmodus

- *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 wird keine neue Vorschubgeschwindigkeit angegeben

11.5.60 G96, G97 Spindelsteuerungsmodus

```
G96 <D-> S- <$-> (Modus konstante Oberflächengeschwindigkeit)
G97 S- <$-> (Drehzahlmodus)
```

1. *D* - maximale Drehzahl (RPM), optional
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
- Eine Vorschubbewegung wird im G96-Modus festgelegt, während sich die Spindel nicht dreht

11.5.61 G98, G99 Festzyklusrücklauf Ebene

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 Kurzübersichtstabelle

Code	Beschreibung
M0 M1	Programm-Pause
M2 M30	Programmende
M60	Palettenwechsel Pause
M3 M4 M5	Spindelsteuerung
M6	Werkzeugwechsel
M7 M8 M9	Kühlmittelkontrolle
M19	Spindel ausrichten
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)

Code	Beschreibung
M67	Steuerung des Analogausgangs
M68	Steuerung des Analogausgangs
:mcode:m70,M70	Modalstatus speichern
M71	Gespeicherten Modalzustand ungültig machen
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 Programmpause

- 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.

Anmerkung
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.
- Ursprungs-Offsets sind auf den Standard eingestellt (wie *G54*).
- 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*).

Anmerkung
Codezeilen nach M2/M30 werden nicht ausgeführt. Durch Drücken von *Zyklusstart* wird das Programm am Anfang der Datei gestartet.

Warnung
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

11.6.6.1 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.

11.6.6.2 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.

Anmerkung
Das *T*-Wort ist eine ganze Zahl, welche die Werkzeugfachnummer (engl. tool pocket number) im Karussell bezeichnet (nicht seinen Index).

Warnung
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 Orientierung (engl. orient spindle)

```
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)
- $' Die auszurichtende Spindel (bestimmt eigentlich nur, welche HAL-Stifte die Spindelpositionsbefehle tragen)

M19 ist ein Befehl der Modalgruppe 7, wie M3, M4 und M5. M19 wird durch einen der Befehle M3, M4, M5 gelöscht.

Für die Spindelausrichtung ist ein Quadratur-Encoder mit einem Index erforderlich, der die Position und die Drehrichtung der Spindelwelle erfasst.

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* (out float) Gewünschte Spindelausrichtung für M19. Wert des M19 R-Wort-Parameters plus dem Wert des [RS274NGC]ORIENT_OFFSET INI-Parameters.
  - *spindle.N.orient-mode* (out s32) Gewünschter Spindeldrehungsmodus. Entspricht dem M19 P-Parameterwort, Voreinstellung = 0.
  - *spindle.N.orient* (out bit) Zeigt den Beginn des Spindelorientierungszyklus an. Wird von M19 gesetzt. Gelöscht durch einen der Befehle M3,M4,M5. Wenn *spindle-orient-fault* während *spindle-orient true* nicht Null ist, schlägt der Befehl M19 mit einer Fehlermeldung fehl.
  - *spindle.N.is-oriented* (in bit) Pin bestätigt die Spindelorientierung. Schließt den Orientierungszyklus ab. Wenn spindel-orient wahr war, als spindel-oriented überprüft wurde, wird der spindel-orient-Pin gelöscht und der spindle-locked Pin wird gesichert. Auch der spindle-brake Pin ist dann gesichert.
  - *spindle.N.orient-fault* (in s32) Eingang des Fehlercodes für den Orientierungszyklus. Jeder Wert ungleich Null führt zum Abbruch des Orientierungszyklus.
  - *spindle.N.locked* (out bit) Spindel Orientierung komplett-Pin. Wird durch einen der Parameter M3,M4,M5 gelöscht.

11.6.9 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.

11.6.10 M50 Vorschub-Neufestsetzungs-Steuerung (engl. feed override control)

- *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).

11.6.11 M51 Spindeldrehzahl-Neufestsetzuung (engl. override) Steuerung

- 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).

11.6.12 M52 Adaptive Vorschubregelung

- *M52 <P1>* - Verwendung eines adaptiven Vorschubs. Die Angabe P1 ist optional.
- *M52 P0* - Verwendung des adaptiven Feeds beenden.

Wenn der adaptive Vorschub aktiviert ist, wird ein externer Eingangswert zusammen mit dem Vorschub-Override-Wert der Benutzeroberfläche und der befohlenen Vorschubgeschwindigkeit verwendet, um die tatsächliche Vorschubgeschwindigkeit einzustellen. In LinuxCNC wird der HAL-Pin *motion.adaptive-feed* für diesen Zweck verwendet. Werte auf *motion.adaptive-feed* sollte im Bereich von -1 (programmierte Geschwindigkeit in umgekehrter Richtung) bis 1 (volle Geschwindigkeit). 0 ist gleichbedeutend mit feed-hold.

Anmerkung
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.

11.6.13 M53 Vorschub-Halt-Steuerung

- *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.

11.6.14 M61 Aktuelles Werkzeug setzen

- *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.

Warnung
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

11.6.15 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.

Das P-Wort gibt die Anzahl der digitalen Ausgänge an. Das P-Wort reicht von 0 bis zu einem Standardwert von 3. Bei Bedarf kann die Anzahl der E/A durch Verwendung des Parameters num_dio beim Laden des Motion Controllers erhöht werden. Siehe Abschnitt zu Bewegungen für weitere Informationen.

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.

Anmerkung
M62-65 funktionieren nur dann, wenn die entsprechenden motion.digital-out-*nn*-Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.16 M66 Warten auf Eingabe

```
M66 P- | E- <L->
```

- *P-* - specifies the digital input number from 0 to 3. (Adjustable from motmod argument num_dio)
- *E-* - specifies the analog input number from 0 to 3. (Adjustable from 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.

Anmerkung
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
```

11.6.17 M67 Analoger Ausgang,Synchronisiert

```
M67 E- Q-
```

- *M67* - stellt einen mit der Bewegung synchronisierten analogen Ausgang ein.
- *E-* - output number ranging from 0 to 3 (Adjustable from 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.

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.

Anmerkung
M67 funktioniert nur, wenn die entsprechenden motion.analog-out-*nn* Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.18 M68 Analogausgang, Sofort

```
M68 E-Q-
```

- *M68* - sofort einen analogen Ausgang einstellen.
- *E-* - output number ranging from 0 to 3. (Adjustable from 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.

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.

Anmerkung
M68 funktioniert nur dann, wenn die entsprechenden motion.analog-out-*nn* Pins in Ihrer HAL-Datei mit Ausgängen verbunden sind.

11.6.19 M70 Modalen Zustand 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 Gespeicherter ZustandM70 Gespeicherter 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)
- feed hold setting (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.

11.6.20 M71 Gespeicherten modalen Zustand 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.

11.6.21 M72 Wiederherstellung des modalen Zustands

Modaler Zustand 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
```

11.6.22 M73 Modaler Zustand speichern und automatisch wiederherstellen

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
```

11.6.23 M98 und M99

Der Interpreter unterstützt Haupt- und Unterprogramme im Fanuc-Stil mit den M-Codes *M98* und *M99*. Siehe Fanuc-Style Programme.

11.6.23.1 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
```

11.6.24 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.

Anmerkung
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. 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 M1nn-Programms sollte die GUI neu gestartet werden, damit das neue Programm berücksichtigt wird, da sonst ein *Unbekantner M-Code*-Fehler auftritt.

Warnung
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.

11.7.2 Nummerierung

Nummerierte O-Codes müssen für jedes Unterprogramm eine eindeutige Nummer haben,

Beispiel für eine Nummerierung

```
(der Beginn von o100)
o100 sub
(beachten Sie, dass der if-endif-Block eine andere Nummer verwendet)
  (der Anfang von o110)
  o110 if [#2 GT 5]
    (etwas Code hier)
  (das Ende von o110)
  o110 endif
  (etwas mehr Code hier)
(das Ende von o100)
o100 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:

- Die gleiche Nummer für mehr als einen Block verwendet wird.
- Andere Wörter in einer Zeile mit einem O-Wort verwendet werden.
- Kommentare in einer Zeile mit einem O-Wort verwendet werden.

Anmerkung
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.

11.7.4 Unterprogramme (engl. subroutines)

Unterroutinen beginnen mit *Onnn sub* und enden mit *Onnn endsub*. Die Zeilen zwischen *Onnn sub* 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]
```

Unterprogrammkörper dürfen nicht verschachtelt werden. Sie dürfen erst aufgerufen werden, nachdem sie definiert wurden. Sie können von anderen Funktionen aufgerufen werden und können sich selbst rekursiv aufrufen, wenn dies sinnvoll ist. Die maximale Verschachtelungsebene eines Unterprogramms ist 10.

Unterprogramme können den Wert von Parametern über #30 ändern, und diese Änderungen werden für den aufrufenden Code sichtbar. Unterprogramme können auch den Wert von globalen benannten Parametern ändern.

11.7.4.1 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_STY = 1` in den Abschnitt `[RS274NGC]` der INI-Datei deaktiviert werden.

Anmerkung
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 ”O”-Aufruf von rs274ngc 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 Schleifen

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

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 Wiederholen

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 (engl. expressions)
- Binäre Operatoren
- Funktionen

11.7.9 Aufrufen von Dateien

Um eine separate Datei mit einem Unterprogramm aufzurufen, geben Sie der Datei den gleichen Namen wie Ihrem Aufruf und fügen Sie ein sub und endsub in die Datei ein. Die Datei muss sich in dem Verzeichnis befinden, auf das *PROGRAM_PREFIX* oder *SUBROUTINE_PATH* in der INI-Datei verweist. Der Dateiname darf nur **Kleinbuchstaben**, Zahlen, Bindestriche und Unterstriche enthalten. Eine benannte Subroutinendatei kann nur eine einzige Subroutinendefinition enthalten.

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
```

Anmerkung
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*> <gcode:predefined-named-parameters,predefi 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 (engl. errors)

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

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: Vorschub 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 *gcode:g84, 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 <mcode:m6,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.

Anmerkung
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.

Anmerkung
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.

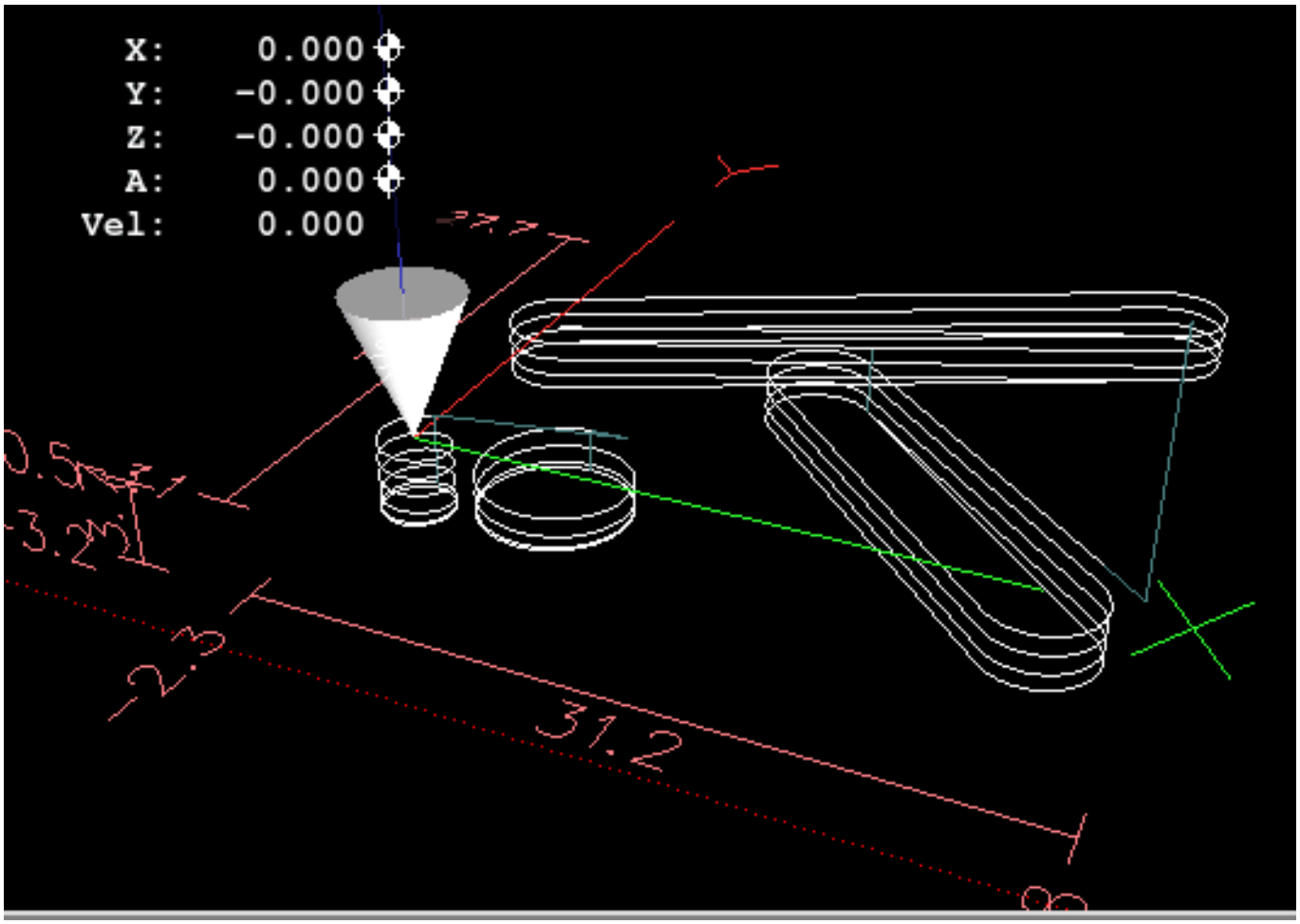
11.9.1 Beispiele für eine Fräsmaschine

11.9.1.1 Fräsen von Spiralbohrungen

- Dateiname: useful-subroutines.ngc
- Beschreibung: Unterprogramm zum Fräsen einer Bohrung mit Hilfe von Parametern.

11.9.1.2 Schlitzen (engl. slotting)

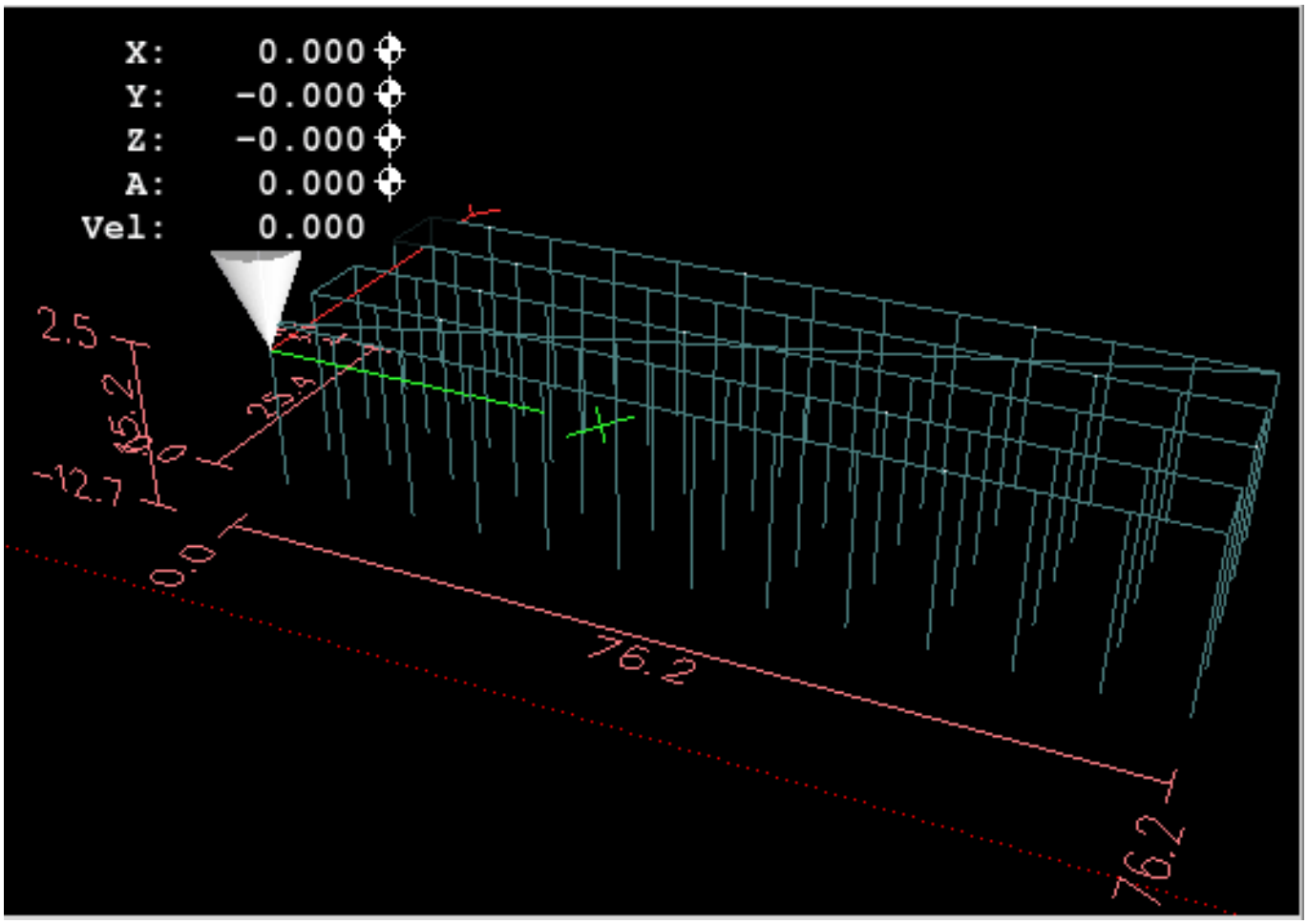
- Dateiname: useful-subroutines.ngc
- Beschreibung: Unterprogramm zum Fräsen einer Nut mit Parametern.



11.9.1.3 Rastersonde (engl. grid probe)

- Dateiname: gridprobe.ngc
- Beschreibung: Rechteckige Sondierung

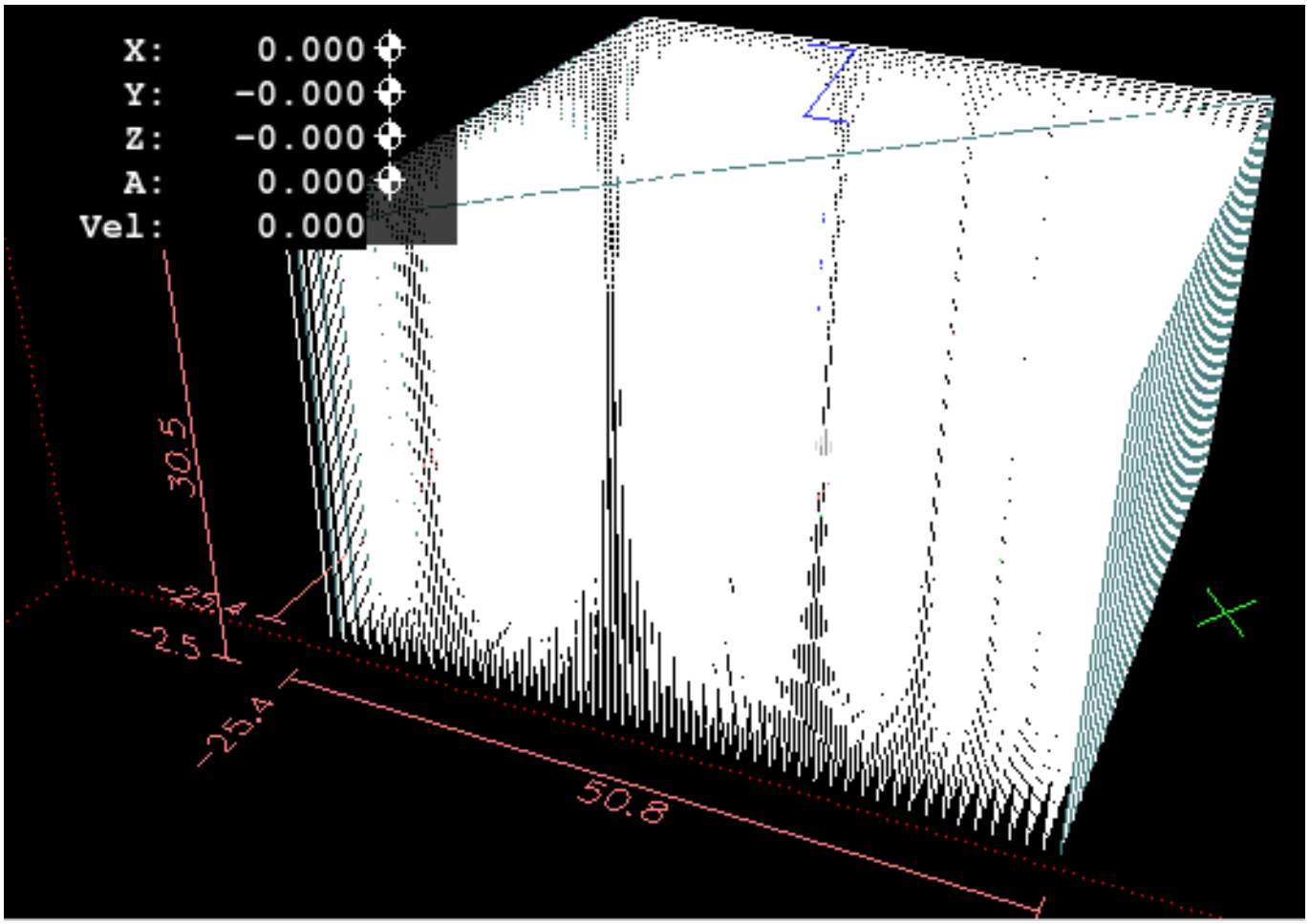
Dieses Programm testet wiederholt in einem regelmäßigen XY-Raster und schreibt die getestete Position in die Datei *probe-results.txt* im selben Verzeichnis wie die .ini-Datei.



11.9.1.4 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.



11.9.1.5 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.

11.9.1.6 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.

11.9.1.7 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.

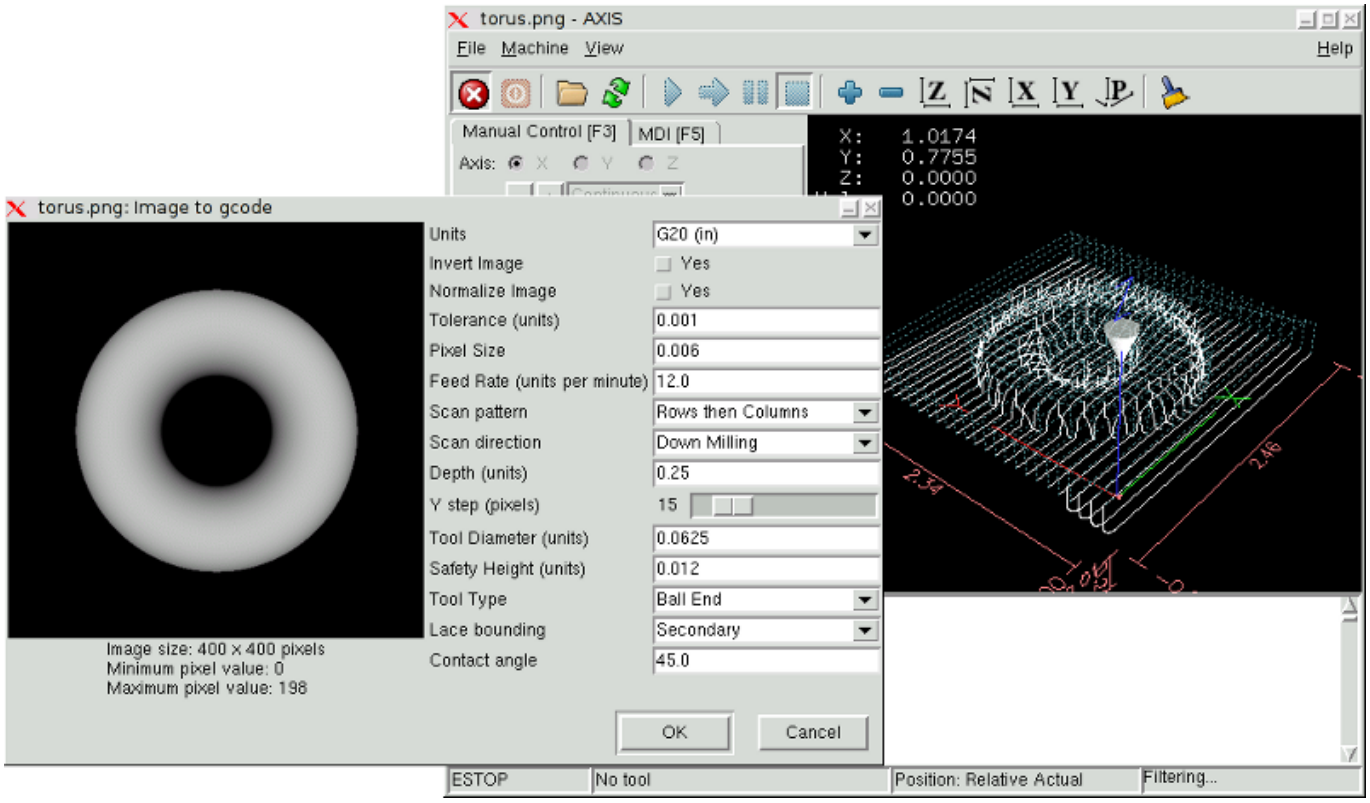
11.9.2 Beispiele für Drehmaschinen

11.9.2.1 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

11.10.4.1 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.

11.10.4.2 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.

11.10.4.3 Bild normalisieren (engl. normalize image)

Wenn ”ja”, wird das dunkelste Pixel auf Schwarz und das hellste Pixel auf Weiß umgestellt.

11.10.4.4 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.

11.10.4.5 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.

11.10.4.6 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.

11.10.4.7 Tauchvorschubgeschwindigkeit (Einheiten pro Minute)

Die Vorschubgeschwindigkeit für die erste Eintauchbewegung.

11.10.4.8 Vorschubgeschwindigkeit (Einheiten pro Minute)

Die Vorschubgeschwindigkeit für andere Teile der Bahn.

11.10.4.9 Spindeldrehzahl (RPM)

Der Spindeldrehzahl-S-Code, der in die G-Code-Datei eingefügt werden soll.

11.10.4.10 Abtastmuster (engl. scan pattern)

Mögliche Scanmuster sind:

- Reihen
- Spalten
- Zeilen, dann Spalten
- Spalten, dann Zeilen

11.10.4.11 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.

11.10.4.12 Tiefe (engl. depth) (Einheiten)

Die Oberseite des Materials liegt immer bei *Z=0*. Der tiefste Schnitt in das Material ist bei *Z=-depth*.

11.10.4.13 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*.

11.10.4.14 Werkzeug-Durchmesser

Der Durchmesser des schneidenden Teils des Werkzeugs.

11.10.4.15 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.

11.10.4.16 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”

11.10.4.17 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.

11.10.4.18 Kontaktwinkel

When *Lace bounding* is not *None*, slopes greater than *Contact angle* are considered to be *strong* slopes, and slopes less than that angle are considered to be weak slopes.

11.10.4.19 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.

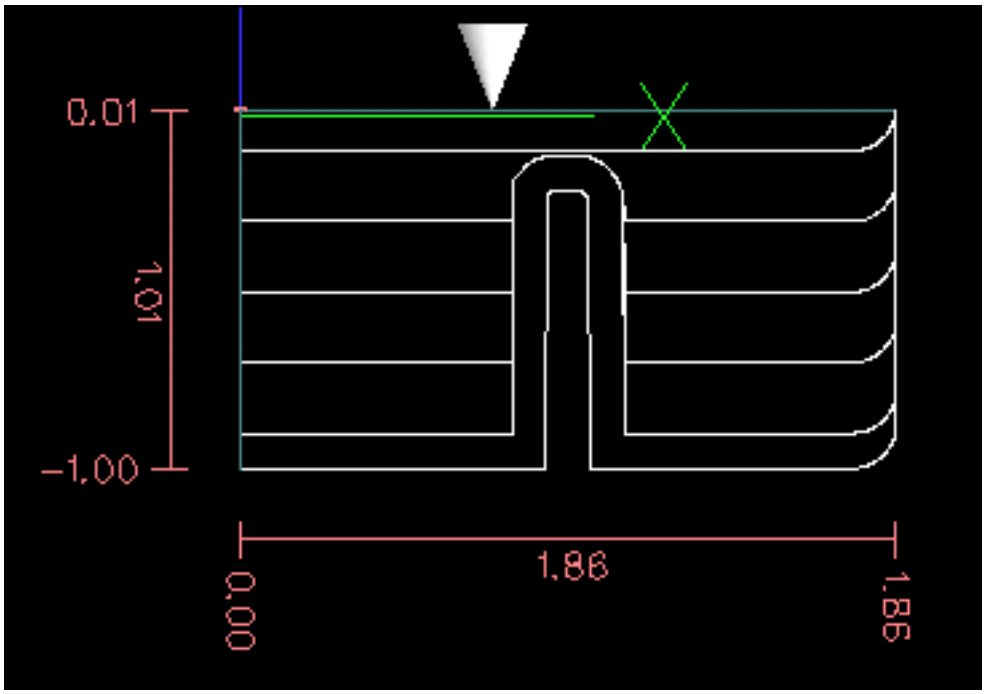


Abbildung 11.17: Schruppen und Schlichten

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

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

Kapitel 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.

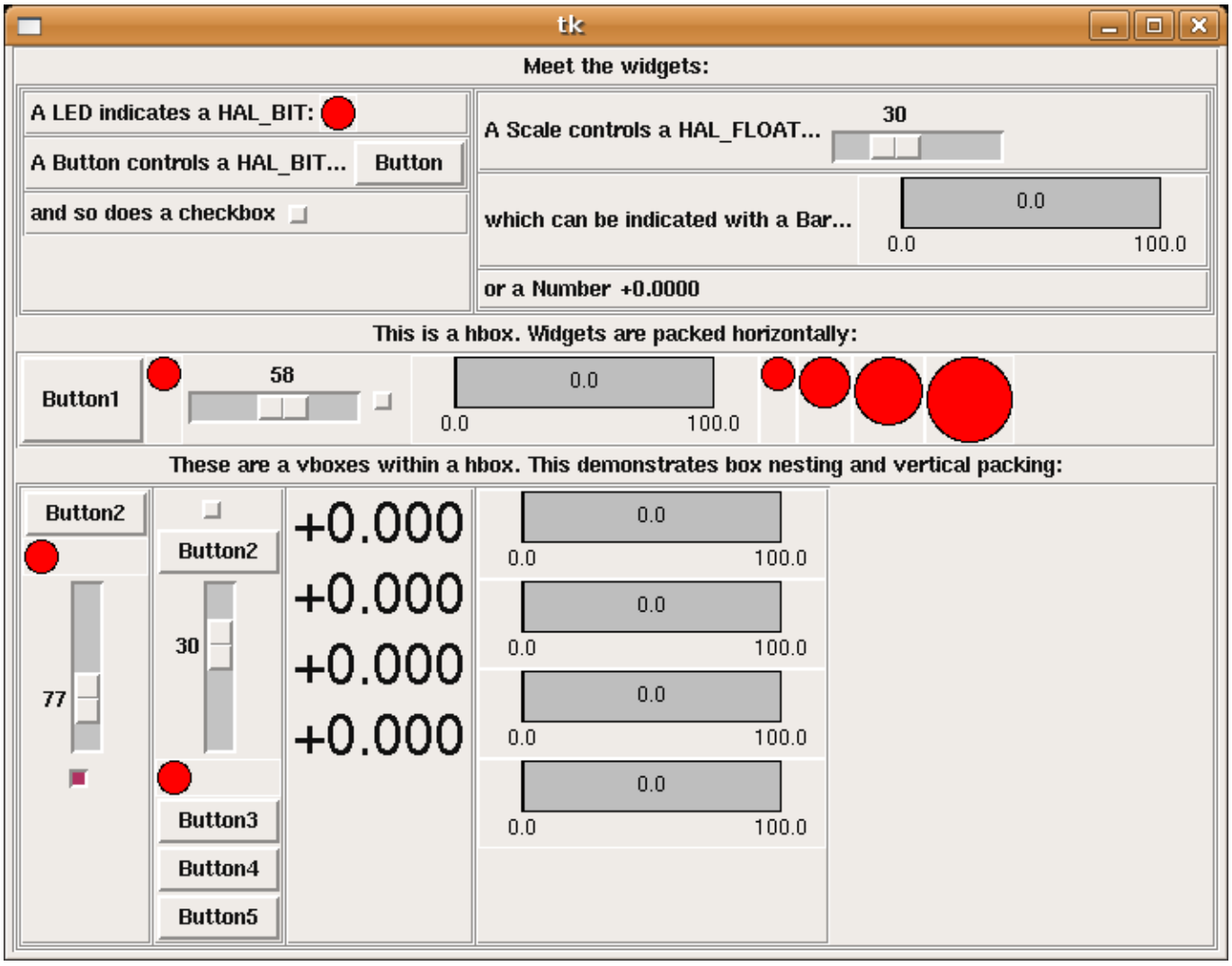


Abbildung 12.1: PyVCP Widgets Showcase

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>
```

Abbildung 12.2: 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 erstellt das Panel für Sie, das zwei Widgets enthält, ein Label mit dem Text *This is a LED indicator* (Dies ist eine LED-Anzeige) und eine LED, die den Zustand eines HAL-BIT-Signals anzeigt. Es wird auch eine HAL-Komponente mit dem Namen *mypanel* erstellt (alle Widgets in diesem Panel sind mit Pins verbunden, die mit *mypanel* beginnen). Da innerhalb des <led>-Tags kein <halpin>-Tag vorhanden war, benennt PyVCP den HAL-Pin für das LED-Widget automatisch 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 AXIS

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.

In addition to or instead of displaying a PyVCP panel as described above, it is possible to display one or more PyVCP panels as embedded tabs in the AXIS GUI. This is achieved by the following in the `[DISPLAY]` section of the `.ini` file:

```
EMBED_TAB_NAME    = Spindle
EMBED_TAB_COMMAND = pyvcp spindle.xml
```

The text label of the AXIS tab will display `Spindle`.

12.1.4.1 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.

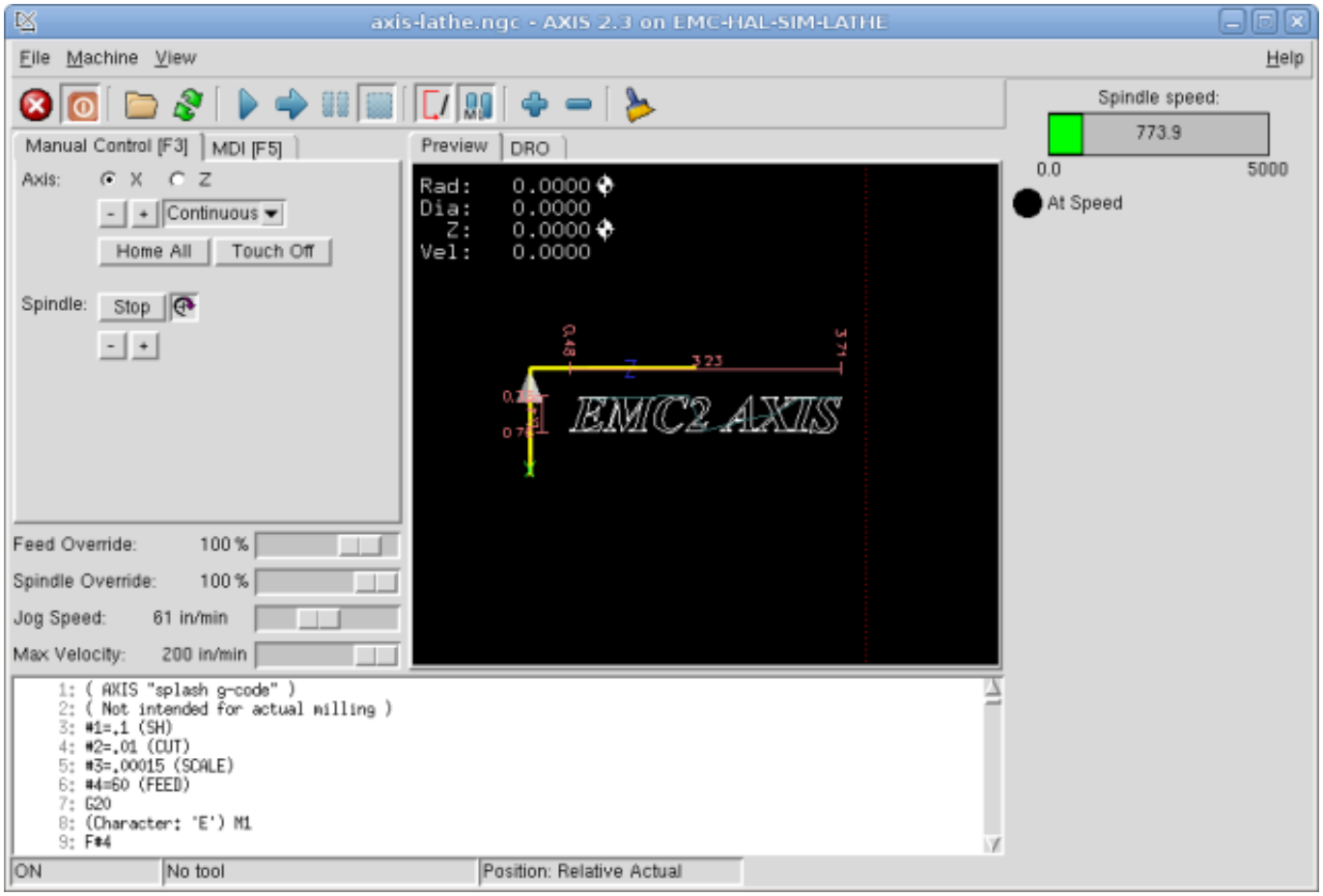
Damit unser Widget tatsächlich die Spindel-Geschwindigkeit anzeigt, muss es mit dem entsprechenden HAL-Signal verbunden werden. Eine HAL-Datei, die ausgeführt wird, sobald AXIS und PyVCP gestartet sind, kann im Abschnitt [HAL] der INI-Datei angegeben werden:

```
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-Signale gibt es in zwei Varianten: Bits und Zahlen. Bits sind Aus/Ein-Signale. Zahlen können *float*, *s32* oder *u32* sein. Für weitere Informationen über HAL-Datentypen siehe den Abschnitt HAL Data. Das PyVCP-Widget kann entweder den Wert des Signals mit einem Indikator-Widget anzeigen oder den Signalwert mit einem Kontroll-Widget verändern. Es gibt also vier Klassen von PyVCP-Widgets, die Sie mit einem HAL-Signal verbinden können. Eine fünfte Klasse von Hilfs-Widgets ermöglicht es Ihnen, Ihr Panel zu organisieren und zu beschriften.

- Widgets zur Anzeige von ”Bit”-Signalen: `led`, `rectled`.
- Widgets zur Steuerung von *Bit*-Signalen: `button`, `checkbutton`, `radiobutton`.
- Widgets zur Anzeige von numerischen Signalen: *number*, *s32*, *u32*, *bar*, *meter*.
- Widgets zur Steuerung von numerischen Signalen: `spinbox`, `scale`, `jogwheel`.
- Hilfs-Widgets: `hbox`, `vbox`, `table`, `label`, `labelframe`.

12.1.6.1 Syntax

Jedes Widget wird kurz beschrieben, gefolgt von dem verwendeten Markup und einem Screenshot. Alle Tags innerhalb des Haupt-Widget-Tags sind optional.

12.1.6.2 Allgemeine Anmerkungen

Gegenwärtig werden sowohl eine tagbasierte 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 http://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.

12.1.6.3 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.
- *(”Helvetica”,20)* - 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* hinzufügen.

```
<label>
    <text>”Dies ist ein Label:”</text>
    <font>(”Helvetica”,20)</font>
</label>
```

Der obige Code ergab dieses Beispiel:

Abbildung 12.3: Beispiel für ein einfaches Etikett

12.1.6.4 Multi_Label

Eine Erweiterung der Textbeschriftung.

Wählbare Textbeschriftung, kann bis zu 6 Beschriftungslegenden anzeigen, wenn der zugehörige Bit-Pin aktiviert ist.

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*.

12.1.6.5 LEDs

Eine LED wird verwendet, um den Status eines *bit* halpin anzuzeigen. Die LED-Farbe ist on_color, wenn der Halpin true ist, und off_color, wenn nicht.

- *<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. Standard 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:

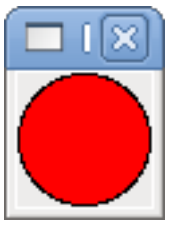


Abbildung 12.4: Beispiel für eine runde LED

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.

Abbildung 12.5: Beispiel für eine einfache Rechteck-LED

12.1.6.6 Buttons

Eine Taste wird verwendet, um einen BIT-Pin zu steuern. Der Pin wird auf True gesetzt, wenn die Taste gedrückt und gehalten wird, und wird auf False gesetzt, wenn die Taste losgelassen wird. Schaltflächen können die folgenden optionalen Optionen verwenden.

- *<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 Eine Texttaste steuert einen ”bit”-HAL-Pin. Der HAL-Pin liefert falsch, bis der Button gedrückt wird, dann ist er wahr. Der Button ist eine Momenttaste.

Die Text Button hat einen optionalen Deaktivierungsstift, der beim Hinzufügen von <disable_pin>True</disa 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:

Abbildung 12.6: Beispiel für einfache Buttons

Checkbutton Ein Checkbutton steuert ein bit HAL-Pin. Der HAL-Pin wird auf True gesetzt, wenn der Button angekreuzt ist, und auf False, wenn der Button nicht angekreuzt ist. Der Checkbutton ist ein Toggle-Button. Die Checkbuttons können anfänglich auf TRUE oder FALSE gesetzt werden. Ein Pin namens changepin wird ebenfalls automatisch erstellt, der den Checkbutton über HAL umschalten kann, wenn der verknüpfte Wert geändert wird, um die Anzeige aus der Ferne zu aktualisieren.

Abbildung 12.7: Nicht-markierter (engl. unchecked) button

Abbildung 12.8: 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:

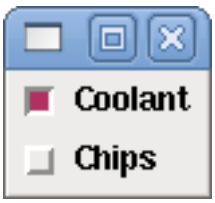


Abbildung 12.9: Einfaches Checkbutton-Beispiel

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 Halbpins 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:

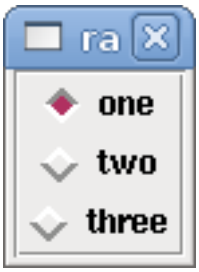


Abbildung 12.10: Einfaches Radiobutton-Beispiel

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” der ausgewählte Wert.
Verwenden Sie dieses Tag ”<orient>HORIZONTAL</orient>”, um die Anzeige horizontal zu gestalten.

12.1.6.7 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 Das Zahlen-Widget zeigt den Wert eines Gleitkomma-Signals an.

```
<number>
    <halpin>”my-number”</halpin>
    <font>(”Helvetica”,24)</font>
    <format>”+4.4f”</format>
</number>
```

Der obige Code ergab dieses Beispiel:

Abbildung 12.11: 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.

s32-Nummer Das s32-Zahlen-Widget zeigt den Wert einer s32-Zahl an. Die Syntax ist die gleiche wie die von *number*, mit Ausnahme des Namens, der <s32> lautet. Stellen Sie sicher, dass die Breite breit genug ist, um die größte Zahl, die Sie voraussichtlich verwenden werden, abzudecken.

```
<s32>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"6d"</format>
    <width>6</width>
</s32>
```

Der obige Code ergab dieses Beispiel:

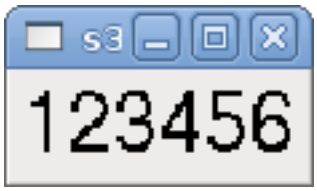


Abbildung 12.12: Einfaches s32 Zahlenbeispiel

u32 Zahl Das Widget u32 number zeigt den Wert einer u32-Zahl an. Die Syntax ist die gleiche wie die von *number*, mit Ausnahme des Namens, der <u32> lautet.

Bar (engl. für Balken) Ein Balken-Widget zeigt den Wert eines FLOAT-Signals sowohl grafisch mit einer Balkenanzeige als auch numerisch an. Die Farbe des Balkens kann als eine Farbe für den gesamten Bereich festgelegt werden (Standard mit fillcolor) oder so eingestellt werden, dass sich die Farbe in Abhängigkeit vom Wert des Halbsignals ändert (Bereich1, Bereich2, Bereich3 müssen alle festgelegt werden; wenn Sie nur zwei Bereiche wünschen, setzen Sie zwei davon auf dieselbe Farbe).

- `<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), sets the first range and color.
- <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:

Abbildung 12.13: Einfaches Bar-Beispiel

Meter Das Messgerät zeigt den Wert eines FLOAT-Signals mit einer herkömmlichen Messuhr an.

```
<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:

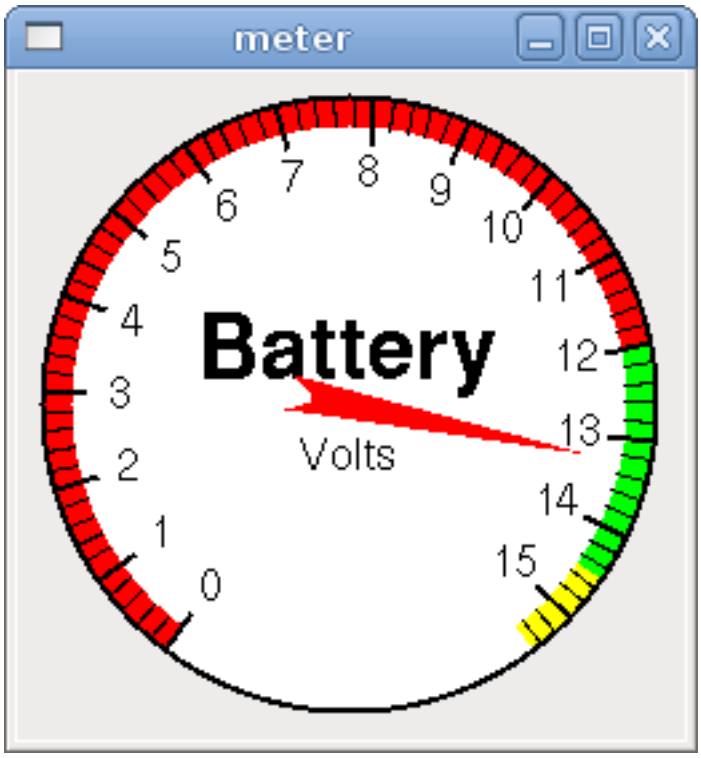


Abbildung 12.14: Einfaches Beispiel für ein Messgerät

12.1.6.8 Numerische Eingaben

Spinbox (Einstellrad) Eine Spinbox steuert einen FLOAT-Stift. Sie erhöhen oder verringern den Wert des Pins, indem Sie entweder auf die Pfeile drücken oder auf die Spinbox zeigen und das Mausrad drehen. Wenn das Feld param_pin auf TRUE(1) gesetzt ist, wird ein Pin erstellt, der verwendet werden kann, um die Spinbox auf einen Anfangswert zu setzen und ihren Wert ohne HID-Eingabe aus der Ferne zu ändern.

```
<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:

Abbildung 12.15: Einfaches Spinbox-Beispiel

Skala Eine Skala steuert einen Float oder einen s32-Pin. Sie erhöhen oder verringern den Wert des Pin, indem Sie entweder den Schieberegler ziehen oder auf die Skala zeigen und das Mausrad drehen. Dem *halpin* werden sowohl *-f* als auch *-i* hinzugefügt, um die float- und s32-Pins zu bilden. Width ist die Breite des Schiebereglers in vertikaler und die Höhe des Schiebereglers in horizontaler Ausrichtung. Wenn das Feld param_pin auf TRUE(1) gesetzt wird, so wird ein Pin erstellt, der dazu verwendet werden kann, die Spinbox auf einen Anfangswert zu setzen und ihren Wert ohne HID-Eingabe ferngesteuert zu ändern.

```
<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:

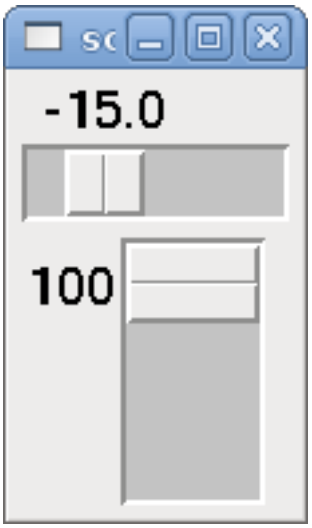


Abbildung 12.16: Beispiel für eine einfache Skalierung

Anmerkung
Beachten Sie, dass standardmäßig "min" angezeigt wird, auch wenn es größer als "max" ist, es sei denn, "min" ist negativ.

Wählscheibe (engl. dial) Das Dial gibt einen HAL-Float aus und reagiert sowohl auf Mausrad als auch auf Ziehen. Doppelklicken Sie mit der linken Maustaste, um die Auflösung zu erhöhen, und

doppelklicken Sie mit der rechten Maustaste, um die Auflösung um eine Ziffer zu reduzieren. Die Ausgabe wird durch die Min- und Max-Werte begrenzt. Das <cpr> ist, wie viele Teilstriche sich auf der Außenseite des Rings befinden (Vorsicht vor hohen Zahlen). Wenn das Feld param_pin auf TRUE(1) gesetzt ist, wird ein Pin erstellt, mit dem die Spinbox auf einen Anfangswert gesetzt und ihr Wert ohne HID-Eingabe aus der Ferne geändert werden kann.

```
<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:

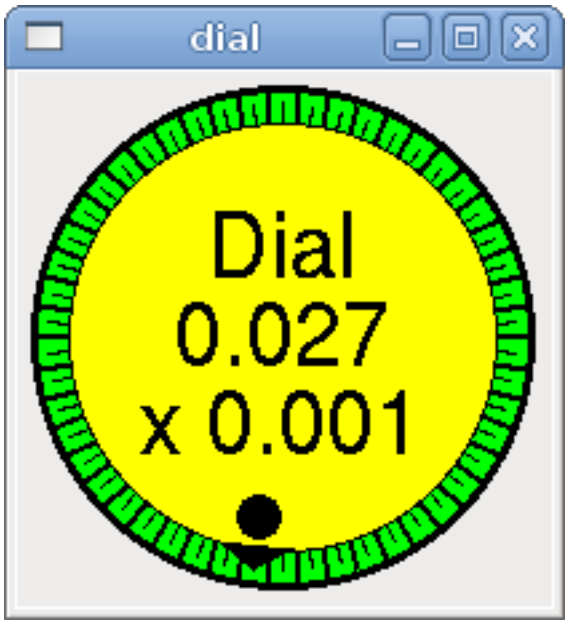


Abbildung 12.17: Einfaches Wählbeispiel

Jogwheel Jogwheel ahmt ein echtes Jogwheel nach, indem es einen FLOAT-Pin ausgibt, der auf- oder abwärts zählt, wenn das Rad gedreht wird, entweder durch Ziehen in einer kreisförmigen Bewegung oder durch Drehen des Mausrads.

Optionale Tags: * *<text>”Mein Text”</text>* zeigt Text an * *<bgcolor>”grey”</bgcolor> <fillcolor>”green”</fillcolor>* Hintergrund- und aktive Farben * *<scale_pin>1</scale_pin>* erzeugt Skalentext und einen FLOAT.scale-Pin zur Anzeige der Jog-Skala * *<clear_pin>1</clear_pin>* erstellt DRO und einen BIT.reset-Pin, um DRO zurückzusetzen. Benötigt scale_pin für skalierte DRO. shift+click setzt DRO ebenfalls zurück

```
<jogwheel>
    <halpin>”my-wheel”</halpin>
    <cpr>45</cpr>
```

```
    <size>250</size>
</jogwheel>
```

Der obige Code ergab dieses Beispiel:

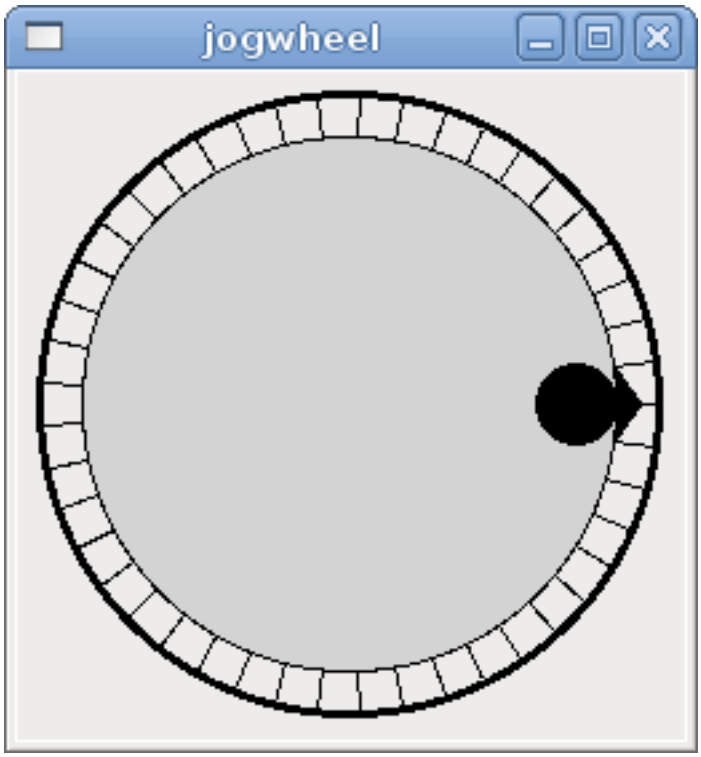


Abbildung 12.18: Einfaches Jogwheel-Beispiel

12.1.6.9 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 Das *image_bit* schaltet zwischen zwei Bildern um, indem es den halpin auf true oder false setzt.

```
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_bit 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.

Abbildung 12.19: Selectimage False Beispiel

Abbildung 12.20: Selectimage True Beispiel

Image u32 Das *image_u32* ist dasselbe wie *image_bit*, außer dass Sie im Wesentlichen eine unbegrenzte Anzahl von Bildern haben und das Bild *auswählen*, indem Sie den Halpin auf einen ganzzahligen Wert mit 0 für das erste Bild in der Bilderliste und 1 für das zweite setzen Bild usw.

```
<image name='stb' file='stb.gif'/>
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_u32 halpin='selectimage' images='stb fwd rev'/>
</vbox>
```

Der obige Code erzeugt das folgende Beispiel, indem das Bild stb.gif hinzugefügt wird.

Abbildung 12.21: Einfaches image_u32 Beispiel mit halpin=0

Abbildung 12.22: Einfache image_u32 Beispiel withhalpin=1

Abbildung 12.23: Einfaches Beispiel image_u32 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.

12.1.6.10 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:

Abbildung 12.24: Container Borders Showcase

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 ja oder nein ist. Die Voreinstellung ist ja.

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:

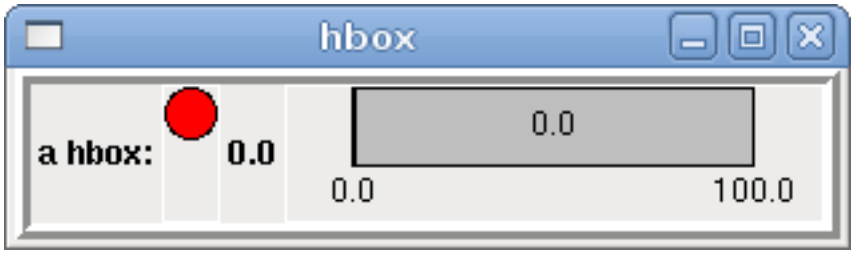


Abbildung 12.25: Einfaches hbox-Beispiel

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:

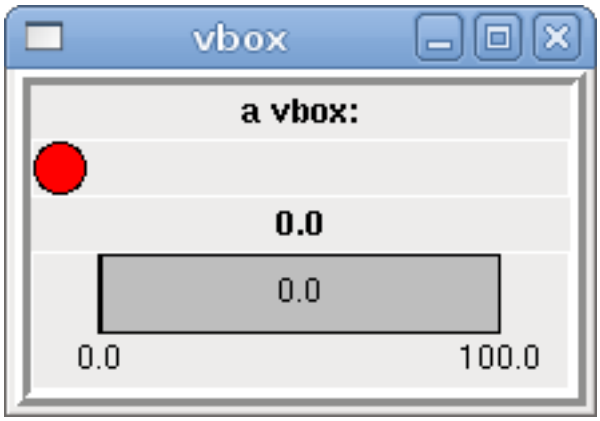


Abbildung 12.26: Einfaches vbox-Beispiel

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 Hbox.

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:

Abbildung 12.27: 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)"/>
    <u32 halpin="test"/>
</table>
```

Der obige Code ergab dieses Beispiel:

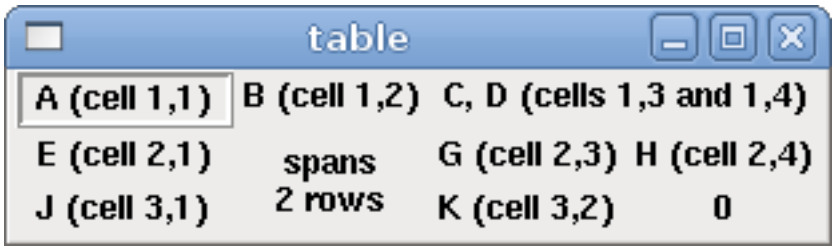
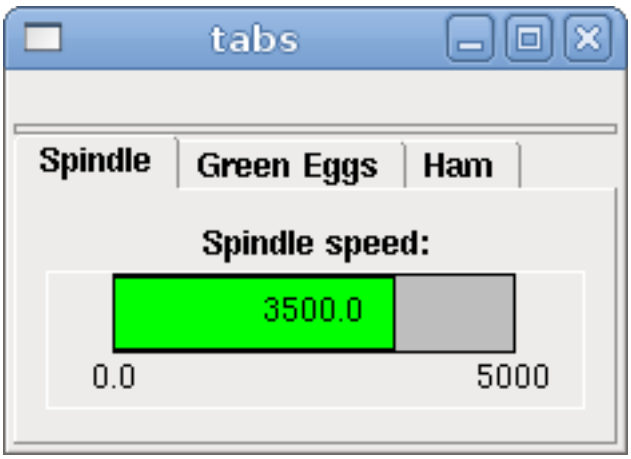


Abbildung 12.28: Beispiel für eine Tabelle

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.



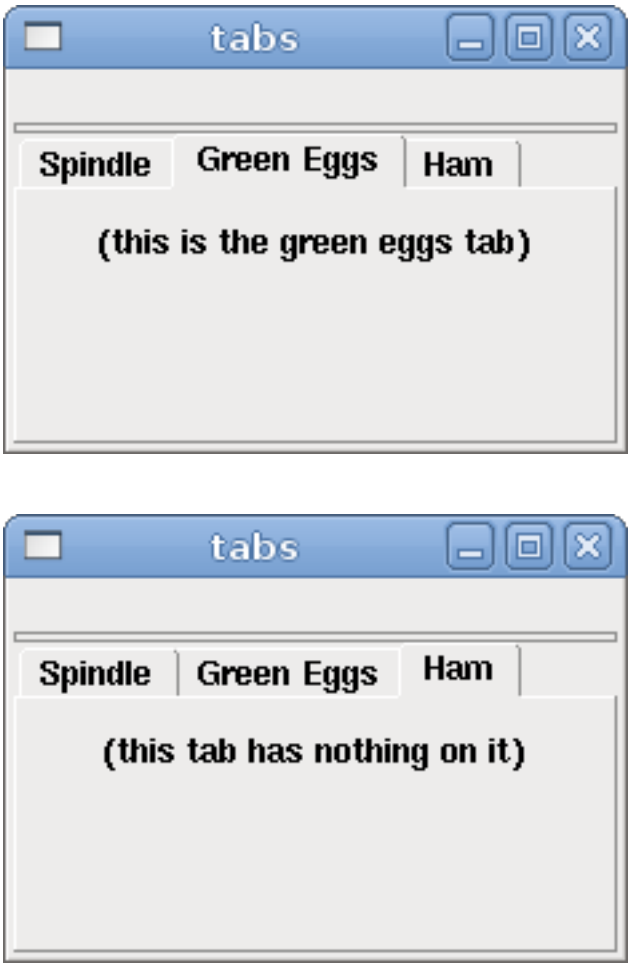


Abbildung 12.29: Einfache Tabs-Beispiel

12.2 PyVCP-Beispiele

12.2.1 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*.

12.2.2 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.

12.2.3 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.

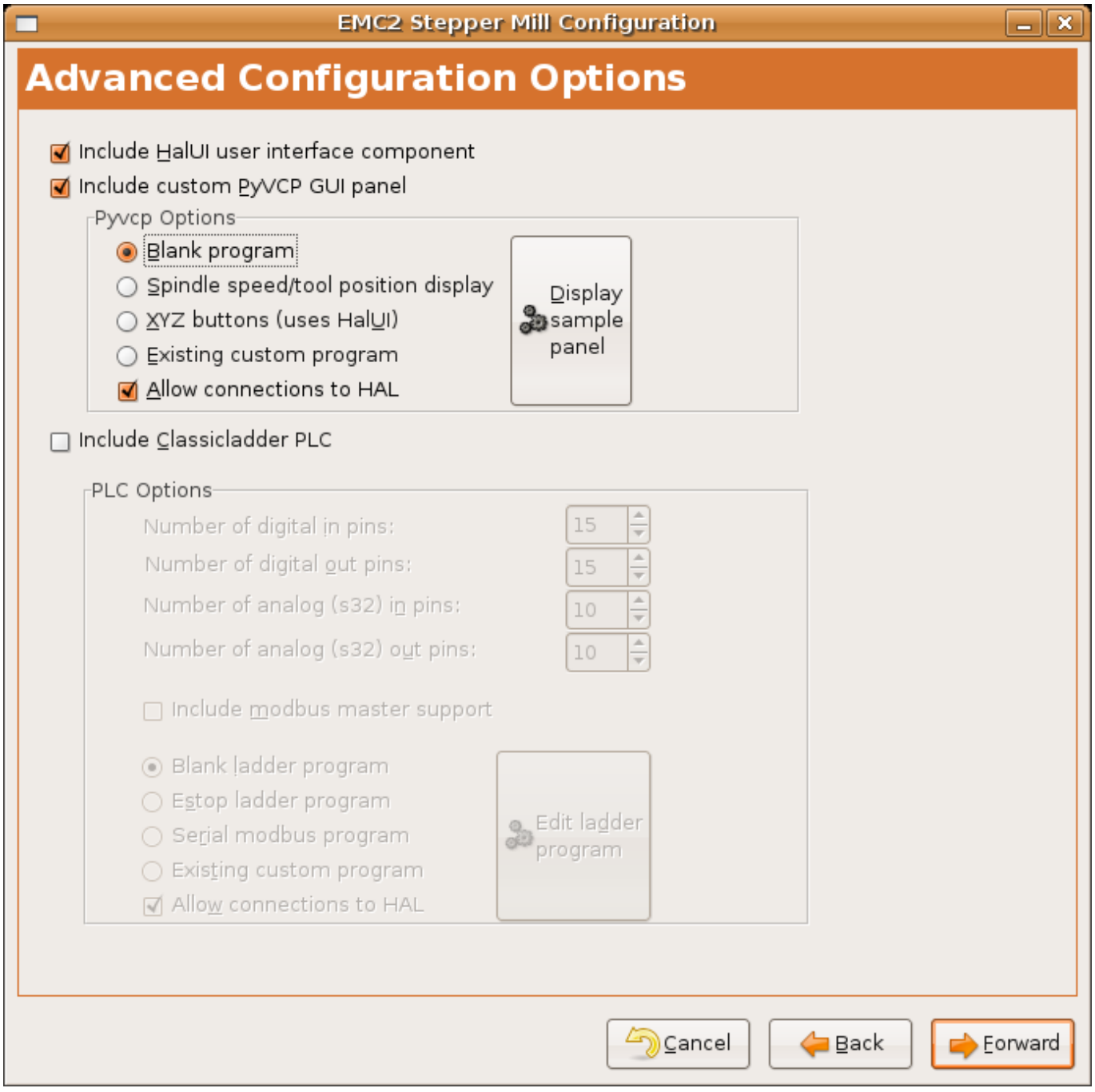


Abbildung 12.30: XYZ-Assistent Konfiguration

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.

12.2.3.1 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.

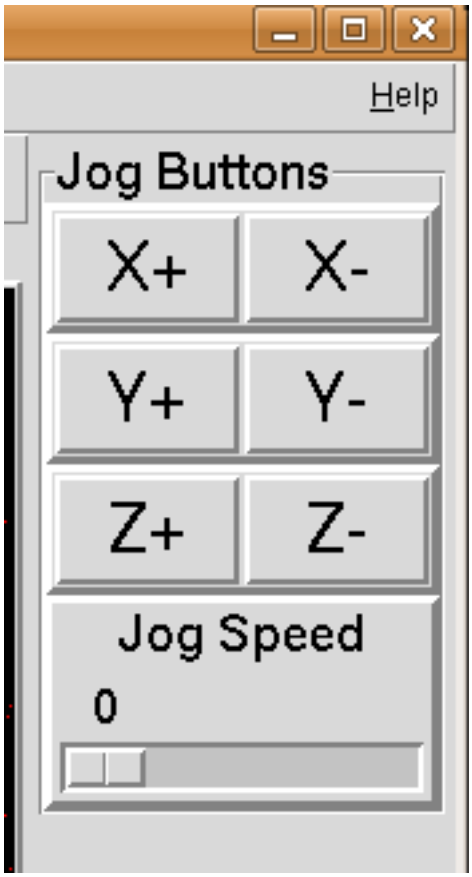
Schauen Sie in den Abschnitt PyVCP Widgets Referenz des Handbuchs für detailliertere Informationen über jedes Widget documentation des widgets.

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.



12.2.3.2 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.

12.2.4 Port-Tester

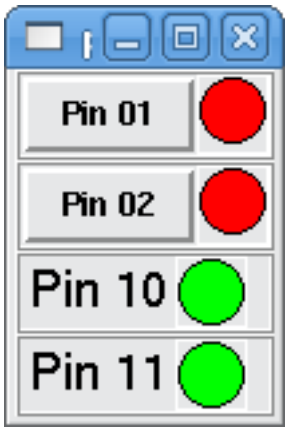
Dieses Beispiel zeigt Ihnen, wie Sie mit PyVCP und HAL einen einfachen Parallelport-Tester erstellen können.

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.



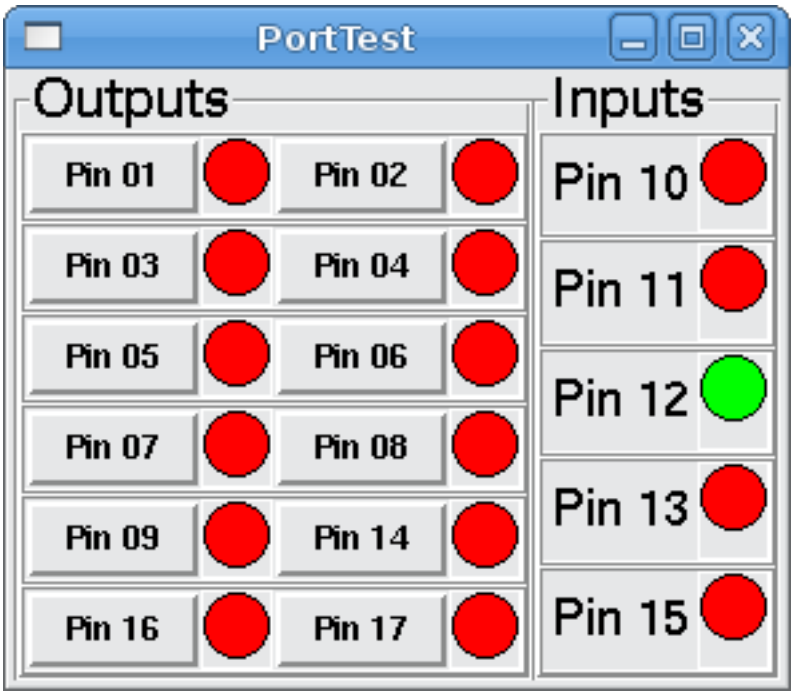
Um die HAL-Befehle auszuführen, die wir benötigen, um alles zum Laufen zu bringen, fügen wir Folgendes in unsere Datei ptest.hal ein.

```
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 bit   IN  FALSE  parport.0.pin-01-out <== pin01
    2 bit   IN  FALSE  parport.0.pin-02-out <== pin02
    2 bit   IN  FALSE  parport.0.pin-03-out
    2 bit   IN  FALSE  parport.0.pin-04-out
    2 bit   IN  FALSE  parport.0.pin-05-out
    2 bit   IN  FALSE  parport.0.pin-06-out
    2 bit   IN  FALSE  parport.0.pin-07-out
    2 bit   IN  FALSE  parport.0.pin-08-out
    2 bit   IN  FALSE  parport.0.pin-09-out
    2 bit   OUT TRUE   parport.0.pin-10-in ==> pin10
    2 bit   OUT FALSE  parport.0.pin-10-in-not
    2 bit   OUT TRUE   parport.0.pin-11-in ==> pin11
    2 bit   OUT FALSE  parport.0.pin-11-in-not
    2 bit   OUT TRUE   parport.0.pin-12-in
    2 bit   OUT FALSE  parport.0.pin-12-in-not
    2 bit   OUT TRUE   parport.0.pin-13-in
    2 bit   OUT FALSE  parport.0.pin-13-in-not
    2 bit   IN  FALSE  parport.0.pin-14-out
    2 bit   OUT TRUE   parport.0.pin-15-in
    2 bit   OUT FALSE  parport.0.pin-15-in-not
    2 bit   IN  FALSE  parport.0.pin-16-out
    2 bit   IN  FALSE  parport.0.pin-17-out
    4 bit   OUT FALSE  ptest.btn01 ==> pin01
```

```
    4 bit   OUT FALSE  ptest.btn02 ==> pin02
    4 bit   IN  FALSE  ptest.led-01 <== pin01
    4 bit   IN  FALSE  ptest.led-02 <== pin02
    4 bit   IN  TRUE   ptest.led-10 <== pin10
    4 bit   IN  TRUE   ptest.led-11 <== pin11
```

Dies zeigt Ihnen, welche Pins IN und welche Pins OUT sind, sowie alle Verbindungen.

12.2.5 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.

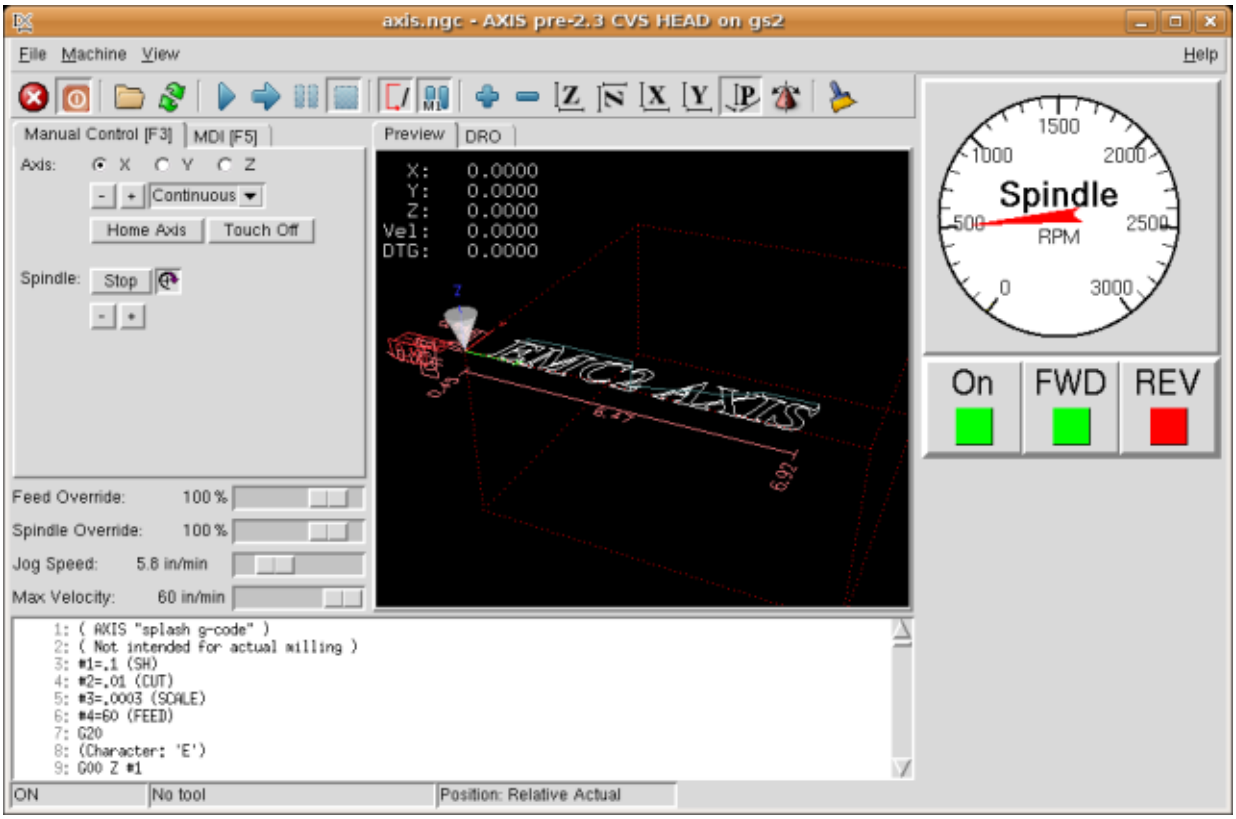
12.2.5.1 Das Panel

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.



12.2.5.2 Die Verbindungen

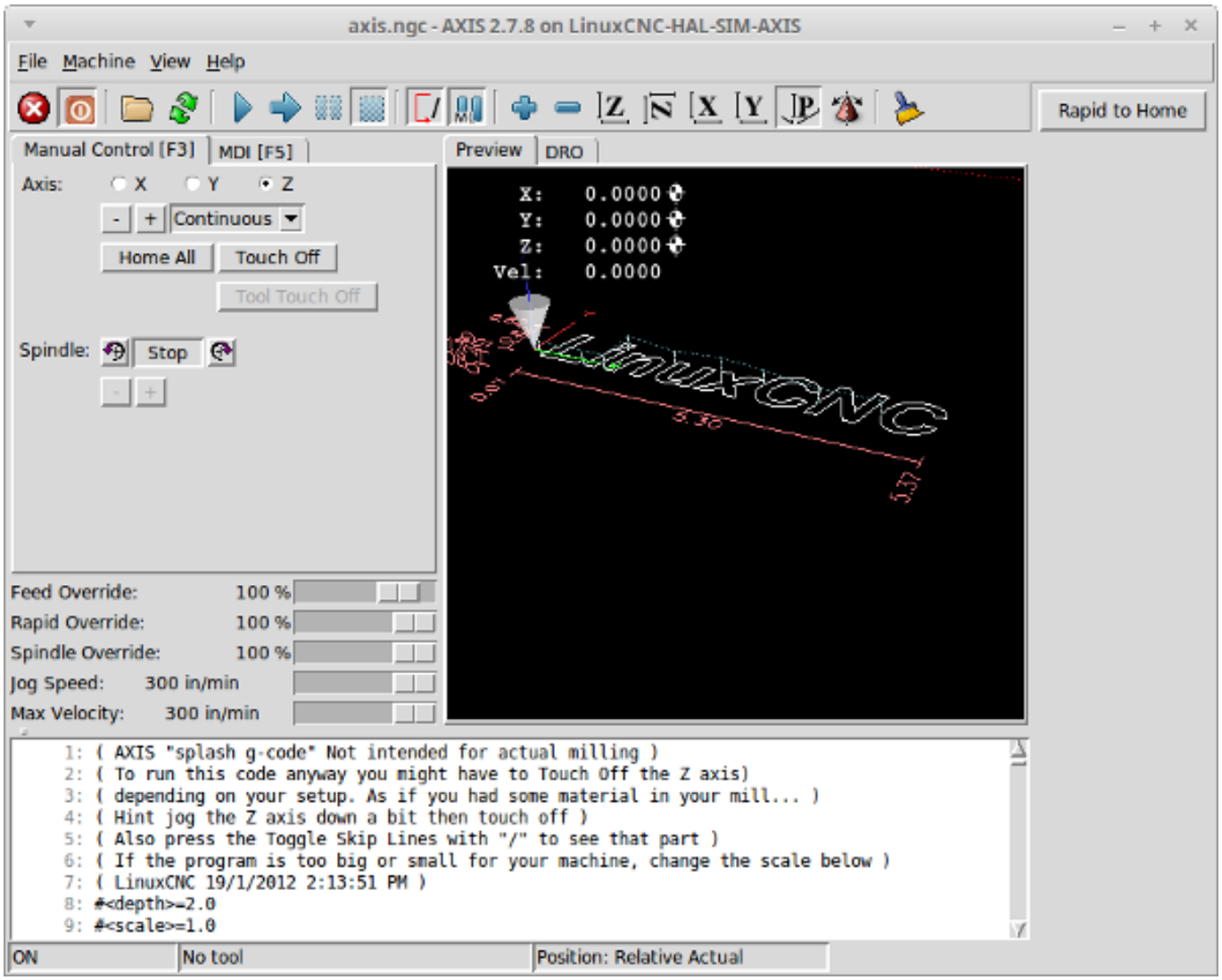
Damit das funktioniert, fügen wir den folgenden Code in die Datei custom_postgui.hal ein.

```
# 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
```

Einige der Zeilen bedürfen vielleicht einer Erläuterung. Die Zeile fwd led verwendet das in der Datei custom.hal erstellte Signal, während die Zeile rev led das Bit spindle-rev verwenden muss. Sie können das Bit spindle-fwd nicht zweimal verknüpfen, also verwenden Sie das Signal, mit dem es verknüpft wurde.

12.2.6 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
```

Anmerkung
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
```

Anmerkung
Informationen über den Befehl net

12.3 GladeVCP: Glade Virtuelles (engl. virtual) Control Panel

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 das umfangreiche GTK+-Widget-Set, 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.

12.3.1.1 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:

- Widget-Set: stützt sich auf das GTK+ Widget-Set.
- Erstellung von Benutzeroberflächen: verwendet den Glade WYSIWYG-Editor für Benutzeroberflächen.
- Jede HAL-Pin-Änderung kann für einen Callback und damit einen benutzerdefinierten Python-Ereignishand 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.

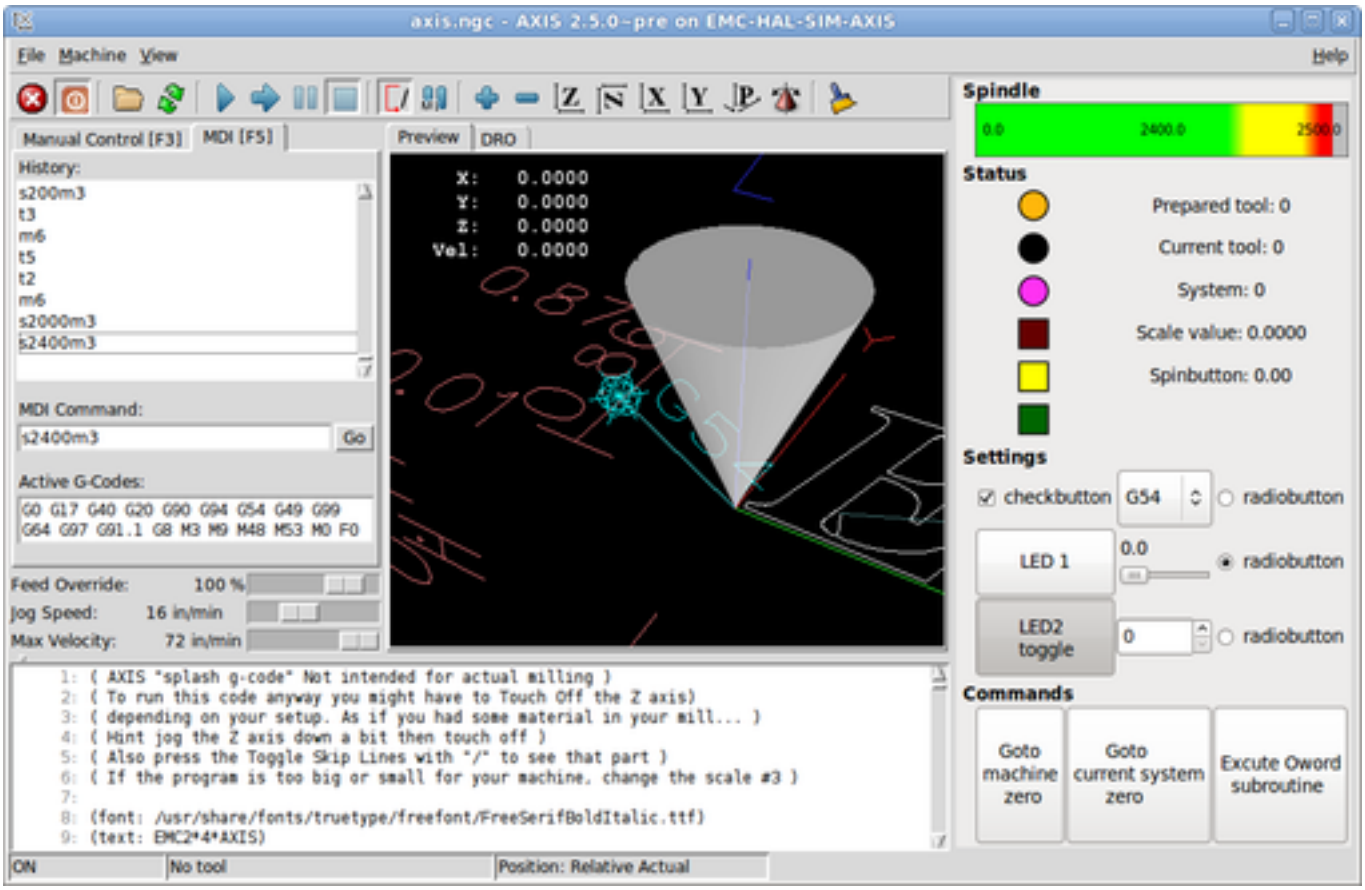
Installiertes LinuxCNC Wenn Sie eine installierte Version von LinuxCNC verwenden, sind die unten gezeigten Beispiele in der configuration picker in der *Sample Configurations* > *apps* > *GladeVCP* Zweig.

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.

Anmerkung
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.

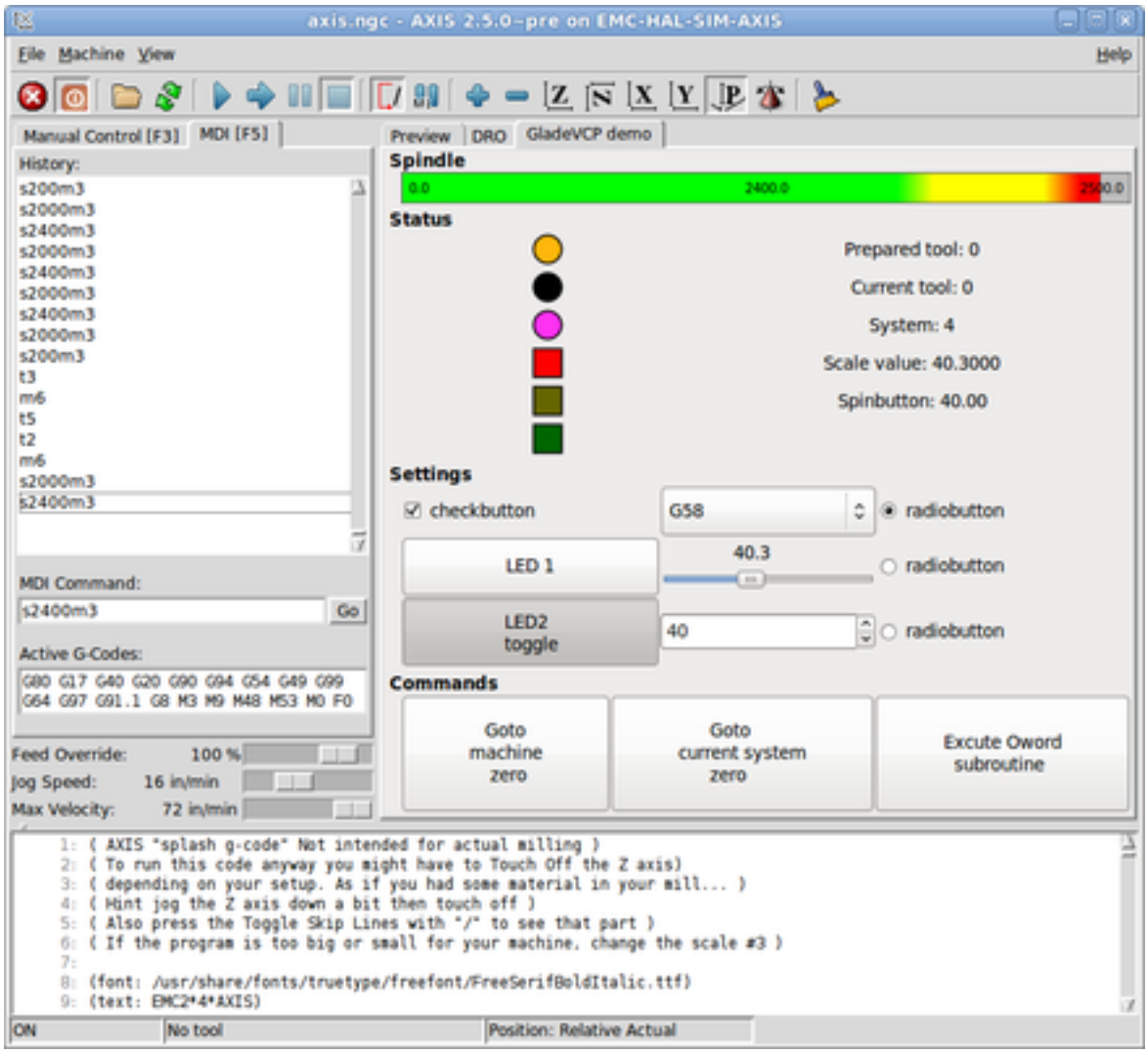
Führen Sie das in AXIS integrierte GladeVCP-Beispielpanel wie PyVCP wie folgt aus:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



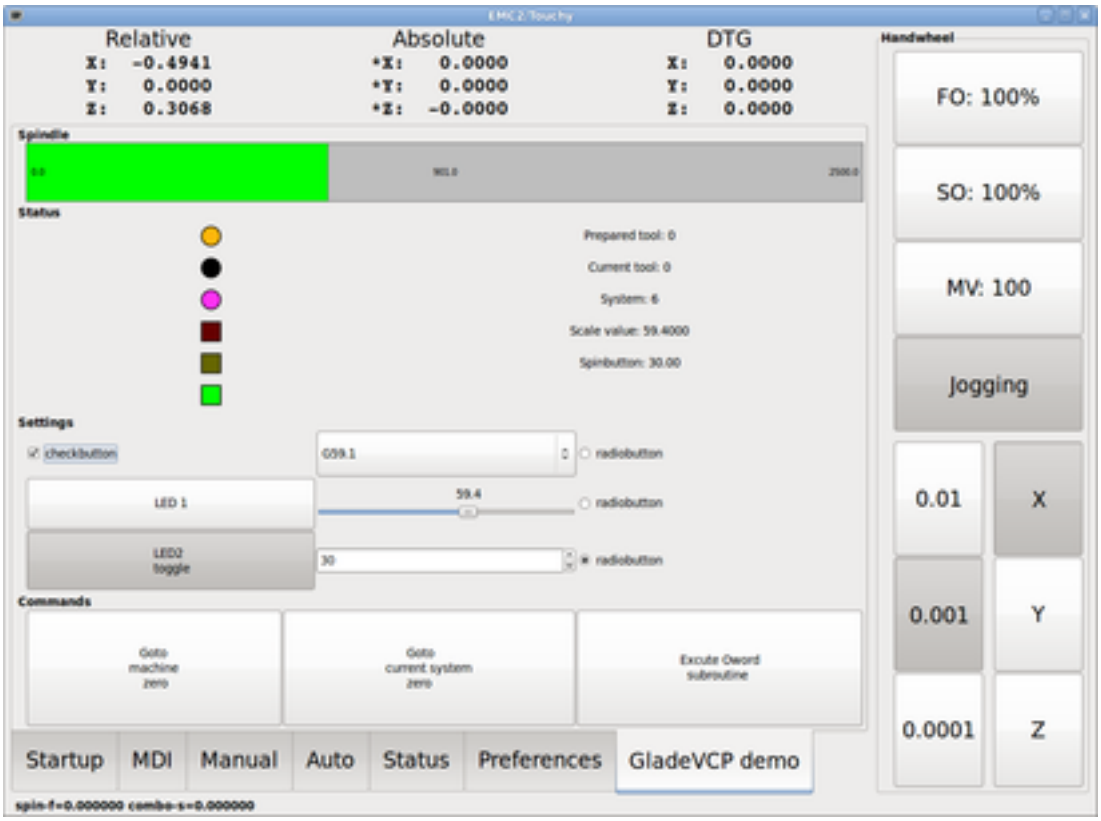
Führen Sie das gleiche Panel aus, aber als Registerkarte innerhalb von AXIS:

```
$ 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.

12.3.2.1 Erkunden des Beispielpanels

While running configs/sim/axis/gladevcp_panel.ini or configs/sim/axis/gladevcp_tab.ini, explore *Show HAL Configuration* - you will find the `gladevcp` HAL component and may observe their pin values while interacting with the widgets in the panel. The HAL setup can be found in *configs/axis/gladevcp/manual-example.hal*.

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.

12.3.2.2 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”.

12.3.2.3 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

12.3.3.1 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.

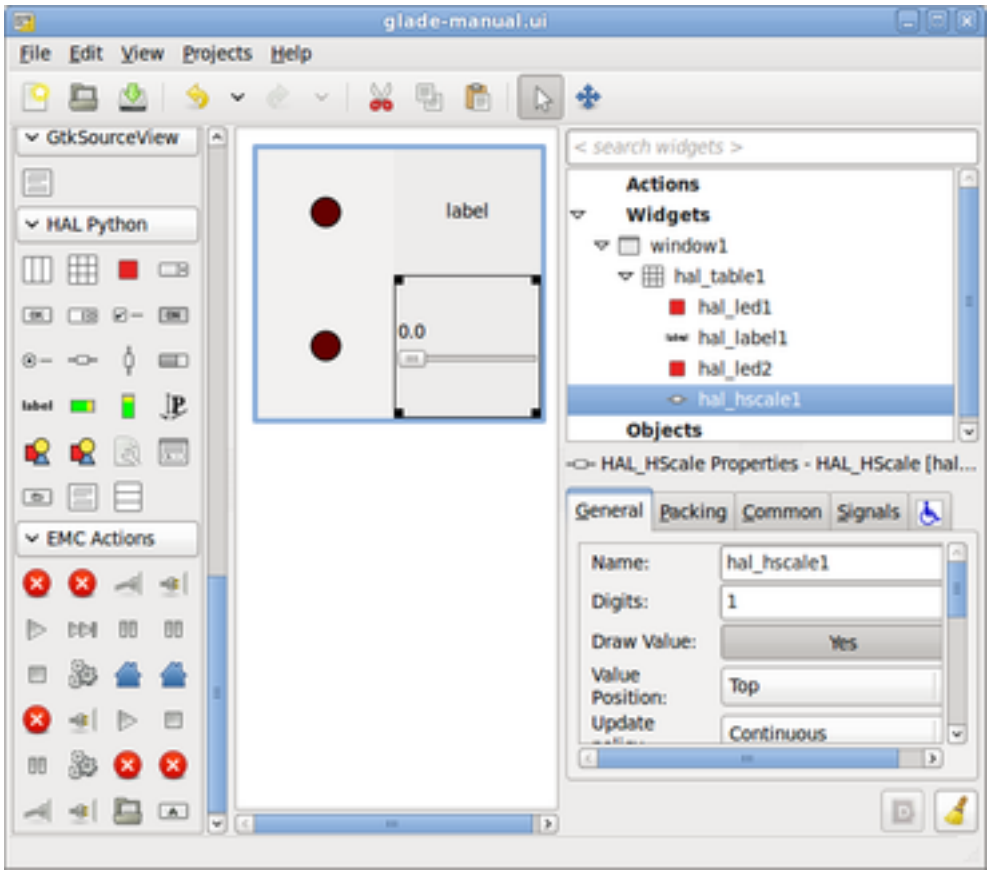
12.3.3.2 Glade ausführen, um eine neue Benutzeroberfläche zu erstellen

Dieser Abschnitt umreißt nur die ersten LinuxCNC-spezifischen Schritte. Weitere Informationen und eine Anleitung zu Glade finden Sie unter http://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 `<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 unter ”Toplevel” (linker Bereich) ”Window” (erstes Symbol) als Fenster der obersten Ebene, das standardmäßig ”window1” heißt. Ändern Sie diesen Namen nicht - GladeVCP verlässt sich auf ihn.
- Blättern Sie auf der linken Registerkarte nach unten und erweitern Sie *HAL Python* und *VCP-Aktionen*.
- einen Container wie eine HAL_Box oder eine HAL_Table aus *HAL Python* in den Rahmen einfügen
- einige Elemente wie LEDs, Schaltflächen usw. auswählen und in einem Container platzieren

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*.

12.3.3.3 Testen eines Panels

Sie sind jetzt bereit, es auszuprobieren (während LinuxCNC, z. B. AXIS läuft) mit:

```
gladevcp myui.ui
```

GladeVCP creates a HAL component named like the basename of the UI file - *myui* in this case - unless overridden by the `-c <component name>` option. If running AXIS, just try *Show HAL configuration* and inspect its pins.

You might wonder why widgets contained a *HAL_Hbox* or *HAL_Table* appear greyed out (inactive). HAL containers have an associated HAL pin which is off by default, which causes all contained widgets to render inactive. A common use case would be to associate these container HAL pins with `halui.machine.is-on` or one of the `halui.mode` signals, to assure some widgets appear active only in a certain state.

To just activate a container, execute the HAL command `setp gladevcp.<container-name> 1`.

12.3.3.4 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.

Achtung
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.

12.3.3.5 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
```

The default HAL component name of a GladeVCP application started with the GLADEVCP option is: `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
```

You may add arbitrary `gladevcp` options here, as long as they dont collide with the above command line options.

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
```

Anmerkung
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)

Anmerkung
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.

12.3.3.6 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
```

Note the `halcmd loadusr` way of starting the tab command - this assures that *POSTGUI_HALFILE* will only be run after the HAL component is ready. In rare cases you might run a command here which uses a tab but does not have an associated HAL component. Such a command can be started without

halcmd loadusr, and this signifies to AXIS that it does not have to wait for a HAL component since there is none.

When changing the component name in the above example, note that the names used in `-Wn <component>` and `-c <component>` must be identical.

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.

Anmerkung
Make sure the UI file is the last option passed to GladeVCP in both the `GLADEVCP=` and `EMBED_TAB_COMMAND=` statements.

12.3.3.7 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
```

Anmerkung
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.

12.3.4 GladeVCP-Befehlszeilenoptionen

See also `man gladevcp`. These are the GladeVCP command line options:

```
Usage: gladevcp [options] myfile.ui

Optionen:

-h, --help::
    Show this help message and exit.
```

```
-c NAME::
    Setzt den Komponentennamen auf NAME. Standard ist der Basisname der UI-Datei.

-d::
    Debug-Ausgabe einschalten

-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
    Position zu bestimmen.

-H DATEI::
    Führt HAL-Anweisungen aus DATEI mit halcmd aus, nachdem die
    Komponente eingerichtet und bereit ist

-m MAXIMUM::
    Erzwingt die Maximierung des Panel-Fensters. Zusammen mit der Option -g geometry
    kann man das Panel auf einen zweiten Monitor verschieben und es zwingen, den gesamten  ↩
        Bildschirm zu nutzen.

-t THEME::
    Setzt das gtk-Theme. Standard ist das Systemthema. Verschiedene Panels können  ↩
        unterschiedliche Themen haben.
    Ein Beispielthema finden Sie im http://wiki.linuxcnc.org/cgi-bin/wiki.pl?GTK_Themes[EMC ↩
         Wiki].

-x XID::
    GladeVCP in ein bestehendes Fenster XID einbinden, anstatt eines
    neuen Fensters der obersten Ebene

-u FILE::
    Dateien als zusätzliche benutzerdefinierte Module mit Handlern verwenden

-U USEROPT::
    übergibt USEROPTs an Python-Module
```

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 need to be embedded in a master GUI window setup.
- GladeVCP panels need to be embedded in a master GUI window setup, e.g., AXIS, or Touchy, Gscreen, or GMOCCAPY (embedded window or as an embedded tab).
- 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.
- Explicitly pass the HAL command file which refers to GladeVCP pins to GladeVCP with the `-H <halcmd file>` option (see previous section).

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-Signale gibt es in zwei Varianten: Bits und Zahlen. Bits sind Aus/Ein-Signale. Zahlen können ”float”, ”s32” oder ”u32” sein. Für weitere Informationen über HAL-Datentypen siehe HAL Handbuch. Die GladeVCP-Widgets können entweder den Wert des Signals mit einem Indikator-Widget anzeigen oder den Signalwert mit einem Kontroll-Widget verändern. Es gibt also vier Klassen von GladeVCP-Widgets, die Sie mit einem HAL-Signal verbinden können. Eine weitere Klasse von Hilfs-Widgets ermöglicht es Ihnen, Ihr Panel zu organisieren und zu beschriften.

- Widgets zur Anzeige von ”Bit”-Signalen: HAL_LED
- Widgets zur Steuerung von „Bit“-Signalen: 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 for display/edit of all axes offsets: OffsetPage
- Widgets for display/edit of all tool offsets: Tooloffset editor
- Widget for G-code display and edit: HAL_Sourceview
- Widget for MDI input and history display: MDI History

12.3.6.1 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' und *HAL_ComboBox*, die zwei Pins haben: einen `<widgetname>-f` (float) und einen `<widgetname>-s` (s32) Pin
- HAL_ProgressBar', die einen `<widgetname>-value`-Eingangspin und einen `<widgetname>-scale`-Eingangspin hat.

12.3.6.2 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 zum Beispiel herauszufinden, welche GtkWidget-bezogenen Methoden, Eigenschaften und Signale ein *HAL_Button* hat, schauen Sie in der Beschreibung von GtkButton im PyGtk Referenzhandbuch 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.

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
: die s32- und float-Pins der Widgets *HAL_Spinbutton* und *HAL_ComboBox* - beachten Sie, dass diese Widgets kein Attribut *hal_pin* haben!

hal_pin_scale
: der Float-Eingangs-Pin des Widgets "HAL_ProgressBar", der den maximalen absoluten Wert des Eingangs darstellt.

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.

12.3.6.3 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!

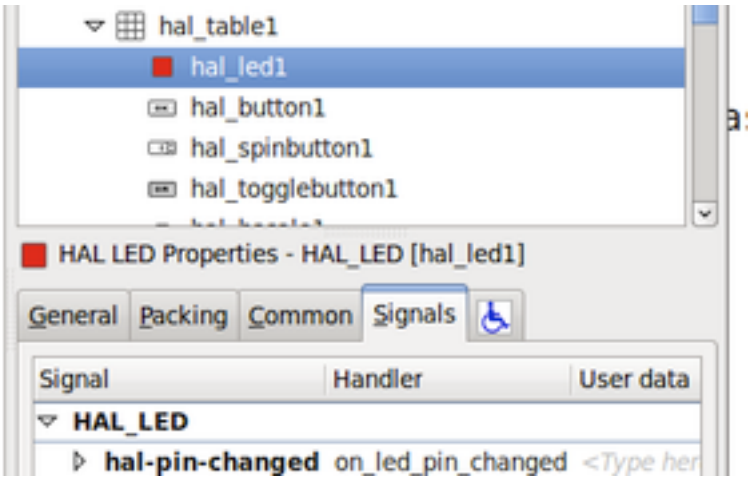
It might be tempting to *set HAL widget input pins* programmatically. Note this defeats the purpose of an input pin in the first place - it should be linked to, and react to signals generated by other HAL components. While there is currently no write protection on writing to input pins in HAL Python, this doesn't make sense. You might use `setp _pinname_ _value_` in the associated HAL file for testing though.

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).

12.3.6.4 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.

12.3.6.5 Buttons

Diese Gruppe von Widgets ist von verschiedenen Gtk-Buttons abgeleitet und besteht aus den Widgets HAL_Button, HAL_ToggleButton, HAL_RadioButton und CheckButton. Alle haben einen einzelnen BIT-Ausgangspin, der den gleichen Namen wie das Widget trägt. Buttons haben keine zusätzlichen Eigenschaften im Vergleich zu ihren Gtk-Basisklassen.

- HAL_Button: instantaneous action, does not retain state. Important signal: `pressed`

- HAL_ToggleButton, HAL_CheckButton: retains on/off state. Important signal: `toggled`
- HAL_RadioButton: a one-of-many group. Important signal: `toggled` (per button).
- Important common methods: `set_active()`, `get_active()`
- Important properties: `label`, `image`

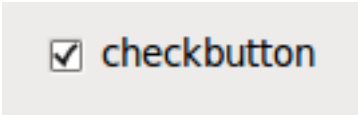


Abbildung 12.31: Checkbutton

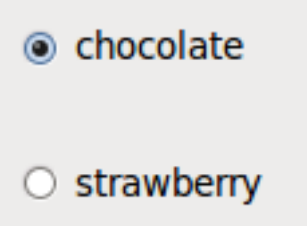


Abbildung 12.32: Radiobuttons

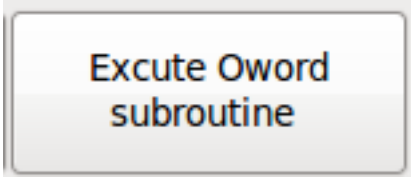


Abbildung 12.33: Toggle-Button

Tipp
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.

12.3.6.6 Skalen (engl. scales)

HAL_HScale und HAL_VScale sind von GtkHScale bzw. GtkVScale abgeleitet.

<widgetname>
out FLOAT pin

<widgetname>-s
out s32 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.

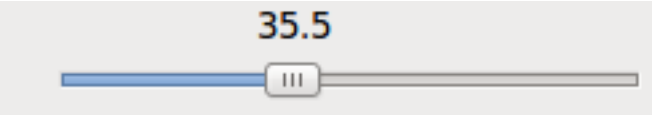


Abbildung 12.34: Beispiel HAL_HScale

12.3.6.7 SpinButton

HAL SpinButton ist von GtkSpinButton abgeleitet und besitzt zwei Pins:

<widgetname>-f
out FLOAT pin

<widgetname>-s
out s32 pin

Um nützlich zu sein, benötigen SpinButtons einen Einstellwert wie Skalen, siehe oben.

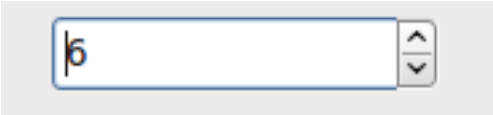


Abbildung 12.35: Beispiel SpinButton

12.3.6.8 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 = Zählungen reduzieren
- Im Uhrzeigersinn = Zählungen erhöhen
- Rad hoch = Zählungen erhöhen
- Rad nach unten = Zählungen reduzieren
- Links Doppelklick = x10 Skala
- Rechter Doppelklick = /10 Skala

hal_dial exportiert seinen Zählwert als HAL-Pins:

<widgetname>
out s32 pin

<widgetname>-scaled
out FLOAT pin

<widgetname>-delta-scaled
out FLOAT pin

`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

- Die Zählung basiert auf der gewählten CPR - sie zählt positiv und negativ. Er ist als s32-Pin verfügbar.
- Der skalierte Zähler ist die CPR-Zahl mal der Skala - er kann positiv und negativ sein.
  Wenn Sie die Skala ändern, spiegelt der Ausgang die Änderung sofort wider. Er ist als FLOAT-Pin verfügbar.
- Delta-scaled-count ist cpr count CHANGE, mal scale. +
  Wenn Sie die Skala ändern, werden nur die Zählerstände nach dieser Änderung skaliert und dann zum aktuellen Wert addiert. +
  Er ist als FLOAT-Pin verfügbar.

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

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:

- `[Widgetname].get_value()`
  Gibt den Zählwert als s32-Ganzzahl zurück
- [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.

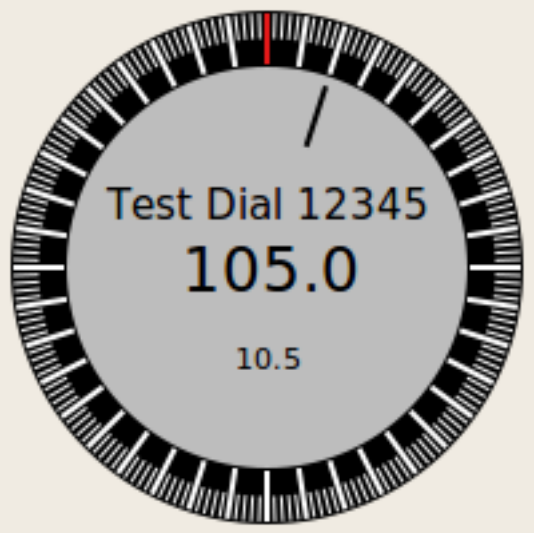


Abbildung 12.36: Beispiel Hal_Dial

12.3.6.9 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 = erhöhte Zählungen
- Rad hoch = Zählungen erhöhen
- Rad nach unten = Zählungen reduzieren

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.

`jogwheel` exportiert seinen Zählwert als HAL-Pin:

<widgetname>-s
: out s32 pin

`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
: Set the content of the label which may be shown over the counts value. The purpose is to give the user an idea about the usage of that jogwheel. If the label given is longer than 12 Characters, it will be cut to 12 Characters.

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)
```

There are two Python methods:

- `[widget name].get_value()`
  Gibt den Zählwert als Ganzzahl zurück
- `[Widgetname].set_label(”string”) +`
  Setzt den Inhalt des Labels mit ”string”

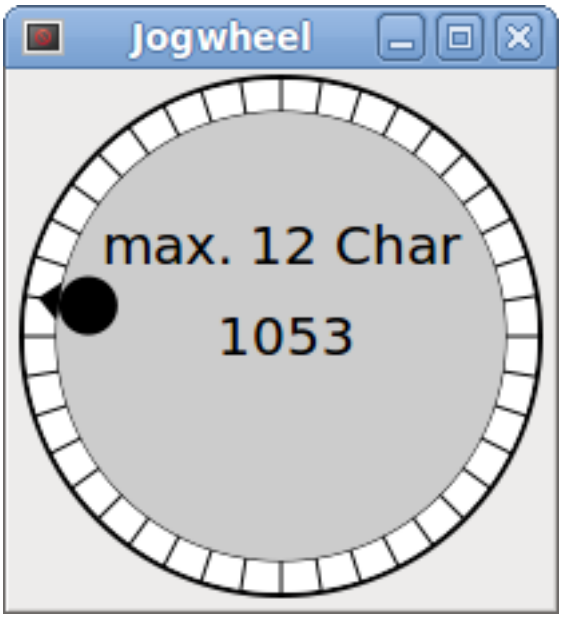


Abbildung 12.37: Beispiel JogWheel

12.3.6.10 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.

speedcontrol bietet einige HAL-Pins:

<widgetname>-value
out float pin +
Der angezeigte Wert des Widgets.

<widgetname>-scaled-value
out float pin +
Der angezeigte Wert geteilt durch den Skalenwert, ist dies sehr nützlich, wenn die Geschwindigkeit in Einheiten / min angezeigt wird, aber LinuxCNC erwartet, dass es in Einheiten / Sekunde sind.

<widgetname>-scale
in float pin +
Die anzuwendende Skalierung. +
Voreingestellt ist 60.

<widgetname>-increase
in Bit-Pin +
Solange der Pin wahr ist, erhöht sich der Wert. +
Sehr praktisch bei angeschlossenem Taster.

<widgetname>-decrease
in Bit-Pin
Solange der Pin wahr ist, wird der Wert verringert.
Sehr praktisch bei angeschlossenem Taster.

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.

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

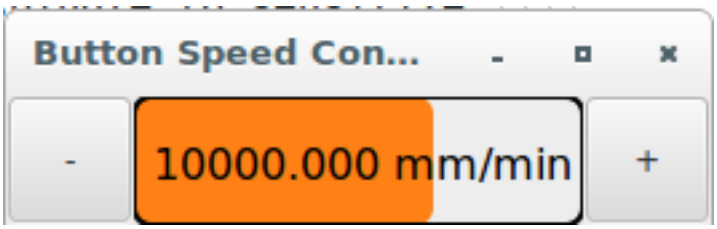


Abbildung 12.38: Beispiel Speedcontrol

12.3.6.11 Label

`hal_label` ist ein einfaches Widget, das auf GtkLabel basiert und einen HAL-Pin-Wert in einem benutzerdefinierten Format darstellt.

label_pin_type
Der HAL-Typ des Pins (0:s32, 1:float, 2:u32), siehe auch den Tooltip auf ’General→HAL pin type ’ (Hinweis: Dies unterscheidet sich von PyVCP, das drei Label-Widgets hat, eines für jeden Typ).

text_template
Bestimmt den angezeigten Text - eine Python-Formatzeichenfolge zur Umwandlung des Pin-Werts in Text. Der Standardwert ist `%s` (Werte werden mit der Funktion str() umgewandelt), kann aber als Argument für Pythons format()-Methode jede beliebige Zahl enthalten. +
Beispiel: `Distance: %.03f` zeigt den Text und den Pin-Wert mit 3 Nachkommastellen, aufgefüllt mit Nullen für einen FLOAT-Pin.

12.3.6.12 Containers (engl. für Behälter)

- HAL_HideTable
- HAL_Table
- State_Sensitive_Table
- HAL_HBox (deprecated)

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 BIT input pin which controls if it's child widgets are hidden or not.

Pin

`<Panel_basename>.<widgetname>`
in bit pin
If the pin is low then child widgets are visible which is the default state.

HAL_Table and HAL_Hbox
Have one HAL BIT input pin which controls if their child widgets are sensitive or not.

Pin , `<Panel_basename>.<widgetname>`
in bit pin
If the pin is low then child widgets are inactive which is the default state.

State_Sensitive_Table
Responds to the state to LinuxCNC's interpreter.
Optionally selectable to respond to *must-be-all-homed*, *must-be-on* and *must-be-idle*.
You can combine them. It will always be insensitive at Estop.
(Has no pin).

Warnung
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.

Tipp
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.

12.3.6.13 LED

Die `hal_led` simuliert eine echte Anzeige-LED.
Sie hat einen einzigen BIT-Eingangspin, der ihren Zustand steuert: EIN oder AUS.

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.

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.

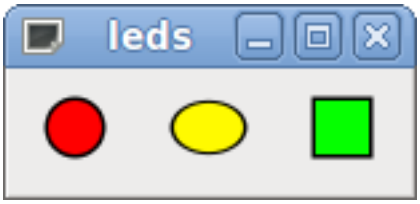


Abbildung 12.39: Beispiel LEDs

12.3.6.14 ProgressBar (engl. für Fortschrittsbalken)

Anmerkung
Dieses Widget wird möglicherweise entfernt.
Verwenden Sie stattdessen die Widgets HAL_HBar und HAL_VBar.

Der `HAL_ProgressBar` ist von gtk.ProgressBar abgeleitet und hat zwei Float-HAL-Eingangspins:

<widgetname>
den aktuell anzuzeigenden Wert

<widgetname>-scale
der maximale absolute Wert der Eingabe

`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 $-2^{24 \text{ bis } +2}24$.

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.

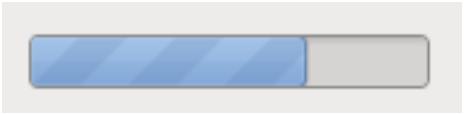


Abbildung 12.40: Beispiel HAL_ProgressBar

12.3.6.15 ComboBox

Die `HAL_ComboBox` ist von der gtk.ComboBox abgeleitet. Sie ermöglicht die Auswahl eines Wertes aus einer Dropdown-Liste.

Die `HAL_ComboBox` exportiert zwei HAL-Pins:

<widgetname>-f
Aktueller Wert, Typ FLOAT

<widgetname>-s
Aktueller Wert, Typ s32

`HAL_ComboBox` hat die folgende Eigenschaft, die in Glade gesetzt werden kann:

column
Der Spaltenindex. +
Typ s32. +
Gültiger Bereich von -1..100. +
Standardwert ist -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.

Im Spaltenmodus (engl. column Mode) (Spalte > -1) wird der gemeldete Wert aus dem ListStore-Array, wie in Glade definiert, ausgewählt. Typischerweise würde Ihre Widget-Definition also zwei Spalten im ListStore haben, eine mit Text, der im Dropdown angezeigt wird, und einen int- oder float-Wert, der für diese Auswahl verwendet wird.

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 http://www.youtube.com/watch?v=Z5_F-rW2cL8.

12.3.6.16 Bars (engl. für Balken)

HAL_Bar- und *HAL_VBar*-Widgets für horizontale und vertikale Balken, die Gleitkommawerte darstellen.

`HAL_Bar` und `HAL_VBar` haben jeweils einen Eingangs-FLOAT-HAL-Pin.

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.

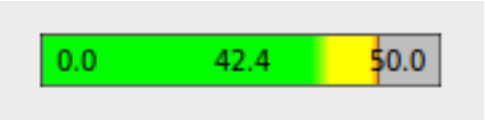


Abbildung 12.41: Horizontaler Balken

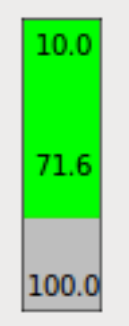


Abbildung 12.42: Vertikaler Balken

12.3.6.17 Meter

HAL_Meter ist ein Widget ähnlich dem PyVCP-Meter - es stellt einen Gleitkommawert dar.

`HAL_Meter` hat einen Eingangs-FLOAT-HAL-Pin.

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.

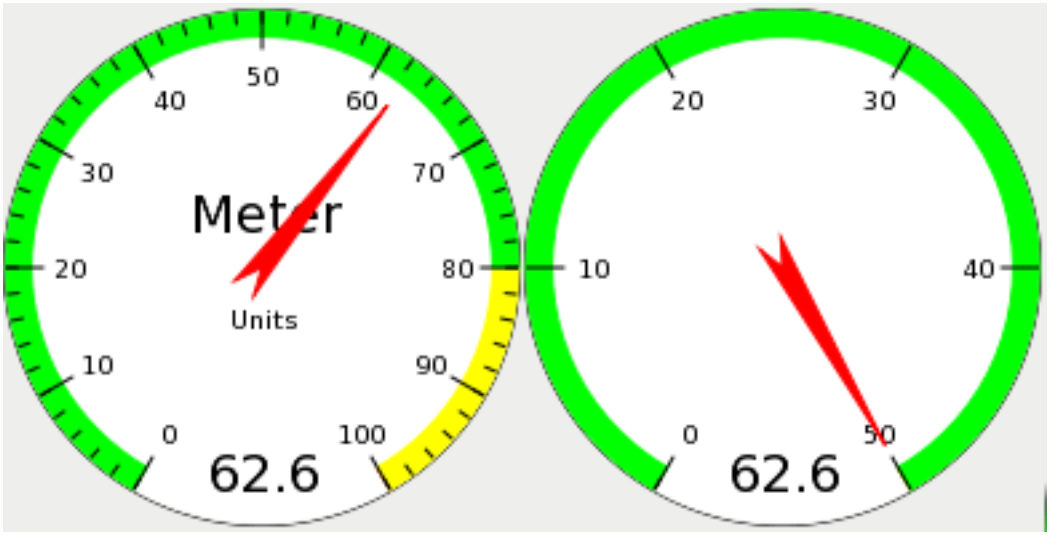


Abbildung 12.43: Beispiel HAL-Messgeräte

12.3.6.18 HAL_Graph

Dieses Widget dient zum Auftragen von Werten über die Zeit.

12.3.6.19 Gremlin tool path preview for NGC files

Gremlin is a plot preview widget similar to the AXIS preview window. It assumes a running LinuxCNC environment like AXIS or Touchy. To connect to it, inspects the INI_FILE_NAME environment variable. Gremlin displays the current NGC file - it does monitor for changes and reloads the ngc file if the file name in AXIS/Touchy changes. If you run it in a GladeVCP application when LinuxCNC is not running, you might get a traceback because the Gremlin widget can't find LinuxCNC status, like the current file name.

Gremlin exportiert keine HAL-Pins.

Gremlin hat die folgenden Eigenschaften:

Anzeige der Werkzeuggeschwindigkeit
Dies zeigt die Werkzeuggeschwindigkeit an.
Vorteinstellung = true.

befohlen zeigen
Dies wählt die zu verwendende DRO aus: befohlene oder tatsächliche Werte.
Voreinstellung = true.

metrische Einheiten verwenden
Dies legt fest was die DRO nutzt: metrische oder imperiale Einheiten.
Voreinstellung = true.

schnelle Bewegungen zeigen
Dies legt fest, dass der Plotter die schnelle Bewegungen zeigt.
Voreinstellung = true.

DTG anzeigen
Hiermit wird die Anzeige des Restweges auf der DRO ausgewählt.
Voreinstellung = true.

relativ anzeigen
Hiermit wird festgelegt, dass die Anzeige der Werte relativ zu den Koordinaten des Benutzersystems oder der Maschine erfolgen soll.
Voreinstellung = true.

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.

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.

Ausmaße anzeigen
Hiermit wird der Plotter angewiesen, die Ausdehnungen anzuzeigen.
Voreinstellung = true.

Werkzeug anzeigen
Hiermit wird der Plotter angewiesen, das Werkzeug zu zeichnen.
Voreinstellung = true.

Programm zeigen
TODO

Verwenden des Gelenkmodus
Wird in nicht trivialkins Maschinen (z.B. Robotern) verwendet.
Standard = false.

Rastergröße
Legt die Größe des Gitters fest (nur in den Ansichten X, Y und Z sichtbar). +
Standardwert ist 0

Standard-Maussteuerung verwenden
This disables the default mouse controls.
This is most useful when using a touchscreen as the default controls do not work well. You can programmatically add controls using Python and the handler file technique.
Default = 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 = Standard: 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 verschieben
- 4 = Links verschieben, mittig 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.

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, 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].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

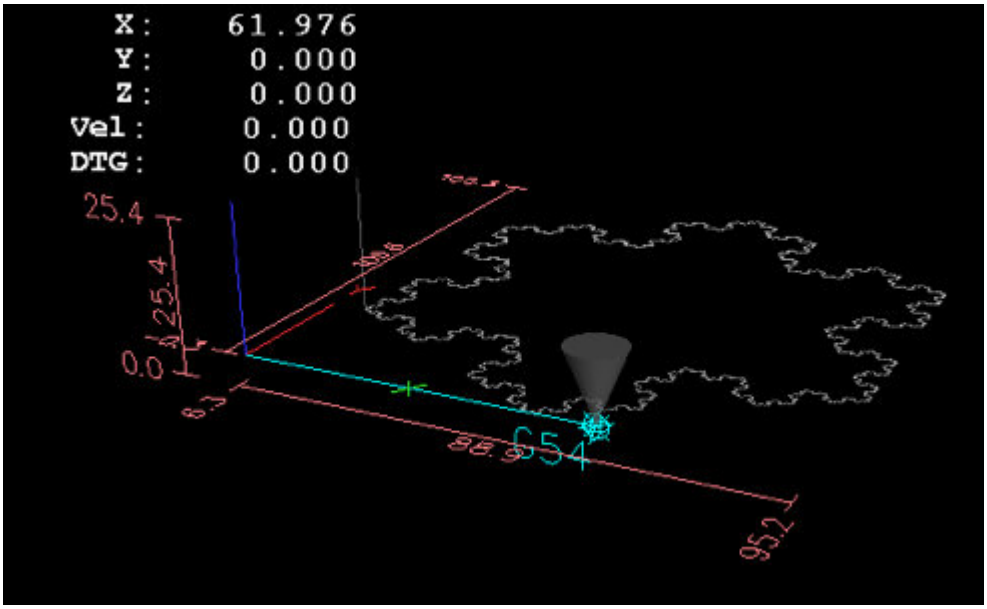


Abbildung 12.44: Gremlin Beispiel

12.3.6.20 HAL_Offset

Das `HAL_Offset`-Widget wird verwendet, um den Offset einer einzelnen Achse anzuzeigen.

`HAL_Offset` hat die folgenden Eigenschaften:

Gelenk Nummer
Dient zur Auswahl, welche Achse (technisch gesehen welches 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
```

Textvorlage für metrische Einheiten
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Textvorlage für imperiale Einheiten
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Referenztyp

```
0:G5x 1:Werkzeug 2:G92 3:Drehung um Z
```

12.3.6.21 DRO-Widget

Das DRO-Widget wird verwendet, um die aktuelle Achsenposition anzuzeigen.

Es hat die folgenden Eigenschaften:

Actual Position (tatsächliche Position)
Wählen Sie die Istposition (Rückmeldung) oder die Sollposition.

Textvorlage für metrische Einheiten
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Textvorlage für imperiale Einheiten
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Referenztyp

- `absolute` (Maschinen-Ursprung), or
- `relativ` (zum aktuellen Ursprung der Benutzerkoordinate - G5x) oder
- `distance-to-go` (engl. für verbleibender Weg) (relativ zum aktuellen Ursprung der Benutzerkoordinate).

Gelenk Nummer
Dient zur Auswahl, welche Achse (technisch gesehen welches 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
Anzeigeeinheiten::
  Dient zum Umschalten der Anzeigeeinheiten zwischen metrisch und imperial.
Hinweise::
  - Wenn die Anzeige rechtsbündig sein soll, setzen Sie die X-Ausrichtung auf 1.0
  - Wenn du andere Farben oder eine andere Größe oder einen anderen Text möchtest, ändere die Attribute im Glade-Editor (z.B. ist Skalierung ein guter Weg, um die Größe des Textes zu ändern)
  - 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. Es somit alles was ↩
      mit einem gtk Label getan werden kann auch auf ein DRO Widget angewendet ↩
      werden.
```

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

Verwenden Sie goobject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property(”display_units_mm”,True)
[widget name].set_property(”actual”,True)
[widget name].set_property(”mm_text_template”,”%f”)
[widget name].set_property(”imperial_text_template”,”%f”)
[widget name].set_property(”Joint_number”,3)
[widget name].set_property(”reference_type”,3)
```

There are two Python methods:

```
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

12.3.6.22 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.

`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.

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_orde`
    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.

Abbildung 12.45: Beispiel: Drei Combi_DRO in einem Fenster

```
X = Relativer Modus +
Y = Absoluter Modus +
Z = DTG-Modus
```

12.3.6.23 IconView (Dateiauswahl)

Dies ist ein Touchscreen-freundliches Widget zur Auswahl einer Datei und zum Wechseln von Verzeichnissen.

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)

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”)
```

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** zurück:

  - Der Name der Schaltfläche ist einer der folgenden: `btn_home`, `btn_dir_up`, `btn_sel_prev`, `btn_sel_nex`
    `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.

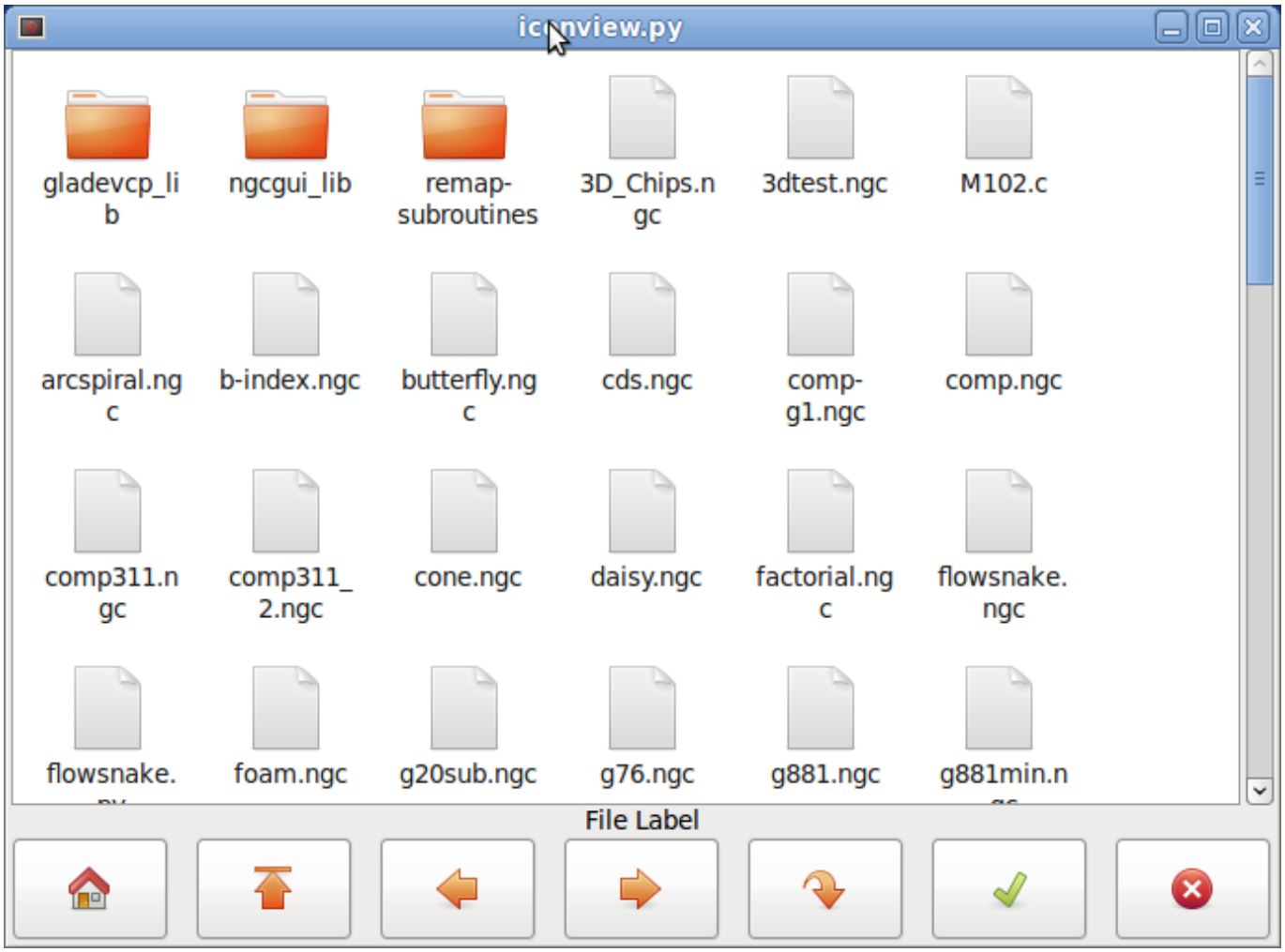


Abbildung 12.46: Iconview Beispiel

12.3.6.24 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.

`Rechner` (engl. calculator) hat folgende Eigenschaften:

Ist bearbeitbar
: Damit kann die Eingabeanzeige über eine Tastatur eingegeben werden.

Schriftart einstellen
: Hier können Sie die Schriftart für die Anzeige einstellen.

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.

12.3.6.25 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.

Anmerkung
LinuxCNC erfordert Mapping von Werkzeug-Aufrufe, um Verschleiß-Offsets anzuwenden.

`tooleditor` hat die folgenden Eigenschaften:

Versteckte Spalten
: 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.

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).

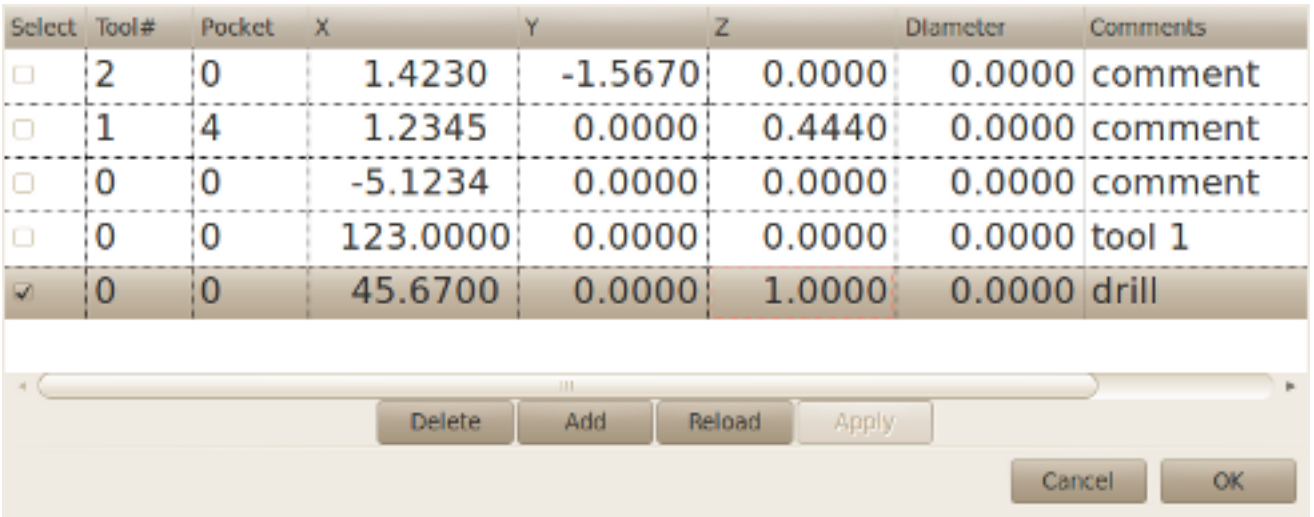


Abbildung 12.47: Werkzeug-Editor (engl. tooleditor) Beispiel

12.3.6.26 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.

Es hat die folgenden Eigenschaften:

Versteckte Spalten
: Eine Liste der auszublendenden Spalten ohne Leerzeichen. Die Spalten werden (in dieser Reihenfolge) wie folgt bezeichnet: `xyzabcuvwt`.
Sie können jede der Spalten ausblenden.

Ausgeblendete Zeilen
: Eine Liste der auszublendenden Zeilen ohne Leerzeichen.
Die Zeilen werden (der Reihe nach) wie folgt bezeichnet: `0123456789abc`.
Sie können jede der Zeilen ausblenden.

Pango-Schriftart
: Legt Art und Größe der Schriftart fest.

Hervorhebungsfarbe
: Beim Bearbeiten ist dies die Hervorhebungsfarbe.

Aktive Farbe
: Wenn `OffsetPage` ein aktives Benutzerkoordinatensystem erkennt, wird diese Farbe für den Text verwendet.

Textvorlage für metrische Einheiten
: Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

Textvorlage für imperiale Einheiten
: Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

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.

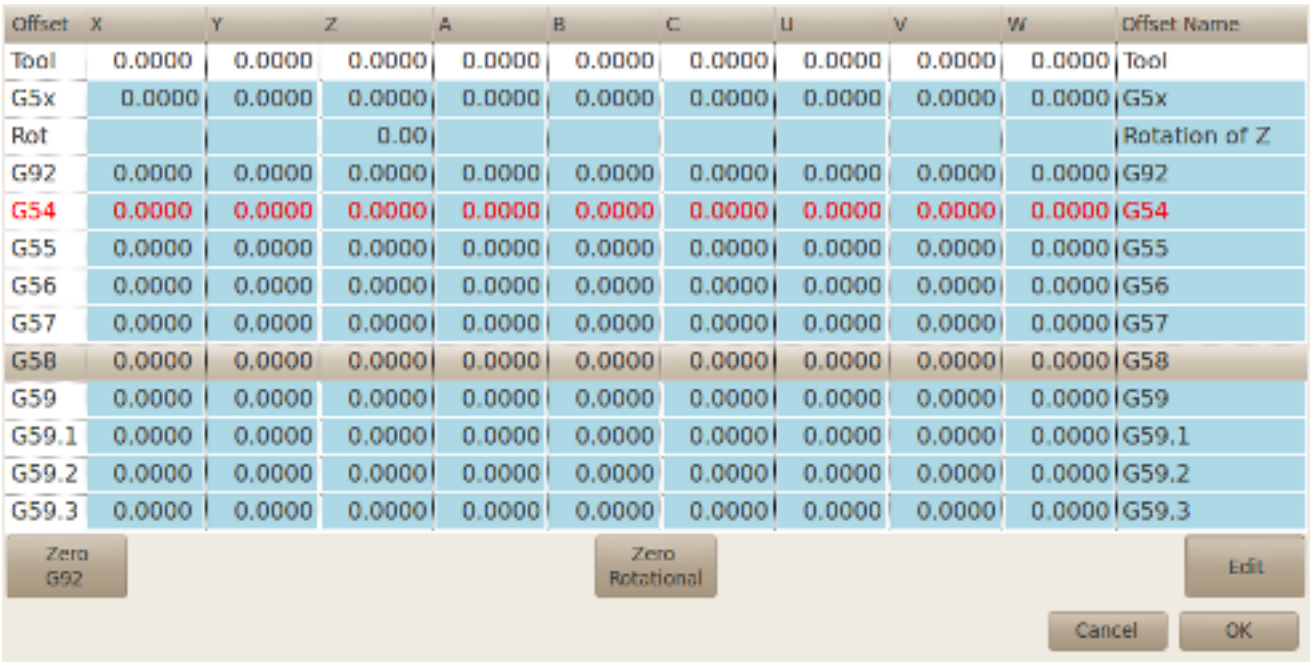


Abbildung 12.48: Beispiel für eine Offset-Seite

12.3.6.27 HAL_sourceview-Widget

Dies ist für die Anzeige und einfache Bearbeitung von G-Code. Es sucht nach `.ngc`-Hervorhebungsspezifikati
in `~/share/gtksourceview-2.0/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.

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

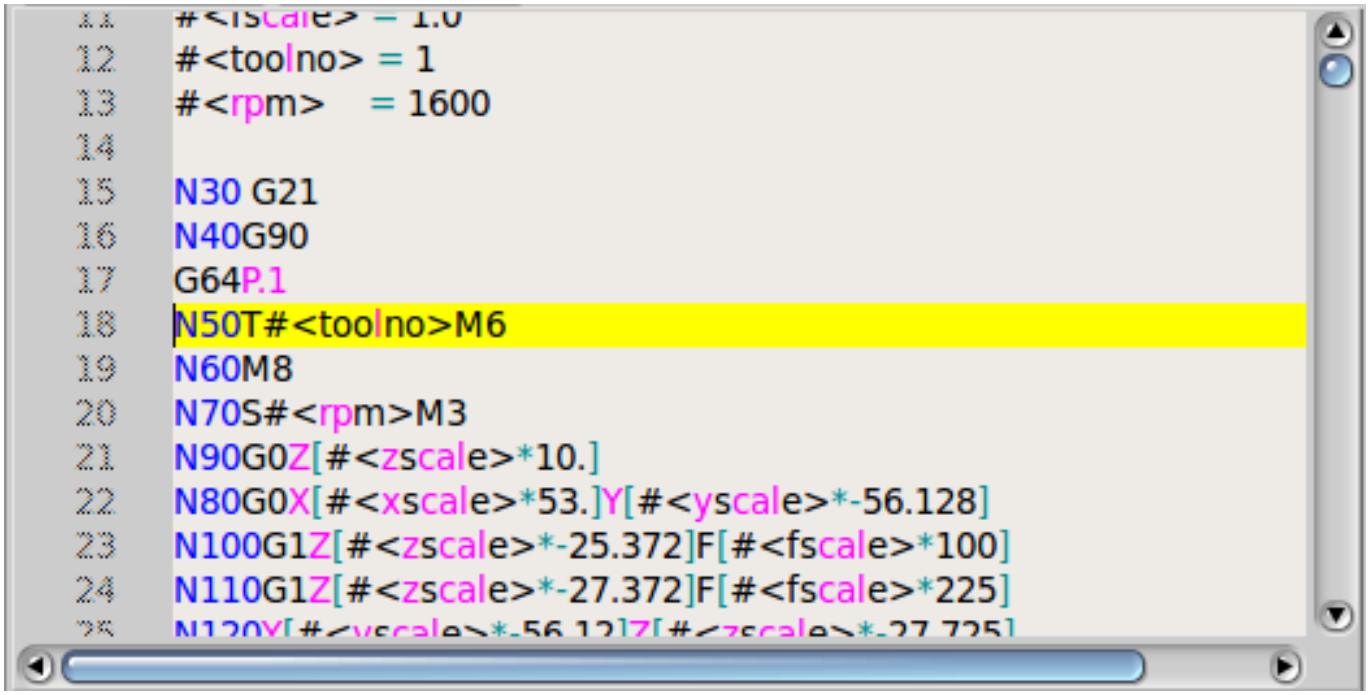


Abbildung 12.49: Quellen-Ansicht (engl. sourceview) Beispiel

12.3.6.28 MDI-Geschichte

Dient zur Anzeige und Eingabe von MDI-Codes.
Sie wird automatisch ausgegraut, wenn MDI nicht verfügbar ist, z. B. bei Not-Aus und Programmlauf.

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

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)
```

12.3.6.29 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:

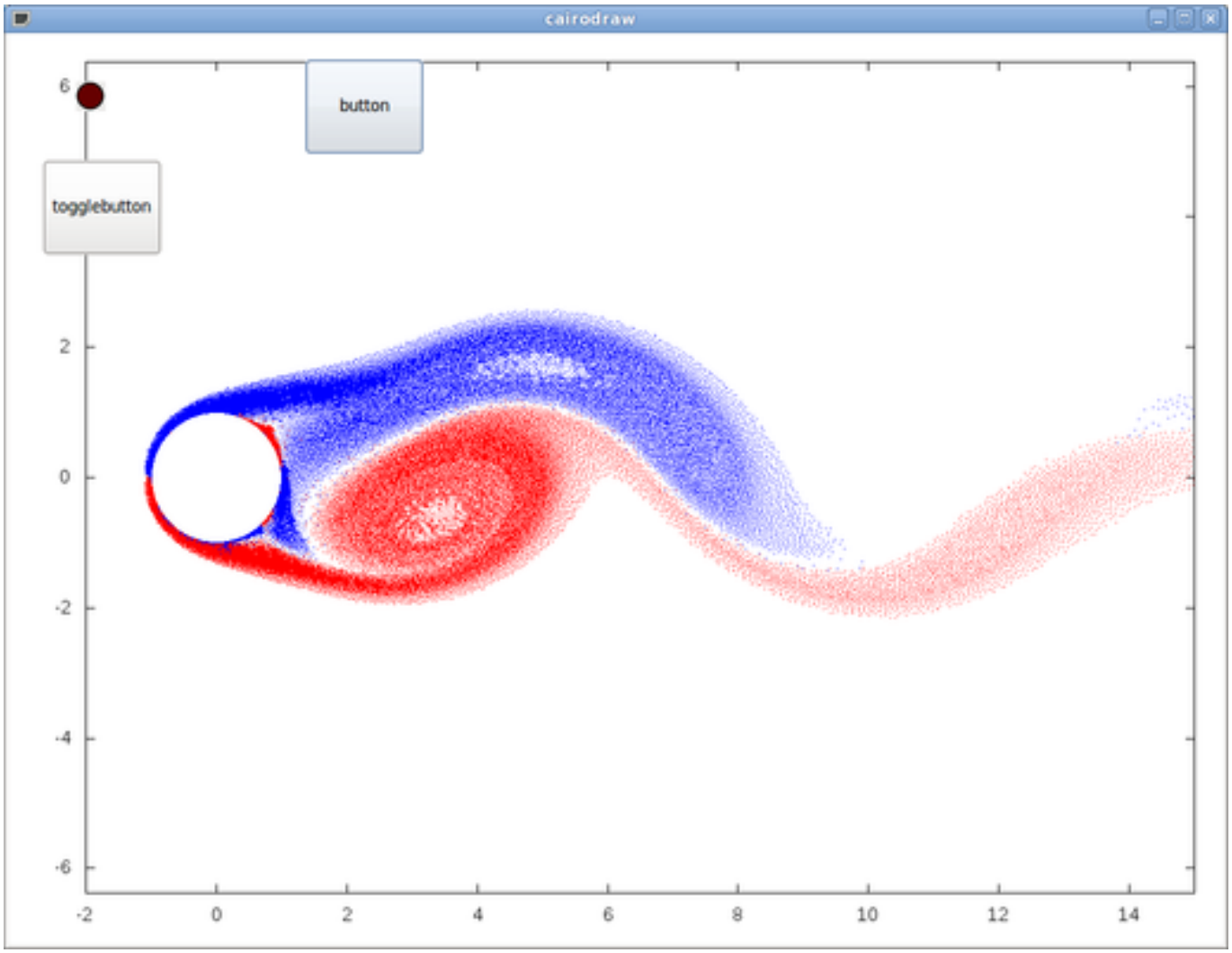


Abbildung 12.50: HAL-Widgets in einem Bitmap Beispiel

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.

Anmerkung
Other than HAL widgets, which interact with HAL pins, VCP Actions interact with LinuxCNC and the G-code interpreter.

VCP Action Widgets sind von dem `Gtk.Action` Widget abgeleitet.

Das Action-Widget in Kürze:

- ist ein in Glade verfügbares Objekt
- 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..)
- 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:

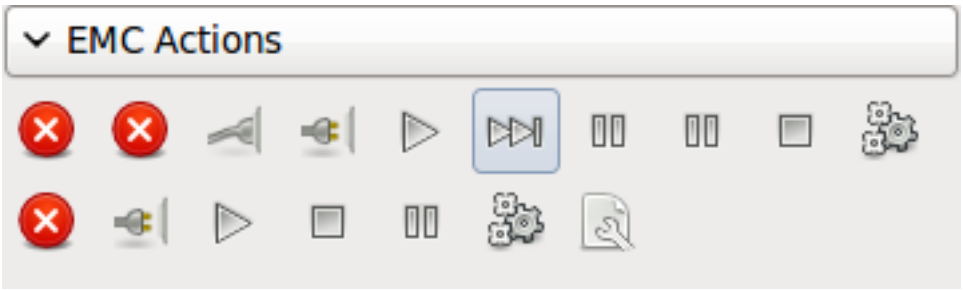


Abbildung 12.51: Action-Widgets

Hover über Werkzeugspitzen (engl. tooltips) liefert eine Beschreibung.

12.3.7.1 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.

12.3.7.2 VCP Action Python

This widget is used to execute small arbitrary Python code.

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` for access to LinuxCNC’s raw status via the LinuxCNC Python module.
- `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

12.3.7.3 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.

12.3.7.4 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.

12.3.7.5 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:

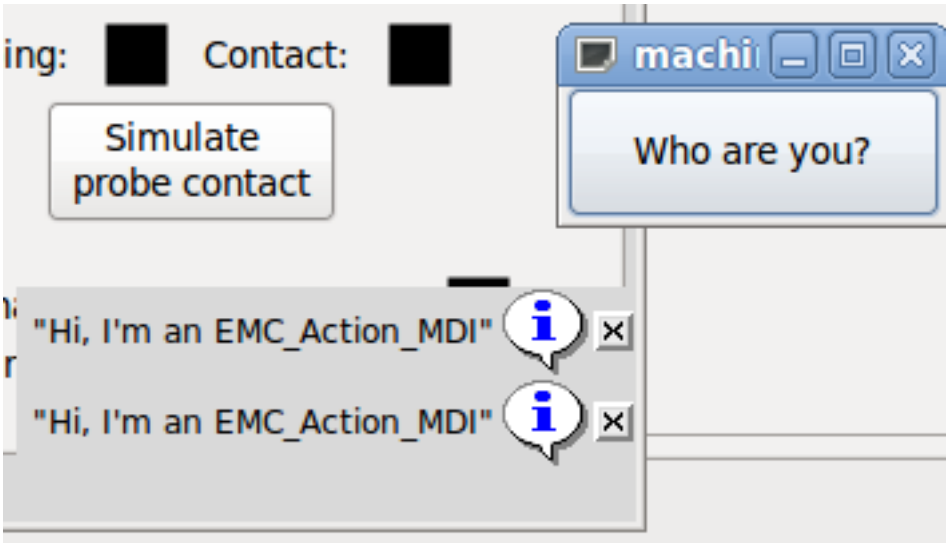


Abbildung 12.52: Action_MDI Einfaches Beispiel

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.

12.3.7.6 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 ihren aktuellen Wert als Parameter in einem MDI-Befehl übergeben.
- Die HAL SpinBox hat einen HAL-Pin vom Typ float mit dem Namen speed-f (siehe HalWidgets-Beschreibung).
-
- 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:

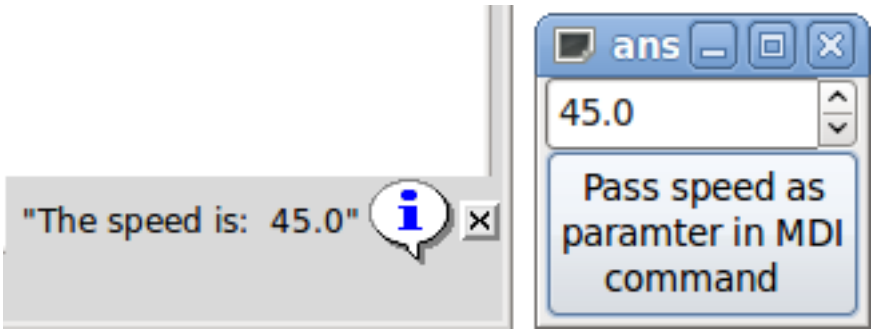


Abbildung 12.53: Action_MDI Parameterübergabe Beispiel

12.3.7.7 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/`

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:

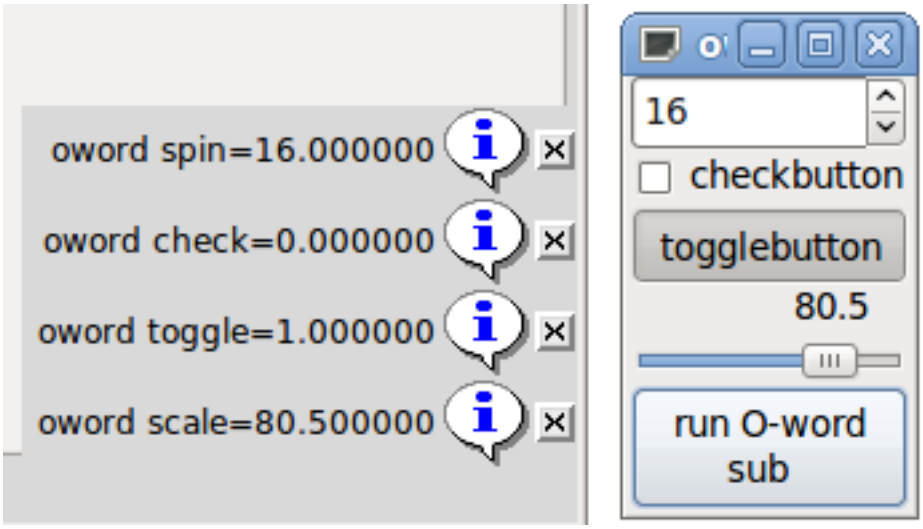


Abbildung 12.54: Action_MDI Erweitertes Beispiel

12.3.7.8 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-Unterprogramm 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.

Here's an example how a feed value might be saved and restored by such handlers (note that LinuxCNC command and status channels are available as `self.linuxcnc` and `self.stat` through the VCP_ActionBase class):

```
    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.

Anmerkung
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.

12.3.7.9 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

12.3.8.1 Benutzerdefinierte Aktionen

Die meisten Widgetsets und die dazugehörigen Editoren für die Benutzeroberfläche unterstützen das Konzept der Callback-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.

12.3.8.2 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')
```

More information can be found here: GladeVCP Libraries modules There is a sample configuration that demonstrates using the core library with GladeVCP's action Python widgets and with a Python handler file. Try loading *sim/axis/gladevcp/gladevcp_panel_tester*.

12.3.8.3 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.

What the +-u+ flag does: all Python functions in this file are collected and setup as potential callback handlers for your Gtk widgets - they can be referenced from Glade *Signals* tabs. The callback handlers are called with the particular object instance as parameter, like the GtkButton instance above, so you can apply any GtkButton method from there.

Oder machen Sie etwas Nützlicheres, wie den Aufruf eines MDI-Befehls!

12.3.8.4 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.

12.3.8.5 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.

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.

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.

For GladeVCP panel which respond to HAL inputs it may be important that the handler code can tell that the GladeVCP panel is currently active and displayed. For example a panel inside the Touchy interface might well need to perform an action when the switch connected to touchy.cycle-start is operated (in the same way that the native tabs respond differently to the same button).
To make this possible, a signal is sent from the GUI (at the time of writing, only Touchy) to the embedded tab. The signal is of type ”Gladevcp” and the two messages sent are ”Visible” and ”Hidden”. (Note that the signals have a fixed length of 20 characters so only the first characters should be used in any comparison, hence the [:7] below.) A sample handler for these signals is:

```
    # 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)
```

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` - refers to the HAL component under construction,
- `builder` - widget tree - result of reading the UI definition (either referring to a GtkBuilder or libglade-type object),
- `useropts` - a list of strings collected from the GladeVCP command line `-U <useropts>` option.

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.

12.3.8.6 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.

12.3.8.7 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.

12.3.8.8 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.

12.3.8.9 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.

12.3.8.10 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.
- Describe these decisions in your handler class' `\__init()__` method through a nested dictionary as follows:

```
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.

12.3.8.11 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.

Achtung
Do not use `window1` (the toplevel window) to connect a *destroy* event. Due to the way a GladeVCP panel interacts with AXIS, if a panel is embedded within AXIS, **window1 will not receive destroy events properly**. However, since on shutdown all widgets are destroyed, anyone will do. Recommended: use a second-level widget - for instance, if you have a table container in your panel, use that.

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.

Achtung
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.

12.3.8.12 Status speichern, wenn Strg-C gedrückt wird

By default, the reaction of GladeVCP to a Ctrl-C event is to just exit - *without* saving state. To make sure that this case is covered, add a handler call `on_unix_signal` which will be automatically be called on Ctrl-C (actually on the SIGINT and SIGTERM signals). Example:

```
def on_unix_signal(self,signum,stack_frame):
    print("on_unix_signal(): signal %d received, saving state" % (signum))
    self.ini.save_state(self)
```

12.3.8.13 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.

12.3.8.14 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.HAL_BIT, hal. ↩
    HAL_IN))
```

To get a callback when this pin's value changes, associate a `value-change` callback with this pin, add:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

and define a callback method (or function, in this case leave out the `self` parameter):

```
# 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()))
```

12.3.8.15 Hinzufügen von Timern

Da GladeVCP Gtk-Widgets verwendet, die sich auf die Basisklasse GObject 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 1 ms), verwenden Sie dies:
# glib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
glib.timeout_add_seconds(1, self._on_timer_tick)
```

12.3.8.16 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.HAL_FLOAT, hal. ↩
            HAL_IN))
        self.max_value.connect('value-changed', self._on_max_value_change)

  inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
  mmin = float(inifile.find("METER", "MIN") or 0.0)
        self.meter = self.builder.get_object('meter')
        self.meter.min = mmin


def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

12.3.8.17 Beispiele und die Entwicklung Ihrer eigenen GladeVCP-Anwendung

Visit `linuxcnc_root_directory/configs/apps/gladevcp` for running examples and starters for your own projects.

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.

6. *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/ap`

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.
- Run GladeVCP or AXIS from a terminal window. If you get Python errors, check whether there's still a `/usr/lib/python2.6/dist-packages/hal.so` file lying around besides the newer `/usr/lib/python2.6` (note the underscore); if yes, remove the `hal.so` file. It has been superseded by hal.py in the same directory and confuses the import mechanism.
- 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 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_GLADEV 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.

Anmerkung
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 Library modules

Libraries are prebuilt Python modules that give added features to GladeVCP. In this way you can select what features you want - yet don’t have to build common ones yourself.

12.4.1 Info

Info is a library to collect and filters data from the INI file.

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 = int(self.INI.find(”KINS”,”JOINTS”)or 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.find(section, ”TYPE”) or ”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 = int(self.INI.find(”TRAJ”, ”SPINDLES”) or 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”)) or None

# 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

This library is used to command LinuxCNC's motion controller. It tries to hide incidental details and add convenience methods for developers.

Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Action
```

To instantiate the module so you can use it add this Python code to your instantiate section:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

ACTION = Action()
```

To access Action commands use general syntax such as these:

```
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 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.

12.5.1 Schaukasten

Einige Beispiele für mit QtVCP erstellte Bildschirme und virtuelle Bedienfelder:

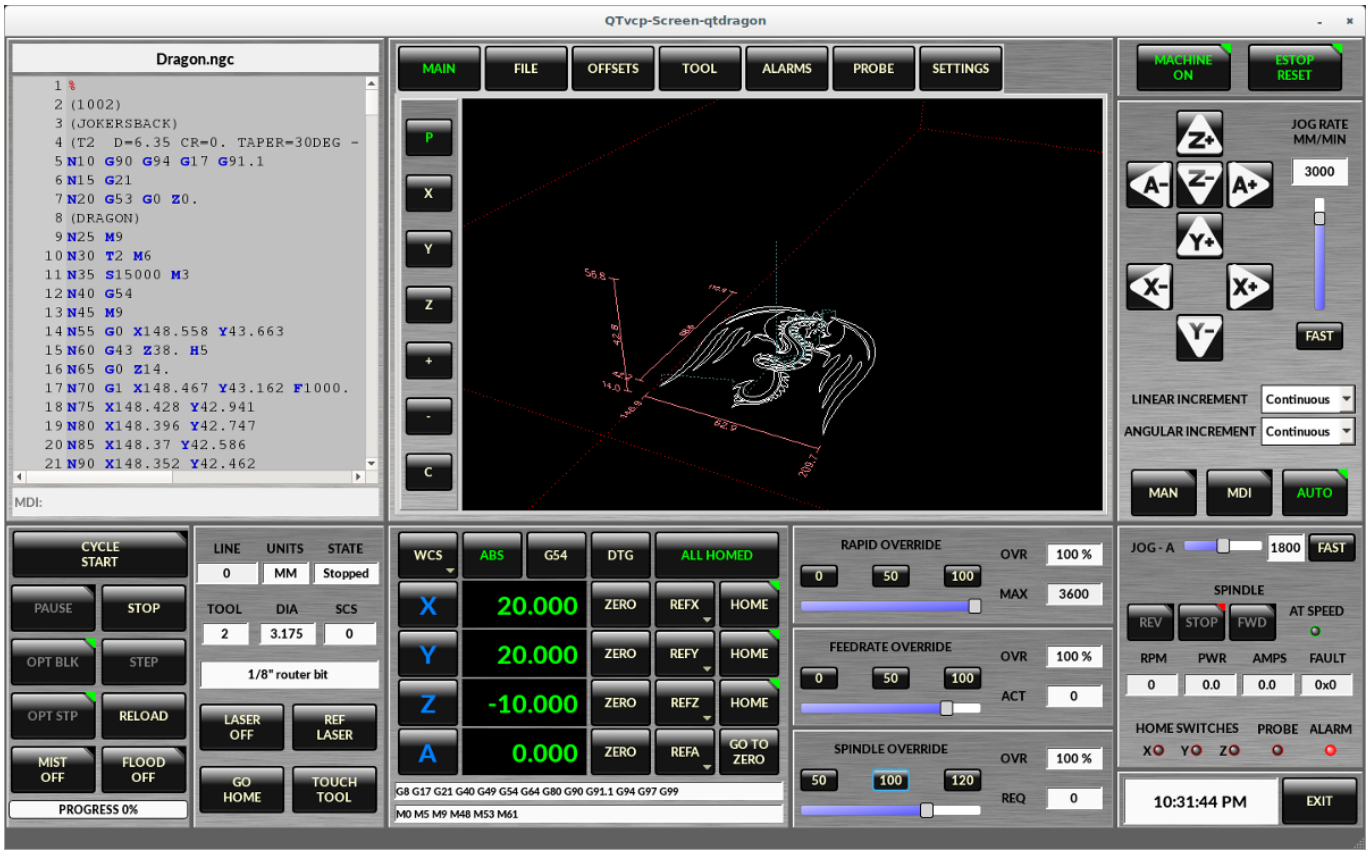


Abbildung 12.55: QtDragon - 3/4-Achsen-Beispiel

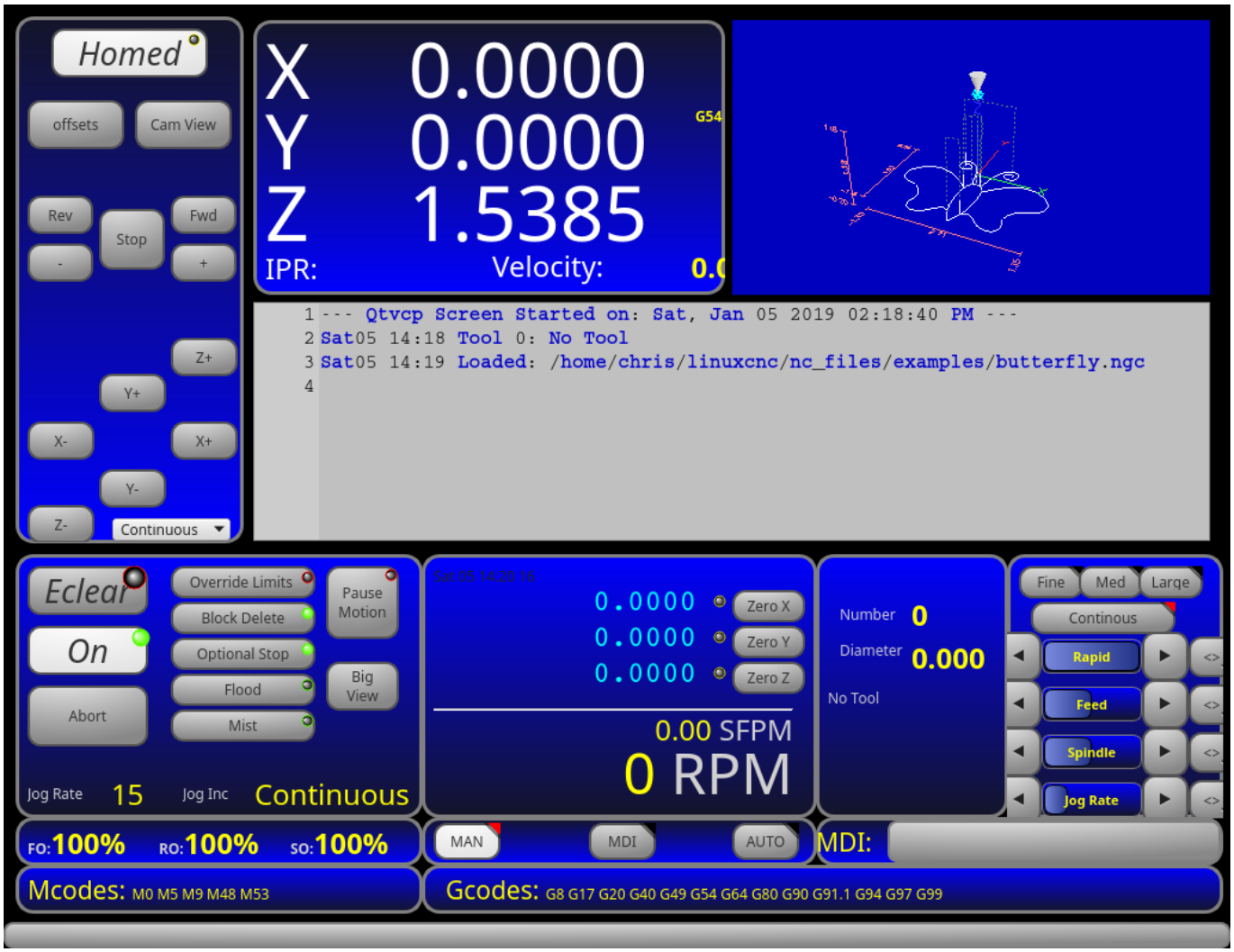


Abbildung 12.56: QtDefault - 3-Achsen-Beispiel

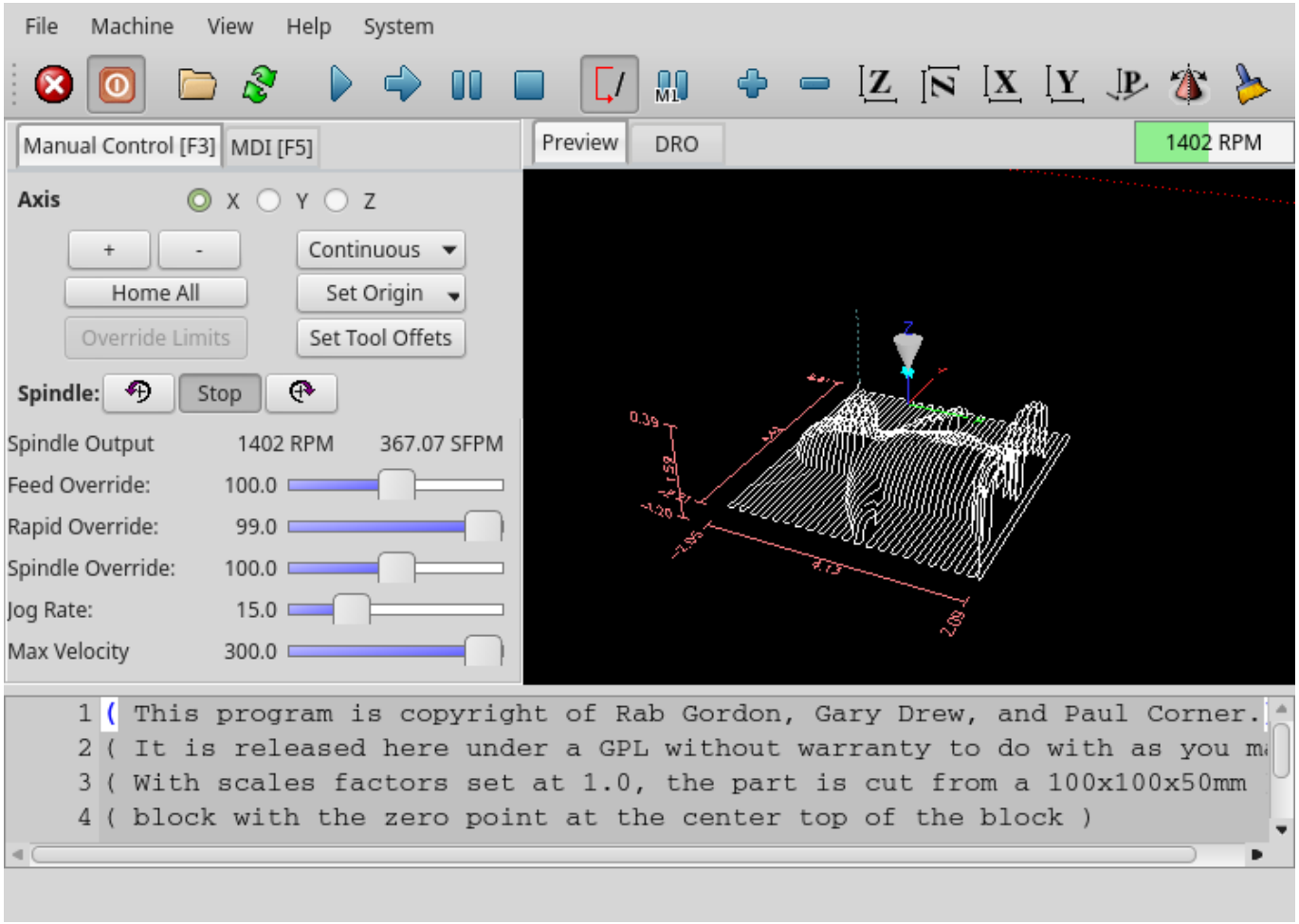


Abbildung 12.57: QtAxis - Beispiel für selbsteinstellende Achsen

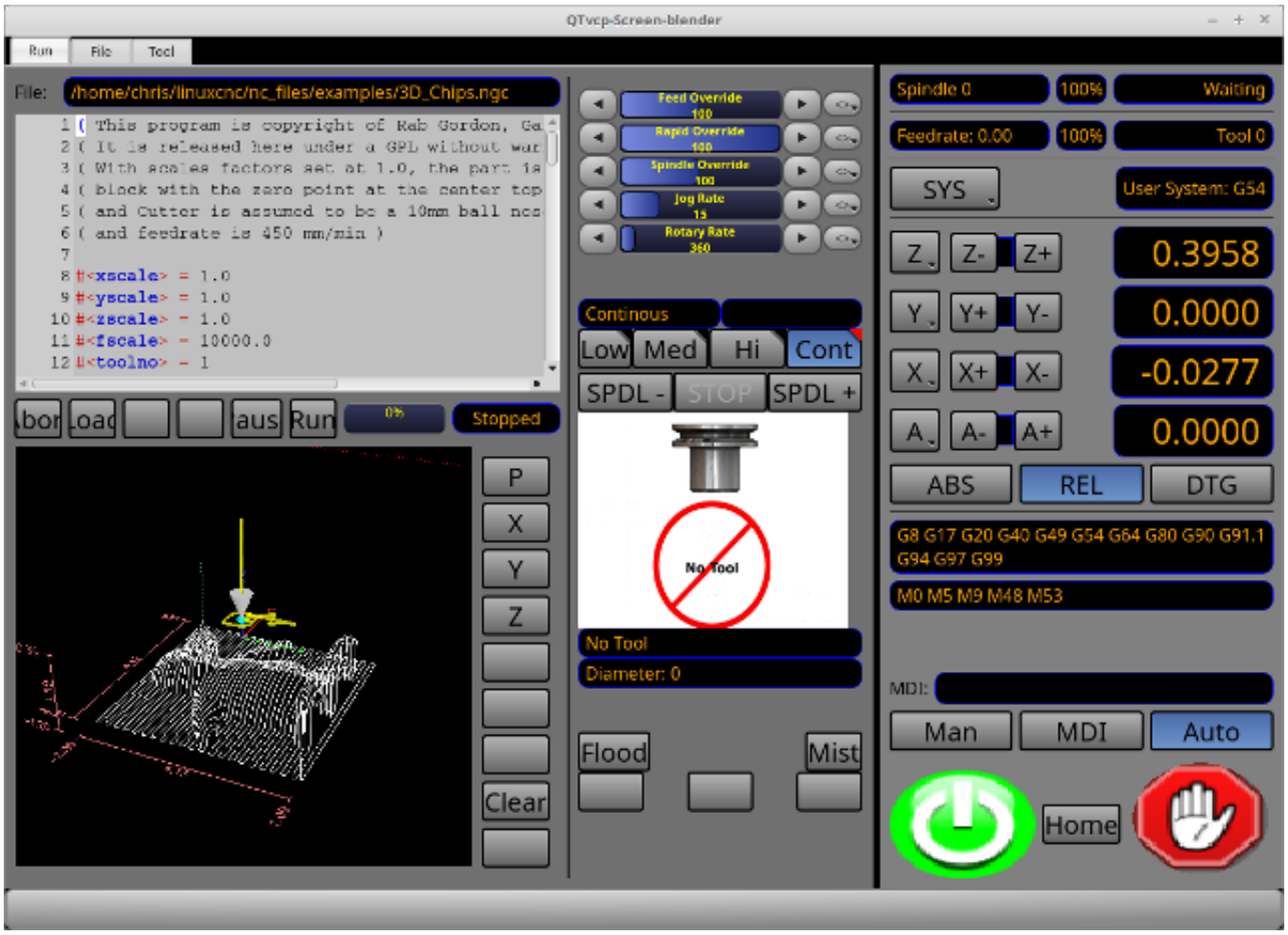


Abbildung 12.58: Blender - 4-Achsen-Beispiel

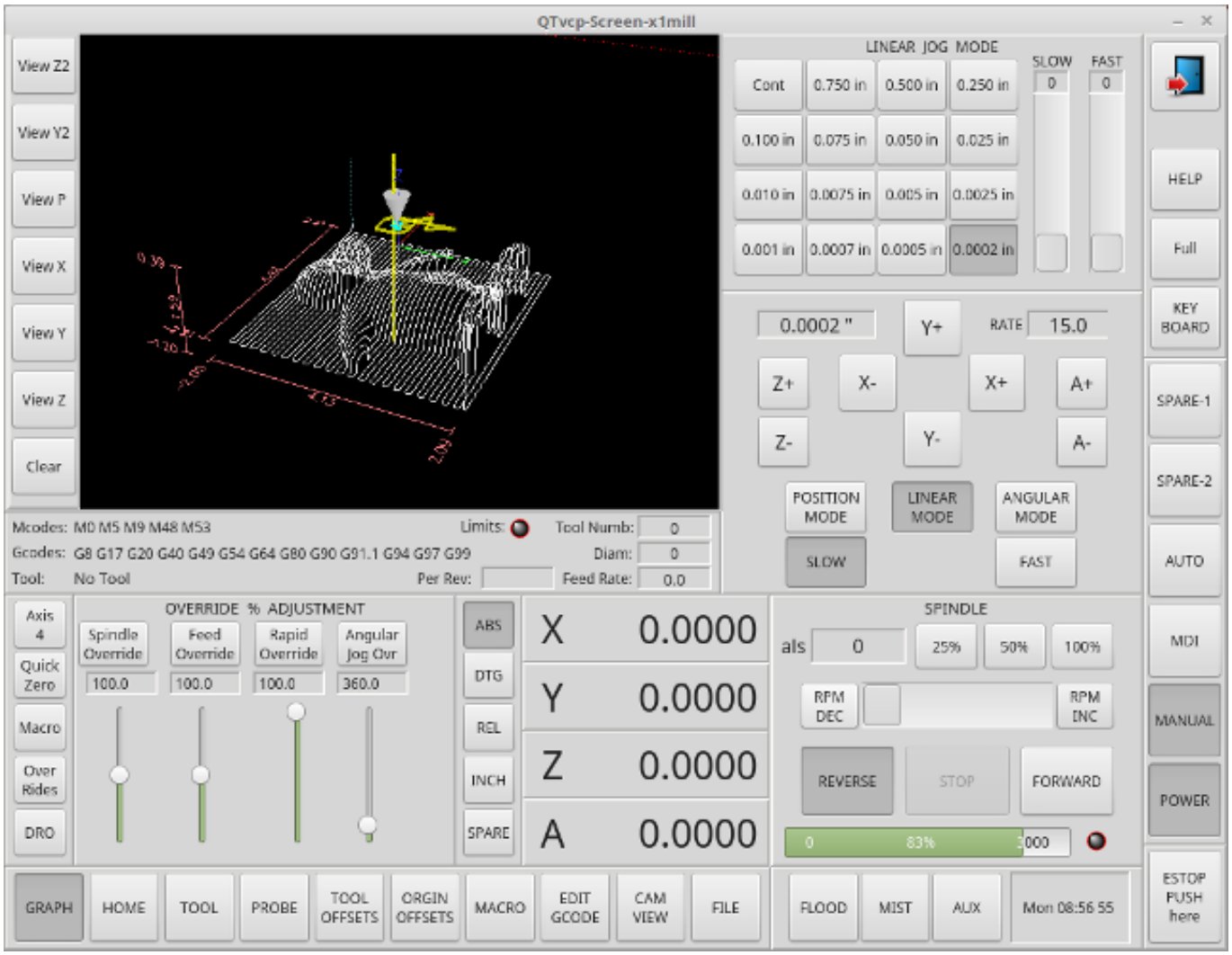


Abbildung 12.59: X1mill - 4-Achsen-Beispiel

Abbildung 12.60: cam_align – Kameraausrichtung VCP

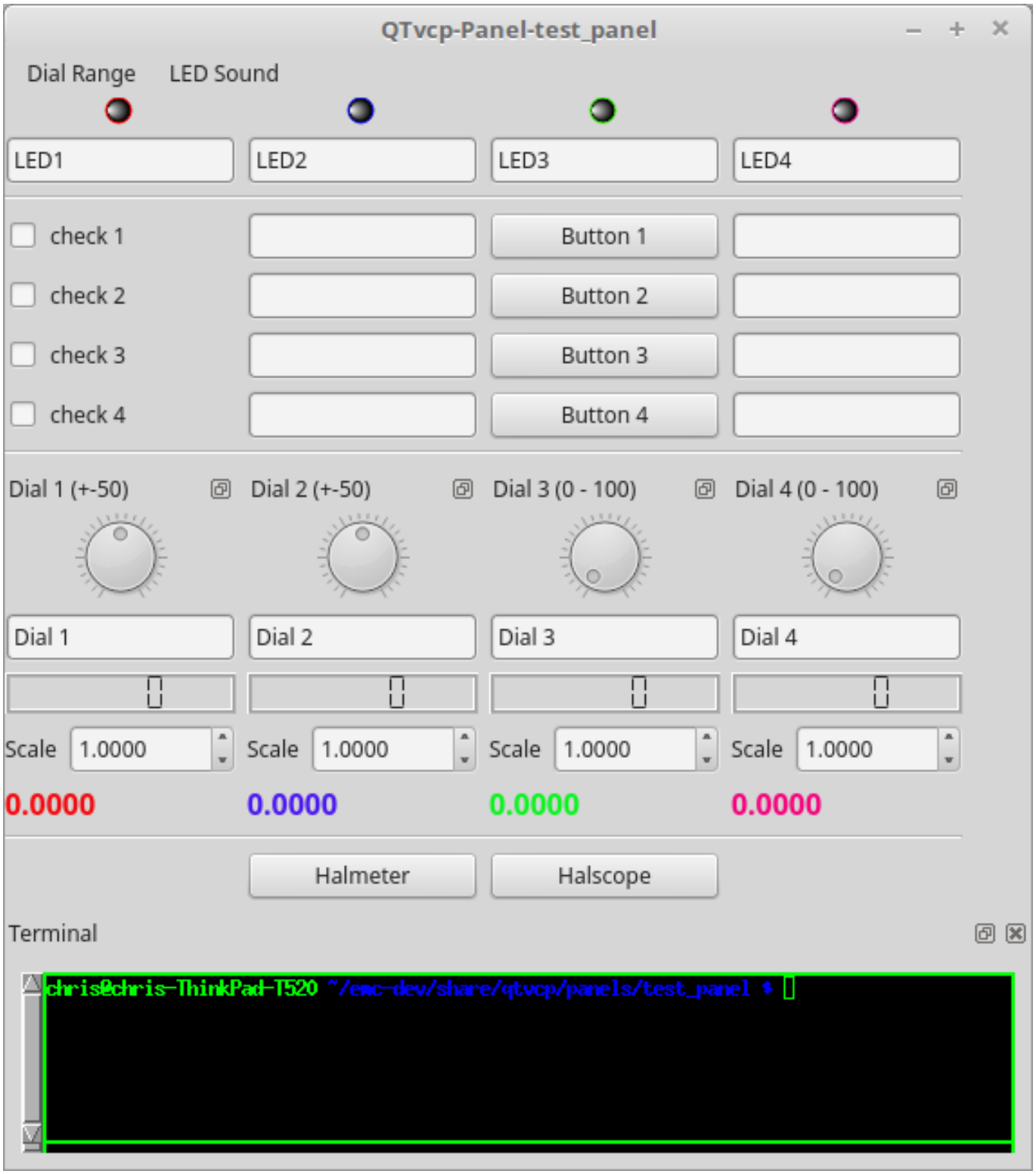


Abbildung 12.61: test_panel - Test Panel VCP

12.5.2 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.

12.5.2.1 QtVCP Widgets

QtVCP nutzt die **PyQt5-Toolkits** für die Einbeziehung von LinuxCNC.

Widget is the *general name for user interface objects* such as buttons and labels in PyQT5.

You are free to use any available **default widgets** in the Qt Designer editor.

There are also **special widgets** made for LinuxCNC that make integration easier.
These are split in three heading on the left side of the editor:

- One is for *HAL only widgets*;
- One is for *CNC control widgets*;
- One is for *Dialog widgets*.

You are free to mix them in any way on your panel.

A very important widget for CNC control is the **`ScreenOptions` widget**: it does not add anything visually to the screen but, allows important details to be selected rather then be coded in the handler file.

12.5.2.2 INI-Einstellungen

If you are using QtVCP to make a CNC control screen, in the INI file, in the `[DISPLAY]` section, add a line with the following pattern:

```
DISPLAY = qtvcp <options> <screen_name>
```

Anmerkung
All `<options>` must appear before `<screen_name>`.

Optionen

- `-d` Debugging on
- `-a` Set window always on top

- `-c` Name der HAL-Komponente. Standardmäßig wird der UI-Dateiname verwendet.
- `-g` Geometry: WIDTHxHEIGHT+XOFFSET+YOFFSET. Example: `-g 200x400+0+100`
- `-m` Fenster maximieren
- `-f` Vollbild des Fensters
- `-t`-Design (engl. theme). Standard ist das Systemdesign
- -x 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` Dateipfad einer Ersatz-Handler-Datei
- `-o` Übergibt einen String an die Handler-Datei von QtVCP unter der Listenvariablen `self.w.USEROPTIONS_` kann mehrfach sein -o

<Bildschirm_name> <<Bildschirmname>` ist der *Basisname der .ui und _handler.py Dateien*.
Wenn `<Bildschirmname>` fehlt, wird der Standardbildschirm geladen.

QtVCP geht davon aus, dass die UI-Datei und die Handler-Datei den **gleichen Basisnamen** verwenden.

QtVCP sucht nach Dateien zuerst im LinuxCNC-Konfigurationsverzeichnis, das gestartet wurde, 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.

12.5.2.3 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.

12.5.2.4 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** haben.

Falls vorhanden, wird eine Handler-Datei geladen.
Es ist nur eine Datei erlaubt.

12.5.2.5 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 sind vorgefertigte *Python-Module*, die **Funktionen** zu QtVCP hinzufügen. Auf diese Weise können Sie auswählen, welche Funktionen Sie wünschen - und müssen nicht selbst bauen. Solche Bibliotheken umfassen:

- `audio_player`
- `aux_program_loader`
- `keybindings`
- `message`
- `preferences`
- `notify`
- `virtual_keyboard`
- `machine_log`

12.5.2.6 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.

To see available themes, they can be loaded by running the following command in a terminal:

```
qtvcp -d -t <theme_name>
```

QtVCP kann auch mit *Qt-Stylesheets (QSS)* unter Verwendung von CSS angepasst werden.

12.5.2.7 Lokale Dateien

If present, local UI files in the configuration folder will be loaded instead of the stock UI files.

Lokale UI-Dateien ermöglichen es Ihnen, anstelle der Standardbildschirme Ihre eigenen Designs 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`.

12.5.2.8 Veränderung mitglieferter Bildschirmmasken

Es gibt *drei Möglichkeiten*, einen Bildschirm/Panel anzupassen.

Minor StyleSheet Changes Stylesheets können zum **Setzen von Qt-Eigenschaften** verwendet werden.
Wenn ein Widget Eigenschaften verwendet, können diese normalerweise durch Stylesheets verändert werden.
z.B.:

```
State_LED #name_of_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
  }
```

Minor Python Code Changes Eine weitere Python-Datei kann verwendet werden, um dem Bildschirm Befehle hinzuzufügen, nachdem die Handler-Datei geparst wurde.

In der *INI-Datei* unter der Überschrift `[DISPLAY]` einfügen **`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.
Beispiel:

```
[ANZEIGE]
USER_COMMAND_FILE = CONFIGDIRECTORY/<Bildschirm_name_hinzugefuegte_Befehle>
```

If no entry is found in the *INI*, QtVCP will look in the **default path**.
The default path is in the configuration directory as a hidden file using the screen basename and rc, ie: **`CONFIGDIRECTORY/.<screen_name>rc`**

Diese Datei wird als Python-Code im **handler-Dateikontext** gelesen und ausgeführt.

Nur lokale Funktionen und lokale Attribute können referenziert werden.
Globale Bibliotheken können nicht referenziert werden. (üblicherweise als große Worte ohne vorangestelltes Selbst).

Was verwendet werden kann, mag je nach Bildschirm und Entwicklungszyklus variieren.

For a valid example:

```
self.w.setWindowTitle('Mein Titel-Test')
```

Full Creative Control 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_dialog
  ```

- 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.
- Bestätigen, um alle Dateien zu kopieren
- Löschen Sie die Dateien, die Sie nicht ändern möchten, damit die Originaldateien verwendet werden.

12.5.3 VCP-Paneele

QtVCP kann verwendet werden, um Bedienfelder zu erstellen, die mit **HAL** verbunden sind.

12.5.3.1 Builtin 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.

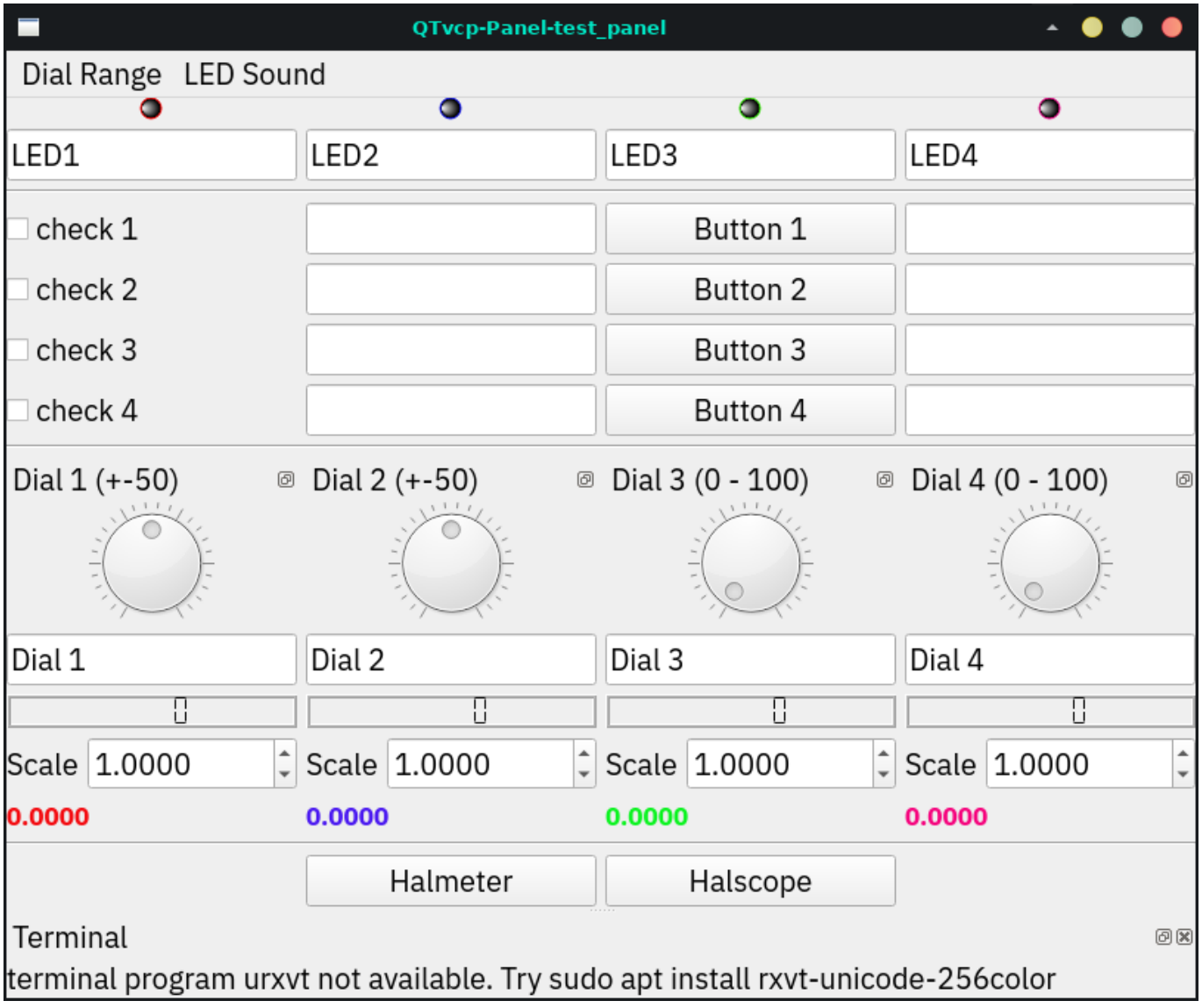


Abbildung 12.62: QtVCP HAL Test Integriertes Panel

`cam_align`
: Ein Kameraanzeige-Widget für die Rotationsausrichtung.

sim_panel
A small control panel to simulate MPG jogging controls etc.
For simulated configurations.

Abbildung 12.63: QtVCP Sim Builtin Panel

vismach_mill_xyz
3D openGL view of a 3-Axis milling machine.

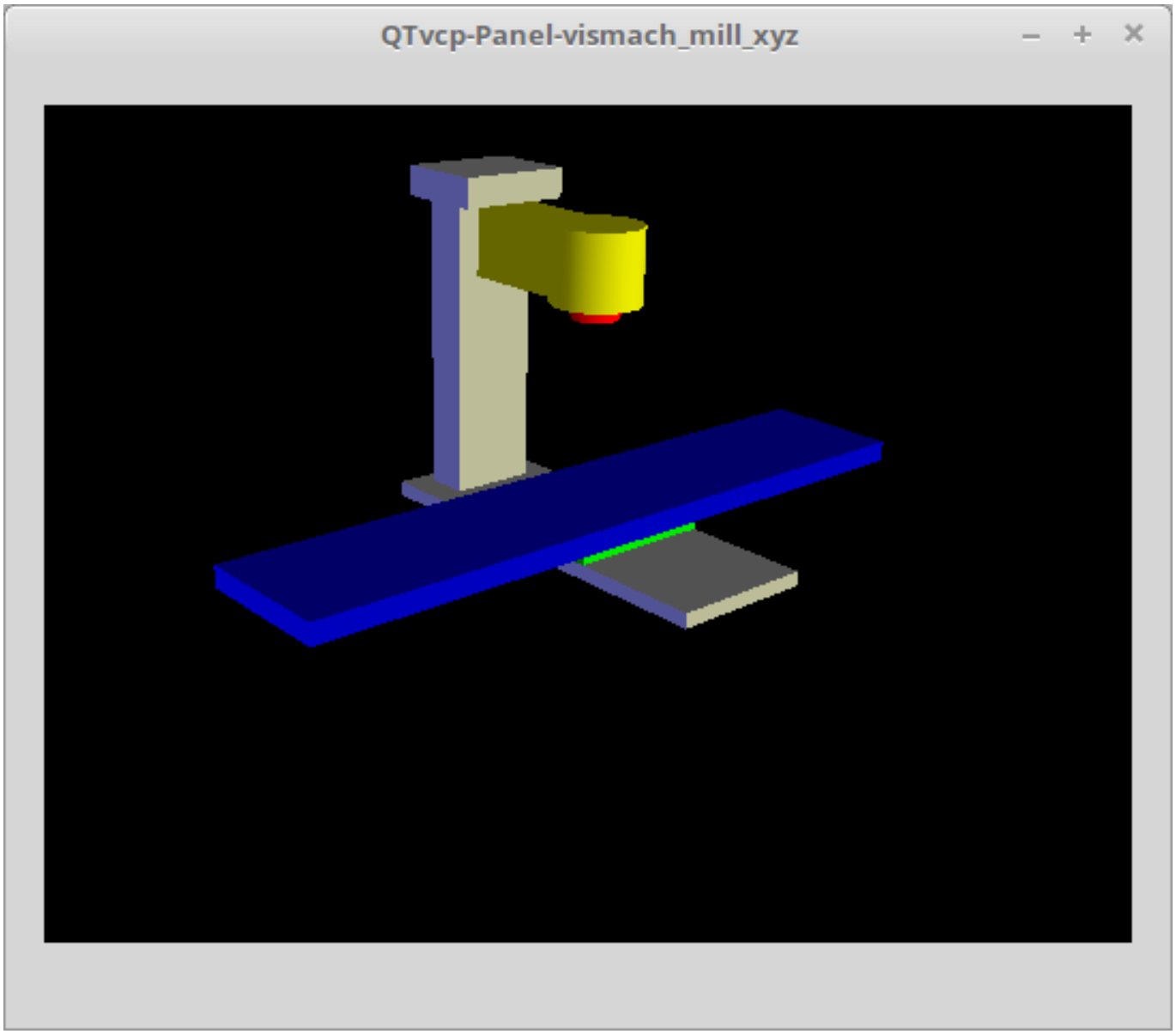


Abbildung 12.64: QtVismach - 3-Axis Mill Builtin Panel

```
loadusr qtvcp test_panel
```

12.5.3.2 Custom Panels

You can of course **make your own panel and load it**.

If you made a UI file named `my_panel.ui` and a HAL file named `my_panel.hal`, you would then load this from a terminal with:

```
halrun -I -f my_panel.hal
```

Example HAL file loading a QtVCP panel

```
# load realtime components
loadrt threads
```

```
loadrt classicladder_rt

# load user space programs
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 this case we load `qtvcp` using **`-Wn`** which waits for the panel to finish loading before continuing to run the next HAL command.
This is to *ensure that the panel created HAL pins are actually done* in case they are used in the rest of the file.

12.5.4 Build A Simple Clean-sheet Custom Screen

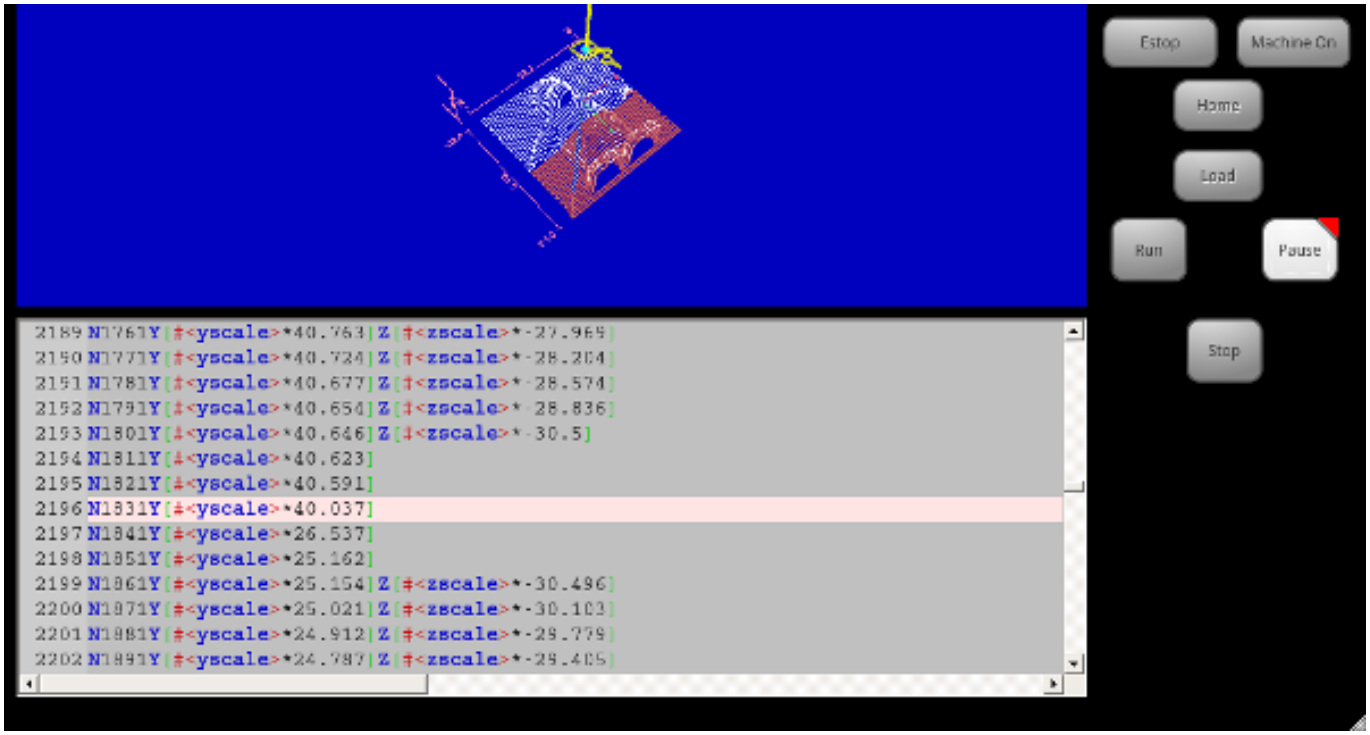


Abbildung 12.65: QtVCP Ugly custom screen

12.5.4.1 Übersicht

To build a panel or screen:

- Use Qt Designer to build a design you like and save it to your configuration folder with a name of your choice, ending with `.ui`
- Modify the configuration INI file to load QtVCP using your new `.ui` file.
- Then connect any required HAL pins in a HAL file.

12.5.4.2 Get Qt Designer To Include LinuxCNC Widgets

Install Qt Designer First you must have the **Qt Designer installed**.
The following commands should add it to your system, or use your package manager to do the same:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

Add qtvcp_plugin.py link to the Qt Designer Search Path Then you must add a link to the `qtvcp_plugin.py` in one of the folders that Qt Designer will search into.

In a *RIP* version of LinuxCNC `qtvcp_plugin.py` will be:

`'~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py'`

For a *Package installed* version it should be:

`'usr/lib/python2.7/qtvcp/plugins/qtvcp_plugin.py' or`
`'usr/lib/python2.7/dist-packages/qtvcp/plugins/qtvcp_plugin.py'`

Make a symbolic link to the above file and move it to one of the places Qt Designer searches in.

Qt Designer searches in these two place for links (pick one):

`'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' or`
`'~/.designer/plugins/python'`

You may need to create the `plugins/python` folder.

Start Qt Designer:

- For a *RIP install*:
  Open a terminal, set the environment for LinuxCNC <1>, then load Qt Designer <2> with :

```
. scripts/rip-environment    (1)
designer -qt=5               (2)
```

- Für eine *Paketinstallation*:
  Öffnen Sie ein Terminal und geben Sie ein:

```
designer -qt=5
```

If all goes right, Qt Designer will launch and you will see the selectable LinuxCNC widgets on the left hand side.

12.5.4.3 Build The Screen `.ui` File

Create MainWindow Widget When Qt Designer is first started there is a *'New Form' dialog* displayed.
Pick *'Main Window'* and press the *'Create'* button.
A *`MainWindow` widget* is displayed.

Warnung
Do not rename this window !
QtVCP requires the name to be `MainWindow`.

Set `MainWindow` Minimum and Maximum Size

- Grab the corner of the window and resize to an appropriate size, say 1000x600.

- Right click on the window and click set *minimum size*.
- Do it again and set *maximum size*.

Our sample widget will now not be resizable.

Add the ScreenOptions Widget Drag and drop the `ScreenOptions` widget anywhere onto the main window.

This widget doesn't add anything visually but sets up some **common options**.

It's recommended to always *add this widget before any other*.

Right click on the main window, not the `ScreenOptions` widget, and set the *layout* as vertical to make the `ScreenOptions` fullsized.

Add Panel Content On the right hand side there is a panel with tabs for a *Property editor* and an *Object inspector*.

On the Object inspector click on the *ScreenOptions*.
Then switch to the Property Editor and, under the *ScreenOptions* heading, toggle **`filedialog_option`**.

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.

Add Action Buttons Add 7 action buttons on to the main window.

If you double click the button, you can add text.
Edit the button labels for *Estop*, *Machine On*, *Home*, *Load*, *Run*, *Pause* and *stop*.

Action buttons *default to no action* so we must change the properties for defined functions. You can edit the properties:

- directly in the *property editor* on the right side of Qt Designer, or
- conveniently, left double clicking on the button to launch a *properties dialog* that allows selecting actions while only displaying relevant data to the action.

Wir werden zunächst den bequemen Weg beschreiben:

- Right click the *Machine On* button and select *Set Actions*.

- When the dialog displays, use the combobox to navigate to `MACHINE CONTROLS - Machine On`.
- In this case there is no option for this action so select *OK*.

Now the button will turn the machine on when pressed.

And now the direct way with Qt Designer's property editor:

- Select the *Machine On* button.
- Go to the Property Editor on the right side of Qt Designer.
- Scroll down until you find the *ActionButton* heading.
- Click the `machine_on` action checkbox you will see in the list of properties and values.

The button will now control machine on/off.

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`.

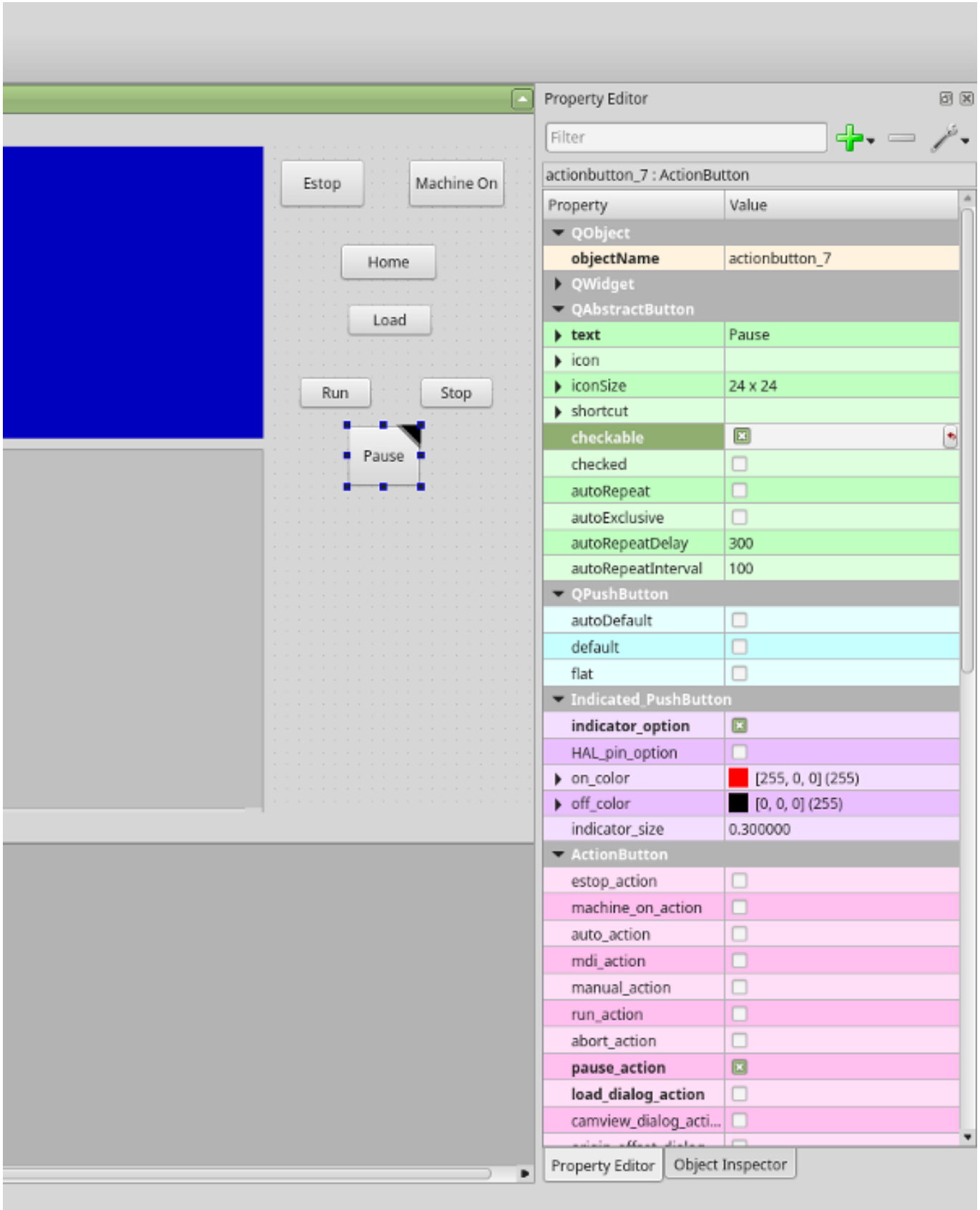


Abbildung 12.66: Qt Designer: Auswahl der Eigenschaften des Pause-Buttons (Schaltfläche)

Save The .ui File Diesen Entwurf müssen wir dann als `tester.ui` im Ordner `sim/qtvcp` speichern.

Wir speichern sie als *tester*, da dies ein Dateiname ist, den qtvcp erkennt und eine eingebaute Handler-Datei verwendet, um sie anzuzeigen.

12.5.4.4 Handler-Datei

A handler file is **required**.

It allows customizations to be written 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.

12.5.4.5 INI Configuration

[DISPLAY] Section 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] Section 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=<filename>`
: Typically `<filename>` would be `+<screen_name>_postgui.hal+`, but can be any legal filename.
You can have *multiple `POSTGUI_HALFILE` lines* in the INI: each will be run one after the other in the order they appear.
These commands are *executed after the screen is built*, guaranteeing the widget HAL pins are available.

`POSTGUI_HALCMD=<command>`
: `<command>` would be *any valid HAL command*.
You can have *multiple `POSTGUI_HALCMD` lines* in the INI: each will be run one after the other in the order they appear.
To guaranty the widget HAL pins are available, these commands are executed:

- *after the screen is built*,
- *after all the POSTGUI_HALFILEs are run*.

In our example there are no HAL pins to connect.

12.5.5 Handler File In Detail

Handler files are used to *create custom controls using Python*.

12.5.5.1 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 allows access to  widgets from the QtVCP files
    # at this point the widgets and hal pins are not instantiated
    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's 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)]
```

12.5.5.2 IMPORT Section

This section is for **importing required library modules** for your screen.

It would be typical to import QtVCP's *keybinding*, *Status* and *Action* libraries.

12.5.5.3 Abschnitt INSTANTIATE BIBRARIES

By instantiating the libraries here we **create global reference**.

You can note this by the commands that don't have `self.` in front of them.

By convention we *capitalize the names of globaly referenced libraries*.

12.5.5.4 HANDLER CLASS Section

The **custom code** is placed *in a class so QtVCP can utilize it*.

Dies ist die Definition der Handler-Klasse.

12.5.5.5 INITIALIZE Section

Like all Python libraries the **+`__init__`+ function** is called when the library is *first instantiated*.

This is where you would set up *defaults*, as well as *reference variables* and *global variables*.

The widget references are not available at this point.

The variables `halcomp`, `widgets` and `paths` give access to QtVCP's HAL component, widgets, and path info respectively.

12.5.5.6 SPECIAL FUNCTIONS Section

There are several *special functions* that QtVCP looks for in the handler file.

If QtVCP finds these it will call them, if not it will silently ignore them.

`initialized__(self):`
This function is *called after the widgets and HAL pins are built*.
You can manipulate the widgets and HAL pins or add more HAL pins here.
Typically there can be

- preferences checked and set,
- styles applied to widgets,
- status of LinuxCNC connected to functions.
- keybindings would be added.

`class_patch__(self):`
Class patching, also known as *monkey patching*, allows to **override function calls in an imported module**.
Class patching must be done *before the module is instantiated*, and it *modifies all instances* made after that.
An example might be patching button calls from the G-code editor to call functions in the handler file instead.

`processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):`
This function is called to facilitate *keyboard jogging* etc.
By using the *`keybinding` library* this can be used to easily add functions bound to keypresses.

`keypress_event__(self,receiver, event):`
This function gives **raw key press events**.
It takes *precedence over* the `processed_key_event`.

`keyrelease_event__(receiver, event):`
This function gives **raw key release events**.
It takes *precedence over* the `processed_key_event`.

`before_loop__(self):`
This function is *called just before the Qt event loop is entered*. At that point, all widgets/libraries/initialization code has completed and the screen is already displayed.

`system_shutdown_request__(self):`
If present, this function **overrides the normal function called for total system shutdown**.
It could be used to do *pre-shutdown housekeeping*.
The *system will not shutdown if using this function*, you will have to do that yourself.
QtVCP/LinuxCNC will shutdown without a prompt after this function returns.

`closing_cleanup__(self):`
This function is *called just before the screen closes*. It can be used to do cleanup before closing.

12.5.5.7 STATUS CALLBACKS Section

By convention this is where you would put functions that are **callbacks from STATUS definitions**.

12.5.5.8 CALLBACKS FROM FORM Section

By convention this is where you would put functions that are **callbacks from the widgets connected to the MainWindow** in the Qt Designer editor.

12.5.5.9 GENERAL FUNCTIONS Section

By convention this is where you put your **general functions**.

12.5.5.10 KEY BINDING Section

If you are *using the `keybinding` library* this is where you place your **custom key call routines**.

The function signature is:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

KEY being the code (from the keybindings library) for the desired key.

12.5.5.11 CLOSING EVENT Section

Putting the **`closeEvent` function here will catch closing events**.

This *replaces any predefined `closeEvent`* function from QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

Anmerkung
It is usually better to use the special `closing_cleanup__` function.

12.5.6 Connecting Widgets to Python Code

It is possible to connect widgets to Python code using **signals and slots**.

In this way you can:

- *Give new functions to LinuxCNC widgets*, or
- *Utilize standard Qt widgets to control LinuxCNC*.

12.5.6.1 Übersicht

In the Qt Designer editor:

- You *create user function slots*
- You *connect the slots to widgets using signals*.

In der Handler-Datei:

- You *create the slot's functions* defined in Qt Designer.

12.5.6.2 Using Qt Designer to add Slots

When you have loaded your screen into Qt Designer, add a plain `PushButton` to the screen.
You could change the name of the button to something interesting like *test_button*.

There are two ways to edit connections - This is the graphical way.

- There is a button in the top tool bar of Qt Designer for editing signals. After pushing it, if you click-and-hold on the button it will show an arrow (looks like a ground signal from electrical schematic).
- Slide this arrow to a part of the main window that does not have widgets on it.
- A *Configure Connections* dialog will pop up.
  - The list on the left are the available signals from the widget.
  - The list on the right are the available slots on the main window and you can add to it.
- Pick the signal `clicked()` - this makes the slots side available.
- Click *Edit* on the slots list.
- A *Slots/Signals of MainWindow* dialog will pop up.
- On the slots list at the top there is a + icon - click it.
- You can now edit a new slot name.
- Erase the default name `slot()` and change it to `test_button()`
- Drücken Sie die Taste *OK*.
- You'll be back to the *Configure Connections* dialog.
- Now you can select your new slot in the slot list.
- Then press *OK* and save the file.

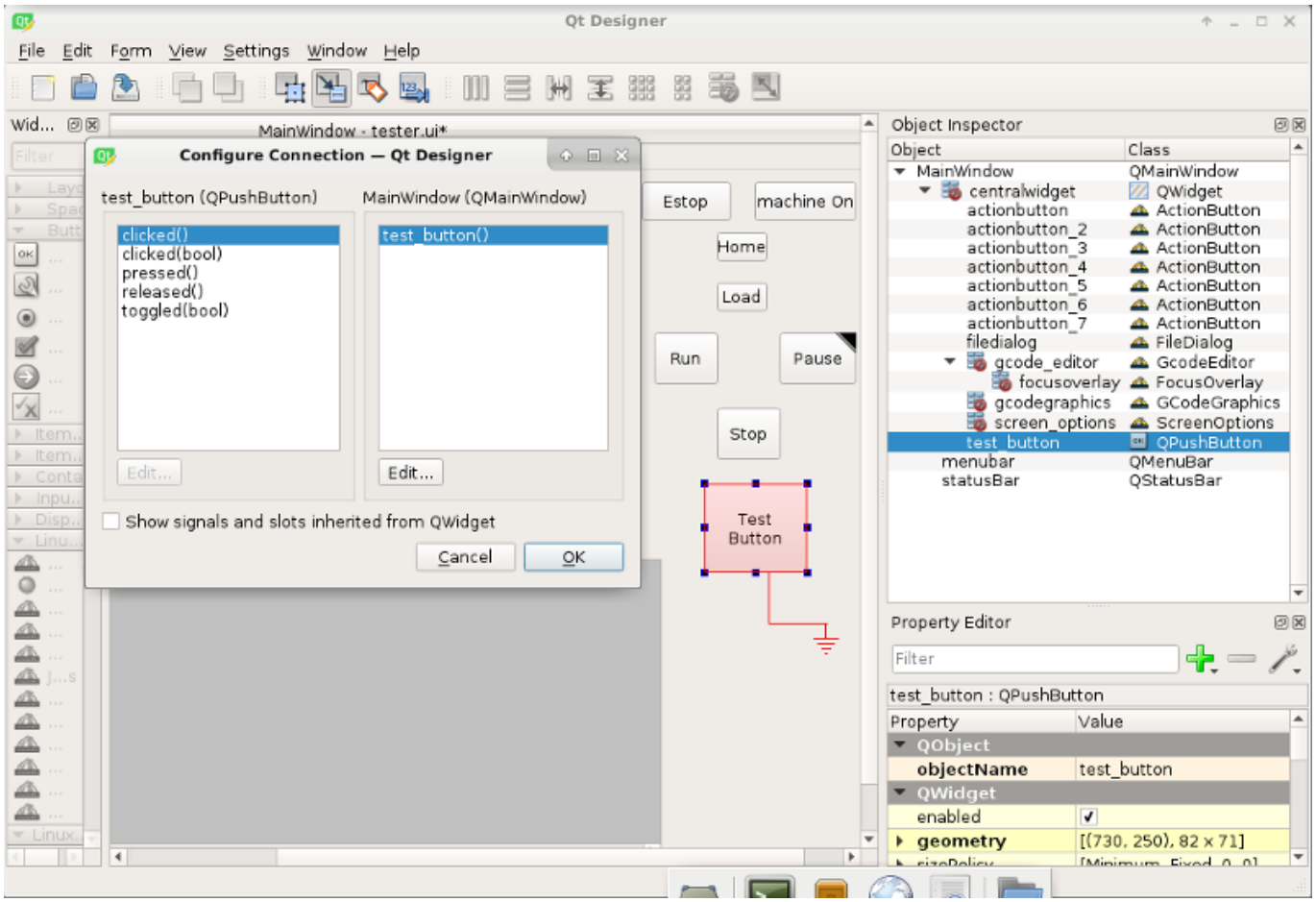


Abbildung 12.67: Qt Designer Signal/Slot Selection

12.5.6.3 Python Handler Changes

Nun müssen Sie **die Funktion in die Handler-Datei einfügen**.

The function signature is **`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.

Now when you load your screen and press the button it should print the name of the button in the terminal.

12.5.7 More Informations

QtVCP Builtin Virtual Control Panels

QtVCP Widgets

QtVCP Libraries

Qt Vismach

QtVCP Handler File Code Snippets

QtVCP Development

QtVCP Custom Qt Designer Widgets

12.6 QtVCP Virtuelle Kontrollpanels

QtVCP can be used to **create control panels** that interface with *HAL*.

12.6.1 Builtin Virtual Control Panels

Es sind mehrere **integrierte HAL-Panels** verfügbar.

In a terminal type `qtvcp list` to see a list.

12.6.1.1 `copy`

Used for **copying QtVCP's builtin Screens/VCP Panels/QtVismach code to a folder** so one can *customize* it.

In a terminal run:

```
qtvcp copy
```

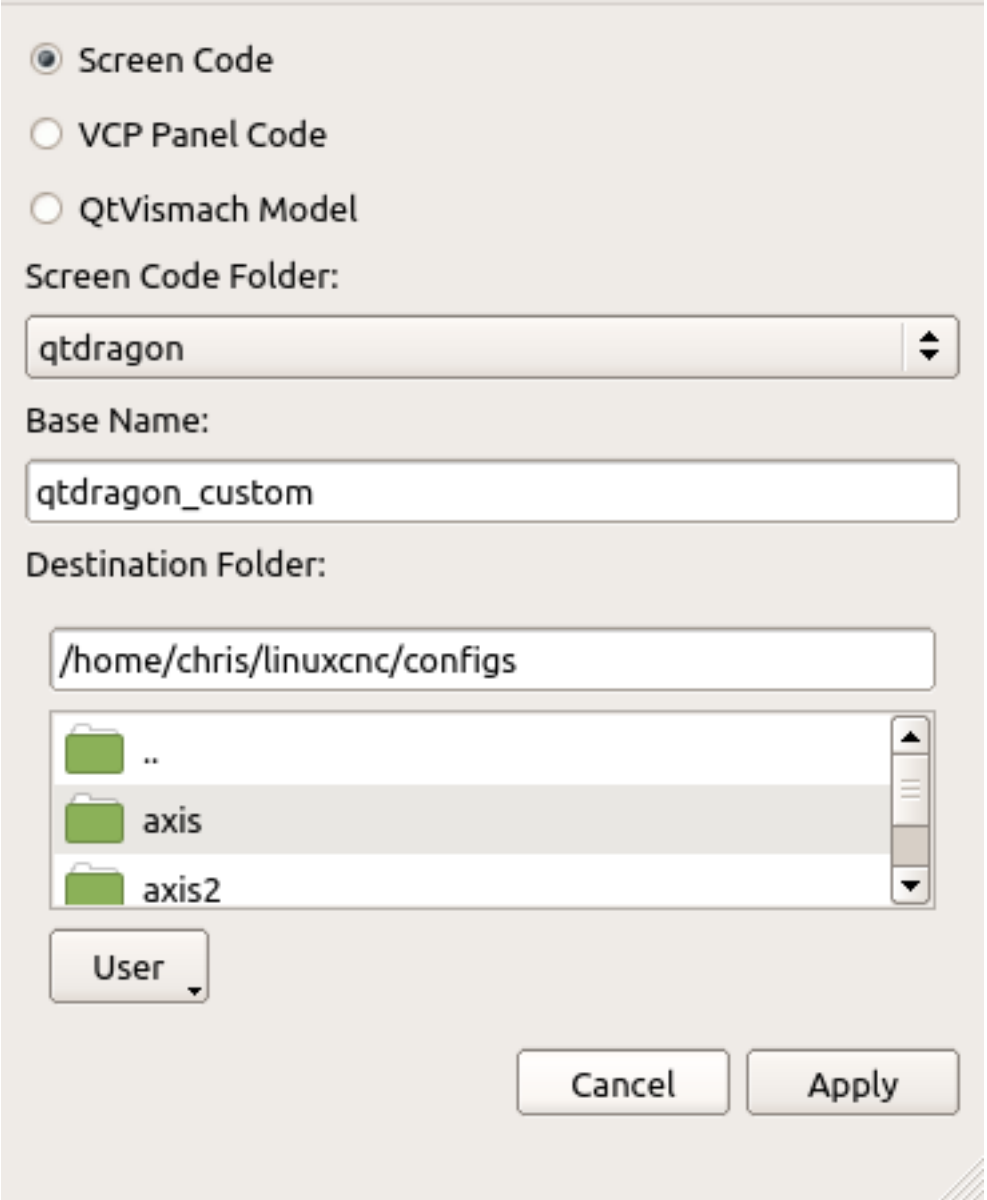


Abbildung 12.68: QtVCP `copy` Dialog - Screen, VCP Panel or QtVismach Code Copy Panel

12.6.1.2 `test_dial`

- This panel has a **dial that adjusts S32 and Float HAL output pins**.
- The dial's range can be adjusted from a drop down menu.
- The output can be scaled with the `spinbox`.
- A `combobox` can be used to automatically select and connect to a signal.

```
loadusr qtvcp test_dial
```

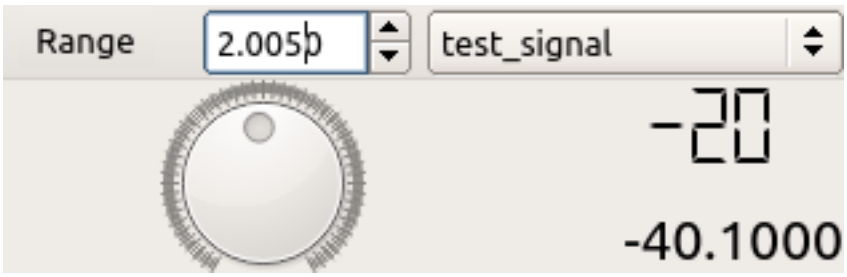


Abbildung 12.69: QtVCP `test_dial` Panel - Test Dial VCP

12.6.1.3 test_button

- This panel has a **button that will set a HAL pin**.
- The button can be selected as a *momentary* or a *toggle* button.
- The button's *indicator color* can be adjusted from a drop down menu.
- You can add more buttons from the drop down menu.
- You can load a Halmeter from the drop down menu.
- You can load a test LED from the drop down menu.
- The button can be detached from the main windows.

Here is how to load `test_button` from a HAL script:

```
loadusr qtvcp test_button
loadusr qtvcp -o 4 test_button
```

The `-o` switch sets how many buttons the panel starts with.
If loading directly from a terminal omit the `loadusr`.

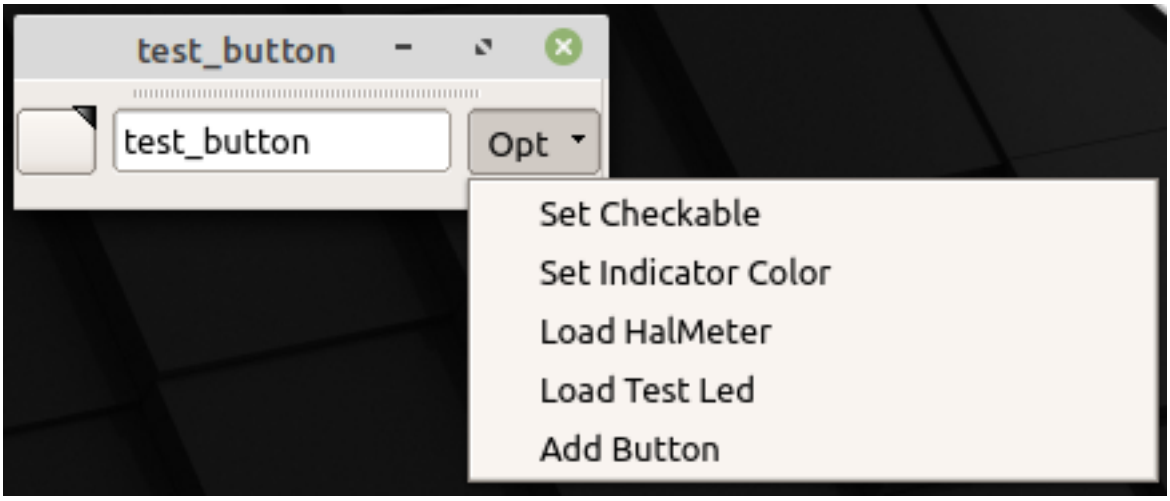


Abbildung 12.70: QtVCP `test_button` - Test Button VCP

12.6.1.4 test_led

- This panel has an **LED that can selected to watch HAL bit pins/signals**.
- The LED's color can be adjusted from a drop down menu.
- The text box and state can be output as speech if sound is selected.
- A `combobox` can be used to automatically select and connect to a pin/signal.
- You can add more LEDs from the drop down menu.
- The LED can be detached from the main windows.

Here is how to load `test_led` from a HAL script:

```
loadusr qtvcp test_led
loadusr qtvcp -o 4 test_led
```

The `-o` switch sets how many LEDs the panel starts with.
If loading directly from a terminal omit the *loadusr*.

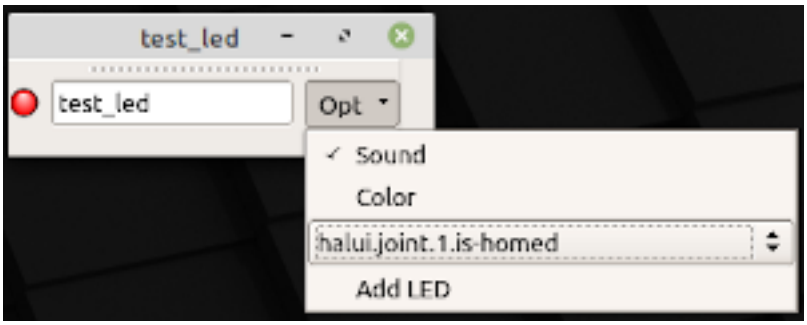


Abbildung 12.71: QtVCP `test_dial` Panel - Test LED VCP

12.6.1.5 test_panel

Collection of useful widgets for testing HAL component, including speech of LED state.

```
loadusr qtvcp test_panel
```

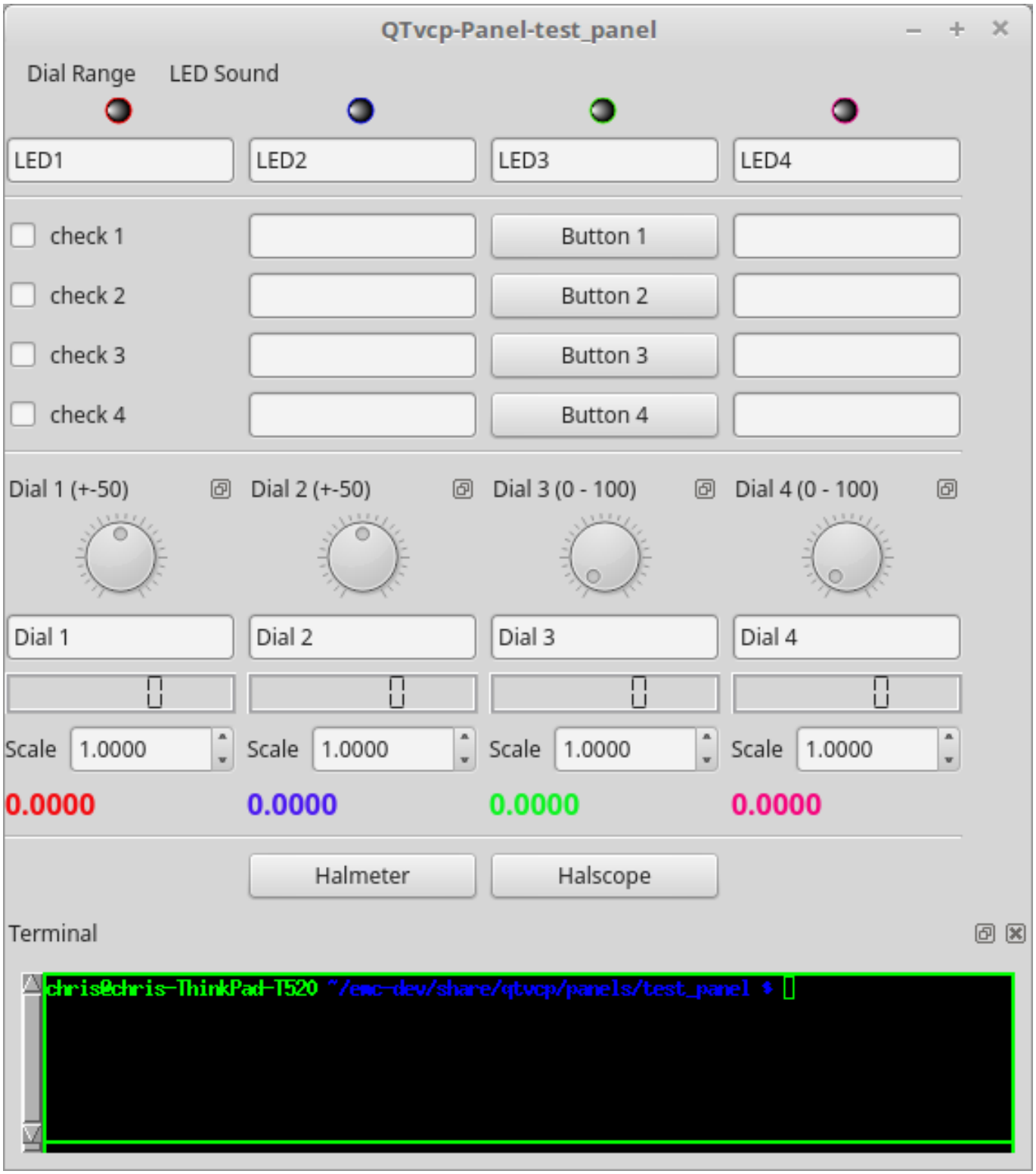


Abbildung 12.72: QtVCP `test_panel` - HAL Component Testing Panel

12.6.1.6 `cam_align`

A **camera display widget for rotational alignment**.

Abbildung 12.73: QtVCP `cam_align` Panel - Camera Based Alignment Panel

12.6.1.7 `sim_panel`

Small control panel to **simulate MPG jogging controls etc**.
For simulated configurations

```
loadusr qtvcp sim_panel
```

Abbildung 12.74: QtVCP `sim_panel` - Simulated Controls Panel For Screen Testing.

12.6.1.8 `tool_dialog`

Manual tool change dialog that gives tool description.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Options:

- `-o notify_on` - *use desktop notify* dialogs instead of QtVCP native ones.
- `-o audio_on` - *play sound on tool change*
- `-o speak_on` - *speak announcement of tool change*

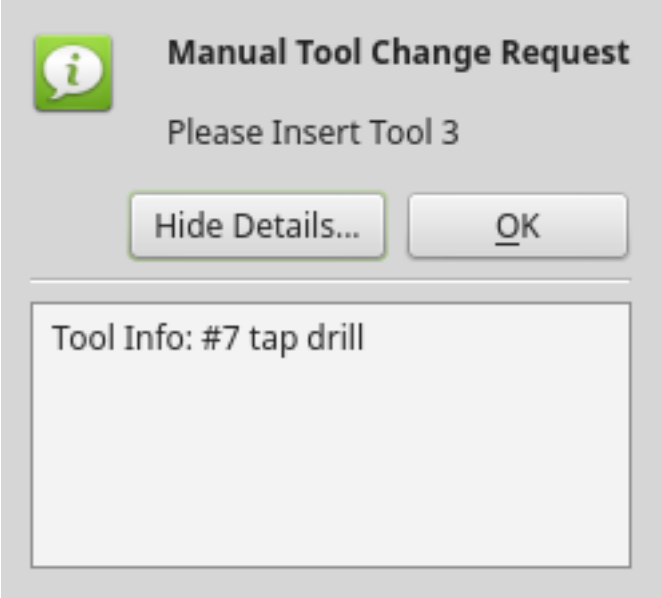


Abbildung 12.75: QtVCP `tool_dialog` - Manual Tool Change Dialog

12.6.2 `vismach` 3D Simulation Panels

These panels are prebuilt simulation of common machine types.
These are also embed-able in other screens such as AXIS or GMOCCAPY.

12.6.2.1 QtVCP `vismach_mill_xyz`

3D OpenGL view of a *3-Axis milling machine*.

```
loadusr qtvcp vismach_mill_xyz
```

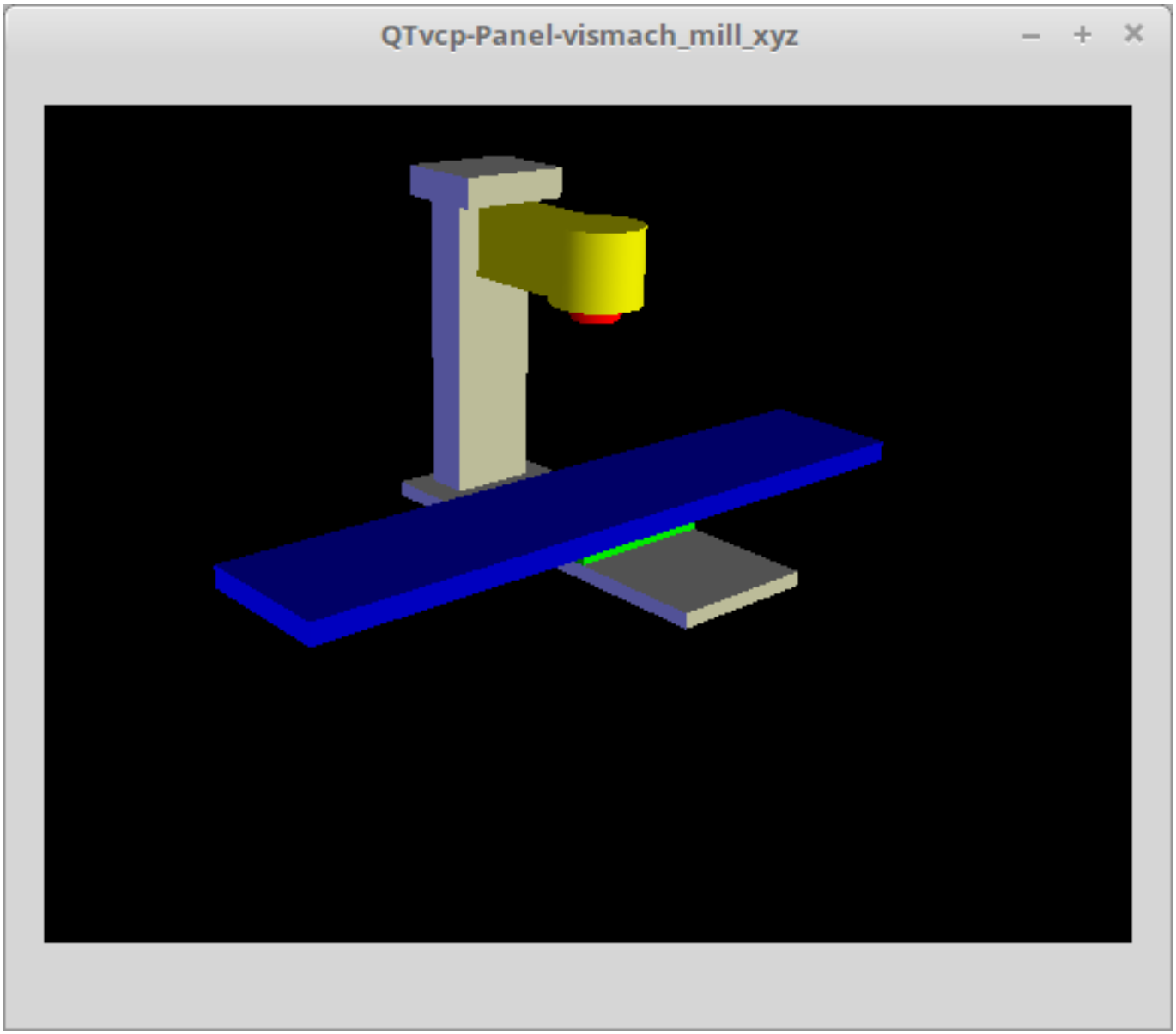


Abbildung 12.76: QtVCP `vismach_mill_xyz` - 3-Axis Mill 3D View Panel

12.6.2.2 QtVCP `vismach_scara`

3D OpenGL view of a *SCARA based milling machine*.

```
loadusr qtvcp vismach_scara
```

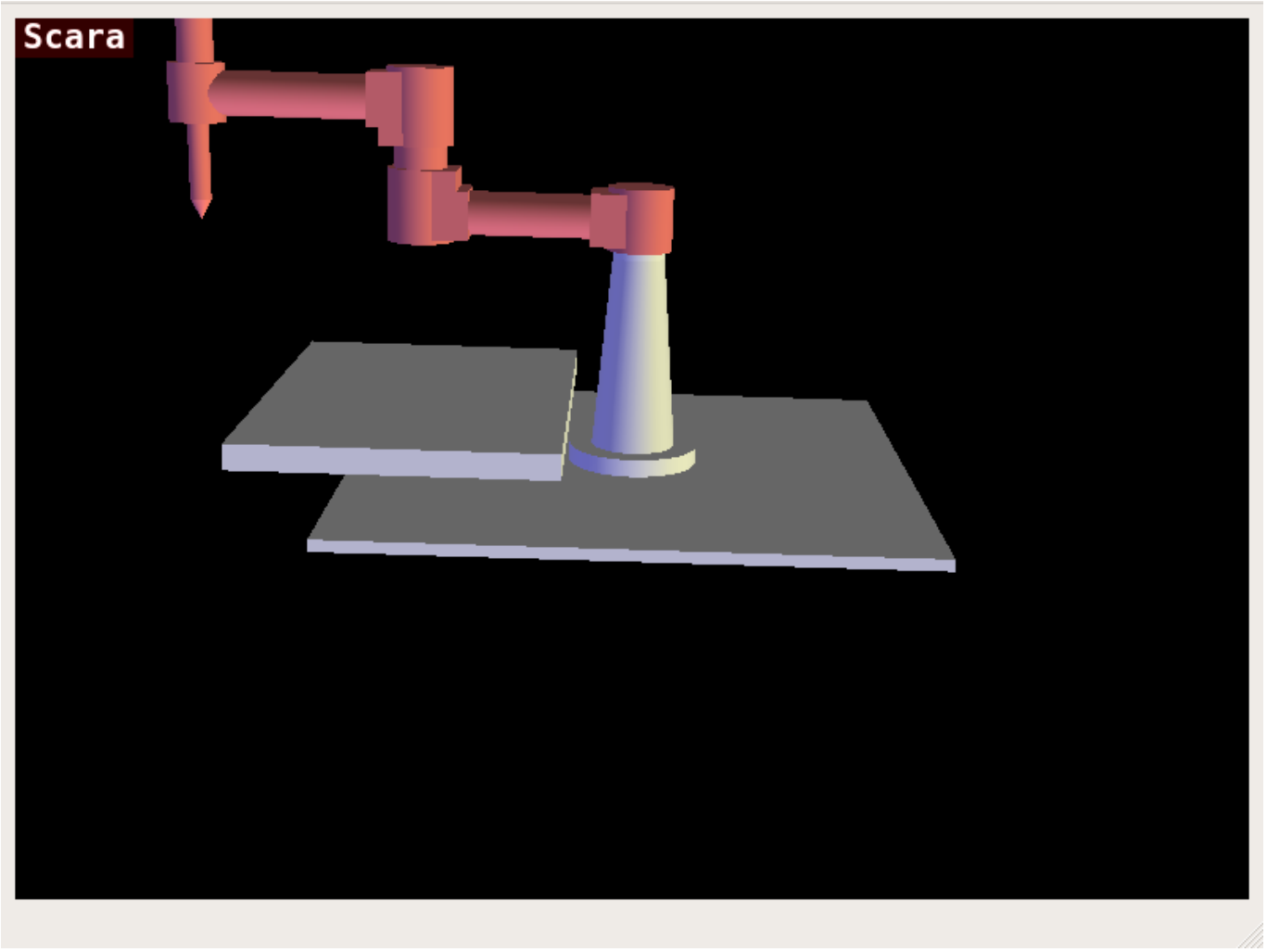


Abbildung 12.77: QtVCP `vismach_scara` - SCARA Mill 3D View Panel

12.6.2.3 QtVCP `vismach_millturn`

3D OpenGL view of a *3-Axis milling machine with an A axis/spindle*.

```
loadusr qtvcp vismach_millturn
```

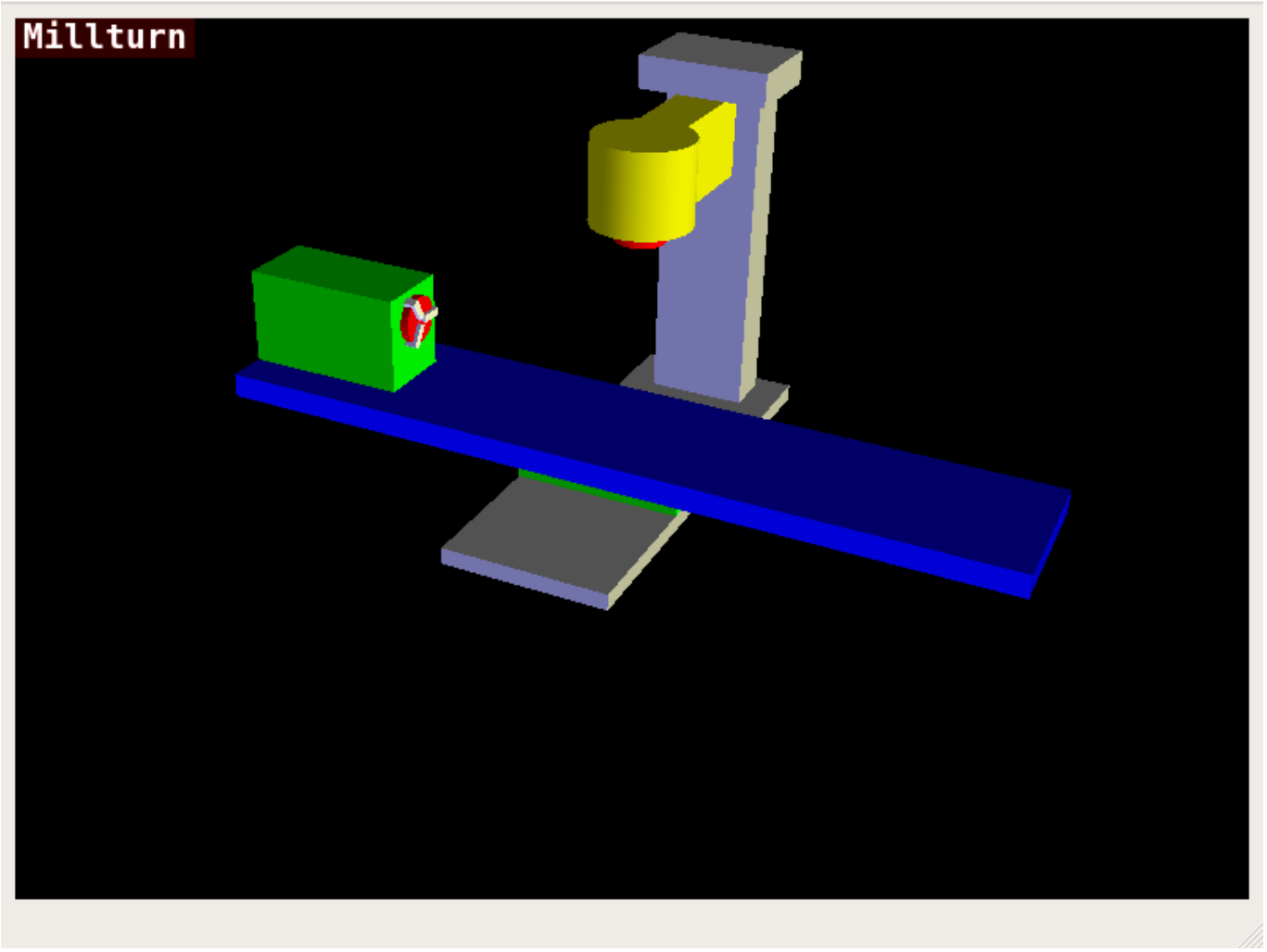


Abbildung 12.78: QtVCP `vismach_millturn` - 4 Axis MillTurn 3D View Panel

12.6.2.4 QtVCP `vismach_mill_5axis_gantry`

3D OpenGL view of a *5-Axis gantry type milling machine*.

```
loadusr qtvcp vismach_mill_5axis_gantry
```

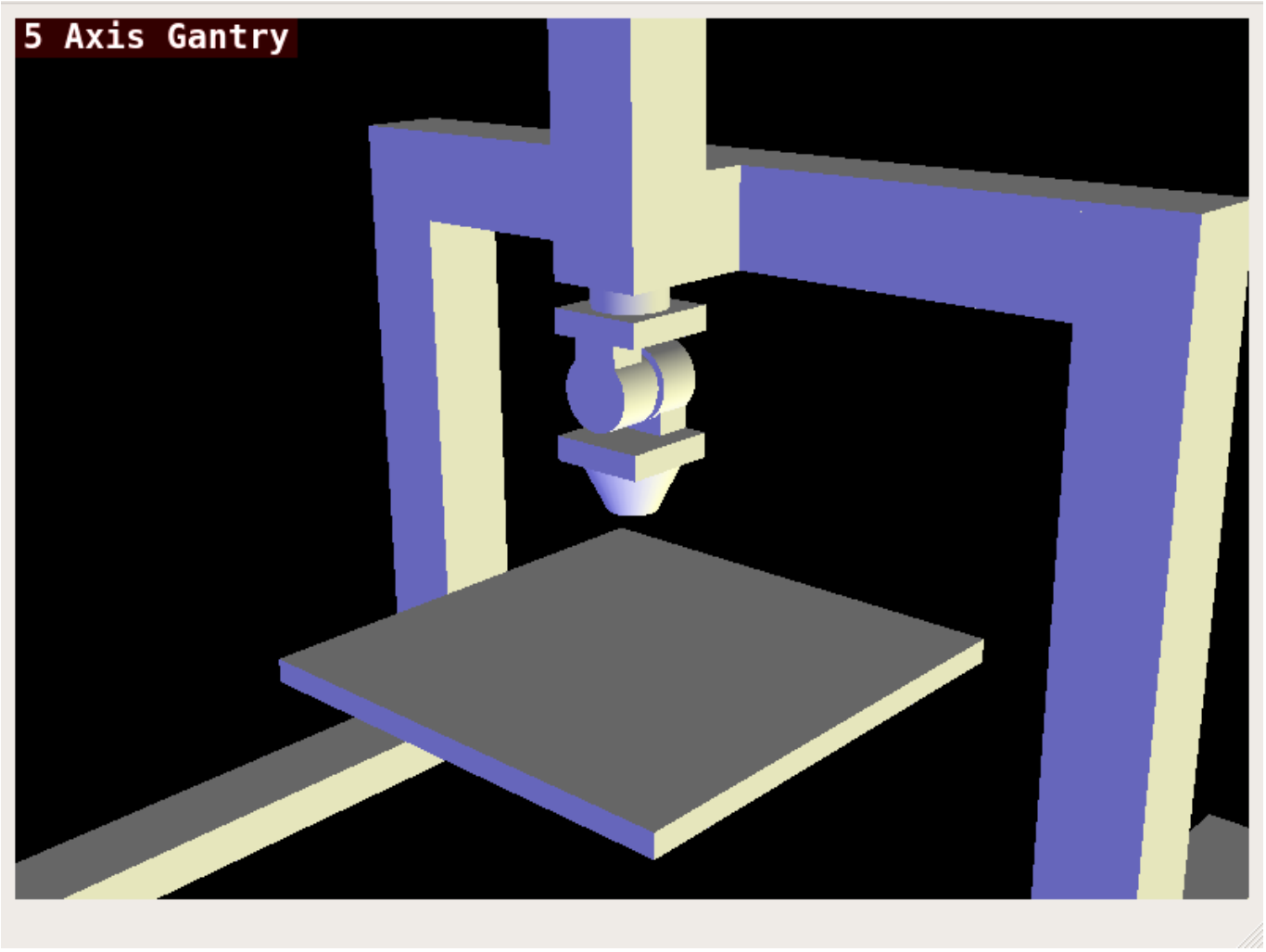


Abbildung 12.79: QtVCP `vismach_mill_5axis_gantry` - 5-AxIs Gantry Mill 3D View Panel

12.6.2.5 QtVCP `vismach_fanuc_200f`

3D openGL view of a *6 joint robotic arm*.

```
loadusr qtvcp vismach_fanuc_200f
```

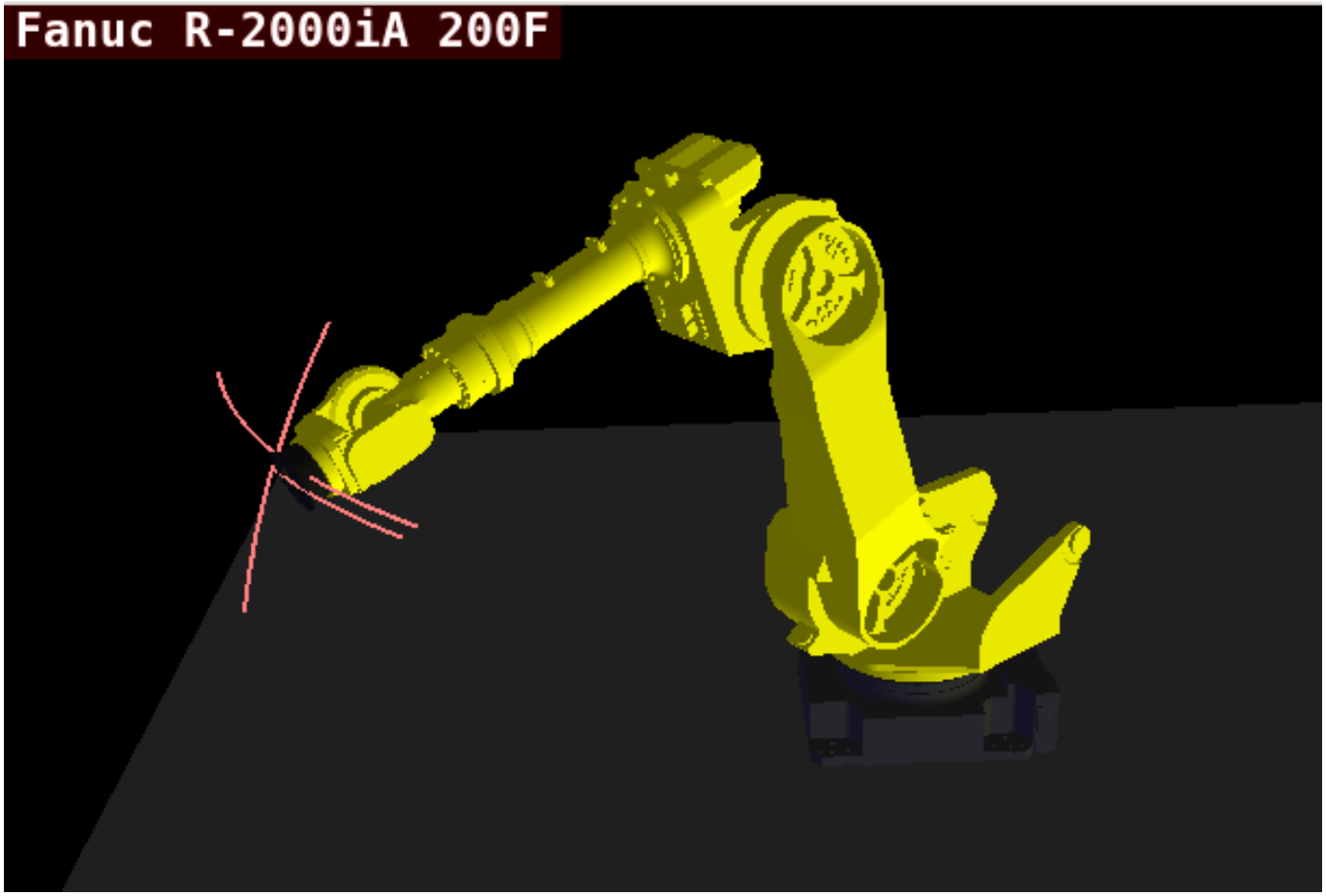


Abbildung 12.80: QtVCP `vismach_fanuc_200f` - 6 Joint Robotic Arm

12.6.3 Custom Virtual Control Panels

You can of course **make your own panel and load it**.

If you made a UI file named `my_panel.ui` and a HAL file named `my_panel.hal`, you would then load this from a terminal with:

```
halrun -I -f my_panel.hal
```

Example HAL file loading a QtVCP panel

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load user space programs
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), (1) In this case we load `qtvcp` using **`-Wn`** which waits for the panel to finish loading before continuing to run the next HAL command.
This is to *ensure that the panel created HAL pins are actually done* in case they are used in the rest of the file.

12.7 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**.

You are free to mix them in any way on your panel.

Anmerkung
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.

12.7.1 Nur HAL-Widgets

Diese Widgets haben normalerweise *HAL-Pins* und **reagieren nicht auf die Maschinensteuerung**.

12.7.1.1 `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 virtual control panels
- mplayer-Videoplayer

12.7.1.2 `Slider` - HAL-Pin-Wert-Anpassungs-Widget

Allows one to **adjust a HAL pin value using a sliding pointer**.

12.7.1.3 `LED` - Indicator Widget

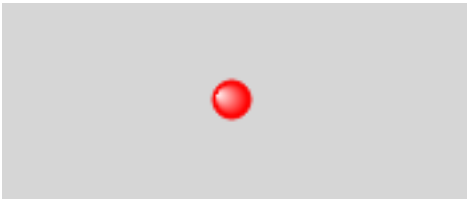


Abbildung 12.81: QtVCP `LED`: LED-Anzeige-Widget

A **LED like indicator** that optionally follows a HAL pin's logic.

`halpin_option`
: Wählt aus, ob die LED einem Eingangs-HAL-Pin oder einem Programmzustand folgt.

`diameter`
: Diameter of the LED

`color`
: Color of the LED when on.

`off_color`
: Color of the LED when off.

`alignment`
: Qt-Hinweis zur Ausrichtung.

`state`
: Current state of LED

`flashing`
: Schaltet die Blinkoption ein und aus.

`flashRate`
: Sets the flash rate.

The `LED` properties can be defined in a *stylesheet* with the following code added to the `.qss` file, `name_of_led` being the widget name defined in Qt Designer's editor:

```
LED #name_0f_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

12.7.1.4 `CheckBox` Widget

This widget allows the user to **check a box to set a HAL pin true or false**.

It is based on PyQt's *QCheckButton*.

12.7.1.5 `RadioButton` Widget

This widget allows a user to **set HAL pins true or false**. Only one `RadioButton` widget of a group can be true at a time.

It is based on PyQt's *QRadioButton*.

12.7.1.6 `Gauge` - Rundes Messuhr-Widget

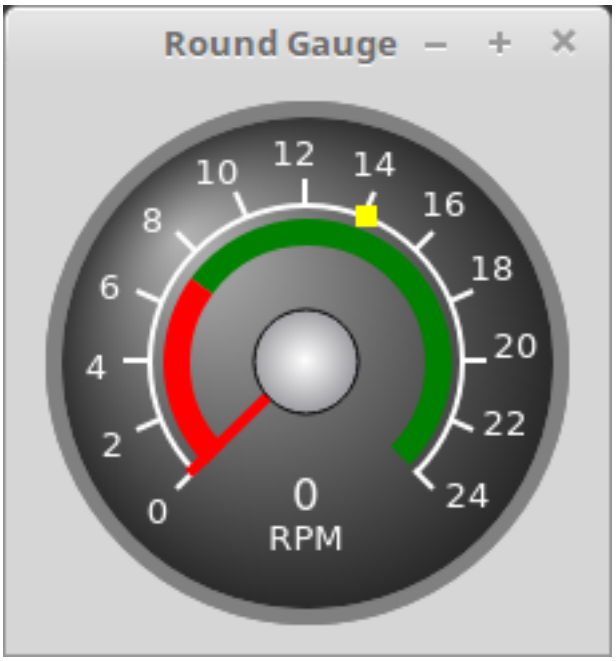


Abbildung 12.82: QtVCP `Gauge`: Round Dial Gauge Widget

Round Gauge kann in einem LinuxCNC GUI verwendet werden, um **einen Eingabeparameter** auf dem Zifferblatt anzuzeigen.

Customizable Parameters 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:

- One is for setting the `value` input
- The other is for setting the `setpoint`.

Wenn diese Option nicht gesetzt ist, müssen `value` und `setpoint` programmatisch, dh in der Handler-Datei, verbunden werden. **`max_reading`**:: This value determines the *highest number displayed* on the gauge face. **`max_value`**:: Dies ist der *maximal zu erwartende Wert des Wert-eingangssignals*.
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**:: This is the color of the *outer ring of the gauge*. **threshold**:: The threshold is the *transition point between the zones*.
It should be set to a value between 0 and the maximum value.
The maximum allowed value is set to the gauge's `max_value` and minimum value is 0. **gauge_label**::
This is the *text below the value readout*, near the bottom of the gauge.
The function of the gauge is then easily visible.

Non Customizable Parameters 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
This is a value that determines the location of a small *marker on the gauge face*. It must be set to a value between 0 and the maximum value.

12.7.1.7 `HALPad` - HAL Buttons Joypad

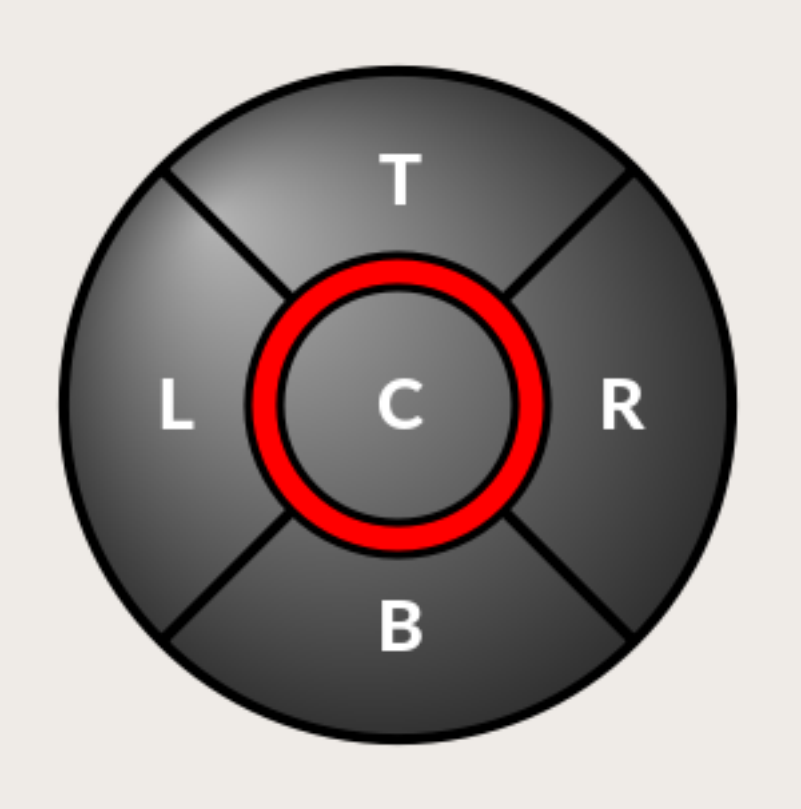


Abbildung 12.83: QtVCP `HALPad`: HAL Buttons Joypad

Dieses Widget sieht aus und funktioniert wie ein **5-Tasten-D-Pad**, mit einem LED-Ring.

Jede Taste hat einen wählbaren Typ (Bit, S32 oder Float) als HAL-Pin.

Der LED-Mittelring hat wählbare Farben für Aus und Ein und wird über einen Bit-HAL-Pin gesteuert.

HALPad ENUMS There are *enumerated constants* used:

- To reference **indicator positions**:
  - NONE
  - LEFT
  - RIGHT
  - CENTER
  - TOP
  - BOTTOM
  - LEFTRIGHT
  - TOPBOTTOM
- Für **HAL-Pins Typ**:
  - NONE
  - BIT
  - S32
  - FLOAT

You use the widget name in Qt Designer plus the reference constant:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad Properties

pin_name
Optionaler Name, der für den *HAL pins basename* verwendet wird.
Bleibt er leer, wird der Name des Qt Designer Widgets verwendet.

pin_type
Select the *HAL output pin type*. This property is only used at startup. Selection can be set in Qt Designer:

- NONE
- BIT
- S32
- FLOAT

left_image_path , right_image_path , center_image_path , top_image_path , bottom_image_path

File or resource path to an image to display in the described button location.
If the reset button is pressed in the Qt Designer editor property, the image will not be displayed. (allowing optional text)

left_text , right_text , center_text , top_text , bottom_text
A text string to be displayed in the described button location.
If left blank an image can be designated to be displayed.

`true_color` , `false_color`
: Color selection for the center LED ring to be displayed when the `<BASENAME>.light.center` HAL pin is `True` or `False`.

`text_color`
: Color selection for the button text.

`text_font`
: Font selection for the button text.

HALPad Styles The above properties could be set in *styles sheets*.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

12.7.1.8 `PushButton` - HAL Pin Toggle Widget

Mit diesem Widget kann der Benutzer einen **HAL-Pin per Tastendruck auf ”true” oder ”false” setzen**.

As an option it can be a *toggle button*.

For a *LED Indicator Option*, see Abschnitt 12.7.5.1[IndicatedPushButton] below for more info.

Es gibt auch andere Optionen.

It is based on PyQt’s *QPushButton*.

12.7.1.9 `focusOverlay` - Focus Overlay Widget

Dieses Widget legt ein **farbiges Overlay über den Bildschirm**, normalerweise während ein Dialog angezeigt wird.

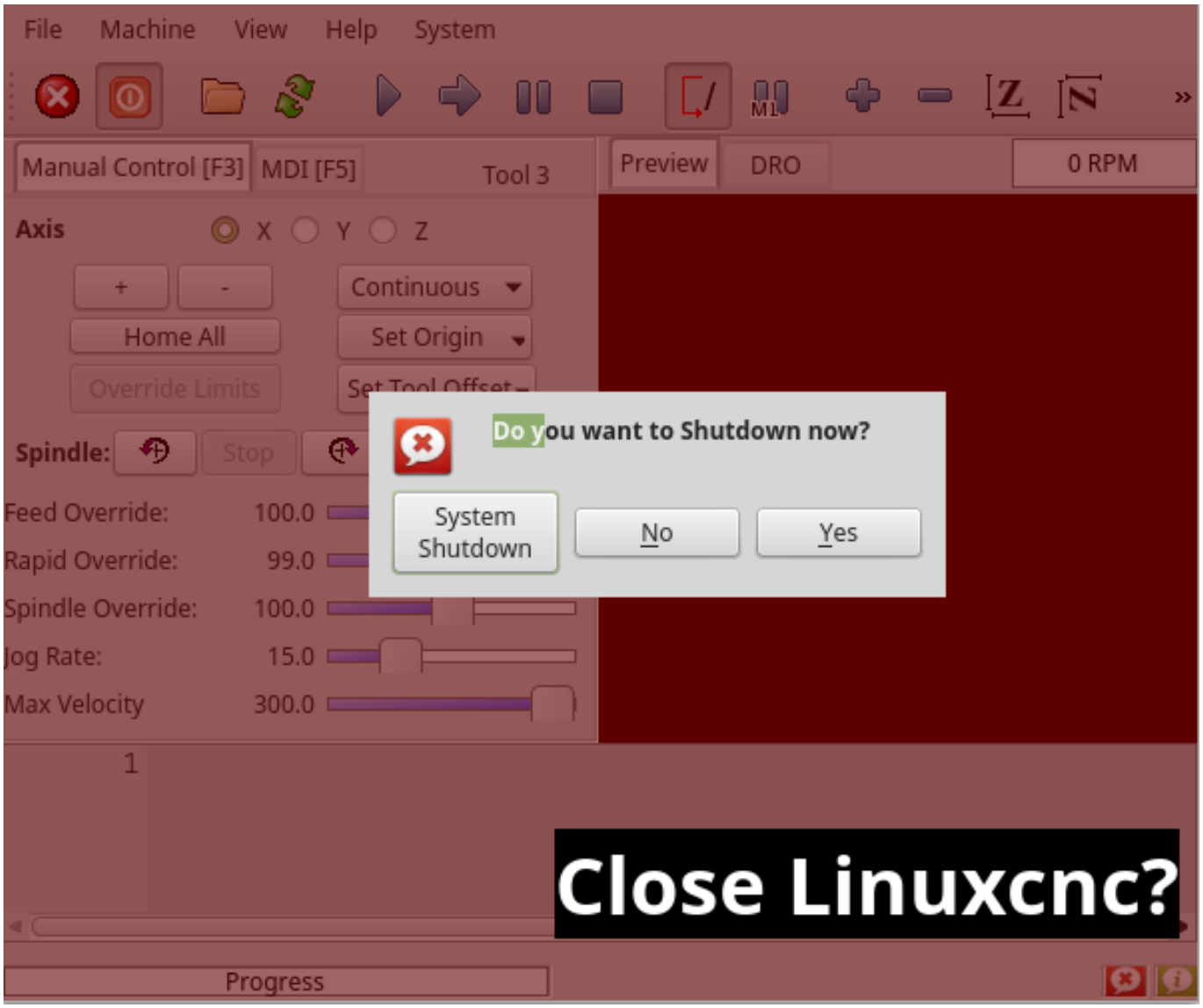


Abbildung 12.84: Beispiel für ein Fokus-Overlay zur Bestätigung der Abschlussaufforderung

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*.

12.7.1.10 `gridLayout` - Grid Layout Widget

This widget **controls if the widgets inside it are enabled or disabled**.

Disabled widgets typically have a different color and do not respond to actions.

It is based on PyQt’s `QGridLayout`.

12.7.1.11 `hal_label` - HAL Label Widget

This widget **displays values sent to it**.

Werte können gesendet werden von:

- *HAL-Pins*
  Der Eingangsstift kann als Bit, S32, Float oder kein Stift ausgewählt werden
- *Programmatically*
- *A `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` for integers
- `%0.4f` für Floats.

A rich text example might be:

```
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 float or a bool signal.

If the property `pin_name` is not set the widget name will be used.

Es gibt Funktionsaufrufe zur Anzeige von Werten:

`[HALLabelName].setDisplay(some_value)`
: Can be used to set the display if no HAL pin is selected.

`[HALLabelName].setProperty(textTemplate,”%d”)`
: Sets the template of the display.

It is based on PyQt’s *QLabel*.

12.7.1.12 `LCDNumber` - Widget zum Auslesen der LCD-Stilnummer

Dieses Widget *zeigt HAL-Float/S32/Bit-Werte in einer LCD-ähnlichen Form an*.

Es kann Zahlen im Dezimal-, Hexadezimal-, Binär- und Oktalformat anzeigen, indem es die Eigenschaft **`Modus`** setzt.

When using floats you can set a formatting string.

You must set the **`digitCount`** property to an appropriate setting to display the largest number.

Eigenschaften

pin_name
Option string to be used as the HAL pin name.
If set to an empty string the widget name will be used.

bit_pin_type
Selects the input pin as type BIT.

s32_pin_type
Selects the input pin as type S32.

float_pin_type
Select the input pin as type `FLOAT`.

floatTemplate
A string that will be used as a Python3 format template to tailor the LCD display.
Only used when a `FLOAT` pin is selected.
eg `{:.2f}` will display a float rounded to 2 numbers after the decimal.
A blank setting will allow the decimal to move as required.

It is based on PyQt's *QLCDNumber*.

12.7.1.13 `DoubleScale` - Spin Button Entry Widget

This widget is a **spin button entry** widget used for *setting a s32 and float HAL pin*.

It has an internal *scale factor*, set to a default of 1, that can be set programmatically or using a QtSignal.

The `setInput` slot can be connected to a integer, or float signal.

`[HALLabelName].setInput(some_value)`
This is a function call to change the internal scaling factor.

The HAL pins will be set to the value of the *internal scale times the widget displayed value*.

12.7.1.14 `GeneralHALInput` - General Signals/Slots Input Connection Widget

This widget is used to **connect an arbitrary Qt widget to HAL using signals/slots**.

It is used *for widgets that should **respond** to HAL pin changes*.

12.7.1.15 `GeneralHALOutput` - General Signals/Slots Output Connection Widget

This widget is used to **connect an arbitrary Qt widget to HAL using signals/slots**.

It is used *for widgets that should **control** HAL pins*.

12.7.1.16 `WidgetSwitcher` - Multi-widget Layout View Switcher Widget

This is used to switch the view of a multi-widget layout to show just one widget, ie to **flip between a large view of a widget and a smaller multi widget view**.

It is *different from a stacked widget* as it can pull a widget from anywhere in the screen and place it in it's page with a different layout than it originally had.

The *original widget must be in a layout* for switcher to put it back.

In Qt Designer you will:

- Add the `WidgetSwitcher` widget on screen.
- Right click the `WidgetSwitcher` and add a page.
- Populate it with the widgets/layouts you wish to see in a default form.
- Add as many pages as there are views to switch to.
- On each page, add a layout widget.
  After adding the layout you must right click the widget switcher again and set the layout option.
- Click on the `WidgetSwitcher` widget and then scroll to the bottom of the property editor.
- Look for the dynamic property `widget_list` and double click to the right of it.
- A dialog pops up allowing you to add the names of the widgets to move to the pages you added to the `WidgetSwitcher`.

There are *function calls* to display specific widgets.
By calling one of these functions, you control what widget is currently displayed:

`[_WidgetSwitcherName_].show_id_widget(_number_)`, `[_WidgetSwitcherName_].show_named_widget(`

This shows the `page 0` layout, and puts all other widgets back to where they were as initially built in Qt Designer.

`[_WidgetSwitcherName_].show_next()`
Show next widget.

It is based on the *QStack* widget.

12.7.2 Machine Controller Widgets

These widgets **interact with the Machine Controller state**.

12.7.2.1 `ActionButton` - Machine Controller Action Control Widget

These buttons are used for **control actions on the machine controller**.

They are built on top of `IndicatedPushButton` so can have LEDs overlaid.

Anmerkung
If you left double click on this widget you can launch a dialog to set any of these actions.
The dialogs will help to set the right related data to the selected action.
You can also change these properties directly in the property editor.

Actions You can select one of these:

`Estop`, `Machine On`, `Auto`, `mdi`, `manual`, `run`, `run_from_line status`
Gets line number from `STATUS` message `gcode-line-selected`.

`run_from_line slot`
Gets line number from Qt Designer int/str slot `setRunFromLine`.

`abort`, `pause`, `load dialog`
Requires a dialog widget present.

Camview dialog
Requires `camview` dialog widget present.

origin offset dialog
Requires origin offset dialog widget present.

macro dialog
Requires macro dialog widget present.

Launch Halmeter , Launch Status , Launch Halshow , Home
Set the joint number to -1 for `all-home`.

Unhome
Set the joint number to -1 for `all-unhome`.

Home Selected
Homes the joint/axis selected by `STATUS`.

Unhome Selected
Unhomes the joint/axis selected by `STATUS`.

zero axis , zero G5X
Zeros the current user coordinate system offsets.

zero G92
Zeros the optional `G92` offsets.

zero Z rotational
Zeros the rotation offset.

jog joint positive
Set the joint number.

jog joint negative
Set the joint number.

jog selected positive
Selected with a different widget or `STATUS`.

jog selected negative
Selected with a different widget or `STATUS`.

jog increment
Set metric/imperial/angular numbers.

jog rate
Set the float/alt float number.

feed override
Set the float/alt float number.

rapid override
Set the float/alt float number.

spindle override
Set the float/alt float number.

spindle fwd , spindle backward , spindle stop , spindle up , spindle down , view change
Set `view_type_string`.

limits override , flood , mist , block delete , optional stop , mdi command
Set `command_string`, ie calls a hard coded MDI command

INI mdi number
Set `ini_mdi_number`, ie calls an INI based MDI command

dro absolute , dro relative , dro dtg , exit screen
Closes down LinuxCNC

Override limits
Temporarily override hard limits

launch dialogs
Pops up dialogs if they are included in ui file.

set DRO to relative , set DRO to absolute , set DRO to distance-to-go

Attributes These set *attributes* of the selected action (availability depends on the widget):

toggle float option
Allows jog rate and overrides to toggle between two rates.

joint number
Selects the joint/axis that the button controls.

incr imperial number
Sets the imperial jog increment (set negative to ignore).

incr mm number
Sets the metric jog increment (set negative to ignore).

incr angular number
Sets the angular jog increment (set negative to ignore).

float number
Used for `jograte` and overrides.

float alternate number
For `jograte` and overrides that can toggle between two float numbers.

view type string
Can be:

- `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 command string that will be invoked if the MDI command action is selected.

ini_mdi_number
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.
Die Befehle werden durch das `;` getrennt.
Das `Label` wird nach dem Komma gesetzt, und das Symbol `\n` fügt einen Zeilenumbruch hinzu.

```
[MDI_COMMAND_LIST]
MDI_COMMAND = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND = G53 G0 Z0;G53 G0 X0 Y0, Goto\nMachn\nZero
```

Aktionbuttons sind eine Unterklasse von Abschnitt 12.7.5.1[IndicatedPushButton].
Siehe die folgenden Abschnitte für weitere Informationen über:

- LED Indikator Option
- Enabled on State
- Text Changes On State
- Call Python Command On State

12.7.2.2 ActionToolButton - Optional Actions Menu Button Widget

ActionToolButton buttons are similar in concept to action buttons, but they use *QToolButtons* to allow for **optional actions** to be selected by pushing and holding the button till the option menu pops up. Currently there is only one option: userView.

It is based on PyQt's *QToolButton*. **userView Record and Set User View Widget**

User View tool button allows to **record and return to an arbitrary graphics view**.

Press and hold the button to have the menu pop up and press *record view* to record the currently displayed graphics view.

Click the button normally to return to the last recorded position.

The recorded position will be remembered at shutdown if a preference file option is set up.

Anmerkung
Due to programming limitations, the recorded position may not show exactly the same. Particularly, if you pan zoomed out and pan zoomed in again while setting the desired view.
Best practice is to select a main view, modify as desired, record, then immediately click the button to switch to the recorded position. If it is not as you like, modify it's existing position and re-record.

12.7.2.3 RoundButton - Round Shapped ActionButton Widget

Round buttons work the same as *ActionButtons* other than the button is cropped round.

They are intended only to be visually different.

They have *two path properties* for displaying **images on true and false**.

12.7.2.4 AxisToolButton - Select and Set Axis Widget

This allows one to **select and set an axis**.

If the button is set checkable, it will indicate which axis is selected.

If you press and hold the button a pop up menu will show allowing one to:

- Nullen der Achse
- Divide the axis by 2
- Set the axis arbitrarily
- Reset the axis to the last number recorded

You select the axis by setting the joint number.

You can select a halpin option that is set true when the axis is selected.

It is based on PyQt's *QToolButton*.

12.7.2.5 `CamView` - Workpiece Alignment and Origin Setting Widget

This widget **displays a image from a web camera**.

It *overlays an adjustable circular and cross hair target* over the image.

CamView was built with precise visual positioning in mind.

Diese Funktion dient der **Ausrichtung des Werkstücks oder der Nullteilmerkmale mithilfe einer Webcam**.

It uses *OpenCV* vision library.

12.7.2.6 `DROLabel` - Axis Position Display Widget

This will **display the current position of an axis**.

`Qjoint_number`
: Joint number of offset to display (10 will specify rotational offset).

`Qreference_type`
: Actual, relative or distance to go (0,1,2).

`metric_template`
: Format of display ie `%10.3f`.

`imperial_template`
: format of display ie `%9.4f`.

`` `angular_template` ``
: Format of display ie `%Rotational: 10.1f`.

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.

Here is a sample stylesheet entry that:

- Sets the font of all `DRO_Label` widgets,
- Sets the text template (to set resolution) of the DRO,
- Then sets the text color based on the Qt `isHomed` property.

```
DROLabel {
    font: 25pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.4f’;
    qproperty-metric_template: ’%10.3f’;
    qproperty-angular_template: ’%11.2f’;
}

DROLabel[isHomed=false] {
    color: red;
}

DROLabel[isHomed=true] {
    color: green;
}
```

Here is how you specify a particular widget by it’s `objectName` in Qt Designer.

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

It is based on PyQt’s *QLabel*.

12.7.2.7 `GcodeDisplay` - G-code Text Display Widget

This **displays G-code in text form**, highlighting the currently running line.

Dies kann auch Folgendes anzeigen:

- **MDI-Verlauf**, wenn sich LinuxCNC im *MDI*-Modus befindet.
- **Log-Einträge**, wenn sich LinuxCNC im *MANUAL*-Modus befindet.
- **Preference file entries** if you enter `PREFERENCE` in capitals into the `MDILine` widget.

It has a *signal* **`percentDone(int)`** that can be connected to a slot (such as a `progressBar` to display percent run).

`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 festgelegt werden, indem der folgende Code zur .qss-Datei hinzugefügt wird.

```
EditorBase{
    qproperty-styleColorBackground: lightblue;
    qproperty-styleColor0: black;
    qproperty-styleColor1: #000000; /* black */
    qproperty-styleColor2: red;
    qproperty-styleColor3: black;
    qproperty-styleColor4: yellow;
    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-styleFontMargin: ”Times,14,-1,0,90,0,0,0,0,0”;
}
```

Für den *Standard-G-Code-Lexer* des Widgets `GcodeDisplay`:

- **styleColor0 = Default**: digit characters
- **styleColor1 = Comments**: characters inside of *msg()*
- **styleColor2 = Key**: alphabetic characters
- **styleColor3 = Assignment**: (%, <, >, #, =)
- **styleColor4 = Value**: ([,])

Font definitions:

```
”style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0”
```

It is based on PyQt’s *QsciScintilla*.

12.7.2.8 `GcodeEditor` - G-code Program Editor Widget

This is an extension of the `GcodeDisplay` widget that **adds editing convenience**.
It is based on PyQt's *QWidget* which incorporates `GcodeDisplay` widget.

12.7.2.9 `GCodeGraphics` - G-code Graphic Backplot Widget

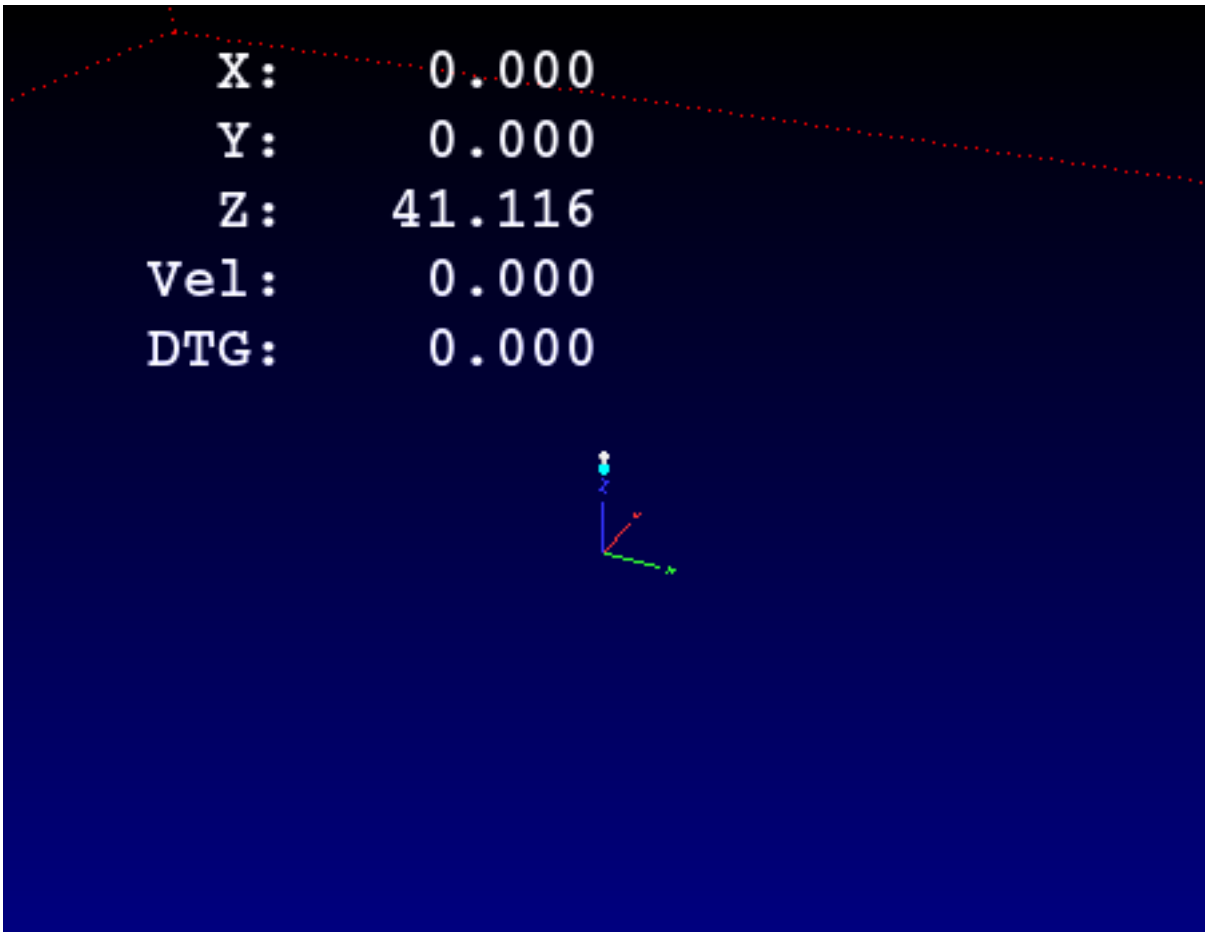


Abbildung 12.85: QtVCP GcodeGraphics: G-code Graphic Backplot Widget

This **displays the current G-code in a graphical form**.
Stylesheets Properties

`+_view+` *(string)*
Sets the *default view orientation* on GUI load.
Valid choices for a lathe are p, y, y2. For other screens, valid choices are p, x, y, z, z2.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_view: z;
}
```

`+_dro+` *(bool)*
Determines whether or not to *show the DRO*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

+_dtg+ ***(bool)***
Determine whether or not to *show the Distance To Go*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

+_metric+ ***(bool)***
Determines whether or not to *show the units in metric by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

+_overlay+ ***(bool)***
Determines whether or not to *show the overlay by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

+_offsets+ ***(bool)***
Determines whether or not to *show the offsets by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

+_small_origin+ ***(bool)***
Determines whether or not to *show the small origin by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

overlay_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default overlay color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

background_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default background color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

+_use_gradient_background+ ***(bool)***
Determines whether or not *use a gradient background by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
}
```

jog_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default jog color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

Feed_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default feed color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

Rapid_color ***(primary, secondary, or RGBA formatted color)***
Sets the *default rapid color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

InhibitControls ***(bool)***
Determines whether or not to *inhibit external controls by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

MouseButtonMode ***(int)***
Changes the *mouse button behavior* to rotate, move or zoom within the preview.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

There are 12 valid modes:

Mode	Move	Zoom	Rotate
0	Left	Middle	Right
1	Middle	Right	Left
2	Middle	Left	Right
3	Left	Right	Middle
4	Right	Left	Middle
5	Right	Middle	Left

Modes 6-11 are intended for machines that only require a 2D preview such as plasma or some Lathes and have no rotate button assigned.

Mode	Move	Zoom
6	Left	Middle
7	Middle	Left
8	Right	Left
9	Left	Right
10	Middle	Right
11	Right	Middle

`MouseWheelInvertZoom` *(bool)*
Determines wether or not to *invert the zoom direction* when zooming with the mouse wheel. The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-MouseWheelInvertZoom:True;
}
```

ACTION functions The `ACTION` library can control the G-code graphics widget.

`ACTION.RELOAD_DISPLAY()`
Reload the current program which recalculates the origin/offsets.

`ACTION.SET_GRAPHICS_VIEW(_view_)`
The following `view` commands can be sent:

- `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`

`ACTION.ADJUST_PAN(_X,Y_)`
Directly set the relative pan of view in x and y direction.

`ACTION.ADJUST_ROTATE(_X,Y_)`
Directly set the relative rotation of view in x and y direction.

It is based on PyQt's *OpenGL* widget.

12.7.2.10 `StateLabel` - Controller Modes State Label Display Widget

This will **display a label based on the machine controller modes true/false states**.

You can select between different texts based on true or false.

States Selection Properties The states are selectable via these properties:

`css_mode_status`
`True` when machine is in `G96` *Constant Surface Speed Mode*.

`diameter_mode_status`
`True` when machine is in `G7` *Lathe Diameter Mode*.

`fpr_mode_status`
`True` when machine is in `G95` *Feed per revolution Mode*.

`metric_mode_status`
`True` when machine is in `G21` *Metric Mode*.

Text templates properties

`true_textTemplate`
This will be the text set when the option is `True`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in true state, might be: *Metric Mode*

`false_textTemplate`
This will be the text set when the option is `False`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in false state, might be: *Imperial Mode*.

It is based on PyQt's *QLabel*.

12.7.2.11 `StatusLabel` - Controller Variables State Label Display Widget

This will **display a label based on variable status of the machine controller**. You can change how the state will be displayed by substituting.

You can use rich text for different fonts/colors etc.

Selectable States These states are selectable:

`actual_spindle_speed_status`
Used to display the actual spindle speed as *reported from the HAL pin* `spindle.0.speed-i`.
It's converted to *RPM*.
A `textTemplate` of `%d` would typically be used.

`actual_surface_speed_status`
Used to display the actual cutting surface speed on a lathe based on X axis and spindle speed.
It's converted to distance per minute.
A `textTemplate` of `%4.1f` (feet per minute) and `altTextTemplate` of `%d` (meters per minute) would typically be used.

`blendcode_status`
Shows the current `G64` setting.

`current_feedrate_status`
Shows the current actual feedrate.

`current_FPU_status`
Shows the current actual feed per unit.

`fcode_status`
Shows the current programmed `F` code setting.

`feed_override_status`
Shows the current feed override setting in percent.

`filename_status`
Shows the last loaded file name.

`filepath_status`
Shows the last loaded full file path name.

`gcode_status`
Shows all active G-codes.

`gcode_selected_status`
Show the current selected G-code line.

`halpin_status`
Shows the HAL pin output of a selected HAL pin.

`jograte_status`
Shows the current QtVCP based Jog Rate.

`jograte_angular_status`
Shows the current QtVCP based Angular Jog Rate.

`jogincr_status`
Shows the current QtVCP based Jog increment.

`jogincr_angular_status`
Shows the current QtVCP based Angular Jog increment.

machine_state_status
Shows the current *machine interpreter state* using the text described from the state_list.
The interpreter states are:

- `Estopped`
- `Running`
- `Stopped`
- `Paused`
- `Waiting`
- `Reading`

max_velocity_override_status
Shows the current max axis velocity override setting.

mcode_status
Shows *all active M-codes*.

requested_spindle_speed_status
Shows the requested spindle speed - actual may be different.

rapid_override_status
Shows the current rapid override setting in (0-100) percent.

spindle_override_status
Shows the current spindle override setting in percent.

timestamp_status
Shows the time based on the system settings.
An example of a useful `textTemplate` setting: `%I:%M:%S %p`.
See the Python time module for more info

tool comment_status
Returns the comment text from the current loaded tool.

tool diameter_status
Returns the diameter from the current loaded tool.

tool_number_status
Returns the tool number of the current loaded tool.

tool_offset_status
Returns the offset of the current loaded tool, indexed by `index_number` to select axis (0=x,1=y,etc)

user_system_status
Shows the *active user coordinate system* (`G5x` setting).

Other Properties

index_number
Integer that specifies the tool status index to display.

state_label_list
List of labels used for different machine states.

halpin_names
Name of the halpin to monitor (including HAL component basename).

textTemplate
: This is usually used for **imperial (G20) or angular numerical settings**, though not every option has imperial/metric conversion.
This uses *Python formatting rules* to set the text output.
One can use `%s` for no conversion, `%d` for integer conversion, `%f` for float conversion, etc.
You can also use *Qt rich text* code.
Typical template used for formatting imperial float numbers to text would be `%9.4f` or `%9.4f inch`.

alt_textTemplate
: This is usually used for **metric (G21) numerical settings**.
This uses *Python formatting rules* to set the text output.
Typical template used for formatting metric float to text would be `%10.3f` or `%10.3f mm`.

It is based on PyQt's *QLabel*.

12.7.2.12 `StatusImageSwitcher` - Controller Status Image Switcher

Status image switcher will **switch between images based on LinuxCNC states**.

***watch_spindle**
: Toggles between *3 images*: `stop`, `fwd`, `revs`.

***watch_axis_homed**
: Toggles between *2 images*: `axis not homed`, `axis homed`.

***watch_all_homed**
: Would toggle between *2 images*: `not all homed`, `all homed`.

***watch_hard_limits**
: Would toggle between *2 images or one per joint*.

Here is an example of using it to display an icon of Z axis homing state:

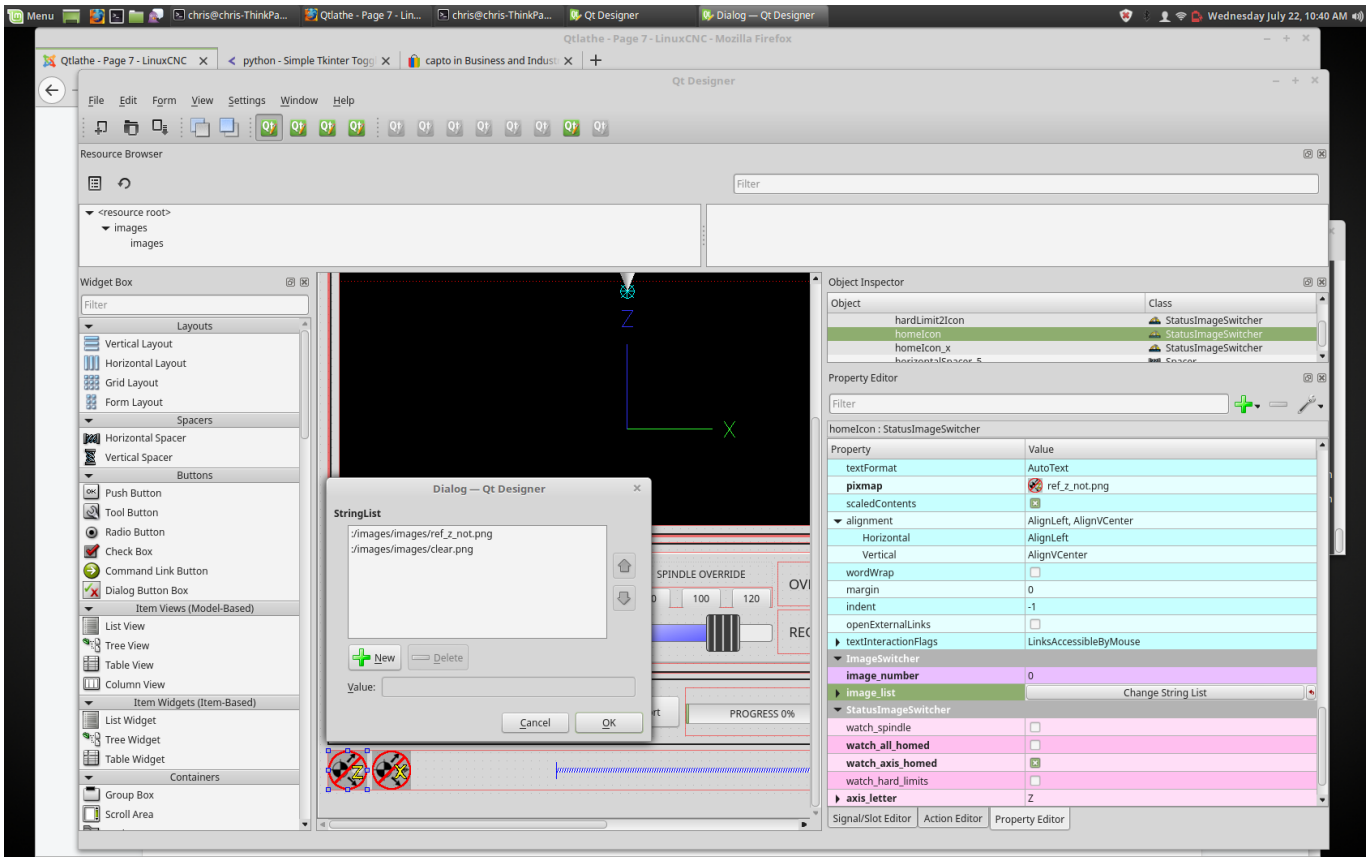


Abbildung 12.86: QtVCP `StatusImageSwitcher`: Controller Status Image Switcher

In the properties section notice that:

- `watch_axis_homed` is checked
- `axis_letter` is set to Z

If you double click the `image_list` a dialog will show and allow you to add image paths to.

If you have one image as an icon and one *clear image* then that will look like it shows and *hides the icon*.

Selecting image paths can be done by selecting the `pixmap` property and selecting an image.

Anmerkung
The `pixmap` setting is for test display only and will be ignored outside of Qt Designer.

- Right click the image name and you should see *Copy path*
- Click *Copy path*
- Now double click the *image list* property so the dialog shows.
- Click the *New* button
- Paste the image path in the entry box

Do that again for the next image.
Use a clear image to represent a hidden icon.

You can *test the images display* from the image list by changing the `image number`. In this case `0` is unhomed and `1` would be homed.
This is for test display only and will be ignored outside of Qt Designer.

12.7.2.13 `StatusStacked` - Mode Status Display Switching Widget

Dieses Widget **zeigt eines von drei Panels an, je nach Modus von LinuxCNC**.

This allows you to automatically display different widgets on *Manual*, *MDI* and *Auto* modes. **TODO** It is based on PyQt's *QStacked* widget.

12.7.2.14 `JogIncrements` - Jog Increments Value Selection Widget

This widget allows the user to **select jog increment values for jogging**.

The jogging values come from the *INI file* under:

- `[DISPLAY]INCREMENTS`, or
- `[DISPLAY]ANGULAR_INCREMENTS`

This will be *available to all widgets* through `STATUS`.

You can select linear or angular increments by the property **`linear_option`** in Qt Designer property editor.

It is based on PyQt's *ComboBox*.

12.7.2.15 `ScreenOption` - General Options Setting widget

This widget doesn't add anything visually to a screen but **sets up important options**.

This is the *preferred way to use these options*.

Eigenschaften These properties can be set in Qt Designer, in Python handler code or (if appropriate) in stylesheets.

These include:

`halCompBaseName`
: If left empty QtVCP will use the screen's name as the HAL component's basename.
If set, QtVCP will use this string as the HAL component's basename.
If the `-c` command line option is used when loading QtVCP, it will use the name specified on the command line - it overrides all above options.
If you programmatically set the basename in the `handlerfile` - it will override all above options.
This option cannot be set in stylesheets.

`notify_option`
: Hooking into the desktop notification bubbles for error and messages.

`notify_max_messages`
: Number of messages shown on screen at one time.

`catch_close_option`
: Catching the close event to pop up a *'are you sure' prompt*.

close_overlay_color
Color of transparent layer shown when quitting.

catch_error_option
Monitoring of the LinuxCNC error channel.
This also sends the message through `STATUS` to anything that registers.

play_sounds_option
Playing sounds using `beep`, `espeak` and the system sound.

use_pref_file_option
Setting up a *preferences file path*.
Using the magic word `WORKINGFOLDER` in the preference file path will be replaced with the launched configuration path, ie. `WORKINFOLDER/my_preferences`.

use_send_zmq_option
Used to initiate *ZMQ based outgoing messages*.

use_receive_zmq_messages
Used to initiate *ZMQ based in coming messages*.
These messages *can be used to call functions in the handler file*, allowing **external programs to integrate tightly with QtVCP** based screens.

embedded_program_option
Embed programs defined in the *INI*.

default_embed_tab
This is the property for a *default location to embed external programs*.
It should be set to name of a tab page widget in Qt Designer.

focusOverlay_option
Focus_overlay will put a transparent image or colored panel over the main screen to emphasize focus to an external event - typically a dialog.

messageDialog_option
Sets up the message dialog - used for general messages.

message_overlay_color
Color of transparent layer shown when the message dialog is shown.

closeDialog_option
Sets up the standard close screen prompt dialog.

entryDialog_option
Sets up the numerical entry dialog.

entryDialogSoftKey_option
Sets up a floating software keyboard when entry dialog is focused.

entry_overlay_color
Color of transparent layer shown when the entry dialog is shown.

toolDialog_option
Sets up the manual tool change dialog, including HAL pin.

tool_overlay_color
Color of transparent layer shown when the tool dialog is shown.

ToolUseDesktopNotify
Option zur Verwendung von Desktop-Benachrichtigungsdialogen für manuelle Werkzeugwechseldialoge.

ToolFrameless
Frameless dialogs can not be easily moved by users.

fileDialog_option
Sets up the file choosing dialog.

file_overlay_color
Color of transparent layer shown when the file dialog is shown.

keyboardDialog_option
Sets up a keyboard entry widget.

keyboard_overlay_color
Color of transparent layer shown when the keyboard dialog is shown.

vesaProbe_option
Sets up the Versa style probe dialog.

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
Color of transparent layer shown when the `macroTab` dialog is shown.

camViewDialog_option
Richtet den Kameraausrichtungsdialog ein.

camView_overlay_color
Color of transparent layer shown when the `camView` dialog is shown.

toolOffset_option
Sets up the tool offset display/editor dialog.

toolOffset_overlay_color
Color of transparent layer shown when the `toolOffset` dialog is shown.

originOffset_option
Richtet das Dialogfeld für die Anzeige/Editierung des Ursprungs ein.

originOffset_overlay_color
Color of transparent layer shown when the `originOffset` dialog is shown.

calculatorDialog_option
Sets up the calculator entry dialog.

calculator_overlay_color
Color of transparent layer shown when the calculator dialog is shown.

machineLogDialog_option
Richtet einen Dialog ein, um Protokolle von der Maschine und QtVCP anzuzeigen.

machineLog_overlay_color
Color of transparent layer shown when the `machineLog` dialog is shown.

runFromLineDialog_option
Richtet einen Dialog ein, der die Startoptionen anzeigt, wenn die Maschinenausführung von einer beliebigen Zeile aus gestartet wird.

runFromLine_overlay_color
Color of transparent layer shown when the `runFromLine` dialog is shown.

Setting Properties Programmatically 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)
}
```

```
/* red, green, blue 0-255, alpha 0-100% or 0.0 to 1.0 */
/* use widget class name */
/* matches/applied to all widgets of this class */
ScreenOptions {
  qproperty-close_overlay_color: rgba(0, 255, 0, 0.75)
}
```

Einige Einstellungen werden nur beim Start geprüft und führen daher nicht zu 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)*
: Whether to display errors/messages in the system's notification mechanism.

`notify_max_msgs` *(int)*
: Number of displayed errors at one time.

`shutdown_check` *(bool)*
: Ob ein Bestätigungsdialog erscheinen soll.

`sound_player_on` *(bool)*
: Turns all sounds on or off.

`[MCH_MSG_OPTIONS]`

`mchnMsg_play_sound` *(bool)*
: To play alert sound when dialog pops.

`mchnMsg_speak_errors` *(bool)*
: To use Espeak to speak error messages.

mchnMsg_speak_text ***(bool)***
To use Espeak to speak all other messages.

mchnMsg_sound_type ***(str)***
Sound to play when messages displayed. See notes below.

[USER_MSG_OPTIONS]

usermsg_play_sound ***(bool)***
To play alert sound when dialog pops.

userMsg_sound_type ***(str)***
Sound to play when user messages displayed. See notes below.

userMsg_use_focusOverlay ***(bool)***

[SHUTDOWN_OPTIONS]

shutdown_play_sound ***(bool)*** **, shutdown_alert_sound_type** ***(str)***
Sound to play when messages displayed. See notes below.

shutdown_exit_sound_type ***(str)***
Sound to play when messages displayed. See notes below.

shutdown_msg_title ***(str)***
Kurzer Titelstring, der im Dialog angezeigt wird.

shutdown_msg_focus_text ***(str)***
Large text string to superimpose in focus layer.

shutdown_msg_detail ***(str)***
Längere beschreibende Zeichenfolge zur Anzeige im Dialog

NOTIFY_OPTIONS

notify_start_greeting ***(bool)***
Whether to display a greeting dialog on start up.

notify_start_title ***(str)***
Short Title string.
If the speak option is also selected it will be spoken with Espeak.

notify_start_detail ***(str)***
Longer description string.

notify_start_timeout ***(int)***
Time in seconds to display before closing.

*_sound_type Einträge

- **System Sounds**
  In Debian/Ubuntu/Mint based installations these *system sounds* should be available as sound-type entries above:

  - ERROR
  - READY
  - DONE
  - ATTENTION
  - RING
  - LOGIN
  - LOGOUT
  - BELL

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 Beeps**
  If the `beep` *kernel module* is installed and it is not disabled, these sound-type entries are available:
  - `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 '_my message_'`**

12.7.2.16 `StatusSlider` - Controller-Einstellungs-Schieberegler-Widget

This widget allow the user to **adjust a LinuxCNC setting via a slider**.

Die Kurzbeschreibung kann folgendes anpassen:

- Jog rate
- Winkel-Jog-Rate
- Vorschubgeschwindigkeit
- Spindel-Override-Rate
- Eilgang Übersteuerungsrate (engl. rapid override rate)

Eigenschaften `StatusSlider` has the following properties:

`halpin_option`
: Sets option to make a HAL float pin that reflects current value.

`rapid_rate`
: Selects a rapid override rate slider.

`feed_rate`
: Selects a feed override rate slider.

`spindle_rate`
: Selects a spindle override rate slider.

`jograte_rate`
: Wählt einen linearen Jograte-Schieberegler aus.

`jograte_angular_rate`
: Selects a angular jograte slider.

`max_velocity_rate`
: Selects a maximum velocity rate slider.

`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;
}
```

It is based on PyQt's *QSlider*.

12.7.2.17 StateLED - Controller-Status-LED-Widget

This widget gives **status on the selected LinuxCNC state**.

States Die Statusoptionen sind:

is_paused_status , is_estopped_status , is_on_status , is_idle_status_ , is_homed_status , is_fl

Eigenschaften Es gibt Eigenschaften, die geändert werden können:

halpin_option
Fügt einen Ausgangspin hinzu, der den ausgewählten Zustand wiedergibt.

invert_state_status
Invert the LED state compared to the LinuxCNC state.

diameter
Diameter of the LED.

color
Color of the LED when on.

off_color
Color of the LED when off.

alignment
Qt-Hinweis zur Ausrichtung.

state
Aktueller Zustand der LED (zum Testen in Qt Designer).

flashing
Schaltet die Blinkoption ein und aus.

flashRate
Sets the flash rate.

The LED properties can be defined in a stylesheet with the following code added to the `.qss` file.

```
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.

It is based on the *LED* widget.

12.7.2.18 `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:

- Jog rate
- Winkel-Jog-Rate
- Vorschubgeschwindigkeit
- Spindel-Override-Rate
- Eilgang Übersteuerungsrate (engl. rapid override rate)

It is based on PyQt's *QProgressBar*.

12.7.2.19 `SystemToolButton` - Widget zur Auswahl des Benutzersystems

Mit diesem Widget können Sie **manuell ein Benutzersystem auswählen, indem Sie es gedrückt halten**.

Wenn Sie den Text der Schaltfläche nicht festlegen, wird sie automatisch auf das aktuelle System aktualisiert.

It is based on PyQt's *QToolButton*.

12.7.2.20 `MacroTab` - Spezielles Makro-Widget

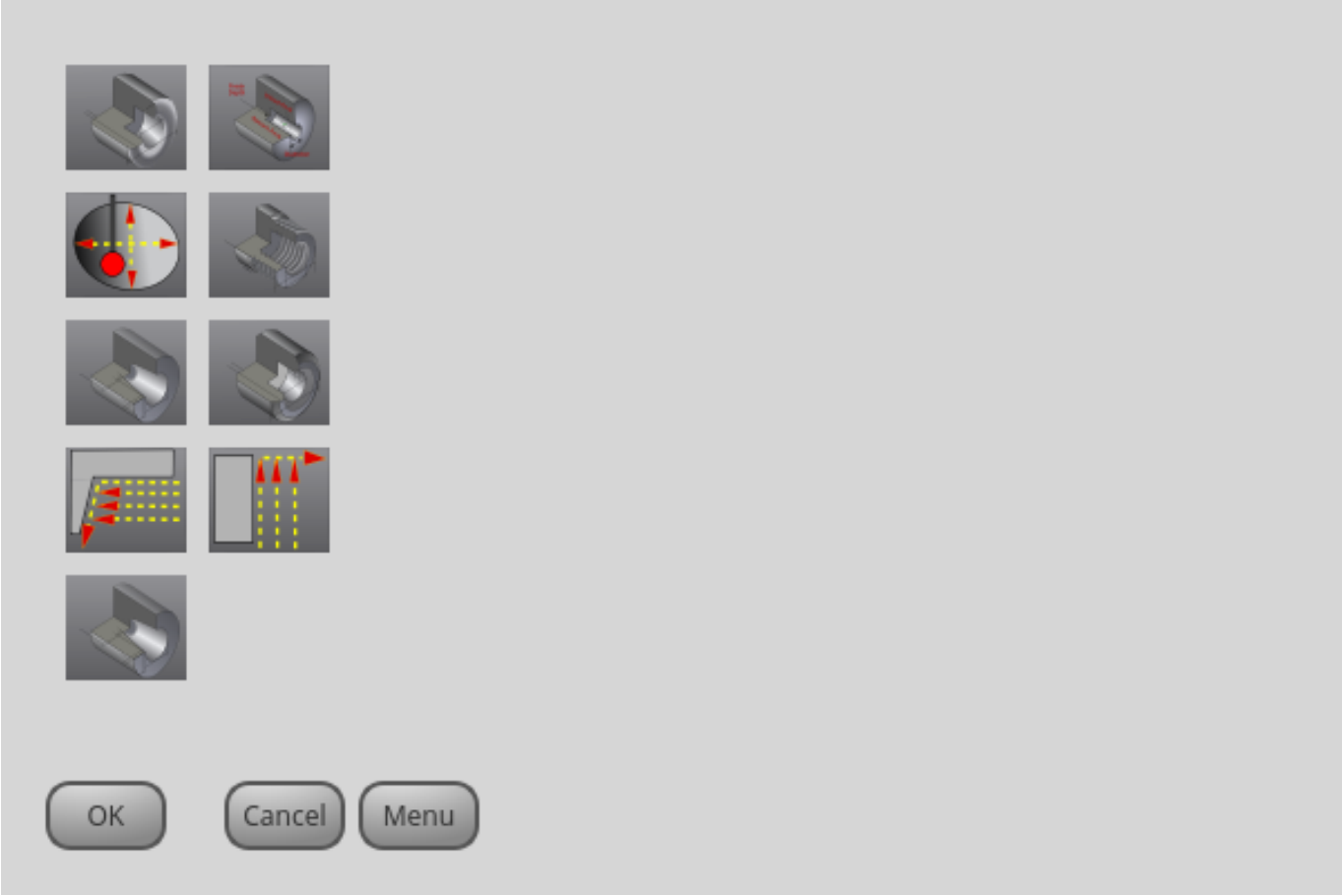


Abbildung 12.87: QtVCP `MacroTab`: Spezielles Makro-Widget

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 ”true” oder ”false” in der Liste verwenden, wird ein ”**checkbutton**” angezeigt.

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)
  : **Dateiname des Bildes** 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:

LOAD:yes
: Zeigt einen Button zum Laden an.

SAVE:yes
: Zeigt einen Button zum Speichern an.

12.7.2.21 `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 Link:../hal/tools.html#sec:halmeter[`halmeter`] Dienstprogramm.

`HALSHOW`
Startet das LinuxCNC Link:../hal/halshow.html#cha:halshow[`halshow`] Dienstprogramm.

`HALSCOPE`
Startet das LinuxCNC Link:../hal/tutorial.html#sec:tutorial-halscope[`halscope`] Dienstprogramm.

`STATUS`
Startet das LinuxCNC Link:../man/man1/linuxcnctop.1.html[`status`] Dienstprogramm.

`CALIBRATION`
Startet den LinuxCNC Link:../getting-started/updating-linuxcnc.html#_calibration_emccalib_tcl[calibr
utility.

`CLASSICLADDER`
Startet die Link:../ladder/classic-ladder.html[ClassicLadder GUI], wenn die *ClassicLadder real-time HAL component* durch die Konfigurationsdateien des Rechners geladen wurde.

`PREFERENCE`
Lädt die Einstellungsdatei in den `GcodeEditor`.

`CLEAR HISTORY`
Löscht den MDI-Verlauf.

`net`
Siehe Link:../man/man1/halcmd.1.html#COMMANDS[`halcmd net` commands].
Wenn der Befehl nicht erfolgreich ist, wird ein Fehler ausgegeben.

- *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`

Anmerkung
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*.

12.7.2.22 `MDIHistory` - MDI-Befehlsverlaufs-Widget

Zeigt eine **scrollbare Liste vergangener MDI-Befehle** an.

Für MDI-Befehle ist eine Bearbeitungszeile eingebettet.

Die gleichen eingebetteten MDILine-Befehle können über dieses Widget aufgerufen 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'
```

12.7.2.23 `MDITouchy` - Touchscreen-MDI-Eingabe-Widget

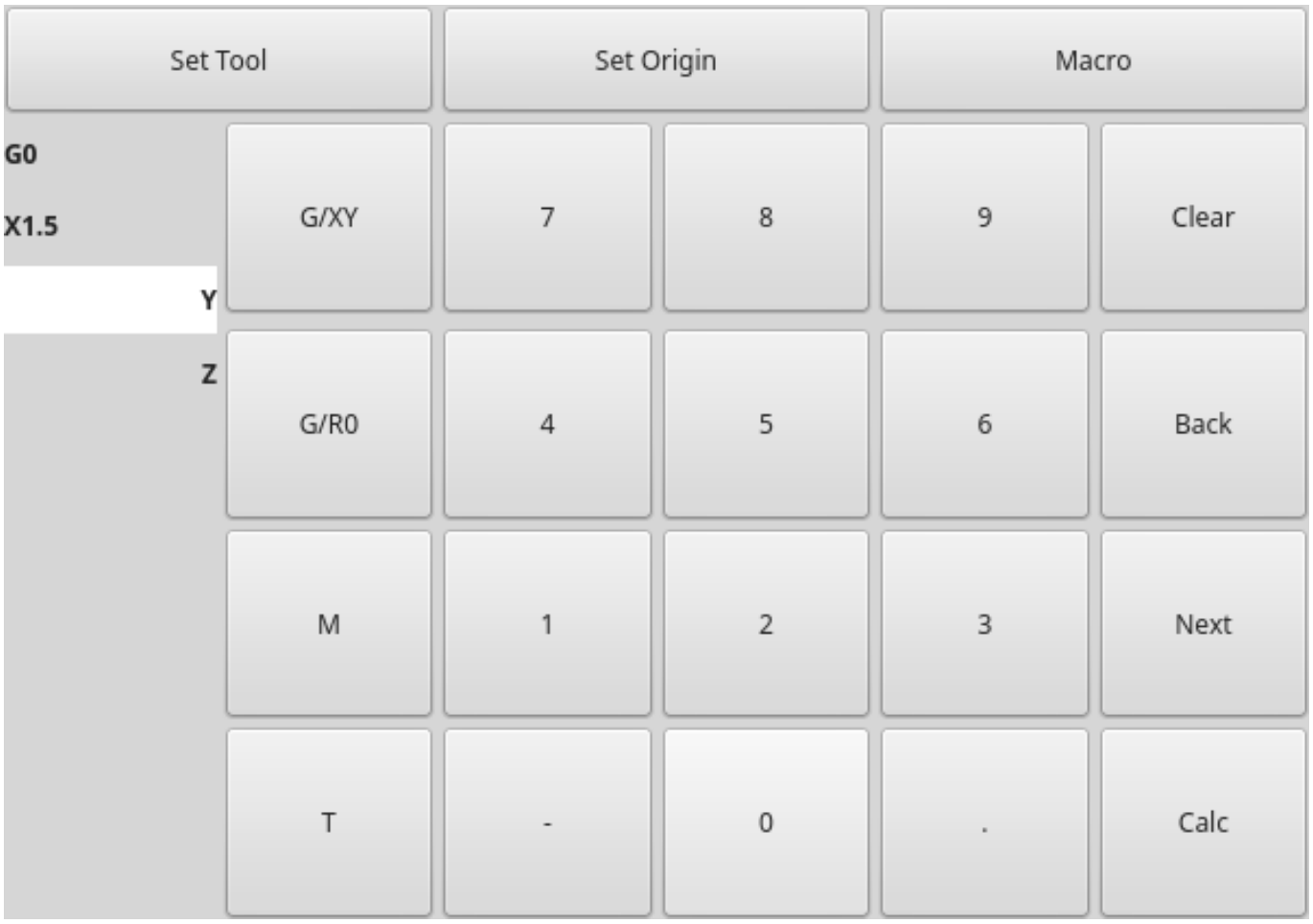


Abbildung 12.88: QtVCP `MDITouchy`: Touchscreen-MDI-Eingabe-Widget

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/RO", "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` FIXME
- `Set Origin` FIXME
- `Macro` FIXME

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 von Aktionsschaltflächen**
  Wenn das Widget in Qt Designer *mditouchy* genannt wurde, verwenden Sie die Option *Call Python commands* der Aktionsschaltfläche Schaltfläche die Option *Python-Befehle aufrufen* und geben Sie 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.

12.7.2.24 `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

Abbildung 12.89: 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.

The display and entry will change between metric and imperial based on LinuxCNC's *current* `G20` / `G21` setting.

The current in-use user system will be highlighted.

Extra actions can be integrated to manipulate settings.
These actions depend on extra code added either to a combined widget, like `originoffsetview` dialog, or the screens handler code.
Typical actions might be *Clear Current User offsets* or *Zero X*.

Clicking on the columns and rows allows one to adjust the settings.

A dialog can be made to popup for data or text entry.

The comments section will be recorded in the preference file.

Es basiert auf PyQt's *QTableView*, *QAbstractTableModel*, und *ItemEditorFactory*.
Eigenschaften, Funktionen und Stile der PyQt-Basisobjekte sind immer verfügbar.

Eigenschaften `OriginOffsetView` has the following properties:

`dialog_code_string`
: Sets which dialog will pop up with numerical entry.

`test_dialog_code_string`
: Sets which dialog will pop up with text entry.

`metric_template`
: Metric numerical data format.

`imperial_template`
: Imperial numerical data format.

`styleCodeHighlight`
: Current in-use user system highlight color.

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')
```

- Or (if appropriate) in stylesheets

```
OriginOffsetView{
  qproperty-styleColorHighlist: lightblue;
}
```

12.7.2.25 `StateEnableGridlayout` - Controller State Enabled Container Widget

```
 _disable the widgets inside it depending on
LinuxCNC’s current state_.
```

This is a **container that other widgets can be placed in**.

Embedded widgets are be greyed-out when the `StateEnableGridlayout` is disabled.

It can selectably react to:

- Machine on
- Interpreter idle
- Estop off
- All-homed

It is based on PyQt’s *QGridLayout*.

12.7.2.26 `MachineLog` - Machine Events Journal Display Widget

FIXME MachineLog documentation

12.7.2.27 `JointEnableWidget` - FIXME

FIXME JointEnableWidget documentation

12.7.2.28 `StatusImageSwitcher` - Controller Status Image Switching Widget

Dieses Widget wird **Bilder basierend auf dem LinuxCNC-Status anzeigen**.

You can watch:

- the state of the spindle,
- the state of all homed,
- the state of a certain axis homed,
- the state of hard limits.

It is based on PyQt’s FIXME

12.7.2.29 `FileManager` - File Loading Selector Widget

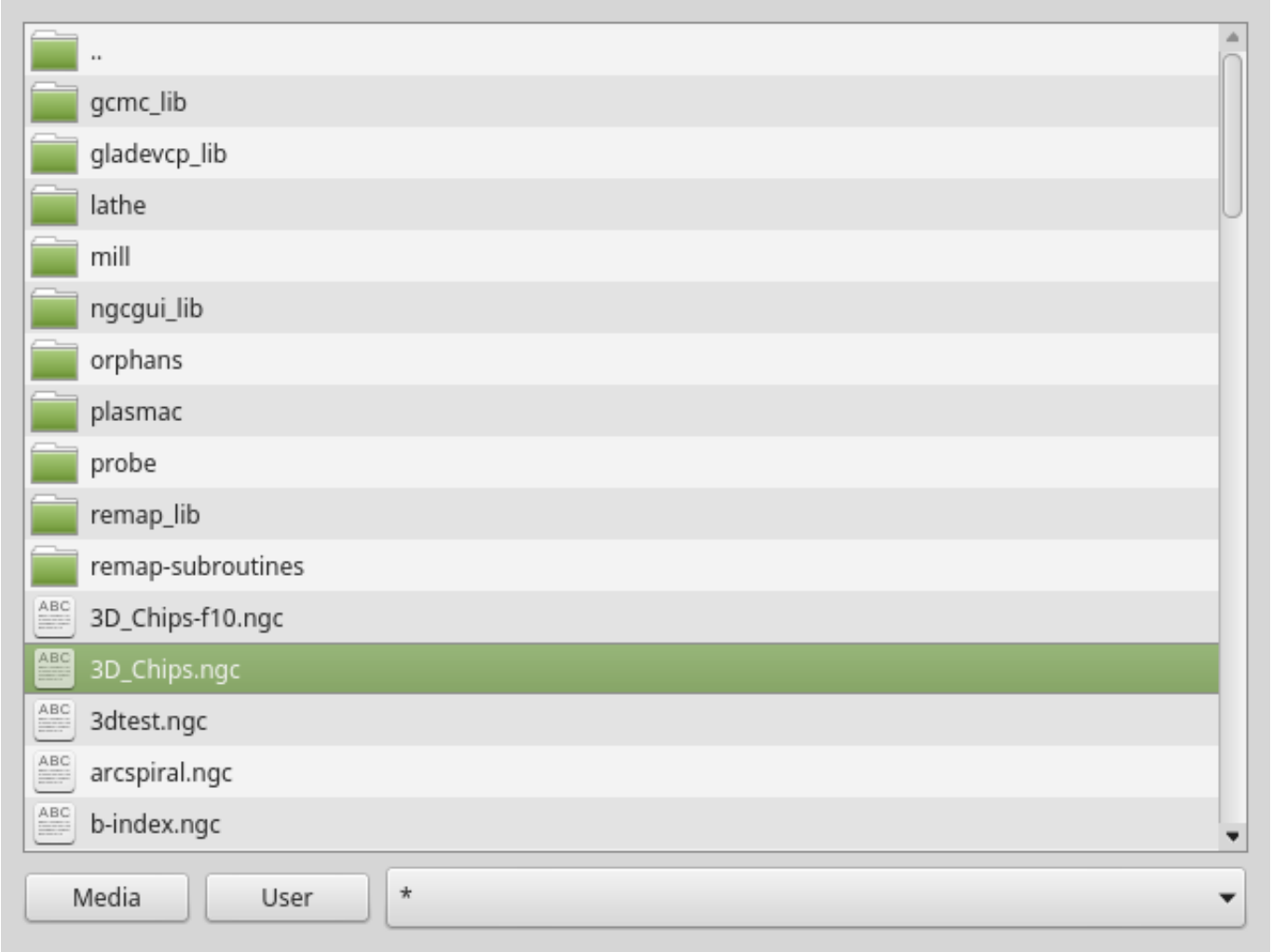


Abbildung 12.90: QtVCP `FileManager`: File Loading Selector Widget

This widget is used to **select files to load**.

Sie verfügt über die Möglichkeit, die Namen mit Hardware wie einem Handgerät (engl. MPG) zu kennzeichnen.

One can class patch the function `load(self,fname)` to customize file loading.

The function `getCurrentSelected()` will return a Python tuple, containing the file path and whether it's a file.

```
temp = FILEMANAGER.getCurrentSelected()
print('filepath={}'.format(temp[0]))
if temp[1]:
    print('Is a file')
```

It is based on PyQt's FIXME

12.7.2.30 `RadioAxisSelector` - FIXME

FIXME RadioAxisSelector documentation

12.7.2.31 `ToolOffsetView` - Tools Offsets View And Edit Widget

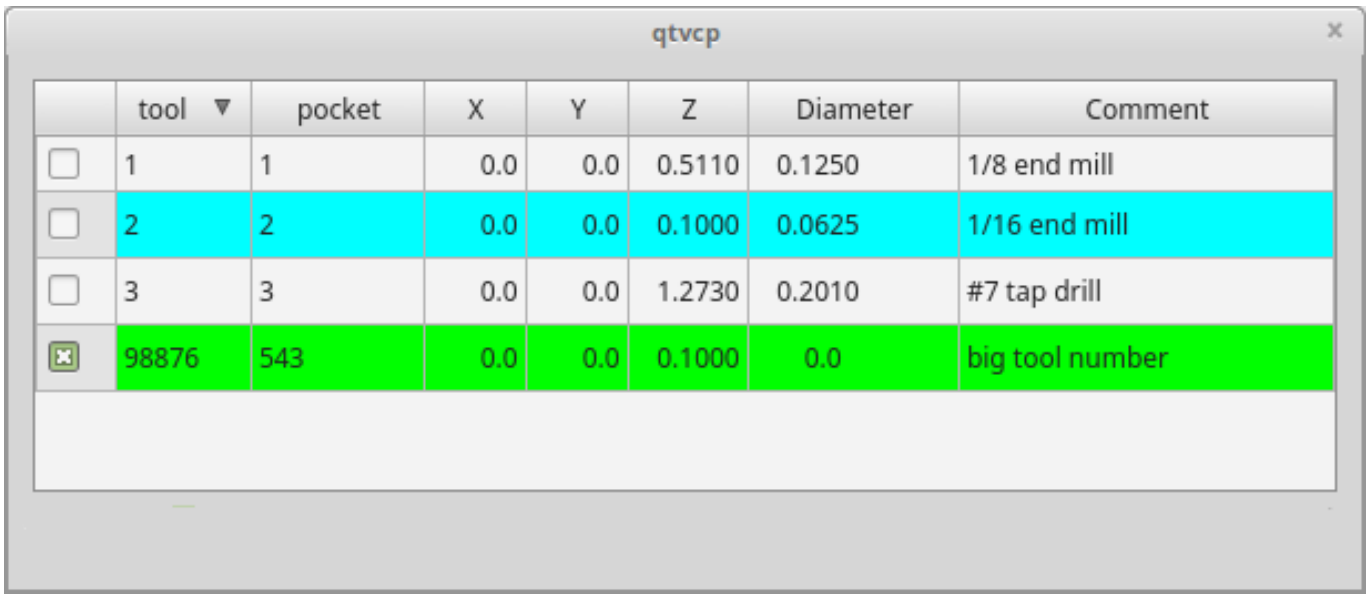


Abbildung 12.91: QtVCP `ToolOffsetView`: Tools Offsets View And Edit Widget

This widget **displays and allows one to modify tools offsets**.

It will *update LinuxCNC's tool table* for changes made or found.

The tool settings can only be changed in LinuxCNC after homing and when the motion controller is idle.

The display and entry will change between metric and imperial based on LinuxCNC's *current* `G20/G21` setting.

The current in-use tool will be highlighted, and the current selected tool will be highlighted in a different color.

The checkbox beside each tool can be used to select too for an *action* that depends on extra code added either to a combined widget like the `toolOffsetView` dialog or the screens handler code. Typical actions are *load selected tool*, *delete selected tools*, etc.

Clicking on the columns and rows allows one to adjust the settings.

A dialog can be made to popup for data or text entry.

The comments section will typically be displayed in the manual tool change dialog.

If using a *lathe configuration*, there can be columns for X and Z wear.

To use these columns to adjust the *tool wear*, it requires a remapped tool change routine.

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`
: Sets which dialog will pop up with numerical entry.

`test_dialog_code_string`
: Sets which dialog will pop up with text entry.

metric_template
Metric numerical data format.

imperial_template
Imperial numerical data format.

styleCodeHighlight
Current tool-in-use highlight color.

styleCodeSelected
Selected highlight color.

In a handler file:

```
self.w.tooloffsetview.setProperty('dialog_code','CALCULATOR')
self.w.tooloffsetview.setProperty('metric_template','%10.3f')
```

and in style sheets:

```
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()
```

12.7.2.32 `BasicProbe` - Einfaches Fräs-Tast-Widget

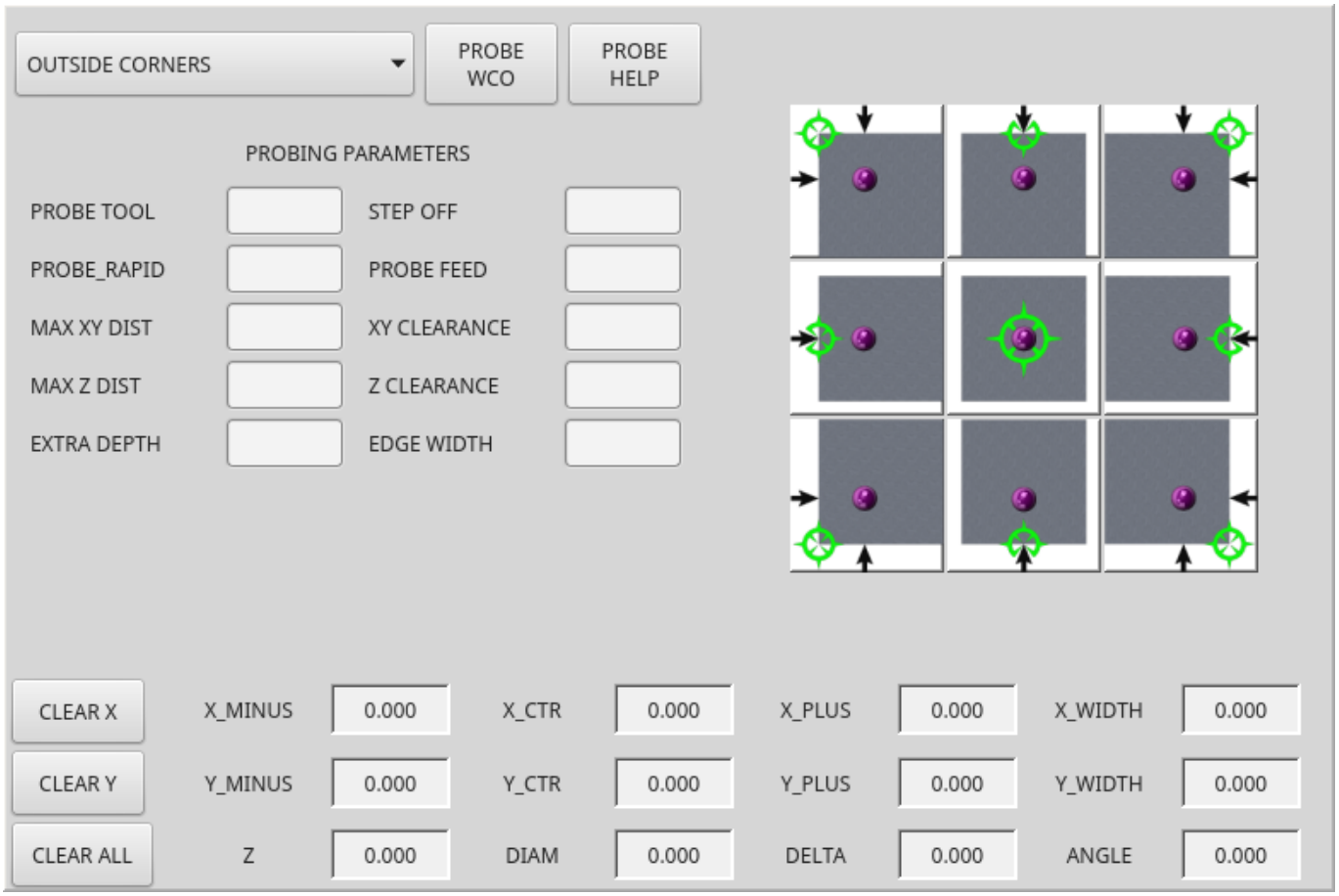


Abbildung 12.92: QtVCP *BasicProbe*: Einfaches Fräs-Tast-Widget

Widget zum **Sondieren auf einer Fräse**. Wird vom *QtDragon*-Bildschirm verwendet.

12.7.3 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)
```

- **Eine Funktion zum Aufrufen eines Dialogs hinzufügen**:
  Diese Funktion muss *ein Nachrichten-Diktat erstellen, das an den Dialog gesendet wird*.
  Diese Nachricht wird in der allgemeinen Nachricht zurückgegeben mit dem Zusatz der *Return-Variable*.
  Es ist möglich, *zusätzliche Benutzerinformationen* zu der Nachricht 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)
```

- **Add a callback function that processes the general message**:
  Keep in mind this function will *get all general messages* so the `dict` keynames are not guaranteed to be there. Using the `.get()` function and/or using `try/except` is advisable. This function should:

  - 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))
```

12.7.3.1 `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.

12.7.3.2 `ToolDialog` - Dialog-Widget für den manuellen Werkzeugwechsel

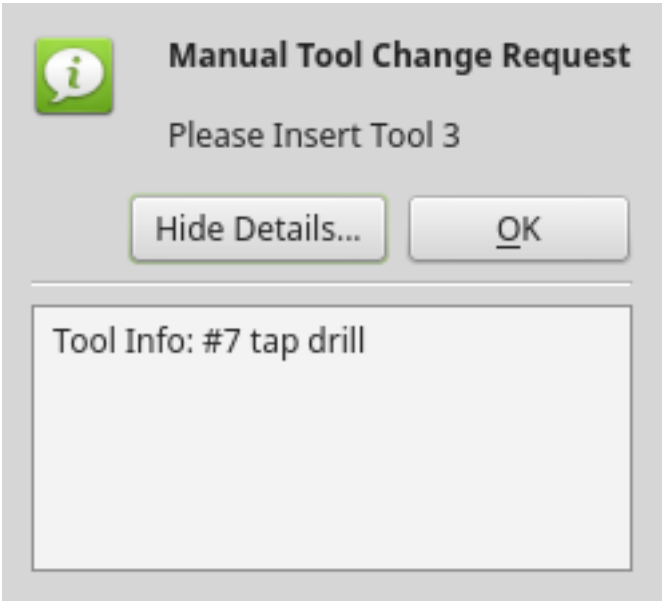


Abbildung 12.93: QtVCP `ToolDialog`: Dialog zum manuellen Werkzeugwechsel

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.

12.7.3.3 `FileDialog` - Dialog Widget zum Laden und Speichern von Dateien

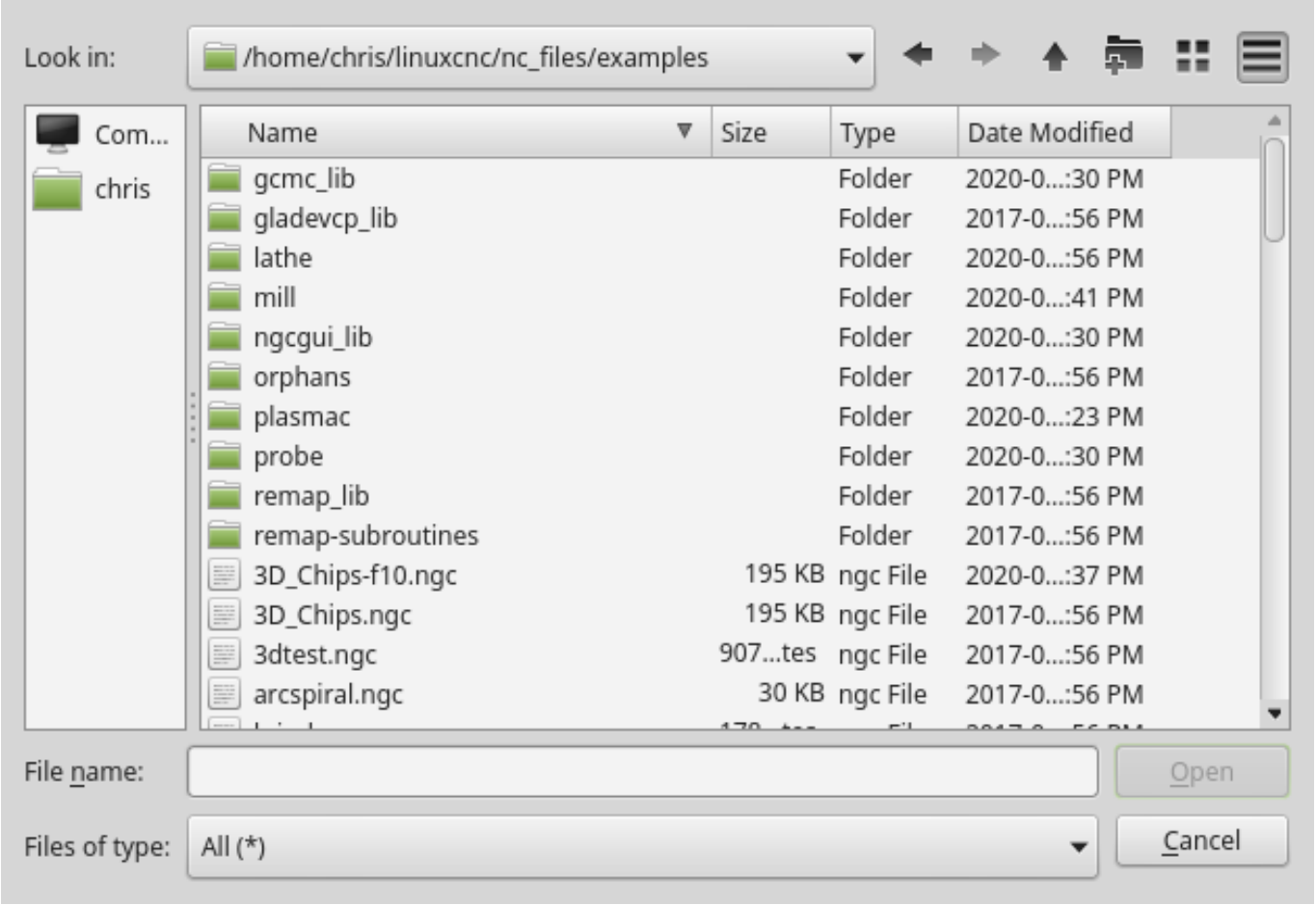


Abbildung 12.94: QtVCP `FileDialog`: Widget für das Laden und Speichern von Dateien

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.

12.7.3.4 `OriginOffsetDialog` - Dialogfeld-Widget für die Einstellung des Ursprungsversatzes

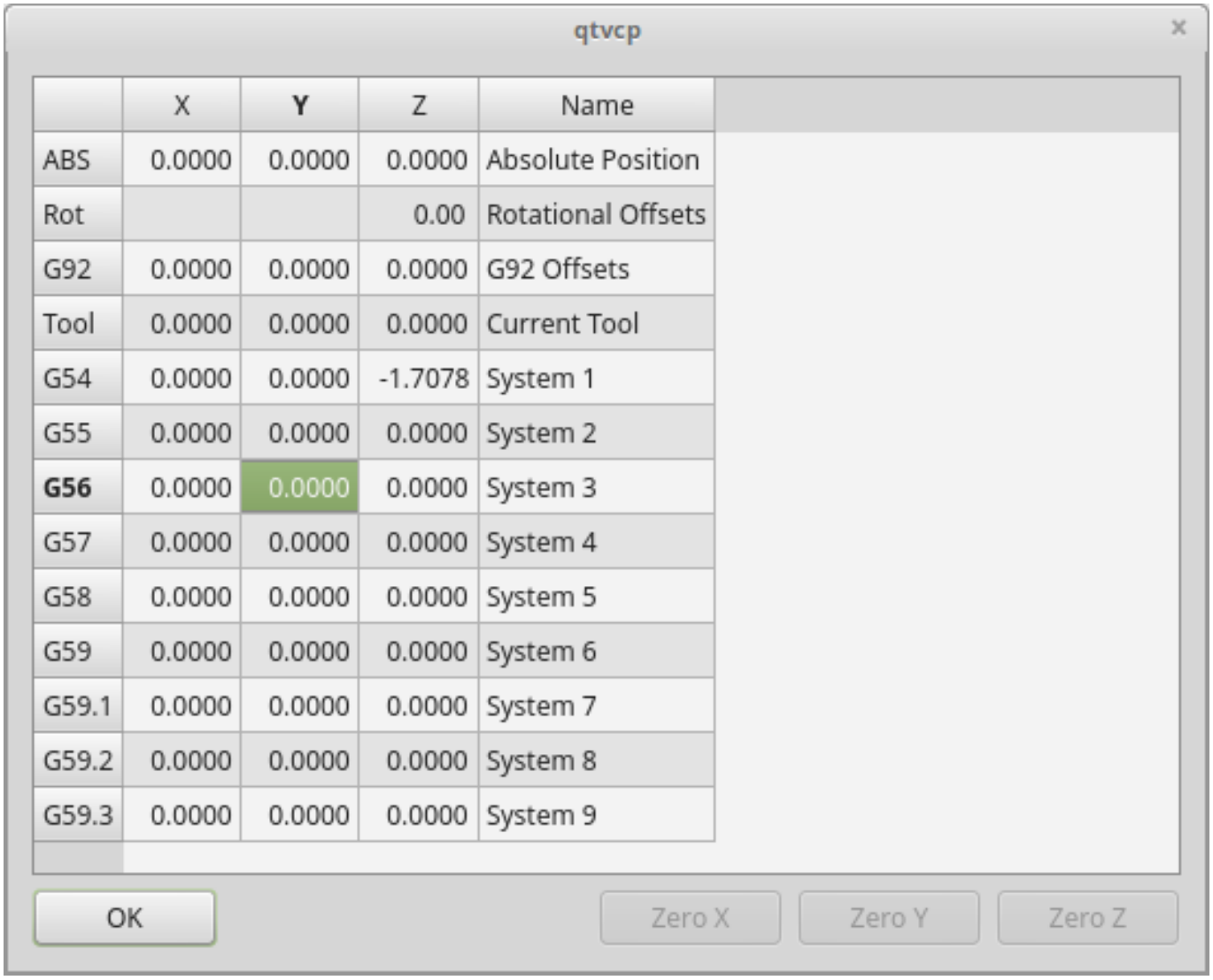


Abbildung 12.95: QtVCP `OriginOffsetDialog`: Widget zur Einstellung des Ursprungsversatzes

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*.

12.7.3.5 `ToolOffsetDialog` - Dialogfenster-Widget zur Einstellung des Werkzeugversatzes

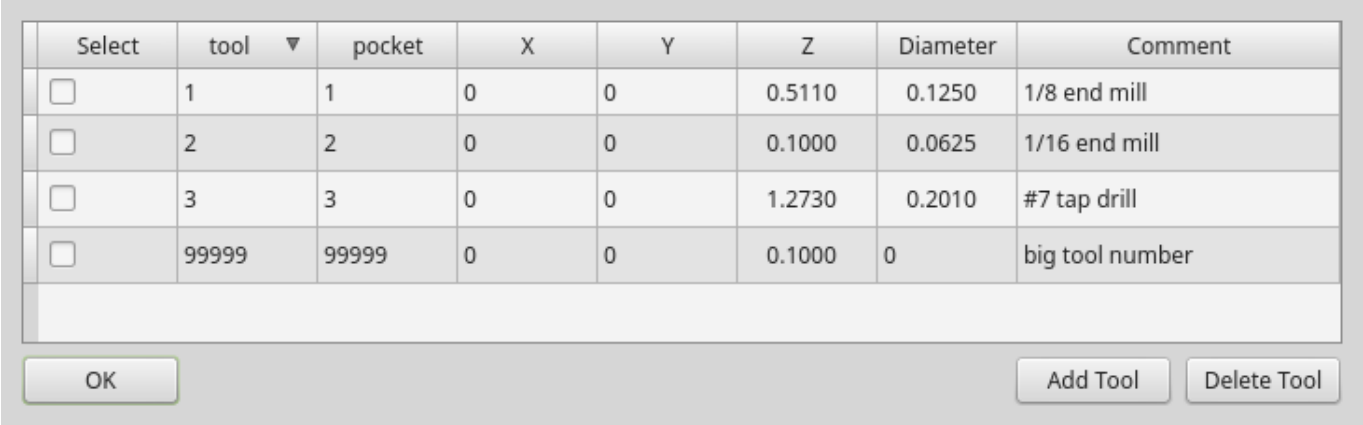


Abbildung 12.96: QtVCP `ToolOffsetDialog`: Werkzeug-Offset-Einstellungsdialog-Widget

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*.

12.7.3.6 `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.
When using ``STATUS``'s `request-dialog` function, the default launch name is **`MACROTAB`**.
Es basiert auf PyQts *QDialog*.

12.7.3.7 `CamViewDialog` - WebCam Part Alignment Dialog Widget

This is a dialog to **display the CamView widget for Webcam part alignment**.
When using ``STATUS``'s `request-dialog` function, the default launch name is **`CAMVIEW`**.
Es basiert auf PyQts *QDialog*.

12.7.3.8 `EntryDialog` - Edit Line Dialog Widget

This is a dialog to **display an edit line for information entry**, such as origin offset.
It returns the entry via `STATUS` messages using a Python `DICT`.
The `DICT` contains at minimum, the name of the dialog requested and an id code.
When using ``STATUS``'s `request-dialog` function, the default launch name is **`ENTRY`**.
Es basiert auf PyQts *QDialog*.

12.7.3.9 `CalculatorDialog` - Calculator Dialog Widget

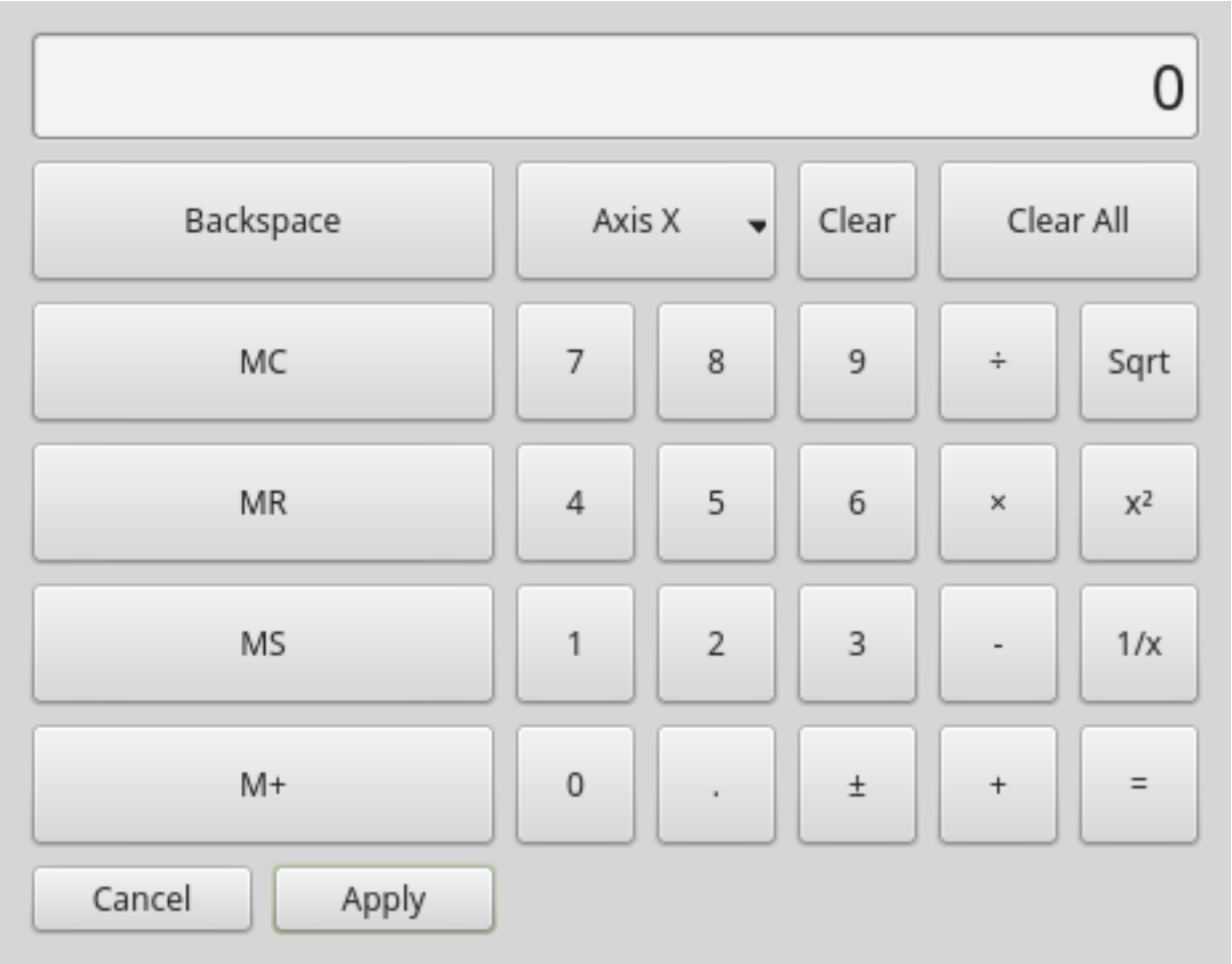


Abbildung 12.97: QtVCP `CalculatorDialog`: Calculator Dialog Widget

This is a dialog to **display a calculator for numeric entry**, such as origin offset.
It returns the entry via `STATUS` messages using a Python `DICT`.
The `DICT` contains at minimum, the name of the dialog requested and an id code.
When using ``STATUS``'s `request-dialog` function, the default launch name is **`CALCULATOR`**.
Es basiert auf PyQts *QDialog*.

12.7.3.10 `RunFromLine` - Run-From-Line Dialog Widget

Abbildung 12.98: QtVCP `RunFromLine`: Run-From-Line Dialog Widget

Dialog to **preset spindle settings before running a program from a specific line**.
Es basiert auf PyQts *QDialog*.

12.7.3.11 `VersaProbeDialog` - Part Touch Probing Dialog Widget

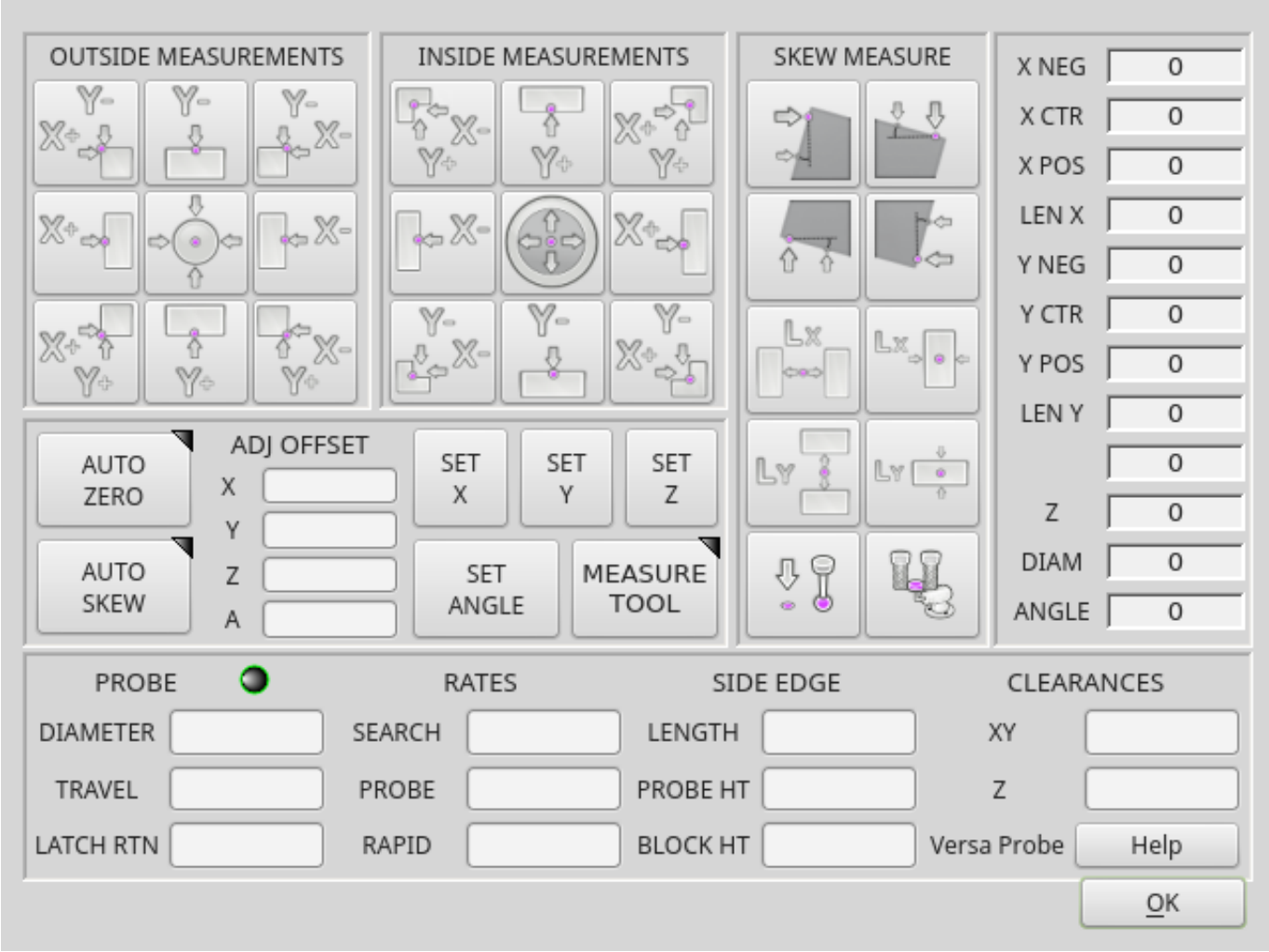


Abbildung 12.99: QtVCP `VersaProbeDialog`: Part Touch Probing Dialog Widget

This is a dialog to display a **part probing screen based on Versa Probe**.
Es basiert auf PyQts *QDialog*.

12.7.3.12 `MachineLogDialog` - Machine and Debugging Logs Dialog Widget

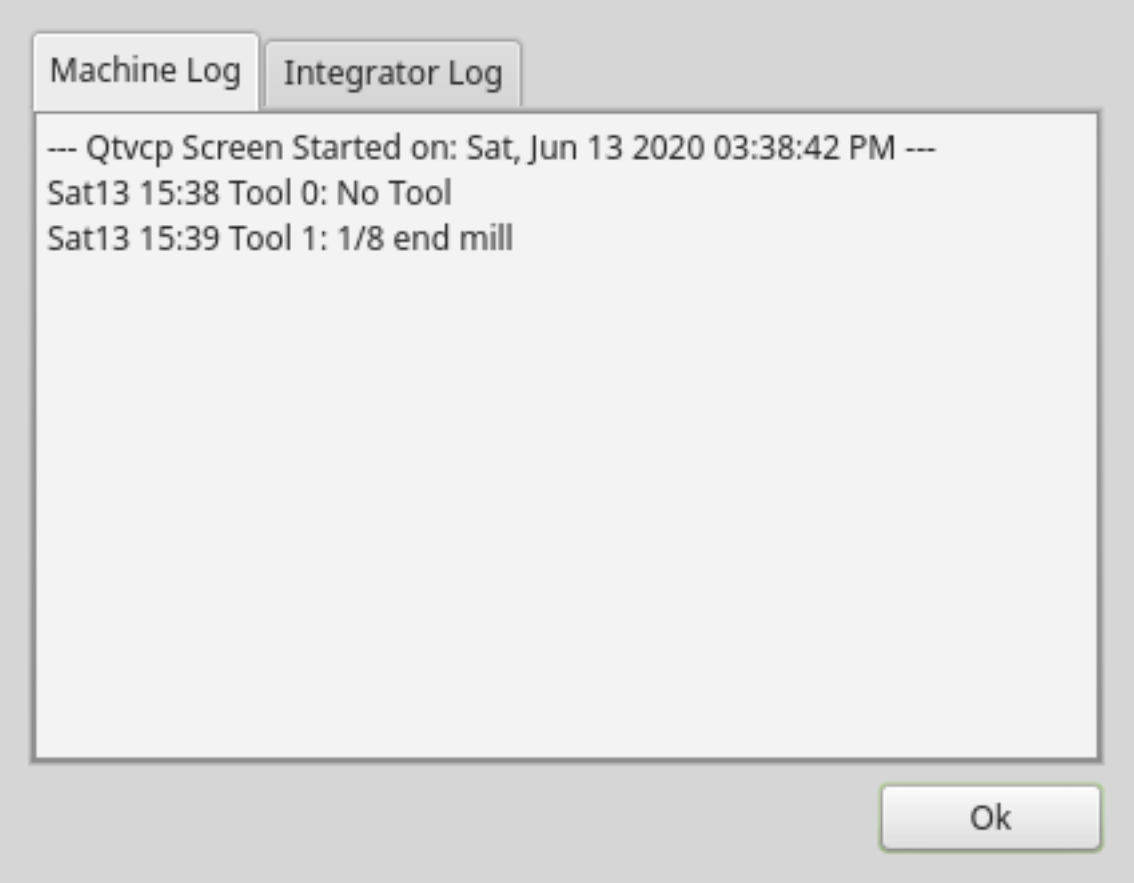


Abbildung 12.100: QtVCP `MachineLogDialog`: Machine and Debugging Logs Dialog Widget

This is a dialog to **display the machine log and QtVCP's debugging log**.
Es basiert auf PyQts *QDialog*.

12.7.4 Other Widgets

Other available widgets:

12.7.4.1 `NurbsEditor` - NURBS Editing Widget

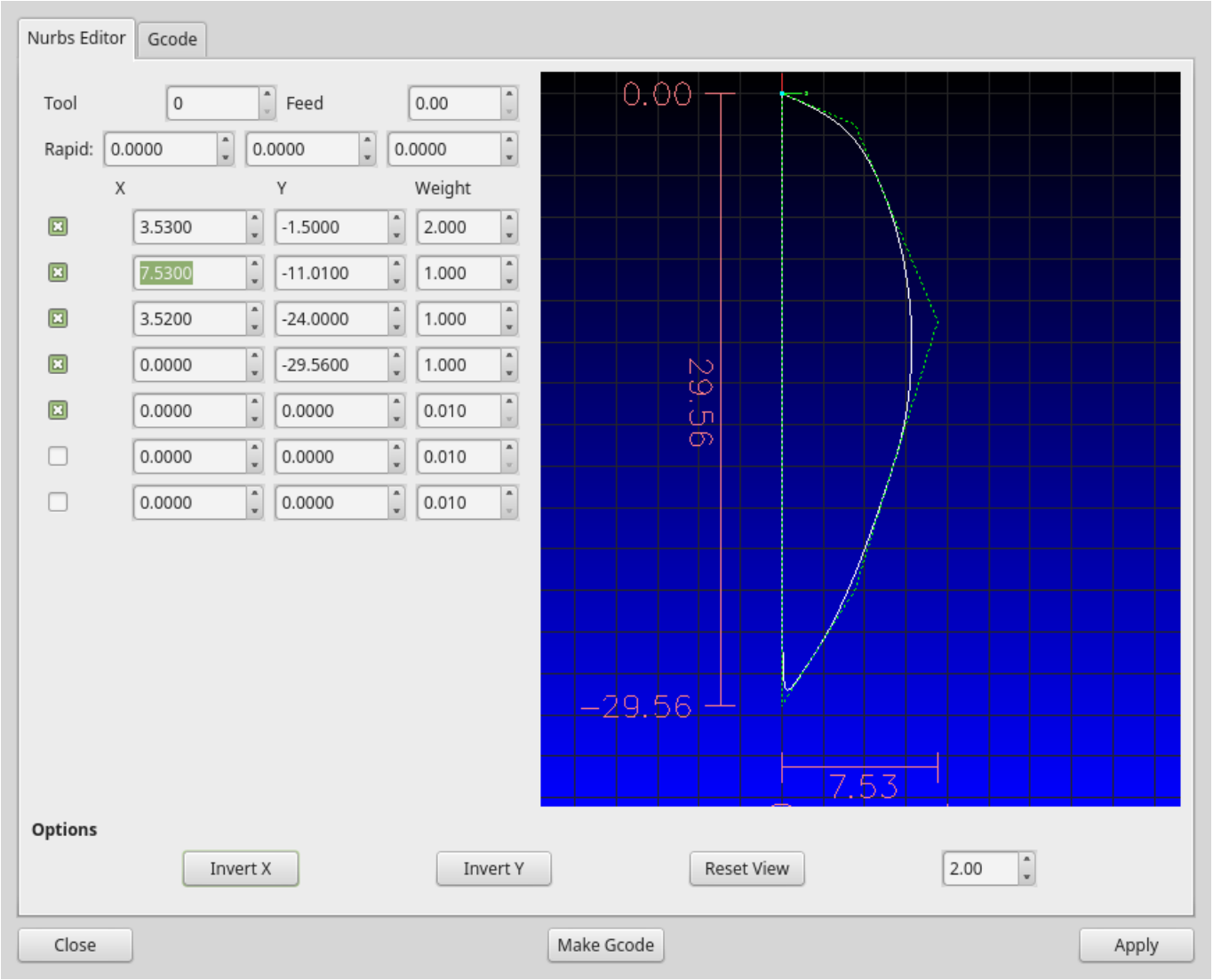


Abbildung 12.101: QtVCP `NurbsEditor`: NURBS Editing Widget

The Nurbs editor allows you to **manipulate a NURBS based geometry** on screen and then **convert NURBS to G-code**.

You can edit the G-code on screen and then send it to LinuxCNC.

Es basiert auf PyQts *QDialog*.

12.7.4.2 `JoyPad` - 5 button D-pad Widget

It is the base class for the `HALPad` widget.

This widget looks and acts like a **5 button D-pad, with a LED like indicators in a ring**.

You can put text or icons in each of the button positions.

You can *connect to output signals* when the buttons are pressed.

There are also *input slots* to change the color of the indicator(s).

ENUMS There are **enumerated constants used to reference indicator positions**.
They are used in Qt Designer editor's property editor or in Python code.

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)
```

Useful Override-able Functions As coded they *issue signals for the button pressed or released*.

On signal outputs a string code for the button, one signal outputs a bool value.

```
def _pressedOutput(self, btncode):
    self.joy_btn_pressed.emit(btncode)
    self['joy_{}_pressed'.format(btncode.lower())].emit(True)

def _releasedOutput(self, btncode):
    self.joy_btn_released.emit(btncode)
    self['joy_{}_pressed'.format(btncode.lower())].emit(False)
```

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_)
: Sets the button's icon pixmap.

set_button_text(_button_, _text_)
: Sets the button's icon text.

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 These signals will be **sent when buttons are pressed**.

They can be connected to in Qt Designer editor or Python code.

The first two output a string that indicates the button pressed:

joy_btn_pressed *(string)* , joy_btn_released *(string)* , joy_l_pressed *(bool)* , joy_l_released *(boo*

Sie basieren auf PyQt's *Signal* (`QtCore.pyqtSignal()`)

Slots Slots können im Qt Designer-Editor oder in Python-Code verbunden werden:

set_colorStateTrue() , set_colorStateFalse() , set_colorState(_bool_) , *set_true_color(*str*)* , se

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

A file path or resource path to an image to display in the described button location.
If the reset button is pressed in Qt Designer editor property, the image will not be displayed (allowing optionally text).

left_text , right_text , center_text , top_text , bottom_text
A text string to be displayed in the described button location.
If left blank an image can be designated to be displayed.

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
Color selection for the button text.

button_font
Font selection for the button text.

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*, 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')
```

12.7.5 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.

12.7.5.1 IndicatedPushButtons

Diese Klasse **verändert das Verhalten von `QPushButton`**.

`indicator_option` *setzt eine* LED *auf die Oberseite der Taste*.

Abbildung 12.102: QtVCP *PushButton*: Angezeigte Aktionstaste, LED-Anzeigeoption

Es kann ein *Dreieck*, *Kreis*, *obere Leiste* oder *seitliche Leiste* sein.
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):

`on_color` , **`off_color`** , **`indicator_size`** , **`circle_diameter`** , **`shape_option`** , **`right_edge_offset`** , **`top_`**

Indicator corner radius.

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*
- *Block Delete*
- *Optional Stop*
- *Manual*
- *MDI*
- *Auto*
- *Spindle Stopped*
- *Spindle Forward*
- *Spindle Reverse*
- *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;
}
```

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 , isManSensitive , isMDISen

Sie können mehrere Eigenschaften für kombinierte Anforderungen auswählen.

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”;
}
```

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')`

12.7.6 Import-Only 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.

12.7.6.1 Automatische Höhe

Widget zur Messung von zwei Höhen mit einer Sonde.
Für die Einrichtung.

12.7.6.2 G-Code Dienstprogramm

Widgets for performing common machining processes.

12.7.6.3 Facing

Slab or face a definable area with different strategies.

12.7.6.4 Loch-Kreis (engl. hole circle)

Drill multiple holes on a bolt hole circle.

12.7.6.5 Qt NGCGUI

QtVCP's Version des NGC Unterprogramm-Selektors.

12.7.6.6 Qt PDF

Ermöglicht das Hinzufügen von ladbaren PDFs zu einem Bildschirm.

12.7.6.7 Qt Vismach

Verwenden Sie dies, um OpenGl simulierte Maschinen zu erstellen/hinzuzufügen.

12.8 QtVCP Libraries modules

Libraries are **prebuilt Python modules that give added features to QtVCP**.
In this way you can select what features you want - yet don't have to build common ones yourself.

12.8.1 Status

Status is a library that **sends GObject messages based on LinuxCNC's current state**.
It is an *extension of GladeVCP's GStat object*.

It also has some functions to report status on such things as internal jog rate.

You *connect a function* call to the STATUS message you are interested in, and QtVCP will call this function when the message is sent from STATUS.

12.8.1.1 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()
```

- **Connect to STATUS messages**
  Use *GObject syntax*.

12.8.1.2 Beispiel

So können Sie z. B. Ein- und Ausschaltmeldungen der Maschine auffangen.

Anmerkung
The example below shows the *two common ways of connecting signals*, one of them *using lambda*. **`lambda`** is *used to strip off or manipulate arguments* from the status message before calling the function.
You can see the difference in the called function signature: the one that uses lambda does not accept the status object - lambda did not pass it to the function.

- Place these commands into the `[INITIALIZE]` section of the Python handler file:

```
STATUS.connect('state-on', self.on_state_on)
STATUS.connect('state-off', lambda: w, self.on_state_off())
```

  In this example code, when LinuxCNC is in *”machine on” state* the function `self.on_state_on` will be called.
  When LinuxCNC is in *”machine off” state* the function `self.on_state_off` will be called.

- These would call functions that looks like these:

```
def on_state_on(self, status_object):
    print('LinuxCNC machine is on')
def on_state_off(self):
    print('LinuxCNC machine is off')
```

12.8.2 `Info`

`Info` is a library to **collect and filter data from the INI file**.

12.8.2.1 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 = int(self.INI.find("KINS","JOINTS")or 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.find(section, "TYPE") or "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 = int(self.INI.find("TRAJ", "SPINDLES") or 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
```

12.8.2.2 User message 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")) or None
```

12.8.2.3 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)
```

12.8.2.4 Helpers

There are some *helper functions* - mostly used for widget support:

get_error_safe_setting(_self_, _heading_, _detail_, default=_None_), convert_metric_to_mac

Get filter extensions in Qt format.

12.8.2.5 Anwendung

- **Import `Info` module**
  Add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Info
```

- **Instantiate `Info` module**
  Add this Python code to your instantiate section:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

INFO = Info()
```

- **Access `INFO` data** Use this general syntax:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.8.3 Action

Action library is used to **command LinuxCNC's motion controller**.

It tries to hide incidental details and add convenience methods for developers.

12.8.3.1 Helpers

There are some **helper functions**, mostly used for this library's support:

get_jog_info (_num_) , jnum_check(_num_) , ensure_mode(_modes_) , open_filter_program(_filena

Open G-code filter program.

12.8.3.2 Anwendung

- **Import Action module**
  Add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Action
```

- **Instantiate Action module**
  Add this Python code to your instantiate section:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

ACTION = Action()
```

- **Access ACTION commands**
  Use general syntax such as these:

```
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.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)
```

12.8.4 Tool

This library **handles tool offset file changes**.

Warnung
LinuxCNC doesn’t handle third party manipulation of the tool file well.

12.8.4.1 Helpers

GET_TOOL_INFO(_toolnumber_)
This will return a Python **list of information on the requested tool number**.

GET_TOOL_ARRAY()
This return a single Python **list of Python lists of tool information**.
This is a raw list formed *from the system tool file*.

ADD_TOOL(_newtool_ = [_-99, 0,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’, 0,’New Too

This will return a Python **tuple of two Python lists of Python lists of tool information**:

- **[0]** will be *real tools informations*
- **[1]** will be *wear tools informations* (tool numbers will be over 10000; Fanuc style tool wear)

By default, adds a blanck tool entry with tool number -99.
You can preload the `newtool` array with tool information.

DELETE_TOOLS(_toolnumber_)
Delete the numbered tool.

SAVE_TOOLFILE(_toolarray_)
This will **parse the toolarray and save it to the tool file** specified in the *INI file* as the tool path.

This tool *array must contain all the available tools information*.

This array is expected to use the LinuxCNC *raw tool array*, i.e. doesn't have tool wear entries.

It will return True if there was an error.

CONVERT_TO_WEAR_TYPE(_toolarray_)
This function **converts a LinuxCNC raw tool array to a QtVCP tool array**.

QtVCP's tool array includes entries for X and Z axis tool wear.

LinuxCNC supports tool wear by adding **tool wear information into tool entries above 10000**.

Anmerkung
This also **requires remap code to add the wear offsets at tool change time**.

CONVERT_TO_STANDARD_TYPE(_toolarray_)
This function **converts QtVCP's tool array into a LinuxCNC raw tool array**.

QtVCP's array includes entries for X and Z axis tool wear.

LinuxCNC supports tool wear by adding **tool wear information into tool entries above 10000**.

Anmerkung
This also **requires remap code to add the wear offsets t tool change time**.

12.8.5 Path

Path module gives **reference to important files paths**.

12.8.5.1 Referenced Paths

PATH.PREFS_FILENAME
The preference file path.

PATH.WORKINGDIR
The directory QtVCP was launched from.

PATH.IS_SCREEN
Is this a screen or a VCP ?

PATH.CONFIGPATH
Launched configuration folder.

PATH.RIPCONFIGDIR
The Run-in-place config folder for QtVCP screens.

PATH.BASEDIR
Base folder for LinuxCNC.

PATH.BASENAME
The Qt Designer files name (no ending).

PATH.IMAGEDIR
The QtVCP image folder.

PATH.SCREENDIR
The QtVCP builtin Screen folder.

PATH.PANELDIR
The QtVCP builtin VCP folder.

PATH.HANDLER
Handler file Path.

PATH.HANDLERDIR
Directory where the Python handler file was found.

PATH.XML
QtVCP UI file path.

PATH.HANDLERDIR
Directory where the UI file was found.

PATH.QSS
QtVCP QSS file path.

PATH.PYDIR
LinuxCNC's Python library.

PATH.LIBDIR
The QtVCP library folder.

PATH.WIDGET
The QtVCP widget folder.

PATH.PLUGIN
The QtVCP widget plugin folder.

PATH.VISMACHDIR
Directory where prebuilt Vismach files are found.

Not currently used:

PATH.LOCALEDIR
Locale translation folder.

PATH.DOMAIN
Translation domain.

12.8.5.2 Helpers

Es sind einige Hilfsfunktionen verfügbar:

```
file_list = PATH.find_vismach_files()
directory_list = PATH.find_screen_dirs()
directory_list = PATH.find_panel_dirs()
```

12.8.5.3 Anwendung

- **Import `Path` module**
  Add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Path
```

- **Instantiate `Path` module**
  Add this Python code to your instantiate section:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

PATH = Path()
```

12.8.6 `VCPWindow`

`VCPWindow` module gives **reference to the `MainWindow` and widgets**.

Typically this would be used for a library (eg, toolbar library uses it) as the widgets get a reference to the `MainWindow` from the `_hal_init()` function.

12.8.6.1 Anwendung

- **Import `VCPWindow` module**
  Add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **Instantiate `VCPWindow` module**+ Add this Python code to your instantiate section:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

WIDGETS = VCPWindow()
```

12.8.7 `Aux_program_loader`

`Aux_program_loader` module allows an easy way to **load auxiliary programs LinuxCNC often uses**.

12.8.7.1 Helpers

load_halmeter()
Halmeter is used to **display one HAL pin data**.
Load a `halmeter` with:

```
AUX_PRGM.load_halmeter()
```

load_ladder()
Load *ClassicLadder* PLC program:

```
AUX_PRGM.load_ladder()
```

load_status()
Load LinuxCNC `status` program:

```
AUX_PRGM.load_status()
```

load_halshow()
Load *HALshow*, configure display program:

```
AUX_PRGM.load_halshow()
```

load_halscope()
Load *HALscope* program:

```
AUX_PRGM.load_halscope()
```

load_tooledit()
Load *Tooledit* program:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

load_calibration()
Load *Calibration* program:

```
AUX_PRGM.load_calibration()
```

keyboard_onboard()
Load *onboard/Matchbox keyboard*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

12.8.7.2 Anwendung

- **Import `Aux_program_loader` module**
  Add this Python code to your import section:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Instantiate `Aux_program_loader` module**
  Add this Python code to your instantiate section:

```
##########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
##########################################

AUX_PRGM = Aux_program_loader()
```

12.8.8 Keylookup

`Keylookup` module is used to **allow keypresses to control behaviors** such as jogging.

It's used inside the handler file to facilitate creation of **key bindings** such as keyboard jogging etc.

12.8.8.1 Anwendung

Import Keylookup module Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.keybindings import Keylookup
```

Instantiate Keylookup module To instantiate `Keylookup` module* so you can use it, add this Python code to your instantiate section:

```
##########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
##########################################

KEYBIND = Keylookup()
```

Anmerkung
`Keylookup` requires code under the `processed_key_event` function to call `KEYBIND.call()`.
Most handler files already have this code.

In the handler file, under the *initialized function* use this general syntax to **create keybindings**:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Here we add a keybinding for `F10`, `F11` and `F12`:

```
########################################
# 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):
    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)
```

And then we need to **add the functions that get called**.
In the handler file, under the `KEY BINDING CALLS` section, add this:

```
####################
# 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))
```

12.8.9 Messages

`Messages` module is used to **display pop up dialog messages on the screen**.

These messages are:

- *defined in the INI file under the `[DISPLAY]` heading*, and
- *controlled by HAL pins*.

12.8.9.1 Eigenschaften

`_BOLDTEXT`
: Generally is a title.

`_TEXT`
: Text below title, and usually longer.

`_DETAIL`
: Text hidden unless clicked on.

`_PINNAME`
: Basename of the HAL pin(s).

`_TYPE`
: Specifies whether it is a:

  - **Status message** - shown in the *status bar and the notify dialog*.
    Requires no user intervention.
  - **OK message** - *requiring the user to click OK to close the dialog*.
    OK messages have *two HAL pins*:
    - One HAL pin to launch the dialog, and
    - One to signify it's waiting for response.
  - **Yes/No message** - *requiring the user to select yes or no buttons to close the dialog*.
    Yes/No messages have *three hal pins*:
    - Eine, um den Dialog anzuzeigen,
    - Eine für das Warten, und
    - eine für die Antwort.

Standardmäßig sendet es `STATUS`-Nachrichten für `focus_overlay` und einen Warnton.

12.8.9.2 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.

12.8.10 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.

12.8.10.1 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.

12.8.11 Preferences

Das Modul **Preferences** ermöglicht das **Laden und Speichern von Einstellungsdaten dauerhaft auf Speichermedien** zu speichern.

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.

12.8.12 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.

12.8.12.1 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

12.8.12.2 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()
```

  The **_register_messages()** function connects the audio player to the STATUS library so sounds can be played with the STATUS message system.

12.8.12.3 Beispiel

To play sounds upon STATUS messages, use these general syntaxes:

```
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')
```

12.8.13 Virtuelle Tastatur

Diese Bibliothek ermöglicht es Ihnen, **mit STATUS-Nachrichten eine virtuelle Tastatur zu starten**.
It uses Onboard or Matchbox libraries for the keyboard.

12.8.14 Toolbar Actions

Diese Bibliothek liefert **vorgefertigte Untermenüs und Aktionen für Symbolleistenmenüs und Symbolleistenschaltflächen**.
Toolbuttons, menu and toolbar menus are:

- *built in Qt Designer*, and
- *assigned actions/submenus in the handler file*.

12.8.14.1 Actions

estop , power , load , reload , gcode_properties , run , pause , abort , block_delete , optional_stop

Toggles dimensions display.

12.8.14.2 Submenus

recent_submenu , home_submenu , unhome_submenu , zero_systems_submenu , grid_size_submenu

Menu to set graphic grid size

12.8.14.3 Anwendung

Here is the typical code to add to the relevant *handler file* sections:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

########################################
**** Bibliotheken instanziieren Abschnitt **** #
########################################

TOOLBAR = ToolBarActions()
```

12.8.14.4 Beispiele

- Assigning Tool Actions To Toolbar Buttons

```
########################################
# 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):
    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))
```

12.8.15 Qt Vismach Machine Graphics library

`Qt_vismach` is a *set of Python functions* that can be **used to create and animate models of machines**.

Vismach:

- *displays the model* in a **3D viewport**
- *animates the model parts* as the values of associated HAL pins change.

This is the *Qt based version* of the library, there is also a tkinter version available in LinuxCNC.

The Qt version *allows embedding the simulation in other screens*.

12.8.15.1 Integrierte Beispiele

There are included *sample panels* in QtVCP for:

- a 3-Axis XYZ mill,
- a 5-Axis gantry mill,
- a 3-Axis mill with an A axis/spindle, and
- a scara mill.

Most of these samples, if loaded after a running LinuxCNC configuration (including non-QtVCP based screens), will react to machine movement.
Some require HAL pins to be connected for movement.

From a terminal (pick one):

```
qtvcp vismach_mill_xyz
qtvcp vismach_scara
qtvcp vismach_millturn
qtvcp vismach_5axis_gantry
```

12.8.15.2 Primitives-Bibliothek

Provides the **basic building blocks of a simulated machine**.

`Collection`
: A `collection` is an **object of individual machine parts**.

  Diese enthält eine **hierarchische Liste** von primitiven Formen oder *STL-Objekten*, auf die Operationen angewendet werden können.

`Translate`
: This object will perform an **OpenGL translation** calculation *on a Collection object*.

  Translation refers to *moving an object in straight line* to a different position on screen.

`Scale`
: This object will perform an **OpenGL scale** function *on a collection object*.

`HalTranslate`
: This object will perform an **OpenGL translation** calculation *on a Collection object*, **offset by the HAL pin value**.

  Translation refers to moving an object in straight line to a different position on screen.

  You can either:

- *read a pin from a component owned by the Vismach object*, or
- *direktes Lesen eines HAL-Systempins*, wenn das Komponentenargument auf `None` gesetzt ist.

`Rotate`
This object will perform an **OpenGL rotation** calculation *on a Collection object*.

`HalRotate`
This object will perform an **OpenGL rotation** calculation *on a Collection object*, **offset by the HAL pin value**.

You can either:

- *read a pin from a component owned by the vismach* object, or
- *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

It reads the `halui.tool.diameter` and `motion.tooloffset.z` *HAL pins*.

Example from mill_xyz sample:

```
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`
Bewege und drehe ein Objekt, um von einer `capture()`-Koordinate aus zu zeigen System zu einem anderen.

Base object to *hold coordinates for primitive shapes*.

`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`
Baue eine Kugel aus den Koordinaten *center* und *radius*.

`TriangleXY, TriangleXZ, TriangleYZ`
Baue ein Dreieck in der *angegebenen Ebene* durch Angabe der Eckpunkte Z-Koordinaten_ für jede Seite

`ArcX`
Bogen erstellen durch Angabe

`Box`
Erstellen Sie eine Box, die durch den coordinates_ _6-Scheitelpunkt angegeben wird.

`BoxCentered`
Build a box centered on origin by specifying the *width in X and Y*, and the *height in Z*.

`BoxCenteredXY`

- Erstellen Sie eine Box, die in X und Y zentriert ist und von Z = 0 * aus ausgeführt wird, indem Sie Folgendes angeben die *width in X und Y* und läuft nach oben oder unten auf die angegebene *height in Z*.

`Capture`
Capture current transformation matrix of a collection.

Anmerkung
Dies *transformiert vom aktuellen Koordinatensystem in das Ansichtsfenstersystem*, NICHT in das Weltsystem.

`Hud`
Heads up display draws a *semi-transparent text box*.

Use:

- `HUD.strs` for things that must be *updated constantly*,
- `HUD.show("stuff")` for one-shot things like error messages.

`Color`
Wendet eine Farbe auf die *Teile einer Collection* **an**.

`AsciiSTL, AsciiOBJ`
Lädt eine STL- oder OBJ-Datendatei als *Vismach part*.

12.8.15.3 Anwendung

Import a simulation Here is how one might import the XYZ_mill simulation in a QtVCP panel or screen handler file.

```
############################
# **** IMPORT SECTION **** #
############################

import mill_xyz as MILL
```

Instantiate and use the simulation widget Instanziieren Sie das Simulations-Widget und fügen Sie es dem Hauptlayout des Bildschirms hinzu:

```
    ########################################
    # Special Functions called from QtVCP
    ########################################

    # At this point:
    #   * the widgets are instantiated,
    #   * the HAL pins are built but HAL is not set ready.
    def initialized__(self):
        machine = MILL.Window()
        self.w.mainLayout.addWidget(machine)
```

12.8.15.4 Mehr zum Thema

More information on how to build a custom machine simulation in the Qt Vismach chapter.

12.9 QtVismach

Vismach is a set of **Python functions that can be used to create and animate models of machines**.

This chapter is about the Qt embedded version of Vismach.

12.9.1 Einführung

Vismach displays the model in a **3D viewport** and the **model parts are animated as the values of associated HAL pins change**.

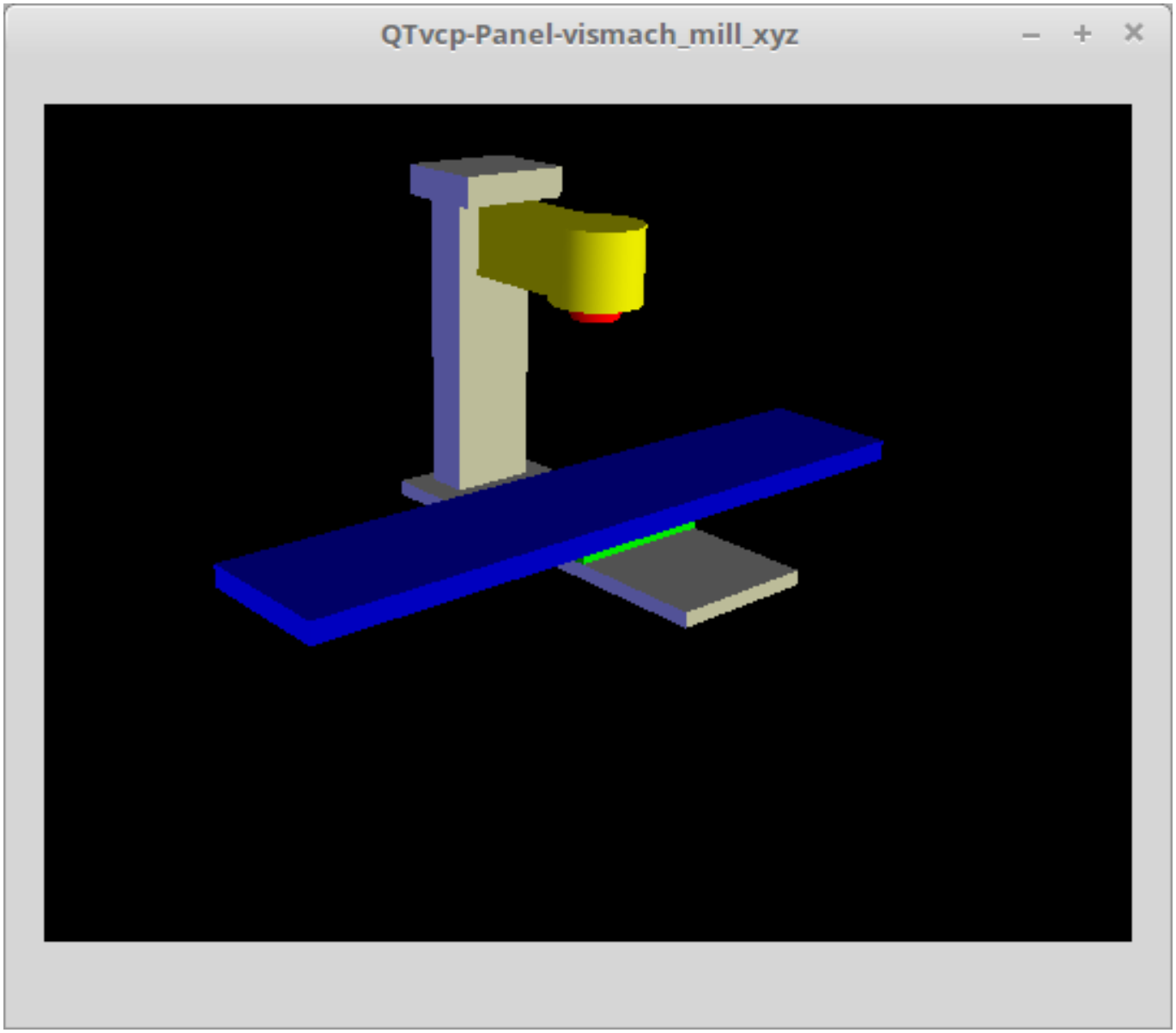


Abbildung 12.103: QtVismach 3D Viewport

The Vismach 3D viewport view can be manipulated as follows:

- **zoom** by *scroll wheel*
- **pan** by *middle button drag*
- **rotate** by *right-button drag*
- **tilt** by *left button drag*

A **Vismach model** takes the form of a *Python script* and can use standard Python syntax.

This means that there is more than one way to lay out the script, but in the examples given in this document the simplest and most basic of them will be used.

The basic sequence in creating the Vismach model is:

1. Create the parts
2. Define how they move
3. Assemble into movement groups

12.9.2 Hierarchy of Machine Design

The model follows **logical tree design**.

Picture the tree, with root/trunk, branches and smaller branches off it. If you move the larger branch, smaller branches will move with it, but if you move smaller branch larger will not.

Machine design follows that conceptual design.

Taking the mill shown in the 3D Viewport picture above for example:

- if you move X, it can move on its own,
- but if you move Y, it will also move X assembly, as it is attached to Y assembly.

So for this machine, tree looks like this:

```
model
  |
  |---frame
  |     |
  |     |---base
  |     |
  |     |---column
  |     |
  |     |---top
  |
  |---yassembly
  |       |
  |       |---xassembly
  |       |       |
  |       |       |---xbase
  |       |       |
  |       |       |---work
  |       |
  |       |---ybase
  |
  |---zassembly
           |
           |---zframe
           |     |
```

```
          |     |---zbody
          |     |
          |     |---spindle
          |
          |---toolassembly
                    |
                    |---cat30
                    |
                    |---tool
                         |
                         |---tooltip
                         |
                         |---(tool cylinder function)
```

As you can see, *lowest parts must exist first before it can be grouped with others into assembly*. So you *build upwards* from lowest point in tree and assemble them together.

Same is applicable for any design of machine: look at machine arm example and you will see that it starts with tip and adds to larger part of arm then it finally groups with base.

12.9.3 Start the script

Zu Testzwecken ist es nützlich, das Shebang `#!/usr/bin/env python3` einzubinden, damit *die Datei als Skript ausgeführt werden kann*.

The first thing to do is to *import the required libraries*.

```
#!/usr/bin/env python3

import hal
import math
import sys

from qtvcp.lib.qt_vismach.qt_vismach import *
```

12.9.4 HAL pins.

Originally the vismach library required creating a component and connecting HAL pins to control the simulation.

`qt_vismach` can read the HAL system pins directly or if you wish, to use separate HAL pins that you must define in a HAL component:

```
c = hal.component("samplegui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

12.9.5 Erstellen von Teilen

12.9.5.1 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 *Rohdaten*:

```
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 ...")
```

Die Teile werden im Vismach-Raum an den *gleichen Stellen wie im STL- oder OBJ-Raum* erstellt, was bedeutet, dass es möglich sein kann, das Modell im CAD-Paket zusammenzusetzen.

12.9.5.2 Aufbau aus geometrischen Primitiven

Alternativ können Teile auch *im Modellskript aus einer Reihe von Formprimitiven* erstellt werden.

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 = Cyl**

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, z**

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)
Create an *arc shape*.

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])
```

12.9.6 Moving Model Parts

Möglicherweise müssen Teile im Vismach-Bereich bewegt werden, um das Modell zusammenzubauen. Möglicherweise müssen sie auch verschoben werden, um die Animation zu erstellen, da die Drehachse der Animation am Ursprung erstellt wird (aber sich mit dem Teil bewegt).

12.9.6.1 Translating Model parts

part1 = Translate([part1], x, y, z)
Verschiebe Teil1 um die angegebenen Abstände in x, y und z.

12.9.6.2 Rotating Model Parts

part1 = Rotate([part1], theta, x, y, z)
Rotate the part by angle theta about an axis between the origin and x, y, z.

12.9.7 Animating Parts

To **animate the model controlled by the values of HAL pins** there are two functions `HalTranslate`, `HalRotate` and `HalToolCylinder`.

For parts to move inside an assembly they need to have their HAL motions defined before being assembled with the "Collection" command.

The **rotation axis and translation vector move with the part**:

- as it is moved by the Vismach script during model assembly, or
- as it moves in response to the HAL pins as the model is animated.

12.9.7.1 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
The *HAL component* is the next argument.
In QtVCP if you are reading *system pins* directly then the component argument is set to `None`.

hal_pin
The *name of the HAL pin* that will animate the motion.
This needs to match an existing HAL pin that describes the joint position such as:

```
"joint.2.pos-fb"
```

Andernfalls würde die Komponenteninstanz und der Pin-Name dieser Komponente angegeben werden. `xs, ys, zs`;; The *X, Y, Z scales*.
For a Cartesian machine created at 1:1 scale this would typically be `1,0,0` for a motion in the positive X direction.
However if the STL file happened to be in cm and the machine was in inches, this could be fixed at this point by using 0.3937 (1cm /2.54in) as the scale.

12.9.7.2 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
Defines the *axis of rotation* from the origin the point of coordinates (x,y,z).
When the part is moved back away from the origin to its correct location the axis of rotation can be considered to remain ”embedded” in the part.

angle_scale
Drehwinkel werden in Grad angegeben. Für ein Drehgelenk mit einer Skalierung von 0-1 müssten Sie also eine Winkelskala von 360 verwenden.

12.9.8 Assembling the model

In order for parts to move together they need to be assembled with the **`Collection()` command**.

It is important to **assemble the parts and define their motions in the correct sequence**.

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.
- Create the draw bar
- Define the motion of the draw bar
- 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])
```

Finally a **single collection of all the machine parts, floor and work** (if any) needs to be created.

For a *serial machine* each new part will be added to the collection of the previous part.

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])
```

12.9.9 Weitere Funktionen

part = Color([_colorspec_], [_part_])
Setzt die *Anzeigefarbe des Teils*.
Beachten Sie, dass im Gegensatz zu den anderen Funktionen, die Definition des Teils in diesem Fall an zweiter Stelle steht.

colorspec
Drei RGB-Werte und Deckkraft.
Zum Beispiel [1,0,0,0.5] für ein Rot mit 50% Deckkraft.

myhud = Hud() , myhud.show("_Mill_XYZ_")
Erzeugt ein *Heads-up-Display* in der Vismach-GUI, um Elemente wie Achsenpositionen, Titel oder Meldungen anzuzeigen.

part = Capture()
This sets the current position in the model.

main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)
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.

model
Should be a collection that contains all the machine parts.

tooltip und work
Need to be created by `Capture()` to visualize their motion in the backplot.
See `mill_xyz.py` for an example of how to connect the tool tip to a tool and the tool to the model.

size
Legt die Ausdehnung des Volumens fest, das in der Ausgangsansicht angezeigt wird.
"hud" bezieht sich auf eine Head-up-Anzeige der Achsenpositionen.

"rotation_vectors" oder "lat, lon"
Can be used to set the original viewpoint.
It is advisable to do as the default initial viewpoint is rather unhelpful from immediately overhead.

12.9.10 Basic structure of a QtVismach script

```
# imports
import hal
from qtvcp.lib.qt_vismach.qt_vismach import *

# hier HAL-Pins erstellen, falls erforderlich
#c = hal.component("samplegui")
#c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_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

        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)
```

12.9.11 Builtin Vismach Sample Panels

QtVCP builtin Vismach Panels

12.10 QtVCP: Building Custom Widgets

12.10.1 Übersicht

Building custom widgets allows one to **use the Qt Designer editor to place a custom widget** *rather than doing it manually in a handler file*.

A useful custom widgets would be a great way to contribute back to LinuxCNC.

12.10.1.1 Widgets

Widget is the *general name for the UI objects* such as buttons and labels in PyQt.

There are also **special widgets made for LinuxCNC** that make integration easier.

All these widgets can be *placed with Qt Designer editor* - allowing one to *see the result* before actually loading the panel in LinuxCNC.

12.10.1.2 Qt Designer

Qt Designer is a *WYSIWYG (What You See is What You Get) editor for placing PyQt widgets*.

It's original intend was for building the graphic widgets for programs.
We leverage it to **build screens and panels for LinuxCNC**.

In Qt Designer, on the left side of the editor, you find **three categories of LinuxCNC widgets**:

- *HAL only widgets*.
- *LinuxCNC-Controller-Widgets*.
- *dialog widgets*.

For Qt Designer to *add custom widgets* to it's editor it must have a **plugin** added to the right folder.

12.10.1.3 Initialization Process

QtVCP does *extra setup* for **widgets subclassed from `_HALWidgetBase`**, aka "HAL-ified" widgets.

This includes:

- Injecting *important variables*,
- Calling an *extra setup function*
- Calling a *closing cleanup function* at shutdown.

These functions are not called when the Qt Designer editor displays the widgets.

Wenn QtVCP einen Bildschirm aus der *.ui*-Datei erstellt:

1. It searches for all the HAL-ified widgets.

2. It finds the `ScreenOptions` widget, to collect information it needs to inject into the other widgets
3. It instantiates each widget and if it is a HAL-ified widget, calls the `hal_init()` function.
   `hal_init()` is defined in the base class and it:
   a. Adds variables such as the preference file to every HAL-ified widget.
   b. Call `+_hal_init()+` on the widget.
      `+_hal_init()+` allows the widget designer to do setup that requires access to the extra variables.

Here is a description of the extra variables injected into ”HAL-ified” widgets:

`self.HAL_GCOMP`
The *HAL component instance*

`self.HAL_NAME`
This *widget's name* as a string

`self.QT_OBJECT_`
This *widget's object instance*

`self.QTVCP_INSTANCE_`
The *very top level parent* of the screen

`self.PATHS_`
The *QtVCP's path library instance*

`self.PREFS_`
The *optional preference file instance*

`self.SETTINGS_`
The `Qsettings` *object instance*

12.10.1.4 cleanup process

When QtVCP closes, it calls the **`+_hal_cleanup()+`** function *on all HAL-ified widgets*.

The base class creates an empty `+_hal_cleanup()+` function, which can be redefined in the custom widget subclass.

This can be used to do such things as record preferences, etc.

This function is not called when the Qt Designer editor displays the widgets.

12.10.2 Custom HAL Widgets

HAL widgets are the simplest to show example of.
`qtvcp/widgets/simple_widgets.py` holds many HAL only widgets.

Lets look at a snippet of `simple_widgets.py`:

Im Abschnitt ”Imports” This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

################################
# Imports
################################
from PyQt5 import QtWidgets                     # ❶
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase    # ❷
import hal                                      # ❸
```

In this case we need access to:

1. PyQt's QtWidgets library,
2. LinuxCNC's HAL library, and
3. QtVCP's widget `baseclass` 's **`_HalSensitiveBase`** for *automatic HAL pin setup* and to *disable/enable the widget* (also known as input sensitivity).
   There is also `_HalToggleBase`, and `_HalScaleBase` functions available in the library.`_HalToggleBase`, and `_HalScaleBase`

Im Abschnitt WIDGET Here is a *custom widget* based on PyQt's **`QGridLayout`** widget.

`QGridLayout` allows one to:

- *Place objects in a grid* fashion.
- *Enable/disable all widgets inside it* based on a **HAL pin state**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):     # 1
    def __init__(self, parent = None):                           # 2
        super(GridLayout, self).__init__(parent)                 # 3
```

Line by Line:

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 BIT pin.
2. This is the function *called when the widget is first made* (said instantiated) - this is pretty standard.
3. This function initializes our widget's **`Super` classes**.
   `Super` just means the *inherited baseclasses*, that is `QWidget` and `_HalSensitiveBase`.
   Pretty standard other than the widget name will change.

12.10.3 Custom Controller Widgets Using `STATUS`

Widget that interact with LinuxCNC's controller are only a little more complicated and they require some *extra libraries*.

In this cut down example we will add properties that can be changed in Qt Designer.

This LED indicator widget will respond to selectable LinuxCNC controller states.

```
#!/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)
```

12.10.3.1 In The *Imports* Section

This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5.QtCore import pyqtProperty       # ❶
from qtvcp.widgets.led_widget import LED    # ❷
from qtvcp.core import Status               # ❸
```

We import

❶ `pyqtProperty` so we can interact with the Qt Designer editor,

❷ `LED` because our custom widget is based on it,

❸ `Status` because it gives us status messages from LinuxCNC.

12.10.3.2 In The *Instantiate Libraries* Section

Here we create the `Status` library instance:

```
########################################
# **** instantiate libraries section **** #
########################################
STATUS = Status()
```

Typically we instantiated the library *outside of the widget class* so that the reference to it is **global** - meaning you don't need to use `self.` in front of it.

By convention we use *all capital* letters in the name for global references.

12.10.3.3 In The *Custom Widget Class Definition* Section

This is the meat and potatoes of our custom widget.

Class definition and instance initialization function

```
class StateLed(LED):                            # ❶
    def __init__(self, parent=None):            # ❷
        super(StateLed, self).__init__(parent)  # ❸
        self.has_hal_pins = False               # ❹
        self.setState(False)                    # ❺
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False
```

1. Defines the **name** *of our custom widget* and what other *class it inherits from*.
   In this case we inherit `LED` - a QtVCP widget that represents a status light.
2. Typical of most widgets - called when the widget is first made.
3. Typical of most widgets - calls the parent (super) widget initialization code.
   Then we set some attributes:
4. Inherited from `Lcnc_Led` - we set it here so no HAL pin is made.
5. Inherited from `Lcnc_led` - we set it to make sure the LED is off.

The other attributes are for the selectable options of our widget.

Widget's HAL initialization function

```
    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))
```

This function connects `STATUS` (LinuxCNC status message library) to our widget so that the LED will on or off based on the selected state of the controller.

We have two states we can choose from `is_estopped` or `is_on`.
Depending on which is active our widget get connected to the appropriate STATUS messages.

`+_hal_init()+` is *called on each widget that inherits* `+_HalWidgetBase+`, when QtVCP first builds the screen.
You might wonder why it's called on this widget since we didn't have `+_HalWidgetBase+` in our class definition (`class Lcnc_State_Led(Lcnc_Led):`) - it's called because `Lcnc_Led` inherits `+_HalWidgetBase+`.

In this function you have access to some extra information (though we don't use them in this example):

`self.HAL_GCOMP`
: the *HAL component* instance

`self.HAL_NAME`
: This *widget's name* as a string

`self.QT_OBJECT_`
: dieses *Widgets PyQt-Objekt als Instanz*

`self.QTVCP_INSTANCE_`
: The *very top level parent* of the screen

`self.PATHS_`
: The *instance of QtVCP's path* library

`self.PREFS_`
: die *Instanz einer optionalen Präferenzdatei*

`self.SETTINGS_`
: das `Qsettings` *Objekt*

We could use this information to create HAL pins or look up image paths etc.

```
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Lets look at this line more closely:

- `STATUS` is very common theme is widget building.
  `STATUS` uses `GObject` message system to send messages to widgets that register to it.
  This line is the registering process.
- `state-estop` is the message we wish to listen for and act on. There are many messages available.
- `lambda w:self._flip_state(True)` is what happens when the message is caught.
  The lambda function accepts the widget instance (`w`) that GObject sends it and then calls the function `self._flip_state(True)`.
  Lambda was used to strip the (`w`) object before calling the `self._flip_state` function.
  It also allowed use to send `self._flip_state()` the True state.

```
    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

This is the function that actually flips the state of the LED.
It is what gets called when the appropriate STATUS message is accepted.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

The function called looks like this:

```
    def _set_feedrate_text(self, widget, data):
```

in which the widget and any data must be accepted by the function.

```
    ########################################################################
    # 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
```

This is *how Qt Designer sets the attributes of the widget*.
This *can also be called directly* in the widget.

```
    ######################################
    # 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)
```

This is the **registering of properties in Qt Designer**.

The **property name**:

- is the *text used in Qt Designer,*
- *cannot be the same as the attributes* they represent.

These properties show in Qt Designer in the order they appear here.

12.10.4 Custom Controller Widgets with Actions

Here is an example of a widget that sets the user reference system.

It changes:

- the machine controller state using the `ACTION` library,
- whether the button can be clicked or not using the `STATUS` library.

```
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

# Instantiate the libraries with global reference
# STATUS gives us status messages from LinuxCNC
# INFO holds INI details
# ACTION gives commands to LinuxCNC
STATUS = Status()
INFO = Info()
ACTION = Action()

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)
```

12.10.5 Stylesheet Property Changes Based On Events

It's possible to **have widgets restyled when events change**. You must explicitly *"polish" the widget* to have PyQt redo the style.
This is a relatively expensive function so should be used sparingly.

This example sets an `isHomed` property based on LinuxCNC's homed state and in turn uses it to change stylesheet properties:

This example will set the property isHomed based on LinuxCNC's homed state.

```
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))

    # update ishomed property
    # polish widget so stylesheet sees the property change
    # some stylesheets color the text on 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)
```

Here is a sample stylesheet to change text color based on home state.

In this case any widget based on the HomeLabel widget above will change text color.
You would usually pick specific widgets using `HomeLabel #specific_widget_name[homed=true]`:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

12.10.6 Use Stylesheets To Change Custom Widget Properties

```
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)
```

Sample stylesheet that sets a custom widget property.

```
Label{
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
}
```

12.10.7 Widget Plugins

We must *register our custom widget* for Qt Designer to use them.

Here are a typical samples.
They would need to be added to `qtvcp/plugins/`
Then `qtvcp/plugins/qtvcp_plugin.py` would need to be adjusted to *import* them.

12.10.7.1 Gridlayout Example

```
#!/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"
```

12.10.7.2 SystemToolbutton Example

```
#!/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"
```

12.10.7.3 Making a plugin with a MenuEntry dialog box

It possible to add an entry to the dialog that pops up when you right click the widget in the layout.

This can do things such as selecting options in a more convenient way.

This is the plugin used for *action buttons*.

```
#!/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):

    # The __init__() method is only used to set up the plugin and define its
    # initialized variable.
    def __init__(self, parent=None):
        super(ActionButtonPlugin, self).__init__(parent)
        self.initialized = False

    # The initialize() and isInitialized() methods allow the plugin to set up
    # any required resources, ensuring that this can only happen once for each
    # plugin.
    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

    # This factory method creates new instances of our custom widget
    def createWidget(self, parent):
        return ActionButton(parent)

    # This method returns the name of the custom widget class
    def name(self):
        return "ActionButton"

    # Returns the name of the group in Qt Designer's widget box
```

```
    def group(self):
        return "LinuxCNC - Controller"

    # Returns the icon
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('actionbutton')))

    # Returns a tool tip short description
    def toolTip(self):
        return "Action button widget"

    # Returns a short description of the custom widget for use in a "What's
    # This?" help message for the widget.
    def whatsThis(self):
        return ""

    # Returns True if the custom widget acts as a container for other widgets;
    def isContainer(self):
        return False

    # Returns an XML description of a custom widget instance that describes
    # default values for its properties.
    def domXml(self):
        return '<widget class="ActionButton" name="actionbutton" />\n'

    # Returns the module containing the custom widget class. It may include
    # a module path.
    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
```

12.11 QtVCP Handler File Code Snippets

12.11.1 Preference File Loading/Saving

Hier erfahren Sie, wie Sie **die Einstellungen beim Start und beim Beenden laden und speichern**.

Prerequisites

- *Preference file option must be set* in the `ScreenOptions` widget.
- *Preference file path must be set* in the `INI` configuration.

Reading preferences at launch time Under the **`def initialized__(self):`** function add:

```
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')
```

Writing preferences at close time In the **`closing_cleanup__()`** function, add:

```
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')
```

12.11.2 Use QSettings To Read/Save Variables

Here is how to **load and save variables using PyQt's QSettings** functions:

Good practices

- *Use Group to keep names organized and unique.*
- *Account for none value* returned when reading a setting which has no entry.
- *Set defaults to cover the first time it is run* using the `or _<default_value>_` syntax.

Anmerkung
The file is actually saved in `~/.config/QtVcp`

Beispiel In diesem Beispiel:

- We add `or 20` and `or 2.5` as defaults.
- The names `MyGroupName`, `int_value`, `float_value`, `myInteger`, and `myFloat` are user defined.
- Under the **`def initialized__(self):`** function add:

```
# 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()
```

- Under the **`def closing_cleanup__(self):`** function add:

```
# 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()
```

12.11.3 Add A Basic Style Editor

Being able to **edit a style on a running screen** is convenient.

Import StyleSheetEditor module in the IMPORT SECTION:

```
from qtvcp.widgets.stylesheeteditor import StyleSheetEditor as SSE
```

Instantiate StyleSheetEditor module in the INSTANTIATE SECTION:

```
STYLEEDITOR = SSE()
```

Create a keybinding in the INITIALIZE SECTION: Under the `+__init__.(self, halcomp, widgets, paths):+` function add:

```
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()
```

12.11.4 Dialog-Eintrag anfordern

QtVCP uses STATUS messages to **pop up and return information from dialogs**.

Prebuilt dialogs keep track of their last position and include options for focus shading and sound.

To *get information back from the dialog* requires using a STATUS **`general`** message.

Import and Instantiate the Status module in the IMPORT SECTION

```
from qtvcp.core import Status
STATUS = Status()
```

This loads and initializes the `Status` library.

Register function for STATUS general messages in the INITIALIZE SECTION Under the `+__init__.(halcomp, widgets, paths)+` function:

```
STATUS.connect('general',self.return_value)
```

This registers STATUS to call the function `self.return_value` when a general message is sent.

Add entry dialog request function in the GENERAL FUNCTIONS SECTION

```
def request_number(self):
    mess = {'NAME':'ENTRY','ID':'FORM__NUMBER', 'TITLE':'Set Tool Offset'}
    STATUS.emit('dialog-request', mess)
```

Die Funktion

- Creates a Python `dict` with:
  - **`NAME`** - needs to be set to the *dialogs unique launch name*.
    `NAME` sets which dialog to request.
    `ENTRY` or `CALCULATOR` allows entering numbers.
  - **`ID`** - needs to be set to a *unique name that the function supplies*.
    `ID` should be a unique key.
  - **`TITLE`** sets the dialog title.
  - **Arbitrary data** can be added to the `dict`.
    The dialog will ignore them but send them back to the return code.
- Sends the `dict` as a **`dialog-request`** STATUS message

Add message data processing function in the CALLBACKS FROM STATUS SECTION

```
# Process the STATUS return message from 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))
```

This catches all general messages so it must *check the dialog type and id code* to confirm it's our dialog.

In this case we had requested an `ENTRY` dialog and our unique id was `FORM_NUMBER`, so now we know the message is for us.

`ENTRY` or `CALCULATOR` dialogs return a float number.

12.11.5 Sprechen Sie eine Startup-Begrüßung

This requires the `espeak` library installed on the system.

Import and Instantiate the Status in the IMPORT SECTION

```
from qtvcp.core import Status
STATUS = Status()
```

Emit spoken message in the 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*.

12.11.6 ToolBar Functions

Toolbar buttons and submenus are added in Qt Designer but the code to make them do something is added in the handler file.

To **add a submenus** in Qt Designer:

- Add a `Qaction` by typing in the toolbar column then clicking the + icon on the right.
- This will add a sub column that you need to type a name into.
- Now the original `Qaction` will be a `Qmenu` instead.
- Now erase the `Qaction` you added to that `Qmenu`, the menu will stay as a menu.

In this example we assume you added a toolbar with one submenu and three actions. These actions will be configured to create:

- a recent file selection menu,
- an about pop up dialog action,
- a quit program action, and
- a user defined function action.

The `objectName` of the toolbar button is used to identify the button when configuring it - *descriptive names help*.

Using the action editor menu, right click and select edit.
Edit the object name, text, and button type for an appropriate action.

In this example the:

- submenu name must be `menuRecent`,
- actions names must be `actionAbout`, `actionQuit`, `actionMyFunction`

Loads the toolbar_actions library in the IMPORT SECTION

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

Instantiate ToolBarActions module in the INSTANTIATE LIBRARY SECTION

```
TOOLBAR = ToolBarActions()
```

Configure submenus and actions in the SPECIAL FUNCTIONS SECTION Under the `def initialized_` function add:

```
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)
```

Define the user function in the GENERAL FUNCTIONS SECTION

```
def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

The function to be called if the action "My Function" button is pressed.

12.11.7 Add HAL Pins That Call Functions

In this way you *don't need to poll the state of input pins*.

Loads the Qhal library in the IMPORT SECTION

```
from qtvcp.core import Qhal
```

This is to allow access to **QtVCP's HAL component**.

Instantiate Qhal in the INSTANTIATE LIBRARY SECTION

```
QHAL = Qhal()
```

Add a function that gets called when the pin state changes Under the `initialised__` function, make sure there is an entry similar to this:

```
########################################
# 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):
    self.pin_cycle_start_in = QHAL.newpin('cycle-start-in',QHAL.HAL_BIT, QHAL.HAL_IN)
    self.pin_cycle_start_in.value_changed.connect(lambda s: self.cycleStart(s))
```

Define the function called by pin state change in the GENERAL FUNCTIONS SECTION

```
#####################################
# 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*.

12.11.8 Add A Special Max Velocity Slider Based On Percent

Some times you want to **build a widget to do something not built in**.

The built in Max velocity slider acts on units per minute, here we show how to do on percent.

The **`STATUS`** command makes sure the slider adjusts if LinuxCNC changes the current max velocity.

`valueChanged.connect()` *calls a function when the slider is moved*.

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))
```

12.11.9 Toggle Continuous Jog On and Off

Generally selecting continuous jogging is a momentary button, that requires you to select the previous jog increment after.

We will build a button that toggles between continuous jog and whatever increment that was already selected.

In Qt Designer:

- Add an `ActionButton` with no action
- Call it `btn_toggle_continuous`.
- Set the `AbstractButton` property `checkable` to `True`.
- Set the `ActionButton` properties `incr_imperial_number` and `incr_mm_number` to `0`.
- Use Qt Designer's slot editor to use the button signal `clicked(bool)` to call form's handler function `toggle_continuous_clicked()`.
  See Using Qt Designer To Add Slots section for more informations.

Then add this code snippets to the handler file under the `initialized__` function:

```
# 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"
```

In the `GENERAL FUNCTIONS SECTION` add:

```
######################
# GENERAL FUNCTIONS #
######################

# if it isn't continuous, record the latest jog increment
# and untoggle the continuous button
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 FROM FORM #
########################

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)
```

12.11.10 Class Patch The File Manager Widget

Anmerkung
Class patching (monkey patching) is a little like *black magic* - so use it *only if needed*.

The File manager widget is designed to load a selected program in LinuxCNC. But maybe you want to print the file name first.

We can ”class patch” the library to *redirect the function call*.

In the `IMPORT SECTION` add:

```
from qtvcp.widgets.file_manager import FileManager as FM
```

Here we are going to:

1. *Keep a reference to the original function* (1) so we can still call it
2. *Redirect the class to call our custom function* (2) in the handler file instead.

```
########################################
# Special Functions called from QtVCP    #
########################################

# For changing functions in widgets we can 'class patch'.
# class patching must be done before the class is instantiated.
def class_patch__(self):
    self.old_load = FM.load # keep a reference of the old function (1)
    FM.load = self.our_load # redirect function to our handle file function (2)
```

3. *Write a custom function to replace the original*:
   This function must have the **same signature as the original function**.
   In this example we are still going to call the original function by using the reference to it we recorded earlier.
   It *requires the first argument to be the widget instance*, which in this case is `self.w.filemanager` (the name given in the Qt Designer editor).

```
####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

Now our custom function will print the file path to the terminal before loading the file.
Obviously boring but shows the principle.

Anmerkung
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)
```

12.11.11 Adding Widgets Programmatically

In some situation it is only possible to **add widgets with Python code** rather then using the Qt Designer editor.

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` gives us *information about the machine configuration*.

```
############################
# **** 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:

- *Catch* the `actual-spindle-speed-changed` *signal*
- *Aufruf* der _Funktion `self.update_spindle()`

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to  widgets from the QtVCP files
# at this point the widgets and hal pins are not instantiated
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__()`:

```
########################################
# Special Functions called from QtScreen #
########################################

# 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()
```

Create the widgets building functions 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)                                  # (10)
    w.setLayout(hbox)

    # den Container zur Ecke des rechten Tab-Widgets hinzufügen
    self.w.rightTab.setCornerWidget(w) # (11)
```

(1), (1) Dies initialisiert das grundlegende StateLed-Widget und verwendet von da an `self.w.led` als Referenz.

(2), (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) This is the extra function call required with some QtVCP widgets.
If `HAL_NAME` is omitted it will use the widget's `objectName` if there is one.
It gives the special widgets reference to:

`self.HAL_GCOMP`
: the *HAL component* instance

`self.HAL_NAME`
: This *widget's name* as a string

`self.QT_OBJECT_`
: dieses *Widgets PyQt-Objekt als Instanz*

`self.QTVCP_INSTANCE_`
: The *very top level parent* of the screen

`self.PATHS_`
: The *instance of QtVCP's path* library

`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. add a QHBoxLayout to the QWidget.

9, 10. Then we add our QProgress bar and LED to the layout.

11. Finally we add the QWidget (with our QProgress bar and LED in it) to the tab widget's corner.

Create the STATUS monitoring function Now we build the function to actually update out the `QProgressBar` when `STATUS` updates the spindle speed:

```
########################
# callbacks from 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 this case we chose to display left-to-right or right-to-left, depending if we are turning clockwise or anticlockwise.

2. This formats the writing in the bar.

3. This sets the length of the colored bar.

12.11.12 Update/Read Objects Periodically

Sometimes you need to **update a widget or read a value regularly** that isn't covered by normal libraries.

Here we update an LED based on a watched HAL pin every 100ms.

We assume there is an LED named `led` in the Qt Designer UI file.

Load the Qhal library for access to QtVCP's HAL component In the `IMPORT SECTION` add:

```
from qtvcp.core import Qhal
```

Instantiate Qhal In the `INSTANTIATE LIBRARY SECTION` add:

```
QHAL = Qhal()
```

Fügen Sie nun diese Abschnitte hinzu bzw. ändern Sie sie so, dass sie einen ähnlichen Code enthalten wie dieser:

Register a function to be called at CYCLE_TIME period This is usually every 100ms

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to widgets from the QtVCP files
# at this point the widgets and hal pins are not instantiated
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    # register a function to be called at CYCLE_TIME period (usually every 100ms)
    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)
```

12.11.13 External Control With ZMQ

QtVCP can automatically set up **ZMQ messaging** *to send and/or receive remote messages from external programs*.

It uses ZMQ's **publish/subscribe messaging pattern**.

As always, consider **security** before letting programs interface though messaging.

12.11.13.1 ZMQ Messages Reading

Sometimes you want to **control the screen with a separate program**.

Enable reception of ZMQ messages In the `ScreenOptions` widget, you can select the property **`use_receive_zmq_option`**.
You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to widgets from the QtVCP files
# at this point the widgets and hal pins are not instantiated
def __init__(self,halcomp,widgets,paths):
    # directly select ZMQ message receiving
    self.w.screen_options.setProperty('use_receive_zmq_option',True)
```

This **allows an external program to call functions in the handler file**.

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)
```

Create an external program sending ZMQ messages that will trigger function call Here is a sample external program to call a function.
It alternates between two data sets every second.
Run this in a separate terminal from LinuxCNC to see the sent messages.

```
#!/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
# makes a dict of function to call plus any arguments
x = {                                   # ❶
  "FUNCTION": "test_zmq_function",
  "ARGS": [True,200]
}
# convert to json object
m1 = json.dumps(x)

# prebuild message 2
x = {                                   # ❷
  "FUNCTION": "test_zmq_function",
  "ARGS": [False,0],
}
# convert to json object
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))
```

❶, ❷ Set the **function to call** and the **arguments to send** to that function.

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.

12.11.13.2 ZMQ Messages Writing

You may also want to **communicate with an external program from the screen**.

In the `ScreenOptions` widget, you can select the property **`use_send_zmq_message`**.
You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

Enable sending of ZMQ messages

```
########################
# **** 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)
```

This allows sending messages to a separate program.
The message sent will depend on what the external program is expecting.

Create a function to send ZMQ messages 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.

```
#####################
# general functions #
#####################
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)
```

Use or create a program that will receive ZMQ messages 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)))
```

12.11.14 Sending Messages To Status Bar Or Desktop Notify Dialogs

Es gibt mehrere Möglichkeiten, dem Benutzer **Informationen zu übermitteln**.

A **status bar** is used for *short information* to show the user.

Anmerkung
Not all screens have a status bar.

Status bar usage example

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` is in seconds and we assume `statusbar` is the Qt Designer set name of the widget.

You can also use the `Status` library to send a message to the `notify` library if it is enabled (usually set in `ScreenOptions` widget): this will send the message to the statusbar and the **desktop notify dialog**.

The messages are also recorded until the user erases them using controls.
The users can recall any recorded messages.

There are several options:

`STATUS.TEMPORARY_MESSAGE`
Show the message for a short time only.

`STATUS.OPERATOR_ERROR` , `STATUS.OPERATOR_TEXT` , `STATUS.NML_ERROR` , `STATUS.NML_TEXT`

Example of sending an operator message:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

You can send messages thru LinuxCNC's operator message functions.
These are usually caught by the notify system, so are equal to above.
They would be printed to the terminal as well.

```
ACTION.SET_DISPLAY_MESSAGE('MESSAGE')
ACTION.SET_ERROR_MESSAGE('MESSAGE')
```

12.11.15 Catch Focus Changes

Focus is used to **direct user action** such as keyboard entry to the proper widget.

Get currently focused widget

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

Get focused widget when focus changes

```
# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
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')
```

Notice we sometimes compare to `widget`, sometimes to `widget.parent()`.

This is because *some QtVCP widgets are built from multiple **sub-widgets*** and the latter actually get the focus; so we need to **check the parent** of those sub-widgets.

Other times the main widget is what gets the focus; ie G-code display widget can be set to accept focus and in that case there are no sub-widgets in it so comparing to the `widget.parent()` would get you the container that holds the G-code widget.

12.12 QtVCP Development

12.12.1 Übersicht

The intention of QtVCP is to **supply an infrastructure to support screen and VCP panel building for LinuxCNC**.

By providing a *diverse widget* set and supporting *custom coding*, QtVCP hopes that development energy will be expended in *one toolkit* rather than continuous re-invention.

By using the same toolkit across many screens/panels users should have an easier time customizing/-creating these, and developers should find it easier to help trouble shoot with less effort.

QtVCP uses a **Qt Designer built `.ui` file** and a **Python handler file** to **load and control a screen/-panel that displays Qt widgets** to **control LinuxCNC's motion controller or HAL pins**.

There are *builtin screens and panels*, easily loaded by a user, or users can build/modify one of their own.

QtVCP uses **libraries and custom widgets** to hide some of the complexity of interfacing to LinuxCNC. By using QtVCP's library rather than LinuxCNC's, we can mitigate minor LinuxCNC code changes.

12.12.2 Builtin Locations

Builtin screens and panels are stored in separate folders:

- *Screens* in `share/qtvcp/screens`
- *Panels* in `share/qtvcp/panels`
- *Stock images* in `share/qtvcp/images`

Screens and panels are sorted by their folder name, which is also the name used to load them.

Inside the folder would be:

- the `.ui` *file*,
- die *Handler-Datei*, und
- possibly the `.qss` *theme file*.

12.12.3 QtVCP Startup To Shutdown

QtVCP source is located in `+src/emc/usr_intf/qtvcp+` folder of LinuxCNC source tree.

12.12.3.1 QtVCP Startup

When QtVCP first starts:

1. It must decide if this object is a screen or a panel.
2. It searches for and collects information about paths of required files and useful folders.
3. It then:
   a. Builds the HAL component,
   b. Loads the window instance,
   c. Adds handler extensions,
   d. Installs an event filter.

   Now the window/widgets are instantiated, the HAL pins are built.
   This also initiates the +_init_hal()+ function of the widgets. . Die Handler-Funktion +initialized__(wird aufgerufen . The STATUS library is forced to update. . HAL component is set ready at this point. . A variety of optional switch arguments are set, including calling a POSTGUI HAL file (if a screen). . Terminate signals are trapped and QtVCP now polls for events.

12.12.3.2 QtVCP Shutdown

Finally when QtVCP is asked to shutdown:

1. It calls shutdown functions in the handler file,
2. STATUS monitoring is shut down
3. HAL component gets killed

12.12.4 Path Information

When QtVCP loads it collects paths informations.

This is available in the handler file's +__init__()+ function as **path**:

IMAGEDIR
: Path of builtin images

SCREENDIR
: Path of builtin motion controller screens

PANELDIR
: Path of builtin accessory panels

WORKINGDIR
: Path of where QtVCP was launched from

CONFIGPATH
: Path of the launched configuration

BASEDIR
: General path, used to derive all paths

BASENAME
: Generic name used to derive all paths

LIBDIR
Path of QtVCP's Python library

HANDLER
Pfad der Handler-Datei

XML
Path of .ui file

DOMAIN
Path of translation

IS_SCREEN
Screen/panel switch

12.12.5 Idiosyncrasies

These try to cover non-obvious situations.

12.12.5.1 Error Code Collecting

LinuxCNC's error code collecting can only be read from one place.

Wenn es gelesen wird, ist es ***konsumiert***, d.h. *kein anderes Objekt kann es lesen*.

In QtVCP screens, it is recommended to *use the* `ScreenOptions` *widget to set up error reading*.

Errors are then ***sent to other objects*** *via* **`STATUS`** *signals*.

12.12.5.2 Jog Rate

LinuxCNC has no internal record of jog rate: *you must specify it at the time of jogging*.

QtVCP uses the `STATUS` library to *keep track of the latest linear and angular jog rates*.

It is **always specified in machine units per minute** and *must be converted when in non-machine units mode*.
So, if your machine is imperial based but you are in metric mode, changes to jog rate sent to `ACTION` functions must be converted to imperial.
In the same manner, if the machine is metric based and you are in imperial mode, changes to jog rate must be sent to `ACTION` functions in metric units.
For angular jog rates the units don't change in metric/imperial mode so you can send them to `ACTION` functions without conversion.

While you are free to ignore this jogging record while building screens, anyone modifying your screen and using the builtin jog rate widgets would not get the desired results as the `ACTION` library's **`DO_JOG`** function gets it's jog rate from the `STATUS` library.

12.12.5.3 Keybinding

Warnung
Keybinding is always a *difficult-to-get-right-in-all-cases* affair.

Custom keybinding functions are to be *defined in the handler file*.

Most importantly widgets that require regular key input and not jogging, should be checked for in the `processed_key_event__` function.

12.12.5.4 Preference File

Some QtVCP widgets use the preference file to record important informations.

This *requires the preference file to be set up early* in the widget initialization process.
The easiest way to do this is to **use the `ScreenOptions` widget**.

12.12.5.5 Widget Special Setup Functions

QtVCP looks for and calls the `+_hal_init()+` function *when the widget is first loaded*.

It is not called when using Qt Designer editor.

After this function is called the widget has access to some special variables:

`self.HAL_GCOMP`
: The *HAL component* instance

`self.HAL_NAME`
: This *widget's name* as a string

`self.QT_OBJECT_`
: dieses *Widgets PyQt-Objekt als Instanz*

`self.QTVCP_INSTANCE_`
: The *very top level parent* of the screen

`self.PATHS_`
: The *instance of QtVCP's path* library

`self.PREFS_`
: The *instance of an optional preference file*

`self.SETTINGS_`
: The `Qsettings` *object*

When making a custom widget, _import and sub class _the **`+_HalWidgetBase+`** class for this behavior.

12.12.5.6 Dialogs

Dialogs (AKA "pop up windows") are *best loaded with the* `ScreenOptions` *widget*, but they can be placed on the screen in Qt Designer.

It doesn't matter where on the layout but *to make them hidden*, cycle the `state` property to true then false.

By default, if there is a preference file, the dialogs will remember their last size/placement.
It is possible to override this so they open in the same location each time.

12.12.5.7 Styles (Themes)

While it is possible to set styles *in Qt Designer*, it is more convenient to change them later if they are all set in a ***separate `.qss` file***.

The file should be put in the *same location as the handler file*.

Kapitel 13

Programmierung der Benutzeroberfläche

13.1 Panelui

13.1.1 Einführung

Panelui is a userspace component to interface buttons to LinuxCNC or HAL: * It decodes MESA 7I73 style key-scan codes and calls the appropriate routine. * It gets input from a realtime component - sampler. Sampler gets it's input from either the MESA 7i73 or sim_matrix_kb component. * Panelui is configurable using an INI style text file to define button types, HAL pin types, and/or commands. * It can be extended using a Python based *handler* file to add functions.

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= This is used to designate the key that the button responds to. It can be NONE or ROW number and column number eg R1C2. A row and column can only be used once.
- OUTPUT= Hier wird der Ausgabetyp der Schaltfläche festgelegt, z. B. S32, U32, FLOAT, BIT, 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]
 # Die doppelte(n) Klammer(n) definieren die Gruppe(n) von Optionsschaltflächen.
# Der Gruppenname muss eindeutig sein und Groß- und Kleinschreibung wird beachtet.
# Die Ausgabe der Gruppe wird durch die aktive Schaltfläche gesteuert, nicht direkt durch ↩
    den Keycode.
# DEFAULT verweist auf eine Schaltfläche in der Gruppe durch den Namen und unterscheidet ↩
    zwischen Groß- und Kleinschreibung.
  [[group1_name]]
    KEY = NONE
    OUTPUT = FLOAT
    DEFAULT = small
    # Die dreifachen Klammerabschnitte definieren die Schaltflächen in dieser Gruppe.
    # Die Namen der Schaltflächen müssen eindeutig sein und Groß- und Kleinschreibung wird ↩
        beachtet.
    # Es muss mindestens zwei Schaltflächen in einer Gruppe geben.
    #
    # Diese Schaltfläche mit dem Namen 'small' wird durch die Taste Zeile 0 Spalte 1 ↩
        gesteuert.
    # Sie bewirkt, dass der Gruppenausgang .0001 ist, wenn sie gedrückt wird.
    # Sie hat keinen eigenen Ausgang, aber einen Status Pin, der den aktuellen Zustand der ↩
        Taste anzeigt.
    # Da diese Taste in einer Gruppe ist, hat DEFAULT keine Bedeutung. # da OUTPUT nicht ' ↩
        COMMAND' ist,
    # werden _COMMAND-Einträge ignoriert.
    [[[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
     # Diese Schaltfläche mit dem Namen 'large' wird von der Taste Zeile 0 Spalte 2 ↩
         gesteuert.
     # Sie bewirkt, dass der Gruppenausgang 1000 ist, wenn sie gedrückt wird.
     # Er hat einen eigenen S32-Ausgang, der bei true 20 und bei false 0 ist.
     # Sie hat auch einen Status-Pin, der ihren aktuellen Zustand anzeigt.
     # Da diese Taste in einer Gruppe ist, hat DEFAULT keine Bedeutung.
     # Da OUTPUT nicht 'COMMAND' ist, werden _COMMAND-Einträge ignoriert.
    [[[large]]]
      KEY = R0C2
      GROUP = group1_name
      GROUP_OUTPUT = 1000
      OUTPUT = S32
```

```
    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]
  # Jeder Button-Name in doppelten Klammern muss eindeutig sein und Groß- und  ↩
      Kleinschreibung wird unterschieden.
  # Diese Schaltfläche mit dem Namen 'tool_change' wird von der Taste in Zeile 2 Spalte 5  ↩
      gesteuert.
  # Sie hat einen BIT-Ausgang, der bei einem wahren Zustand 1 und bei einem falschen  ↩
      Zustand 0 ausgibt.
  # Er hat auch einen Status-Pin, der seinen aktuellen Zustand anzeigt.
  # DEFAULT setzt diesen bei der ersten Initialisierung auf true.
  # Die _COMMAND werden nicht verwendet, da OUTPUT nicht auf COMMAND gesetzt ist, aber die  ↩
      Validierung wird
  # die Zeilen dennoch hinzufügen.
  [[tool_change]]
    KEY = R2C5
    OUTPUT = BIT
    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

- required arguments: axis number (int), direction (int)

incremental_jog

- required arguments: axis number (int), direction (int), distance (float)

quill_up

- optional arguments: machine Z axis absolute position (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)]

# Also required - handler class class HandlerClass:

    # This will be pretty standard to gain access to everything
    # linuxcnc_stat: is the python status instance of linuxcnc
    # linuxcnc_cmd: is the python command instance of linuxcnc
    # commands: is the command instance so one can call the internal routines
    # master: give access to the master functions/data
```

```
    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)

        # call a mdi command to print a msg in linuxcnc
        # This requires linuxcnc to be homed, but does not check for that.
        # parent commands expect a widget_instance - None is substituted
        self.parent.mdi(None,'(MSG, Hello Linuxcnc World!)')

    # Each call to this function will cycle the mode of linuxcnc
    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, often required
    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

13.2 Filter-Programme

13.2.1 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.

13.2.2 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.

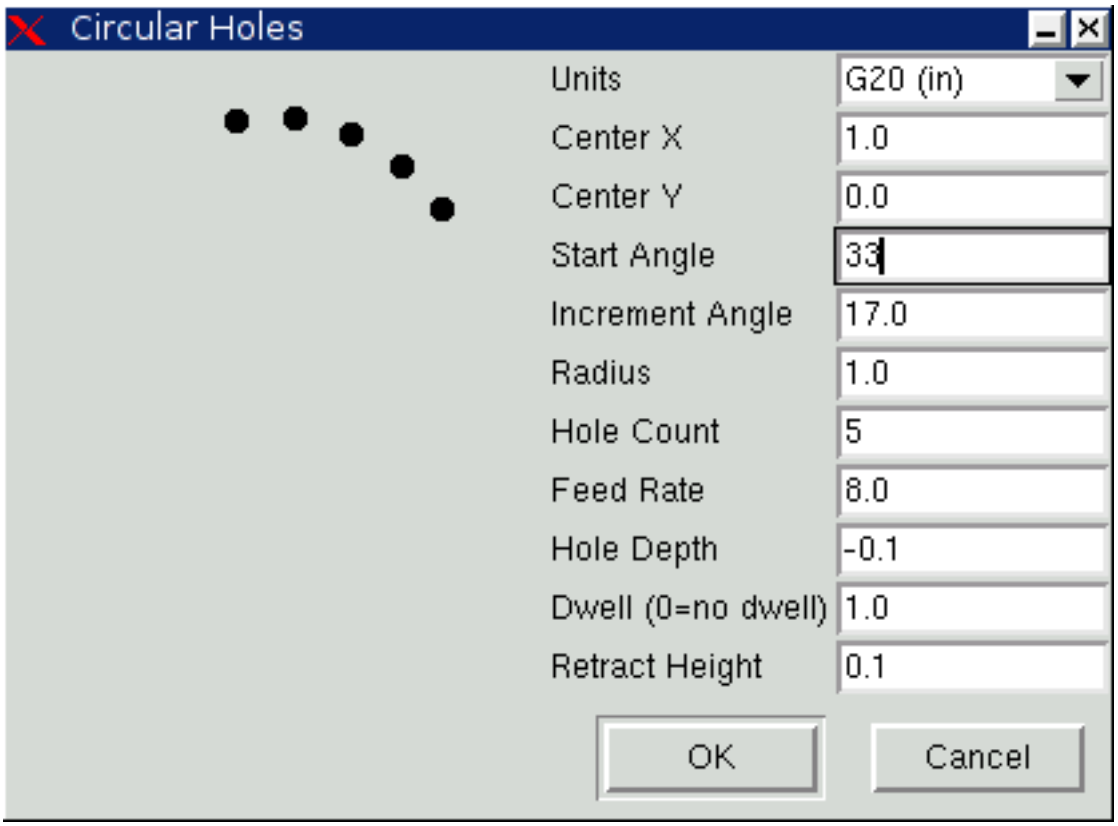


Abbildung 13.1: Kreisförmige Löcher

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.

13.2.3 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-Standardfehlerst
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_() )
```

13.3 HAL-Benutzeroberfläche

13.3.1 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
```

13.3.2 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.

Beispiel 13.1 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  bit   IN  FALSE  halui.mdi-command-00 <== quill-up
   10  bit   IN  FALSE  halui.mdi-command-01 <== reference-pos
   10  bit   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).

13.3.3 Beispiel-Konfiguration

Eine Beispiel-Sim-Konfiguration (`configs/sim/axis/halui_pyvcp/halui.ini`) ist in der Distribution enthalten.

13.3.4 Halui-Pin-Referenz

Alle halui-Pins sind auch in der halui-Manualseite dokumentiert:

```
$ man halui
```

13.3.4.1 Abort

- halui.abort' (bit, in) - Pin zum Senden einer Abbruchmeldung (löscht die meisten Fehler)

13.3.4.2 E-Stop

- halui.estop.activate' (bit, in) - Pin für die Anforderung des Notausschalters
- halui.estop.is-activated' (Bit, out) - zeigt an, dass der Not-Aus-Schalter zurückgesetzt wurde
- halui.estop.reset' (bit, in) - Pin für die Anforderung eines Not-Aus-Resets

13.3.4.3 Vorschub Neufestsetzung (engl. override)

- *halui.feed-override.count-enable* (bit, in) - muss wahr sein, damit *counts* oder *direct-value* funktioniert.
- *halui.feed-override.counts* (s32, in) - counts * scale = FO percentage. Kann mit einem Encoder oder *direct-value* verwendet werden.
- *halui.feed-override.decrease* (bit, in) - Pin zum Verringern des FO (-=Skala)
- *halui.feed-override.increase* (bit, in) - Pin zur Erhöhung des FO (+=Skala)
- *halui.feed-override.reset* (bit, in) - Pin zum Zurücksetzen des FO (scale=1.0)
- *halui.feed-override.direct-value* (bit, in) - falsch, wenn der Encoder verwendet wird, um die Anzahl zu ändern, wahr, wenn die Anzahl direkt eingestellt wird.
- *halui.feed-override.scale* (float, in) - Pin zum Einstellen der Skala für die Erhöhung und Verringerung des *feed-override*.
- *halui.feed-override.value* (float, out) - aktueller FO-Wert

13.3.4.4 Mist

- *halui.mist.is-on* (bit, out) - zeigt an, dass Nebel eingeschaltet ist
- *halui.mist.off* (bit, in) - Pin zum Anfordern von Nebel
- halui.mist.on' (bit, in) - Pin zur Abfrage von Nebel ein

13.3.4.5 Flood

- *halui.flood.is-on* (bit, out) - zeigt an, dass die Flut an ist
- *halui.flood.off* (bit, in) - Pin zum Anforderung des Ausschaltens der Flut
- *halui.flood.on* (bit, in) - Pin für die Anforderung für das Einschalten der Flut

13.3.4.6 Referenzfahrt (engl. homing)

- *halui.home-all* (bit, in) - Pin zum Anfordern einer Referenzfahrt aller Achsen. Dieser Pin ist nur vorhanden, wenn HOME_SEQUENCE in der INI-Datei festgelegt ist.

13.3.4.7 Lube

- halui.lube.is-on' (bit, out) - zeigt an, dass das Schmiermittel eingeschaltet ist
- halui.lube.off' (bit, in) - Pin für die Anforderung von Schmiermittel aus
- halui.lube.on" (bit, in) - Stift für die Anforderung von Schmiermittel auf

13.3.4.8 Machine

- halui.machine.units-per-mm' (float out) - Pin für Maschineneinheiten-pro-mm (inch:1/25.4, mm:1) entsprechend der inifile-Einstellung: [TRAJ]LINEAR_UNITS
- halui.machine.is-on' (bit, out) - zeigt an, dass die Maschine eingeschaltet ist
- *halui.machine.off* (bit, in) - Pin zum Anfordern der Maschinenabschaltung
- *halui.machine.on* (bit, in) - Pin zum Anfordern der Maschinen-Einschaltung

13.3.4.9 Max. Geschwindigkeit

The maximum linear velocity can be adjusted from 0 to the MAX_VELOCITY that is set in the [TRAJ] section of the INI file.

- *halui.max-velocity.count-enable* (bit, in) - muss true sein, damit *counts* oder *direct-value* funktionieren.
- *halui.max-velocity.counts* (s32, in) - counts * scale = MV percentage. Kann mit einem Encoder oder *direct-value* verwendet werden.
- *halui.max-velocity.direct-value* (bit, in) - false bei Verwendung des Encoders zum Ändern der Anzahl, true beim direkten Festlegen von Zählungen.
- *halui.max-velocity.decrease* (bit, in) - Pin zur Verringerung der maximalen Geschwindigkeit
- *halui.max-velocity.increase* (bit, in) - Pin zur Erhöhung der maximalen Geschwindigkeit
- *halui.max-velocity.scale* (float, in) - der Betrag, der auf die aktuelle maximale Geschwindigkeit bei jedem Übergang von Aus zu Ein des An- oder Abnahmestiftes in Maschineneinheiten pro Sekunde angewendet wird.
- *halui.max-velocity.value* (float, out) - ist die maximale lineare Geschwindigkeit in Maschineneinheiten pro Sekunde.

13.3.4.10 MDI

- halui.mdi-command-<nn>' (bit, in) - halui versucht, den in der INI-Datei definierten MDI-Befehl zu senden. <nn> ist eine zweistellige Zahl, die bei 00 beginnt.
  Wenn das Kommando erfolgreich ist, dann wird es LinuxCNC in den MDI-Modus setzen und dann zurück in den manuellen Modus.
  Wenn keine [HALUI]MDI_COMMAND Variablen in der INI-Datei gesetzt sind, werden keine halui.mdi-command-<nn> Pins von halui exportiert.

13.3.4.11 Joint

N = Gelenknummer (0 … num_joints-1)
Beispiel:

- *halui.joint.N.select* (Bit in) - Pin zur Auswahl von Gelenk N
- *halui.joint.N.is-s selected* (bit out) – Status-Pin, dass Gelenk N ausgewählt ist
- *halui.joint.N.has-fault* (bit out) - Status-Pin, der angibt, dass Gelenk N einen Fehler hat
- *halui.joint.N.home* (Bit in) - Pin für Referenzfahrt von Gelenk N
- *halui.joint.N.is-homed* (bit out) - Status-Pin, der angibt, dass Gelenk N referenziert ist
- *halui.joint.*N.*on-hard-max-limit* (bit out) - Status-Pin, der anzeigt, dass Gelenk *N* am positiven Hardware-Limit liegt
- *halui.joint.*N.*on-hard-min-limit* (bit out) - Status-Pin, der anzeigt, dass sich Gelenk *N* am negativen Hardware-Limit befindet
- *halui.joint.*N.*on-hard-max-limit* (bit out) - Status-Pin, der anzeigt, dass Gelenk *N* am positiven Hardware-Limit liegt
- *halui.joint.*N.*on-soft-min-limit* (bit out) - Status-Pin, der anzeigt, dass sich Gelenk *N* an der negativen Softwaregrenze befindet
- *halui.joint.*N.*override-limits* (bit out) - Status-Pin, der angibt, dass die Grenzen von Gelenk *N* vorübergehend außer Kraft gesetzt werden
- *halui.joint.*N.*unhome* (bit in) - Pin für das Aufheben der Referenzierung von Gelenk *N*
- *halui.joint.selected* (u32 out) - ausgewählte Gelenknummer (0 … num_joints-1
- *halui.joint.selected.has-fault* (bit out) - Status-Pin ausgewähltes Gelenk ist fehlerhaft
- *halui.joint.selected.home* (Bit in) - Pin für das Homing des ausgewählten Gelenks
- *halui.joint.s selected.is-homed* (bit out) - Status-Pin, der angibt, dass das ausgewählte Gelenk referenziert ist
- *halui.joint.selected.on-hard-max-limit* (bit out) - Status-Pin, der anzeigt, dass sich das ausgewählte Gelenk auf dem positiven Hardware-Limit befindet
- *halui.joint.selected.on-hard-min-limit* (bit out) - Status-Pin, der anzeigt, dass sich das ausgewählte Gelenk am negativen Hardware-Limit befindet
- *halui.joint.s selected.on-soft-max-limit* (bit out) - Status-Pin, der angibt, dass sich das ausgewählte Gelenk auf der positiven Softwaregrenze befindet
- *halui.joint.selected.on-soft-min-limit* (bit out) - Status-Pin, der anzeigt, dass sich das ausgewählte Gelenk auf dem negativen Software-Limit befindet
- *halui.joint.s selected.override-limits* (bit out) - Status-Pin, der angibt, dass die Grenzen des ausgewählten Gelenks vorübergehend außer Kraft gesetzt werden
- *halui.joint.s selected.unhome* (bit in) - Pin zum Unhoming des ausgewählten Gelenks

13.3.4.12 Gelenk-Joggen

N = Anzahl der Gelenke (0 ... num_joints-1)

- *halui.joint.jog-deadband* (float in) - Pin zum Einstellen der Jog-Analog-Totzone (Jog-Analogeingänge, die kleiner/langsamer als dieser - im absoluten Wert - sind, werden ignoriert)
- *halui.joint.jog-speed* (float in) - Pin zur Einstellung der Jog-Geschwindigkeit für Plus/Minus-Jogging.
- *halui.joint.*N.*analog* (float in) - Pin zum Joggen des Gelenks *N* mit einem Float-Wert (z. Joystick). Der Wert, der normalerweise zwischen 0,0 und ±1,0 festgelegt ist, wird als Jog-Speed-Multiplikator verwendet.
- *halui.joint.*N.*increment* (float in) - Pin zum Einstellen des Jog-Inkrements für Gelenk *N* bei Verwendung von increment-plus/minus
- *halui.joint.*N.*increment-minus* (bit in) - eine steigende Kante lässt das Gelenk *N* um den Inkrementbetrag in die negative Richtung joggen
- *halui.joint.*N.*increment-plus* (bit in) - eine steigende Kante lässt das Gelenk *N* um den Inkrementbetrag in die positive Richtung joggen
- *halui.joint.*N.*minus* (bit in) - Pin für Jogginggelenk *N* in negativer Richtung bei der halui.joint.jog-Geschwindigkeitsgeschwindigkeit
- *halui.joint.*N.*plus* (bit in) - Pin für Jogginggelenk *N* in positiver Richtung bei der halui.joint.jog-Geschwindigkeitsgeschwindigkeit
- *halui.joint.selected.increment* (float in) - Pin zum Einstellen des Jog-Inkrements für das ausgewählte Gelenk bei Verwendung von increment-plus/minus
- *halui.joint.s selected.increment-minus* (bit in) – eine ansteigende Flanke lässt das ausgewählte Gelenk um den Inkrementbetrag in die negative Richtung joggen
- halui.joint.selected.increment-plus' (Bit in) - eine steigende Flanke bewirkt, dass das ausgewählte Gelenk um den Betrag des Inkrements in die positive Richtung bewegt wird
- halui.joint.selected.minus' (bit in) - Pin zum Joggen des ausgewählten Gelenks in negativer Richtung mit der halui.joint.jog-speed Geschwindigkeit
- halui.joint.selected.plus' (bit in) - Pin für das Joggen des ausgewählten Gelenks in positiver Richtung mit der halui.joint.jog-speed Geschwindigkeit

13.3.4.13 Achse

L = Buchstabe der Achse (xyzabcuvw)

- *halui.axis.*L.*select* (Bit) - Pin zur Auswahl der Achse anhand des Buchstaben
- *halui.axis.*L.*is-selected* (Bit out) - Status-Pin, dass die Achse *L* ausgewählt ist
- *halui.axis.*L.*pos-commanded* (float out) - Befohlene Achsenposition in Maschinenkoordinaten
- *halui.axis.*L.*pos-feedback* float out) - Rückmeldung der Achsposition in Maschinenkoordinaten
- *halui.axis.*L.*pos-relative* (float out) – Rückmeldung der Achsenposition in relativen Koordinaten

13.3.4.14 Achsen-Jogging

L = Buchstabe der Achse (xyzabcuvw)

- *halui.axis.jog-deadband* (float in) - Pin zum Einstellen der Jog-Analog-Totzone (Jog-Analogeingänge, die kleiner/langsamer als dieser (im absoluten Wert) sind, werden ignoriert)
- *halui.axis.jog-speed* (float in) - Pin zum Einstellen der Jog-Geschwindigkeit für Plus/Minus-Joggen.
- *halui.axis.*L.*analog* (float in) - Pin zum Joggen der Achse *L* mit einem Float-Wert (z.B. Joystick). Der Wert, der normalerweise zwischen 0,0 und ±1,0 festgelegt ist, wird als Jog-Speed-Multiplikator verwendet.
- *halui.axis.*L.*increment* (float in) - Pin zum Einstellen des Jog-Inkrements für Achse *L* bei Verwendung von Increment-Plus/Minus
- *halui.axis.*L.*increment-minus* (bit in) – eine ansteigende Flanke bewirkt, dass die Achse *L* um den Inkrementbetrag in die negative Richtung joggt
- *halui.axis.*L.*increment-plus* (Bit in) - eine steigende Kante lässt die Achse *L* um den Inkrementbetrag in die positive Richtung joggen
- *halui.axis.*L.*minus* (Bit in) - Pin für Jogging entlang der Achse *L* in negativer Richtung bei der halui.axis.jog-Geschwindigkeitsgeschwindigkeit
- *halui.axis.*L.*plus* (bit in) - Pin zum Joggen der Achse *L* in positiver Richtung mit der Geschwindigkeit halui.axis.jog-speed
- *halui.axis.selected* (U32 out) - Ausgewählte Achse (nach Index: 0:x 1:y 2:Z 3:A 4:B 5:CR 6:U 7:V 8:W)
- *halui.axis.selected.increment* (float in) - Pin zum Einstellen des Jog-Inkrements für die ausgewählte Achse bei Verwendung von increment-plus/minus
- *halui.axis.s selected.increment-minus* (bit in) - eine steigende Flanke bewirkt, dass die ausgewählte Achse um den Inkrementbetrag in die negative Richtung joggt
- *halui.axis.selected.increment-plus* (Bit in) - Eine steigende Kante lässt die ausgewählte Achse um den Inkrementbetrag in die positive Richtung joggen
- *halui.axis.selected.minus* (Bit in) - Pin zum Joggen der ausgewählten Achse in negativer Richtung bei der halui.axis.jog-Geschwindigkeitsgeschwindigkeit
- *halui.axis.selected.plus* (pin in) - zum Joggen des ausgewählten Achsenbits in positiver Richtung bei der halui.axis.jog-Geschwindigkeitsgeschwindigkeit

13.3.4.15 Modus

- *halui.mode.auto* (bit, in) - Pin zum Anfordern des automatischen Modus
- *halui.mode.is-auto* (bit, out) - zeigt an, dass der Auto-Modus eingeschaltet ist
- *halui.mode.is-joint* (bit, out) - zeigt an, dass der Gelenk-für-Gelenk (engl. joint by joint)-Jogging-Modus eingeschaltet ist
- *halui.mode.is-manual* (bit, out) - zeigt an, dass der manuelle Modus eingeschaltet ist
- *halui.mode.is-mdi* (bit, out) - zeigt an, dass der MDI-Modus eingeschaltet ist
- *halui.mode.is-teleop* (bit, out) - zeigt an, dass der koordinierte Jog-Modus eingeschaltet ist
- *halui.mode.joint* (bit, in) - Pin für die Abfrage des Joint-by-Joint-Jog-Modus

- *halui.mode.manual* (bit, in) - Pin für die Anforderung des manuellen Modus
- halui.mode.mdi’ (bit, in) - Pin zur Abfrage des MDI-Modus
- *halui.mode.teleop* (bit, in) - Pin zum Anfordern des koordinierten Jog-Modus

13.3.4.16 Programm

- *halui.program.block-delete.is-on* (bit, out) - Status-Pin, der anzeigt, dass Block delete on ist
- *halui.program.block-delete.off* (bit, in) - Pin zum Anfordern, dass das Blocklöschen deaktiviert ist
- *halui.program.block-delete.on* (bit, in) - Pin zum Anfordern, dass das Blocklöschen aktiviert ist
- *halui.program.is-idle* (bit, out) - Status-Pin, die anzeigt, dass kein Programm läuft
- *halui.program.is-paused* (bit, out) - Status-Pin, der angibt, dass ein Programm angehalten wurde
- *halui.program.is-running* (bit, out) - Status-Pin, der angibt, dass ein Programm ausgeführt wird
- *halui.program.optional-stop.is-on* (bit, out) - Status-Pin zur Angabe, dass der optionale Stopp eingeschaltet ist
- *halui.program.optional-stop.off* (bit, in) - Pin, der anfordert, dass der optionale Stopp ausgeschaltet ist
- halui.program.optional-stop.on’ (bit, in) - Pin, der anfordert, dass der optionale Stopp eingeschaltet ist
- *halui.program.pause* (bit, in) - Pin zum Anhalten eines Programms
- *halui.program.resume* (bit, in) - Pin zum Fortsetzen eines pausierten Programms
- *halui.program.run* (bit, in) - Pin zum Ausführen eines Programms
- *halui.program.step* (bit, in) - Pin für das Steppen eines Programms
- *halui.program.stop* (bit, in) - Pin zum Stoppen eines Programms

13.3.4.17 Eilgang-Override (engl. rapid override)

- halui.rapid-override.count-enable’ (Bit in (Standard: TRUE)) - Wenn TRUE, wird Rapid Override geändert, wenn sich die Zählerstände ändern.
- halui.rapid-override.counts’ (s32 in) - counts X scale = Rapid Override Prozentsatz. Kann mit einem Encoder oder *direct-value* verwendet werden.
- halui.rapid-override.decrease’ (bit in) - Pin zum Verringern des Rapid Override (-=scale)
- *halui.rapid-override.direct-value* (Bit in) - pin, um den direkten Wert zu aktivieren Rapid Override-Eingabe
- *halui.rapid-override.increase* (bit in) - Pin zur Erhöhung des Rapid Override (+=scale)
- *halui.rapid-override.scale* (float in) - Pin zum Einstellen der Skala beim Ändern der Rapid Override
- *halui.rapid-override.value* (float out) - aktueller Rapid Override-Wert
- *halui.rapid-override.reset* (bit, in) - Pin zum Zurücksetzen des Rapid-Override-Wertes (Skala=1.0)

13.3.4.18 Spindel Neufestsetzung (engl. override)

- *halui.spindle*.N.*override.count-enable* (bit, in) - muss wahr sein, damit *counts* oder *direct-value* funktioniert.
- *halui.spindle*.N.*override.counts* (s32, in) - zählt * Skala = SO-Prozentsatz. Kann mit einem Encoder oder ”Direct-Value” verwendet werden.
- *halui.spindle*.N.*override.decrease* (bit, in) - Pin zum Verringern der SO (-=Skala)
- *halui.spindle*.N.*override.direct-value* (bit, in) - false, wenn der Encoder zum Ändern der Zählerstände verwendet wird, true, wenn die Zählerstände direkt gesetzt werden.
- *halui.spindle*.N.*override.decrease* (bit, in) - Pin zum Verringern der SO (-=Skala)
- *halui.spindle*.N.*override.scale* (float, in) - Pin zum Einstellen der Skala beim Ändern der SO
- *halui.spindle*.N.*override.value* (float, out) - aktueller SO-Wert
- *halui.spindle*.N.*override.reset* (bit, in) - Pin zum Zurücksetzen des SO-Werts (scale=1.0)

13.3.4.19 Spindel

- *halui.spindle*.N.*brake-is-on* (bit, out) - zeigt an, dass die Bremse eingeschaltet ist
- *halui.spindle*.N.*brake-off* (bit, in) - Pin zur Deaktivierung der Spindel/Bremse
- *halui.spindle*.N.*brake-off* (bit, in) - Pin zur Deaktivierung der Spindel/Bremse
- *halui.spindle*.N.*decrease* (bit, in) - verringert die Spindeldrehzahl
- *halui.spindle*.N.*forward* (bit, in) - startet die Spindel mit Bewegung im Uhrzeigersinn
- *halui.spindle*.N.*increase* (bit, in)- erhöht die Spindeldrehzahl
- *halui.spindle*.N.*is-on* (bit, out) - zeigt an, dass die Spindel eingeschaltet ist (in beide Richtungen)
- *halui.spindle*.N.*reverse* (bit, in)- startet die Spindel mit einer Bewegung gegen den Uhrzeigersinn
- *halui.spindle*.N.*runs-backward* (bit, out) - zeigt an, dass die Spindel eingeschaltet ist und umgekehrt
- *halui.spindle*.N.*runs-forward* (bit, out) - zeigt an, dass die Spindel eingeschaltet und vorwärts läuft
- *halui.spindle*.N.*start* (bit, in) - startet die Spindel
- *halui.spindle*.N.*stop* (bit, in) - stoppt die Spindel

13.3.4.20 Werkzeug

- *halui.tool.length-offset.a* (float out) - aktuell angewendeter Werkzeuglängenversatz für die A-Achse
- *halui.tool.length-offset.b* (float out) - aktuell angewendeter Werkzeuglängenversatz für die B-Achse
- *halui.tool.length-offset.c* (float out) – aktuell angewendeter Werkzeuglängenversatz für die C-Achse
- *halui.tool.length-offset.u* (float out) - aktuell angewendeter Werkzeuglängenversatz für die U-Achse
- *halui.tool.length-offset.v* (float out) – aktuell angewendeter Werkzeuglängenversatz für die V-Achse
- *halui.tool.length-offset.w* (float out) - aktuell angewendeter Werkzeuglängenversatz für die W-Achse
- *halui.tool.length-offset.x* (float out) – aktuell angewendeter Werkzeuglängenversatz für die X-Achse
- *halui.tool.length-offset.y* (float out) - aktuell angewendeter Werkzeuglängenversatz für die Y-Achse

- *halui.tool.length-offset.z* (float out) - aktuell angewendeter Werkzeuglängenversatz für die Z-Achse
- *halui.tool.diameter* (float out) - Aktueller Werkzeugdurchmesser oder 0, wenn kein Werkzeug geladen ist.
- *halui.tool.number* (u32, out) - zeigt das aktuell ausgewählte Werkzeug an

13.4 Halui-Beispiele

Damit alle Halui-Beispiele funktionieren, müssen Sie die folgende Zeile in den [HAL]-Abschnitt der INI-Datei einfügen.

```
HALUI = halui
```

13.4.1 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.

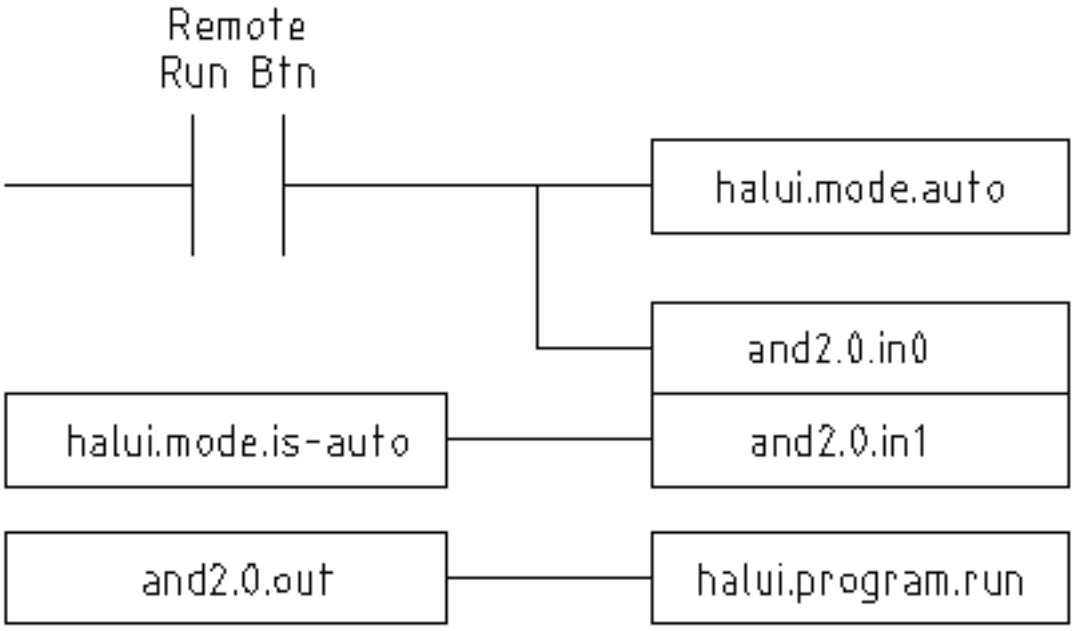


Abbildung 13.2: Beispiel für Fernstart

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
```

13.4.2 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 Ihrem E / A verbunden werden, um auf das Programm Pause oder fortzusetzen, wenn das externe System will LinuxCNC fortzusetzen.

```
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.

13.5 Python-Schnittstelle

Diese Dokumentation beschreibt das `linuxcnc` Python-Modul, das eine Python-API für die Kommunikation mit LinuxCNC bereithält.

13.5.1 Das Python-Modul `linuxcnc`

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

13.5.2 Verwendungsmuster für die LinuxCNC NML-Schnittstelle

Das allgemeine Muster für die Verwendung von `linuxcnc` ist in etwa wie folgt:

- das Modul `linuxcnc` importieren
- 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

linuxcnc" definiert auch den Python-Ausnahmetyp "error", um Fehlerberichte zu unterstützen.

13.5.3 Lesen des LinuxCNC-Status

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))
```

LinuxCNC verwendet den standardmäßig einkompilierten Pfad zur NML-Konfigurationsdatei, sofern er nicht überschrieben wird, siehe Lesen von INI-Datei Werten für ein Beispiel.

13.5.3.1 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 (engl. für Achsen)
(returns integer) - Anzahl der Achsen. Abgeleitet vom [TRAJ]COORDINATES INI-Eintrag.

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, 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_SYSTE 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.

g92_offset
(returns tuple of floats) - Pose des aktuellen g92-Offsets.

gcodes
(gibt Tupel von ganzen Zahlen zurück) - Aktive G-Codes für jede modale Gruppe. G-Code-Konstanten G_0, G_1, G_2, G_3, G_4, G_5, G_5_1, G_5_2, G_5_3, G_7, G_8, G_100, G_17, G_17_1, G_18, G_18_1, G_19, G_19_1, G_20, G_21, G_28, G_28_1, G_30, G_30_1, G_33, G_33_1, G_38_2, G_38_3, G_38_4, G_38_5, G_40, G_41, G_41_1, G_42, G_42_1, G_43, G_43_1, G_43_2, G_49, G_50, G_51, G_53, G_54, G_55, G_56, G_57, G_58, G_59, G_59_1, G_59_2, G_59_3, G_61, G_61_1, G_64, G_73, G_76, G_80, G_81, G_82, G_83, G_84, G_85, G_86, G_87, G_88, G_89, G_90, G_90_1, G_91, G_91_1, G_92, G_92_1, G_92_2, G_92_3, G_93, G_94, G_95, G_96, G_97, G_98, G_99

homed
(gibt ein Tupel von ganzen Zahlen zurück) - aktuell referenzierte Gelenke, 0 = nicht referenziert, 1 = referenziert.

id
(gibt ganze Zahl zurück) - aktuell ausgeführte Bewegungskennung.

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.

lube
(gibt ganze Zahl zurück) - Flag *Schmiermittel ein*.

lube_level
(gibt ganze Zahlen zurück) - spiegelt ”iocontrol.0.lube_level” 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.

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 <sec:the-spindle-dictionary,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.

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.werkzeug_tabelle[0].zoffset)
else:
    print("Kein Werkzeug geladen")
```

velocity
(returns float) - Diese Eigenschaft ist definiert, hat aber keine sinnvolle Interpretation.

13.5.3.2 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.

13.5.3.3 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. 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.

13.5.4 Das `Spindel`-Wörterbuch

brake (engl. für Bremse)
(gibt ganze Zahl zurück) - Wert des Spindelbremsflags.

direction (engl. für Richtung)
(returns integer) - Drehrichtung der Spindel. 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
(returns float) - Spindeldrehzahlwert, U/min, > 0: im Uhrzeigersinn, < 0: gegen den Uhrzeigersinn.

13.5.5 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 also 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
    c.mdi(”G0 X10 Y20 Z30”)
```

13.5.6 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)
```

13.5.6.1 `linuxcnc.command` Attribute

`serial`
: die Seriennummer des aktuellen Befehls

13.5.6.2 `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 einstellen.

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.

Warnung
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 gesendeten Befehls warten. Wenn die Zeitüberschreitung in Sekunden nicht angegeben wird, ist der Standardwert 5 Sekunden. Rückgabe -1 bei Zeitüberschreitung, Rückgabe RCS_DONE oder RCS_ERROR je nach Status der Befehlsausführung.

13.5.7 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)
```

13.5.8 Lesen von INI-Datei Werten

Hier ist ein Beispiel für das Lesen von Werten aus einer INI-Datei durch das Objekt `linuxcnc.ini`:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Ausführen wie folgt:
# python3 ini-example.py ~/emc2-dev/configs/sim/axis/axis_mm.ini

import sys
import linuxcnc

inifile = linuxcnc.ini(sys.argv[1])

# inifile.find() returns None if the key wasn't found - the
# following idiom is useful for setting a default value:
```

```
machine_name = inifile.find("EMC", "MACHINE") or "unknown"
print("machine name: ", machine_name)

# inifile.findall() gibt eine Liste von Übereinstimmungen oder eine leere Liste zurück
# falls der Schlüssel nicht gefunden wurde:

extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print("extensions: ", extensions)

# Standard-NML-Datei durch INI-Parameter neufestsetzen, falls angegeben
nmlfile = inifile.find("EMC", "NML_FILE")
if nmlfile:
    linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)
```

Oder für die selbe INI-Datei wie LinuxCNC:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# starten mit:
# 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
```

13.5.9 Der Typ `linuxcnc.positionlogger`

Einige Verwendungshinweise können von *src/emc/usr_intf/gremlin/gremlin.py* entnommen werden.

13.5.9.1 Members

`npts`
Anzahl der Punkte.

13.5.9.2 Methoden

`start(float)`
den Positionslogger starten und alle ARG-Sekunden ausführen

`clear()`
den Positionslogger löschen

`stop()`
den Positionslogger anhalten

`call()`
Plotte jetzt den Backplot.

`last([int])`
Gibt den letzten Punkt auf dem Plot oder keinen (als Python Ausdruck `None`) zurück

13.6 GStat

13.6.1 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):
```

13.6.2 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.

13.6.2.1 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

# Erstelle eine Komponente und Pins
h = hal.component("metric_status")
h.newpin("g20", hal.HAL_BIT, hal.HAL_OUT)
h.newpin("g21", hal.HAL_BIT, hal.HAL_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*.

13.6.2.2 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('MACHIBE 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)]
```

13.6.2.3 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('MACHIBE 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)]
```

13.6.3 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 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
(returns pyobject, pyobject, pyobject, pyobject) - Sent every 100 ms.
Returns tuples of position, relative position, distance-to-go and the joint actual position. Before homing, on multi-joint axes, only joint position is valid.

current-z-rotation
(returns float) - Sent as the current rotatated angle around the Z axis changes

requested-spindle-speed-changed
(liefert Float) - Wird gesendet, wenn sich die aktuell angeforderte Drehzahl ändert

actual-spindle-speed-changed
(returns float) - Sent when the actual RPM changes based on the HAL pin *spindle.0.speed-in*.

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 String zurück) - soll gesendet werden, wenn G-Code geladen wird.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

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
(returns string, string) - intended to be sent when updating the machine.
This depends on the widget/libraries used.

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
(returns Python dict) - intended to be sent when requesting a gui dialog.
It uses a Python dict for communication. The dict must include the following keyname pair:

- 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.

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:

- ID: *UNIQUE_ID_CODE*
  The dict usually has more keyname pair - it depends on implementation.

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.

progress
(liefert Integer, Python-Objekt) - soll gesendet werden, um den Fortschritt eines Filterprogramms anzuzeigen.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

following-error
(gibt eine Python-Liste zurück) - gibt eine Liste aller Gelenke zurück, die aktuell einem Fehler folgen.

13.6.4 Funktionen

These are convenience functions that are commonly used in programming.

set_jograte
(float) - LinuxCNC has no internal concept of jog rate -each GUI has its own. This is not always convenient.
This function allows one to set a jog rate for all objects connected to the signal *jograte-changed*.
It defaults to 15.
GSTAT.set_jog_rate(10) would set the jog rate to 10 machine-units-per-minute and emit the jograte-changed signal.

get_jograte()
(Nothing) - x = GSTAT.get_jograte() would return GSTAT's current internal jograte (float).

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) -

is_all_homed
(nothing) - This will return the current state of all_homed (BOOL).

machine_is_on
(nothing) - This will return the current state of machine (BOOL).

estop_is_clear
(nothing) - This will return the state of Estop (BOOL)

set_tool_touchoff
(tool,axis,value) - This command will

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. invoke the MDI command: G10 L10 P[TOOL] [AXIS] [VALUE],
4. warten, bis der Vorgang abgeschlossen ist,
5. invoke G43,
6. warten, bis der Vorgang abgeschlossen ist,
7. zurück in den ursprünglichen Modus wechseln.

set_axis_origin
(axis,value) - This command will

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. invoke the MDI command: G10 L20 P0 [AXIS] [VALUE],
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) - This function checks for required LinuxCNC mode.
It returns a Python tuple (state, mode)
mode will be set the mode the system is in
state will set to:

- false if mode is 0
- false if machine is busy
- true if LinuxCNC is in the requested mode
- None if possible to change, but not in requested mode

get_current_mode
(nothing) - gibt eine ganze Zahl zurück: den aktuellen LinuxCNC-Modus.

set_selected_joint
(integer) - records the selected joint number internally.
requests the joint to be selected by emitting the
joint-selection-changed message.

get_selected_joint
(None) - returns integer representing the internal selected joint number.

set_selected_axis
(string) - records the selected axis letter internally.
Requests the axis to be selected by emitting the *axis-selection-changed* message.

get_selected_axis
(None) - returns string representing the internal selected axis letter.

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) -

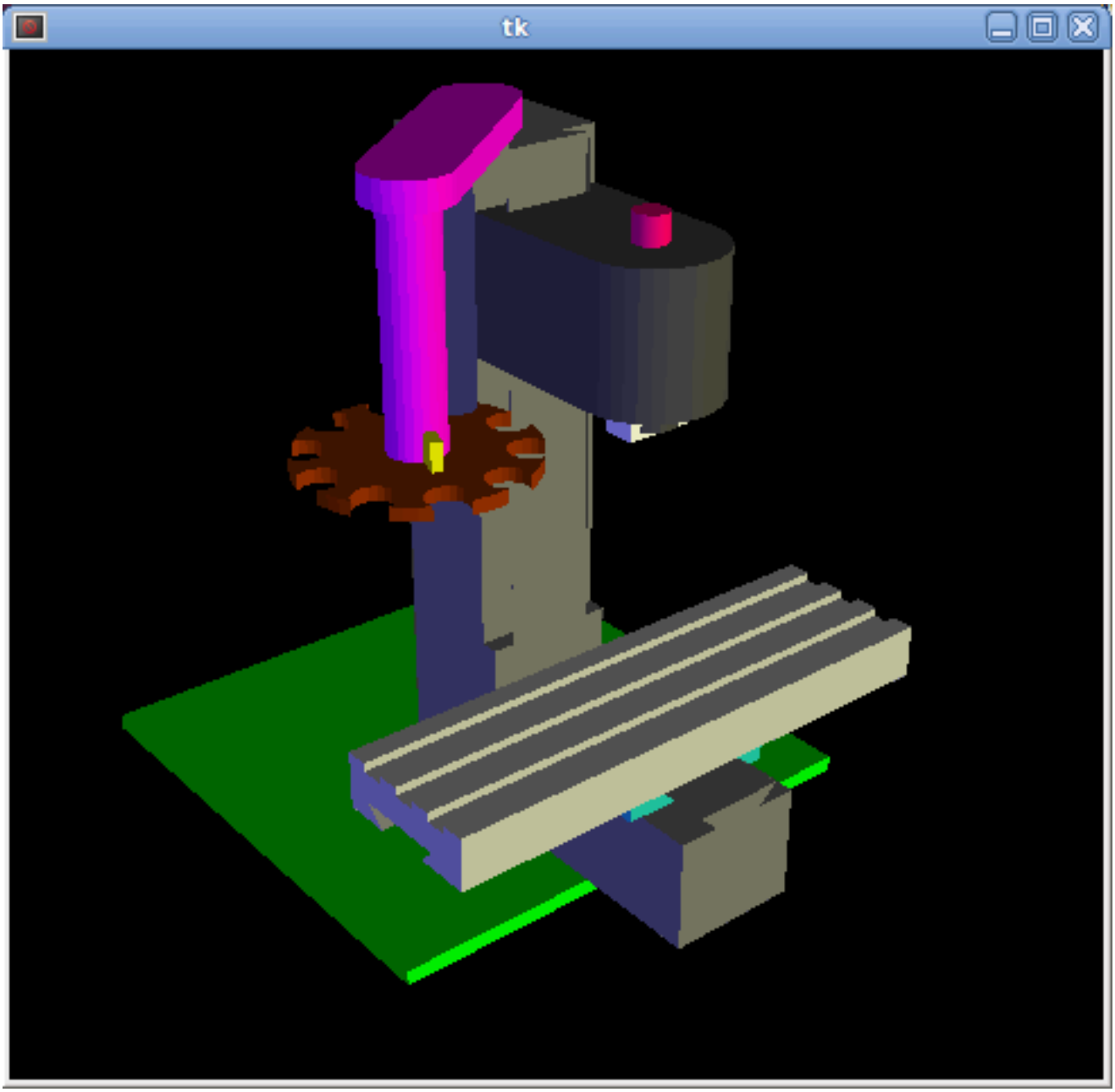
shutdown
(None) -

13.6.5 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.

13.7 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.

13.7.1 Start the script

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
```

13.7.2 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 Userspace 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.HAL_FLOAT, hal.HAL_IN)
c.newpin(”joint1”, hal.HAL_FLOAT, hal.HAL_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.

13.7.3 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])
```

13.7.4 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.

13.7.5 Animating Parts

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.

13.7.6 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.
- Create the draw bar.
- Define the motion of the draw bar.
- 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:

- For a *serial machine* each new part will be added to the collection of the previous part.
- 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])
```

13.7.7 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.
- `part = Capture()`
  Ich habe keine Ahnung, was das bewirkt! Aber es scheint wichtig für die Werkzeugspitzen-Visualisierung zu sein ...
- `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()` erstellt werden, um ihre Bewegung im Backplot zu visualisieren.
    Siehe `scaragui.py` für ein Beispiel, wie man die Werkzeugspitze mit einem Werkzeug und das Werkzeug mit dem Modell verbindet.
  - 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.

13.7.8 Grundstruktur eines Vismach-Skripts.

```
#imports
from vismach import *
import hal
# Erstellen der HAL Komponenten and Pins
comp = hal.component("compname")
comp.newpin("pin_name", hal.HAL_FLOAT, hal.HAL_IN)
...
# Erstellen von 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)
...
# Erstellen des übergeordneten (engl. top-level) Modells
model = Collection([base, saddle, head, carousel])
# Start der Visualisierung
main(model, tooltip, work, 100, lat=-75, lon=215)
```

Teil III

Glossar, Copyright & Geschichte

Kapitel 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.

Kapitel 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.

Achse (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, 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, 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, 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.

Comp
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 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 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 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
Eine Zahl, die einen Dezimalpunkt hat, bsw. 12.3. In HAL wird sie (engl.) als Float bezeichnet.

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. 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, 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 Joggin 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
Siehe Echtzeit.

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
Eine 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, muss unter Linux ein Echtzeit-Kernel wie RTAI installiert und die Software für die Ausführung in der speziellen Echtzeitumgebung erstellt werden. Aus diesem Grund läuft Echtzeit-Software im Kernel-Space.

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
Eine ganze Zahl, die ein positives oder negatives Vorzeichen haben kann. In HAL wird sie als s32 bezeichnet. (Eine 32-Bit-Ganzzahl mit Vorzeichen hat einen nutzbaren Bereich von -2.147.483.647 bis +2.147.483.647.)

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
Eine ganze Zahl, die kein Vorzeichen hat. In HAL wird sie als u32 bezeichnet. (Eine vorzeichenlose 32-Bit-Ganzzahl hat einen nutzbaren Bereich von Null bis 4.294.967.296.)

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.

Kapitel 16

Copyright

16.1 Juristischer Abschnitt

Die Übersetzungen dieser Datei im Quellbaum sind nicht rechtsverbindlich.

16.1.1 Copyright-Bedingungen

16.1.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

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. MODIFICATIONS

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:

A. 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. TRANSLATION

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. TERMINATION

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. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://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: How to use this License for your documents

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.

Kapitel 17

LinuxCNC Geschichte

17.1 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 http://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 http://linuxcnc.org/files/Use-of-Open-Source-Distribution-for-a-Machine-Tool-Controller.pdf verfügbar

Kapitel 18

Index