# LinuxCNC V2.9.10, 02 Jul 2026

# Inhaltsverzeichnis

**I Erste Schritte & Konfiguration 1**

**1 Erste Schritte mit LinuxCNC 2**

1.1 Über LinuxCNC . . . 2
1.1.1 Die Software . . . 2
1.1.2 Das Betriebssystem . . . 2
1.1.3 Getting Help . . . 3
1.1.3.1 Web-Forum . . . 3
1.1.3.2 IRC . . . 3
1.1.3.3 Mailingliste . . . 3
1.1.3.4 LinuxCNC-Wiki . . . 4
1.1.3.5 Fehlerberichte . . . 4
1.2 System Requirements . . . 4
1.2.1 Mindestanforderungen . . . 4
1.2.2 Kernel- und Versionsanforderungen . . . 5
1.2.2.1 Preempt-RT mit dem Paket *linuxcnc-uspace* . . . 5
1.2.2.2 RTAI mit *linuxcnc*-Paket . . . 5
1.2.2.3 Xenomai mit *linuxcnc-uspace* Paket . . . 5
1.2.2.4 RTAI mit *linuxcnc-uspace*-Paket . . . 5
1.2.3 Problematische Hardware . . . 5
1.2.3.1 Laptops . . . 5
1.2.3.2 Videokarten . . . 6
1.3 Getting LinuxCNC . . . 6
1.3.1 Das Festplattenabbild (engl. kurz image) herunterladen . . . 6
1.3.1.1 Normales Herunterladen . . . 6
1.3.1.2 Herunterladen mit zsync . . . 7
1.3.1.3 Überprüfen des Abbilds . . . 7
1.3.2 Schreiben des Abbilds auf ein bootfähiges Gerät . . . 7
1.3.2.1 Raspberry Pi Image . . . 8

1.3.2.2 AMD-64 (x86-64, PC) Image using GUI tools . . . 8
1.3.2.3 Command line - Linux . . . 8
1.3.2.4 Command line - MacOS . . . 8
1.3.3 Testen von LinuxCNC . . . 9
1.3.4 LinuxCNC installieren . . . 9
1.3.5 Updates to LinuxCNC . . . 10
1.3.6 Install Problems . . . 10
1.3.7 Alternate Install Methods . . . 10
1.3.7.1 Installing on Debian Trixie (with Preempt-RT kernel) . . . 11
1.3.7.2 Installing on Debian Trixie (with experimental RTAI kernel) . . . 12
1.3.7.3 Installing on Raspbian 12 . . . 12
1.4 Running LinuxCNC . . . 12
1.4.1 Aufrufen von LinuxCNC . . . 12
1.4.2 Configuration Launcher . . . 13
1.4.3 Nächste Schritte für die Konfiguration . . . 15
1.4.4 Simulator-Konfigurationen . . . 15
1.4.5 Konfigurationsressourcen . . . 16
1.5 Updating LinuxCNC . . . 16
1.5.1 Upgrade auf die neue Version . . . 16
1.5.1.1 Apt Sources Konfiguration . . . 17
1.5.1.2 Upgrade auf die neue Version . . . 19
1.5.1.3 Ubuntu . . . 20
1.5.2 Aktualisieren ohne Netzwerk . . . 20
1.5.3 Updating Configuration Files for 2.9 . . . 20
1.5.3.1 Stricter handling of pluggable interpreters . . . 20
1.5.3.2 Canterp . . . 21
1.5.3.3 Spindle limits in the INI . . . 21
1.5.4 New HAL components . . . 21
1.5.4.1 Non-Realtime . . . 21
1.5.4.2 Echtzeit . . . 21
1.5.5 New Drivers . . . 21
1.6 Linux FAQ . . . 22
1.6.1 Automatic Login . . . 22
1.6.1.1 Debian . . . 22
1.6.1.2 Ubuntu . . . 22
1.6.2 Automatisches Starten . . . 22
1.6.3 Terminal . . . 23
1.6.4 Man Pages . . . 23
1.6.5 Module auflisten . . . 23

1.6.6 Bearbeiten einer root-Datei . . . 23
1.6.6.1 Der Weg über die Befehlszeile . . . 24
1.6.6.2 Der GUI-Weg . . . 24
1.6.6.3 Root-Zugriff . . . 24
1.6.7 Terminal-Befehle . . . 24
1.6.7.1 Arbeitsverzeichnis . . . 24
1.6.7.2 Changing Directories . . . 24
1.6.7.3 Auflisten von Dateien in einem Verzeichnis . . . 25
1.6.7.4 Suchen einer Datei . . . 25
1.6.7.5 Suche nach Text . . . 25
1.6.7.6 Diagnosemeldungen . . . 26
1.6.8 Bequemlichkeiten . . . 26
1.6.8.1 Terminal Launcher . . . 26
1.6.9 Hardware-Probleme . . . 26
1.6.9.1 Hardware-Informationen . . . 26
1.6.9.2 Monitor-Auflösung . . . 26
1.6.10 Pfade . . . 27

**2 Allgemeine Benutzerinformationen 28**
2.1 User Foreword . . . 28
2.2 LinuxCNC User Introduction . . . 29
2.2.1 Einführung . . . 29
2.2.2 How LinuxCNC Works . . . 29
2.2.3 Graphical User Interfaces . . . 31
2.2.4 Benutzerschnittstellen . . . 39
2.2.5 Virtuelle Schalttafeln . . . 39
2.2.6 Sprachen . . . 42
2.2.7 Think Like a CNC Operator . . . 42
2.2.8 Modes of Operation . . . 43
2.3 Important User Concepts . . . 43
2.3.1 Trajectory Control . . . 43
2.3.1.1 Trajectory Planning . . . 43
2.3.1.2 Path Following . . . 44
2.3.1.3 Programming the Planner . . . 44
2.3.1.4 Planning Moves . . . 45
2.3.2 G-Code . . . 46
2.3.2.1 Defaults . . . 46
2.3.2.2 Feed Rate . . . 46
2.3.2.3 Tool Radius Offset . . . 46

2.3.3 Referenzfahrt (engl. homing) . . . 46
2.3.4 Tool Changes . . . 46
2.3.5 Koordinatensysteme . . . 46
2.3.5.1 G53 Machine Coordinate . . . 47
2.3.5.2 G54-59.3 User Coordinates . . . 47
2.3.5.3 When You Are Lost . . . 47
2.3.6 Machine Configurations . . . 47
2.4 Starting LinuxCNC . . . 49
2.4.1 Running LinuxCNC . . . 49
2.4.1.1 Configuration Selector . . . 50
2.5 CNC Machine Overview . . . 51
2.5.1 Mechanische Bestandteile . . . 51
2.5.1.1 Axes . . . 51
2.5.1.2 Spindel . . . 52
2.5.1.3 Kühlmittel . . . 52
2.5.1.4 Feed and Speed Override . . . 52
2.5.1.5 Schalter zum Löschen von Blöcken . . . 52
2.5.1.6 Optionaler Programm-Stopp-Schalter . . . 52
2.5.2 Steuerungs- und Datenkomponenten . . . 53
2.5.2.1 Lineare Achsen . . . 53
2.5.2.2 Rotationsachsen . . . 53
2.5.2.3 Controlled Point . . . 53
2.5.2.4 Koordinierte lineare Bewegung . . . 53
2.5.2.5 Feed Rate . . . 54
2.5.2.6 Cooling . . . 54
2.5.2.7 Verweilen (engl. dwell) . . . 54
2.5.2.8 Einheiten . . . 54
2.5.2.9 Aktuelle Position . . . 54
2.5.2.10 Ausgewählte Ebene . . . 55
2.5.2.11 Werkzeug-Karussell . . . 55
2.5.2.12 Werkzeugwechsel . . . 55
2.5.2.13 Paletten-Shuttle . . . 55
2.5.2.14 Geschwindigkeits-Neufestsetzung (engl. override) . . . 55
2.5.2.15 Path Control Mode . . . 55
2.5.3 Interpreter-Interaktion mit Schaltern . . . 55
2.5.3.1 Feed and Speed Override Switches . . . 56
2.5.3.2 Schalter zum Löschen von Blöcken . . . 56
2.5.3.3 Optionaler Programm-Stopp-Schalter . . . 56
2.5.4 Werkzeugtabelle . . . 56

2.5.5 Parameter . . . 56
2.6 Lathe User Information . . . 57
2.6.1 Drehbank-Modus . . . 57
2.6.2 Lathe Tool Table . . . 58
2.6.3 Lathe Tool Orientation . . . 58
2.6.4 Werkzeug Touch Off . . . 60
2.6.4.1 X Touch Off . . . 60
2.6.4.2 Z Touch-Off . . . 60
2.6.4.3 Der Z-Maschinenversatz (engl. machine offset) . . . 61
2.6.5 Spindelsynchronisierte Bewegung . . . 61
2.6.6 Bögen . . . 62
2.6.6.1 Bögen und Drehmaschinendesign . . . 62
2.6.6.2 Radius & Durchmesser-Modus . . . 62
2.6.7 Werkzeugpfad . . . 62
2.6.7.1 Kontrollpunkt . . . 62
2.6.7.2 Schneidwinkel ohne Fräser Compensation . . . 63
2.6.7.3 Schneiden eines Radius . . . 64
2.6.7.4 Verwenden der Fräser (engl. cutter)-Kompensation . . . 66
2.7 Plasma Cutting Primer for LinuxCNC Users . . . 66
2.7.1 Was ist Plasma? . . . 66
2.7.2 Bogen-Initialisierung . . . 67
2.7.2.1 Hochfrequenzstart . . . 67
2.7.2.2 Blowback Start . . . 68
2.7.3 CNC-Plasma . . . 68
2.7.4 Auswahl einer Plasmamaschine für CNC-Bearbeitungen . . . 69
2.7.5 Arten der Brennerhöhensteuerung . . . 70
2.7.6 Lichtbogen-OK-Signal . . . 71
2.7.7 Erfassung der Anfangshöhe . . . 71
2.7.7.1 Gleitende Schalter (engl. float switches) . . . 71
2.7.7.2 Ohmsche Erfassung . . . 72
2.7.7.3 Hypersensing with a MESA THCAD-5 . . . 72
2.7.7.4 Beispiel HAL-Code für Hypersensing . . . 73
2.7.8 THC-Verzögerung . . . 74
2.7.9 Abtastung der Brennerspannung . . . 74
2.7.10 Brenner Behinderung (engl. torch breakaway) . . . 75
2.7.11 Corner Lock / Velocity Anti-Dive . . . 75
2.7.12 Void / Kerf Crossing . . . 75
2.7.13 Hole And Small Shape Cutting . . . 75
2.7.14 I/O Pins For Plasma Controllers . . . 76

2.7.14.1Arc OK (input) . . . 77
2.7.14.2Brenner an (Ausgang) . . . 77
2.7.14.3Gleitender Schalter (engl. float switch) (input) . . . 77
2.7.14.4Ohmscher Sensor aktivieren (Ausgang) . . . 77
2.7.14.5Ohmsche Sensorik (engl. ohmic sensing) (Eingang) . . . 78
2.7.14.6Brenner-Abreißsensor (engl. torch breakaway sensor) . . . 78
2.7.15G-code For Plasma Controllers . . . 78
2.7.15.1Aktivieren/Deaktivieren des THC-Betriebs: . . . 78
2.7.16External Offsets and Plasma Cutting . . . 79
2.7.17Reading Arc Voltage With The Mesa THCAD . . . 80
2.7.17.1THCAD Connections . . . 80
2.7.17.2THCAD Initial Testing . . . 80
2.7.17.3Which Model THCAD To Use? . . . 81
2.7.18Post Processors And Nesting . . . 81
2.7.19Designing For Noisy Electrical Environments . . . 82
2.7.20Water Tables . . . 82
2.7.21Downdraft Tables . . . 83
2.7.22Designing For Speed And Acceleration . . . 83
2.7.23Distance Travelled Per Motor Revolution . . . 83
2.7.24QtPlasmaC LinuxCNC Plasma Configuration . . . 83
2.7.25Hypertherm RS485 Control . . . 83
2.7.26Post Processors For Plasma Cutting . . . 84

**3 Konfigurationsassistenten 85**

3.1 Stepper Configuration Wizard . . . 85
3.1.1 Einführung . . . 85
3.1.2 Startseite . . . 86
3.1.3 Grundlegende Informationen . . . 87
3.1.4 Latenz-Test . . . 88
3.1.5 Einrichtung der parallelen Schnittstelle . . . 90
3.1.6 Einrichtung des zweiten parallelen Ports . . . 91
3.1.7 Konfiguration der Achsen . . . 92
3.1.7.1 Finding Maximum Velocity . . . 94
3.1.7.2 Finding Maximum Acceleration . . . 95
3.1.8 Spindel-Konfiguration . . . 96
3.1.8.1 Spindle Speed Control . . . 96
3.1.8.2 Spindle-synchronized motion . . . 97
3.1.8.3 Determining Spindle Calibration . . . 97
3.1.9 Optionen . . . 98

3.1.10 Complete Machine Configuration . . . 99
3.1.11 Achsen Verfahrwege und Referenzpunkte . . . 99
3.1.11.1 Betrieb ohne Endschalter . . . 100
3.1.11.2 Operating without Home Switches . . . 100
3.1.11.3 Verdrahtungsoptionen für Referenz- und Endschalter . . . 100
3.2 Mesa-Konfigurationsassistent . . . 101
3.2.1 Schritt für Schritt Anleitung . . . 102
3.2.2 Erstellen oder bearbeiten . . . 102
3.2.3 Grundlegende Informationen zur Maschine . . . 103
3.2.4 Externe Konfiguration . . . 106
3.2.5 GUI-Konfiguration . . . 107
3.2.6 Mesa-Konfiguration . . . 110
3.2.7 Mesa I/O-Einrichtung . . . 112
3.2.8 Konfiguration des parallelen Anschlusses . . . 116
3.2.9 Konfiguration der Achsen . . . 117
3.2.10 Spindel-Konfiguration . . . 125
3.2.11 Weitere Optionen für Fortgeschrittene . . . 126
3.2.12 HAL-Komponenten . . . 127
3.2.13 PnCconf für Fortgeschrittene . . . 129

**4 Konfiguration 131**
4.1 Integrator-Konzepte . . . 131
4.1.1 Dateispeicherorte . . . 131
4.1.1.1 Installiert . . . 131
4.1.1.2 Befehlszeile . . . 132
4.1.2 Dateien . . . 132
4.1.3 Schrittmotor-Systeme (engl. stepper systems) . . . 132
4.1.3.1 Basiszeitraum (engl. base period) . . . 132
4.1.3.2 Schritt-Timing . . . 133
4.1.4 Servosysteme . . . 133
4.1.4.1 Grundbetrieb . . . 133
4.1.4.2 Proportionaler Ausdruck . . . 135
4.1.4.3 Integraler Begriff . . . 135
4.1.4.4 Differenzierender-Anteil (D-Anteil) . . . 136
4.1.4.5 Schleifenabstimmung . . . 136
4.1.4.6 Manuelle Abstimmung . . . 136
4.1.5 RTAI . . . 136
4.1.5.1 ACPI . . . 136
4.1.6 Computer/Machine Interface Hardware Options . . . 137

4.1.6.1 litehm2/rv901t . . . 137
4.2 Latency Testing . . . 137
4.2.1 What is latency? . . . 137
4.2.2 Latency Tests . . . 137
4.2.2.1 Latenz-Test . . . 138
4.2.2.2 Latency Plot . . . 139
4.2.2.3 Latenz-Histogramm . . . 140
4.2.3 Latency tuning . . . 141
4.2.3.1 Tuning the BIOS for latency . . . 141
4.2.3.2 Tuning Preempt-RT for latency . . . 142
4.3 Stepper-Abstimmung . . . 142
4.3.1 Das Beste aus Software Stepping herausholen . . . 142
4.3.1.1 Führen Sie einen Latenztest durch . . . 143
4.3.1.2 Finden Sie heraus, was Ihre Antriebe erwarten . . . 143
4.3.1.3 Wählen Sie Ihren BASE_PERIOD . . . 144
4.3.1.4 Verwenden Sie steplen, stepspace, dirsetup und/oder dirhold . . . 145
4.3.1.5 Nicht raten! . . . 145
4.4 INI Configuration . . . 146
4.4.1 The INI File Components . . . 146
4.4.1.1 Kommentare . . . 146
4.4.1.2 Sections . . . 147
4.4.1.3 Variablen . . . 147
4.4.1.4 Custom Sections and Variables . . . 148
4.4.1.5 Include Files . . . 148
4.4.2 INI File Sections . . . 149
4.4.2.1 [EMC] Section . . . 149
4.4.2.2 [DISPLAY] Section . . . 149
4.4.2.3 [FILTER] Section . . . 153
4.4.2.4 [RS274NGC] Section . . . 155
4.4.2.5 [EMCMOT] Section . . . 156
4.4.2.6 [TASK] Section . . . 157
4.4.2.7 [HAL] section . . . 157
4.4.2.8 [HALUI] section . . . 158
4.4.2.9 [APPLICATIONS] Section . . . 158
4.4.2.10[TRAJ] Section . . . 159
4.4.2.11[KINS] Section . . . 162
4.4.2.12[AXIS_<letter>] Section . . . 162
4.4.2.13[JOINT_<num>] Sections . . . 163
4.4.2.14[SPINDLE_<num>] Section(s) . . . 169

4.4.2.15[EMCIO] Section . . . 170
4.5 Konfiguration der Referenzfahrt (engl. homing) . . . 171
4.5.1 Übersicht . . . 171
4.5.2 Voraussetzung . . . 171
4.5.3 Separater Home-Schalter Beispiel-Layout . . . 172
4.5.4 Gemeinsamer End-/Hauptschalter Beispiel-Layout . . . 173
4.5.5 Referenzfahrt Abfolge . . . 174
4.5.6 Konfiguration . . . 176
4.5.6.1 HOME_SEARCH_VEL . . . 176
4.5.6.2 HOME_LATCH_VEL . . . 176
4.5.6.3 HOME_FINAL_VEL . . . 176
4.5.6.4 HOME_IGNORE_LIMITS . . . 177
4.5.6.5 HOME_USE_INDEX . . . 177
4.5.6.6 HOME_INDEX_NO_ENCODER_RESET . . . 177
4.5.6.7 HOME_OFFSET . . . 177
4.5.6.8 HOME . . . 177
4.5.6.9 HOME_IS_SHARED . . . 178
4.5.6.10HOME_ABSOLUTE_ENCODER . . . 178
4.5.6.11HOME_SEQUENCE . . . 178
4.5.6.12VOLATILE_HOME . . . 179
4.5.6.13LOCKING_INDEXER . . . 179
4.5.6.14Immediate Homing . . . 180
4.5.6.15Inhibiting Homing . . . 180
4.6 I/O Control V2 . . . 181
4.6.1 Beschreibung . . . 181
4.6.2 Anwendung . . . 182
4.6.3 Pins . . . 182
4.6.4 Parameter . . . 183
4.6.5 Kommunikation . . . 183
4.7 Konfiguration der Drehmaschine . . . 184
4.7.1 Standard-Ebene . . . 184
4.7.2 INI-Einstellungen . . . 185
4.8 Stepper Schnellstart . . . 186
4.8.1 Latenz-Test . . . 186
4.8.2 Sherline . . . 186
4.8.3 Xylotex . . . 186
4.8.4 Maschineninformationen . . . 186
4.8.5 Informationen zur Pinbelegung . . . 186
4.8.6 Mechanische Informationen . . . 187

4.9 Stepper Configuration . . . 188
4.9.1 Einführung . . . 188
4.9.2 Maximale Schrittgeschwindigkeit . . . 188
4.9.3 Pinbelegung . . . 189
4.9.3.1 Standard-Pinbelegung HAL . . . 189
4.9.3.2 Übersicht . . . 191
4.9.3.3 Ändern der Datei standard_pinout.hal . . . 191
4.9.3.4 Ändern der Polarität eines Signals . . . 191
4.9.3.5 Hinzufügen einer PWM-Spindeldrehzahlregelung . . . 192
4.9.3.6 Hinzufügen eines Aktivierungssignals (engl. enable) . . . 192
4.9.3.7 Externe NOTAUS (engl, ESTOP)-Taste . . . 192
4.10 Stepper Diagnostics . . . 192
4.10.1 Häufige Probleme . . . 192
4.10.1.1 Stepper bewegt sich einen Schritt . . . 192
4.10.1.2 Keine Stepper bewegen sich . . . 193
4.10.1.3 Abstand nicht korrekt . . . 193
4.10.2 Fehlermeldungen . . . 193
4.10.2.1 Folgender Fehler . . . 193
4.10.2.2 RTAPI-Fehler . . . 193
4.10.3 Testen . . . 194
4.10.3.1 Schritt-Timing . . . 194
4.11 Filter-Programme . . . 195
4.11.1 Einführung . . . 195
4.11.2 Einrichten der INI für Programmfilter . . . 195
4.11.3 Erstellung von Filterprogrammen auf Python-Basis . . . 196

**5 HAL (Hardware Abstraction Layer) 199**

5.1 HAL Introduction . . . 199
5.1.1 HAL Overview . . . 199
5.1.2 Communication . . . 201
5.1.3 HAL System Design . . . 202
5.1.3.1 Part Selection . . . 203
5.1.3.2 Interconnection Design . . . 204
5.1.3.3 Implementation . . . 204
5.1.3.4 Testen . . . 204
5.1.3.5 Summary . . . 204
5.1.4 HAL Concepts . . . 205
5.1.5 HAL components . . . 207
5.1.6 Timing Issues In HAL . . . 207

5.2 HAL Basics . . . 208
5.2.1 HAL Commands . . . 208
5.2.1.1 loadrt . . . 209
5.2.1.2 addf . . . 209
5.2.1.3 loadusr . . . 210
5.2.1.4 net . . . 211
5.2.1.5 setp . . . 212
5.2.1.6 sets . . . 213
5.2.1.7 unlinkp . . . 213
5.2.1.8 Veraltete Befehle . . . 213
5.2.2 HAL Data . . . 214
5.2.2.1 Bit . . . 214
5.2.2.2 Gleitkommazahl (engl. float) . . . 214
5.2.2.3 s32 . . . 214
5.2.2.4 u32 . . . 214
5.2.3 HAL Files . . . 214
5.2.4 HAL Parameter . . . 215
5.2.5 Basic Logic Components . . . 216
5.2.5.1 and2 . . . 216
5.2.5.2 not . . . 216
5.2.5.3 or2 . . . 217
5.2.5.4 xor2 . . . 217
5.2.6 Logic Examples . . . 218
5.2.7 Conversion Components . . . 218
5.2.7.1 weighted_sum . . . 218
5.3 HAL TWOPASS . . . 220
5.3.1 TWOPASS . . . 220
5.3.2 Post GUI (lat. für nach dem GUI auszuführen) . . . 222
5.3.3 Ausschließen von HAL-Dateien . . . 222
5.3.4 Beispiele . . . 223
5.4 HAL Tutorial . . . 223
5.4.1 Einführung . . . 223
5.4.2 Halcmd . . . 223
5.4.2.1 Notation . . . 224
5.4.2.2 Befehl-Vervollständigung durch Tabulator-Taste . . . 224
5.4.2.3 Die RTAPI-Umgebung . . . 224
5.4.3 Ein einfaches Beispiel . . . 224
5.4.3.1 Laden einer Komponente . . . 224
5.4.3.2 Untersuchung der HAL . . . 225

5.4.3.3 Echtzeitcode zum Laufen bringen . . . 226
5.4.3.4 Ändern von Parametern . . . 228
5.4.3.5 Speichern der HAL-Konfiguration . . . 228
5.4.3.6 Halrun beenden . . . 229
5.4.3.7 Wiederherstellung der HAL-Konfiguration . . . 229
5.4.3.8 HAL aus dem Speicher entfernen . . . 229
5.4.4 Halmeter . . . 229
5.4.5 Stepgen Example . . . 232
5.4.5.1 Installieren der Komponenten . . . 232
5.4.5.2 Verbinden von Pins mit Signalen . . . 233
5.4.5.3 Einrichten der Echtzeitausführung - Threads und Funktionen . . . 234
5.4.5.4 Parameter einstellen . . . 235
5.4.5.5 Ausführen! . . . 236
5.4.6 Halscope . . . 236
5.4.6.1 Anschließen der Oszilloskop-Sonden . . . 238
5.4.6.2 Erfassen unserer ersten Wellenformen . . . 241
5.4.6.3 Vertikale Anpassungen . . . 242
5.4.6.4 Triggering (automatisches Auslösen) . . . 243
5.4.6.5 Horizontale Anpassungen . . . 245
5.4.6.6 Weitere Kanäle . . . 246
5.4.6.7 Weitere Samples . . . 247
5.5 HAL Examples . . . 247
5.5.1 Verbinden von zwei Ausgängen . . . 247
5.5.2 Manueller Werkzeugwechsel . . . 248
5.5.3 Geschwindigkeit berechnen . . . 249
5.5.4 Details zum Softstart . . . 250
5.5.5 Stand-Alone HAL . . . 252
5.6 Core Components . . . 253
5.6.1 Bewegung (engl. motion) . . . 253
5.6.1.1 Optionen . . . 254
5.6.1.2 Pins . . . 254
5.6.1.3 Parameter . . . 256
5.6.1.4 Funktionen . . . 256
5.6.2 Spindel . . . 256
5.6.2.1 Pins . . . 257
5.6.3 Achs- und Gelenkpins und Parameter . . . 258
5.6.4 iocontrol . . . 258
5.6.4.1 Pins . . . 258
5.6.5 INI-Einstellungen . . . 259

5.6.5.1 Pins . . . 259
5.7 HAL Component List . . . 260
5.7.1 Komponenten . . . 260
5.7.1.1 User Interfaces (non-realtime) . . . 260
5.7.1.2 Motion (non-realtime) . . . 261
5.7.1.3 Hardware-Treiber . . . 261
5.7.1.4 Mesa und andere I/O-Karten (Echtzeit) . . . 261
5.7.1.5 Utilities (non-realtime) . . . 262
5.7.1.6 Signalverarbeitung (Echtzeit) . . . 263
5.7.1.7 Kinematiken (Echtzeit) . . . 264
5.7.1.8 Motion control (Realtime) . . . 265
5.7.1.9 Motor control (Echtzeit) . . . 265
5.7.1.10 Sonstiges (Echtzeit) . . . 265
5.7.2 HAL-API-Aufrufe . . . 266
5.7.3 RTAPI-Aufrufe . . . 267
5.8 Beschreibungen der HAL-Komponenten . . . 268
5.8.1 StepGen . . . 268
5.8.1.1 Pins . . . 270
5.8.1.2 Parameter . . . 271
5.8.1.3 Step Types . . . 271
5.8.1.4 Funktionen . . . 272
5.8.2 PWMgen . . . 273
5.8.2.1 Ausgangstypen (engl. output types) . . . 273
5.8.2.2 Pins . . . 273
5.8.2.3 Parameter . . . 274
5.8.2.4 Funktionen . . . 274
5.8.3 Encoder . . . 275
5.8.3.1 Pins . . . 276
5.8.3.2 Parameter . . . 277
5.8.3.3 Funktionen . . . 277
5.8.4 PID . . . 277
5.8.4.1 Pins . . . 278
5.8.4.2 Funktionen . . . 280
5.8.5 Simulated Encoder . . . 280
5.8.5.1 Pins . . . 280
5.8.5.2 Parameter . . . 281
5.8.5.3 Funktionen . . . 281
5.8.6 Debounce . . . 281
5.8.6.1 Pins . . . 281

5.8.6.2 Parameter . . . 282
5.8.6.3 Funktionen . . . 282
5.8.7 SigGen . . . 282
5.8.7.1 Pins . . . 282
5.8.7.2 Parameter . . . 283
5.8.7.3 Funktionen . . . 283
5.8.8 lut5 . . . 283
5.9 HAL Component Generator . . . 284
5.9.1 Installation . . . 285
5.9.2 Compiling . . . 285
5.9.2.1 Inside the source tree . . . 285
5.9.2.2 Realtime components outside the source tree . . . 285
5.9.2.3 Non-realtime components outside the source tree . . . 286
5.9.3 Verwendung einer Komponente . . . 287
5.9.4 Definitionen . . . 287
5.9.5 Erstellung einer Instanz . . . 287
5.9.6 Implizite Parameter . . . 287
5.9.7 Syntax . . . 287
5.9.7.1 HAL-Funktionen . . . 290
5.9.7.2 Optionen . . . 290
5.9.7.3 Lizenz und Urheberschaft . . . 291
5.9.7.4 Datenspeicherung pro Instanz . . . 291
5.9.7.5 Kommentare . . . 292
5.9.8 Einschränkungen . . . 292
5.9.9 Bequemlichkeits-Makros . . . 292
5.9.10 Komponenten mit einer Funktion . . . 293
5.9.11 Komponenten-Persönlichkeit . . . 293
5.9.12 Beispiele . . . 294
5.9.12.1 Konstante . . . 294
5.9.12.2 sincos . . . 294
5.9.12.3 out8 . . . 294
5.9.12.4 hal_loop . . . 295
5.9.12.5 arraydemo . . . 295
5.9.12.6 rand . . . 296
5.9.12.7 logic (using personality) . . . 296
5.9.12.8 Allgemeine Funktionen . . . 297
5.9.13 Verwendung der Kommandozeile . . . 297
5.10 HALTCL-Dateien . . . 298
5.10.1 Kompatibilität . . . 298

5.10.2 Haltcl-Befehle . . . 298
5.10.3 Haltcl INI-Datei-Variablen . . . 298
5.10.4 Konvertieren von HAL-Dateien in Tcl-Dateien . . . 300
5.10.5 Haltcl Anmerkungen . . . 300
5.10.6 Haltcl Beispiele . . . 300
5.10.7 Haltcl Interaktiv . . . 301
5.10.8 Haltcl-Verteilungsbeispiele (sim) . . . 301
5.11 HAL-Benutzeroberfläche . . . 301
5.11.1 Einführung . . . 301
5.11.2 MDI . . . 302
5.11.3 Beispiel-Konfiguration . . . 302
5.11.4 Halui-Pin-Referenz . . . 302
5.11.4.1 Abbrechen . . . 302
5.11.4.2 Notaus (engl. E-Stop) . . . 303
5.11.4.3 Vorschub Neufestsetzung (engl. override) . . . 303
5.11.4.4 Nebel (engl. mist) . . . 303
5.11.4.5 Flut-Kühlmittel (engl. flood) . . . 303
5.11.4.6 Referenzfahrt (engl. homing) . . . 303
5.11.4.7 Schmiermittel (engl. lube) . . . 303
5.11.4.8 Maschine . . . 304
5.11.4.9 Max. Geschwindigkeit . . . 304
5.11.4.10 MDI . . . 304
5.11.4.11 Gelenk . . . 304
5.11.4.12 Gelenk-Joggen . . . 305
5.11.4.13 Achse . . . 306
5.11.4.14 Achsen-Jogging . . . 306
5.11.4.15 Modus . . . 307
5.11.4.16 Programm . . . 307
5.11.4.17 Eilgang-Override (engl. rapid override) . . . 308
5.11.4.18 Spindel Neufestsetzung (engl. override) . . . 308
5.11.4.19 Spindel . . . 309
5.11.4.20 Werkzeug . . . 309
5.12 Halui Examples . . . 309
5.12.1 Ferngesteuerter Start . . . 310
5.12.2 Pause & Fortsetzen . . . 310
5.13 Creating Non-realtime Python Components . . . 311
5.13.1 Basic usage example . . . 311
5.13.2 Non-realtime components and delays . . . 312
5.13.3 Pins und Parameter erstellen . . . 312

5.13.3.1Ändern des Präfixes . . . 313
5.13.4Lesen und Schreiben von Pins und Parametern . . . 313
5.13.4.1Ansteuerung der Ausgangsstifte (HAL_OUT) . . . 313
5.13.4.2Ansteuerung von bidirektionalen (HAL_IO) Pins . . . 313
5.13.5Beenden . . . 313
5.13.6Hilfreiche Funktionen . . . 314
5.13.7Konstanten . . . 314
5.13.8System-Informationen . . . 314
5.14Canonical Device Interfaces . . . 314
5.14.1Digital Input . . . 315
5.14.1.1Pins . . . 315
5.14.1.2Parameter . . . 315
5.14.1.3Funktionen . . . 315
5.14.2Digital Output . . . 315
5.14.2.1Pins . . . 315
5.14.2.2Parameter . . . 315
5.14.2.3Funktionen . . . 315
5.14.3Analog Input . . . 315
5.14.3.1Pins . . . 315
5.14.3.2Parameter . . . 316
5.14.3.3Funktionen . . . 316
5.14.4Analog Output . . . 316
5.14.4.1Pins . . . 316
5.14.4.2Parameter . . . 316
5.14.4.3Funktionen . . . 316
5.15HAL Tools . . . 317
5.15.1Halcmd . . . 317
5.15.2Halmeter . . . 317
5.15.3Halshow . . . 319
5.15.4Halscope . . . 320
5.15.5Sim-Pin . . . 320
5.15.6simulate_probe (Sonde simulieren) . . . 321
5.15.7HAL Histogramm . . . 322
5.15.8Halreport . . . 323

**6 Hardware-Treiber 326**
6.1 Parallelport-Treiber . . . 326
6.1.1 Laden . . . 327
6.1.2 PCI-Port-Adresse . . . 329
6.1.3 Pins . . . 330
6.1.4 Parameter . . . 330
6.1.5 Funktionen . . . 330
6.1.6 Häufige Probleme . . . 331
6.1.7 DoubleStep verwenden . . . 331
6.1.8 probe_parport . . . 331
6.1.8.1 Installation von probe_parport . . . 331
6.2 AX5214H Driver . . . 332
6.2.1 Installation . . . 332
6.2.2 Pins . . . 332
6.2.3 Parameter . . . 333
6.2.4 Funktionen . . . 333
6.3 General Mechatronics Treiber . . . 333
6.3.1 I/O-Anschlüsse . . . 335
6.3.1.1 Pins . . . 336
6.3.1.2 Parameter . . . 336
6.3.2 Achsen-Anschlüsse . . . 337
6.3.2.1 Achsen-Schnittstellenmodule . . . 337
6.3.2.2 Encoder . . . 338
6.3.2.3 StepGen Modul . . . 341
6.3.2.4 Freigabe- und Fehlersignale . . . 344
6.3.2.5 Achsen-DAC . . . 344
6.3.3 CAN-Bus-Servoverstärker . . . 345
6.3.3.1 Pins . . . 346
6.3.3.2 Parameter . . . 346
6.3.4 Watchdog-Timer . . . 346
6.3.4.1 Pins . . . 346
6.3.4.2 Parameter . . . 347
6.3.5 End-, Referenzpunkt- und Notaus-Schalter . . . 347
6.3.5.1 Pins . . . 348
6.3.5.2 Parameter . . . 348
6.3.6 Status-LEDs . . . 348
6.3.6.1 CAN . . . 348
6.3.6.2 RS485 . . . 349
6.3.6.3 EMC . . . 349

6.3.6.4 Booten . . . 349
6.3.6.5 Fehler . . . 349
6.3.7 RS485 E/A-Erweiterungsmodule . . . 349
6.3.7.1 Relais-Ausgangsmodul . . . 350
6.3.7.2 Digitales Eingangsmodul . . . 351
6.3.7.3 DAC & ADC-Modul . . . 352
6.3.7.4 Teach Pendant Modul . . . 353
6.3.8 Errata . . . 354
6.3.8.1 GM6-PCI-Karte Errata . . . 354
6.4 GS2 VFD Driver . . . 354
6.4.1 Kommandozeilen-Optionen . . . 354
6.4.2 Pins . . . 355
6.4.3 Parameter . . . 356
6.5 HAL Driver for Raspberry Pi GPIO pins . . . 356
6.5.1 Purpose . . . 356
6.5.2 Anwendung . . . 356
6.5.3 Pins . . . 357
6.5.4 Parameter . . . 357
6.5.5 Funktionen . . . 358
6.5.6 Pin-Nummerierung . . . 358
6.5.7 Known Bugs . . . 358
6.6 Generic driver for any GPIO supported by gpiod. . . . 358
6.6.1 Purpose . . . 358
6.6.2 Anwendung . . . 359
6.6.3 Pins . . . 360
6.6.4 Parameter . . . 360
6.6.5 Funktionen . . . 360
6.6.6 Pin Identification . . . 360
6.6.7 Troubleshooting permissions problems. . . . 361
6.6.8 Author . . . 361
6.6.9 Known Bugs . . . 361
6.7 Mesa HostMot2-Treiber . . . 361
6.7.1 Einführung . . . 361
6.7.2 Firmware-Binärdateien . . . 362
6.7.3 Installieren der Firmware . . . 362
6.7.4 Laden von HostMot2 . . . 362
6.7.5 Watchdog . . . 363
6.7.5.1 Pins . . . 363
6.7.5.2 Parameter . . . 363

6.7.6 HostMot2-Funktionen . . . 363
6.7.7 Pinbelegungen . . . 364
6.7.8 PIN-Dateien . . . 365
6.7.9 Firmware . . . 365
6.7.10 HAL-Pins . . . 365
6.7.11 Konfigurationen . . . 366
6.7.12 GPIO . . . 368
6.7.12.1 Pins . . . 368
6.7.12.2 Parameter . . . 369
6.7.13 StepGen . . . 369
6.7.13.1 Pins . . . 369
6.7.13.2 Parameter . . . 370
6.7.13.3 Ausgangsparameter . . . 370
6.7.14 PWMGen . . . 371
6.7.14.1 Pins . . . 371
6.7.14.2 Parameter . . . 371
6.7.14.3 Ausgangsparameter . . . 372
6.7.15 Encoder . . . 372
6.7.15.1 Pins . . . 372
6.7.15.2 Parameter . . . 372
6.7.16 5I25 Konfiguration . . . 373
6.7.16.1 Firmware . . . 373
6.7.16.2 Konfiguration . . . 373
6.7.16.3 SSERIAL-Konfiguration . . . 374
6.7.16.4 7I77 Grenzwerte . . . 374
6.7.17 Beispielkonfigurationen . . . 374
6.8 MB2HAL . . . 374
6.8.1 Einführung . . . 374
6.8.2 Anwendung . . . 375
6.8.3 Optionen . . . 375
6.8.3.1 Init-Abschnitt . . . 375
6.8.3.2 Transaktionsabschnitte . . . 376
6.8.3.3 Fehlercodes . . . 377
6.8.4 Beispiel Konfigurationsdatei . . . 378
6.8.5 Pins . . . 383
6.8.5.1 fnct_01_read_coils . . . 383
6.8.5.2 fnct_02_read_discrete_inputs . . . 383
6.8.5.3 fnct_03_read_holding_registers . . . 383
6.8.5.4 fnct_04_read_input_registers . . . 383

6.8.5.5 fnct_05_write_single_coil . . . 383
6.8.5.6 fnct_06_write_single_register . . . 383
6.8.5.7 fnct_15_write_multiple_coils . . . 384
6.8.5.8 fnct_16_write_multiple_registers . . . 384
6.9 Mitsub VFD-Treiber . . . 384
6.9.1 Kommandozeilen-Optionen . . . 384
6.9.2 Pins . . . 385
6.9.3 HAL-Beispiel . . . 385
6.9.4 Konfigurieren des Mitsubishi VFD für die serielle Nutzung . . . 386
6.9.4.1 Anschließen der seriellen Schnittstelle . . . 386
6.9.4.2 Modbus-Einrichtung . . . 386
6.10 Motenc Treiber . . . 387
6.10.1 Pins . . . 387
6.10.2 Parameter . . . 388
6.10.3 Funktionen . . . 388
6.11 Opto22 Treiber . . . 388
6.11.1 Die Adapterkarte . . . 389
6.11.2 Der Treiber (engl. driver) . . . 389
6.11.3 Pins . . . 389
6.11.4 Parameter . . . 389
6.11.5 FUNKTIONEN . . . 390
6.11.6 Konfigurieren von E/A-Ports (engl. I/O ports) . . . 390
6.11.7 Pin-Nummerierung . . . 391
6.12 Pico-Treiber . . . 391
6.12.1 Kommandozeilen-Optionen . . . 391
6.12.2 Pins . . . 392
6.12.3 Parameter . . . 394
6.12.4 Funktionen . . . 395
6.13 Pluto P-Treiber . . . 395
6.13.1 Allgemeine Informationen . . . 395
6.13.1.1 Anforderungen . . . 395
6.13.1.2 Verbinder . . . 395
6.13.1.3 Physikalische Stifte (engl.+ inzwischen auch deutsch: pins) . . . 396
6.13.1.4 LED . . . 396
6.13.1.5 Power . . . 396
6.13.1.6 PC-Schnittstelle . . . 396
6.13.1.7 Neuerstellung der FPGA-Firmware . . . 397
6.13.1.8 Für weitere Informationen . . . 397
6.13.2 Pluto-Servo . . . 397

6.13.2.1 Pinbelegung . . . 397
6.13.2.2 Input-Latching und Output-Aktualisierung . . . 399
6.13.2.3 HAL-Funktionen, Pins und Parameter . . . 399
6.13.2.4 Kompatible Treiber-Hardware . . . 399
6.13.3 Pluto Step . . . 399
6.13.3.1 Pinbelegung . . . 400
6.13.3.2 Input-Latching und Output-Aktualisierung . . . 400
6.13.3.3 Schritt (engl. Step)-Wellenform-Timings . . . 401
6.13.3.4 HAL-Funktionen, Pins und Parameter . . . 401
6.14 Powermax Modbus-Treiber . . . 401
6.14.1 Pins . . . 402
6.14.2 Beschreibung . . . 402
6.14.3 Referenz: . . . 403
6.15 Servo To Go-Treiber . . . 403
6.15.1 Installation . . . 403
6.15.2 Pins . . . 404
6.15.3 Parameter . . . 404
6.15.4 Funktionen . . . 404
6.16 Shuttle . . . 405
6.16.1 Beschreibung . . . 405
6.16.2 Einrichtung . . . 405
6.16.3 Pins . . . 405
6.17 VFS11 VFD-Treiber . . . 406
6.17.1 Kommandozeilen-Optionen . . . 406
6.17.2 Pins . . . 407
6.17.3 Parameter . . . 408
6.17.4 INI-Datei-Konfiguration . . . 408
6.17.5 HAL-Beispiel . . . 409
6.17.6 Bedienung des Panels . . . 410
6.17.7 Reaktion auf Fehler (engl. error recovery) . . . 410
6.17.8 Konfigurieren des VFS11 VFD für die Modbus-Nutzung . . . 410
6.17.8.1 Anschließen der seriellen Schnittstelle . . . 410
6.17.8.2 Modbus-Einrichtung . . . 411
6.17.9 Hinweis zur Programmierung . . . 411

**7 Hardware-Beispiele 412**
7.1 PCI-Parallelport . . . 412
7.2 Spindelsteuerung . . . 413
7.2.1 0-10 Volt Spindle Speed . . . 413
7.2.2 PWM Spindle Speed . . . 413
7.2.3 Spindle Enable . . . 413
7.2.4 Spindle Direction . . . 414
7.2.5 Spindle Soft Start . . . 414
7.2.6 Spindel-Feedback . . . 415
7.2.6.1 Spindelsynchronisierte Bewegung . . . 415
7.2.6.2 Spindle At Speed . . . 416
7.3 MPG Handgeräte . . . 416
7.4 GS2 Spindel . . . 419
7.4.1 Beispiel . . . 419

**8 ClassicLadder 420**
8.1 ClassicLadder Einführung . . . 420
8.1.1 Geschichte . . . 420
8.1.2 Einführung . . . 420
8.1.3 Beispiel . . . 421
8.1.4 Grundlegende selbsthaltende Ein-Aus-Schaltung . . . 422
8.2 ClassicLadder Programming . . . 423
8.2.1 SPS / Kontaktplan Konzepte . . . 423
8.2.2 Sprachen . . . 423
8.2.3 Komponenten . . . 423
8.2.3.1 Dateien . . . 423
8.2.3.2 Echtzeit-Modul . . . 424
8.2.3.3 Variablen . . . 424
8.2.4 Loading the ClassicLadder non-realtime module . . . 425
8.2.5 ClassicLadder GUI . . . 425
8.2.5.1 Sektions-Manager . . . 426
8.2.5.2 Abschnittsanzeige . . . 426
8.2.5.3 Die Variablenfenster . . . 428
8.2.5.4 Symbol-Fenster . . . 430
8.2.5.5 Das Editor-Fenster . . . 431
8.2.5.6 Konfigurationsfenster . . . 432
8.2.6 SPS Objekte . . . 434
8.2.6.1 KONTAKTE . . . 434
8.2.6.2 IEC-TIMER . . . 434

8.2.6.3 ZEITGLIEDER (engl. timers) . . . 435
8.2.6.4 KIPPSTUFEN (engl. monostables) . . . 435
8.2.6.5 ZÄHLER (engl. counters) . . . 436
8.2.6.6 VERGLEICHEN . . . 436
8.2.6.7 VARIABLENZUWEISUNG . . . 437
8.2.6.8 SPULEN (engl. coils) . . . 439
8.2.7 ClassicLadder Variablen . . . 440
8.2.8 GRAFCET (State Machine) Programmierung . . . 441
8.2.9 Modbus . . . 443
8.2.10 MODBUS-Einstellungen . . . 447
8.2.10.1 MODBUS-Info . . . 447
8.2.10.2 Kommunikationsfehler . . . 447
8.2.11 Fehlersuche bei Modbus-Problemen . . . 448
8.2.11.1 Anfrage . . . 449
8.2.11.2 Fehlerreaktion . . . 450
8.2.11.3 Datenantwort . . . 451
8.2.11.4 MODBUS-Fehler . . . 452
8.2.12 Einrichten von ClassicLadder . . . 452
8.2.12.1 Hinzufügen der Module . . . 452
8.2.12.2 Hinzufügen der Kontaktplanlogik . . . 453
8.3 ClassicLadder Beispiele . . . 458
8.3.1 Umlaufender (engl. wrapping) Zähler . . . 458
8.3.2 Extra-Impulse zurückweisen . . . 459
8.3.3 Externer Notaus . . . 460
8.3.4 Beispiel für Timer/Bedienung . . . 463

**9 Fortgeschrittene Themen 465**
9.1 Kinematik . . . 465
9.1.1 Einführung . . . 465
9.1.1.1 Gelenke(engl. joints) vs. Achsen (engl. axes) . . . 465
9.1.2 Triviale Kinematik . . . 466
9.1.3 Nicht-triviale Kinematik . . . 467
9.1.3.1 Vorwärts-Transformation . . . 468
9.1.3.2 Inverse Transformation . . . 469
9.1.4 Details zur Implementierung . . . 469
9.1.4.1 Kinematikmodul unter Verwendung der Vorlage userkins.comp . . . 471
9.2 Setting up ”modified” Denavit-Hartenberg (DH) parameters for *genserkins* . . . 471
9.2.1 Vorspiel . . . 471
9.2.2 Allgemeines . . . 471

9.2.3 Modifizierte DH-Parameter . . . 472
9.2.4 Modifizierte DH-Parameter, wie sie in *Genserkins* verwendet werden . . . 472
9.2.5 Nummerierung der Verbindungen und Parameter . . . 473
9.2.6 Wie fange ich an? . . . 473
9.2.7 Sonderfälle . . . 473
9.2.8 Detailliertes Beispiel (RV-6SL) . . . 473
9.2.9 Danksagungen . . . 492
9.3 5-Axis Kinematics . . . 492
9.3.1 Einführung . . . 492
9.3.2 5-Achsen-Werkzeugmaschinen-Konfigurationen . . . 492
9.3.3 Werkzeugausrichtung und -position . . . 492
9.3.4 Translations- und Rotationsmatrizen . . . 493
9.3.5 Tisch Dreh-/Schwenkkonfigurationen mit 5 Achsen (engl. Table Rotary/Tilting 5-Axis Configurations) . . . 494
9.3.5.1 Transformationen für eine xyzac-trt-Werkzeugmaschine mit Werkstückversatz . . . 497
9.3.5.2 Transformationen für eine xyzac-trt-Maschine mit Drehachsenverschiebungen . . . 501
9.3.5.3 Transformationen für eine xyzbc-trt-Maschine mit Drehachsenverschiebungen . . . 504
9.3.6 Beispiele für Dreh-/Kipptische . . . 507
9.3.6.1 Vismach Simulationsmodelle . . . 507
9.3.6.2 Werkzeuglängenkompensation . . . 507
9.3.7 Kundenspezifische Kinematik-Komponenten . . . 508
9.3.8 Abbildungen . . . 509
9.3.9 VERWEISE . . . 511
9.4 Schaltbare Kinematik (switchkins) . . . 511
9.4.1 Einführung . . . 511
9.4.2 Schaltbare Kinematik-Module . . . 512
9.4.2.1 Identitätsbrief-Zuweisungen . . . 512
9.4.2.2 Rückwärtskompatibilität . . . 513
9.4.3 HAL-Pins . . . 513
9.4.3.1 HAL Pin Summary . . . 513
9.4.4 Anwendung . . . 513
9.4.4.1 HAL-Verbindungen . . . 513
9.4.4.2 G-/M-Code-Befehle . . . 514
9.4.4.3 INI file limit settings . . . 514
9.4.4.4 Überlegungen zum Offset . . . 516
9.4.5 Simulationskonfigurationen . . . 516
9.4.6 Kinematische Bestimmungen des Benutzers . . . 517

9.4.7 Warnungen . . . 517
9.4.8 Code Anmerkungen . . . 517
9.5 PID Tuning . . . 518
9.5.1 PID-Regler (engl. PID controller) . . . 518
9.5.1.1 Grundlagen des Regelkreises . . . 518
9.5.1.2 Theorie . . . 519
9.5.1.3 Schleifenabstimmung (engl. loop tuning) . . . 519
9.6 Neuzuordnung (engl. remap) für das Erweitern von G-Code . . . 520
9.6.1 Einführung: Erweiterung des RS274NGC-Interpreters durch Remapping von Codes 520
9.6.1.1 Eine Definition: Neuzuordnung von Codes . . . 520
9.6.1.2 Warum sollten Sie den RS274NGC Interpreter erweitern? . . . 521
9.6.2 Erste Schritte . . . 522
9.6.2.1 Integrierte Neuzuordnungen . . . 522
9.6.2.2 Auswahl eines Codes . . . 523
9.6.2.3 Handhabung der Parameter . . . 523
9.6.2.4 Handhabung der Ergebnisse . . . 524
9.6.2.5 Ausführungsreihenfolge . . . 524
9.6.2.6 Ein minimales Beispiel für neu zugeordneten Code . . . 524
9.6.3 Neuzuordnung konfigurieren . . . 525
9.6.3.1 Die REMAP-Anweisung . . . 525
9.6.3.2 Useful REMAP option combinations . . . 526
9.6.3.3 The argspec parameter . . . 526
9.6.4 Aktualisieren einer bestehenden Konfiguration für die Neuzuordnung . . . 530
9.6.5 Codes für den Wechsel des Remapping-Werkzeugs: T, M6, M61 . . . 530
9.6.5.1 Übersicht . . . 530
9.6.5.2 Verstehen der Rolle von ”iocontrol” mit neu zugeordneten Werkzeugwechselcodes . . . 531
9.6.5.3 Specifying the M6 replacement . . . 532
9.6.5.4 Configuring `iocontrol` with a remapped M6 . . . 534
9.6.5.5 Writing the change and prepare O-word procedures . . . 534
9.6.5.6 Making minimal changes to the built in codes, including `M6` . . . 534
9.6.5.7 Specifying the T (prepare) replacement . . . 535
9.6.5.8 Fehlerbehandlung: Umgang mit Abbrüchen . . . 536
9.6.5.9 Fehlerbehandlung: Fehlschlagen einer NGC-Prozedur mit neu zugeordnetem Code . . . 537
9.6.6 Umschlüsselung anderer bestehender Codes: S, M0, M1, M60 . . . 538
9.6.6.1 Automatic gear selection be remapping S (set spindle speed) . . . 538
9.6.6.2 Anpassen des Verhaltens von M0, M1, M60 . . . 538
9.6.7 Creating new G-code cycles . . . 538

9.6.8 Embedded Python konfigurieren . . . 539
9.6.8.1 Python plugin : INI file configuration . . . 539
9.6.8.2 Executing Python statements from the interpreter . . . 540
9.6.9 Programming Embedded Python in the RS274NGC Interpreter . . . 540
9.6.9.1 The Python plugin namespace . . . 540
9.6.9.2 Der Interpreter aus der Sicht von Python . . . 540
9.6.9.3 Die Interpreterfunktionen __init__ und __delete__ . . . 540
9.6.9.4 Calling conventions: NGC to Python . . . 541
9.6.9.5 Aufrufkonventionen: Python zu NGC . . . 544
9.6.9.6 Eingebaute Module . . . 546
9.6.10 Hinzufügen vordefinierter benannter Parameter . . . 546
9.6.11 Standardmäßige Glue (Programmierer-Slang für verbindende)-Routinen . . . 546
9.6.11.1 T: prepare_prolog and prepare_epilog . . . 547
9.6.11.2 M6: change_prolog and change_epilog . . . 547
9.6.11.3 G-Code-Zyklen: cycle_prolog und cycle_epilog . . . 548
9.6.11.4 S (Set Speed) : setspeed_prolog and setspeed_epilog . . . 549
9.6.11.5 F (Set Feed) : setfeed_prolog and setfeed_epilog . . . 549
9.6.11.6 M61 Set tool number : settool_prolog and settool_epilog . . . 549
9.6.12 Remapped code execution . . . 549
9.6.12.1 NGC procedure call environment during remaps . . . 549
9.6.12.2 Nested remapped codes . . . 549
9.6.12.3 Sequence number during remaps . . . 550
9.6.12.4 Debugging-Flags . . . 550
9.6.12.5 Fehlersuche in eingebettetem Python-Code . . . 550
9.6.13 Axis Preview and Remapped code execution . . . 551
9.6.14 Remappable Codes . . . 552
9.6.14.1 Existing codes which can be remapped . . . 552
9.6.14.2 Currently unallocated G-codes: . . . 552
9.6.14.3 Derzeit nicht zugewiesene M-Codes: . . . 555
9.6.14.4 Vorauslesezeit und Ausführungszeit . . . 556
9.6.14.5 Plugin/Pickle-Hack . . . 556
9.6.14.6 Modul, Methoden, Klassen, usw. Referenz . . . 556
9.6.15 Einführung: Erweiterung der Task-Ausführung . . . 556
9.6.15.1 Warum sollten Sie die Task-Ausführung ändern wollen? . . . 556
9.6.15.2 Ein Diagramm: task, interp, iocontrol, UI (??) . . . 556
9.6.16 Modelle der Aufgaben (engl. task) -Ausführung . . . 556
9.6.16.1 Traditionelle Ausführung von iocontrol/iocontrolv2 . . . 557
9.6.16.2 IO-Verfahren neu definieren . . . 557
9.6.16.3 Python-Prozeduren zur Ausführungszeit . . . 557

9.6.17 Eine kurze Übersicht über die LinuxCNC-Programmausführung . . . 557
9.6.17.1 Zustand des Interpreters . . . 557
9.6.17.2 Task and Interpreter interaction, Queuing and Read-Ahead . . . 557
9.6.17.3 Predicting the machine position . . . 558
9.6.17.4 Queue-busters break position prediction . . . 558
9.6.17.5 How queue-busters are dealt with . . . 558
9.6.17.6 Word order and execution order . . . 559
9.6.17.7 Parsen (engl. parsing, für die Interpretation des Quellcodes) . . . 559
9.6.17.8 Ausführung . . . 559
9.6.17.9 Prozedurausführung . . . 559
9.6.17.10 Wie der Werkzeugwechsel derzeit funktioniert . . . 559
9.6.17.11 How T*x* (Prepare Tool) works . . . 560
9.6.17.12 How M6 (Change tool) works . . . 560
9.6.17.13 How M61 (Change tool number) works . . . 562
9.6.18 Status . . . 562
9.6.19 Changes . . . 562
9.6.20 Debugging . . . 562
9.7 Moveoff Component . . . 563
9.7.1 Ändern einer bestehenden Konfiguration . . . 564
9.8 Eigenständiger Interpreter . . . 567
9.8.1 Anwendung . . . 567
9.8.2 Beispiel . . . 568
9.9 External Axis Offsets . . . 568
9.9.1 INI File Settings . . . 568
9.9.2 HAL-Pins . . . 569
9.9.2.1 Per-Axis Motion HAL Pins . . . 569
9.9.2.2 Other Motion HAL Pins . . . 569
9.9.3 Anwendung . . . 569
9.9.3.1 Offset-Berechnung . . . 569
9.9.3.2 Maschine aus/Maschine ein . . . 569
9.9.3.3 Weiche Grenzwerte . . . 570
9.9.3.4 Anmerkungen . . . 570
9.9.3.5 Warnung . . . 571
9.9.4 Related HAL Components . . . 571
9.9.4.1 eoffset_per_angle.comp . . . 571
9.9.5 Testen . . . 571
9.9.6 Beispiele . . . 572
9.9.6.1 eoffsets.ini . . . 572
9.9.6.2 jwp_z.ini . . . 572

9.9.6.3 dynamische_offsets.ini . . . 572
9.9.6.4 opa.ini (eoffset_per_angle) . . . 573
9.10 Tool Database Interface . . . 573
9.10.1 Interface . . . 573
9.10.1.1 INI-Datei Einstellungen . . . 573
9.10.1.2 **db_program** operation (v2.1) . . . 574
9.10.1.3 Anwendung . . . 575
9.10.1.4 Example program . . . 575
9.10.1.5 Python tooldb module . . . 576
9.10.2 Simulationskonfigurationen . . . 576
9.10.2.1 Anmerkungen . . . 577

**II Anwendung 578**

**10 Benutzerschnittstellen 579**
10.1 AXIS GUI . . . 579
10.1.1 Erste Schritte . . . 580
10.1.1.1 INI-Einstellungen . . . 581
10.1.1.2 Eine typische Sitzung . . . 581
10.1.2 AXIS Fenster . . . 582
10.1.2.1 Menüpunkte . . . 582
10.1.2.2 Schaltflächen der Symbolleiste . . . 586
10.1.2.3 Grafischer Anzeigebereich . . . 587
10.1.2.4 Textanzeigebereich . . . 589
10.1.2.5 Manuelle Steuerung . . . 589
10.1.2.6 MDI . . . 592
10.1.2.7 Vorschub Neufestsetzung (engl. override) . . . 592
10.1.2.8 Spindeldrehzahl-Anpassung . . . 592
10.1.2.9 Jog-Geschwindigkeit . . . 593
10.1.2.10 Max. Geschwindigkeit . . . 593
10.1.3 Tastatursteuerung . . . 593
10.1.3.1 Vorschub-Neufestsetzung (engl. override)-Tasten . . . 593
10.1.4 Show LinuxCNC Status (`linuxcnctop`) . . . 594
10.1.5 MDI-Schnittstelle . . . 595
10.1.6 axis-remote . . . 596
10.1.7 Manueller Werkzeugwechsel . . . 596
10.1.8 Python modules . . . 596
10.1.9 Using AXIS in Lathe Mode . . . 597
10.1.10 Verwendung von AXIS im Modus Schaumstoffschneiden (engl. foam cutting mode) 600

10.1.1 Erweiterte Konfiguration . . . 601
10.1.1.1 Programm-Filter . . . 602
10.1.1.2 Die X-Ressourcen-Datenbank . . . 603
10.1.1.3 Jogwheel . . . 603
10.1.1.4 ~/.axisrc . . . 603
10.1.1.5 USER_COMMAND_FILE . . . 604
10.1.1.6 user_live_update() . . . 604
10.1.1.7 user_hal_pins() . . . 604
10.1.1.8 Externer Editor . . . 604
10.1.1.9 Virtual Control Panel . . . 604
10.1.1.10 Vorschau Kontrolle . . . 604
10.1.12 Axisui . . . 605
10.1.13 Hinweise zur AXIS-Anpassung . . . 606
10.1.13.1 Die Update-Funktion . . . 606
10.1.13.2 Deaktivieren des Schließen-Dialogs . . . 606
10.1.13.3 Ändern Sie die Textschriftart . . . 606
10.1.13.4 Ändern der Rapid Rate mit Tastenkombinationen . . . 606
10.1.13.5 Lesen der INI-Datei . . . 607
10.1.13.6 Read LinuxCNC Status . . . 607
10.1.13.7 Ändern der aktuellen Ansicht . . . 607
10.1.13.8 Erstellen neuer AXISUI HAL-Pins . . . 607
10.1.13.9 Neue HAL-Komponente und Pins erstellen . . . 608
10.1.13.10 Tabs wechseln mit HAL-Pins . . . 608
10.1.13.11 Hinzufügen einer GOTO Referenzpunkt (engl. Home)-Taste . . . 608
10.1.13.12 Button zum manuellen Rahmen hinzufügen . . . 609
10.1.13.13 Interne Variablen lesen . . . 610
10.1.13.14 Widgets ausblenden . . . 611
10.1.13.15 Ändern eines Labels . . . 611
10.1.13.16 Einen bestehenden Befehl umleiten . . . 611
10.1.13.17 Ändern Sie die DRO-Farbe . . . 611
10.1.13.18 Ändern der Buttons der Werkzeugleiste . . . 611
10.1.13.19 Plotterfarben ändern . . . 612
10.2 GMOCCAPY . . . 613
10.2.1 Einführung . . . 613
10.2.2 Anforderungen . . . 614
10.2.3 How to get GMOCCAPY . . . 614
10.2.4 Basiseinstellung . . . 615
10.2.4.1 Die DISPLAY-Sektion . . . 616
10.2.4.2 Der TRAJ Abschnitt . . . 617

10.2.4.3 Makro-Buttons . . . 618
10.2.4.4 Embedded Tabs and Panels . . . 620
10.2.4.5 User Created Messages . . . 623
10.2.4.6 Vorschau Kontrolle . . . 624
10.2.4.7 User Command File . . . 624
10.2.4.8 User CSS File . . . 625
10.2.4.9 Protokollierung (engl. logging) . . . 625
10.2.5 HAL-Pins . . . 626
10.2.5.1 Right and Bottom Button Lists . . . 626
10.2.5.2 Velocities and Overrides . . . 629
10.2.5.3 Jog HAL Pins . . . 632
10.2.5.4 Jog Velocities and Turtle-Jog HAL Pin . . . 632
10.2.5.5 Jog Increment HALPins . . . 632
10.2.5.6 Hardware-Entsperr-Pin . . . 633
10.2.5.7 Fehler/Warnung Pins . . . 633
10.2.5.8 Vom Benutzer erstellte HAL-Pins für Nachrichten . . . 633
10.2.5.9 Spindel-Feedback-Pins . . . 634
10.2.5.10 Pins to Indicate Program Progress Information . . . 634
10.2.5.11 Tool Related Pins . . . 635
10.2.6 Automatische Werkzeugmessung . . . 636
10.2.6.1 Provided Pins . . . 637
10.2.6.2 INI File Modifications . . . 638
10.2.6.3 Benötigte Dateien . . . 638
10.2.6.4 Benötigte HAL-Verbindungen . . . 639
10.2.7 Die Einstellungsseite . . . 639
10.2.7.1 Erscheinungsbild . . . 640
10.2.7.2 Hardware . . . 645
10.2.7.3 Erweiterte Einstellungen . . . 647
10.2.8 Icon Themen . . . 649
10.2.8.1 Benutzerdefiniertes Symboldesign (eigentlich engl. icon theme) . . . 650
10.2.8.2 Symbolische Icons . . . 650
10.2.9 Drehmaschinen-spezifischer Abschnitt . . . 651
10.2.10 Plasmaspezifischer Abschnitt . . . 654
10.2.11 Videos on YouTube . . . 654
10.2.11.1 Grundlegende Verwendung . . . 655
10.2.11.2 Simulierte Jog-Wheels . . . 655
10.2.11.3 Einstellungen Seite . . . 655
10.2.11.4 Simulierte Hardware-Taste . . . 655
10.2.11.5 Benutzer-Registerkarten . . . 655

10.2.11.6 Videos zur Werkzeugvermessung . . . 655
10.2.12 Known Problems . . . 655
10.2.12.1 Seltsame Zahlen im Infobereich . . . 655
10.2.12.2 Nicht endendes Makro . . . 656
10.3 The Touchy Graphical User Interface . . . 656
10.3.1 Panel-Konfiguration . . . 657
10.3.1.1 HAL-Verbindungen . . . 657
10.3.1.2 Empfohlen für jede Einrichtung . . . 658
10.3.2 Einrichtung . . . 658
10.3.2.1 Touchy aktivieren . . . 658
10.3.2.2 Einstellungen . . . 659
10.3.2.3 Makros . . . 659
10.4 Gscreen . . . 659
10.4.1 Einführung . . . 659
10.4.1.1 Glade-Datei . . . 664
10.4.1.2 PyGTK . . . 664
10.4.2 GladeVCP . . . 665
10.4.2.1 Übersicht . . . 665
10.4.2.2 Ein GladeVCP-Panel erstellen . . . 666
10.4.3 Erstellen eines einfachen benutzerdefinierten Bildschirms . . . 667
10.4.4 Beispiel für eine Handler-Datei . . . 669
10.4.4.1 Hinzufügen von Funktionen für Tastenkombinationen . . . 670
10.4.4.2 LinuxCNC-Status Status . . . 671
10.4.4.3 Jogging-Tasten . . . 671
10.4.5 Gscreen Start . . . 671
10.4.6 INI-Einstellungen . . . 673
10.4.7 Benutzerdialog-Meldungen . . . 673
10.4.7.1 Kopieren Sie die Datei ”Stock Handler/Glade” zur Bearbeitung . . . 674
10.5 QtDragon GUI . . . 675
10.5.1 Einführung . . . 675
10.5.1.1 QtDragon . . . 676
10.5.1.2 QtDragon_hd . . . 677
10.5.2 Getting Started - The INI File . . . 677
10.5.2.1 Anzeige (engl. display) . . . 677
10.5.2.2 Einstellungen . . . 678
10.5.2.3 Protokollierung (engl. logging) . . . 678
10.5.2.4 Override-Kontrollen . . . 678
10.5.2.5 Spindelsteuerungen . . . 678
10.5.2.6 Jogging-Inkremente . . . 679

10.5.2.7 Jog-Geschwindigkeit . . . 679
10.5.2.8 Dialogsystem für Benutzermeldungen . . . 679
10.5.2.9 Benutzerdefinierte VCP-Panels einbetten . . . 679
10.5.2.10 Subroutine Paths . . . 680
10.5.2.11 Vorschau Kontrolle . . . 681
10.5.2.12 Programmerweiterungen/Filter . . . 681
10.5.2.13 Sonden-/Touchplate-/Lasereinstellungen . . . 681
10.5.2.14 Abort detection . . . 682
10.5.2.15 Startup codes . . . 682
10.5.2.16 Makro-Buttons . . . 682
10.5.2.17 Post GUI HAL File . . . 683
10.5.2.18 Post GUI HAL Command . . . 683
10.5.2.19 Integrierte Beispielkonfigurationen . . . 683
10.5.3 Tastenbelegungen . . . 683
10.5.4 Buttons . . . 684
10.5.5 Virtuelle Tastatur . . . 684
10.5.6 HAL-Pins . . . 684
10.5.7 HAL-Dateien . . . 685
10.5.8 Manueller Werkzeugwechsel . . . 685
10.5.9 Spindel . . . 686
10.5.10 Automatisches Anheben der Z-Achse bei Pausieren der Spindel . . . 686
10.5.11 Z-Level-Kompensation . . . 686
10.5.11.1 Verwendung von G-code Ripper für die Z-Ebenen-Kompensation . . . 687
10.5.12 Sondieren . . . 689
10.5.12.1 Versa Probe . . . 690
10.5.12.2 Basic probe . . . 692
10.5.13 Touch-Platte . . . 695
10.5.14 Automatische Werkzeugmessung . . . 695
10.5.14.1 Werkstückhöhe Antasten . . . 698
10.5.14.2 Werkzeugmess-Pins . . . 700
10.5.14.3 Änderungen an der INI-Datei für Werkzeugmessungen . . . 700
10.5.14.4 Required HAL Connections . . . 702
10.5.15 Ausführen von gegebener Zeile . . . 702
10.5.16 Laser-Buttons . . . 702
10.5.17 Beschreibung der Registerkarten . . . 702
10.5.17.1 Hauptregisterkarte . . . 702
10.5.17.2 Registerkarte ”Datei” . . . 703
10.5.17.3 Registerkarte ”Offsets” . . . 703
10.5.17.4 Registerkarte ”Werkzeug” . . . 703

10.5.17.5 Registerkarte "Status" . . . 703
10.5.17.6 Registerkarte "Sonde . . . 703
10.5.17.7 Camview-Registerkarte . . . 703
10.5.17.8 G-Codes Registrierkarte . . . 704
10.5.17.9 Registerkarte "Einstellungen" . . . 704
10.5.17.10 Registerkarte "Einstellungen" . . . 705
10.5.17.11 Registerkarte "Dienstprogramme" . . . 705
10.5.17.12 User Tab . . . 705
10.5.18 Stile . . . 706
10.5.19 Anpassung . . . 706
10.5.19.1 Stylesheets . . . 706
10.5.19.2 Qt Designer und Python-Code . . . 707
10.6 NGCGUI . . . 709
10.6.1 Übersicht . . . 709
10.6.2 Beispiel-Konfigurationen . . . 710
10.6.3 Bibliothek (engl. library)-Verzeichnisse (engl. locations) . . . 712
10.6.4 Standalone-Nutzung . . . 713
10.6.4.1 Eigenständiges NGCGUI . . . 713
10.6.4.2 Eigenständiges (engl. standalone) PyNGCGUI . . . 713
10.6.5 NGCGUI einbetten . . . 714
10.6.5.1 NGCGUI in AXIS einbetten . . . 714
10.6.5.2 PyNGCGUI als GladeVCP-Registerkarte in ein GUI einbetten . . . 715
10.6.5.3 Zusätzliche INI-Datei-Elemente, die für NGCGUI oder PyNGCGUI erforderlich sind . . . 715
10.6.5.4 Truetype Tracer . . . 717
10.6.5.5 INI File Path Specifications . . . 717
10.6.5.6 Zusammenfassung der Details der INI-Datei für die Verwendung von NGCGUI . . . 719
10.6.6 Dateianforderungen für NGCGUI-Kompatibilität . . . 720
10.6.6.1 Anforderungen an eine G-code-Unterroutine (.ngc) in einer Datei . . . 720
10.6.6.2 Gcode-Meta-Compiler-Dateianforderungen (.gcmc) . . . 722
10.6.7 DB25 Beispiel . . . 724
10.6.8 Erstellen eines Unterprogramms . . . 726
10.7 TkLinuxCNC GUI . . . 727
10.7.1 Einführung . . . 727
10.7.2 Erste Schritte . . . 727
10.7.2.1 Eine typische Sitzung mit TkLinuxCNC . . . 727
10.7.3 Elemente des TkLinuxCNC-Fensters . . . 728
10.7.3.1 Die wichtigsten Buttons . . . 728

10.7.3.2 Statusleiste der Offset-Anzeige . . . 729
10.7.3.3 Koordinatenanzeigebereich . . . 729
10.7.3.4 TkLinuxCNC Interpreter / Automatic Program Control . . . 729
10.7.3.5 Manuelle Steuerung . . . 730
10.7.3.6 Code-Eingabe . . . 731
10.7.3.7 Jog-Geschwindigkeit . . . 731
10.7.3.8 Vorschub Neufestsetzung (engl. override) . . . 731
10.7.3.9 Spindeldrehzahl Override . . . 731
10.7.4 Tastatursteuerung . . . 731
10.8 QtPlasmaC . . . 732
10.8.1 Preamble . . . 732
10.8.2 License . . . 732
10.8.3 Einführung . . . 732
10.8.4 LinuxCNC installieren . . . 735
10.8.4.1 Wenn der Benutzer kein Linux installiert hat . . . 736
10.8.4.2 Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) . . . 736
10.8.4.3 Package Installation (Buildbot) If The User Has Linux on Debian 11 (Bullseye) or Debian 10 (Buster) . . . 736
10.8.4.4 Run In Place Installation If The User Has Linux Installed . . . 736
10.8.5 Erstellen einer QtPlasmaC Konfiguration . . . 736
10.8.5.1 Modi . . . 736
10.8.5.2 Verfügbare I/Os . . . 737
10.8.5.3 Recommended Settings: . . . 738
10.8.5.4 Configuring . . . 739
10.8.5.5 Qt-Abhängigkeitsfehler . . . 744
10.8.5.6 Erstmalige Einrichtung . . . 744
10.8.6 Migrating to QtPlasmac From PlasmaC (AXIS or GMOCCAPY) . . . 747
10.8.6.1 Quick Method . . . 747
10.8.6.2 Neue Basis-Konfigurationsmethode . . . 749
10.8.7 Other QtPlasmaC Setup Considerations . . . 750
10.8.7.1 Lowpass Filter . . . 750
10.8.7.2 Contact Bounce . . . 751
10.8.7.3 Contact Load . . . 751
10.8.7.4 Desktop-Starthilfe . . . 752
10.8.7.5 QtPlasmaC Dateien . . . 753
10.8.7.6 INI File . . . 754
10.8.8 QtPlasmaC GUI Überblick . . . 755
10.8.8.1 Beenden von QtPlasmaC . . . 755

10.8.8.2 HAUPT (engl. main)-Registerkarte (engl. tab) . . . 756
10.8.8.3 Preview Views . . . 762
10.8.8.4 CONVERSATIONAL Tab . . . 762
10.8.8.5 PARAMETERS Tab . . . 763
10.8.8.6 SETTINGS Tab . . . 768
10.8.8.7 STATISTICS Tab . . . 772
10.8.9 Using QtPlasmaC . . . 772
10.8.9.1 Einheitensysteme . . . 773
10.8.9.2 Präambel und Postambel Codes . . . 773
10.8.9.3 Obligatorische Codes . . . 773
10.8.9.4 Koordinaten . . . 774
10.8.9.5 Cut Feed Rate . . . 774
10.8.9.6 Material-Datei . . . 774
10.8.9.7 Manuelles Materialhandling . . . 776
10.8.9.8 Automatisches Materialhandling . . . 776
10.8.9.9 Material hinzufügen Via Magic Kommentare in G-Code . . . 777
10.8.9.10 Material Konverter . . . 778
10.8.9.11 LASER . . . 781
10.8.9.12 CAMERA . . . 783
10.8.9.13 Pfadtoleranz . . . 785
10.8.9.14 Angehaltene Bewegung . . . 785
10.8.9.15 Pause am Ende von Cut . . . 785
10.8.9.16 Mehrere Werkzeuge . . . 785
10.8.9.17 Geschwindigkeitsreduzierung . . . 786
10.8.9.18 THC (Brennerhöhensteuerung, engl. torch height controller) . . . 787
10.8.9.19 Fräserkompensation . . . 788
10.8.9.20 Initial Height Sense (IHS) Skip . . . 789
10.8.9.21 Sondieren . . . 789
10.8.9.22 Offset Probing . . . 790
10.8.9.23 Cut Types . . . 790
10.8.9.24 Hole Cutting - Intro . . . 791
10.8.9.25 Löcher schneiden . . . 791
10.8.9.26 Hole Cutting - Automatic . . . 793
10.8.9.27 Single Cut . . . 794
10.8.9.28 Thick Materials . . . 796
10.8.9.29 Mesh Mode (Expanded Metal Cutting) . . . 796
10.8.9.30 Ignore Arc OK . . . 796
10.8.9.31 Cut Recovery . . . 797
10.8.9.32 Run From Line . . . 798

10.8.9.3 Scribe . . . 800
10.8.9.4 Spotting . . . 801
10.8.9.5 Benutzerdefinierte Layouts für virtuelle Tastaturen . . . 802
10.8.9.6 Tastatürkürzel . . . 803
10.8.9.7 MDI . . . 805
10.8.10 Conversational Shape Library . . . 806
10.8.10.1 Konversationseinstellungen . . . 808
10.8.10.2 Konversationslinien und Bögen . . . 809
10.8.10.3 Conversational Single Shape . . . 810
10.8.10.4 Conversational Group Of Shapes . . . 811
10.8.10.5 Conversational Block . . . 811
10.8.10.6 Conversational Saving A Job . . . 813
10.8.11 Fehlermeldungen . . . 813
10.8.11.1 Fehlerprotokollierung . . . 813
10.8.11.2 Error Message Display . . . 813
10.8.11.3 Critical Errors . . . 813
10.8.11.4 Warning Messages . . . 815
10.8.12 Updating QtPlasmaC . . . 816
10.8.12.1 Standard Update . . . 816
10.8.12.2 Continuous Update . . . 816
10.8.13 Modify An Existing QtPlasmaC Configuration . . . 816
10.8.14 Customizing QtPlasmaC GUI . . . 816
10.8.14.1 Add A Custom Style . . . 817
10.8.14.2 Create A New Style . . . 817
10.8.14.3 Rückkehr zum Standardstil . . . 818
10.8.14.4 Custom Python Code . . . 818
10.8.14.5 Custom G-code Filter . . . 819
10.8.15 QtPlasmaC Fortgeschrittene Themen . . . 820
10.8.15.1 Benutzerdefinierte Buttons . . . 820
10.8.15.2 Peripheral Offsets (Laser, Camera, Scribe, Offset Probe) . . . 826
10.8.15.3 Keep Z Motion . . . 828
10.8.15.4 Externe HAL-Pins . . . 828
10.8.15.5 Programm-Buttons ausblenden . . . 830
10.8.15.6 Tuning-Modus 0 Arc OK . . . 830
10.8.15.7 Lost Arc Delay . . . 831
10.8.15.8 Zero Window . . . 831
10.8.15.9 Tuning Void Sensing . . . 831
10.8.15.10 Max Offset . . . 832
10.8.15.11 Enable Tabs During Automated Motion . . . 832

10.8.15.02 Override Jog Inhibit Via Z+ Jog . . . 832
10.8.15.03 PlasmaC State Outputs . . . 832
10.8.15.04 PlasmaC Debug Print . . . 833
10.8.15.05 Hypertherm PowerMax Communications . . . 833
10.8.16 Internationalisierung . . . 834
10.8.17 Anhang . . . 836
10.8.17.01 Beispielkonfigurationen . . . 836
10.8.17.02 NGC Beispiele . . . 836
10.8.17.03 QtPlasmaC-spezifische G-Codes . . . 836
10.8.17.04 QtPlasmaC G-code Examples . . . 838
10.8.17.05 Mesa THCAD . . . 840
10.8.17.06 RS485-Verbindungen . . . 841
10.8.17.07 Lichtbogen OK mit einem Reed-Relais . . . 844
10.8.17.08 Schematische Darstellung der Kontaktbelastung . . . 846
10.8.18 Bekannte Probleme . . . 846
10.8.18.1 Tastatur-Jogging . . . 846
10.8.19 Unterstützung . . . 847
10.9 MDRO GUI . . . 847
10.9.1 Einführung . . . 847
10.9.2 Erste Schritte . . . 848
10.9.2.1 INI-Datei Optionen . . . 848
10.9.2.2 Kommandozeilen-Optionen . . . 849
10.9.2.3 Pins . . . 849
10.9.3 MDRO Fenster . . . 849
10.9.4 Index-Operationen . . . 850
10.9.5 Simulation . . . 850

**11 G-Code Programmierung** **851**
11.1 Koordinatensysteme . . . 851
11.1.1 Einführung . . . 851
11.1.2 Maschinenkoordinatensystem . . . 851
11.1.2.1 Maschinenkoordinaten bewegen sich: G53 . . . 851
11.1.3 Koordinatensysteme . . . 852
11.1.3.1 Standard-Koordinatensystem . . . 854
11.1.3.2 Koordinatensystem-Offsets einstellen . . . 854
11.1.4 Lokale und globale Offsets . . . 854
11.1.4.1 Der Befehl G52 . . . 854
11.1.5 G92-Achsen-Offsets . . . 855
11.1.5.1 Die G92-Befehle . . . 855

11.1.5.2G92 Werte festlegen . . . 856
11.1.5.3G92 Persistenz-Vorsichtsmaßnahmen . . . 857
11.1.5.4G92 und G52 Wechselwirkungen - Hinweise zur Vorsicht . . . 858
11.1.6Beispielprogramme mit Offsets/Kompensationen . . . 858
11.1.6.1Beispielprogramm mit Werkstückkoordinaten-Versätzen . . . 858
11.1.6.2Beispielprogramm mit G52-Offsets . . . 859
11.2Tool Compensation . . . 859
11.2.1Touch-Off . . . 859
11.2.1.1Verwendung von G10 L1/L10/L11 . . . 860
11.2.2Werkzeugtabelle . . . 860
11.2.2.1Werkzeugtabellen-Format . . . 860
11.2.2.2Tool IO . . . 862
11.2.2.3Tool Changers . . . 864
11.2.3Tool Length Compensation . . . 864
11.2.4Cutter Radius Compensation . . . 865
11.2.4.1Übersicht . . . 866
11.2.4.2Beispiele . . . 868
11.3GUI zur Werkzeug-Bearbeitung . . . 869
11.3.1Übersicht . . . 869
11.3.2Spaltensortierung . . . 870
11.3.3Spaltenauswahl . . . 871
11.3.4Eigenständige Verwendung . . . 871
11.4G-Code Übersicht . . . 872
11.4.1Übersicht . . . 872
11.4.2Format einer Zeile . . . 873
11.4.2.1/: Block Delete . . . 873
11.4.2.2Line Number . . . 873
11.4.2.3Word . . . 873
11.4.2.4End of Line Marker . . . 874
11.4.2.5Numbers . . . 874
11.4.3Parameter . . . 875
11.4.3.1Numbered Parameters . . . 876
11.4.3.2Subroutine Parameters . . . 878
11.4.3.3Named Parameters . . . 878
11.4.3.4Predefined Named Parameters . . . 879
11.4.3.5System Parameters . . . 881
11.4.4HAL pins and INI values . . . 882
11.4.5Expressions . . . 883
11.4.6Binary Operators . . . 883

11.4.7 Gleichheit und Gleitkommawerte . . . 884
11.4.8 Funktionen . . . 884
11.4.9 Wiederholte Elemente . . . 885
11.4.10 Artikelreihenfolge . . . 885
11.4.11 Befehle und Maschinenmodi . . . 885
11.4.12 Polar Coordinates . . . 886
11.4.13 Modal Groups . . . 887
11.4.14 Kommentare . . . 889
11.4.15 Nachrichten . . . 889
11.4.16 Probe Logging . . . 890
11.4.17 Protokollierung (engl. logging) . . . 890
11.4.18 Debug Messages . . . 890
11.4.19 Print Messages . . . 890
11.4.20 Comment Parameters . . . 890
11.4.21 File Requirements . . . 891
11.4.22 File Size . . . 891
11.4.23 G-code Order of Execution . . . 891
11.4.24 G-code Best Practices . . . 892
11.4.25 Lineare und rotierende Achsen . . . 893
11.4.26 Häufige Fehlermeldungen . . . 893
11.5 G-Codes . . . 894
11.5.1 Konventionen . . . 894
11.5.2 G-Code Quick Reference Table . . . 894
11.5.3 G0 Rapid Move . . . 896
11.5.3.1 Eilgangs-Geschwindigkeitsrate . . . 896
11.5.4 G1 Linear Move . . . 896
11.5.5 G2, G3 Arc Move . . . 897
11.5.5.1 Bögen durch ihr Zentrum beschrieben . . . 898
11.5.5.2 Beispiele für Center-Formate . . . 900
11.5.5.3 Bögen im Radiusformat . . . 901
11.5.6 G4 Dwell . . . 902
11.5.7 G5 Cubic Spline . . . 902
11.5.8 G5.1 Quadratic Spline . . . 903
11.5.9 G5.2 G5.3 NURBS Block . . . 904
11.5.10 G7 Lathe Diameter Mode . . . 905
11.5.11 G8 Lathe Radius Mode . . . 905
11.5.12 G10 L0 Reload Tool Table Data . . . 906
11.5.13 G10 L1 Set Tool Table . . . 906
11.5.14 G10 L2 Set Coordinate System . . . 907

11.5.15 G10 L10 Set Tool Table . . . 908
11.5.16 G10 L11 Set Tool Table . . . 909
11.5.17 G10 L20 Set Coordinate System . . . 909
11.5.18 G17 - G19.1 Plane Select . . . 910
11.5.19 G20, G21 Units . . . 910
11.5.20 G28, G28.1 Go/Set Predefined Position . . . 910
11.5.21 G30, G30.1 Go/Set Predefined Position . . . 911
11.5.22 G33 Spindle Synchronized Motion . . . 911
11.5.23 G33.1 Rigid Tapping . . . 912
11.5.24 G38.n Straight Probe . . . 914
11.5.25 G40 Compensation Off . . . 915
11.5.26 G41, G42 Cutter Compensation . . . 916
11.5.27 G41.1, G42.1 Dynamic Cutter Compensation . . . 917
11.5.28 G43 Tool Length Offset . . . 917
11.5.29 G43.1 Dynamic Tool Length Offset . . . 918
11.5.30 G43.2 Apply additional Tool Length Offset . . . 918
11.5.31 G49 Cancel Tool Length Compensation . . . 919
11.5.32 G52 Local Coordinate System Offset . . . 919
11.5.33 G53 Move in Machine Coordinates . . . 919
11.5.34 G54-G59.3 Select Coordinate System . . . 920
11.5.35 G61 Exact Path Mode . . . 921
11.5.36 G61.1 Exact Stop Mode . . . 921
11.5.37 G64 Path Blending . . . 921
11.5.38 G70 Lathe finishing cycle . . . 921
11.5.39 G71 G72 Lathe roughing cycles . . . 922
11.5.40 G73 Drilling Cycle with Chip Breaking . . . 923
11.5.41 G74 Left-hand Tapping Cycle with Dwell . . . 924
11.5.42 G76 Threading Cycle . . . 925
11.5.43 G80-G89 Canned Cycles . . . 928
11.5.43.1 Geläufige Begriffe . . . 928
11.5.43.2 Anhaftende Begriffe . . . 928
11.5.43.3 Zyklus wiederholen . . . 929
11.5.43.4 Rückzugsmodus . . . 929
11.5.43.5 Canned Cycle Errors . . . 929
11.5.43.6 Vorläufige und zwischenzeitliche Bewegung . . . 929
11.5.43.7 Warum ein Canned Cycle (Zyklus aus der Konserve)? . . . 930
11.5.44 G80 Cancel Canned Cycle . . . 932
11.5.45 G81 Drilling Cycle . . . 933
11.5.46 G82 Drilling Cycle, Dwell . . . 938

11.5.47 G83 Peck Drilling Cycle . . . 938
11.5.48 G84 Right-hand Tapping Cycle, Dwell . . . 939
11.5.49 G85 Boring Cycle, Feed Out . . . 939
11.5.50 G86 Boring Cycle, Spindle Stop, Rapid Move Out . . . 940
11.5.51 G87 Back Boring Cycle . . . 940
11.5.52 G88 Boring Cycle, Spindle Stop, Manual Out . . . 940
11.5.53 G89 Boring Cycle, Dwell, Feed Out . . . 940
11.5.54 G90, G91 Distance Mode . . . 941
11.5.55 G90.1, G91.1 Arc Distance Mode . . . 941
11.5.56 G92 Coordinate System Offset . . . 941
11.5.57 G92.1, G92.2 Reset G92 Offsets . . . 942
11.5.58 G92.3 Restore G92 Offsets . . . 942
11.5.59 G93, G94, G95 Feed Rate Mode . . . 942
11.5.60 G96, G97 Spindle Control Mode . . . 943
11.5.61 G98, G99 Canned Cycle Return Level . . . 944
11.6 M-Codes . . . 944
11.6.1 M-Code Kurzübersichtstabelle . . . 944
11.6.2 M0, M1 Program Pause . . . 945
11.6.3 M2, M30 Program End . . . 945
11.6.4 M60 Pallet Change Pause . . . 946
11.6.5 M3, M4, M5 Spindle Control . . . 946
11.6.6 M6 Tool Change . . . 946
11.6.6.1 Manueller Werkzeugwechsel . . . 946
11.6.6.2 Werkzeugwechsler . . . 947
11.6.7 M7, M8, M9 Coolant Control . . . 947
11.6.8 M19 Orient Spindle . . . 948
11.6.9 M48, M49 Speed and Feed Override Control . . . 948
11.6.10 M50 Feed Override Control . . . 949
11.6.11 M51 Spindle Speed Override Control . . . 949
11.6.12 M52 Adaptive Feed Control . . . 949
11.6.13 M53 Feed Stop Control . . . 949
11.6.14 M61 Set Current Tool . . . 949
11.6.15 M62 - M65 Digital Output Control . . . 950
11.6.16 M66 Wait on Input . . . 950
11.6.17 M67 Analog Output, Synchronized . . . 951
11.6.18 M68 Analog Output, Immediate . . . 952
11.6.19 M70 Save Modal State . . . 952
11.6.20 M71 Invalidate Stored Modal State . . . 953
11.6.21 M72 Restore Modal State . . . 953

11.6.22 M73 Save and Autorestore Modal State . . . 954
11.6.23 M98 und M99 . . . 955
11.6.23.1 Selektive Wiederherstellung des modalen Zustands . . . 955
11.6.24 M100-M199 User Defined Commands . . . 955
11.7 O Codes . . . 957
11.7.1 Verwendung von O-Codes . . . 957
11.7.2 Nummerierung . . . 957
11.7.3 Kommentare . . . 957
11.7.4 Subroutines . . . 958
11.7.4.1 Fanuc-Style Numbered Programs . . . 959
11.7.5 Looping . . . 960
11.7.6 Conditional . . . 961
11.7.7 Repeat . . . 962
11.7.8 Indirection . . . 962
11.7.9 Calling Files . . . 962
11.7.10 Subroutine return values . . . 963
11.7.11 Errors . . . 963
11.8 Other Codes . . . 964
11.8.1 F: Set Feed Rate . . . 964
11.8.2 S: Set Spindle Speed . . . 964
11.8.3 T: Select Tool . . . 964
11.9 G-Code Beispiele . . . 965
11.9.1 Beispiele für eine Fräsmaschine . . . 965
11.9.1.1 Fräsen von Spiralbohrungen . . . 965
11.9.1.2 Schlitzen (engl. slotting) . . . 965
11.9.1.3 Rastersonde (engl. grid probe) . . . 966
11.9.1.4 Intelligente Sonde (engl. smart probe) . . . 967
11.9.1.5 Werkzeuglängen-Messtaster . . . 968
11.9.1.6 Lochsonde (engl. hole probe) . . . 968
11.9.1.7 Fräserkompensation . . . 968
11.9.2 Beispiele für Drehmaschinen . . . 969
11.9.2.1 Gewinde-Drehen (engl. threading) . . . 969
11.10 Vom Bild zu G-Code . . . 969
11.10.1 Was ist eine Tiefenkarte (engl. depth map)? . . . 969
11.10.2 Integration von Image-to-Gcode in die AXIS-Benutzeroberfläche . . . 970
11.10.3 Verwendung von `image-to-gcode` . . . 970
11.10.4 Optionen . . . 970
11.10.4.1 Einheiten . . . 970
11.10.4.2 Bild invertieren (engl. invert image) . . . 970

11.10.4.3 Bild normalisieren (engl. normalize image) . . . 970
11.10.4.4 Erweitern des Bildrandes (engl. expand image border) . . . 970
11.10.4.5 Toleranz (Einheiten) . . . 971
11.10.4.6 Pixelgröße (Einheiten) . . . 971
11.10.4.7 Tauchvorschubgeschwindigkeit (Einheiten pro Minute) . . . 971
11.10.4.8 Vorschubgeschwindigkeit (Einheiten pro Minute) . . . 971
11.10.4.9 Spindeldrehzahl (RPM) . . . 971
11.10.4.10 Abtastmuster (engl. scan pattern) . . . 971
11.10.4.11 Einrichtung . . . 971
11.10.4.12 Tiefe (engl. depth) (Einheiten) . . . 972
11.10.4.13 Schrittweite (engl. step over) (Pixel) . . . 972
11.10.4.14 Werkzeug-Durchmesser . . . 972
11.10.4.15 Sicherheitshöhe . . . 972
11.10.4.16 Werkzeug-Typ . . . 972
11.10.4.17 Face bounding . . . 972
11.10.4.18 Kontaktwinkel . . . 972
11.10.4.19 Schruppversatz und Schrupptiefe pro Durchgang . . . 973
11.11 RS274/NGC Unterschiede . . . 973
11.11.1 Änderungen gegenüber RS274/NGC . . . 973
11.11.2 Ergänzungen zu RS274/NGC . . . 974

**12 Virtuelle Schalttafeln** **976**
12.1 PyVCP . . . 976
12.1.1 Einführung . . . 976
12.1.2 Panel Konstruktion . . . 977
12.1.3 Sicherheit . . . 978
12.1.4 ACHSE . . . 978
12.1.4.1 Beispiel-Panel . . . 978
12.1.5 Eigenständig (engl. stand alone) . . . 980
12.1.6 Widgets . . . 981
12.1.6.1 Syntax . . . 981
12.1.6.2 Allgemeine Anmerkungen . . . 981
12.1.6.3 Label . . . 982
12.1.6.4 Multi_Label . . . 983
12.1.6.5 LEDs . . . 983
12.1.6.6 Buttons . . . 985
12.1.6.7 Nummernanzeigen . . . 987
12.1.6.8 Numerische Eingaben . . . 990
12.1.6.9 Bilder . . . 993

12.1.6.10 Containers (engl. für Behälter) . . . 995
12.2 PyVCP-Beispiele . . . 1000
12.2.1 ACHSE . . . 1000
12.2.2 Schwebende (engl. floating) Panels . . . 1001
12.2.3 Beispiel für Jog-Buttons . . . 1001
12.2.3.1 Erstellen der Widgets . . . 1002
12.2.3.2 Verbindungen herstellen . . . 1004
12.2.4 Port-Tester . . . 1005
12.2.5 GS2-Drehzahlmesser . . . 1008
12.2.5.1 Das Panel . . . 1008
12.2.5.2 Die Verbindungen . . . 1010
12.2.6 Referenzfahrt im Eilgang Button . . . 1010
12.3 GladeVCP: Glade Virtual Control Panel . . . 1012
12.3.1 Was ist GladeVCP? . . . 1012
12.3.1.1 PyVCP im Vergleich zu GladeVCP auf einen Blick . . . 1012
12.3.2 Ein kurzer Rundgang mit dem Beispielpanel . . . 1013
12.3.2.1 Erkunden des Beispielpanels . . . 1016
12.3.2.2 Erkunden der Beschreibung der Benutzeroberfläche . . . 1017
12.3.2.3 Erkunden der Python Callback Funktionen . . . 1017
12.3.3 Erstellen und Integrieren einer Glade-Benutzeroberfläche . . . 1017
12.3.3.1 Voraussetzung ist: Glade-Installation . . . 1017
12.3.3.2 Glade ausführen, um eine neue Benutzeroberfläche zu erstellen . . . 1017
12.3.3.3 Testen eines Panels . . . 1018
12.3.3.4 Vorbereiten der HAL-Befehlsdatei . . . 1019
12.3.3.5 Einbindung in AXIS, wie PyVCP . . . 1019
12.3.3.6 Einbetten als Registerkarte (engl. tab) . . . 1020
12.3.3.7 Integration in Touchy . . . 1021
12.3.4 GladeVCP-Befehlszeilenoptionen . . . 1021
12.3.5 Den GladeVCP-Startvorgang verstehen . . . 1022
12.3.6 HAL Widget-Referenz . . . 1023
12.3.6.1 Benennung von Widgets und HAL-Pins . . . 1023
12.3.6.2 Python-Attribute und Methoden von HAL Widgets . . . 1024
12.3.6.3 Einstellung von Pin- und Widget-Werten . . . 1024
12.3.6.4 Das hal-pin-changed-Signal . . . 1025
12.3.6.5 Buttons . . . 1025
12.3.6.6 Skalen (engl. scales) . . . 1026
12.3.6.7 SpinButton . . . 1027
12.3.6.8 Hal_Dial . . . 1027
12.3.6.9 Jog-Handrad (engl. jog wheel) . . . 1029

12.3.6.10 Geschwindigkeitsregelung . . . 1031
12.3.6.11 Label . . . 1033
12.3.6.12 Containers (engl. für Behälter) . . . 1034
12.3.6.13 LED . . . 1034
12.3.6.14 ProgressBar (engl. für Fortschrittsbalken) . . . 1035
12.3.6.15 ComboBox . . . 1036
12.3.6.16 Bars (engl. für Balken) . . . 1037
12.3.6.17 Meter . . . 1038
12.3.6.18 HAL_Graph . . . 1039
12.3.6.19 Gremlin tool path preview for NGC files . . . 1040
12.3.6.20 HAL_Offset . . . 1042
12.3.6.21 DRO-Widget . . . 1043
12.3.6.22 Combi_DRO Widget . . . 1045
12.3.6.23 IconView (Dateiauswahl) . . . 1048
12.3.6.24 Rechner-Widget . . . 1051
12.3.6.25 Werkzeugeditor-Widget (engl. tooleditor widget) . . . 1052
12.3.6.26 Offset-Seite . . . 1054
12.3.6.27 HAL_sourceview-Widget . . . 1056
12.3.6.28 MDI-Geschichte . . . 1057
12.3.6.29 Animierte Funktionsdiagramme: HAL-Widgets in einer Bitmap . . . 1058
12.3.7 Referenz zu Aktions-Widgets . . . 1058
12.3.7.1 VCP Action-Widgets . . . 1059
12.3.7.2 VCP Action Python . . . 1059
12.3.7.3 VCP ToggleAction-Widgets . . . 1060
12.3.7.4 Die Action_MDI Toggle und Action_MDI Widgets . . . 1061
12.3.7.5 Ein einfaches Beispiel: Ausführen eines MDI-Befehls bei Button-Druck . . . 1061
12.3.7.6 Parameterübergabe mit Action_MDI- und ToggleAction_MDI-Widgets . . . 1061
12.3.7.7 Ein fortgeschrittenes Beispiel: Übergabe von Parametern an eine O-Wort-Unterroutine . . . 1062
12.3.7.8 Vorbereitung einer MDI-Aktion und anschließendes Aufräumen . . . 1062
12.3.7.9 Verwendung des LinuxCNC Stat-Objekts zum Umgang mit Statusänderungen . . . 1063
12.3.8 GladeVCP-Programmierung . . . 1064
12.3.8.1 Benutzerdefinierte Aktionen . . . 1064
12.3.8.2 Core-Bibliothek . . . 1064
12.3.8.3 Ein Beispiel: Hinzufügen benutzerdefinierter Callback-Funktionen in Python 1065
12.3.8.4 HAL-Wertänderungs-Ereignisse . . . 1065
12.3.8.5 Programmiermodell . . . 1066
12.3.8.6 Initialisierungssequenz . . . 1067

12.3.8.7 Mehrere Callbacks mit demselben Namen . . . 1068
12.3.8.8 Die GladeVCP `-U <useropts>` Flag . . . 1068
12.3.8.9 Persistente Variablen in GladeVCP . . . 1068
12.3.8.10 Verwendung persistenter Variablen . . . 1069
12.3.8.11 Speichern des Status beim Herunterfahren von GladeVCP . . . 1070
12.3.8.12 Status speichern, wenn Strg-C gedrückt wird . . . 1071
12.3.8.13 Manuelle Bearbeitung von INI-Dateien (.ini) . . . 1071
12.3.8.14 Hinzufügen von HAL-Pins . . . 1071
12.3.8.15 Hinzufügen von Timern . . . 1071
12.3.8.16 HAL-Widget-Eigenschaften programmatisch einstellen . . . 1072
12.3.8.17 Value-changed callback with hal_glib . . . 1072
12.3.8.18 Beispiele und die Entwicklung Ihrer eigenen GladeVCP-Anwendung . . . 1073
12.3.9 FAQ . . . 1073
12.3.10 Fehlersuche . . . 1074
12.3.11 Implementierungshinweis: Schlüsselbehandlung in AXIS . . . 1074
12.3.12 Hinzufügen von benutzerdefinierten Widgets . . . 1074
12.3.13 GladeVCP-Hilfsanwendungen . . . 1075
12.4 GladeVCP Library modules . . . 1075
12.4.1 Info . . . 1075
12.4.2 Action . . . 1077
12.5 QtVCP . . . 1079
12.5.1 Schaukasten . . . 1079
12.5.2 Übersicht . . . 1086
12.5.2.1 QtVCP Widgets . . . 1087
12.5.2.2 INI-Einstellungen . . . 1087
12.5.2.3 Qt Designer UI Datei . . . 1088
12.5.2.4 Handler-Dateien . . . 1089
12.5.2.5 Bibliotheken Module . . . 1089
12.5.2.6 Themen . . . 1089
12.5.2.7 Lokale Dateien . . . 1089
12.5.2.8 Veränderung mitglieferter Bildschirmmasken . . . 1090
12.5.3 VCP-Paneele . . . 1093
12.5.3.1 Builtin Panels . . . 1093
12.5.3.2 Benutzerdefinierte Bedienfelder . . . 1097
12.5.4 Build A Simple Clean-sheet Custom Screen . . . 1099
12.5.4.1 Übersicht . . . 1099
12.5.4.2 Holen Sie sich Qt Designer, um LinuxCNC-Widgets einzubinden . . . 1099
12.5.4.3 Build The Screen `.ui` File . . . 1100
12.5.4.4 Handler-Datei . . . 1104

12.5.4.5INI Configuration . . . 1104
12.5.5Handler File In Detail . . . 1104
12.5.5.1Übersicht . . . 1105
12.5.5.2IMPORT Section . . . 1108
12.5.5.3Abschnitt INSTANTIATE BIBRARIES . . . 1108
12.5.5.4HANDLER CLASS Section . . . 1108
12.5.5.5INITIALIZE Section . . . 1108
12.5.5.6Abschnitt zu BESONDEREN FUNKTIONEN . . . 1109
12.5.5.7STATUS CALLBACKS Abschnitt . . . 1110
12.5.5.8CALLBACKS FROM FORM Abschnitt . . . 1110
12.5.5.9GENERAL FUNCTIONS Section . . . 1110
12.5.5.10KEY BINDING Section . . . 1110
12.5.5.11CLOSING EVENT Section . . . 1110
12.5.6Connecting Widgets to Python Code . . . 1110
12.5.6.1Übersicht . . . 1111
12.5.6.2Using Qt Designer to add Slots . . . 1111
12.5.6.3Python Handler Changes . . . 1112
12.5.7Mehr zum Thema . . . 1113
12.6QtVCP Virtuelle Kontrollpanels . . . 1113
12.6.1Builtin Virtual Control Panels . . . 1113
12.6.1.1copy . . . 1113
12.6.1.2spindle_belts . . . 1114
12.6.1.3test_dial . . . 1116
12.6.1.4test_button . . . 1117
12.6.1.5test_led . . . 1117
12.6.1.6test_panel . . . 1118
12.6.1.7cam_align . . . 1119
12.6.1.8sim_panel . . . 1121
12.6.1.9tool_dialog . . . 1122
12.6.2vismach 3D Simulation Panels . . . 1123
12.6.2.1QtVCP vismach_mill_xyz . . . 1123
12.6.2.2QtVCP vismach_router_atc . . . 1124
12.6.2.3QtVCP vismach_scara . . . 1125
12.6.2.4QtVCP vismach_millturn . . . 1126
12.6.2.5QtVCP vismach_mill_5axis_gantry . . . 1127
12.6.2.6QtVCP vismach_fanuc_200f . . . 1128
12.6.3Custom Virtual Control Panels . . . 1129
12.6.4Embedding QtVCP Virtual Control Panels into QtVCP Screens . . . 1130
12.6.4.1Embedding Commands . . . 1130

12.6.4.2 Location of builtin Panels . . . 1130
12.6.4.3 Location of Custom Panels . . . 1131
12.6.4.4 Handler Programming Tips . . . 1131
12.6.4.5 Designer Widget Tips . . . 1131
12.6.4.6 Handler Patching - Subclassing Builtin Panels . . . 1131
12.7 QtVCP Widgets . . . 1132
12.7.1 Nur HAL-Widgets . . . 1133
12.7.1.1 XEmbed - Widget zum Einbetten von Programmen . . . 1133
12.7.1.2 Slider - HAL-Pin-Wert-Anpassungs-Widget . . . 1133
12.7.1.3 LED - Indicator Widget . . . 1133
12.7.1.4 CheckBox Widget . . . 1134
12.7.1.5 RadioButton Widget . . . 1134
12.7.1.6 Gauge - Rundes Messuhr-Widget . . . 1135
12.7.1.7 *HalBar* - HAL Bar Level Indicator . . . 1136
12.7.1.8 HALPad - HAL Buttons Joypad . . . 1138
12.7.1.9 PushButton - HAL Pin Toggle Widget . . . 1139
12.7.1.10 FocusOverlay - Focus Overlay Widget . . . 1140
12.7.1.11 GridLayout - Grid Layout Widget . . . 1140
12.7.1.12 hal_label - HAL Label Widget . . . 1141
12.7.1.13 LCDNumber - Widget zum Auslesen der LCD-Stilnummer . . . 1142
12.7.1.14 DoubleScale - Spin Button Entry Widget . . . 1142
12.7.1.15 GeneralHALInput - General Signals/Slots Input Connection Widget . . . 1142
12.7.1.16 GeneralHALOutput - General Signals/Slots Output Connection Widget . . . 1142
12.7.1.17 WidgetSwitcher - Multi-widget Layout View Switcher Widget . . . 1143
12.7.2 Widgets für Maschinensteuerungen . . . 1143
12.7.2.1 ActionButton - Aktionssteuerungs-Widget der Maschinensteuerung . . . 1143
12.7.2.2 ActionToolButton - Optional Actions Menu Button Widget . . . 1146
12.7.2.3 RoundButton - Round Shapped ActionButton Widget . . . 1146
12.7.2.4 AxisToolButton - Select and Set Axis Widget . . . 1147
12.7.2.5 CamView - Workpiece Alignment and Origin Setting Widget . . . 1147
12.7.2.6 DROLabel - Axis Position Display Widget . . . 1147
12.7.2.7 GcodeDisplay - G-code Text Display Widget . . . 1148
12.7.2.8 GcodeEditor - G-code Program Editor Widget . . . 1150
12.7.2.9 GCodeGraphics - G-code Graphic Backplot Widget . . . 1150
12.7.2.10 *StateLabel* - Controller-Modi Statusbeschriftungsanzeige-Widget . . . 1154
12.7.2.11 StatusLabel - Anzeige-Widget für Controller-Variablen-Zustandsbeschriftung 1155
12.7.2.12 StatusImageSwitcher - Controller Status Image Switcher . . . 1157
12.7.2.13 StatusStacked - Mode Status Display Switching Widget . . . 1159
12.7.2.14 JogIncrements - Jog Increments Value Selection Widget . . . 1159

12.7.2.15 ScreenOption - General Options Setting widget . . . 1159
12.7.2.16 StatusSlider - Controller-Einstellungs-Schieberegler-Widget . . . 1164
12.7.2.17 StateLED - Controller-Status-LED-Widget . . . 1165
12.7.2.18 StatusAdjustmentBar - Widget zum Einstellen von Controller-Werten . . . 1166
12.7.2.19 SystemToolButton - Widget zur Auswahl des Benutzersystems . . . 1167
12.7.2.20 MacroTab - Spezielles Makro-Widget . . . 1167
12.7.2.21 MDILine - MDI-Befehlszeileneingabe-Widget . . . 1169
12.7.2.22 MDIHistory - MDI-Befehlsverlaufs-Widget . . . 1170
12.7.2.23 MDITouchy - Touchscreen-MDI-Eingabe-Widget . . . 1171
12.7.2.24 OriginOffsetView - Ursprungsansicht und Einstellungs-Widget . . . 1173
12.7.2.25 StateEnableGridlayout - Controller State Enabled Container Widget . . . 1174
12.7.2.26 MachineLog - Machine Events Journal Display Widget . . . 1174
12.7.2.27 JointEnableWidget - FIXME . . . 1175
12.7.2.28 StatusImageSwitcher - Controller Status Image Switching Widget . . . 1175
12.7.2.29 FileManager - File Loading Selector Widget . . . 1175
12.7.2.30 RadioAxisSelector - FIXME . . . 1176
12.7.2.31 ToolOffsetView - Tools Offsets View And Edit Widget . . . 1177
12.7.2.32 BasicProbe - Einfaches Fräs-Tast-Widget . . . 1179
12.7.2.33 VersaProbe - Mill Probing Widget . . . 1180
12.7.3 Dialog-Widgets . . . 1180
12.7.3.1 LcncDialog - Allgemeines Nachrichtendialog-Widget . . . 1181
12.7.3.2 ToolDialog - Dialog-Widget für den manuellen Werkzeugwechsel . . . 1182
12.7.3.3 FileDialog - Dialog Widget zum Laden und Speichern von Dateien . . . 1183
12.7.3.4 OriginOffsetDialog - Dialogfeld-Widget für die Einstellung des Ursprungsversatzes . . . 1184
12.7.3.5 ToolOffsetDialog - Dialogfenster-Widget zur Einstellung des Werkzeugversatzes . . . 1185
12.7.3.6 MacroTabDialog - Dialog-Widget zum Starten von Makros . . . 1185
12.7.3.7 CamViewDialog - WebCam Part Alignment Dialog Widget . . . 1185
12.7.3.8 EntryDialog - Edit Line Dialog Widget . . . 1185
12.7.3.9 CalculatorDialog - Calculator Dialog Widget . . . 1186
12.7.3.10 RunFromLine - Run-From-Line Dialog Widget . . . 1187
12.7.3.11 VersaProbeDialog - Part Touch Probing Dialog Widget . . . 1188
12.7.3.12 MachineLogDialog - Machine and Debugging Logs Dialog Widget . . . 1189
12.7.4 Other Widgets . . . 1189
12.7.4.1 NurbsEditor - NURBS Editing Widget . . . 1190
12.7.4.2 JoyPad - 5 button D-pad Widget . . . 1190
12.7.4.3 *WebWidget* . . . 1192
12.7.5 BaseClass/Mixin-Widgets . . . 1193

12.7.5.1 IndicatedPushButtons . . . 1193
12.7.6 Import-Only Widgets . . . 1196
12.7.6.1 Automatische Höhe . . . 1196
12.7.6.2 G-Code Dienstprogramm . . . 1196
12.7.6.3 Facing . . . 1196
12.7.6.4 Loch-Kreis (engl. hole circle) . . . 1196
12.7.6.5 Qt NGCGUI . . . 1196
12.7.6.6 Qt PDF . . . 1198
12.7.6.7 Qt Vismach . . . 1198
12.8 QtVCP Libraries modules . . . 1198
12.8.1 Status . . . 1198
12.8.1.1 Anwendung . . . 1198
12.8.1.2 Beispiel . . . 1199
12.8.2 Info . . . 1199
12.8.2.1 Verfügbare Daten und Voreinstellungen . . . 1199
12.8.2.2 User message dialog info . . . 1200
12.8.2.3 Embedded program info . . . 1201
12.8.2.4 Helpers . . . 1201
12.8.2.5 Anwendung . . . 1201
12.8.3 Action . . . 1201
12.8.3.1 Helpers . . . 1202
12.8.3.2 Anwendung . . . 1202
12.8.4 Tool . . . 1204
12.8.4.1 Helpers . . . 1204
12.8.5 Path . . . 1205
12.8.5.1 Referenzierte Pfade . . . 1205
12.8.5.2 Helpers . . . 1206
12.8.5.3 Anwendung . . . 1206
12.8.6 VCPWindow . . . 1207
12.8.6.1 Anwendung . . . 1207
12.8.7 Aux_program_loader . . . 1207
12.8.7.1 Helpers . . . 1207
12.8.7.2 Anwendung . . . 1208
12.8.8 Keylookup . . . 1208
12.8.8.1 Anwendung . . . 1208
12.8.8.2 Key Defines . . . 1209
12.8.9 Messages . . . 1212
12.8.9.1 Eigenschaften . . . 1212
12.8.9.2 Beispiele . . . 1212

12.8.10 Notify . . . 1213
12.8.10.1 Eigenschaften . . . 1213
12.8.11 Preferences . . . 1214
12.8.12 Player . . . 1214
12.8.12.1 Töne (engl. sounds) . . . 1214
12.8.12.2 Anwendung . . . 1215
12.8.12.3 Beispiel . . . 1215
12.8.13 Virtuelle Tastatur . . . 1215
12.8.14 Toolbar Actions . . . 1215
12.8.14.1 Aktionen . . . 1215
12.8.14.2 Untermenüs . . . 1216
12.8.14.3 Anwendung . . . 1216
12.8.14.4 Beispiele . . . 1216
12.8.15 Qt Vismach Machine Graphics library . . . 1216
12.8.15.1 Integrierte Beispiele . . . 1217
12.8.15.2 Primitives-Bibliothek . . . 1217
12.8.15.3 Anwendung . . . 1219
12.8.15.4 Mehr zum Thema . . . 1219
12.9 QtVismach . . . 1219
12.9.1 Einführung . . . 1220
12.9.2 Hierarchy of Machine Design . . . 1221
12.9.3 Start the script . . . 1222
12.9.4 HAL pins. . . . 1222
12.9.5 Erstellen von Teilen . . . 1222
12.9.5.1 Importieren von STL- oder OBJ-Dateien . . . 1222
12.9.5.2 Aufbau aus geometrischen Primitiven . . . 1223
12.9.6 Moving Model Parts . . . 1223
12.9.6.1 Translating Model parts . . . 1224
12.9.6.2 Rotating Model Parts . . . 1224
12.9.7 Animating Parts . . . 1224
12.9.7.1 HalTranslate . . . 1224
12.9.7.2 HalRotate . . . 1225
12.9.7.3 *HalToolCylinder* . . . 1225
12.9.7.4 *HalToolTriangle* . . . 1225
12.9.8 Zusammenbau des Modells . . . 1225
12.9.9 Weitere Funktionen . . . 1226
12.9.10 Tips . . . 1227
12.9.11 Grundstruktur eines QtVismach-Skripts . . . 1228
12.9.12 Builtin Vismach Sample Panels . . . 1229

12.10 QtVCP: Building Custom Widgets . . . 1229
12.10.1 Übersicht . . . 1229
12.10.1.1 Widgets . . . 1229
12.10.1.2 Qt Designer . . . 1229
12.10.1.3 Initialization Process . . . 1230
12.10.1.4 Aufräum-Prozess . . . 1230
12.10.2 Custom HAL Widgets . . . 1231
12.10.3 Benutzerdefinierte Controller-Widgets mit STATUS . . . 1232
12.10.3.1 In The *Imports* Section . . . 1233
12.10.3.2 Im Abschnitt *Bibliotheken instanziieren* . . . 1233
12.10.3.3 Im Abschnitt "Benutzerdefinierte Widget-Klassendefinition" . . . 1234
12.10.4 Benutzerdefinierte Controller-Widgets mit Aktionen . . . 1236
12.10.5 Stylesheet-Eigenschaftsänderungen auf der Grundlage von Ereignissen . . . 1238
12.10.6 Verwenden von Stylesheets zum Ändern benutzerdefinierter Widget-Eigenschaften 1239
12.10.7 Widget-Plugins . . . 1239
12.10.7.1 Gridlayout Example . . . 1240
12.10.7.2 SystemToolbutton Example . . . 1240
12.10.7.3 Erstellen eines Plugins mit einem Dialogfeld "MenuEntry" . . . 1241
12.11 QtVCP Handler File Code Snippets . . . 1244
12.11.1 Preference File Loading/Saving . . . 1244
12.11.2 Use QSettings To Read/Save Variables . . . 1244
12.11.3 Add A Basic Style Editor . . . 1245
12.11.4 Dialog-Eintrag anfordern . . . 1245
12.11.5 Sprechen Sie eine Startup-Begrüßung . . . 1246
12.11.6 ToolBar-Funktionen . . . 1247
12.11.7 Add HAL Pins That Call Functions . . . 1248
12.11.8 Add A Special Max Velocity Slider Based On Percent . . . 1248
12.11.9 Kontinuierlichen Jog ein- und ausschalten . . . 1249
12.11.10 Class Patch The File Manager Widget . . . 1250
12.11.11 Adding Widgets Programmatically . . . 1251
12.11.12 Objekte periodisch aktualisieren/auslesen . . . 1254
12.11.13 External Control With ZMQ . . . 1255
12.11.13.1 ZMQ Messages Reading . . . 1255
12.11.13.2 ZMQ Messages Writing . . . 1257
12.11.14 Sending Messages To Status Bar Or Desktop Notify Dialogs . . . 1258
12.11.15 Fokusänderungen abfangen . . . 1258
12.11.16 Read Command Line Load Time Options . . . 1259
12.12 QtVCP Development . . . 1260
12.12.1 Übersicht . . . 1260

12.12.2 Builtin Locations . . . 1260
12.12.3 QtVCP vom Startup bis zum Herunterfahren . . . 1261
12.12.3.1 QtVCP Startup . . . 1261
12.12.3.2 QtVCP Herunterfahren (engl. shutdown) . . . 1261
12.12.4 Path Information . . . 1261
12.12.5 Idiosyncrasies . . . 1262
12.12.5.1 Error Code Collecting . . . 1262
12.12.5.2 Log Rate . . . 1262
12.12.5.3 Keybinding . . . 1263
12.12.5.4 Preference File . . . 1263
12.12.5.5 Widget Special Setup Functions . . . 1263
12.12.5.6 Dialogs . . . 1263
12.12.5.7 Styles (Themes) . . . 1264

**13 Programmierung der Benutzeroberfläche 1265**

13.1 Panelui . . . 1265
13.1.1 Einführung . . . 1265
13.1.2 Laden von Befehlen . . . 1265
13.1.3 panelui.ini Dateireferenz . . . 1266
13.1.4 Übersicht zu Internen Anweisungen . . . 1268
13.1.5 ZMQ-Nachrichten . . . 1271
13.1.6 Handler Dateierweiterung . . . 1271
13.2 The LinuxCNC Python module . . . 1272
13.2.1 Einführung . . . 1272
13.2.2 Verwendungsmuster für die LinuxCNC NML-Schnittstelle . . . 1273
13.2.3 Lesen des LinuxCNC-Status . . . 1273
13.2.3.1 linuxcnc.stat-Attribute . . . 1273
13.2.3.2 Das ”Achsen”-Wörterbuch . . . 1279
13.2.3.3 Das `Gelenk`(engl. joint) Wörterbuch . . . 1279
13.2.3.4 Das `Spindel`-Wörterbuch . . . 1280
13.2.4 Vorbereitung des Sendens von Befehlen . . . 1281
13.2.5 Senden von Befehlen über *linuxcnc.command* . . . 1282
13.2.5.1 `linuxcnc.command` Attribute . . . 1282
13.2.5.2 `linuxcnc.command` Methoden: . . . 1282
13.2.6 Lesen des Fehlerkanals . . . 1286
13.2.7 Lesen von INI-Datei Werten . . . 1286
13.2.8 Der Typ `linuxcnc.positionlogger` . . . 1287
13.2.8.1 Members . . . 1287
13.2.8.2 Methoden . . . 1287

13.3 The HAL Python module . . . 1288
13.3.1 Grundlegende Verwendung (engl. basic usage) . . . 1288
13.3.2 Funktionen . . . 1288
13.4 GStat Python Module . . . 1291
13.4.1 Einführung . . . 1291
13.4.2 Beispiel für einen GStat-Code . . . 1291
13.4.2.1 Codemuster für HAL-Komponenten . . . 1292
13.4.2.2 GladeVCP Python-Erweiterung Code-Muster . . . 1292
13.4.2.3 QtVCP Python-Erweiterungscode-Muster . . . 1293
13.4.3 Nachrichten . . . 1294
13.4.4 Funktionen . . . 1300
13.4.5 Bekannte Probleme . . . 1302
13.5 Vismach . . . 1302
13.5.1 Start the script . . . 1304
13.5.2 Erstellen der HAL-Pins. . . . 1304
13.5.3 Erstellen von Teilen . . . 1304
13.5.4 Bewegliche Teile . . . 1305
13.5.5 Animating Parts . . . 1305
13.5.6 Zusammenbau des Modells. . . . 1306
13.5.7 Weitere Funktionen . . . 1307
13.5.8 Grundstruktur eines Vismach-Skripts. . . . 1308

**III Glossar, Copyright & Geschichte 1309**

**14 Umschlagseite 1310**

**15 Glossar 1311**

**16 Copyright 1317**
16.1 Juristischer Abschnitt . . . 1317
16.1.1 Copyright-Bedingungen . . . 1317
16.1.2 GNU Free Documentation License . . . 1317

**17 LinuxCNC Geschichte 1322**
17.1 Origin . . . 1322
17.1.1 Namensänderung . . . 1323
17.1.2 Zusätzliche Informationen . . . 1323

**18 Index 1324**

# 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: **easy discovery and testing without installation with the Live Image** easy installation from the Live Image **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)** graphical interface creation tools (Glade, QT) **ein Interpreter für *G-Code* (die Programmiersprache für RS-274-Werkzeugmaschinen)** ein System zur Bewegungsplanung in Echtzeit mit Vorausschau **Betrieb von Low-Level-Maschinenelektroni 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). * It can make coordinated moves with up to 9 axes and up to 16 extra axes can be controlled individually. * It supports a variety of hardware interfaces. * 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 is available as ready-to-use packages for Debian distributions.

## 1.1.3 Getting Help

### 1.1.3.1 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.

### 1.1.3.2 IRC

IRC stands for Internet Relay Chat. It is a live connection to other LinuxCNC users. The LinuxCNC IRC channel is #linuxcnc on 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:

- *For text*: https://pastebin.com/, http://pastie.org/, https://gist.github.com/
- *For pictures*: https://imagebin.org/, https://imgur.com/, https://bayimg.com/
- *For files*: https://filedropper.com/, https://filefactory.com/, https://1fichier.com/

### 1.1.3.3 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

#### 1.1.3.4 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

Report bugs on the LinuxCNC Github github bug tracker.

## 1.2 System Requirements

### 1.2.1 Mindestanforderungen

The minimum system to run LinuxCNC and Debian / Ubuntu may vary depending on the exact usage. Stepper systems in general require faster threads to generate step pulses than servo systems. You can use the Live CD to test the software before committing to a permanent installation on a computer. Keep in mind that the Latency Test numbers are more important than the processor speed for software step generation. More information on the Latency Test is here. In addition LinuxCNC needs to be run on an operating system that uses a specially modified kernel, see Kernel and Version Requirements.

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

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

- 1.2 GHz 64-bit x86 processor or Raspberry Pi 4 or better.
- 512 MB of RAM, 4 GB with GUI to avoid surprises
- No hard disk for Live CD, 8 GB or more for permanent installation
- Graphics card capable of at least 1024x768 resolution, which is not using the NVidia or ATI fglrx proprietary drivers. Modern onboard graphic chipsets seem to generally be OK.
- Internet connection (not strictly needed, but very useful for updates and for communicating with the LinuxCNC community)

Minimum hardware requirements change as Linux distributions evolve so check the Debian web site for details on the Live CD you’re using. Older hardware may benefit from selecting an older version of the Live CD when available.

If you plan not to rely on the distribution of readily executable programs (”binaries”) but aim at contributing to the source tree of LinuxCNC, then there is a good chance you want a second computer to perform the compilation. Even though LinuxCNC and your developments could likely be executed at the same time with respect to disk space, RAM and even CPU speed, a machine that is busy will have worse latencies, so you are unlikely to compile your source tree and produce chips at the same time.

## 1.2.2 Kernel- und Versionsanforderungen

LinuxCNC requires a kernel modified for realtime use to control real machine hardware. It can, however run on a standard kernel in simulation mode for purposes such as checking G-code, testing config files and learning the system. To work with these kernel versions there are two versions of LinuxCNC distributed. The package names are ”linuxcnc” and ”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

It is also possible to run LinuxCNC with RTAI in user-space mode. As with Xenomai you will need to compile from source to do this.

## 1.2.3 Problematische Hardware

### 1.2.3.1 Laptops

Laptops are not generally suited to real time software step generation. Again a Latency Test run for an extended time will give you the info you need to determine suitability.

#### 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 Getting LinuxCNC

This section describes the recommended way to download and make a fresh install of LinuxCNC. There are also Alternate Install Methods for the adventurous. If you have an existing install that you want to upgrade, go to the Updating LinuxCNC section instead.

---

**Anmerkung**
To operate machinery LinuxCNC requires a special kernel with real-time extensions. There are three possibilities here: preempt-rt, RTAI or Xenomai. In addition there are two versions of LinuxCNC which work with these kernels. See the table below for details. However for code testing and simulation it is possible to run the linuxcnc-uspace application on a stock kernel.

---

Fresh installs of LinuxCNC are most easily created using the Live/Install Image. This is a hybrid ISO filesystem image that can be written to a USB storage device or a DVD and used to boot a computer. At boot time you will be given a choice of booting the ”Live” system (to run LinuxCNC without making any permanent changes to your computer) or booting the Installer (to install LinuxCNC and its operating system onto your computer’s hard drive).

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

This section describes some methods for downloading the Live/Install image.

#### 1.3.1.1 Normales Herunterladen

Software for LinuxCNC to download is presented on the project’s Downloads page. Most users will aim for the disk image for Intel/AMD PCs, the URL will resemble https://www.linuxcnc.org/iso/linuxcnc_2.9.8-amd64.hybrid.iso.

For the Raspberry Pi, multiple images are provided to address differences between the RPi4 and RPi5.

---

**Anmerkung**
Do not use the regular Raspbian distribution for LinuxCNC that may have shipped with your RPi starter kit - that will not have the real-time kernel and you cannot migrate from Raspbian to Debian’s kernel image.

---

#### 1.3.1.2 Herunterladen mit zsync

zsync is a download application that efficiently resumes interrupted downloads and efficiently transfers large files with small modifications (if you have an older local copy). Use zsync if you have trouble downloading the image using the Normal Download method.

zsync unter Linux

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

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

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

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

**zsync unter Windows** There is a Windows port of zsync. It works as a console application and can be downloaded from 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-2.9.8-amd64.iso
```

oder

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

1. Vergleichen Sie dann mit diesen Prüfsummen

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

**Überprüfen Sie md5sum auf Windows oder Mac** Windows does not come with an md5sum program, but there are alternatives. More information can be found at: How To MD5SUM

### 1.3.2 Schreiben des Abbilds auf ein bootfähiges Gerät

The LinuxCNC Live/Install ISO Image is a hybrid ISO image which can be written directly to a USB storage device (flash drive) or a DVD and used to boot a computer. The image is too large to fit on a CD.

#### 1.3.2.1 Raspberry Pi Image

The Raspbery Pi image is a complete SD card image and should be written to an SD card with the [Raspberry Pi Imager App](https://www.raspberrypi.com/software/). Note that the imager app can open the .zip file directly, no need to expand.

#### 1.3.2.2 AMD-64 (x86-64, PC) Image using GUI tools

Download and install [Balena Etcher](https://etcher.balena.io/#download-etcher) (Linux, Windows, Mac) and write the downloaded image to a USB drive.

If your image fails to boot then please also try Rufus. It looks more complicated but seems to be more compatible with various BIOSes.

#### 1.3.2.3 Command line - Linux

1. Schließen Sie ein USB-Speichergerät an (z. B. ein Flash-Laufwerk oder ein Gerät vom Typ Thumb Drive).
2. Determine the device file corresponding to the USB flash drive. This information can be found in the output of `sudo dmesg` after connecting the device. `cat /proc/partitions` may also be helpful.
3. Verwenden Sie den Befehl `dd`, um das Image auf Ihr USB-Speichergerät zu schreiben. Wenn Ihr Speichergerät zum Beispiel als `/dev/sde` angezeigt wird, verwenden Sie diesen Befehl:

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

#### 1.3.2.4 Command line - MacOS

1. Öffnen Sie ein Terminal und geben Sie ein

```
diskutil list
```

2. Stecken Sie den USB-Stick ein und notieren Sie sich den Namen der neuen Festplatte, die angezeigt wird, z. B. /dev/disk5
3. unmount the USB. The number found above should be substitued in place of the N

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

4. Transfer the data with dd, as for Linux above. Note that the disk name has an added ”r” at the begining

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

Schreiben des Abbilds auf eine DVD unter Linux

1. Legen Sie einen DVD-Rohling in Ihren Brenner ein. Ein Fenster ”CD/DVD Creator” oder ”Disc-Typ auswählen” wird angezeigt. Schließen Sie es, da wir es nicht verwenden werden.
2. Suchen Sie das heruntergeladene Bild im Dateibrowser.
3. Klicken Sie mit der rechten Maustaste auf die ISO-Image-Datei und wählen Sie Write to Disc.

4. Wählen Sie die Schreibgeschwindigkeit. Es wird empfohlen, mit der niedrigstmöglichen Geschwindigkeit zu schreiben.
5. Starten Sie den Brennvorgang.
6. Wenn ein Fenster ”Wählen Sie einen Dateinamen für das Disk-Image” erscheint, wählen Sie einfach OK.

Schreiben des Abbilds auf eine DVD unter Windows

1. 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. Open Infra Recorder, and select the *Actions* menu, then *Burn image*.

Schreiben des Abbilds auf eine DVD unter Mac OSX

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

### 1.3.3 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**
If the system does not boot from the DVD or USB stick, it may be necessary to change the boot order in the PC BIOS.

---

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.

At the time of writing the Live Image is only available with the preempt-rt kernel and a matching LinuxCNC. On some hardware this might not offer good enough latency. There is an experimental version available using the RTAI realtime kernel which will often give better latency.

### 1.3.4 LinuxCNC installieren

To install LinuxCNC from the Live CD select *Install (Graphical)* at bootup.

### 1.3.5 Updates to LinuxCNC

With the normal install the Update Manager will notify you of updates to LinuxCNC when you go on line and allow you to easily upgrade with no Linux knowledge needed. It is OK to upgrade everything except the operating system when asked to.

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

### 1.3.6 Install Problems

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 Alternate Install Methods

The easiest, preferred way to install LinuxCNC is to use the Live/Install Image as described above. That method is as simple and reliable as we can make it, and is suitable for novice users and experienced users alike. However, this will typically replace any existing operating system. If you have files on the target PC that you want to keep, then use one of the methods described in this section.

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)

| **Debian Trixie** | **amd64 & arm64** | **preempt-rt** | **linuxcnc-uspace** | **Maschinensteuerung und -simulation** |
|---|---|---|---|---|
| Debian Troxie | amd64 | RTAI | linuxcnc | machine control |
| Distribution | Architektur | Kernel | Paket-Name | Typische Verwendung |
| Debian Bookworm | amd64 & arm64 | preempt-rt | linuxcnc-uspace | Maschinensteuerung und -simulation |
| Debian Bookworm | amd64 | RTAI | linuxcnc | machine control |
| Debian Bullseye | amd64 | preempt-rt | linuxcnc-uspace | Maschinensteuerung und -simulation |
| Jede (engl. any) | Jede (engl. any) | LinuxCNC direkt nach der Installation | linuxcnc-uspace | simulation ONLY |

**Anmerkung**
LinuxCNC v2.9 is not supported on Debian 9 or older.

**Preempt-RT-Kernel** The Preempt-rt kernels are available for Debian from the regular debian.org archive. The package is called `linux-image-rt-*`. Simply install the package in the same way as any other package from the Synaptic Package manager or with apt-get at the command-line.

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

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

LinuxCNC and the RTAI kernel are now only available for 64-bit OSes but there are very few surviving systems that can not run a 64-bit OS.

#### 1.3.7.1 Installing on Debian Trixie (with Preempt-RT kernel)

1. Install Debian Trixie (Debian 13), amd64 version. You can download the installer here: https://www.debi
   distrib/
2. After burning the iso and booting up if you don't want Gnome desktop select *Advanced Options > Alternative desktop environments* and pick the one you like. Then select *Install* or *Graphical Install*.

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

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

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

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

1. Install the Preempt-RT kernel and modules

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

2. Re-boot, and select the Linux 6.1.0-10-rt-amd64 kernel. The exact kernel version might be different, look for the "-rt" suffix. This might be hidden in the "Advanced options for Debian Bookworm" sub-menu in Grub. When you log in, verify that `PREEMPT RT` is reported by the following command.

```
uname -v
```

3. 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.

4. Add the LinuxCNC Archive Signing Key to your apt keyring by downloading [the LinuxCNC installer script](https://www.linuxcnc.org/linuxcnc-install.sh) You will need to make the script executable to run it:

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

Then you can run the installer:

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

#### 1.3.7.2 Installing on Debian Trixie (with experimental RTAI kernel)

1. This kernel and LinuxCNC version can be installed on top of the Live DVD install, or alternatively on a fresh Install of Debian Trixie 64-bit as described above.
2. You can add the LinuxCNC Archive signing key and repository information by downloading and running the installer script as decribed above. If an RTAI kernel is detected it will stop before installing any packages.
3. Aktualisieren Sie die Paketliste von linuxcnc.org

   ```
   sudo apt-get update
   ```

4. Remove the existing uspace version of LinuxCNC and install the new realtime kernel, RTAI and the rtai version of linuxcnc.

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

Reboot the machine, ensuring that the system boots from the new 5.4.258-rtai kernel.

#### 1.3.7.3 Installing on Raspbian 12

Don’t do that. The latencies are too bad with the default kernel and the PREEMPT_RT (the RT is important) kernel of Debian does not boot on the Pi (as of 1/2024). Please refer to the images provided on this web site.

## 1.4 Running LinuxCNC

### 1.4.1 Aufrufen von LinuxCNC

After installation, LinuxCNC starts just like any other Linux program: run it from the terminal by issuing the command *linuxcnc*, or select it in the *Applications -> CNC* menu.

### 1.4.2 Configuration Launcher

When starting LinuxCNC (from the CNC menu or from the command line without specifying an INI file) the Configuration Selector dialog starts.

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

Der Konfigurations Selector bietet eine Auswahl an Konfigurationen:

- *Meine Konfigurationen* - Benutzerkonfigurationen in linuxcnc/configs in Ihrem Home-Verzeichnis.
- *Beispielkonfigurationen* - Beispielkonfigurationen werden, wenn ausgewählt, nach linuxcnc/configs kopiert. Sobald eine Beispielkonfiguration in Ihr lokales Verzeichnis kopiert wurde, bietet der Launcher sie als „Meine Konfigurationen“ an. Die Namen, unter denen diese lokalen Konfigurationen angezeigt werden, entsprechen den Namen der Verzeichnisse innerhalb des Verzeichnisses configs/:
  - *sim* - Konfigurationen, die simulierte Hardware enthalten. Diese können zum Testen oder Lernen, wie LinuxCNC funktioniert, verwendet werden.
  - by_interface’ - Konfigurationen nach GUI geordnet.
  - *by_machine* - Konfigurationen organisiert nach Maschine.
  - *apps* - Anwendungen, die kein Starten von linuxcnc erfordern, aber zum Testen oder Ausprobieren von Anwendungen wie PyVCP oder GladeVCP nützlich sein können.
  - *attic* - Veraltete oder historische Konfigurationen.

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

- *axis* - Tastatur- und Maus-GUI
- *craftsman* - Touch Screen GUI (no longer maintained ???)
- *gmoccapy* - Touchscreen-GUI
- *gscreen* - Touchscreen-GUI
- *pyvcp_demo* - Python Virtual Control Panel
- *qtaxis* - Touch Screen GUI, axis lookalike
- *qtdragon* - Touch Screen GUI
- *qtdragon_hd* - Touch Screen GUI, high definition
- *qtplasmac* - Touch Screen GUI, for plasma tables
- *qttouchy* - Touch Screen GUI
- *tklinuxcnc* - Keyboard and Mouse GUI (no longer maintained)
- *touchy* - Touchscreen-GUI
- *woodpecker* - Touch Screen GUI A GUI configuration directory may contain subdirectories with configurations that illustrate special situations or the embedding of other applications.

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
- scortbot erIII
- sherline
- smithy
- tormach

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

The *apps items* are typically either:

1. utilities that don’t require starting linuxcnc
2. demonstrations of applications that can be used with linuxcnc
   - info - erstellt eine Datei mit Systeminformationen, die für die Problemdiagnose nützlich sein können.
   - gladevcp - Beispiele für GladeVCP-Anwendungen.
   - halrun - Startet halrun in einem Terminal.
   - latency - Anwendungen zur Untersuchung der Latenz
     - latency-histogram-1 - histogram for single servo thread
     - latency-histogram - Histogram
     - latency-test - Standard-Test zur Bestimmung der Latenz
     - latency-plot - Streifendiagramm
   - parport - Anwendungen zum Testen von parport.
   - pyvcp - Beispiele für pyvcp-Anwendungen.
   - xhc-hb04 – Anwendungen zum Testen eines drahtlosen USB-MPG xhc-hb04

---

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

---

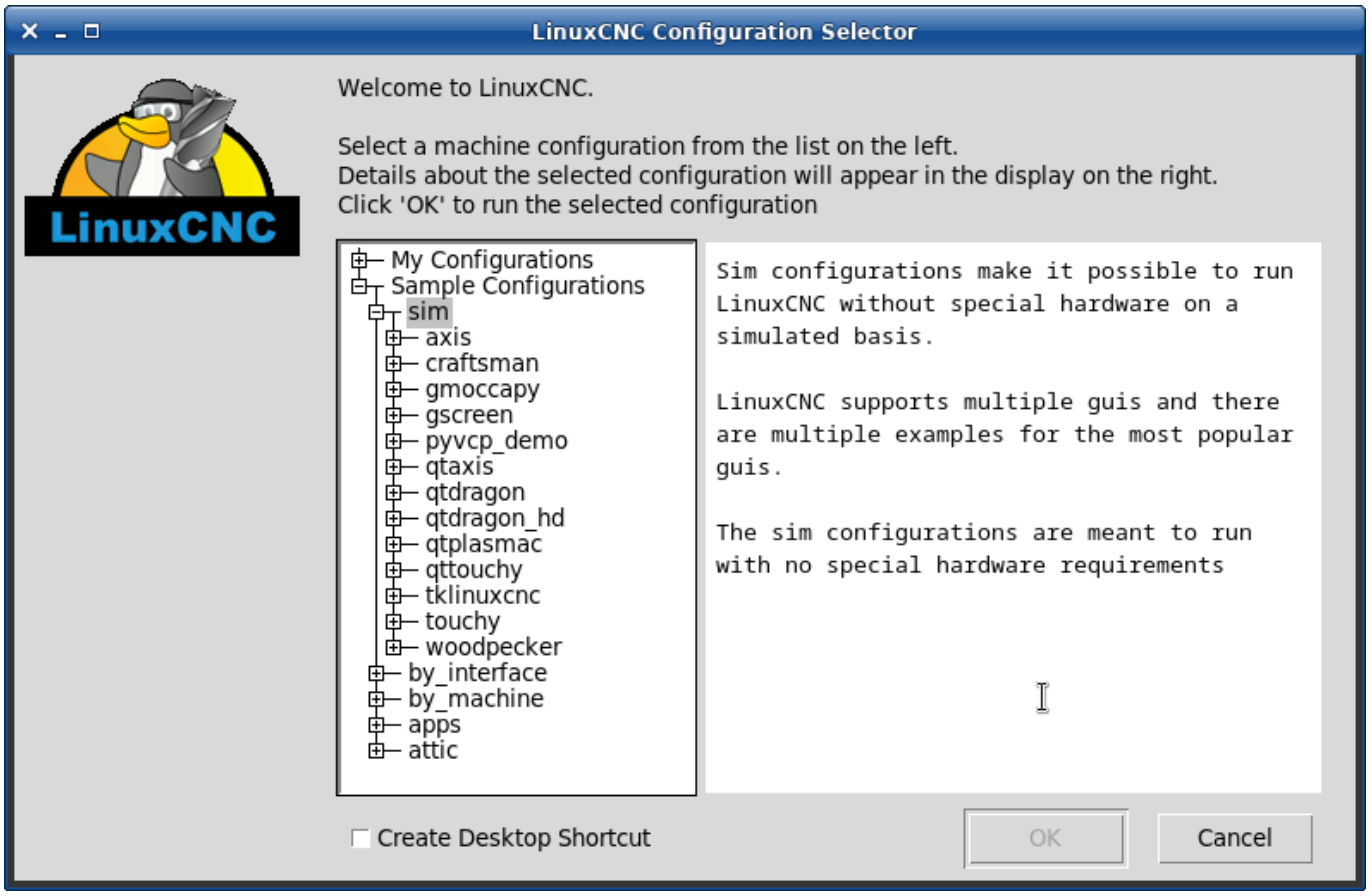



Abbildung 1.1: LinuxCNC-Konfigurationsauswahl

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 Updating LinuxCNC

Updating LinuxCNC to a new minor release (ie to a new version in the same stable series, for example from 2.9.7 to 2.9.8) is an automatic process if your PC is connected to the internet. You will see an update prompt after a minor release along with other software updates. If you don’t have an internet connection to your PC see Updating without Network.

### 1.5.1 Upgrade auf die neue Version

This section describes how to upgrade LinuxCNC from version 2.8.x to a 2.9.y version. It assumes that you have an existing 2.8 install that you want to update.

To upgrade LinuxCNC from a version older than 2.8, you have to first upgrade your old install to 2.8, then follow these instructions to upgrade to the new version.

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.

Furthermore, if you are running Ubuntu Precise, Debian Wheezy or Debian Buster it is well worth considering making a backup of the ”linuxcnc” directory on removable media and performing a clean

install of a newer OS and LinuxCNC version as these releases were EOL in 2017, 2018 and 2022 respectively. If you are running on Ubuntu Lucid then you will have to do this, as Lucid is no longer supported by LinuxCNC (it was EOL in 2013).

To upgrade major versions like 2.8 to 2.9 when you have a network connection at the machine you need to disable the old linuxcnc.org apt sources in the file /etc/apt/sources.list and add a new linuxcnc.org apt source for 2.9, then upgrade LinuxCNC.

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

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

You should be running on Debian Bullseye, Bookworm or Trixie or Ubuntu 20.04 ”Focal Fossa” or newer. LinuxCNC 2.9.y will not run on older distributions than these.

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

```
uname -r
6.1.0-10-rt-amd64
```

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

If you see `-rtai-` in the kernel name then you are running RTAI realtime. See below for the LinuxCNC version to install. RTAI packages are available for Bookworm and Buster but not currently for Bullseye.

#### 1.5.1.1 Apt Sources Konfiguration

- Open the `Software Sources` window. The process for doing this differs slightly on the three supported platforms:
  - Debian:
    - Click on `Applications Menu`, then `System`, then `Synaptic Package Manager`.
    - Klicken Sie in Synaptic auf das Menü `Einstellungen` und dann auf `Repositories`, um das Fenster `Softwarequellen` zu öffnen.
  - Ubuntu Precise:
    - Klicken Sie auf das Symbol ”Dash Home” oben links.
    - Geben Sie in das Feld ”Suche” den Begriff ”Software” ein und klicken Sie dann auf das Symbol ”Ubuntu Software Center”.
    - Klicken Sie im Ubuntu Software Center-Fenster auf das Menü ”Bearbeiten” und dann auf ”Softwarequellen...”, um das Fenster ”Softwarequellen” zu öffnen.
  - Ubuntu Lucid:
    - Klicken Sie auf das Menü ”System”, dann auf ”Verwaltung” und dann auf ”Synaptic Package Manager”.
    - Klicken Sie in Synaptic auf das Menü `Einstellungen` und dann auf `Repositories`, um das Fenster `Softwarequellen` zu öffnen.
- Wählen Sie im Fenster ”Software-Quellen” die Registerkarte ”Andere Software”.
- Löschen oder deaktivieren Sie alle alten linuxcnc.org-Einträge (lassen Sie alle nicht-linuxcnc.org-Zeilen unverändert).
- Klicken Sie auf die Schaltfläche ”Hinzufügen” und fügen Sie eine neue apt-Zeile hinzu. Die Zeile wird auf den verschiedenen Plattformen etwas anders aussehen:

Tabelle 1.1: Tabular overview on variants of the Operating System and the corresponding configuration of the repository. The configuration can be performed in the GUI of the package manager or in the file /etc/apt/sources.list.

| **OS / Realtime Version** | **Repository** |
|---|---|
| Debian Bullseye - preempt | `deb https://linuxcnc.org bullseye base 2.9-uspace` |
| Debian Bookworm - preempt | `deb https://linuxcnc.org bookworm base 2.9-uspace` |
| Debian Bookworm - RTAI | `deb https://linuxcnc.org bookworm base 2.9-rt` |
| Debian Trixie - preempt | `deb https://linuxcnc.org trixie base 2.9-uspace` |
| Debian Trixie - RTAI | `deb https://linuxcnc.org trixie base 2.9-rt` |

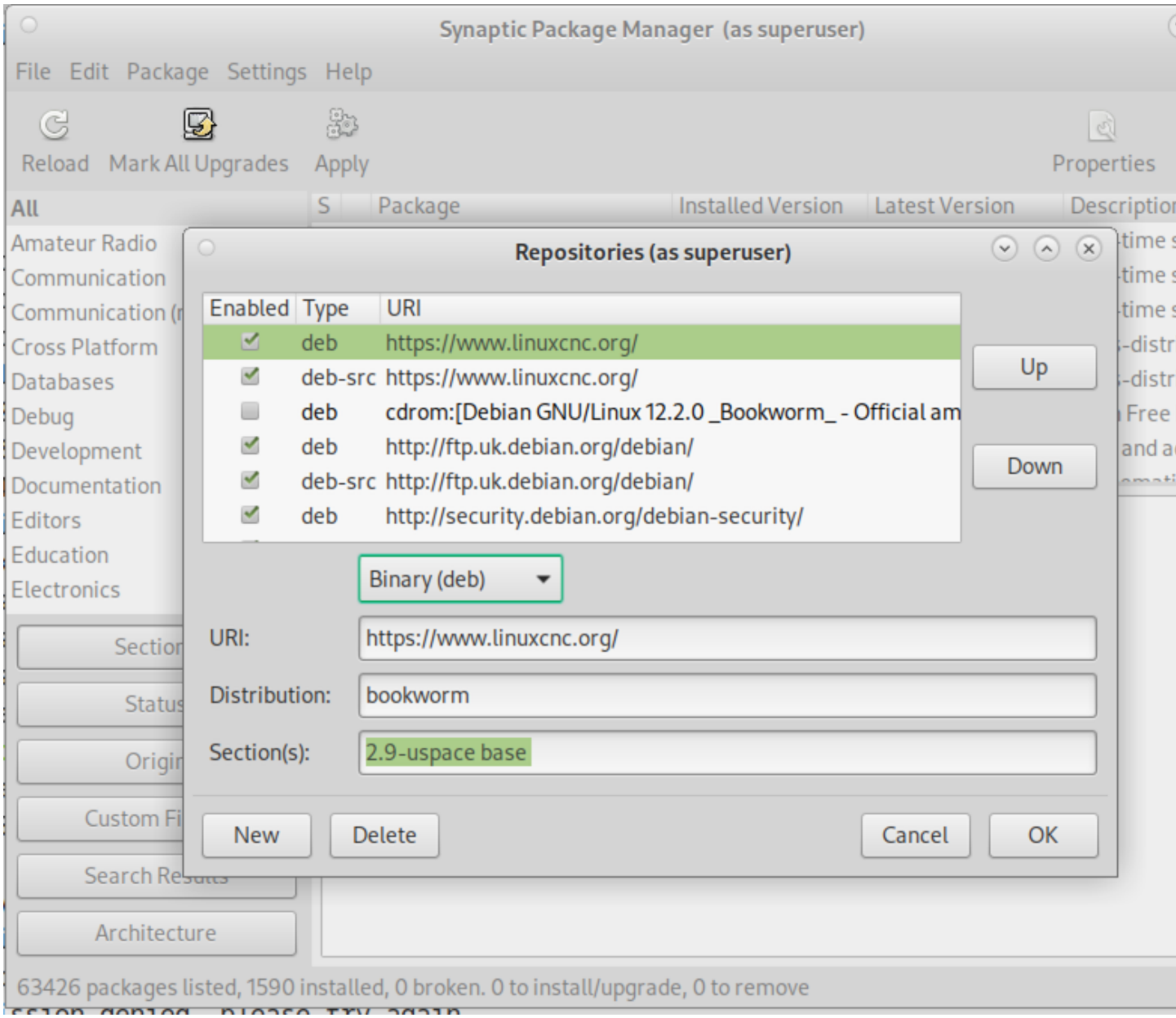



Abbildung 1.2: Figure with a screenshot of the repository configuration of the synaptic package manager.

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

#### 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 uses the 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.
- Click the check box to mark the new linuxcnc and linuxcnc-doc-* packages for upgrade. The package manager may select a number of additional packages to be installed, to satisfy dependencies that the new linuxcnc package has.
- Click the `Apply` button, and let your computer install the new package. The old linuxcnc package will be automatically upgraded to the new one.

#### 1.5.1.3 Ubuntu

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

### 1.5.2 Aktualisieren ohne Netzwerk

To update without a network connection you need to download the .deb then install it with dpkg. The .debs can be found in https://linuxcnc.org/dists/ .

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

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

Pick the OS from the list then pick the major version you want like 2.9-rt for RTAI or 2.9-uspace for preempt-rt.

Next pick the type of computer you have: binary-amd64 for 64-bit PC or binary-arm64 (64bit) for Raspberry Pi.

Next pick the version you want from the bottom of the list like *linuxcnc-uspace_2.9.8_amd64.deb* (choose the latest by date). Download the deb and copy it to your home directory. You can rename the file to something a bit shorter with the file manager like *linuxcnc_2.9.8.deb* then open a terminal and install it with the package manager with this command:

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

### 1.5.3 Updating Configuration Files for 2.9

#### 1.5.3.1 Stricter handling of pluggable interpreters

If you just run regular G-code and you don’t know what a pluggable interpreter is, then this section does not affect you.

A seldom-used feature of LinuxCNC is support for pluggable interpreters, controlled by the undocumented `[TASK]INTERPRETER` INI setting.

Versions of LinuxCNC before 2.9.0 used to handle an incorrect `[TASK]INTERPRETER` setting by automatically falling back to using the default G-code interpreter.

Since 2.9.0, an incorrect `[TASK]INTERPRETER` value will cause LinuxCNC to refuse to start up. Fix this condition by deleting the `[TASK]INTERPRETER` setting from your INI file, so that LinuxCNC will use the default G-code interpreter.

#### 1.5.3.2 Canterp

If you just run regular G-code and you don't use the `canterp` pluggable interpreter, then this section does not affect you.

In the extremely unlikely event that you are using `canterp`, know that the module has moved from `/usr/lib/libcanterp.so` to `/usr/lib/linuxcnc/canterp.so`, and the `[TASK]INTERPRETER` setting correspondingly needs to change from `libcanterp.so` to `canterp.so`.

#### 1.5.3.3 Spindle limits in the INI

It is now possible to add settings to the [SPINDLE] section of the INI file

MAX_FORWARD_VELOCITY = 20000 The maximum spindle speed (in rpm)

MIN_FORWARD_VELOCITY = 3000 The minimum spindle speed (in rpm)

MAX_REVERSE_VELOCITY = 20000 This setting will default to MAX_FORWARD_VELOCITY if omitted.

MIN_REVERSE_VELOCITY = 3000` This setting is equivalent to MIN_FORWARD_VELOCITY but for reverse spindle rotation. It will default to the MIN_FORWARD_VELOCITY if omitted.

INCREMENT = 200 Sets the step size for spindle speed increment / decrement commands. This can have a different value for each spindle. This setting is effective with AXIS and Touchy but note that some control screens may handle things differently.

HOME_SEARCH_VELOCITY = 100 - Accepted but currently does nothing

HOME_SEQUENCE = 0 - Accepted but currently does nothing

### 1.5.4 New HAL components

#### 1.5.4.1 Non-Realtime

mdro mqtt-publisher pi500_vfd pmx485-test qtplasmac-cfg2prefs qtplasmac-materials qtplasmac-plasmac2c qtplasmac-setup sim-torch svd-ps_vfd

#### 1.5.4.2 Echtzeit

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

### 1.5.5 New Drivers

A framework for controlling ModBus devices using the serial ports on many Mesa cards has been introduced. http://linuxcnc.org/docs/2.9/html/drivers/mesa_modbus.html

A new GPIO driver for any GPIO which is supported by the gpiod library is now included: http://linuxcnc.org/docs/2.9/html/drivers/hal_gpio.html

# 1.6 Linux FAQ

These are some basic Linux commands and techniques for new to Linux users. More complete information can be found on the web or by using the man pages.

## 1.6.1 Automatic Login

### 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

A man page (short for manual page) is a form of software documentation usually found on a Unix or Unix-like operating system like Linux.

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 Changing Directories

To change the working directory to the one one level up, i.e., the parent directory, in the terminal window type:

```
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 link:https://en.wikipedia.org/wiki/Separation_of_mechanism_and_policy, 2022-11-13

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 link:https://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

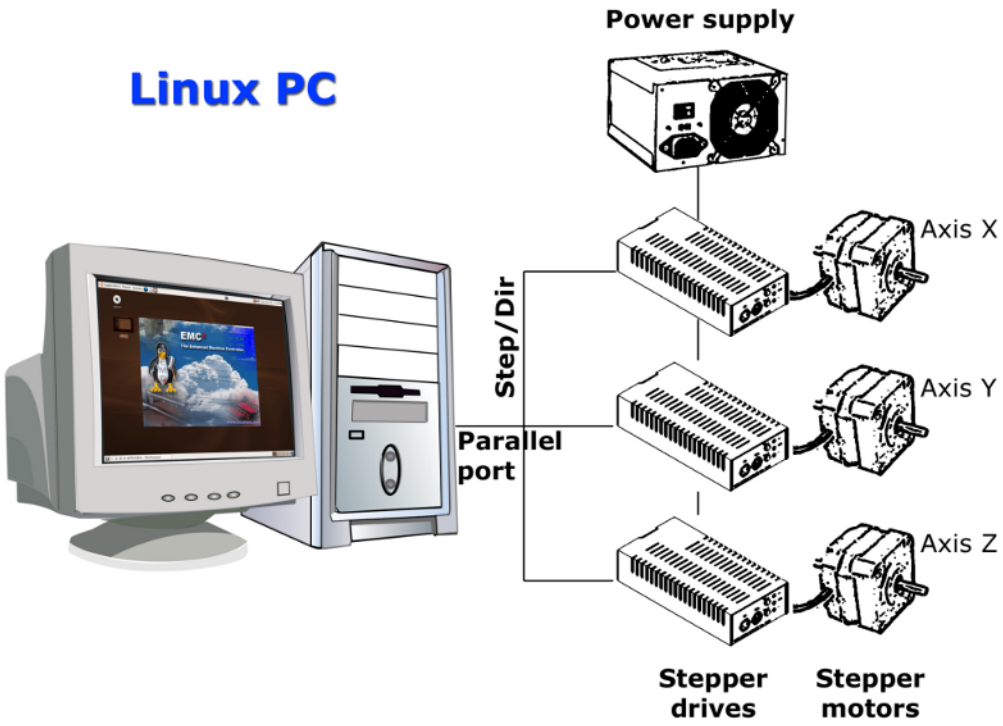



Abbildung 2.1: Simple LinuxCNC Controlled Machine

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:

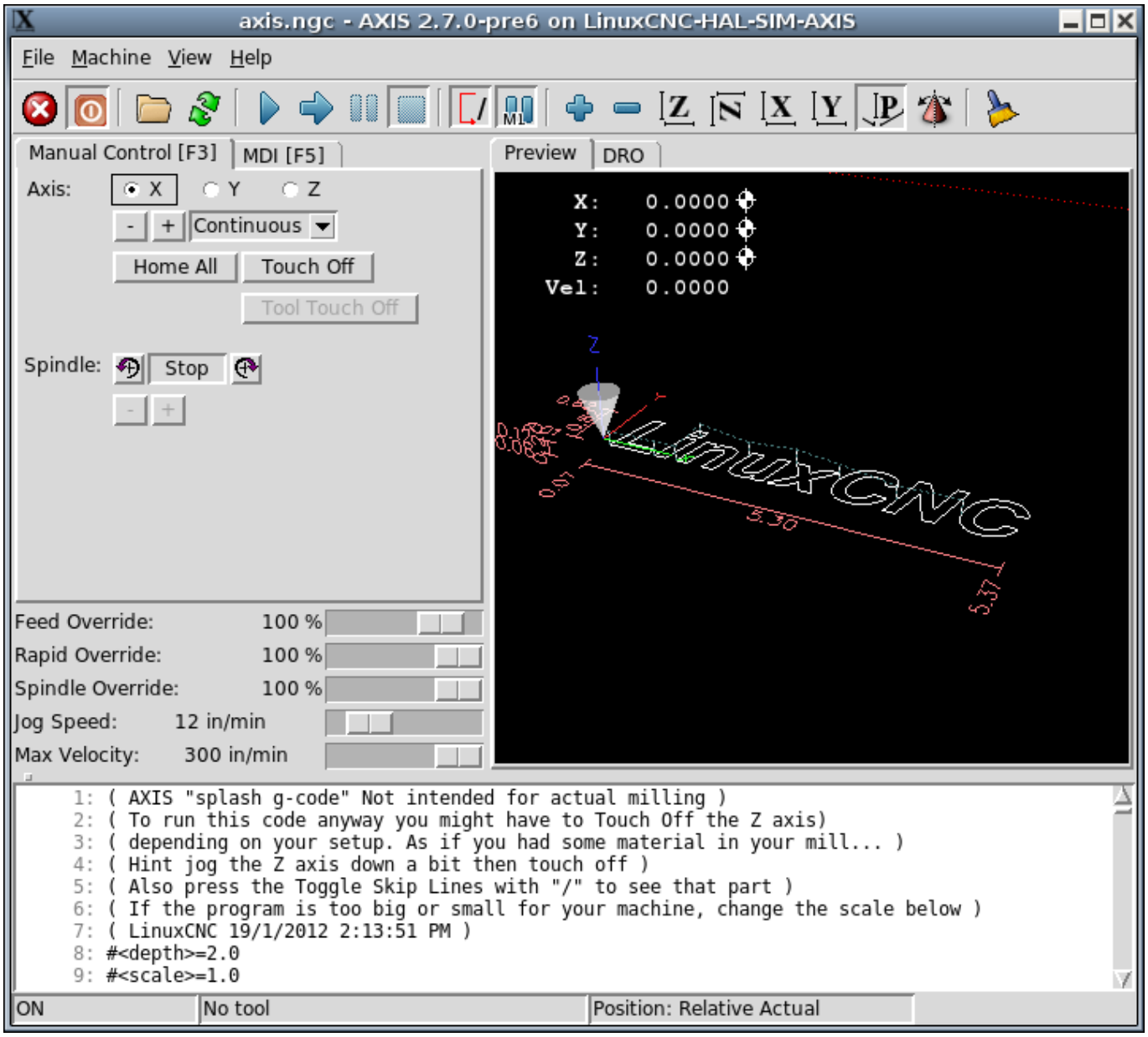



Abbildung 2.2: AXIS, the standard keyboard GUI interface

**Touchy**

Touchy, a touch screens GUI:

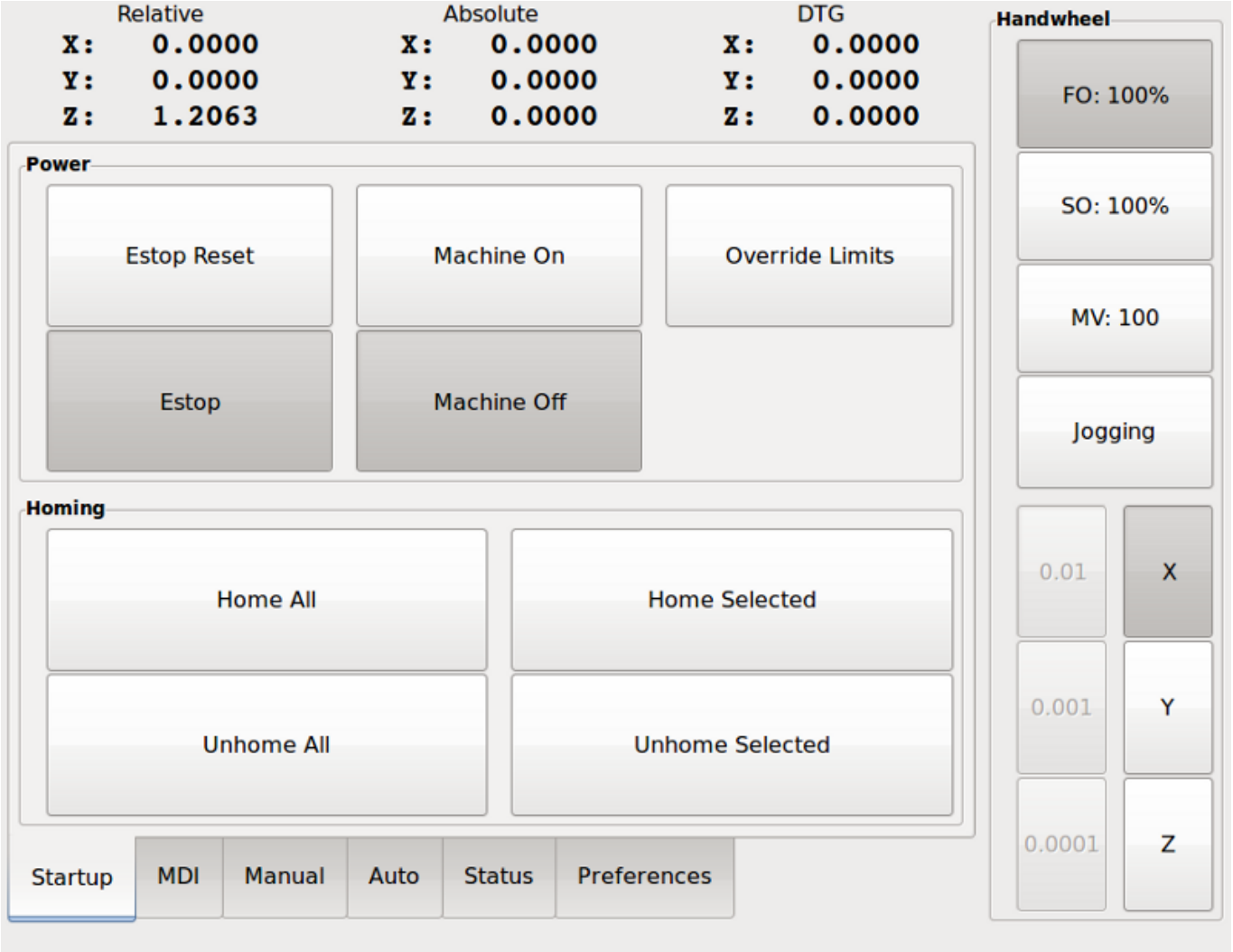



Abbildung 2.3: Touchy, a touch screen GUI

**Gscreen**

Gscreen, a user-configurable touch screen GUI:

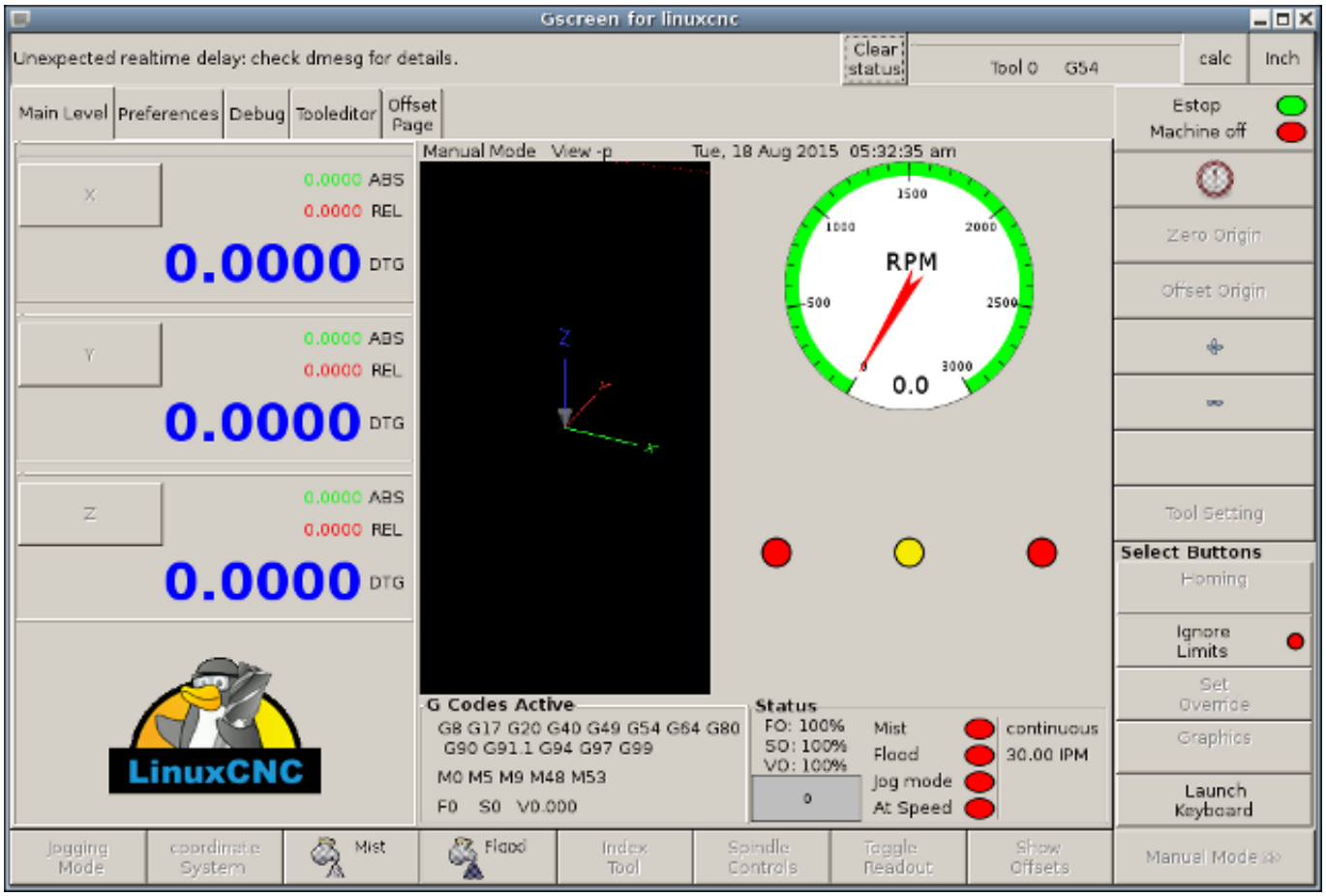



Abbildung 2.4: Gscreen, a configurable base touch screen GUI

**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:

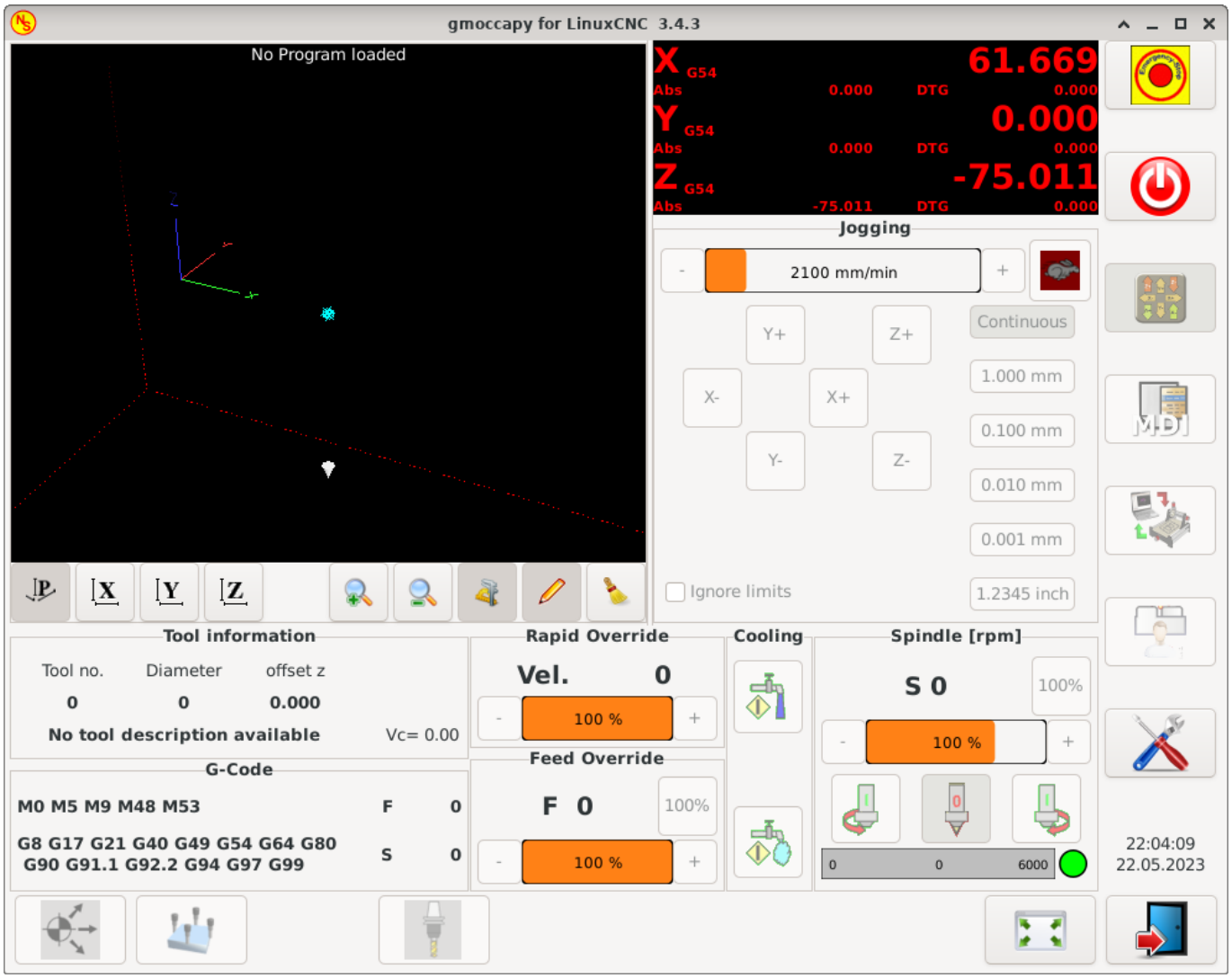



Abbildung 2.5: GMOCCAPY, a touch screen GUI based on Gscreen

**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 screenshot shows NGCGUI embedded into AXIS:

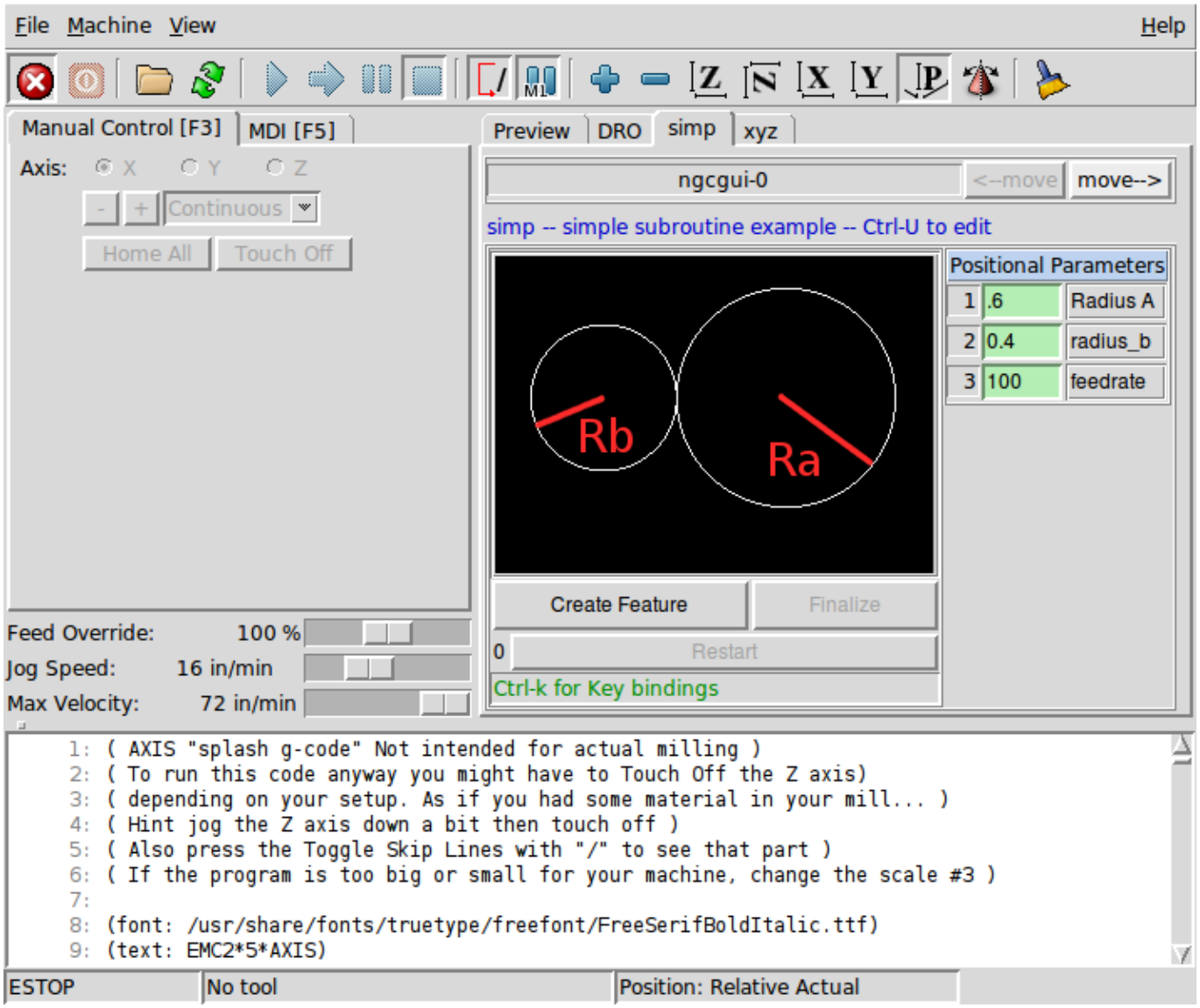



Abbildung 2.6: NGCGUI, a graphical interface integrated into AXIS

**TkLinuxCNC**

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

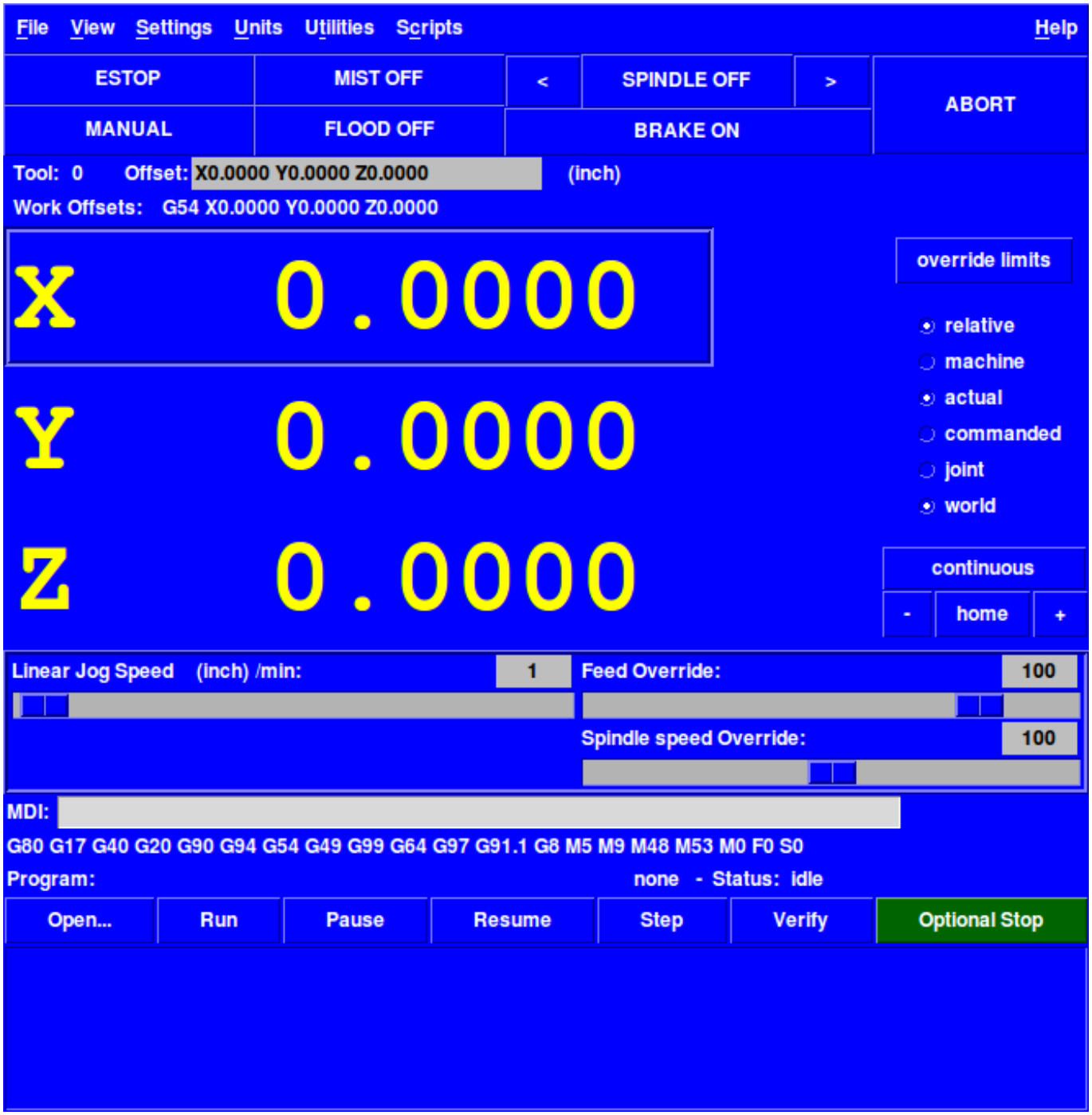



Abbildung 2.7: TkLinuxCNC graphical interface

**QtDragon**

QtDragon, a touch screen GUI based on QtVCP using the PyQt5 library. It comes in two versions *QtDragon* and *QtDragon_hd*. They are very similar in features but QtDragon_hd is made for larger monitors.

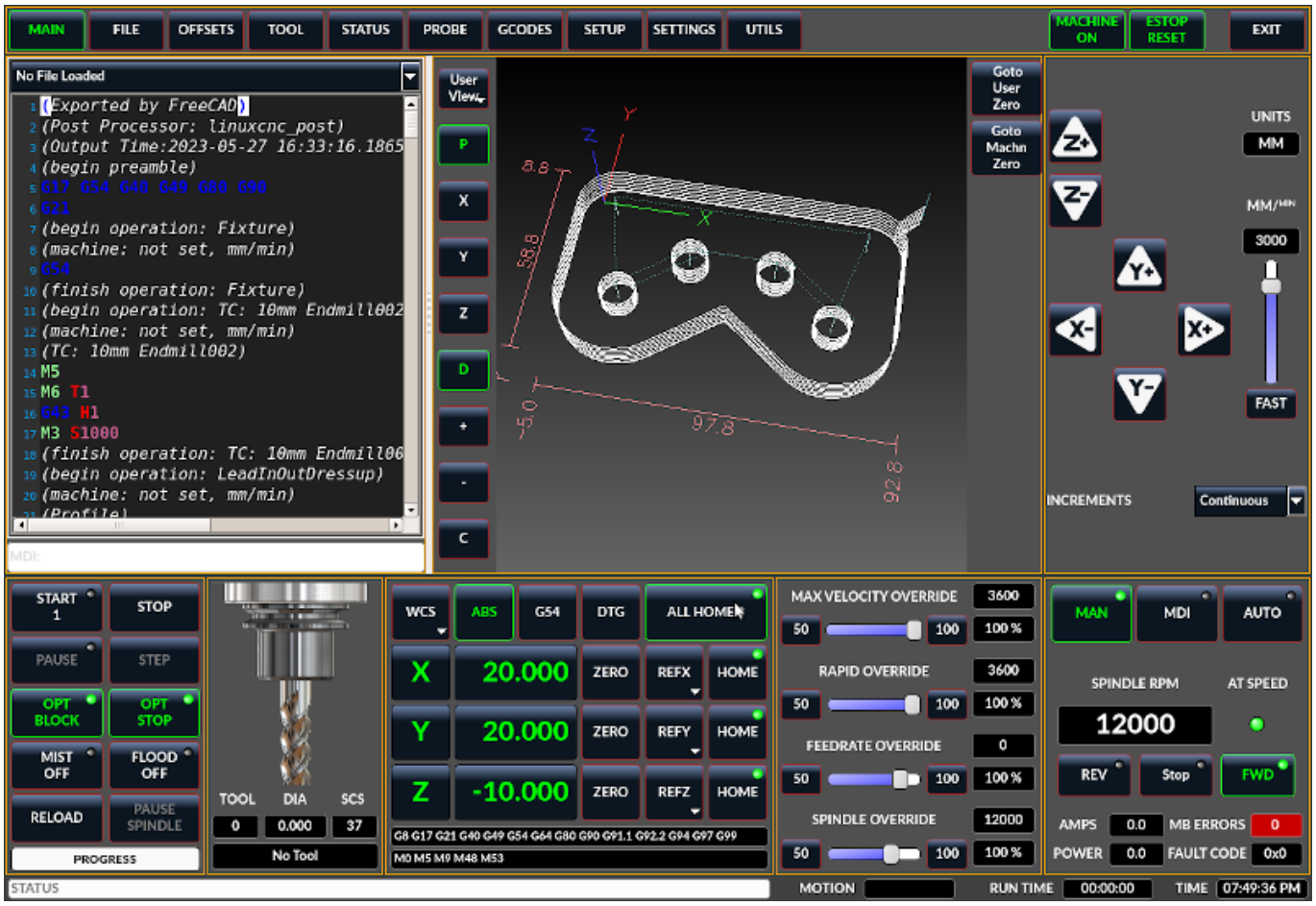



Abbildung 2.8: QtDragon, a touch screen GUI based on QtVCP

**QtPlasmaC**

QtPlasmaC, a touch screen plasma cutting GUI based on QtVCP using the PyQt5 library. It comes in three aspect ratios, 16:9, 4:3, and 9:16.

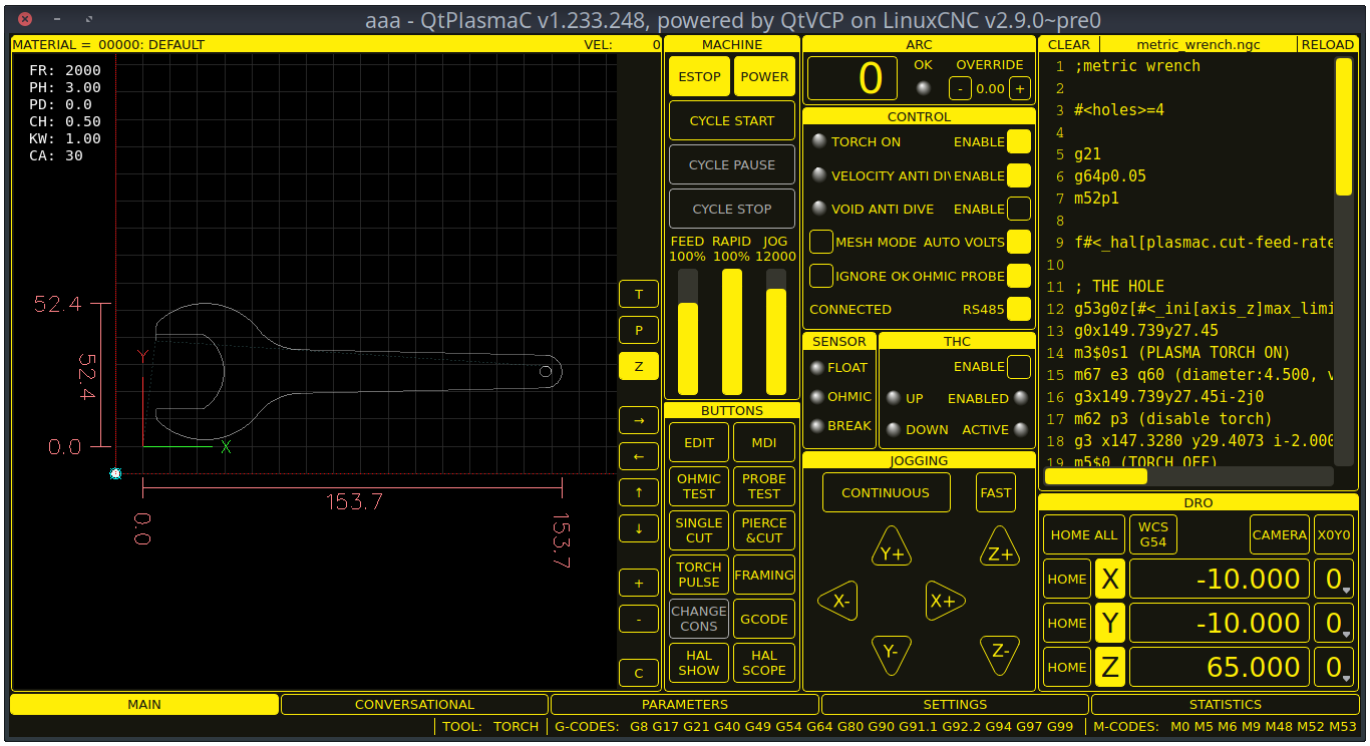



Abbildung 2.9: QtPlasmaC, a touch screen plasma cutting GUI based on QtVCP

### 2.2.4 Benutzerschnittstellen

These User interfaces are a way to interact with LinuxCNC outside of the graphical user interfaces.

**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.

### 2.2.5 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.

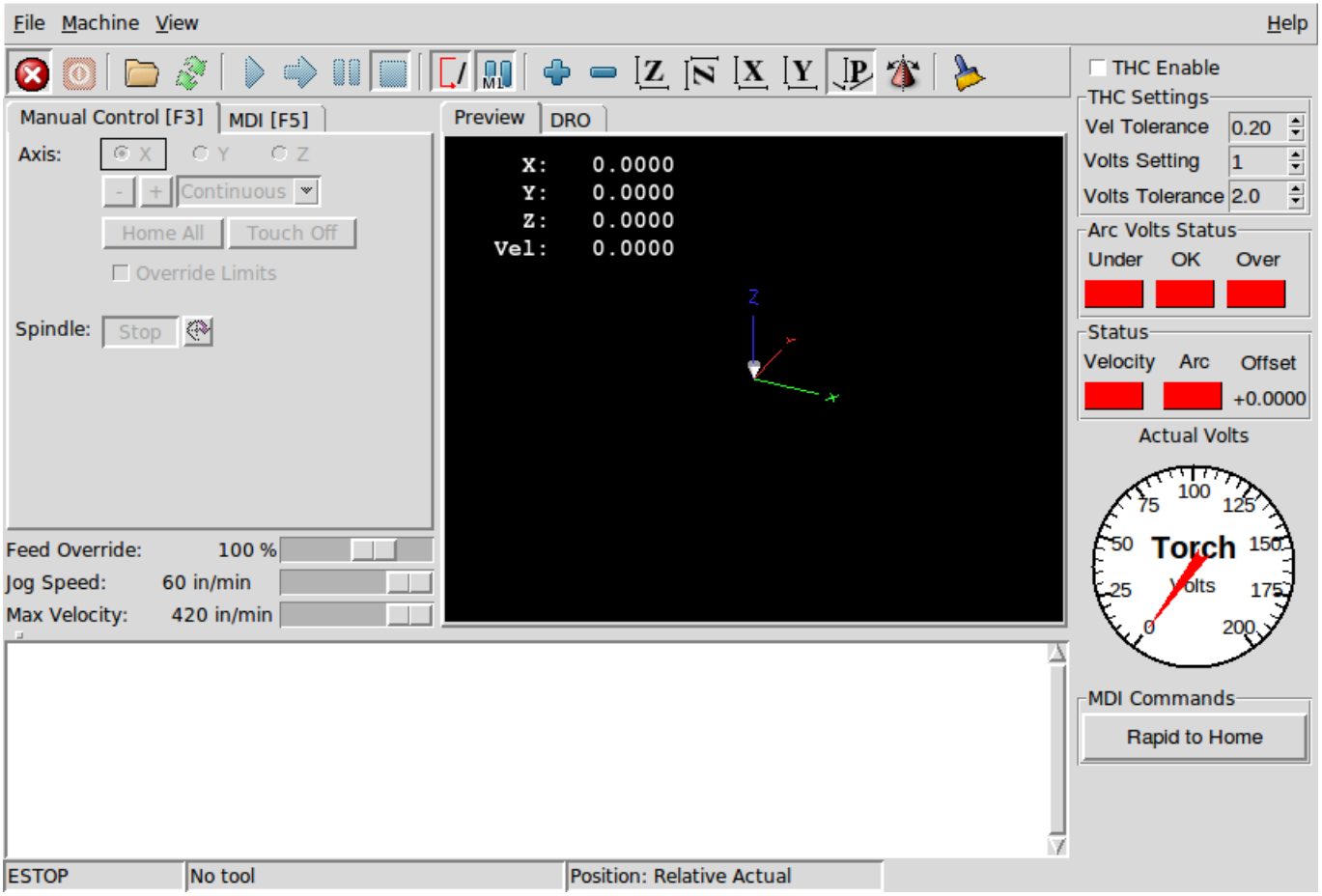



Abbildung 2.10: PyVCP Example Embedded Into AXIS GUI

**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:

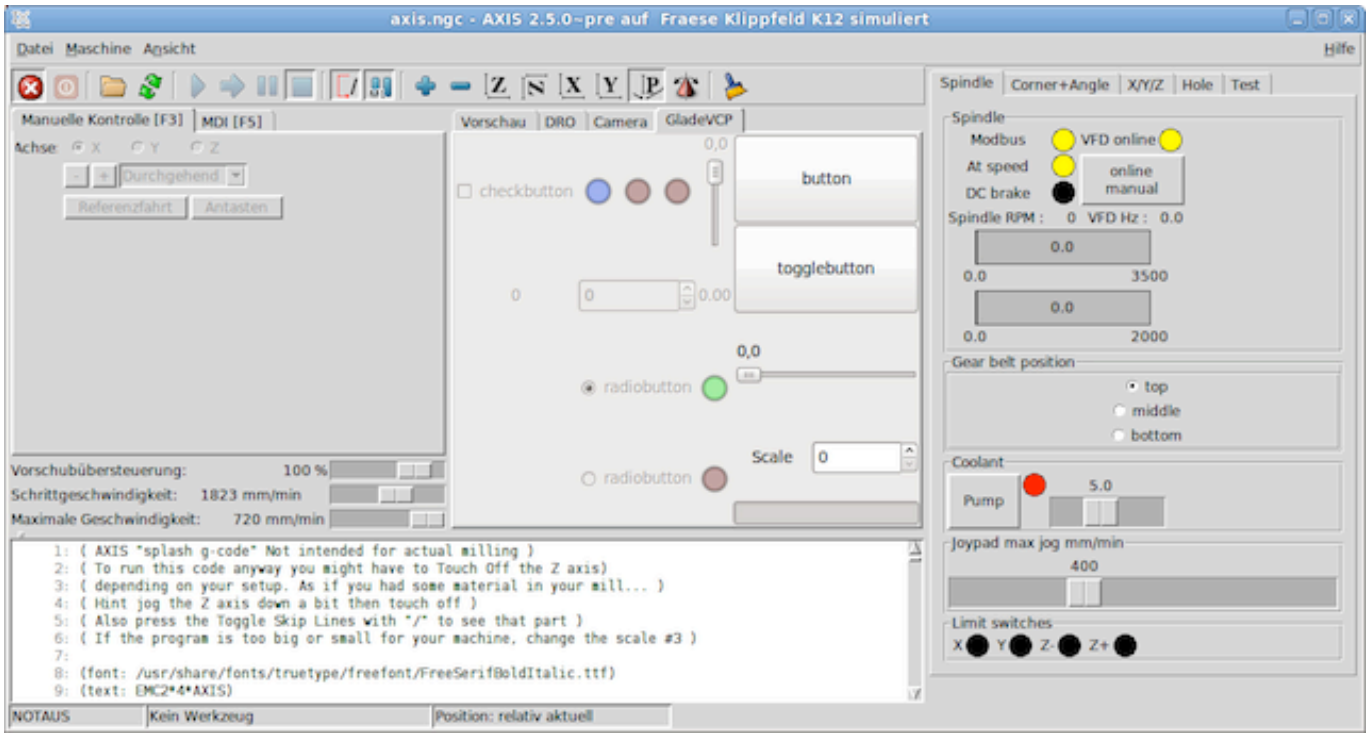



Abbildung 2.11: GladeVCP Example Embedded Into AXIS GUI

**QtVCP**

QtVCP, a PyQt5-based virtual control panel that can be added to most GUIs or run as a standalone panel. QtVCP 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 by extending with python code. QtVCP is also more flexible in how it may be configured to appear on the GUI with many special widgets:

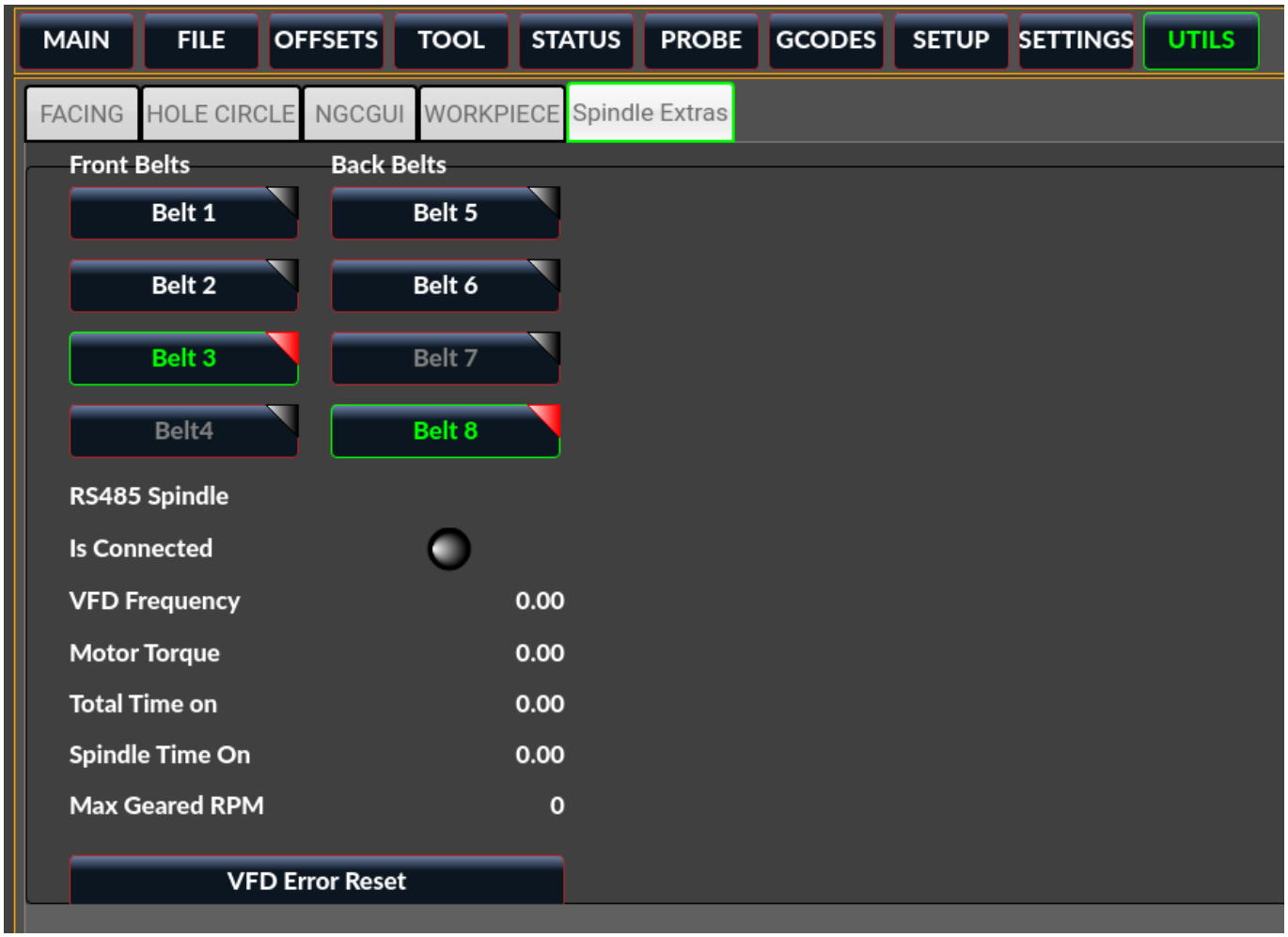



Abbildung 2.12: QtVCP Example Embedded Into QtDragon GUI

### 2.2.6 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.

### 2.2.7 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.

### 2.2.8 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

Trajectory planning, in general, is the means by which LinuxCNC follows the path specified by your G-code program, while still operating within the limits of your machinery.

A G-code program can never be fully obeyed. For example, imagine you specify as a single-line program the following move:

```
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*.

The basic acceleration and deceleration described above is not complex and there is no compromise to be made. In the INI file the specified machine constraints, such as maximum axis velocity and axis acceleration, must be obeyed by the trajectory planner.

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

#### 2.3.1.2 Path Following

A less straightforward problem is that of path following. When you program a corner in G-code, the trajectory planner can do several things, all of which are right in some cases:

- 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.

You can see that there is a trade off here: you can slow down to get better path following, or keep the speed up and have worse path following. Depending on the particular cut, the material, the tooling, etc., the programmer may want to compromise differently.

Rapid moves also obey the current trajectory control. With moves long enough to reach maximum velocity on a machine with low acceleration and no path tolerance specified, you can get a fairly round corner.

#### 2.3.1.3 Programming the Planner

The trajectory control commands are as follows:

**`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.

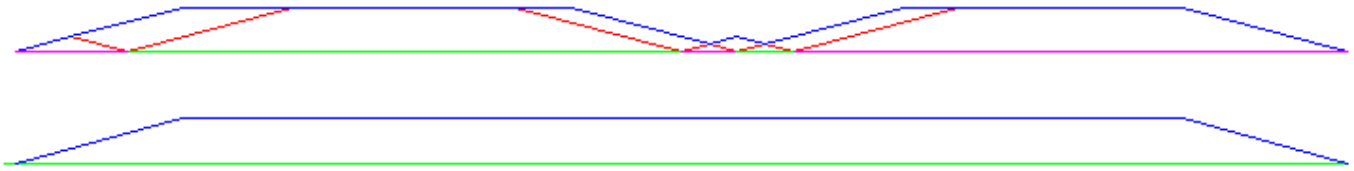


Abbildung 2.13: Naive CAM Detector

#### 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.

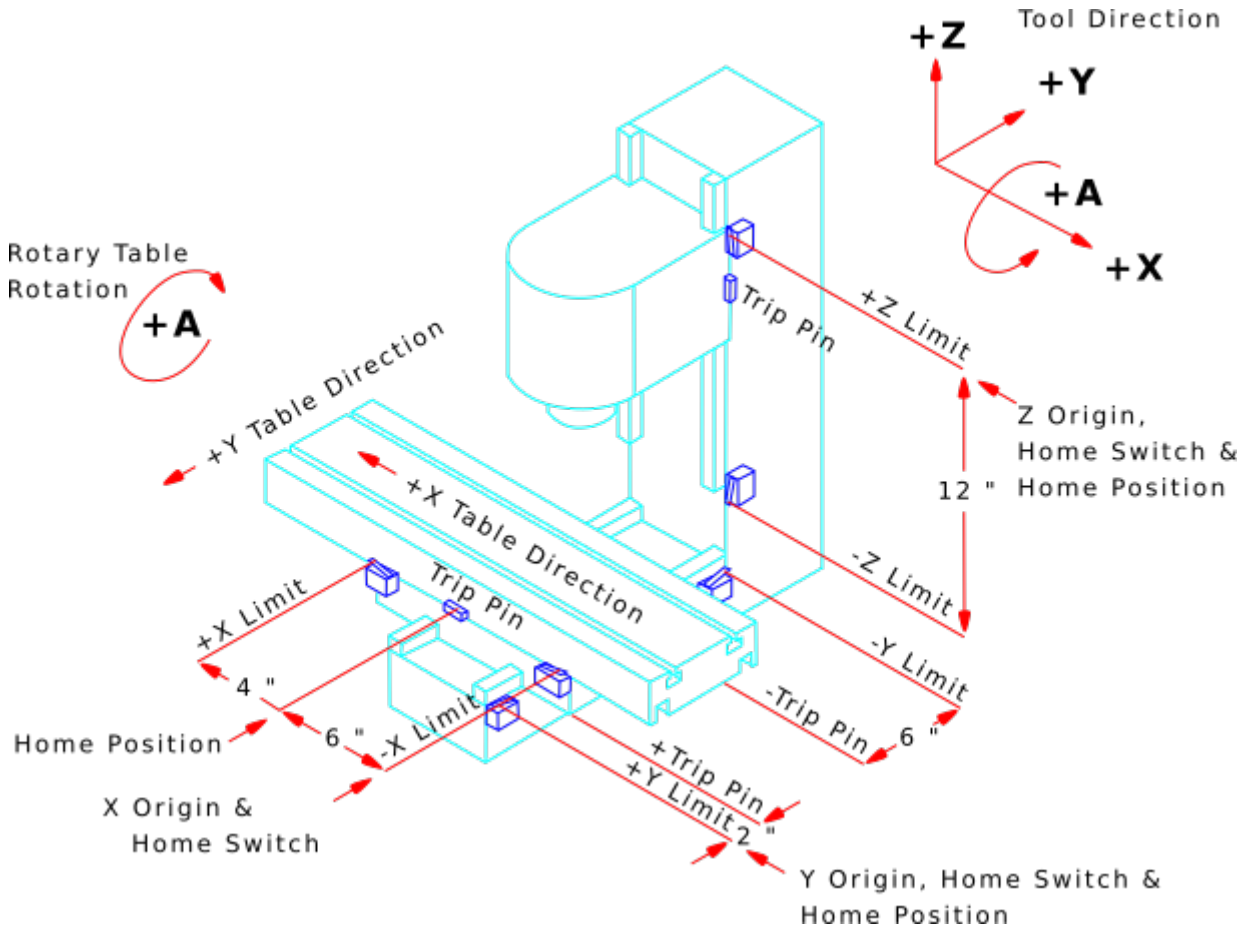



Abbildung 2.14: Typical Mill Configuration

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

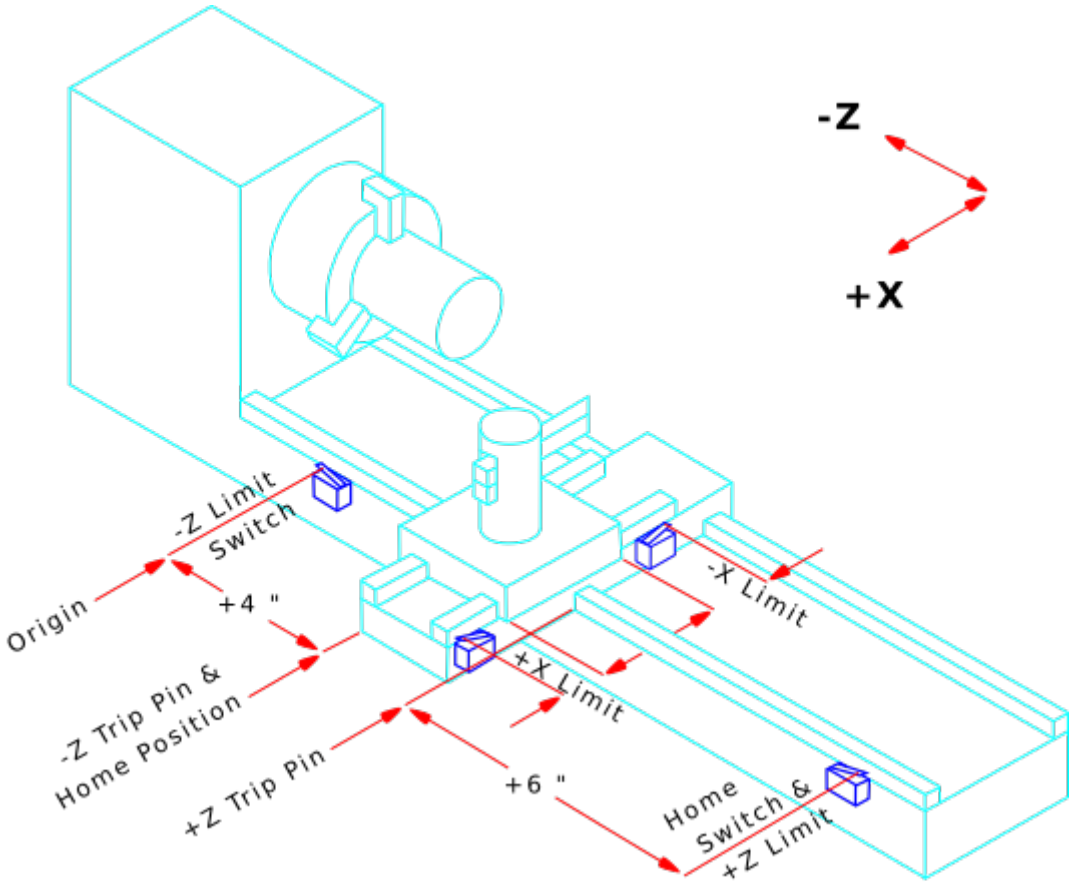



Abbildung 2.15: Typical Lathe Configuration

## 2.4 Starting LinuxCNC

### 2.4.1 Running LinuxCNC

LinuxCNC is started with the script file *linuxcnc*.

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

**linuxcnc script options**

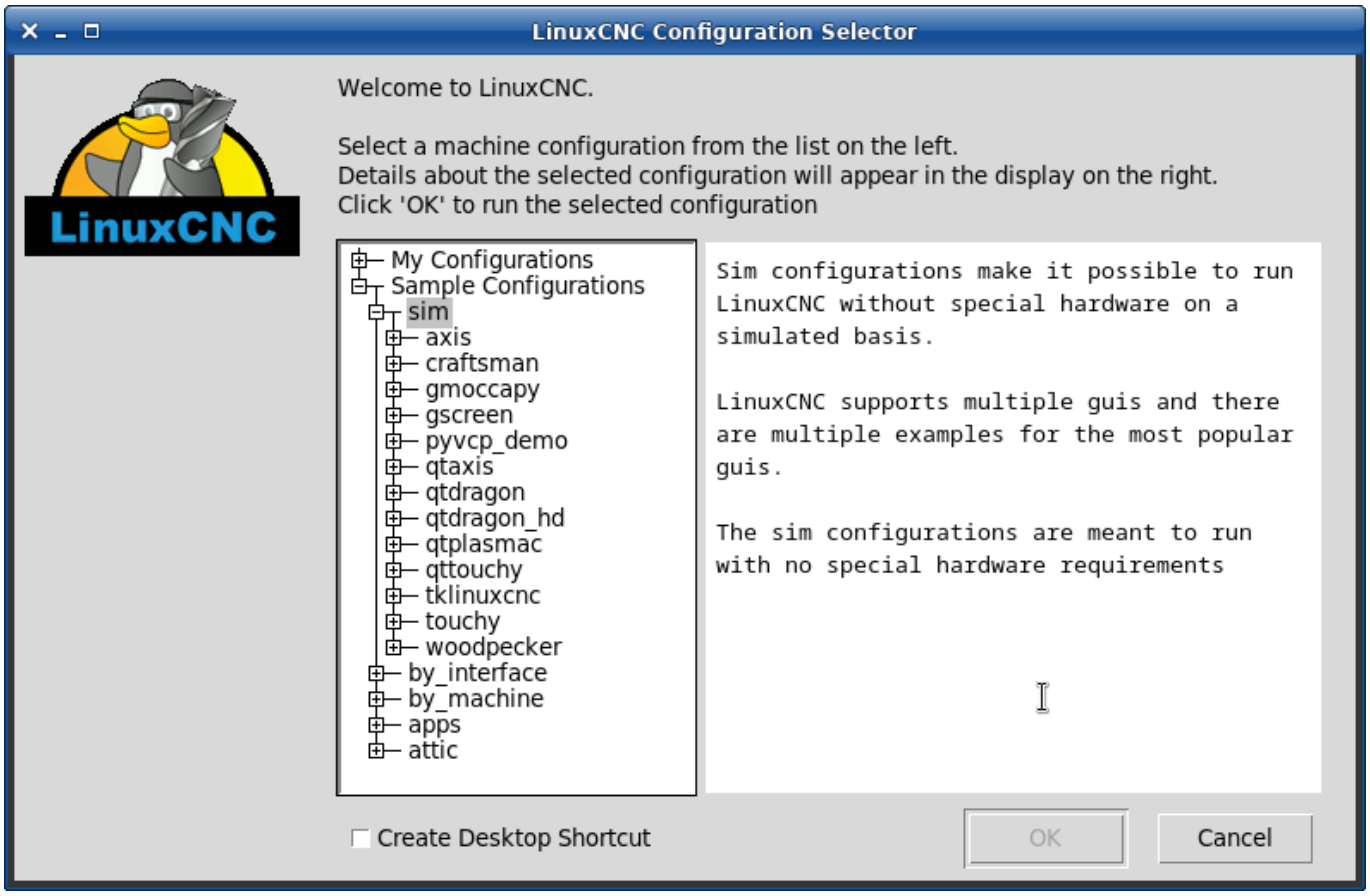
```
linuxcnc: Run LinuxCNC

Usage:
  $ linuxcnc -h
    This help

  $ linuxcnc [Options]
    Choose the configuration INI file graphically

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

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

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

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 Machine Overview

This section gives a brief description of how a CNC machine is viewed from the input and output ends of the Interpreter.

### 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 Axes

Any CNC machine has one or more Axes. Different types of CNC machines have different combinations. For instance, a *4-axis milling machine* may have XYZA or XYZB axes. A lathe typically has XZ axes. A foam-cutting machine may have XYUV axes. In LinuxCNC, the case of a XYYZ *gantry* machine with two motors for one axis is better handled by kinematics rather than by a second linear axis.

**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.

.Primary Linear Axes Die X-, Y- und Z-Achse erzeugen lineare Bewegungen in drei zueinander orthogonalen Richtungen.

.Secondary Linear Axes 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.

.Rotationsachsen 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

A CNC machine typically has a spindle which holds one cutting tool, probe, or the material in the case of a lathe. The spindle may or may not be controlled by the CNC software. LinuxCNC offers support for up to 8 spindles, which can be individually controlled and can run simultaneously at different speeds and in different directions.

#### 2.5.1.3 Kühlmittel

Flood coolant and mist coolant may each be turned on independently. The RS274/NGC language turns them off together see section M7 M8 M9.

#### 2.5.1.4 Feed and Speed Override

A CNC machine can have separate feed and speed override controls, which let the operator specify that the actual feed rate or spindle speed used in machining at some percentage of the programmed rate.

#### 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 Controlled Point

The controlled point is the point whose position and rate of motion are controlled. When the tool length offset is zero (the default value), this is a point on the spindle axis (often called the gauge point) that is some fixed distance beyond the end of the spindle, usually near the end of a tool holder that fits into the spindle. The location of the controlled point can be moved out along the spindle axis by specifying some positive amount for the tool length offset. This amount is normally the length of the cutting tool in use, so that the controlled point is at the end of the cutting tool. On a lathe, tool length offsets can be specified for X and Z axes, and the controlled point is either at the tool tip or slightly outside it (where the perpendicular, axis-aligned lines touched by the *front* and *side* of the tool intersect).

### 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 Feed Rate

The rate at which the controlled point moves is nominally a steady rate which may be set by the user. In the Interpreter, the feed rate is interpreted as follows (unless *inverse time feed* or *feed per revolution* modes are being used, in which case see section 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 Cooling

Flood or droplets cooling can be enabled separately. RS274/NGC language stops them together. See section about cooling control.

#### 2.5.2.7 Verweilen (engl. dwell)

A machining center may be commanded to dwell (i.e., keep all axes unmoving) for a specific amount of time. The most common use of dwell is to break and clear chips, so the spindle is usually turning during a dwell. Regardless of the Path Control Mode (see section Path Control) the machine will stop exactly at the end of the previous programmed move, as though it was in exact path mode.

#### 2.5.2.8 Einheiten

Units used for distances along the X, Y, and Z axes may be measured in millimeters or inches. Units for all other quantities involved in machine control cannot be changed. Different quantities use different specific units. Spindle speed is measured in revolutions per minute. The positions of rotational axes are measured in degrees. Feed rates are expressed in current length units per minute, or degrees per minute, or length units per spindle revolution, as described in section G93 G94 G95.

#### 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 Path Control Mode

The machining center may be put into any one of three path control modes:

**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 Feed and Speed Override Switches

The Interpreter will interpret RS274/NGC commands which enable *M48* or disable *M49* the feed and speed override switches. For certain moves, such as the traverse out of the end of a thread during a threading cycle, the switches are disabled automatically.

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

A tool table is required to use the Interpreter. The file tells which tools are in which tool changer slots and what the size and type of each tool is. The name of the tool table is defined in the INI file:

```
[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 the RS274/NGC language view, a machining center maintains an array of numerical parameters defined by a system definition (RS274NGC_MAX_PARAMETERS). Many of them have specific uses especially in defining coordinate systems. The number of numerical parameters can increase as development adds support for new parameters. The parameter array persists over time, even if the machining center is powered down. LinuxCNC uses a parameter file to ensure persistence and gives the Interpreter the responsibility for maintaining the file. The Interpreter reads the file when it starts up, and writes the file when it exits.

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 Lathe User Information

This chapter will provide information specific to lathes.

### 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
```

Lathe Mode in AXIS does not set your default plane to G18 (XZ). You must program that in the preamble of each G-code file or (better) add it to your INI file, like this:

```
[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 Lathe Tool Table

The ”Tool Table” is a text file that contains information about each tool. The file is located in the same directory as your configuration and is called ”tool.tbl” by default. The tools might be in a tool changer or just changed manually. The file can be edited with a text editor or be updated using G10 L1,L10,L11. There is also a built-in tool table editor in the AXIS display. The maximum number of entries in the tool table is 56. The maximum tool and pocket number is 99999.

Earlier versions of LinuxCNC had two different tool table formats for mills and lathes, but since the 2.4.x release, one tool table format is used for all machines. Just ignore the parts of the tool table that don’t pertain to your machine, or which you don’t need to use. For more information on the specifics of the tool table format, see the Tool Table Section.

### 2.6.3 Lathe Tool Orientation

The following figure shows the lathe tool orientations with the center line angle of each orientation and info on FRONTANGLE and BACKANGLE.

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

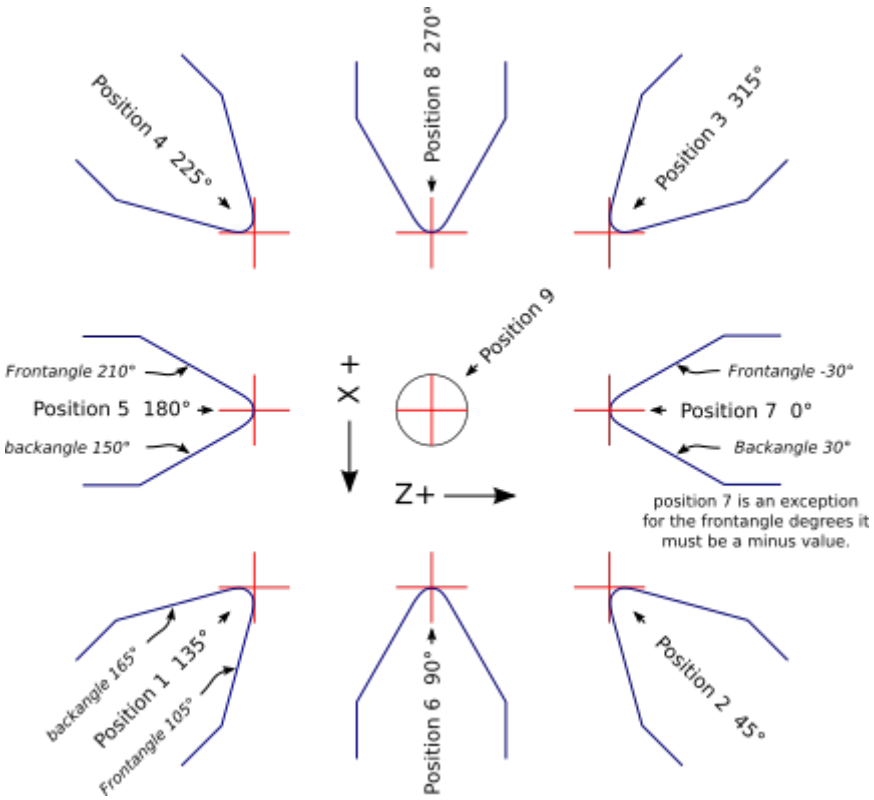
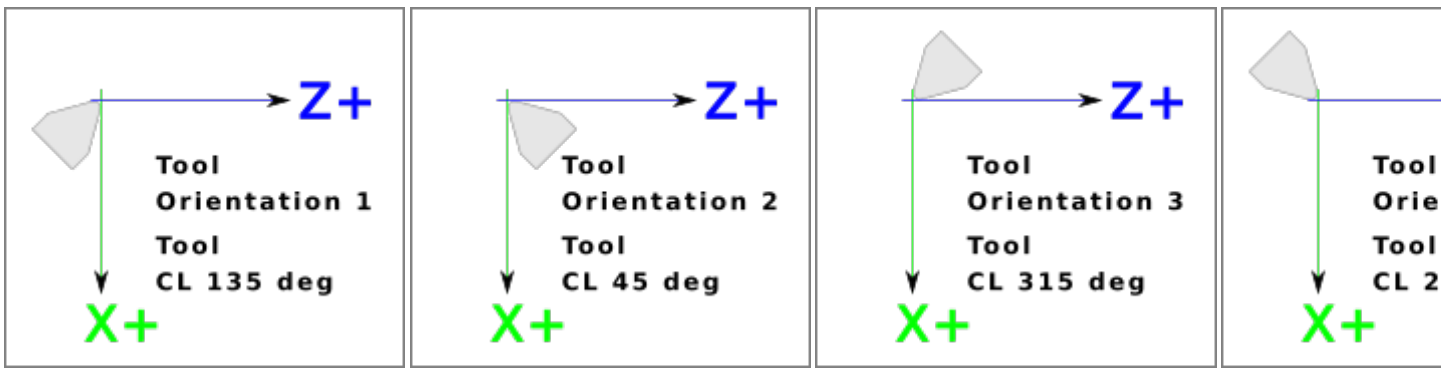


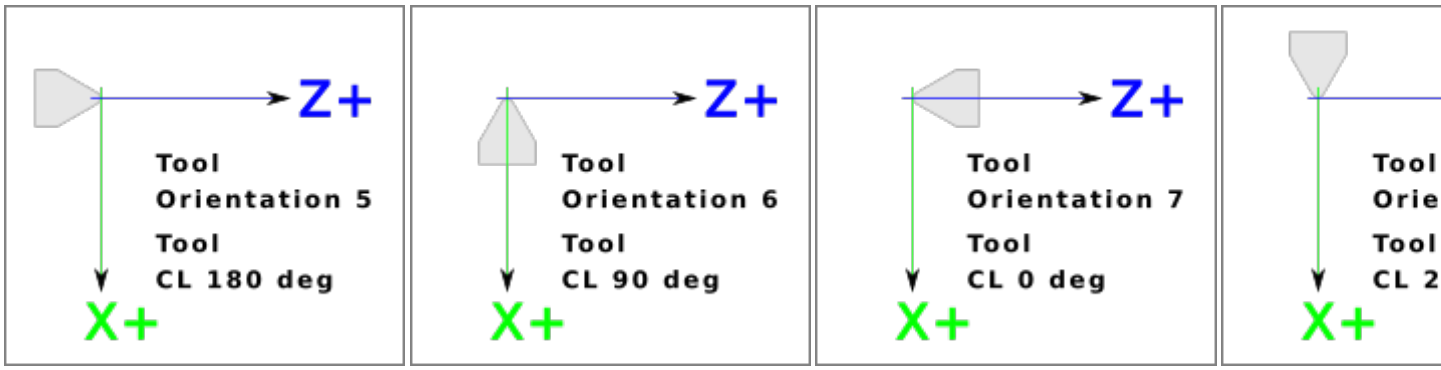

Abbildung 2.16: Ausrichtung des Drehwerkzeugs

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

**Tool Positions 1, 2, 3 & 4**




**Tool Positions 5, 6, 7 & 8**




## 2.6.4 Werkzeug Touch Off

When running in lathe mode in AXIS you can set the X and Z in the tool table using the Touch Off window. If you have a tool turret you normally have *Touch off to fixture* selected when setting up your turret. When setting the material Z zero you have *Touch off to material* selected. For more information on the G-codes used for tools see M6, Tn, and G43. For more information on tool touch off options in AXIS see 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.

One method is to take your normal turning tool and turn down some stock to a known diameter. Using the Tool Touch Off window enter the measured diameter (or radius if in radius mode) for that tool. Then using some layout fluid or a marker to coat the part bring each tool up till it just touches the dye and set its X offset to the diameter of the part used using the tool touch off. Make sure any tools in the corner quadrants have the nose radius set properly in the tool table so the control point is correct. Tool touch off automatically adds a G43 so the current tool is the current offset.

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

Spindle synchronized motion requires a quadrature encoder connected to the spindle with one index pulse per revolution. See the motion man page and the Spindle Control Example for more information.

**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** CSS or Constant Surface Speed uses the machine X origin modified by the tool X offset to compute the spindle speed in RPM. CSS will track changes in tool offsets. The X machine origin should be when the reference tool (the one with zero offset) is at the center of rotation. For more information see the G96 Section.

**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

Calculating arcs can be mind challenging enough without considering radius and diameter mode on lathes as well as machine coordinate system orientation. The following applies to center format arcs. On a lathe you should include G18 in your preamble as the default is G17 even if you're in lathe mode, in the user interface AXIS. Arcs in G18 XZ plane use I (X axis) and K (Z axis) offsets.

### 2.6.6.1 Bögen und Drehmaschinendesign

The typical lathe has the spindle on the left of the operator and the tools on the operator side of the spindle center line. This is typically set up with the imaginary Y axis (+) pointing at the floor.

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.

Some lathes with tools on the back side have the imaginary Y axis (+) pointing up.

G2/G3 Arc directions are based on the axis they rotate around. In the case of lathes, it is the imaginary Y axis. If the Y axis (+) points toward the floor, you have to look up for the arc to appear to go in the correct direction. So looking from above you reverse the G2/G3 for the arc to appear to go in the correct direction.

### 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.

When calculating arcs in diameter mode X is diameter and the X offset (I) is radius even if you're in G7 diameter mode.

## 2.6.7 Werkzeugpfad

### 2.6.7.1 Kontrollpunkt

The control point for the tool follows the programmed path. The control point is the intersection of a line parallel to the X and Z axis and tangent to the tool tip diameter, as defined when you touch off the X and Z axes for that tool. When turning or facing straight sided parts the cutting path and the tool edge follow the same path. When turning radius and angles the edge of the tool tip will not follow the programmed path unless cutter comp is in effect. In the following figures you can see how the control point does not follow the tool edge as you might assume.

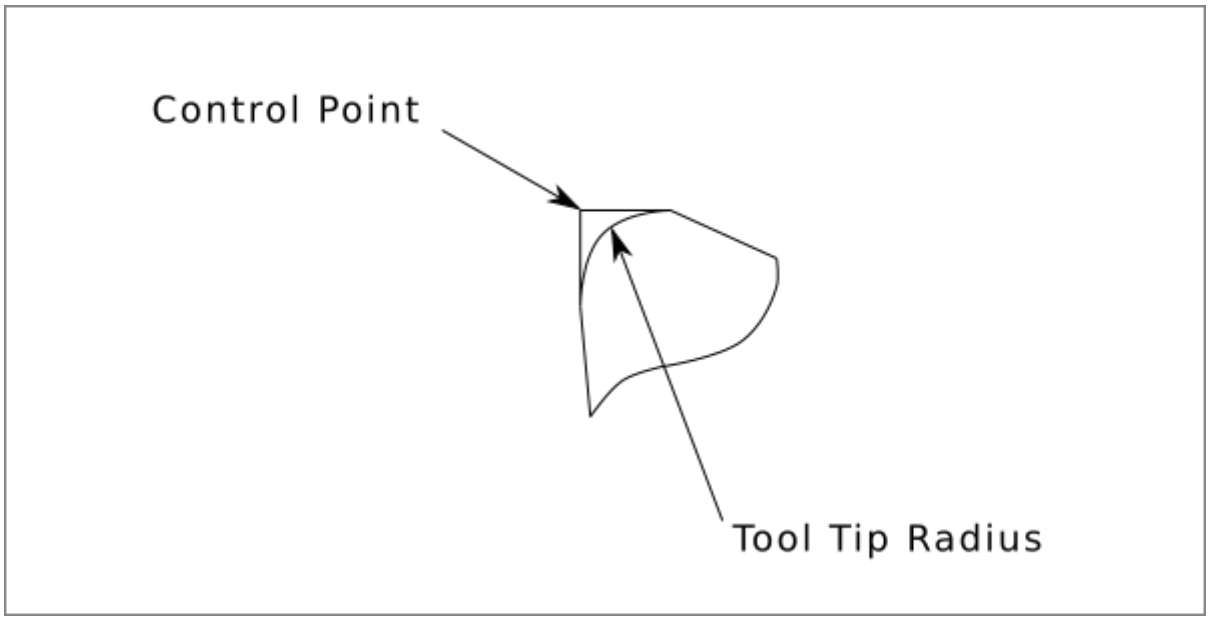



Abbildung 2.17: Kontrollpunkt

#### 2.6.7.2 Schneidwinkel ohne Fräser Compensation

Now imagine we program a ramp without cutter comp. The programmed path is shown in the following figure. As you can see in the figure the programmed path and the desired cut path are one and the same as long as we are moving in an X or Z direction only.

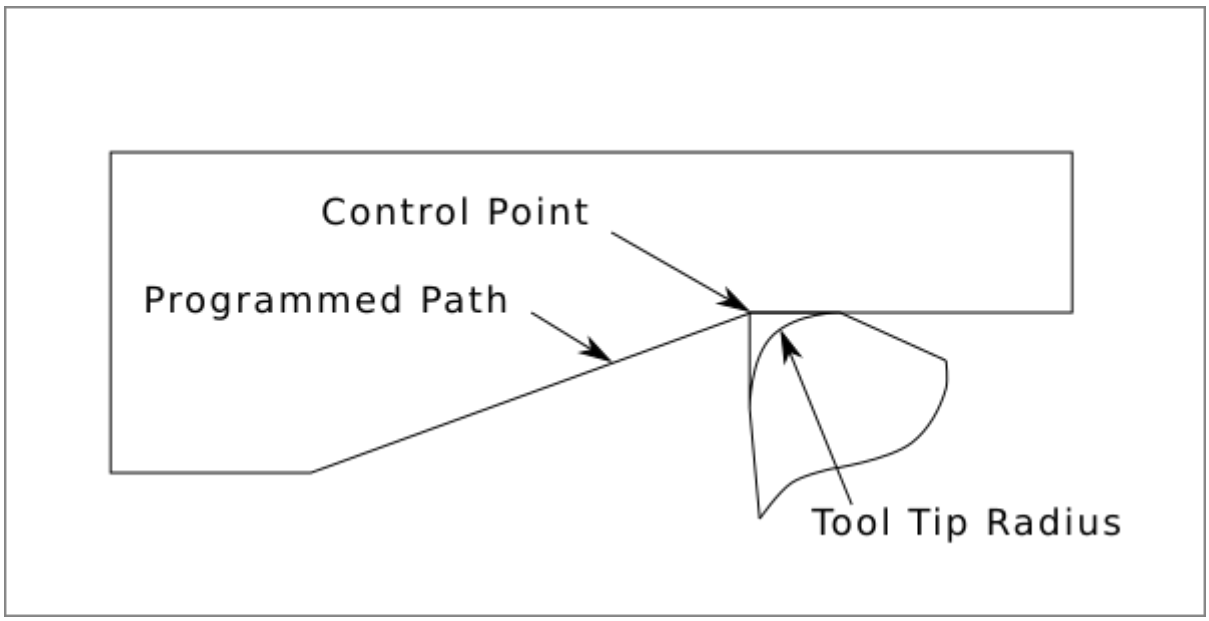



Abbildung 2.18: Rampe Eingang

Now as the control point progresses along the programmed path the actual cutter edge does not follow the programmed path as shown in the following figure. There are two ways to solve this, cutter comp and adjusting your programmed path to compensate for tip radius.

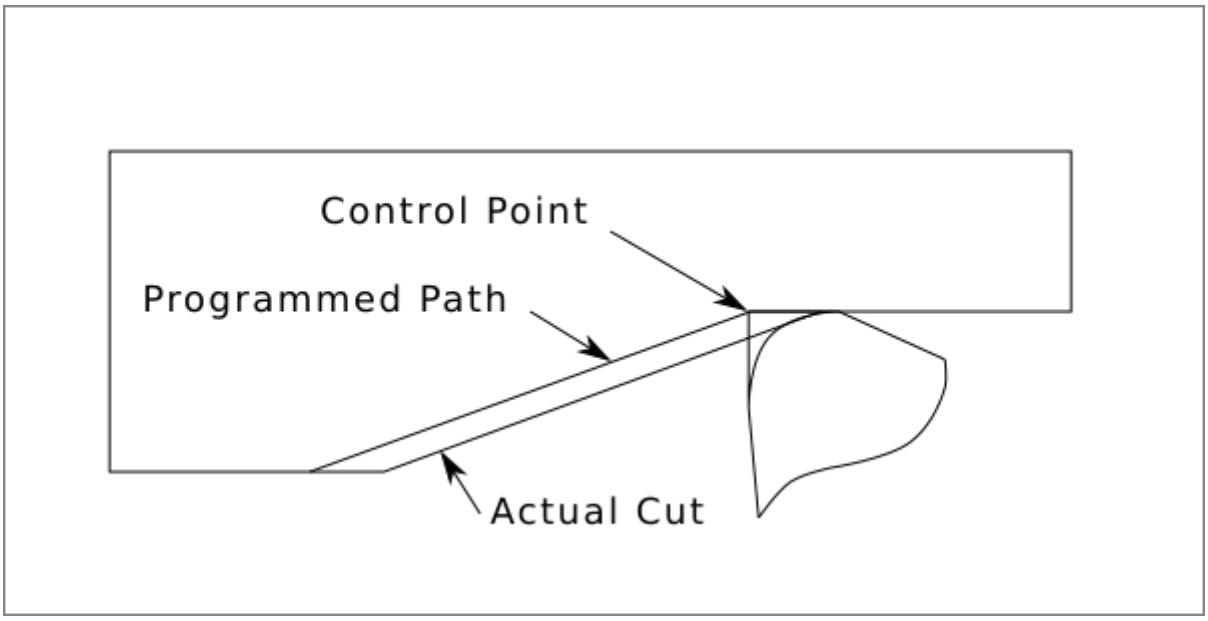



Abbildung 2.19: Rampenpfad

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 this example we will examine what happens during a radius cut without cutter comp. In the next figure you see the tool turning the OD of the part. The control point of the tool is following the programmed path and the tool is touching the OD of the part.

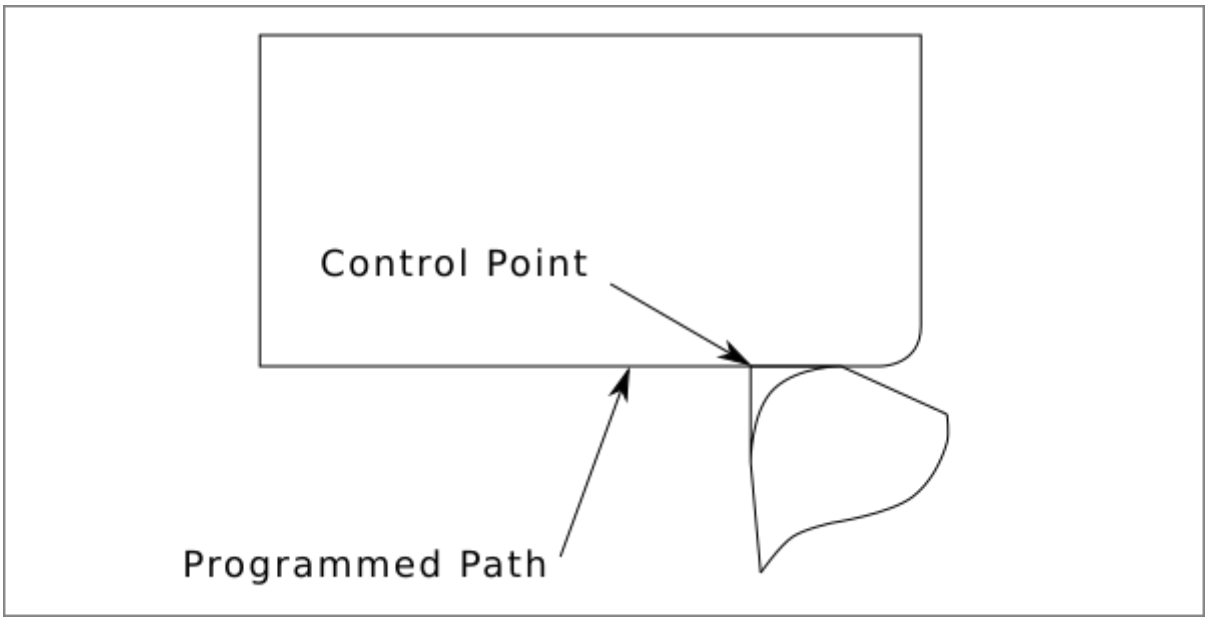



Abbildung 2.20: Turning Cut

In this next figure you can see as the tool approaches the end of the part the control point still follows the path but the tool tip has left the part and is cutting air. You can also see that even though a radius has been programmed the part will actually end up with a square corner.

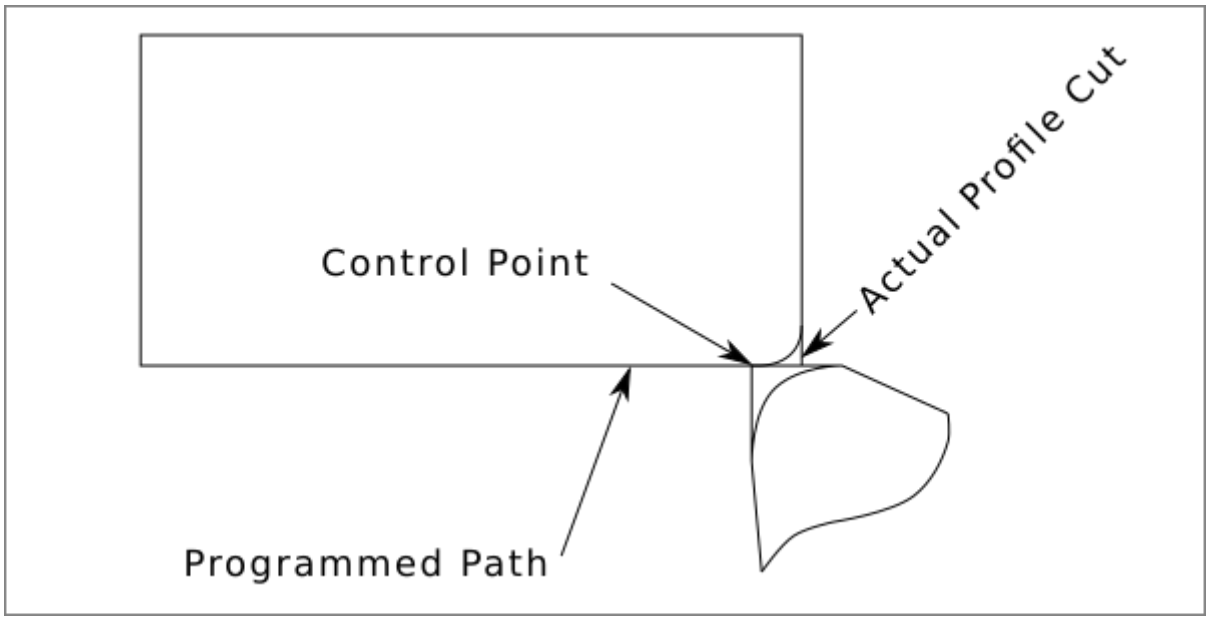



Abbildung 2.21: Radiusschnitt

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

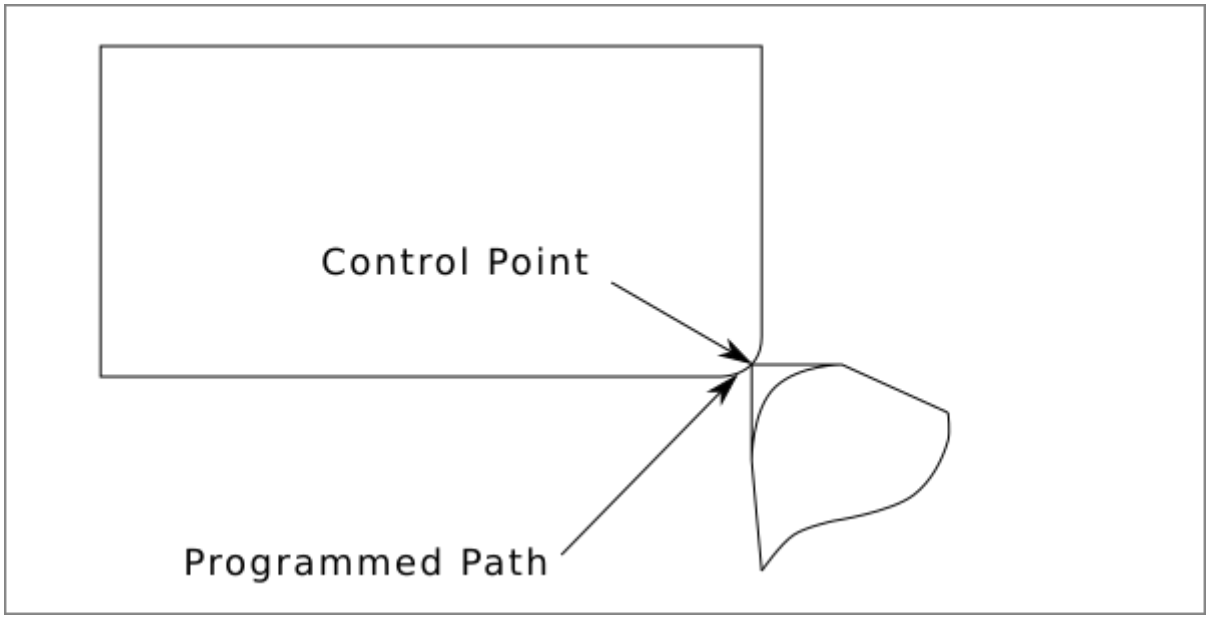



Abbildung 2.22: Radiusschnitt

In the final figure we can see the tool tip will finish cutting the face but leave a square corner instead of a nice radius. Notice also that if you program the cut to end at the center of the part a small amount

of material will be left from the radius of the tool. To finish a face cut to the center of a part you have to program the tool to go past center at least the nose radius of the tool.

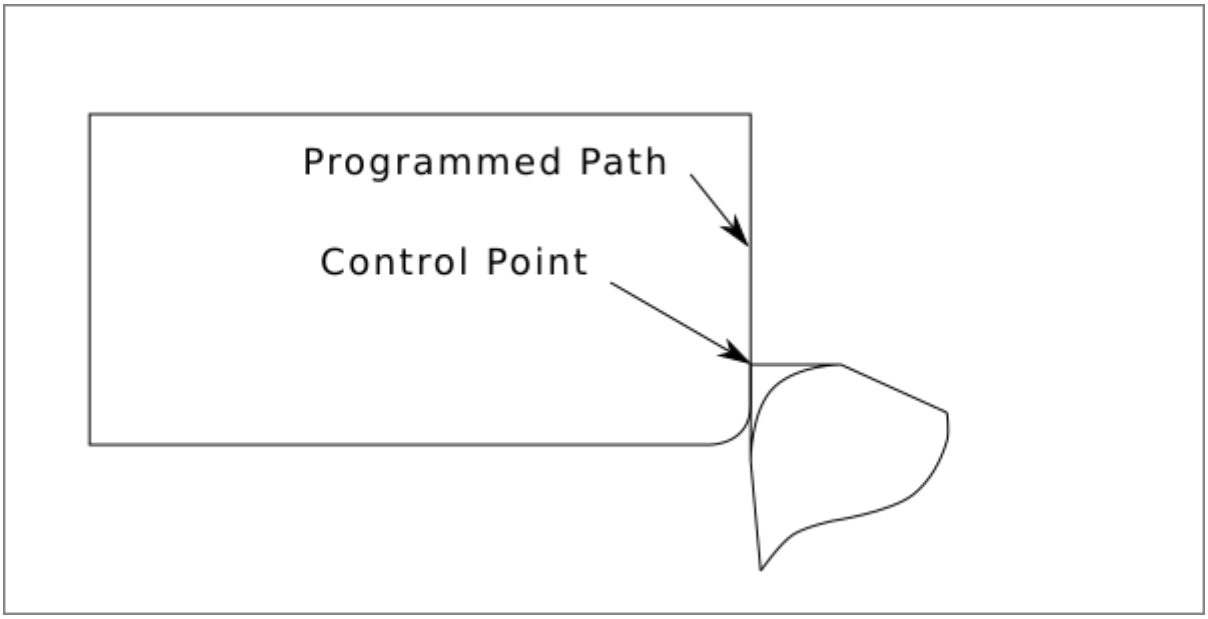



Abbildung 2.23: Face Cut

#### 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 Plasma Cutting Primer for LinuxCNC Users

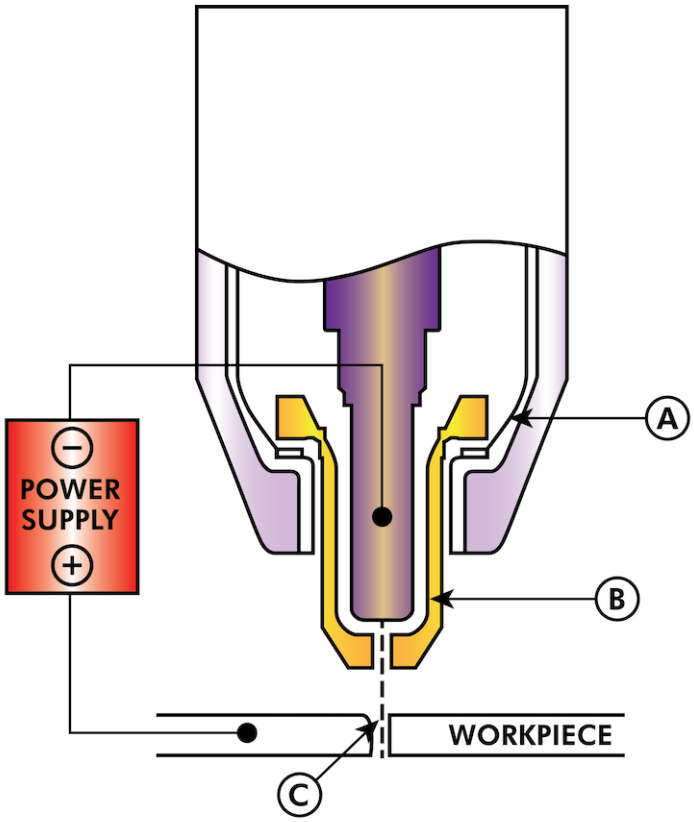
### 2.7.1 Was ist Plasma?

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 a Plasma Cutting Torch a cool gas enters Zone B, where a pilot arc between the electrode and the torch tip heats and ionises the gas. The main cutting arc then transfers to the workpiece through the column of plasma gas in Zone C. By forcing the plasma gas and electric arc through a small orifice, the torch delivers a high concentration of heat to a small area. The stiff, constricted plasma arc is shown in Zone C. Direct current (DC) straight polarity is used for plasma cutting, as shown in the illustration. Zone A channels a secondary gas that cools the torch. This gas also assists the high velocity plasma gas in blowing the molten metal out of the cut allowing for a fast, slag - free cut.




TYPICAL TORCH HEAD DETAIL

## 2.7.2 Bogen-Initialisierung

There are two main methods for arc initialisation for plasma cutters that are designed for CNC operation. Whilst other methods are used on some machines (such as scratch start where physical contact with the material is required), they are unsuited for CNC applications..

### 2.7.2.1 Hochfrequenzstart

This start type is widely employed, and has been around the longest. Although it is older technology, it works well, and starts quickly. But, because of the high frequency high voltage power that is required generated to ionise the air, it has some drawbacks. It often interferes with surrounding electronic circuitry, and can even damage components. Also a special circuit is needed to create a Pilot arc.

Inexpensive models will not have a pilot arc, and require touching the consumable to the work to start. Employing a HF circuit also can increase maintenance issues, as there are usually adjustable points that must be cleaned and readjusted from time to time.

#### 2.7.2.2 Blowback Start

This start type uses air pressure supplied to the cutter to force a small piston or cartridge inside the torch head back to create a small spark between the inside surface of the consumable, ionising the air, and creating a small plasma flame. This also creates a ”pilot arc” that provides a plasma flame that stays on, whether in contact with the metal or not. This is a very good start type that is now used by several manufacturers. It’s advantage is that it requires somewhat less circuitry, is a fairly reliable and generates far less electrical noise.

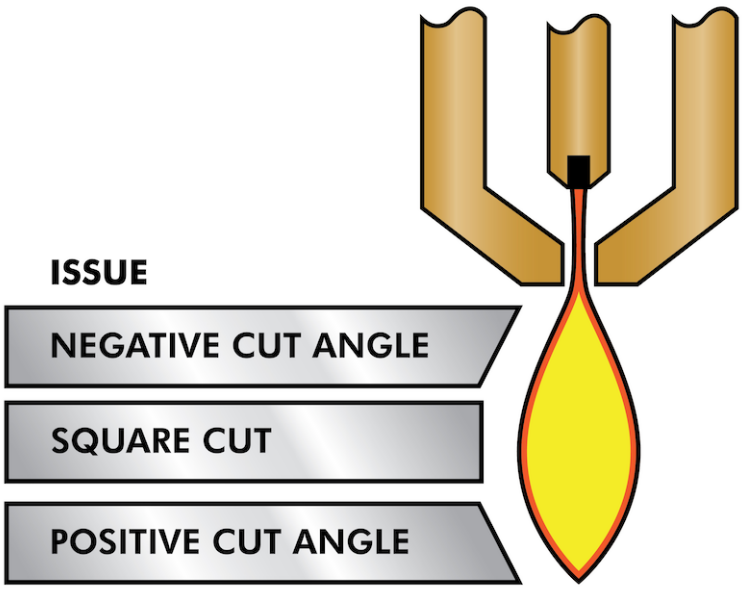
For entry level air plasma CNC systems, the blowback style is much preferred to minimise electrical interference with electronics and standard PCs, but the High frequency start still rules supreme in larger machines from 200 A and up. These require industrial level PCs and electronics, and even commercial manufacturers have had issues with faults because they have failed to account for electrical noise in their designs.

### 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.

A plasma Arc is oval in shape and the cutting height needs to be controlled to minimise bevelled edges. If the torch is too high or too low then the edges can become excessively bevelled. It is also critical that the torch is held perpendicular to the surface.

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



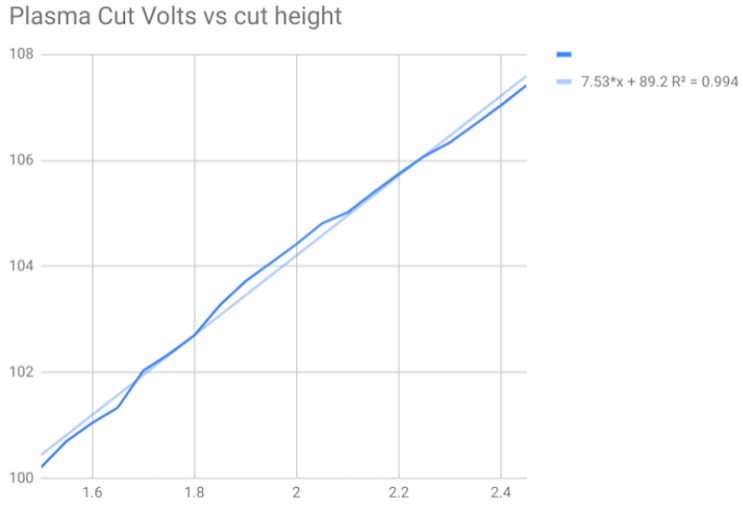
- **Negativer Schnittwinkel:** Brenner zu niedrig, Abstand zwischen Brenner und Werkstück vergrößern.
- **Positiver Schnittwinkel:** Brenner zu hoch, Abstand zwischen Brenner und Werkstück verringern.

---

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

---

The ability to precisely control the cutting height in such a hostile and ever changing environment is a very difficult challenge. Fortunately there is a very linear relationship between Torch height (Arc length) and arc voltage as this graph shows.




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

There are a plethora of plasma machines available on the market today and not all of them are suited for CNC use. CNC Plasma cutting is a complex operation and it is recommended that integrators choose a suitable plasma machine. Failure to do this is likely to cause hours and hours of fruitless trouble shooting trying to work around the lack of what many would consider to be mandatory features.

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 recent times, another class of machine which includes some of these features has become available at around USD $550. One example is the Herocut55i available on Amazon but there is yet no feedback from users. This Machine features a blowback torch, ArcOK output, torch start contacts and raw arc voltage.

### 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 of Hypertherm is on record saying that the best THC controllers were fully integrated into the CNC controller itself. Of course he was referring to high end systems manufactured by Hypertherm, Esab, Thermal Dynamics and others such as Advanced Robotic Technology in Australia, little dreaming that open source could produce systems using this approach that rival high end systems.

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

Plasma machines that have a CNC interface contain a set of dry contacts (eg a relay) that close when a valid arc is established and each side of these contacts are bought out onto pins on the CNC interface. A plasma table builder should connect one side of these pins to field power and the other to an input pin. This then allows the CNC controller to know when a valid arc is established and also when an arc is lost unexpectedly. There is a potential trap here when the input is a high impedance circuit such as a Mesa card. If the dry contacts are a simple relay, there is a high probability that the current passing through the relay is less than the minimum current specification. Under these conditions, the relay contacts can suffer from a buildup of oxide which over time can result in intermittent contact operation. To prevent this from happening, a pull down resistor should be installed on the controller input pin. Care should be taken to ensure that this resistor is selected to ensure the minimum current passes through the relay and is of sufficient wattage to handle the power in the circuit. Finally, the resistor should be mounted in such a way that the generated heat does not damage anything whilst in operation.

If you have an ArcOK signal, it is recommended it is used over and above any synthesised signal to eliminate potential build issues. A synthesised signal available from an external THC or QtPlasmaC's Mode 0 can't fully replace the ArcOK circuitry in a plasma inverter. Some build issues have been observed where misconfiguration or incompatibility with the plasma inverter has occurred from a synthesised ArcOK signal. By and large however, a correctly configured synthesised ArcOK signal is fine.

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, e.g. 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

Because the cutting height is such a critical system parameter and the material surface is inherently uneven, a Z axis mechanism needs a method to sense the material surface. There are three methods this can be achieved:

1. Current sensing to detect increased motor torque,
2. a "float" switch and an electrical or
3. an "ohmic" sensing circuit that is closed when the torch shield contacts the material.

Current sensing is not a viable technique for DIY tables but float switches and ohmic sensing are discussed below:

#### 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.

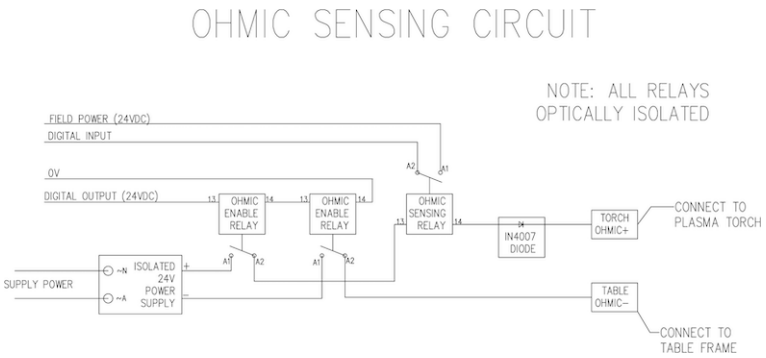
Unabhängig von der verwendeten Sondierungsmethode wird dringend empfohlen, einen gleitenden Schalter einzubauen, damit ein Ausweich- oder Sekundärsignal vorhanden ist, um eine Beschädigung des Brenners bei einem Absturz zu vermeiden.

#### 2.7.7.2 Ohmsche Erfassung

Ohmic sensing relies on contact between the torch and the material acting as a switch to activate an electrical signal that is sensed by the CNC controller. Provided the material is clean, this can be a much more accurate method of sensing the material than a float switch which can cause deflection of the material surface. This ohmic sensing circuit is operating in an extremely hostile environment so a number of failsafes need to be implemented to ensure safety of both the CNC electronics and the operator. In plasma cutting, the earth clamp attached to the material is positive and the torch is negative. It is recommended that:

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

A more sophisticated method of material sensing that eliminates the relays and diodes is to use another THCAD-5 to monitor the material sensing circuit voltage from an isolated power supply. The advantage this has is the THCAD is designed for the hostile plasma electrical environment and totally and safely isolates the logic side from the high voltage side.

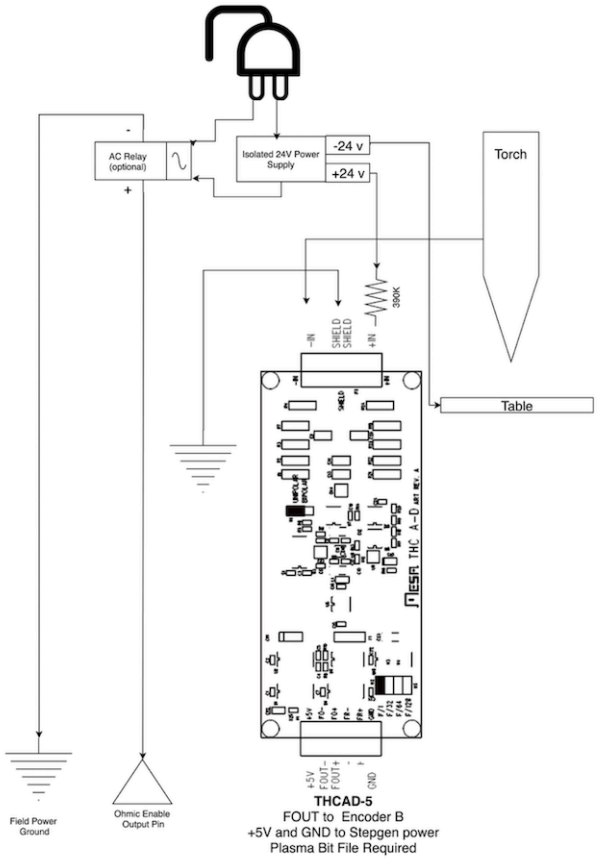
Um diese Methode zu implementieren, ist ein zweiter Encodereingang erforderlich.

If using a mesa card, different firmware is available to provide 2 additional Encoder A inputs on the Encoder B and Encoder Index pins. This firmware is available for download for the 7I76E and 7I96 boards from the Mesa web site on the product pages.

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

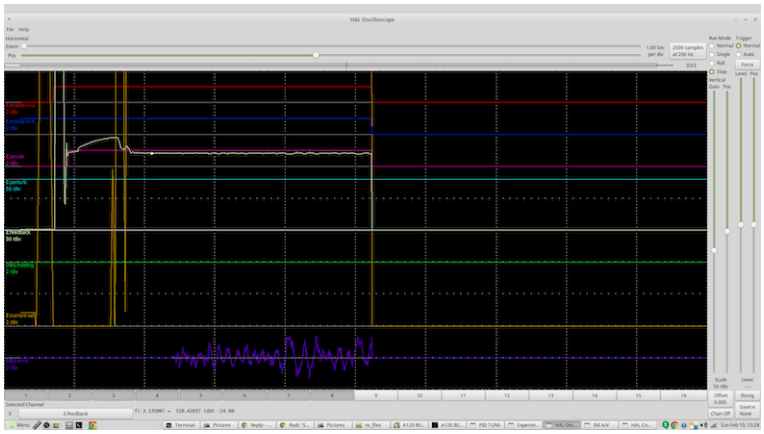
The following HAL code can be pasted into your QtPlasmaC's custom.hal to enable Ohmic sensing on Encoder 2 of a 7I76E. Install the correct bit file and connect the THCAD to IDX+ and IDX-. Be sure to change the calibration settings to agree with your THCAD-5.

```
# --- 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

When an arc is established, arc voltage peaks significantly and then settles back to a stable voltage at cut height. As shown by the green line in the image below.



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)

It is recommended that a mechanism is provided to allow the torch to “break away” or fall off in the case of impact with the material or a cut part that has tipped up. A sensor should be installed to allow the CNC controller to detect if this has occurred and pause the running program. Usually a break away is implemented using magnets to secure the torch to the Z axis stage.

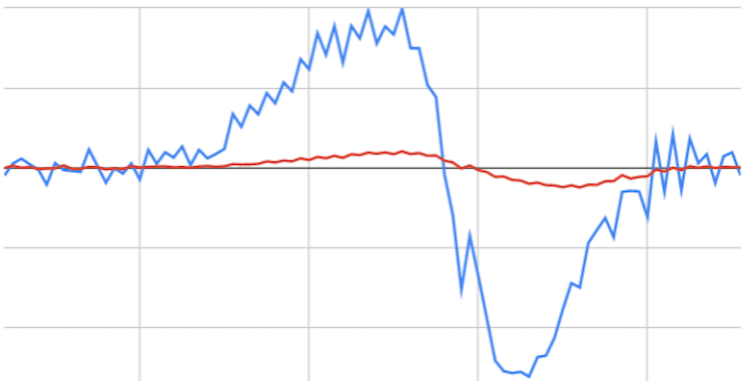
### 2.7.11 Corner Lock / Velocity Anti-Dive

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).

A feature newly added to LinuxCNC 2.9 that is useful for plasma cutting are the state tags. This adds a “tag” that is available to motion containing the current feed and speed rates for all active motion commands.

### 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)

- Triggers a relay to close the torch on switch in the inverter.
- Connect the torch on terminals on the inverter to the relay output terminals.
- Connect one side of the coil to the output pin.
- 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.
- If a Solid State Relay is used, polarity may need to be observed on the outputs.
- Unter bestimmten Umständen kann das integrierte Spindelrelais auf einer Mesa-Karte anstelle eines externen Relais verwendet werden.
- Usually connected to `spindle.0.on`.

**Warnung**
It is strongly recommended that the torch cannot be enabled while this pin is false otherwise the torch will not be extinguished when estop is pressed.

#### 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.
- Connect proximity sensor output to chosen input pin. If mechanical switches are used. Connect one side of the switch to field power and the other side of the switch to input.
- 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.

As can be seen, plasma tables are pin intensive and we have already consumed about 15 inputs before the normal estops are added. Others have other views but it is the writer's opinion that the Mesa 7I76E is preferred over the cheaper 7I96 to allow for MPG's, scale and axis selection switch and other features you may wish to add over time. If your table uses servos, there are a number of alternatives. Whilst there are other suppliers, designing your machine around the Mesa ecosystem will simplify use of their THCAD board to read arc voltage.

#### 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:

**Select Material Settings in QtPlasmaC and Use the Feedrate for that 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).

---

#### 2.7.15.1 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)
```

**Reduce Cutting Speeds: (e.g., for hole cutting)**

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

**Fräserkompensation:**

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

**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 let's consider the Mesa 7I76E and the THCAD-10 for a moment with the following hypothetical calibration data:

- Full scale  928 kHz (928 kHz/32 = 29 kHz)
- 0 V  121.6 kHz (121.6 kHz/32 = 3.8 kHz)

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

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

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

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

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

Voltage = (frequency [Hz] - 3800 Hz) / (2520 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. It is 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 non-realtime 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 non-realtime 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 (e.g., 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 cutter 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 Stepper Configuration Wizard

### 3.1.1 Einführung

LinuxCNC is capable of controlling a wide range of machinery using many different hardware interfaces.

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

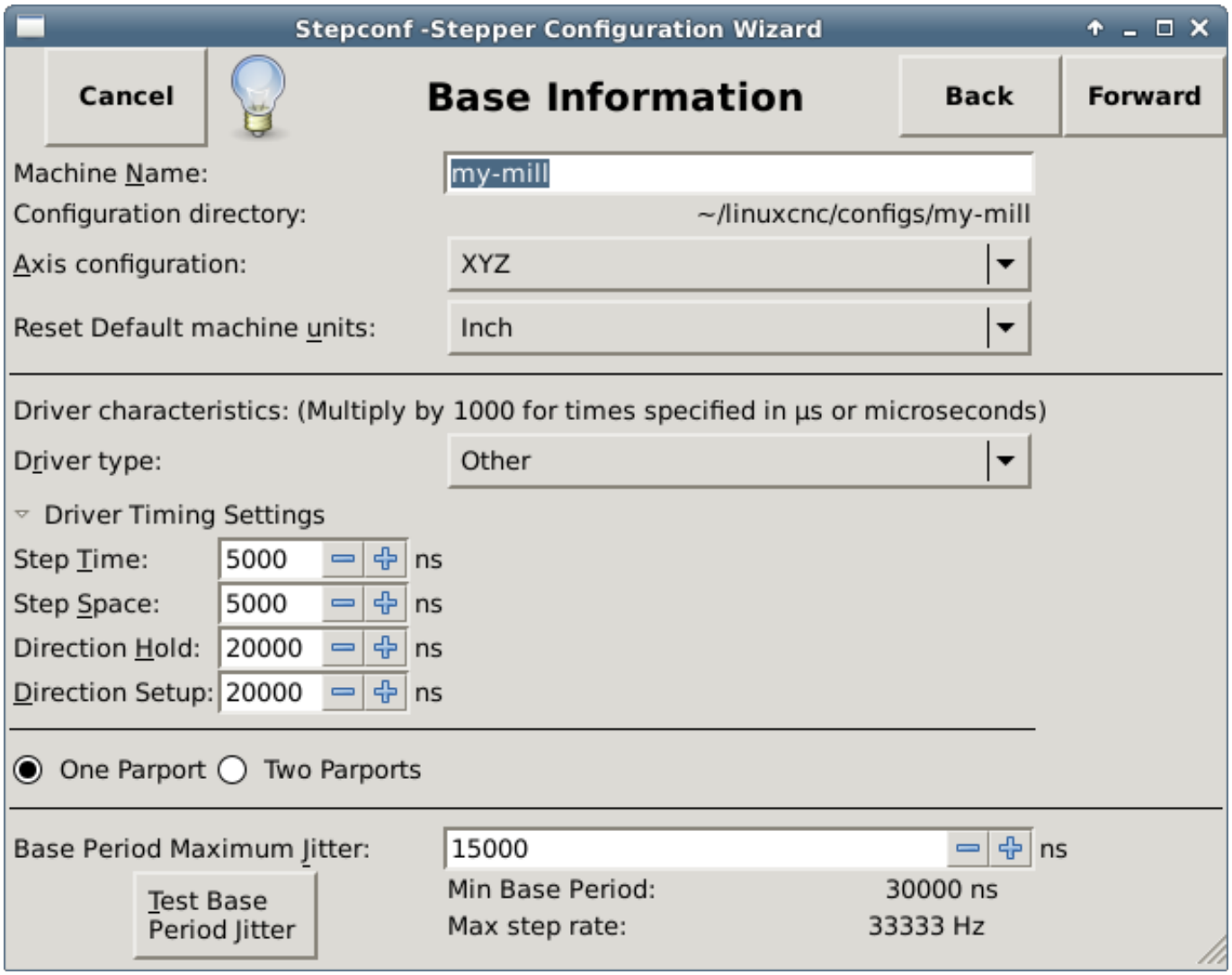



Abbildung 3.2: Seite mit grundlegenden Informationen

- *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.

  A list of some popular drives, along with their timing values, is on the LinuxCNC.org Wiki under 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

While the test is running, you should *abuse* the computer. Move windows around on the screen. Surf the web. Copy some large files around on the disk. Play some music. Run an OpenGL program such as glxgears. The idea is to put the PC through its paces while the latency test checks to see what the worst case numbers are. Run the test at least a few minutes. The longer you run the test the better it will be at catching events that might occur at less frequent intervals. This is a test for your computer only, so no hardware needs to be connected to run the test.

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

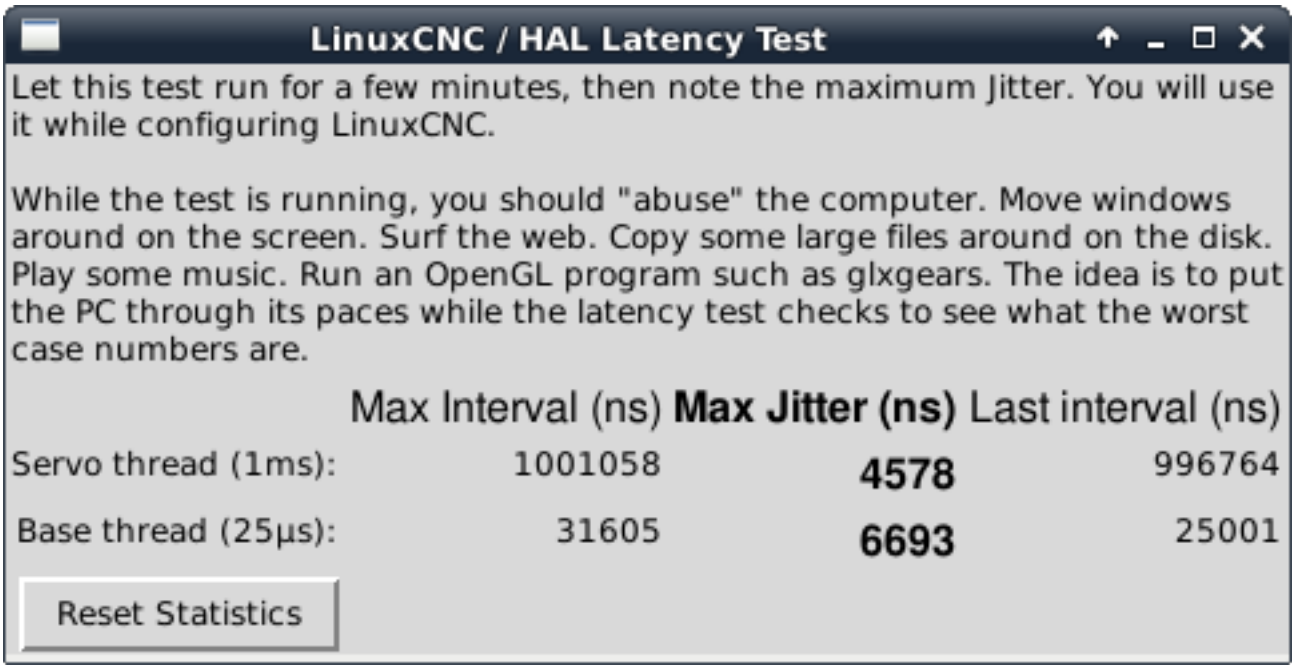



Abbildung 3.3: Latenz-Test

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

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

Die wichtigen Zahlen sind der *maximale Jitter*. Im obigen Beispiel ist 9075 Nanosekunden (ns), oder 9,075 Mikrosekunden (µs), der höchste Jitter. Notieren Sie diese Zahl, und geben Sie sie in das Feld Base Period Maximum Jitter ein.

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

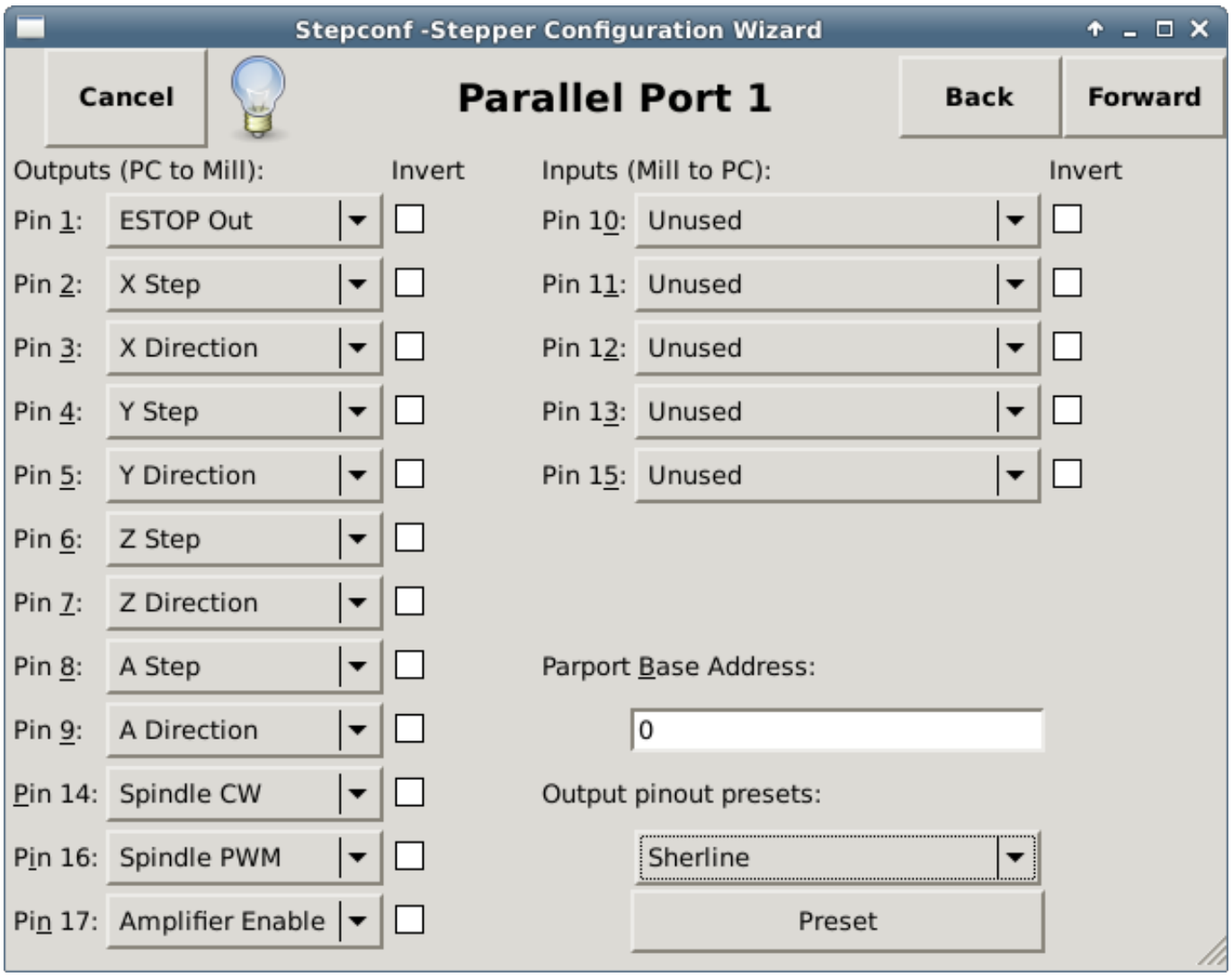



Abbildung 3.4: Parallele Schnittstelle Setup-Seite

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

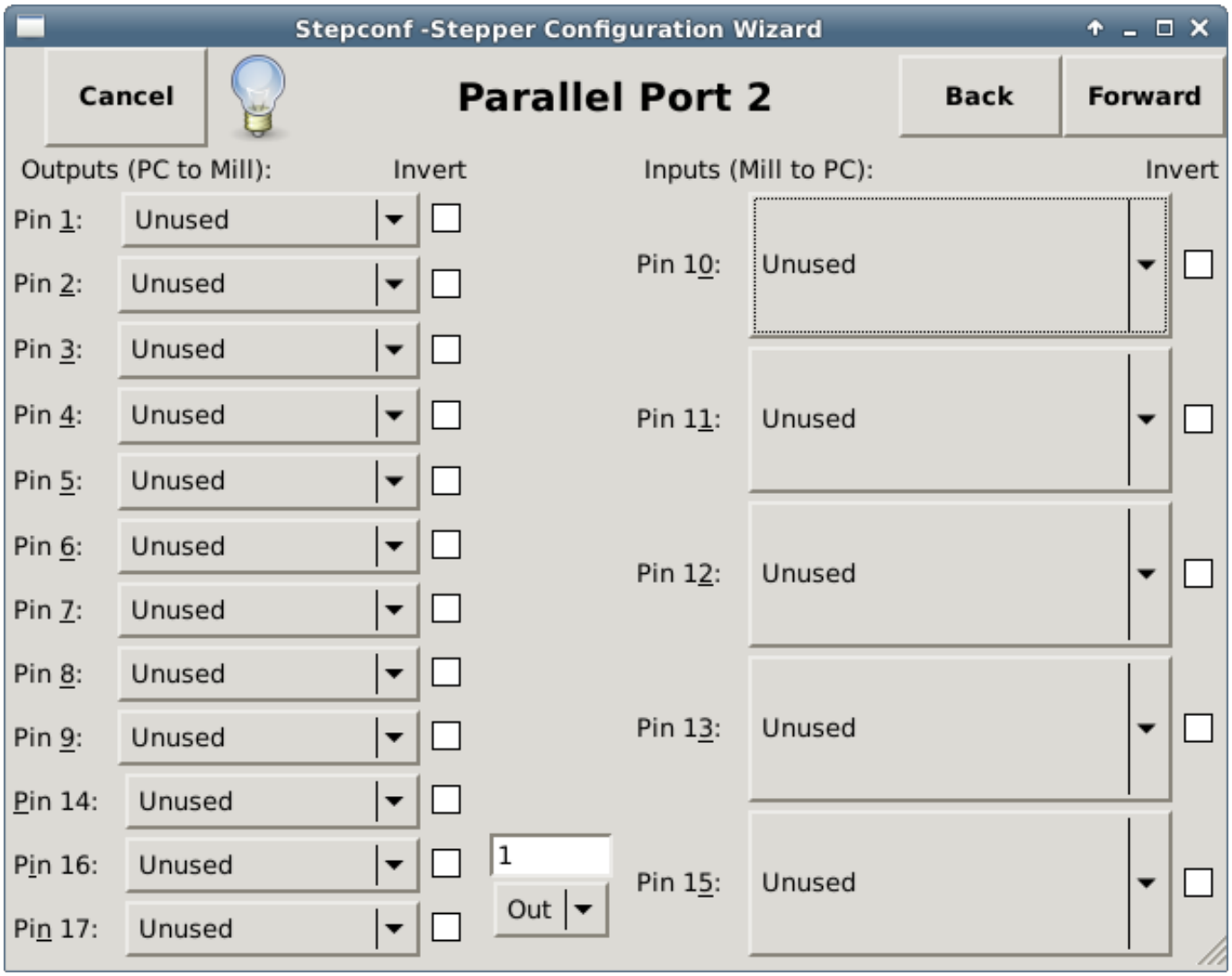



Abbildung 3.5: Einrichten von Parallel Port 2

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 Konfiguration der Achsen

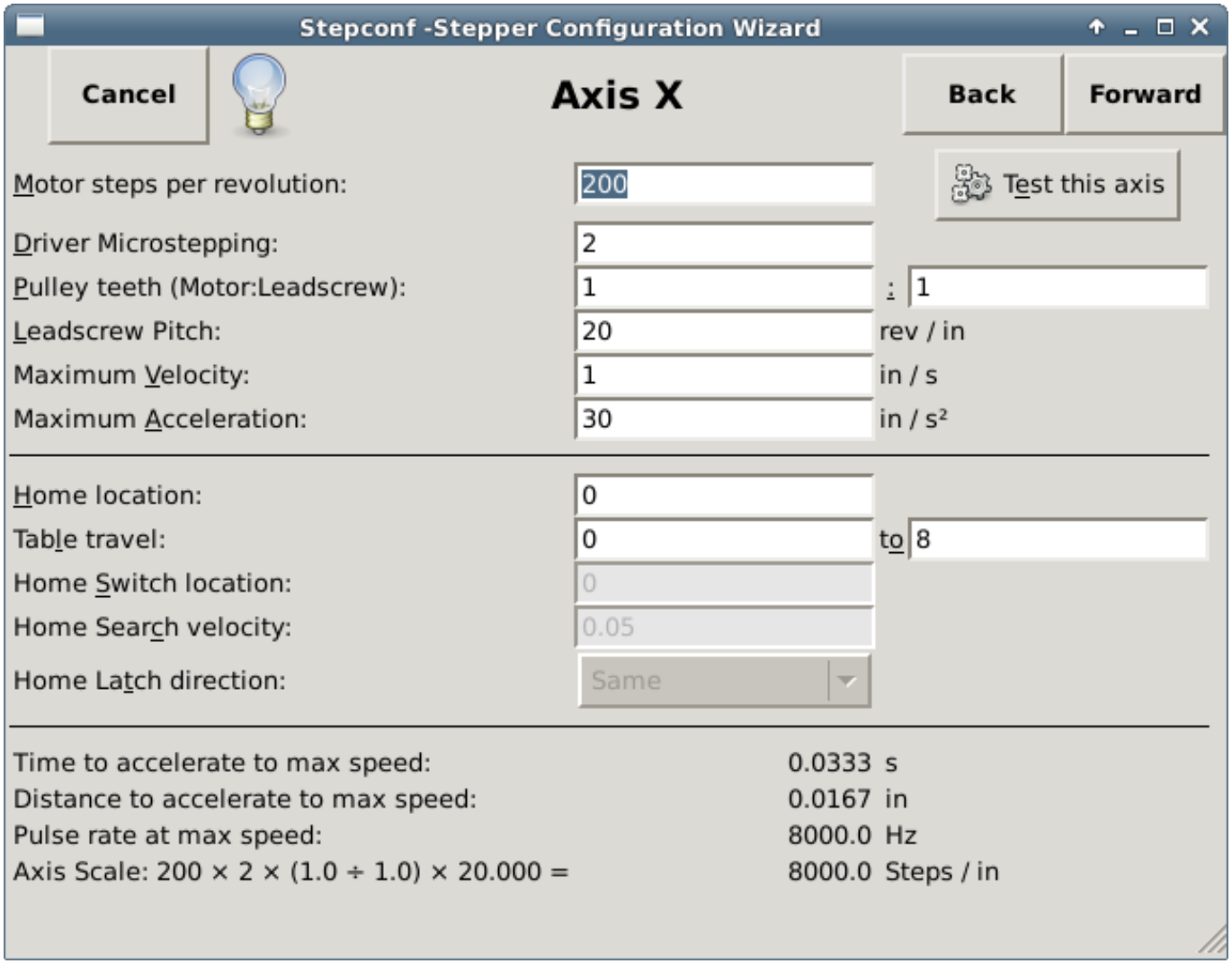



Abbildung 3.6: Achsenkonfiguration am Bildschirm

- *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.

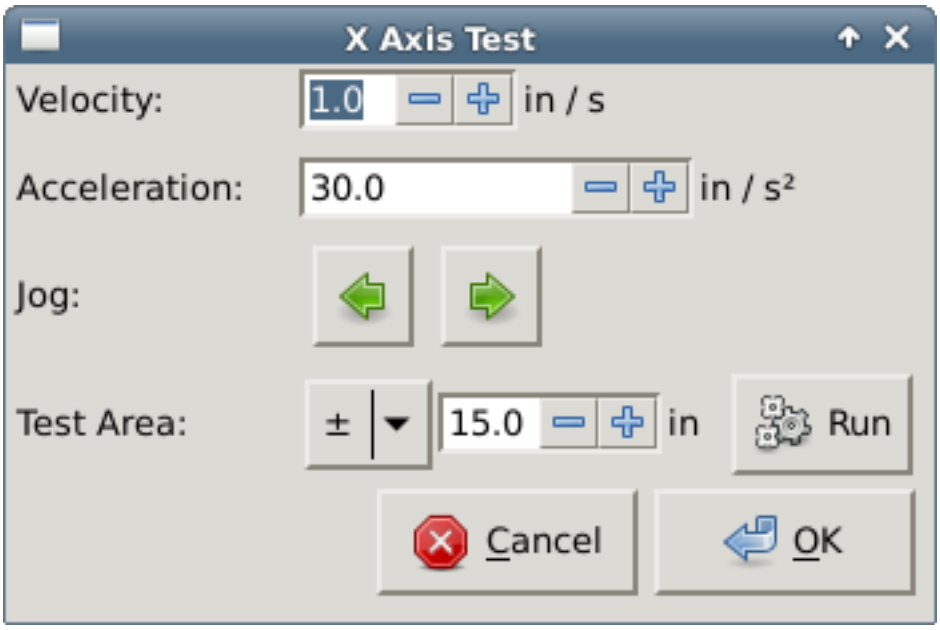



Abbildung 3.7: Achsen-Test

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 Finding Maximum Velocity

Begin with a low Acceleration (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 Finding Maximum Acceleration

With the Maximum Velocity you found in the previous step, enter the acceleration value to test. Using the same procedure as above, adjust the Acceleration value up or down as necessary. In this test, it is important that the combination of Acceleration and Test Area allow the machine to reach the selected Velocity. Once you have found a value at which the axis does not stall or lose steps during this testing procedure, reduce it by 10% and use that as the axis Maximum Acceleration.

### 3.1.8 Spindel-Konfiguration

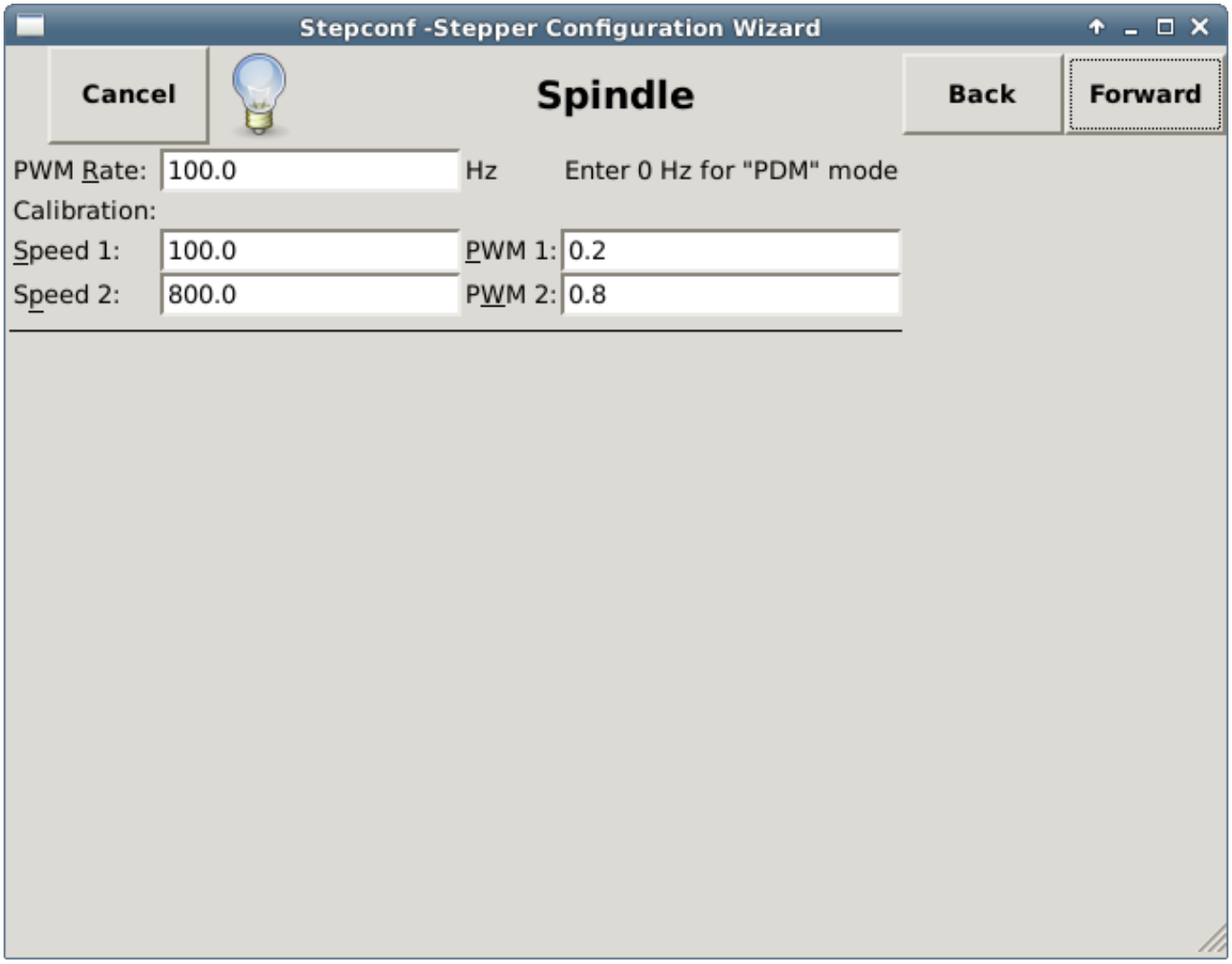



Abbildung 3.8: Seite zur Spindelkonfiguration

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

#### 3.1.8.1 Spindle Speed Control

If *Spindle PWM* appears on the pinout, the following information should be entered:

- *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 Spindle-synchronized motion

When the appropriate signals from a spindle encoder are connected to LinuxCNC via HAL, LinuxCNC supports lathe threading. These signals are:

- *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 Determining Spindle Calibration

Enter the following values in the Spindle Configuration page:

| | | | |
|---|---|---|---|
| **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

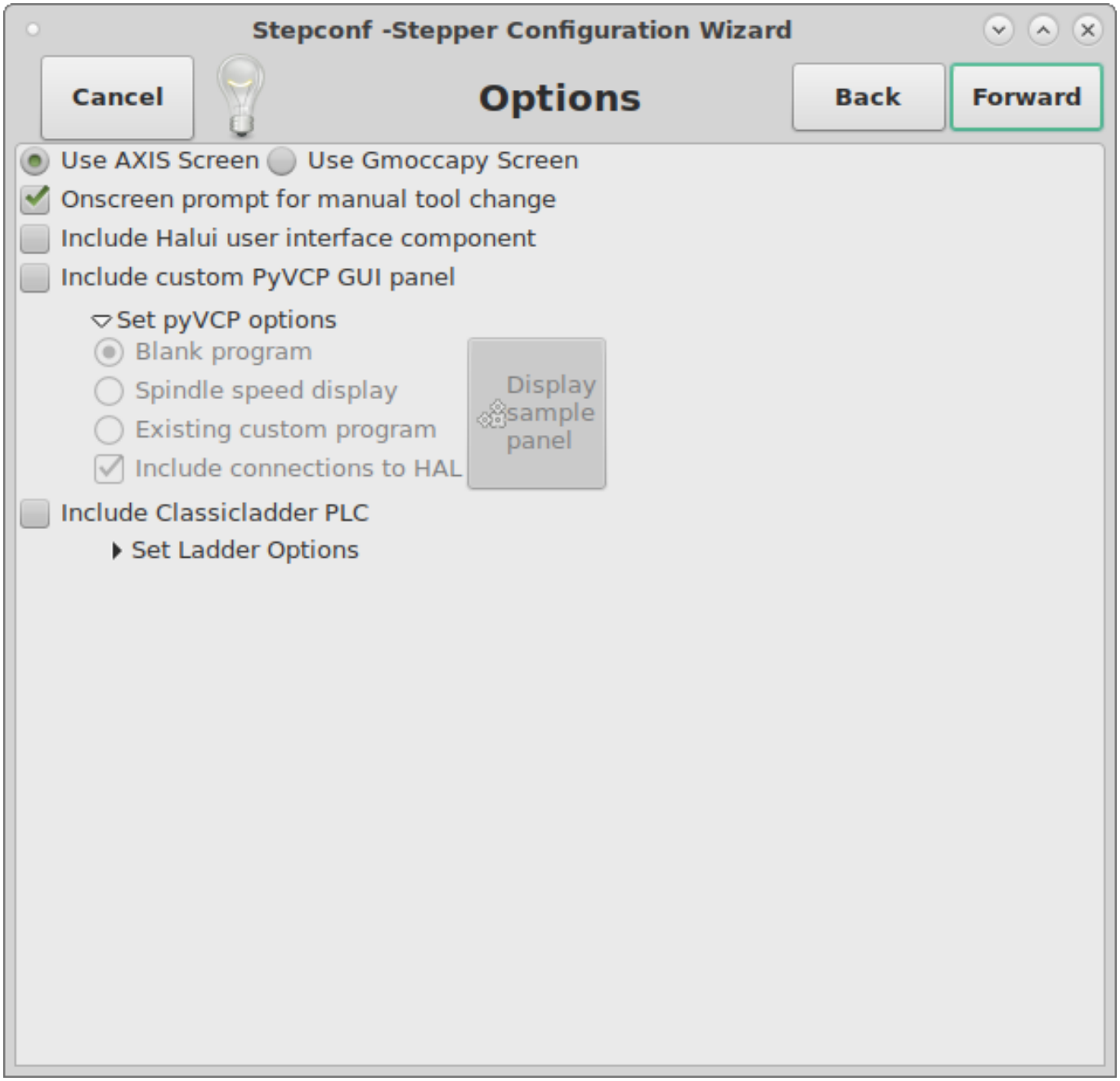



Abbildung 3.9: Erweiterte Optionen bei der Konfiguration

- *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 Complete Machine Configuration

Click *Apply* to write the configuration files. Later, you can re-run this program and tweak the settings you entered before.

### 3.1.11 Achsen Verfahrwege und Referenzpunkte

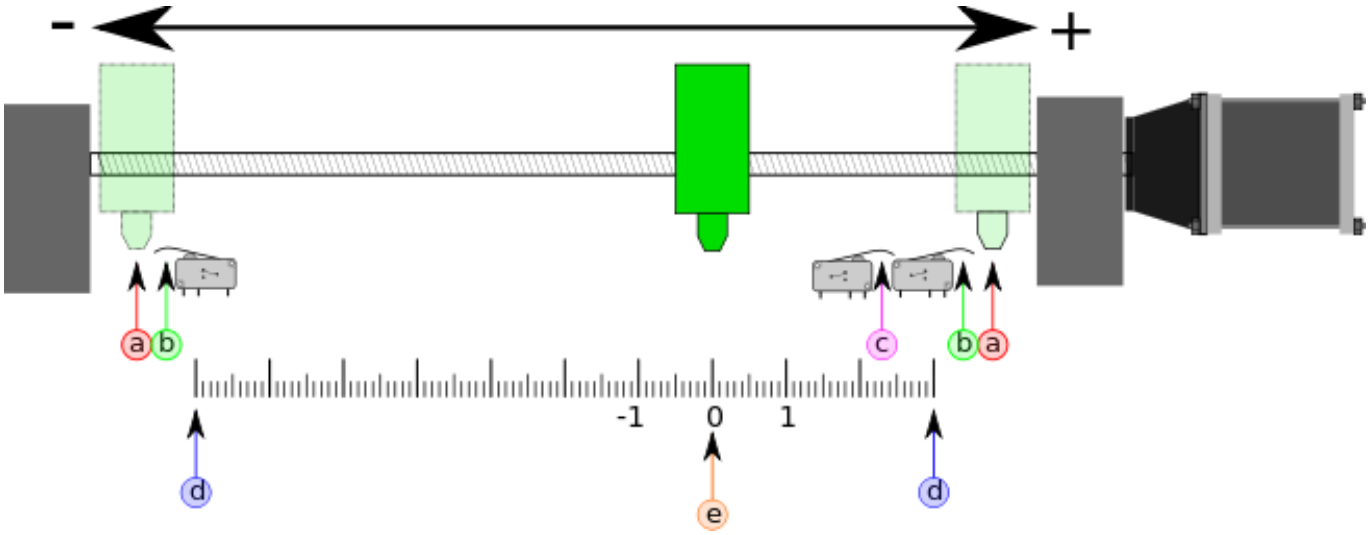



Abbildung 3.10: Achsen Verfahrwege und Referenzpunkt

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 Operating without Home Switches

A machine can be operated without home switches. If the machine has limit switches, but no home switches, it is best to use a limit switch as the home switch (e.g., choose *Minimum Limit + Home X* in the pinout). If the machine has no switches at all, or the limit switches cannot be used as home switches for another reason, then the machine must be homed *by eye* or by using match marks. Homing by eye is not as repeatable as homing to switches, but it still allows the soft limits to be useful.

#### 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;.

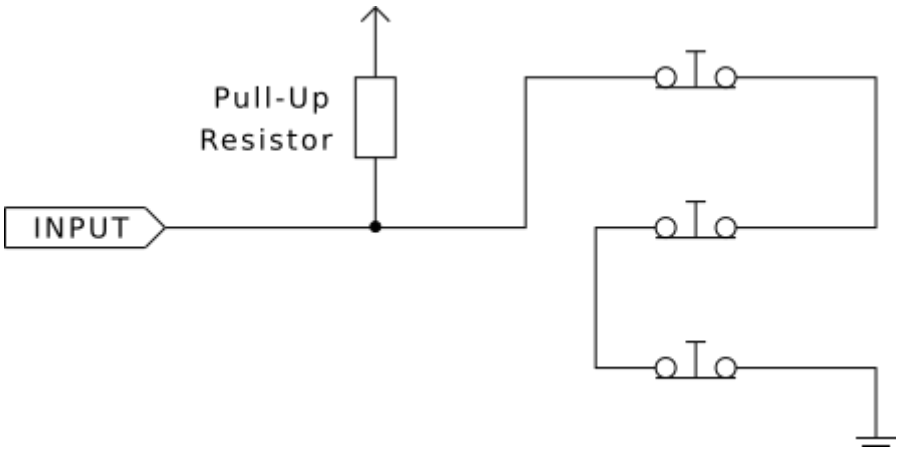



Abbildung 3.11: Normalerweise geschlossene Schalter (engl. normally closed, N/C) in Reihe verdrahtet (vereinfachtes Diagramm)

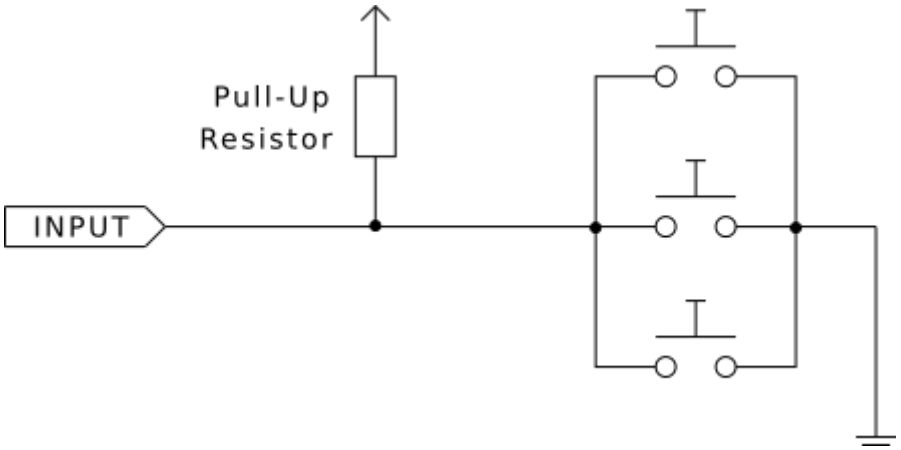



Abbildung 3.12: Normalerweise offene Schalter (engl. normally open switches, N/O) parallel verdrahtet (vereinfachte Darstellung)

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.

It can configure closed loop servo systems or hardware stepper systems. It uses a similar *wizard* approach as StepConf (used for software stepping, parallel port driven systems).

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:

One is to use PnCconf to always configure your system - if you decide to change options, reload PnCconf and allow it to configure the new options. This will work well if your machine is fairly standard and you can use custom files to add non standard features. PnCconf tries to work with you in this regard.

The other is to use PnCconf to build a config that is close to what you want and then hand edit everything to tailor it to your needs. This would be the choice if you need extensive modifications beyond PnCconf’s scope or just want to tinker with / learn about LinuxCNC.

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

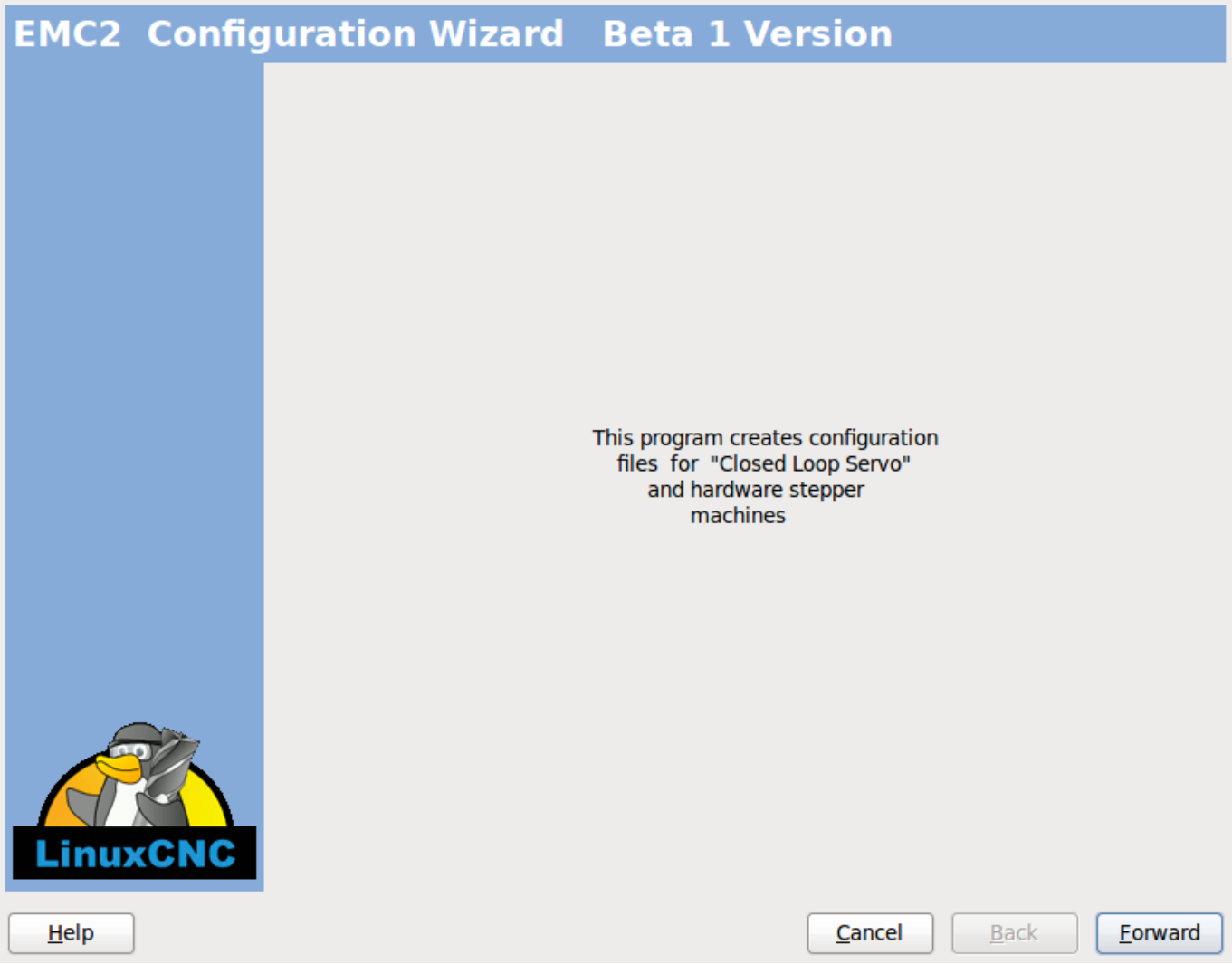



Abbildung 3.13: PnCconf Startfenster

### 3.2.2 Erstellen oder bearbeiten

This allows you to select a previously saved configuration or create a new one. If you pick *Modify a configuration* and then press *Next* a file selection box will show. PnCconf preselects your last saved file. Choose the config you wish to edit. If you made any changes to the main HAL or INI files **PnCconf will over write** those files and those changes will be lost. Some files will not be over written and PnCconf places a note in those files. It also allows you to select desktop shortcut / launcher options. A desktop shortcut will place a folder icon on the desktop that points to your new configuration files. Otherwise you would have to look in your home folder under linuxcnc/configs.

A Desktop launcher will add an icon to the desktop for starting your config directly. You can also

launch it from the main menu by using the Configuration Selector *LinuxCNC* found in CNC menu and selecting your config name.

### 3.2.3 Grundlegende Informationen zur Maschine

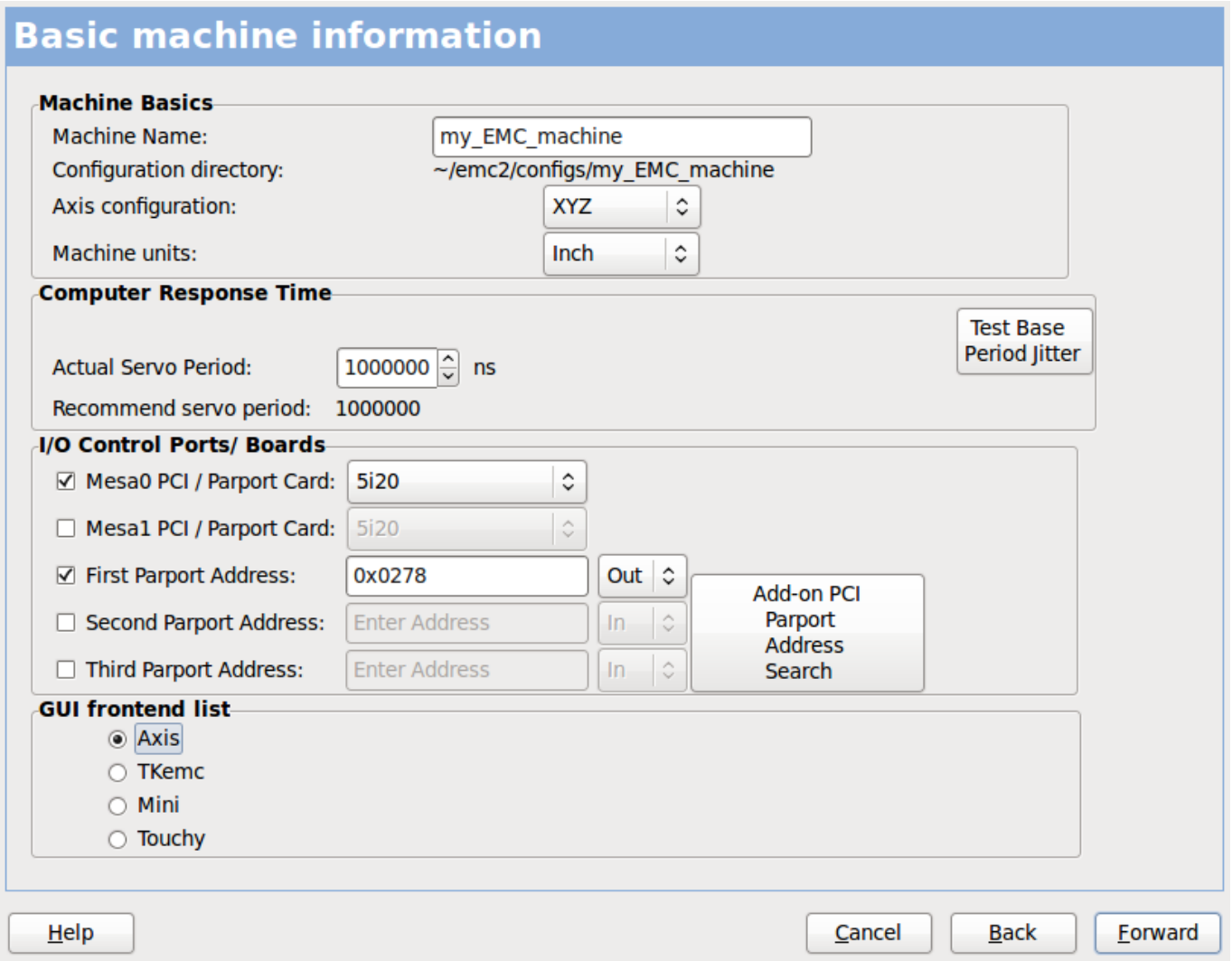



Abbildung 3.14: PnCconf Basic

**Maschinengrundlagen**

If you use a name with spaces PnCconf will replace the spaces with underscore (as a loose rule Linux doesn't like spaces in names) Pick an axis configuration - this selects what type of machine you are building and what axes are available. The Machine units selector allows data entry of metric or imperial units in the following pages.

---

**Tipp**

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

---

**Reaktionszeit des Computers**
: The servo period sets the heart beat of the system. Latency refers to the amount of time the computer can be longer then that period. Just like a railroad, LinuxCNC requires everything on a very tight and consistent time line or bad things happen. LinuxCNC requires and uses a *real time* operating system, which just means it has a low latency ( lateness ) response time when LinuxCNC requires its calculations and when doing LinuxCNCs calculations it cannot be interrupted by lower priority requests (such as user input to screen buttons or drawing etc).

Testing the latency is very important and a key thing to check early. Luckily by using the Mesa card to do the work that requires the fastest response time (encoder counting and PWM generation) we can endure a lot more latency then if we used the parallel port for these things. The standard test in LinuxCNC is checking the BASE period latency (even though we are not using a base period). If you press the *test base period jitter* button, this launches the latency test window ( you can also load this directly from the applications/cnc panel ). The test mentions to run it for a few minutes but the longer the better. Consider 15 minutes a bare minimum and overnight even better. At this time use the computer to load things, use the net, use USB etc we want to know the worst case latency and to find out if any particular activity hurts our latency. We need to look at base period jitter. Anything under 20000 is excellent - you could even do fast software stepping with the machine 20000 - 50000 is still good for software stepping and fine for us. 50000 - 100000 is really not that great but could still be used with hardware cards doing the fast response stuff. So anything under 100000 is usable to us. If the latency is disappointing or you get a bad hiccup periodically you may still be able to improve it.

---

**Tipp**
There is a user compiled list of equipment and the latency obtained on the LinuxCNC wiki: https://wiki.linuxcnc.org/cgi-bin/wiki.pl?Latency-Test Please consider adding your info to the list. Also on that page are links to info about fixing some latency problems.

---

Now we are happy with the latency and must pick a servo period. In most cases a servo period of 1000000 ns is fine (that gives a 1 kHz servo calculation rate - 1000 calculations a second). If you are building a closed loop servo system that controls torque (current) rather then velocity (voltage) a faster rate would be better - something like 200000 (5 kHz calculation rate). The problem with lowering the servo rate is that it leaves less time available for the computer to do other things besides LinuxCNC's calculations. Typically the display (GUI) becomes less responsive. You must decide on a balance. Keep in mind that if you tune your closed loop servo system then change the servo period you probably will need to tune them again.

**E/A-Steueranschlüsse/Karten (engl. I/O Control Ports/Boards)**
: PnCconf is capable of configuring machines that have up to two Mesa boards and three parallel ports. Parallel ports can only be used for simple low speed (servo rate) I/O.

**Mesa**
: You must choose at least one Mesa board as PnCconf will not configure the parallel ports to count encoders or output step or PWM signals. The mesa cards available in the selection box are based on what PnCconf finds for firmware on the systems. There are options to add custom firmware and/or *blacklist* (ignore) some firmware or boards using a preference file. If no firmware is found PnCconf will show a warning and use internal sample firmware - no testing will be possible. One point to note is that if you choose two PCI Mesa cards there currently is no way to predict which card is 0 and which is 1 - you must test - moving the cards could change their order. If you configure with two cards both cards must be installed for tests to function.

**Parallelport**
: Up to 3 parallel ports (referred to as parports) can be used as simple I/O. You must set the address of the parport. You can either enter the Linux parallel port numbering system (0,1,or 2) or enter the actual address. The address for an on board parport is often 0x0278 or 0x0378 (written in hexadecimal) but can be found in the BIOS page. The BIOS page is found when you first start your computer you must press a key to enter it (such as F2). On the BIOS page you can find

the parallel port address and set the mode such as SPP, EPP, etc on some computers this info is displayed for a few seconds during start up. For PCI parallel port cards the address can be found by pressing the *parport address search* button. This pops up the help output page with a list of all the PCI devices that can be found. In there should be a reference to a parallel port device with a list of addresses. One of those addresses should work. Not all PCI parallel ports work properly. Either type can be selected as *in* (maximum amount of input pins) or *out* (maximum amount of output pins).

**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.

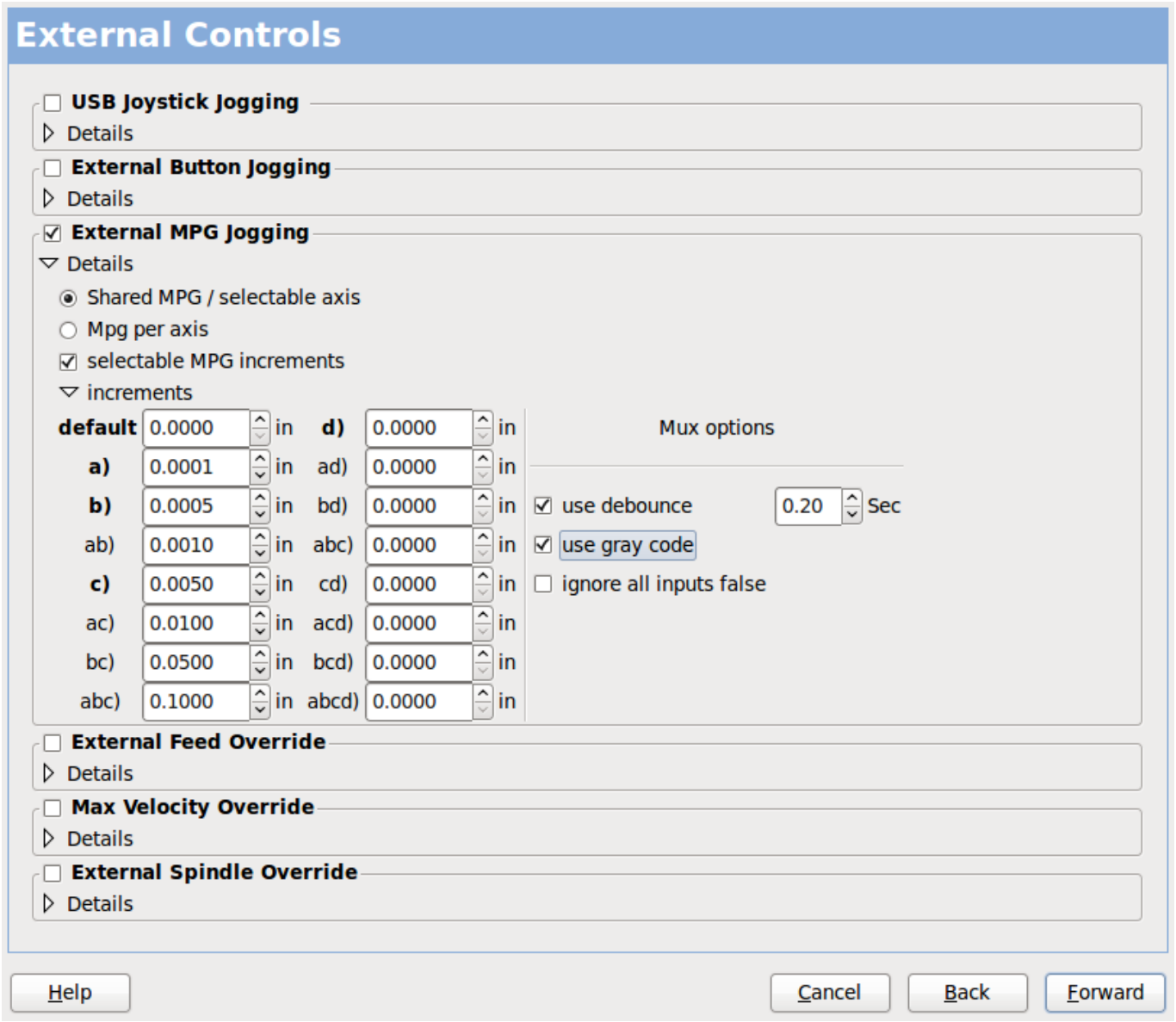



Abbildung 3.15: Externe Steuerelemente

If you select a Joystick for jogging, You will need it always connected for LinuxCNC to load. To use the analog sticks for useful jogging you probably need to add some custom HAL code. MPG jogging requires a pulse generator connected to a MESA encoder counter. Override controls can either use a pulse generator (MPG) or switches (such as a rotary dial). External buttons might be used with a switch based OEM joystick.

**Joystick-Joggen**
: Requires a custom *device rule* to be installed in the system. This is a file that LinuxCNC uses to connect to Linux's device list. PnCconf will help to prepare this file.

  - *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**
: allows jogging the axis with simple buttons at a specified jog rate. Probably best for rapid jogging.

**Handrad (engl. MPG) Jogging**
: Ermöglicht die Verwendung eines Handrads (manuellen Impulsgebers, MPG), um die Achsen der Maschine zu bewegen.

MPG's are often found on commercial grade machines. They output quadrature pulses that can be counted with a MESA encoder counter. PnCconf allows for an MPG per axis or one MPG shared with all axis. It allows for selection of jog speeds using switches or a single speed.

The selectable increments option uses the mux16 component. This component has options such as debounce and gray code to help filter the raw switch input.

**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.

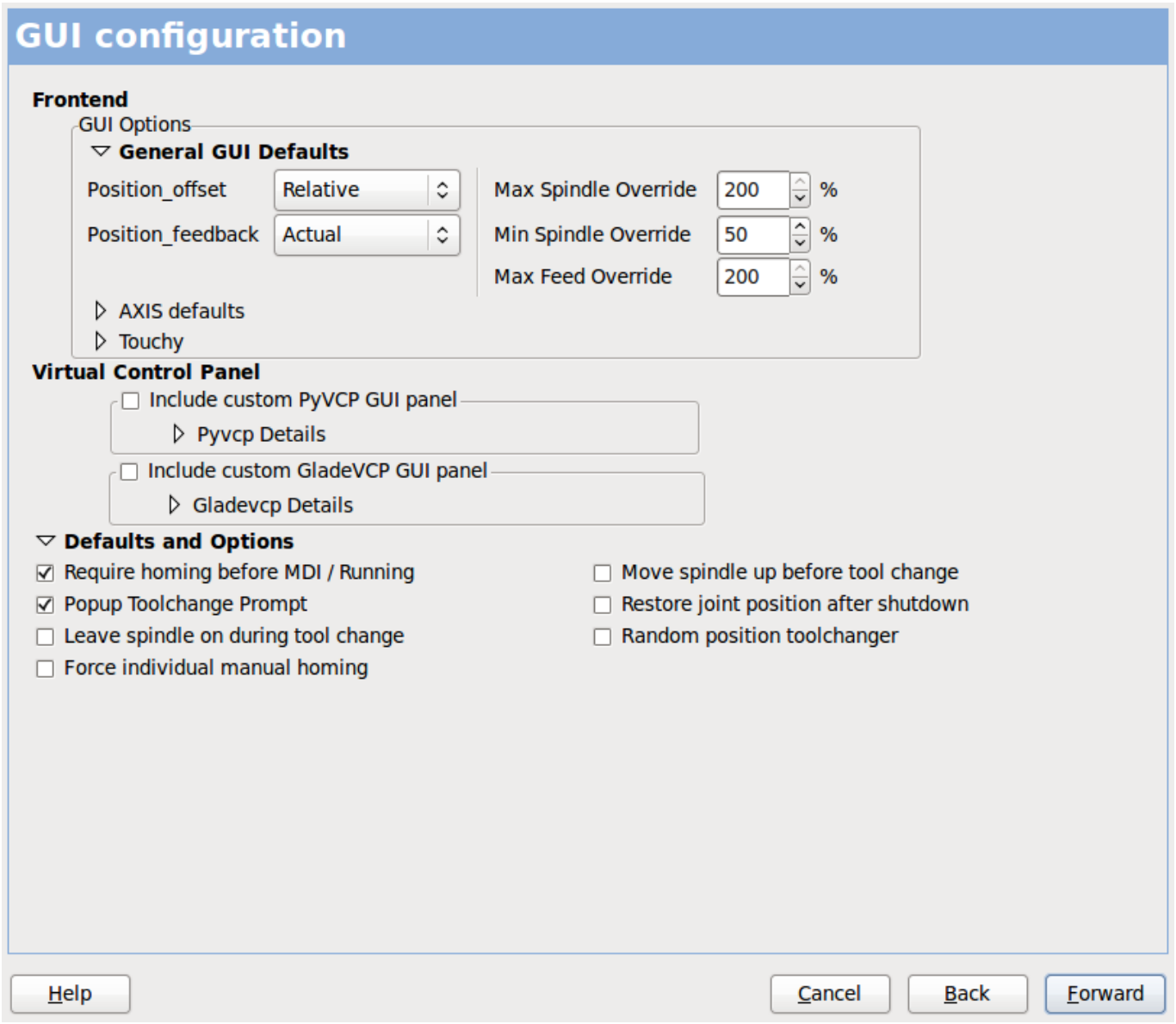



Abbildung 3.16: GUI-Konfiguration

**Front-End GUI-Optionen**

Die Standardoptionen ermöglichen die Auswahl allgemeiner Standardeinstellungen für jeden Anzeigebildschirm.

AXIS defaults are options specific to AXIS. If you choose size, position or force maximize options then PnCconf will ask if it is alright to overwrite a preference file (.axisrc). Unless you have manually added commands to this file it is fine to allow it. Position and force max can be used to move AXIS to a second monitor if the system is capable.

Touchy defaults are options specific to Touchy. Most of Touchy's options can be changed while Touchy is running using the preference page. Touchy uses GTK to draw its screen, and GTK supports themes. Themes controls the basic look and feel of a program. You can download themes from the net or edit them yourself. There are a list of the current themes on the computer that you can pick from. To help some of the text to stand out PnCconf allows you to override the Themes's defaults. The position and force max options can be used to move Touchy to a second monitor if the system is capable.

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**
: Virtual Control Panels allow one to add custom controls and displays to the screen. AXIS and Touchy can integrate these controls inside the screen in designated positions. There are two kinds of VCPs - PyVCP which uses *Tkinter* to draw the screen and GladeVCP that uses *GTK* to draw the screen.

**PyVCP**
: PyVCPs screen XML file can only be hand built. PyVCPs fit naturally in with AXIS as they both use TKinter.

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.

You can use the geometry options to size and move the panel, for instance to move it to a second screen if the system is capable. If you press the *Display sample panel* button the size and placement options will be honored.

**GladeVCP**
: GladeVCPs passen natürlich in den Touchy-Bildschirm, da beide GTK verwenden, um sie zu zeichnen, aber durch die Änderung von GladeVCPs Thema kann es möglich werden, dies optisch ziemlich gut in AXIS zu integrieren (versuchen Sie Redmond).

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

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

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

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

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

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

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

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

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

**Standardeinstellungen und Optionen**

- Referenzfahrt vor Manueller Dateneingabe / Ausführung erforderlich machen
  - Wenn Sie möchten, dass die Maschine vor der Referenzfahrt bewegt werden kann, deaktivieren Sie dieses Kontrollkästchen.
- Popup-Tool-Eingabeaufforderung
  - Wählen Sie zwischen einer Bildschirmabfrage für Werkzeugwechsel oder dem Export von Standardsignalnamen für eine benutzerdefinierte Werkzeugwechsler-HAL-Datei
- Lassen Sie die Spindel während des Werkzeugwechsels eingeschaltet:
  - Verwendet für Drehbänke
- Individuelle manuelle Referenzfahrt erzwingen
- Spindel vor dem Werkzeugwechsel nach oben fahren
- Wiederherstellung der Gelenkposition nach Abschaltung
  - Verwendet für nichttriviale kinematische Maschinen
- Werkzeugwechsler mit zufälliger (engl. random) Position
  - Wird für Werkzeugwechsler verwendet, die das Werkzeug nicht in dieselbe Tasche zurückbringen. Um Werkzeugwechsler zu unterstützen, müssen Sie einen eigenen HAL-Code hinzufügen.

### 3.2.6 Mesa-Konfiguration

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

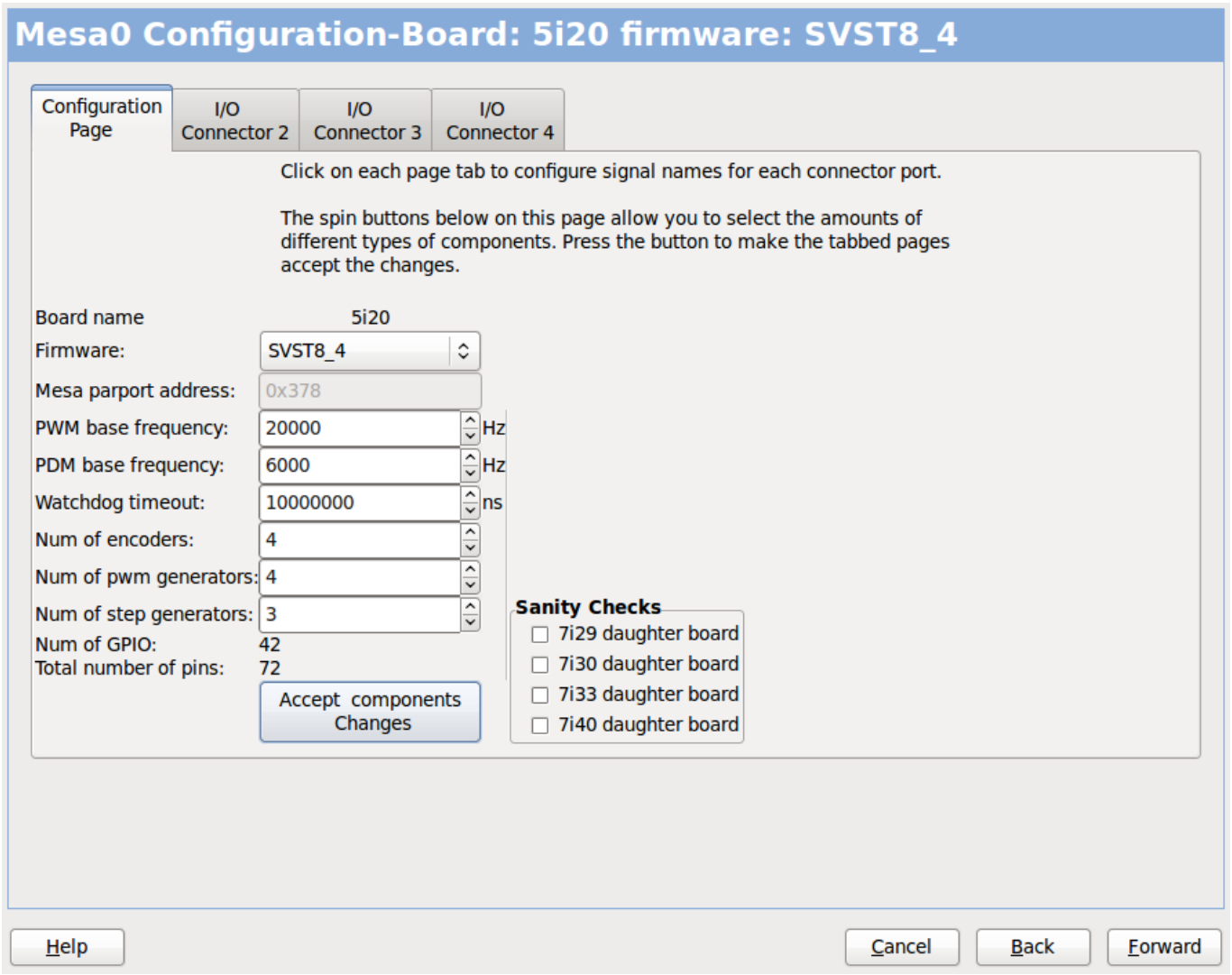



Abbildung 3.17: Mesa Board Konfiguration

Parport address is used only with Mesa parport card, the 7i43. An on board parallel port usually uses 0x278 or 0x378 though you should be able to find the address from the BIOS page. The 7i43 requires the parallel port to use the EPP mode, again set in the BIOS page. If using a PCI parallel port the address can be searched for by using the search button on the basic page.

**Anmerkung**
Viele PCI-Karten unterstützen das EPP-Protokoll nicht richtig.

PDM PWM and 3PWM base frequency sets the balance between ripple and linearity. If using Mesa daughter boards the docs for the board should give recommendations.

**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**
: is used to set how long the MESA board will wait before killing outputs if communication is interrupted from the computer. Please remember Mesa uses *active low* outputs meaning that when the output pin is on, it is low (approx 0 volts) and if it is off the output in high (approx 5 volts) make sure your equipment is safe when in the off (watchdog bitten) state.

**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.

Choosing less then the maximum number of components allows one to gain more GPIO pins. If using daughter boards keep in mind you must not deselect pins that the card uses. For instance some firmware supports two 7i33 cards, If you only have one you may deselect enough components to utilize the connector that supported the second 7i33. Components are deselected numerically by the highest number first then down with out skipping a number. If by doing this the components are not where you want them then you must use a different firmware. The firmware dictates where, what and the max amounts of the components. Custom firmware is possible, ask nicely when contacting the LinuxCNC developers and Mesa. Using custom firmware in PnCconf requires special procedures and is not always possible - though I try to make PnCconf as flexible as possible.

After choosing all these options press the *Accept Component Changes* button and PnCconf will update the I/O setup pages. Only I/O tabs will be shown for available connectors, depending on the Mesa board.

### 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.

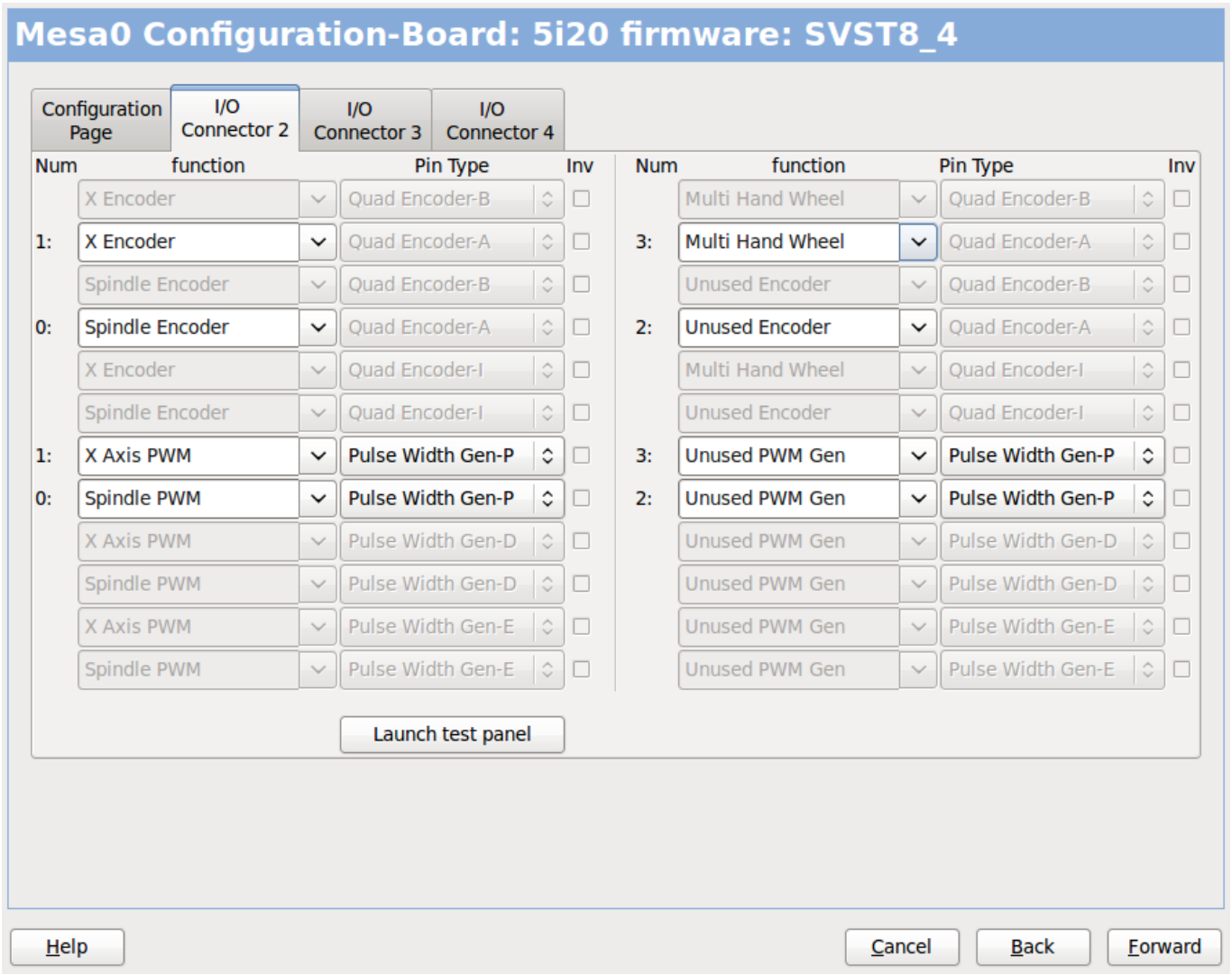



Abbildung 3.18: Mesa I/O C2 Einrichtung

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.

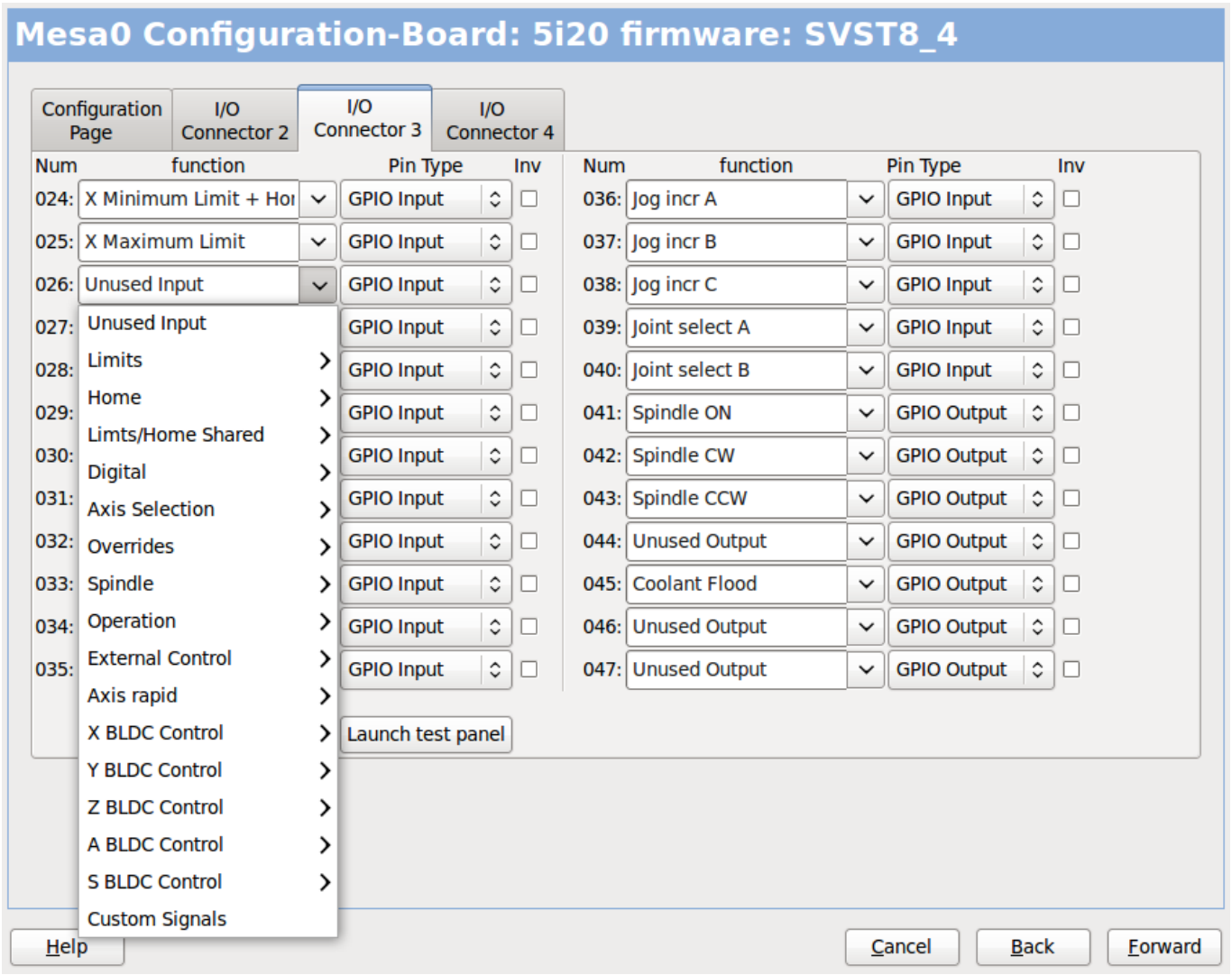



Abbildung 3.19: Mesa I/O C3 Einrichtung

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.

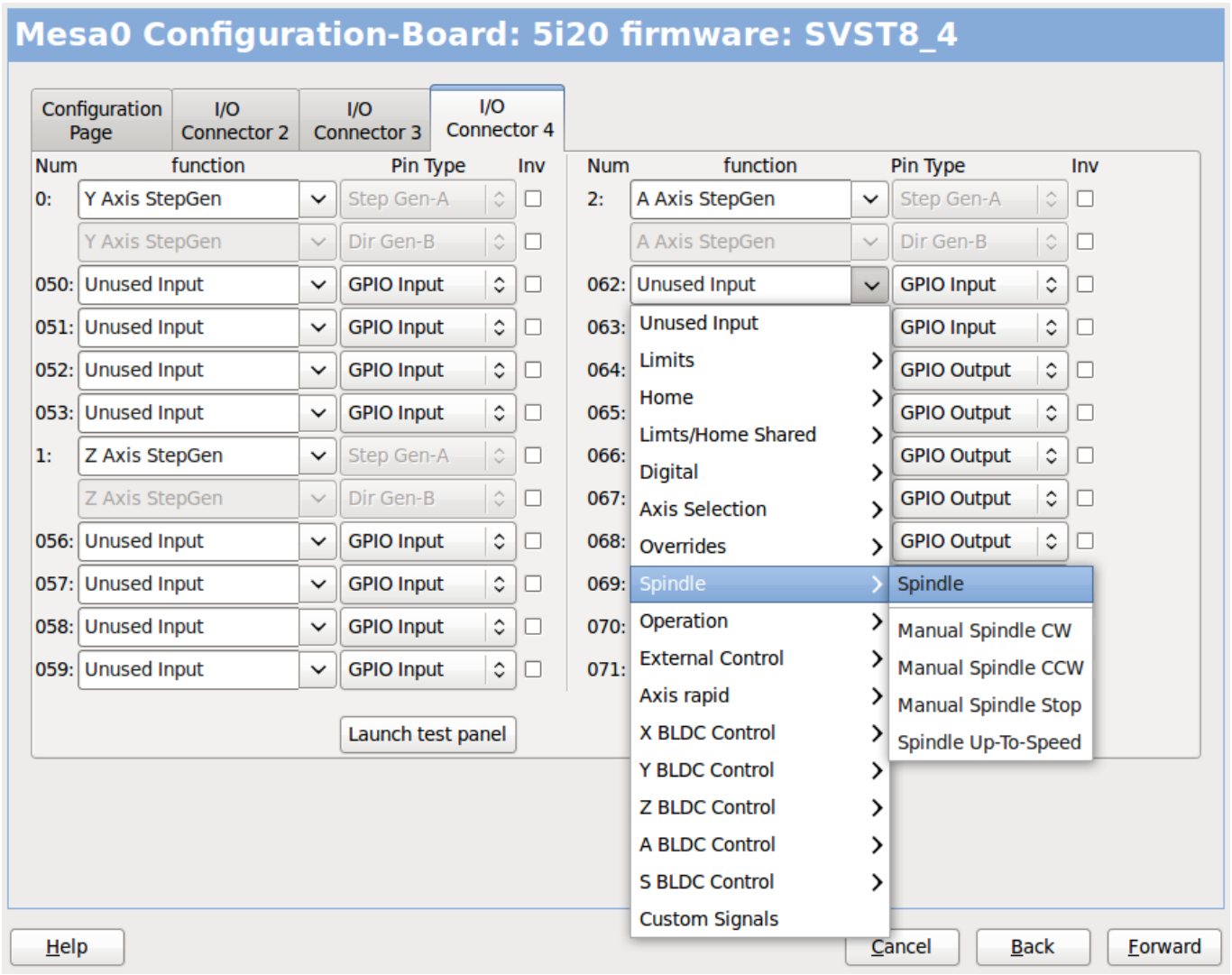
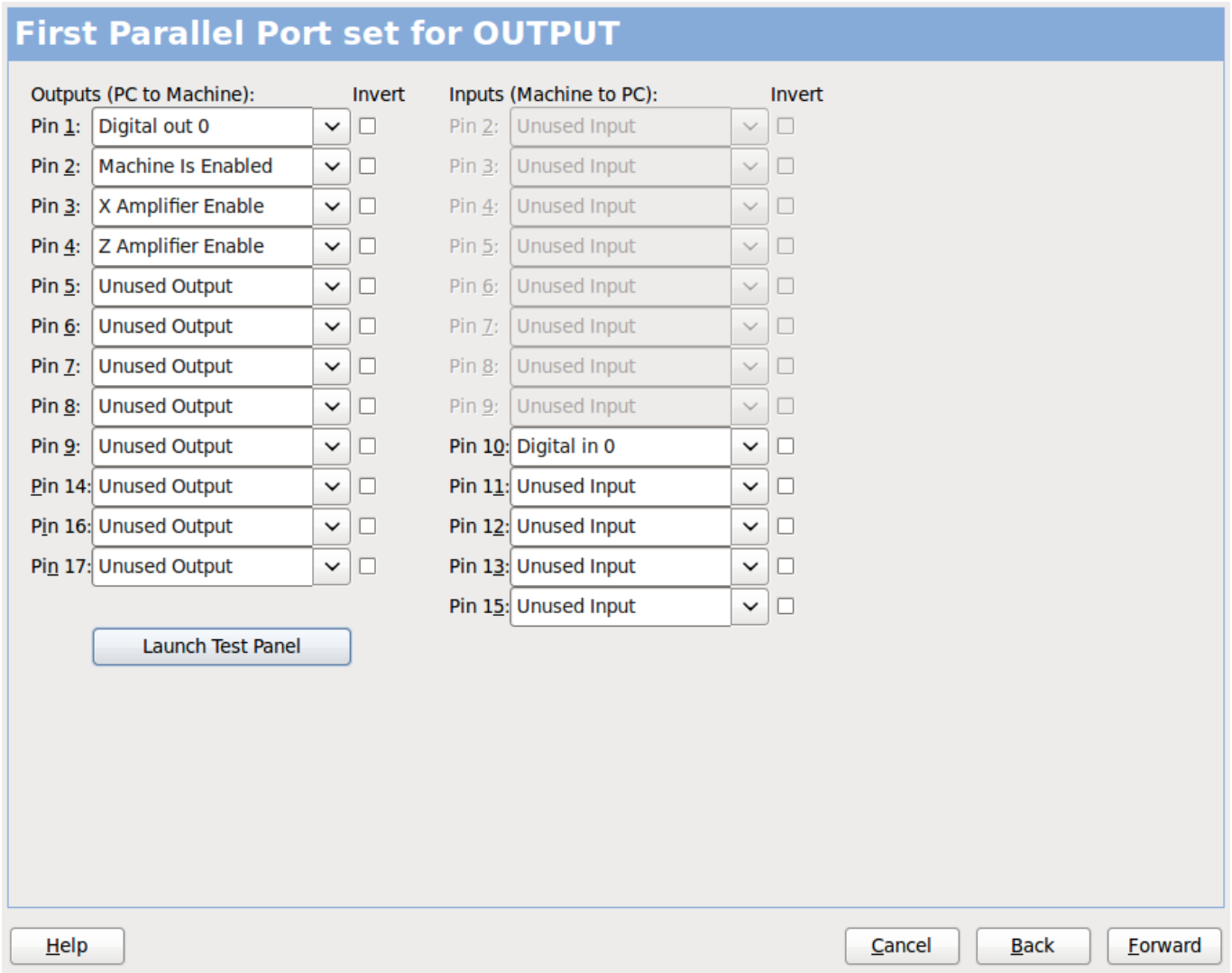



Abbildung 3.20: Mesa I/O C4 Einrichtung

On this tab there are a mix of step generators and GPIO. Step generators output and direction pins can be inverted. Note that inverting a Step Gen-A pin (the step output pin) changes the step timing. It should match what your controller expects.

### 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

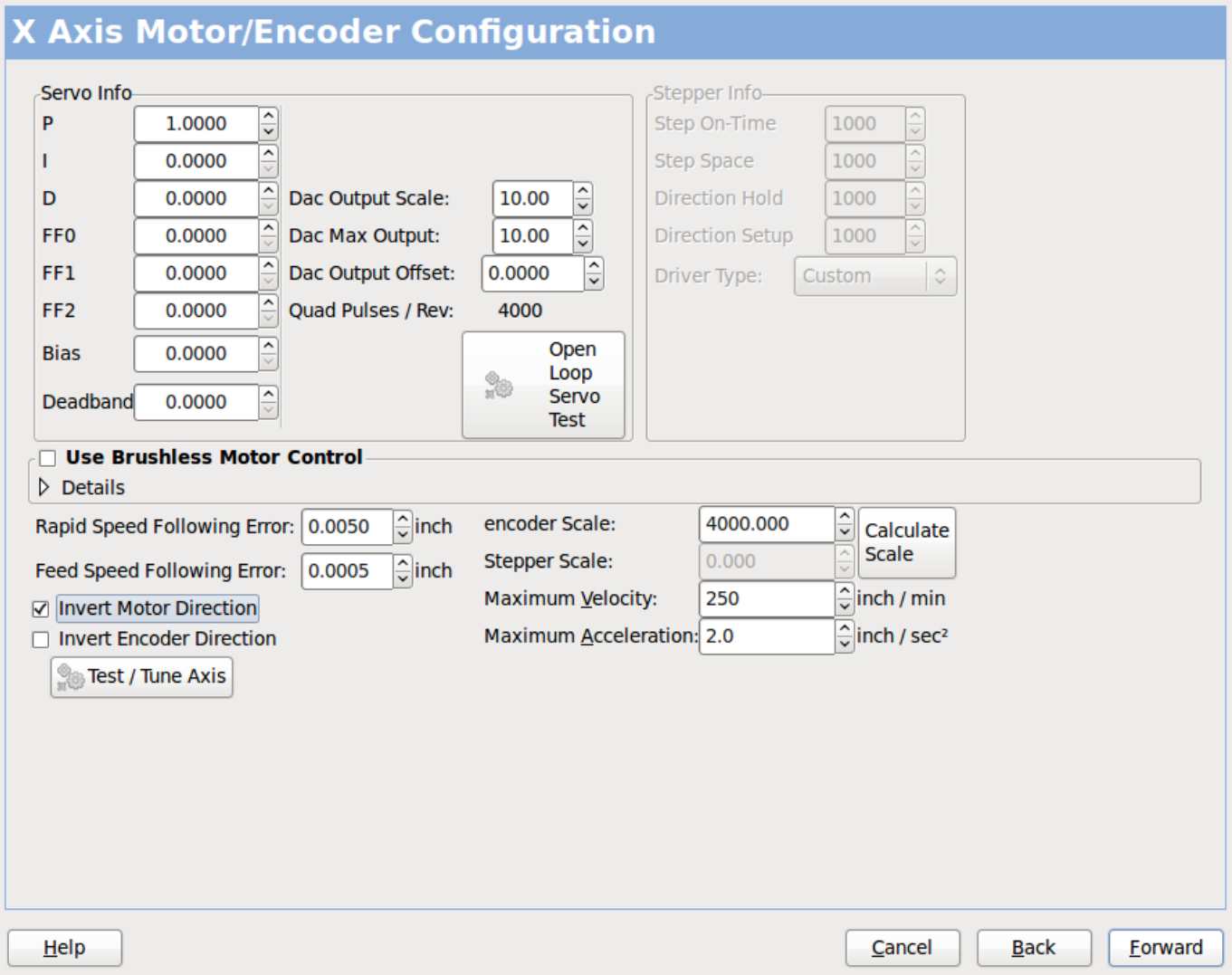



Abbildung 3.21: Konfiguration des Achsantriebs

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**

An open loop test is important as it confirms the direction of the motor and encoder. The motor should move the axis in the positive direction when the positive button is pushed and also the encoder should count in the positive direction. The axis movement should follow the Machinery's Handbook [1] standards or AXIS graphical display will not make much sense. Hopefully the help page and diagrams can help figure this out. Note that axis directions are based on TOOL movement not table movement. There is no acceleration ramping with the open loop test so start with lower DAC numbers. By moving the axis a known distance one can confirm the encoder scaling. The encoder should count even without the amp enabled depending on how power is supplied to the encoder.

[1] "axis nomenclature" in the chapter "Numerical Control" in the "Machinery's Handbook" published by Industrial Press.

**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.

Specifically, when writing outputs, the LinuxCNC first converts the desired output in quasi-SI units to raw actuator values, e.g., volts for an amplifier DAC. This scaling looks like: The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [output SI units]/[actuator units]. For example, on a machine with a velocity mode amplifier such that 1 volt results in 250 mm/sec velocity. Note that the units of the offset are in machine units, e.g., mm/sec, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of your output which yields 0.0 for the actuator output. If the DAC is linearized, this offset is normally 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 Koeffizienten a und b so zu ermitteln, dass meas=a*raw+b
- Beachten Sie, dass wir eine Rohausgabe wünschen, bei der das gemessene Ergebnis mit der befohlenen Ausgabe identisch ist. Das bedeutet
  - cmd=a*raw+b

- raw=(cmd-b)/a

- Folglich können die Koeffizienten a und b aus der linearen Anpassung direkt als Skala und Offset für den Regler verwendet werden.

**MAX OUTPUT**
The maximum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units. The value is applied symmetrically to both the plus and the minus side.

**Tuning-Test**
The tuning test unfortunately only works with stepper based systems. Again confirm the directions on the axis is correct. Then test the system by running the axis back and forth, If the acceleration or max speed is too high you will lose steps. While jogging, Keep in mind it can take a while for an axis with low acceleration to stop. Limit switches are not functional during this test. You can set a pause time so each end of the test movement. This would allow you to set up and read a dial indicator to see if you are losing steps.

**Schrittmotor (Stepper-)Timing**
Stepper timing needs to be tailored to the step controller's requirements. PnCconf supplies some default controller timing or allows custom timing settings. See https://wiki.linuxcnc.org/-cgi-bin/wiki.pl?Stepper_Drive_Timing for some more known timing numbers (feel free to add ones you have figured out). If in doubt use large numbers such as 5000 this will only limit max speed.

**Bürstenlose Motorsteuerung**
These options are used to allow low level control of brushless motors using special firmware and daughter boards. It also allows conversion of HALL sensors from one manufacturer to another. It is only partially supported and will require one to finish the HAL connections. Contact the mail-list or forum for more help.

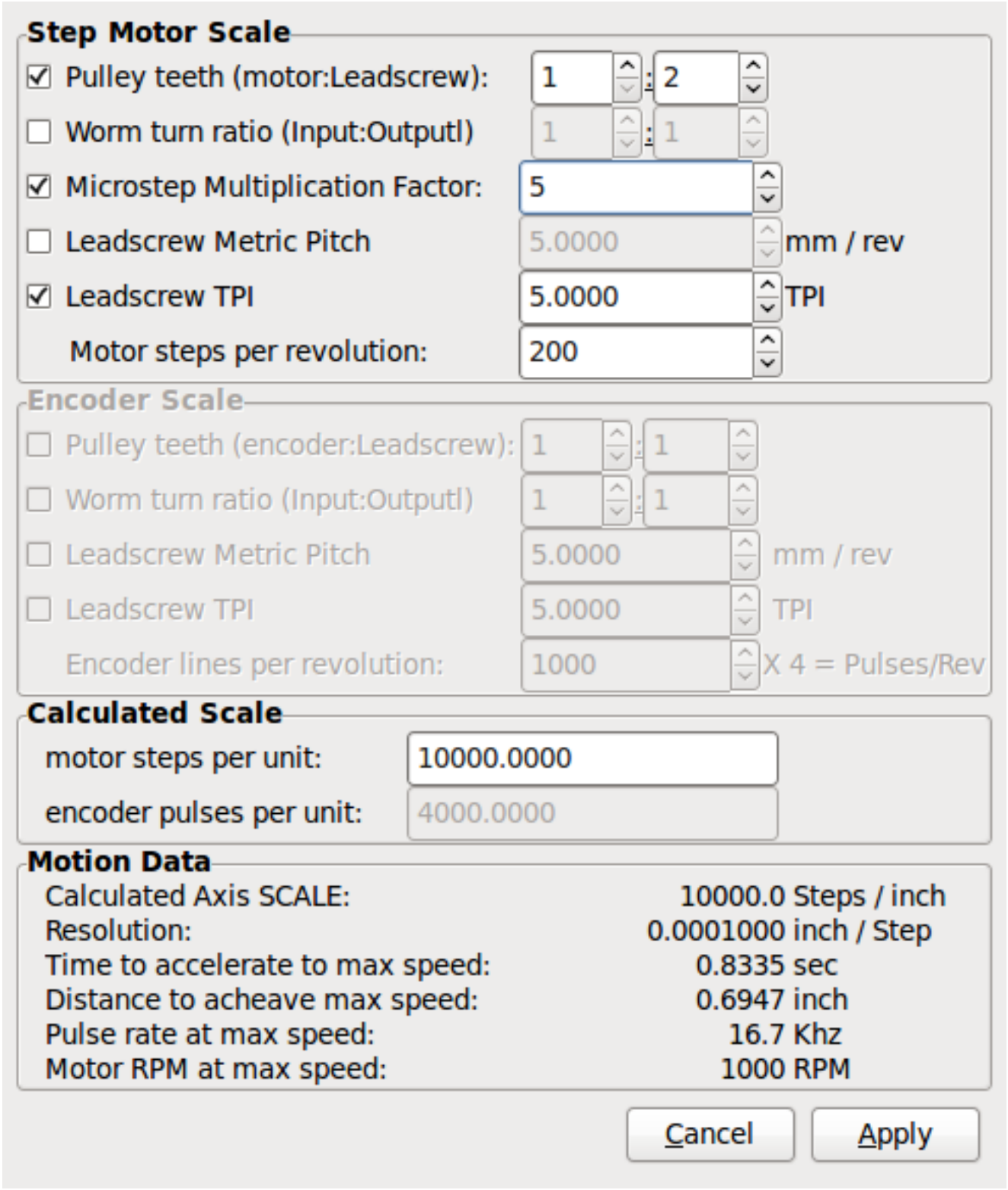



Abbildung 3.22: Berechnung der Achsenskala

The scale settings can be directly entered or one can use the *calculate scale* button to assist. Use the check boxes to select appropriate calculations. Note that *pulley teeth* requires the number of teeth not the gear ratio. Worm turn ratio is just the opposite it requires the gear ratio. If your happy with the scale press apply otherwise push cancel and enter the scale directly.

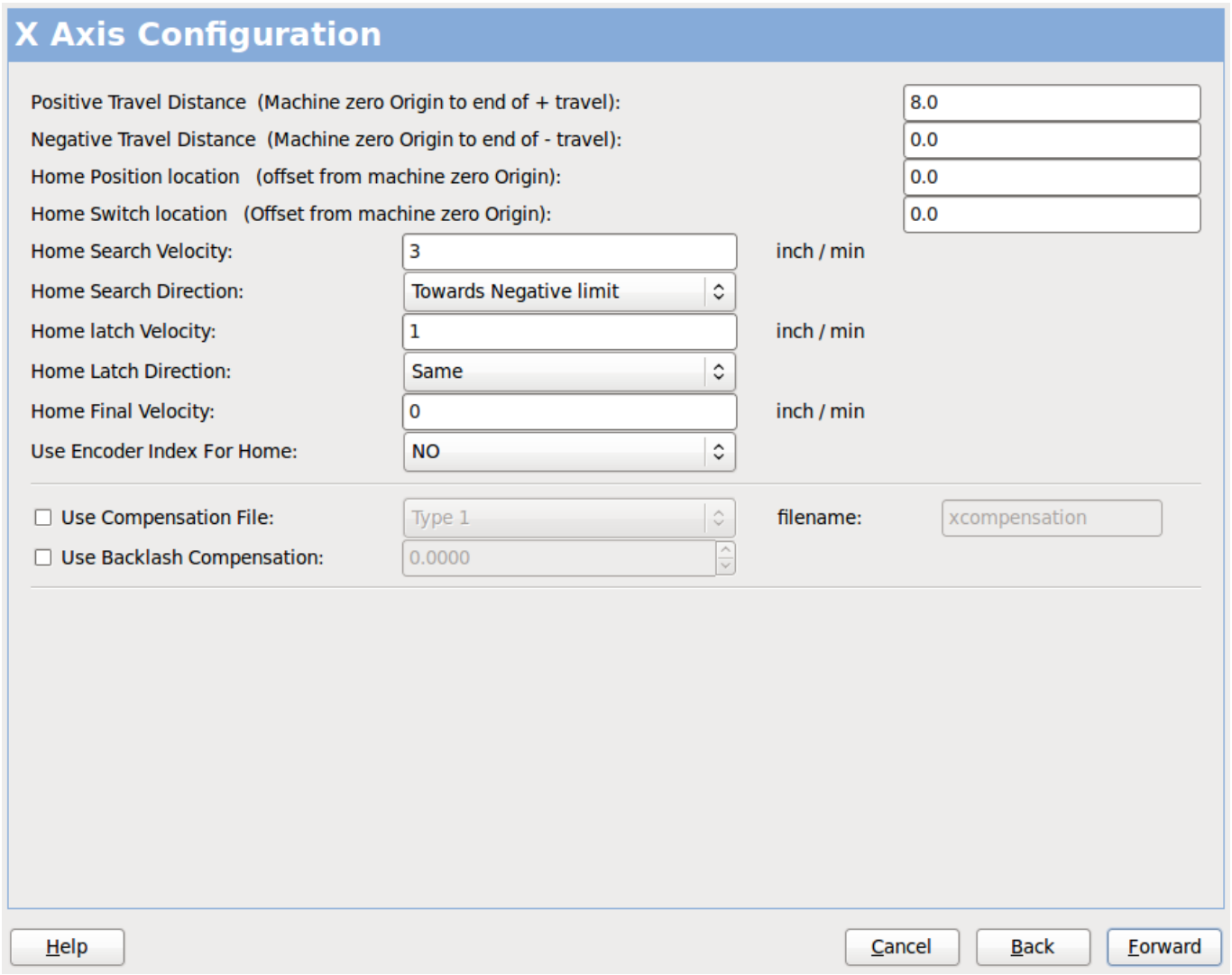



Abbildung 3.23: Konfiguration der Achsen

Also refer to the diagram tab for two examples of home and limit switches. These are two examples of many different ways to set homing and limits.

**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.

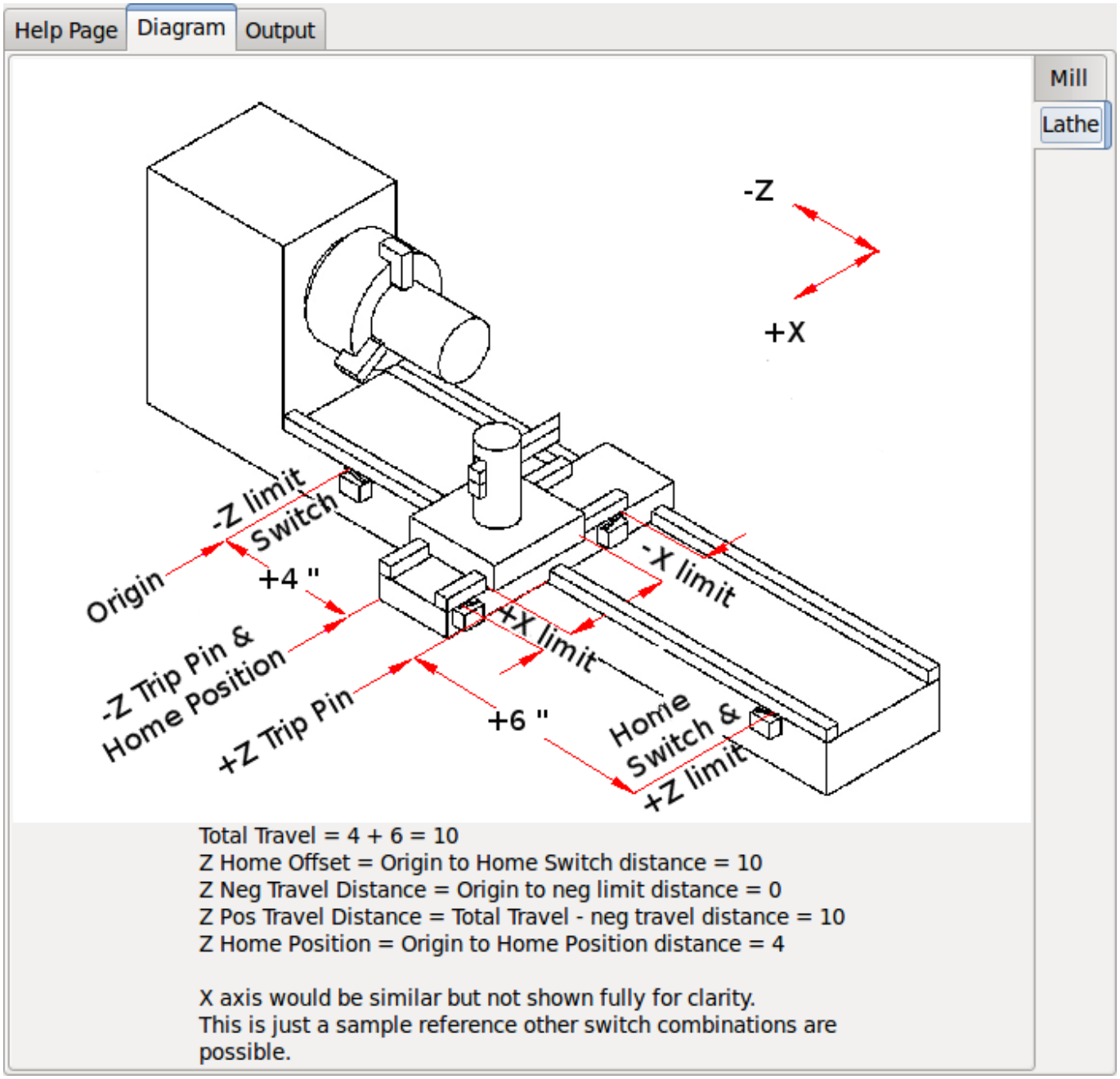



Abbildung 3.24: AXIS-Hilfsdiagramm

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!

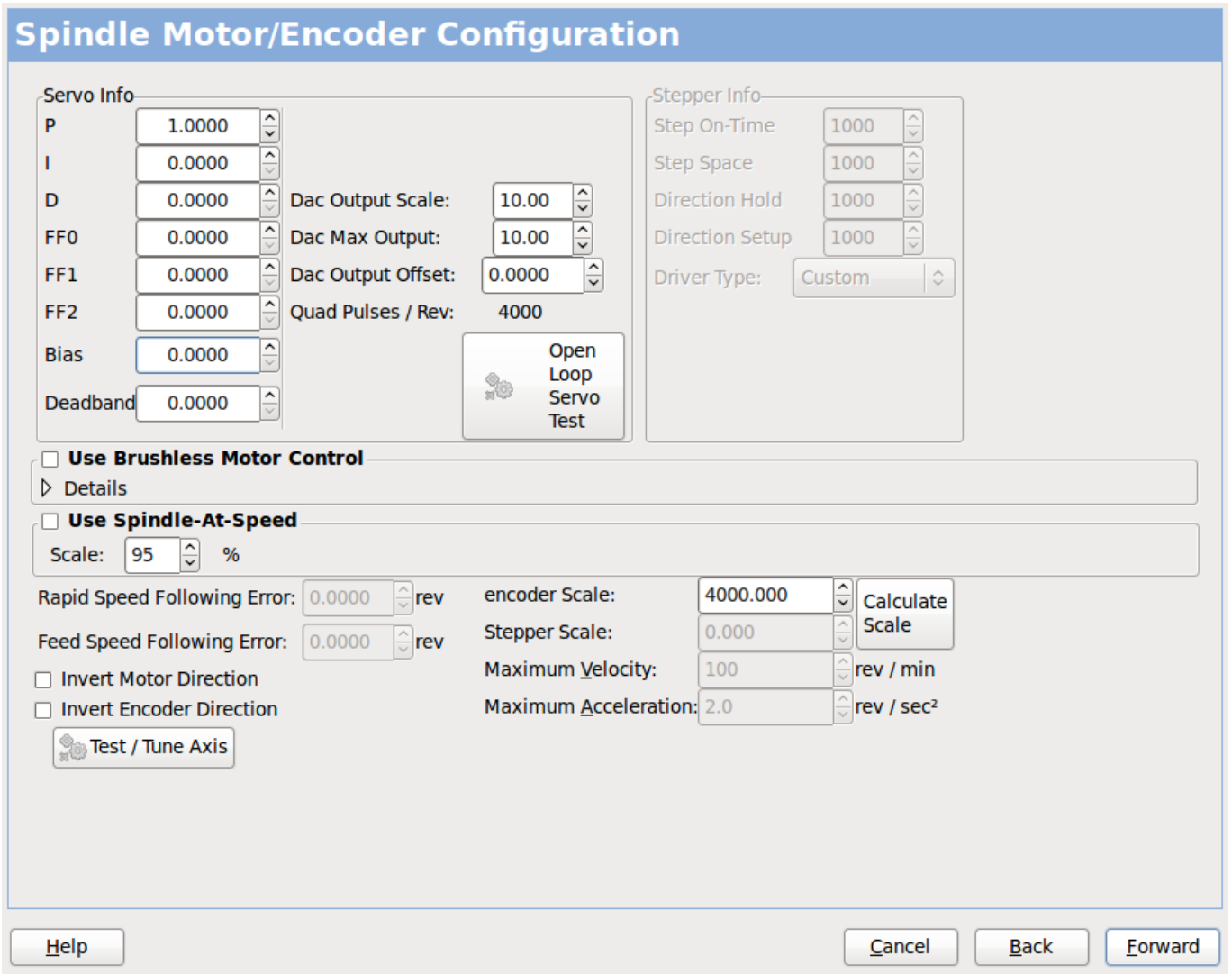



Abbildung 3.25: Spindelmotor/Encoder-Konfiguration

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.

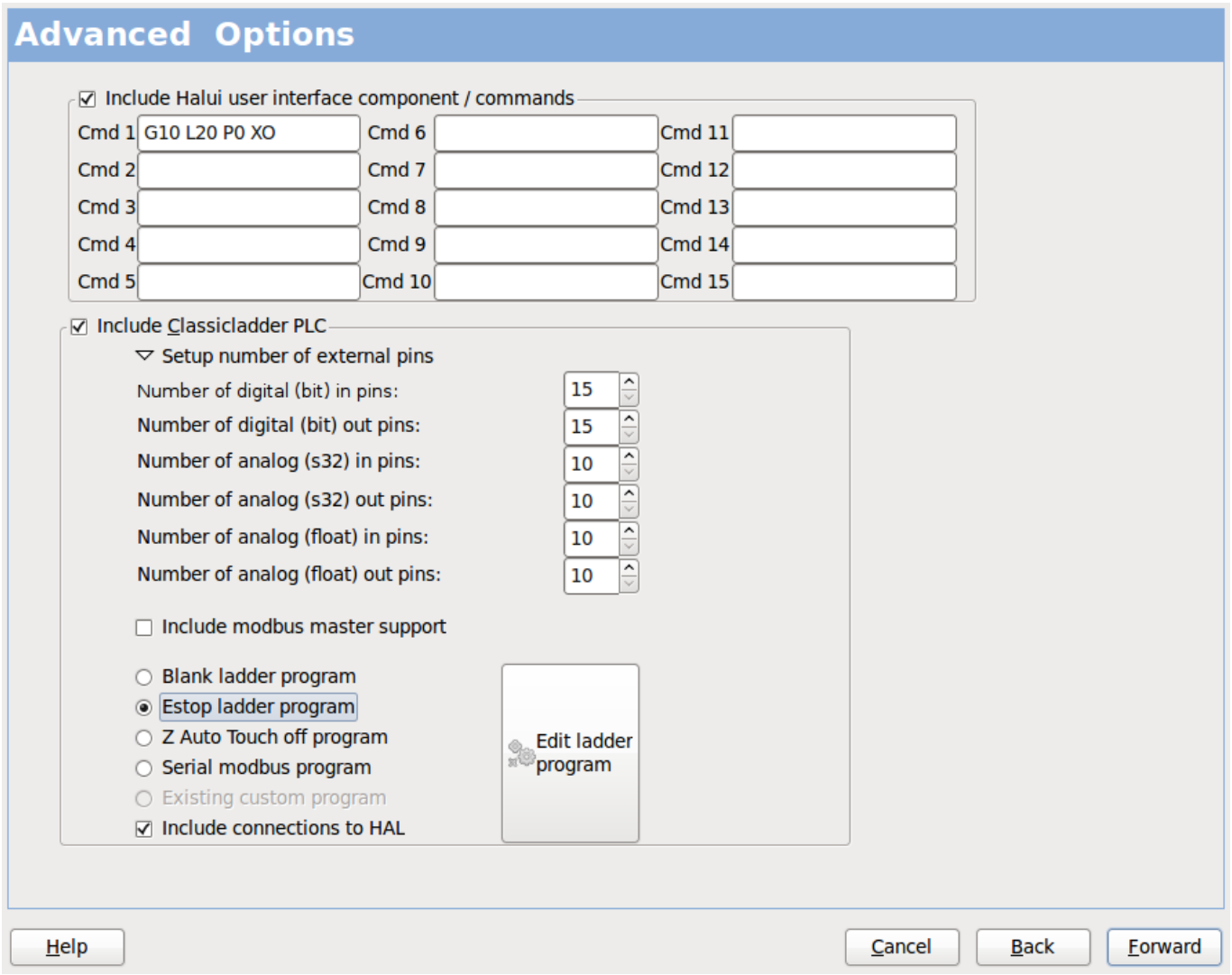



Abbildung 3.26: PnCconf, erweiterte Optionen

### 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.

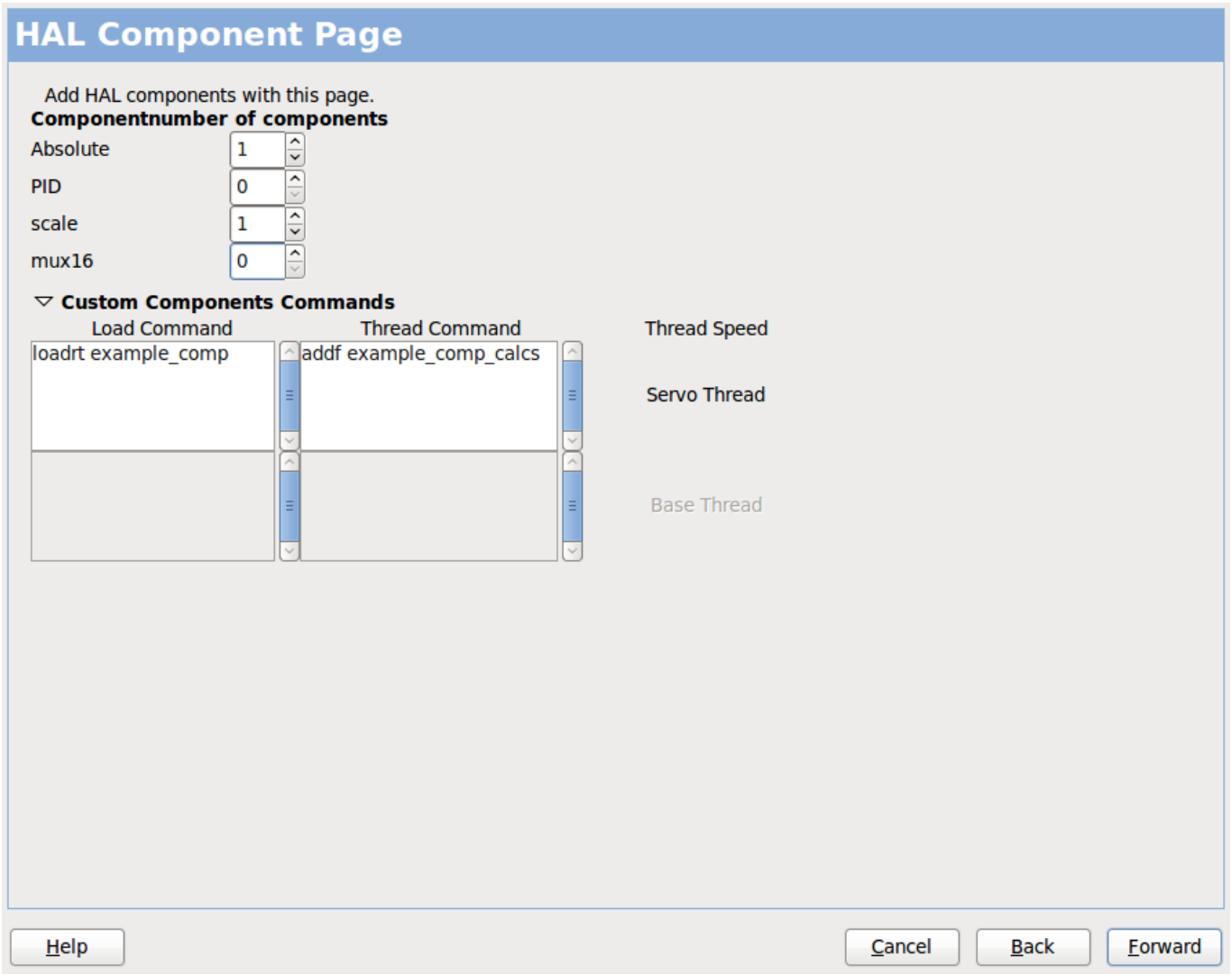



Abbildung 3.27: HAL-Komponenten

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

If your running LinuxCNC from the Live CD or you installed via a .deb and use the configuration picker *LinuxCNC* from the menu LinuxCNC looks in the following directories:

- 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.

The simplest way to avoid this problem is to choose a BASE_PERIOD that is the sum of the longest timing requirement of your drive, and the worst case latency of your computer. This is not always the best choice. For example, if you are running a drive with a 20 µs direction signal hold time requirement, and your latency test said you have a maximum latency of 11 µs , then if you set the BASE_PERIOD to 20+11 = 31 µs you get a not-so-nice 32,258 steps per second in one mode and 16,129 steps per second in another mode.

The problem is with the 20 µs hold time requirement. That plus the 11 µs latency is what forces us to use a slow 31 µs period. But the LinuxCNC software step generator has some parameters that let you increase the various times from one period to several. For example, if *steplen* [2] is changed from 1 to 2, then there will be two periods between the beginning and end of the step pulse. Likewise, if *dirhold* [3] is changed from 1 to 3, there will be at least three periods between the step pulse and a change of the direction pin.

If we can use *dirhold* to meet the 20 µs hold time requirement, then the next longest time is the 4.5 µs high time. Add the 11 µs latency to the 4.5 µs high time, and you get a minimum period of 15.5 µs . When you try 15.5 µs , you find that the computer is sluggish, so you settle on 16 µs . If we leave *dirhold* at 1 (the default), then the minimum time between step and direction is the 16 µs period minus the 11 µs latency = 5 µs , which is not enough. We need another 15 µs . Since the period is 16 µs , we need one more period. So we change *dirhold* from 1 to 2. Now the minimum time from the end of the step pulse to the changing direction pin is 5+16=21 µs , and we don't have to worry about the drive stepping the wrong direction because of latency.

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] steplen refers to a parameter that adjusts the performance of LinuxCNC's built-in step generator, *stepgen*, which is a HAL component. This parameter adjusts the length of the step pulse itself. Keep reading, all will be explained eventually.

[3] dirhold refers to a parameter that adjusts the length of the direction hold time.

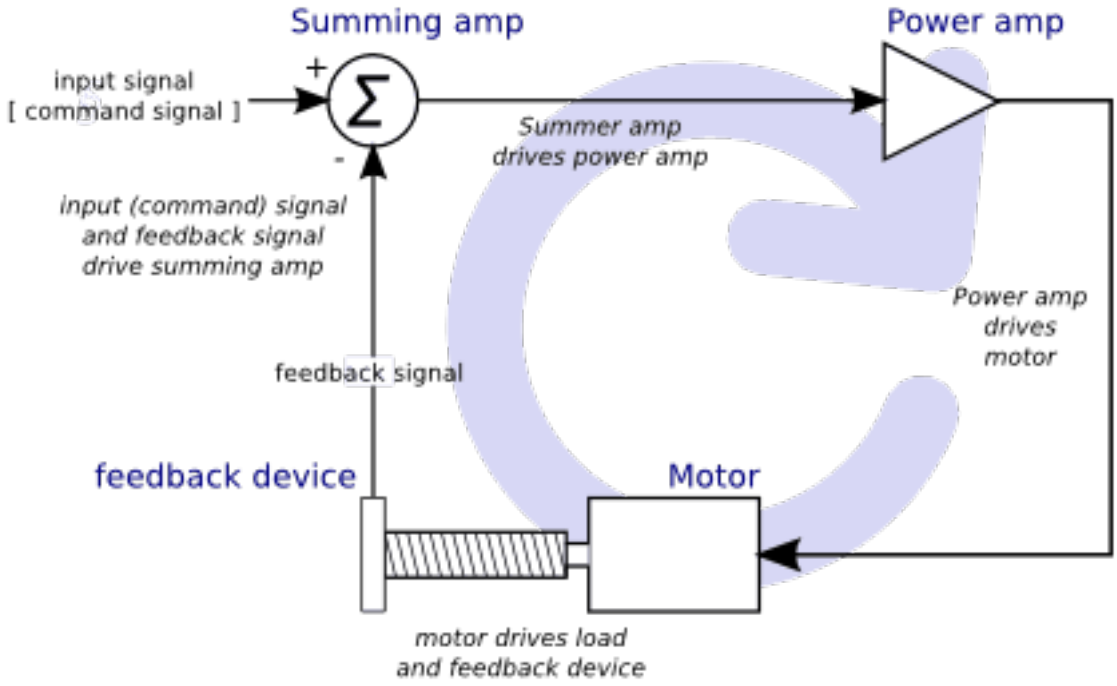



Abbildung 4.1: Servo Loop

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.

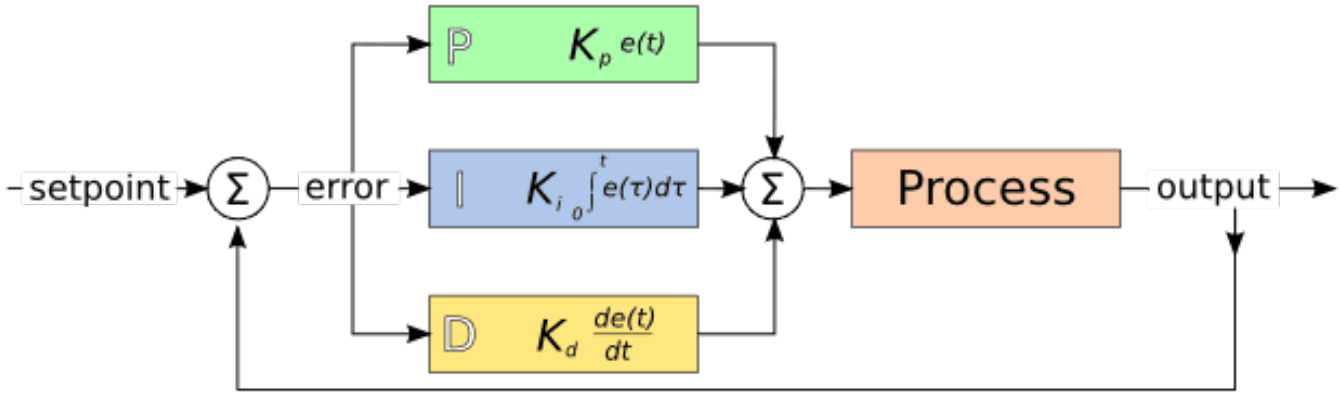



Abbildung 4.2: PID-Schleife

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.

### 4.1.6 Computer/Machine Interface Hardware Options

#### 4.1.6.1 litehm2/rv901t

Litehm2 is a board-agnostic port of the HostMot2 FPGA firmware. The first board it supports is the linsn rv901t, which was originally built as a LED controller board, but due to the available I/O it is well suited to act as a machine controller. It offers around 80 5V-buffered I/O ports and can switch between all input and all output. it is also easily modified to split the ports half/half between input and output. The rv901t interfaces to the computer via Gigabit or 100Mbit Ethernet.

Litehm2 is based on the LiteX framework which supports a wide range of FPGA boards. Currently only the rv901t is supported, but support for more boards is under development.

More information can be found at https://github.com/sensille/litehm2.

## 4.2 Latency Testing

### 4.2.1 What is latency?

Latency is how long it takes the PC to stop what it is doing and respond to an external request, such as running one of LinuxCNC's periodic realtime threads. The lower the latency, the faster you can run the realtime threads, and the smoother motion will be (and potentially faster, in the case of software stepping).

Die Latenzzeit ist viel wichtiger als die CPU-Geschwindigkeit. Ein bescheidener Pentium II, der auf Unterbrechungen jedes Mal innerhalb von 10 Mikrosekunden reagiert, kann bessere Ergebnisse liefern als das neueste und schnellste P4-Hyperthreading-Biest.

The CPU isn't the only factor in determining latency. Motherboards, video cards, USB ports, and a number of other things can hurt the latency. The best way to find out what you are dealing with is to run the latency test.

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

### 4.2.2 Latency Tests

LinuxCNC includes several latency tests. They all produce equivalent information. Running these tests will help determine if a computers is suitable for driving a CNC machine.

---

**Anmerkung**
Do not run LinuxCNC or StepConf while the latency test is running.

---

#### 4.2.2.1 Latenz-Test

To run the test, open a terminal window (in Ubuntu, from Applications → Accessories → Terminal) and run the following command:

```
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
```

After starting a latency test you should see something like this:

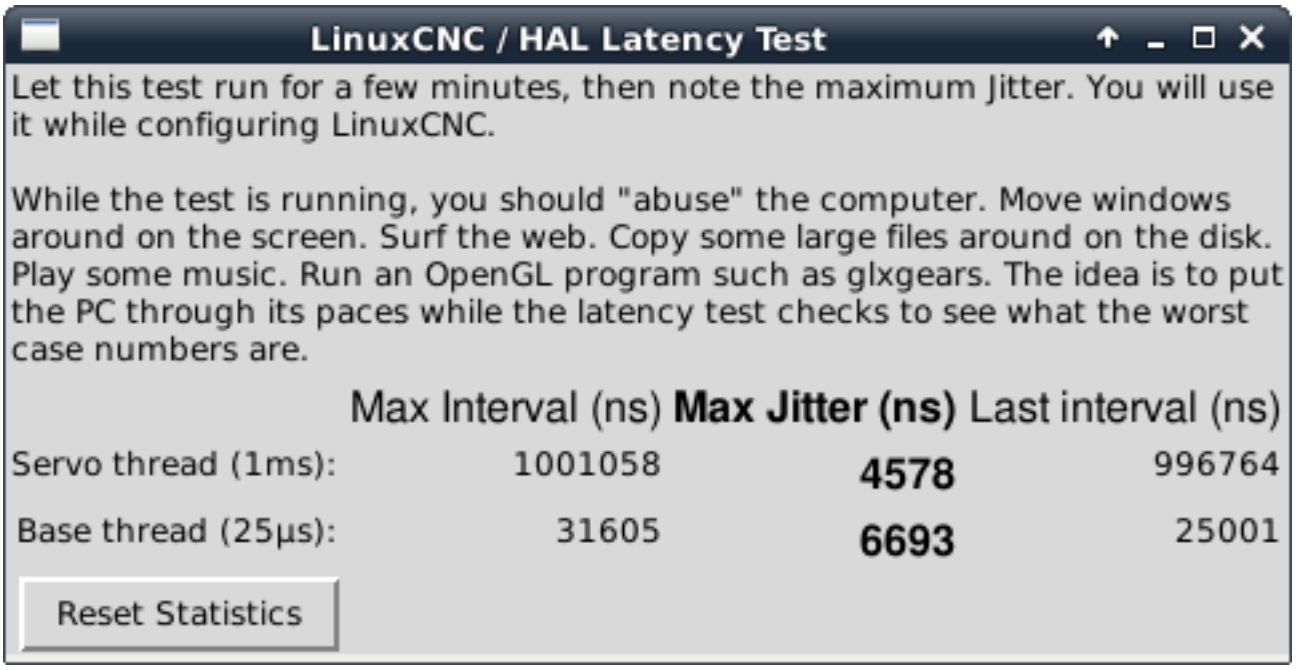



Abbildung 4.3: HAL-Latenz-Test

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.

The important number for software stepping is the *max jitter* of the base thread. In the example above, that is 6693 nanoseconds (ns), or 6.693 microseconds (µs). Record this number, and enter it in StepConf when it is requested.

In the example above, latency-test only ran for a few seconds. You should run the test for at least several minutes; sometimes the worst case latency doesn’t happen very often, or only happens when you do some particular action. For instance, one Intel motherboard worked pretty well most of the

time, but every 64 seconds it had a very bad 300 us latency. Fortunately that was fixable, see https://wiki.linux
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**
If you get high numbers, there may be ways to improve them. Another PC had very bad latency (several milliseconds) when using the onboard video. But a $5 used video card solved the problem. LinuxCNC does not require bleeding edge 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.**

#### 4.2.2.2 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)
```

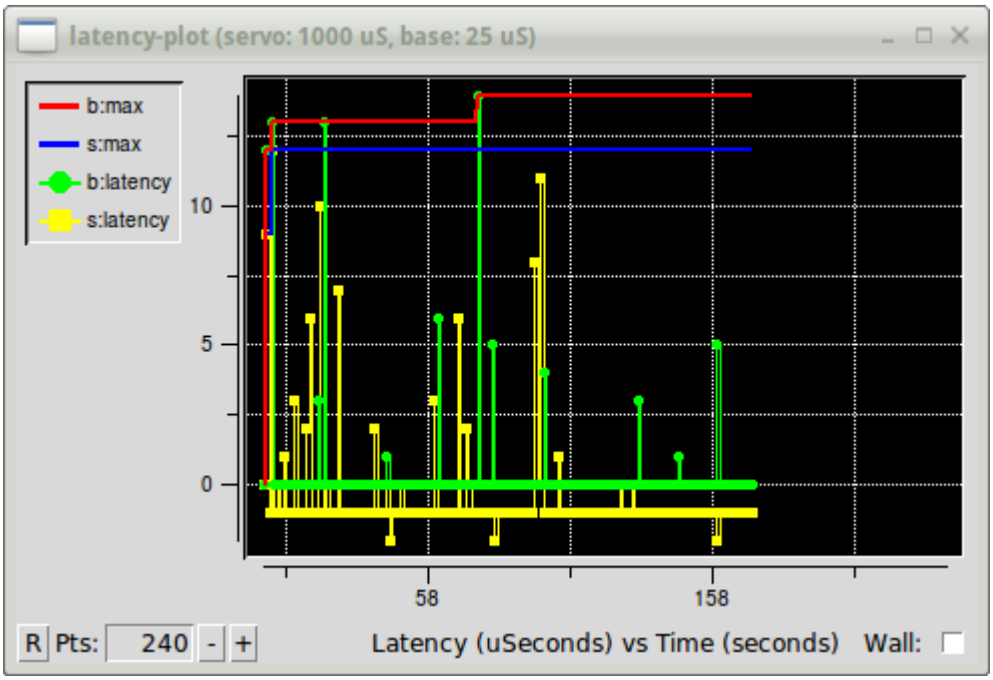



Abbildung 4.4: `latency-plot`-Fenster

#### 4.2.2.3 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
```

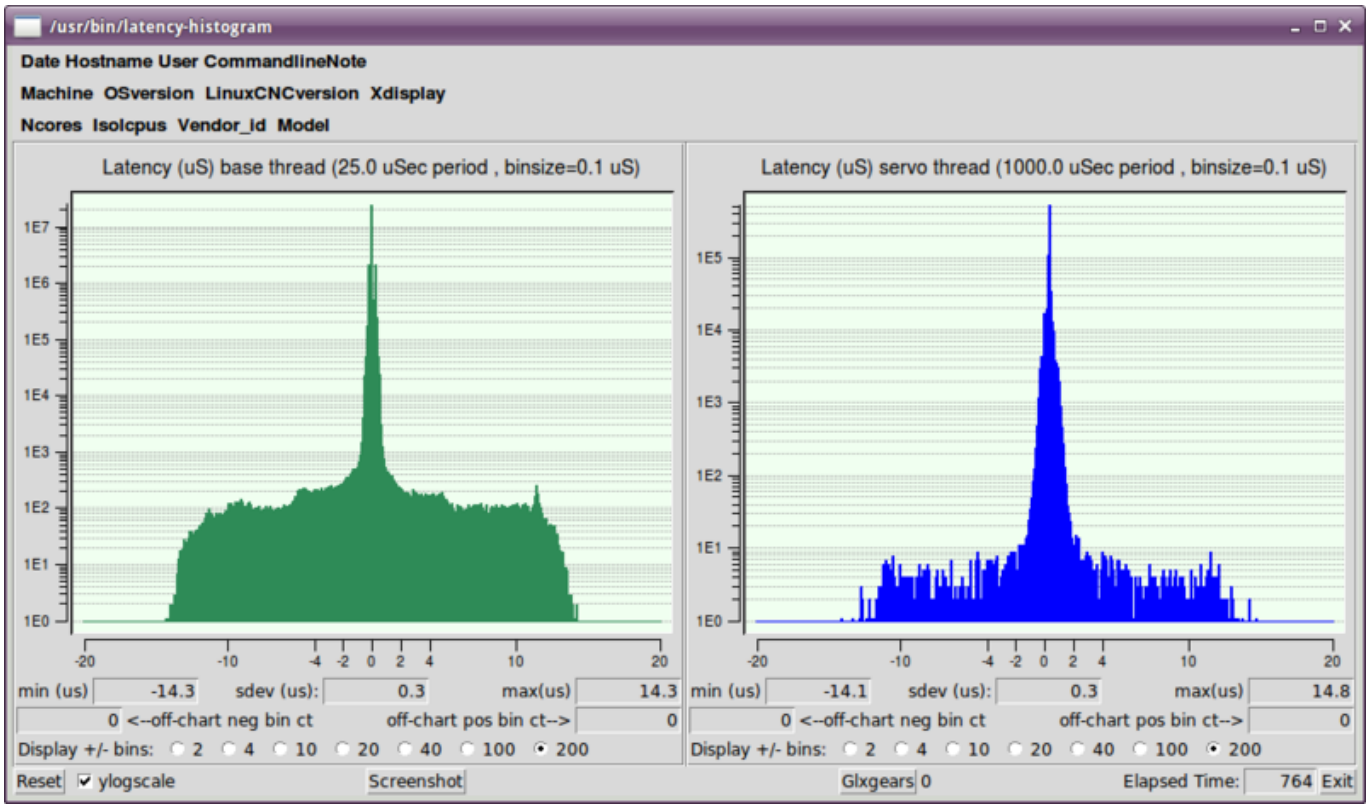



Abbildung 4.5: *Latenz-Histogramm*-Fenster

### 4.2.3 Latency tuning

LinuxCNC can run on many different hardware platforms and with many different realtime kernels, and they all may benefit from tuning for optimal latency.

A primary goal in tuning the system for LinuxCNC is to reserve a CPU for the exclusive use of LinuxCNC's realtime tasks, so that other tasks (both user programs and kernel threads) do not interfere with LinuxCNC's access to that CPU.

When specific tuning options are believed to be universally helpful LinuxCNC does this tuning automatically at startup, but many tuning options are machine-specific and cannot be done automatically. The person installing LinuxCNC will need to experimentally determine the optimal tuning for their system.

#### 4.2.3.1 Tuning the BIOS for latency

PC BIOSes vary wildly in their latency behavior.

Tuning the BIOS is tedious because you have to reboot the computer, make one small tweak in the BIOS, boot Linux, and run the latency test (potentially for a long time) to see what effects your BIOS change had. Then repeat for all the other BIOS settings you want to try.

Because BIOSes are all different and non-standard, providing a detailed BIOS tuning guide is not practical. In general, some things to try tuning in the BIOS are:

- Disable ACPI, APM, and any other power-saving features. This includes anything related to power saving, suspending, CPU sleep states, CPU frequency scaling, etc.

- Disable CPU ”turbo” mode.
- Disable CPU hyperthreading.
- Disable (or otherwise control) System Management Interrupt (SMI).
- Disable any hardware you do not intend to use.

#### 4.2.3.2 Tuning Preempt-RT for latency

The Preempt-RT kernel may benefit from tuning in order to provide the best latency for LinuxCNC. Tuning may be done via the kernel command line, sysctl, and via files in `/proc` and `/sys`.

Some tuning parameters to look into:

**Kernel command line**
Details here: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

- `isolcpus`: Prevent most non-LinuxCNC processes from using these CPUs, leaving more CPU time available for LinuxCNC.
- `irqaffinity`: Select which CPUs service interrupts, so that the CPUs reserved for LinuxCNC realtime don’t have to perform this task.
- `rcu_nocbs`: Prevent RCU callbacks from running on these CPUs.
- `rcu_nocb_poll`: Poll for RCU callbacks instead of using sleep/wake.
- `nohz_full`: Disable clock tick on these CPUs.

**Sysctl**
Details here: https://www.kernel.org/doc/html/latest/scheduler/sched-rt-group.html

- `sysctl.kernel.sched_rt_runtime_us`: Set to -1 to remove the limit on how much time real-time tasks may use.

## 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.

The last number in the column labeled *Max Jitter* is the most important. Write it down - you will need it later. It contains the worst latency measurement during the entire run of the test. In the example above, that is 6693 nano-seconds, or 6,69 micro-seconds, which is excellent. However the example only ran for a few seconds (it prints one line every second). You should run the test for at least several minutes; sometimes the worst case latency doesn’t happen very often, or only happens when you do some particular action. I had one Intel motherboard that worked pretty well most of the time, but every 64 seconds it had a very bad 300 µs latency. Fortunately that is fixable, see 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.

To make the math easier, I've created an Open Office spreadsheet Step Timing Calculator. You enter your latency test result and your stepper drive timing requirements and the spreadsheet calculates the optimum BASE_PERIOD. Next, you test the period to make sure it won't slow down or lock up your PC. Finally, you enter the actual period, and the spreadsheet will tell you the stepgen parameter settings that are needed to meet your drive's timing requirements. It also calculates the maximum step rate that you will be able to generate.

Ich habe der Tabelle ein paar Dinge hinzugefügt, um die maximale Geschwindigkeit und die elektrischen Berechnungen der Stepper zu berechnen.

## 4.4 INI Configuration

### 4.4.1 The INI File Components

A typical INI file follows a rather simple layout that includes;

- Kommentare
- Abschnitte
- Variablen

Each of these elements is separated on single lines. Each end of line or newline character creates a new element.

#### 4.4.1.1 Kommentare

A comment line is started with a ; or a # mark. When the INI reader sees either of these marks at the start a line, the rest of the line is ignored by the software. Comments can be used to describe what an INI element will do.

```
; Dies ist die Konfigurationsdatei meiner Fräsmaschine
# Ich habe sie am 12. Januar 2012 eingerichtet.
```

Comments can also be used to *turn off* a variable. This makes it easier to pick between different variables.

```
DISPLAY = axis
# DISPLAY = touchy
```

In this list, the DISPLAY variable will be set to axis because the other one is commented out. If someone carelessly edits a list like this and leaves two of the lines uncommented, the first one encountered will be used.

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 Sections

Related parts of an INI file are separated into sections. A section name is enclosed in brackets like this: `[THIS_SECTION]`. The order of sections is unimportant. Sections begin at the section name and end at the next section name.

Die folgenden Abschnitte werden von LinuxCNC verwendet:

- [EMC] general information
- [DISPLAY] settings related to the graphical user interface
- [FILTER] settings input filter programs
- [RS274NGC] settings used by the G-code interpreter
- [EMCMOT] settings used by the real time motion controller
- [TASK] settings used by the task controller
- [HAL] specifies .hal files
- [HALUI] MDI commands used by HALUI
- [APPLICATIONS] Other applications to be started by LinuxCNC
- [TRAJ] additional settings used by the real time motion controller
- [JOINT_n] individual joint variables
- [AXIS_l] individual axis variables
- [KINS] kinematics variables
- [EMCIO] settings used by the I/O Controller

#### 4.4.1.3 Variablen

A variable line is made up of a variable name, an equals sign (=), and a value. Everything from the first non-white space character after the = up to the end of the line is passed as the value, so you can embed spaces in string symbols if you want to or need to. A variable name is often called a keyword.

**Beispiel für eine Variable**

```
MACHINE = My Machine
```

A variable line may be extended to multiple lines with a terminal backslash (\) character. A maximum of `MAX_EXTEND_LINES` (==20) are allowed. There must be no whitespace following the trailing backslash character.

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
```

**boolsche Variablen** Boolesche Werte können eines von `TRUE`, `YES` oder `1` für wahr/aktiviert und eines von `FALSE`, `NO` oder `0` für falsch/deaktiviert sein. Der Fall wird ignoriert.

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

Variables that are used by LinuxCNC must always use the section names and variable names as shown.

#### 4.4.1.4 Custom Sections and Variables

Most sample configurations use custom sections and variables to put all of the settings into one location for convenience.

Um eine benutzerdefinierte Variable zu einem bestehenden LinuxCNC-Abschnitt hinzuzufügen, fügen Sie die Variable einfach in diesen Abschnitt ein.

**Custom Variable Example, assigning the value LINEAR to the variable TYPE, and the value 16000 to the variable SCALE.**

```
[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.

---

To use the custom variables in G-code, use the global variable syntax `#<_ini[section]variable>`. The following example shows a simple Z-axis touch-off routine for a router or mill using a probe plate.

**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 Files

An INI file may include the contents of another file by using a #INCLUDE directive.

**#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
```

The #INCLUDE directives are supported for one level of expansion only — an included file may not include additional files. The recommended file extension is *.inc*. Do *not* use a file extension of *.ini* for included files.

## 4.4.2 INI File Sections

### 4.4.2.1 [EMC] Section

* `VERSION = 1.1` - The version number for the configuration. Any value other than 1.1 will cause the configuration checker to run and try to update the configuration to the new style joint axes type of configuration. * `MACHINE = My Controller` - This is the name of the controller, which is printed out at the top of most graphical interfaces. You can put whatever you want here as long as you make it a single line long. * `DEBUG = 0` - Debug level 0 means no messages will be printed when LinuxCNC is run from a terminal. Debug flags are usually only useful to developers. See src/emc/nml_intf/debugflags.h for other settings.

### 4.4.2.2 [DISPLAY] Section

Different user interface programs use different options, and not every option is supported by every user interface. There are several interfaces, like AXIS, GMOCCAPY, Touchy, QtVCP's QtDragon and Gscreen. AXIS is an interface for use with normal computer and monitor, Touchy is for use with touch screens. GMOCCAPY can be used both ways and offers also many connections for hardware controls. Descriptions of the interfaces are in the Interfaces section of the User Manual.

- `DISPLAY = axis` - The file name of the executable providing the user interface to use. Prominent valid options are (all in lower case): `axis`, `touchy`, `gmoccapy`, `gscreen`, `tklinuxcnc`, `qtvcp`, `qtvcp-qtdragon` or `qtvcp-qtplasmac`.
- `POSITION_OFFSET = RELATIVE` - The coordinate system (`RELATIVE` or `MACHINE`) to show on the DRO when the user interface starts. The RELATIVE coordinate system reflects the G92 and G5*x* coordinate offsets currently in effect.
- `POSITION_FEEDBACK = COMMANDED` - The coordinate value (`COMMANDED` or `ACTUAL`) to show on the DRO when the user interface starts. In AXIS this can be changed from the View menu. The COMMANDED position is the position requested by LinuxCNC. The ACTUAL position is the feedback position of the motors if they have feedback like most servo systems. Typically the COMMANDED value is used.

- `DRO_FORMAT_MM = %+08.6f` - Override the default DRO formatting in metric mode (normally 3 decimal places, padded with spaces to 6 digits to the left). The example above will pad with zeros, display 6 decimal digits and force display of a + sign for positive numbers. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `DRO_FORMAT_IN = % 4.1f` - Override the default DRO formatting in imperial mode (normally 4 decimal places, padded with spaces to 6 digits to the left). The example above will display only one decimal digit. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `CONE_BASESIZE = .25` - Override the default cone/tool base size of .5 in the graphics display.
- `MAX_FEED_OVERRIDE = 1.2` - The maximum feed override the user may select. 1.2 means 120% of the programmed feed rate.
- `MIN_SPINDLE_OVERRIDE = 0.5` - The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.)
- `MIN_SPINDLE_0_OVERRIDE = 0.5` - The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.) On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `MAX_SPINDLE_OVERRIDE = 1.0` - The maximum spindle override the user may select. 1.0 means 100% of the programmed spindle speed.
- `MAX_SPINDLE_0_OVERRIDE = 1.0` - The maximum feed override the user may select. 1.2 means 120% of the programmed feed rate. On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `DEFAULT_SPINDLE_SPEED = 100` - The default spindle RPM when the spindle is started in manual mode. If this setting is not present, this defaults to 1 RPM for AXIS and 300 RPM for GMOCCAPY.

  - *veraltet* - stattdessen den Abschnitt [SPINDLE_n] verwenden
- `DEFAULT_SPINDLE_0_SPEED = 100` - The default spindle RPM when the spindle is started in manual mode. On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP-based user interfaces.

  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `SPINDLE_INCREMENT = 200` - The increment used when clicking increase/decrease buttons. Only used by the QtVCP based user interfaces.

  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `MIN_SPINDLE_0_SPEED = 1000` - The minimum RPM that can be manually selected. On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP-based user interfaces.

  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.
- `MAX_SPINDLE_0_SPEED = 20000` - The maximum RPM that can be manually selected. On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP-based user interfaces.

  - *deprecated* - stattdessen den Abschnitt [SPINDLE_n] verwenden.

- `PROGRAM_PREFIX = ~/linuxcnc/nc_files` - The default directory for G-code files, named subroutines, and user-defined M-codes. The `PROGRAM_PREFIX` directory is searched before the directories listed in `[RS274]SUBROUTINE_PATH` and `[RS274]USER_M_PATH`.
- `INTRO_GRAPHIC = emc2.gif` - The image shown on the splash screen.
- `INTRO_TIME = 5` - The maximum time to show the splash screen, in seconds.
- `CYCLE_TIME = 100` - Cycle time of the display GUI. Depending on the screen, this can be in seconds or ms (ms preferred). This is often the update rate rather then sleep time between updates. If the update time is not set right the screen can become unresponsive or very jerky. A value of 100 ms (0.1 s) is a common setting though a range of 50 - 200 ms (.05 - .2 s) may be useable. An under powered CPU may see improvement with a longer setting. Usually the default is fine.
- `PREVIEW_TIMEOUT = 5` - Timeout (in seconds) for loading graphical preview of G-code. Currently AXIS only.

---

**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 ./gla ./gladevcp/manual-example.ui`

---

**Anmerkung**
Different user interface programs use different options, and not every option is supported by every user interface. See AXIS GUI document for AXIS details. See GMOCCAPY document for GMOCCAPY details.

---

- `DEFAULT_LINEAR_VELOCITY = .25` - The default velocity for linear jogs, in machine units per second.
- `MIN_VELOCITY = .01` - The approximate lowest value the jog slider.
- `MAX_LINEAR_VELOCITY = 1.0` - The maximum velocity for linear jogs, in machine units per second.
- `MIN_LINEAR_VELOCITY = .01` - The approximate lowest value the jog slider.
- `DEFAULT_ANGULAR_VELOCITY = .25` - The default velocity for angular jogs, in machine units per second.
- `MIN_ANGULAR_VELOCITY = .01` - The approximate lowest value the angular jog slider.
- `MAX_ANGULAR_VELOCITY = 1.0` - The maximum velocity for angular jogs, in machine units per second.
- `INCREMENTS = 1 mm, .5 in, ...` - Defines the increments available for incremental jogs. The INCREMENTS can be used to override the default. The values can be decimal numbers (e.g., 0.1000) or fractional numbers (e.g., 1/16), optionally followed by a unit (cm, mm, um, inch, in or mil). If a unit is not specified the machine unit is assumed. Metric and imperial distances may be mixed: INCREMENTS = 1 inch, 1 mil, 1 cm, 1 mm, 1 um is a valid entry.
- `GRIDS = 10 mm, 1 in, ...` - Defines the preset values for grid lines. The value is interpreted the same way as `INCREMENTS`.
- `OPEN_FILE = /full/path/to/file.ngc` - The file to show in the preview plot when AXIS starts. Use a blank string "" and no file will be loaded at start up. GMOCCAPY will not use this setting, as it offers a corresponding entry on its settings page.

- `EDITOR = gedit` - The editor to use when selecting File > Edit to edit the G-code from the AXIS menu. This must be configured for this menu item to work. Another valid entry is `gnome-terminal -e vim`. This entry does not apply to GMOCCAPY, as GMOCCAPY has an integrated editor.
- `TOOL_EDITOR = tooledit` - The editor to use when editing the tool table (for example by selecting "File > Edit tool table..." in AXIS). Other valid entries are `gedit`, `gnome-terminal -e vim`, and `gvim`. This entry does not apply to GMOCCAPY, as GMOCCAPY has an integrated editor.
- `PYVCP = /filename.xml` - The PyVCP panel description file. See the PyVCP Chapterfor more information.
- `PYVCP_POSITION = BOTTOM` - The placement of the PyVCP panel in the AXIS user interface. If this variable is omitted the panel will default to the right side. The only valid alternative is `BOTTOM`. See the PyVCP Chapter for more information.
- `LATHE = 1` - Any non-empty value (including "0") causes axis to use "lathe mode" with a top view and with Radius and Diameter on the DRO.
- `BACK_TOOL_LATHE = 1` - Any non-empty value (including "0") causes axis to use "back tool lathe mode" with inverted X axis.
- `FOAM = 1` - Any non-empty value (including "0") causes axis to change the display for foam-cutter mode.
- `GEOMETRY = XYZABCUVW` - Controls the **preview** and **backplot** of motion. This item consists of a sequence of axis letters and control characters, optionally preceded with a "-" sign:

  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. Each letter specified must occur in `[TRAJ]COORDINATES` to have an effect.
  5. Ein "**-**" Zeichen vor einem beliebigen Buchstaben kehrt die Richtung der Operation um.
  6. The translation and rotation operations are evaluated **right-to-left**. So using `GEOMETRY=XYZBC` specifies a C rotation followed by a B rotation followed by Z, Y, X translations. The ordering of consecutive translation letters is immaterial.
  7. The proper GEOMETRY string depends on the machine configuration and the kinematics used to control it. The order of the letters is important. For example, rotating around C then B is different than rotating around B then C.
  8. Rotations are by default applied with respect to the machine origin. Example: `GEOMETRY=CXYZ` first translates the control point to X, Y, Z and then performs a C rotation about the Z axis centered at the machine origin.
  9. UVW translation example: `GEOMETRY=XYZUVW` causes UVW to move in the coordinate system of the tool and XYZ to move in the coordinate system of the material.
  10. Foam-cutting machines (`FOAM = 1`) should specify "XY;UV" or leave the value blank even though this value is presently ignored in foam-cutter mode. A future version may define what ";" means, but if it does "XY;UV" will mean the same as the current foam default.
  11. Experimental: If the exclamation mark (`!`) character is included in the GEOMETRY string, display points for A, B, C rotations respect the X, Y, Z offsets set by G5x, G92 codes. Example: Using `GEOMETRY = !CXZ` for a machine with `[TRAJ]COORDINATES=XZC`. This provision applies for liveplots only — G-code previews should be done with zero G5x, G92 offsets. This can be facilitated by setting offsets in programs only when task is running as indicated by `#<_task> == 1`. If nonzero offsets exist at startup due to persistence, offsets should be zeroed and preview reloaded.

---

**Anmerkung**
If no `[DISPLAY]GEOMETRY` is included in the INI file, a default is provided by the `[DISPLAY]DISPLAY` GUI program (typically "XYZABCUVW").

---

- ARCDIVISION = 64 - Set the quality of preview of arcs. Arcs are previewed by dividing them into a number of straight lines; a semicircle is divided into **ARCDIVISION** parts. Larger values give a more accurate preview, but take longer to load and result in a more sluggish display. Smaller values give a less accurate preview, but take less time to load and may result in a faster display. The default value of 64 means a circle of up to 3 inches will be displayed to within 1 mil (.03%).
- MDI_HISTORY_FILE = - The name of a local MDI history file. If this is not specified, AXIS will save the MDI history in **.axis_mdi_history** in the user`s home directory. This is useful if you have multiple configurations on one computer.
- JOG_AXES = - The order in which jog keys are assigned to axis letters. The left and right arrows are assigned to the first axis letter, up and down to the second, page up/page down to the third, and left and right bracket to the fourth. If unspecified, the default is determined from the [TRAJ]COORDINATES, [DISPLAY]LATHE and [DISPLAY]FOAM values.
- JOG_INVERT = - For each axis letter, the jog direction is inverted. The default is ”X” for lathes and blank otherwise.

---

**Anmerkung**
The settings for JOG_AXES and JOG_INVERT apply to world mode jogging by axis coordinate letter and are in effect while in world mode after successful homing. When operating in joint mode prior to homing, keyboard jog keys are assigned in a fixed sequence: left/right: joint0, up/down: joint1, page up/page down: joint2, left/right bracket: joint3

---

- 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**
The following [DISPLAY] item is used by the TKLinuxCNC interface only.

---

- HELP_FILE = tklinucnc.txt - Path to help file.

#### 4.4.2.3 [FILTER] Section

AXIS and GMOCCAPY have the ability to send loaded files through a filter program. This filter can do any desired task: Something as simple as making sure the file ends with M2, or something as complicated as detecting whether the input is a depth image, and generating G-code to mill the shape it defines. The [FILTER] section of the INI file controls how filters work. First, for each type of file, write a PROGRAM_EXTENSION-line. Then, specify the program to execute for each type of file. This program is given the name of the input file as its first argument, and must write RS274NGC code to standard output. This output is what will be displayed in the text area, previewed in the display area, and executed by LinuxCNC when Run.

- 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
```

In this way, any Python script can be opened, and its output is treated as G-code. One such example script is available at nc_files/holecircle.py. This script creates G-code for drilling a series of holes along the circumference of a circle. Many more G-code generators are on the LinuxCNC Wiki site https://wiki.linuxcnc.org/.

Python filters should use the print function to output the result to AXIS.

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] Section

* PARAMETER_FILE = myfile.var - The file located in the same directory as the INI file which contains the parameters used by the interpreter (saved between runs). * ORIENT_OFFSET = 0 - A float value added to the R word parameter of an M19 Orient Spindle operation. Used to define an arbitrary zero position regardless of encoder mount orientation. * RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92.1 G94 G97 G98 - A string of NC codes that the interpreter is initialized with. This is not a substitute for specifying modal G-codes at the top of each NGC file, because the modal codes of machines differ, and may be changed by G-code interpreted earlier in the session. * SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs - Specifies a colon (:) separated list of up to 10 directories to be searched when single-file subroutines are specified in G-code. These directories are searched after searching [DISPLAY]PROGRAM_PREFIX (if it is specified) and before searching [WIZARD]WIZARD_ROOT (if specified). The paths are searched in the order that they are listed. The first matching subroutine file found in the search is used. Directories are specified relative to the current directory for the INI file or as absolute paths. The list must contain no intervening whitespace. * CENTER_ARC_RADIUS_TOLERANCE_INCH = *n* (Default: 0.00005) * CENTER_ARC_RADIUS_TOLERANCE_MM = *n* (Default: 0.00127) * USER_M_PATH = myfuncs:/tmp/mcodes:exper
- Specifies a list of colon (:) separated directories for user defined functions. Directories are specified relative to the current directory for the INI file or as absolute paths. The list must contain no intervening whitespace.

+ Es wird nach jeder möglichen benutzerdefinierten Funktion gesucht, typischerweise (M100-M199). Die Reihenfolge der Suche ist:

+ . [DISPLAY]PROGRAM_PREFIX (if specified) . If [DISPLAY]PROGRAM_PREFIX is not specified, search the default location: nc_files . Then search each directory in the list [RS274NGC]USER_M_PATH.

+ The first executable M1*xx* found in the search is used for each M1*xx*.

+

---

**Anmerkung**
The maximum number of USER_M_PATH directories is defined at compile time (typ: USER_DEFINED_FUNCTION_MAX_DIRS == 5).

---

- INI_VARS = 1 (Default: 1)
  Allows G-code programs to read values from the INI file using the format #<_ini[section]name>. See G-code Parameters.
- HAL_PIN_VARS = 1 (Default: 1)
  Allows G-code programs to read the values of HAL pins using the format #<_hal[HAL item]>. Variable access is read-only. See G-code Parameters for more details and an important caveat.
- RETAIN_G43 = 0 (Default: 0)
  When set, you can turn on G43 after loading the first tool, and then not worry about it through the program. When you finally unload the last tool, G43 mode is canceled.
- OWORD_NARGS = 0 (Default: 0)
  If this feature is enabled then a called subroutine can determine the number of actual positional parameters passed by inspecting the #<n_args> parameter.
- NO_DOWNCASE_OWORD = 0 (Default: 0)
  Preserve case in O-word names within comments if set, enables reading of mixed-case HAL items in structured comments like (debug, #<_hal[MixedCaseItem]).

- `OWORD_WARNONLY = 0` (Default: 0)
  Warn rather than error in case of errors in O-word subroutines.
- `DISABLE_G92_PERSISTENCE = 0` (Default: 0) Allow to clear the G92 offset automatically when config start-up.
- `DISABLE_FANUC_STYLE_SUB = 0` (Default: 0) If there is reason to disable Fanuc subroutines set it to 1.

---

**Anmerkung**
The above six options were controlled by the FEATURES bitmask in versions of LinuxCNC prior to 2.8. This INI tag will no longer work.
For reference:

```
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` is a valid search path but the Wizard has not been fully implemented and the results of using it are unpredictable.

---

- `LOG_LEVEL = 0` Specify the log_level (default: 0)
- `LOG_FILE = file-name.log`
  For specify the file used for log the data.
- `REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure` See Remap Extending G-code chapter for details.
- `ON_ABORT_COMMAND=O <on_abort> call` See Remap Extending G-code chapter for details.

#### 4.4.2.5 [EMCMOT] Section

This section is a custom section and is not used by LinuxCNC directly. Most configurations use values from this section to load the motion controller. For more information on the motion controller see the Motion section.

- `EMCMOT = motmod` - the motion controller name is typically used here.
- `BASE_PERIOD = 50000` - the *Base* task period in nanoseconds.
- `SERVO_PERIOD = 1000000` - This is the ”Servo” task period in nanoseconds.
- `TRAJ_PERIOD = 100000` - This is the *Trajectory Planner* task period in nanoseconds.
- `COMM_TIMEOUT = 1.0` - Number of seconds to wait for Motion (the realtime part of the motion controller) to acknowledge receipt of messages from Task (the non-realtime part of the motion controller).
- `HOMEMOD` = *alternate_homing_module* [home_parms=value] The HOMEMOD variable is optional. If specified, use a specified (user-built) module instead of the default (homemod). Module parameters (home_parms) may be included if supported by the named module. The setting may be overridden from the command line using the -m option ($ linuxcnc -h).

#### 4.4.2.6 [TASK] Section

* `TASK = milltask` - Specifies the name of the *task* executable. The *task* executable does various things, such as - communicate with the UIs over NML, - communicate with the realtime motion planner over non-HAL shared memory, and - interpret G-code. Currently there is only one task executable that makes sense for 99.9% of users, milltask. * `CYCLE_TIME = 0.010` - The period, in seconds, at which TASK will run. This parameter affects the polling interval when waiting for motion to complete, when executing a pause instruction, and when accepting a command from a user interface. There is usually no need to change this number.

#### 4.4.2.7 [HAL] section

* `HALFILE = example.hal` - Execute the file *example.hal* at start up.

+

If `HALFILE` is specified multiple times, the files are interpreted in the order they appear in the INI file. HAL files are descriptive, the execution of what is described in HAL files is triggered by the threads in which functions are embedded, not by the reading of the HAL file. Almost all configurations will have at least one `HALFILE`, and stepper systems typically have two such files, i.e., one which specifies the generic stepper configuration (*core_stepper.hal*) and one which specifies the machine pin out (*xxx_pinout.hal*).

HAL files specified in the `HALFILES` variable are found using a search. If the named file is found in the directory containing the INI file, it is used. If the named file is not found in this INI file directory, a search is made using a system library of HAL files.

If LinuxCNC is started with the `linuxcnc` script using the "`-H` *dirname*" option, the specified dirname is prepended to the search described above so that *dirname* is searched first. The "`-H` *dirname*" option may be specified more than once, directories are prepended in order.

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*] ...] - Execute the tcl file *texample.tcl* at start up with arg1, arg2, etc. as argv list. Files with a .tcl suffix are processed as above but use haltcl for processing. See the HALTCL Chapter for more information.
- `HALFILE = LIB:sys_example.hal` - Execute the system library file *sys_example.hal* at start up. Explicit use of the LIB: prefix causes use of the system library HALFILE without searching the INI file directory.
- `HALFILE = LIB:sys_texample.tcl` [*arg1* [*arg2* ...]] - Execute the system library file *sys_texample.tcl* at start up. Explicit use of the LIB: prefix causes use of the system library HALFILE without searching the INI file directory.

HALFILE items specify files that loadrt HAL components and make signal connections between component pins. Common mistakes are

1. omission of the addf statement needed to add a component's function(s) to a thread,
2. incomplete signal (net) specifiers.

Omission of required addf statements is almost always an error. Signals usually include one or more input connections and a single output (but both are not strictly required). A system library file is provided to make checks for these conditions and report to stdout and in a pop-up GUI:

```
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` - Use twopass processing for loading HAL components. With TWOPASS processing, lines of files specified in `[HAL]HALFILE` are processed in two passes. In the first pass (pass0), all HALFILES are read and multiple appearances of loadrt and loadusr commands are accumulated. These accumulated load commands are executed at the end of pass0. This accumulation allows load lines to be specified more than once for a given component (provided the names= names used are unique on each use). In the second pass (pass1), the HALFILES are reread and all commands except the previously executed load commands are executed.
- `TWOPASS = nodelete verbose` - The `TWOPASS` feature can be activated with any non-null string including the keywords verbose and nodelete. The verbose keyword causes printing of details to stdout. The nodelete keyword preserves temporary files in /tmp.

Weitere Informationen finden Sie im Kapitel HAL TWOPASS.

- `HALCMD =` *command* - Execute *command* as a single HAL command. If `HALCMD` is specified multiple times, the commands are executed in the order they appear in the INI file. `HALCMD`-lines are executed after all `HALFILE`-lines.
- `SHUTDOWN = shutdown.hal` - Execute the file *shutdown.hal* when LinuxCNC is exiting. Depending on the hardware drivers used, this may make it possible to set outputs to defined values when LinuxCNC is exited normally. However, because there is no guarantee this file will be executed (for instance, in the case of a computer crash), it is not a replacement for a proper physical e-stop chain or other protections against software failure.
- `POSTGUI_HALFILE = example2.hal` - Execute *example2.hal* after the GUI has created its HAL pins. Some GUIs create HAL pins and support the use of a postgui halfile to use them. GUIs that support postgui HAL files include Touchy, AXIS, Gscreen, and GMOCCAPY.
  See section PyVCP with AXIS for more information.
- `HALUI = halui` - adds the HAL user interface pins.
  For more information see the HAL User Interface chapter.

#### 4.4.2.8 [HALUI] section

* `MDI_COMMAND = G53 G0 X0 Y0 Z0` - An MDI command can be executed by using `halui.mdi-command-00`. Increment the number for each command listed in the [HALUI] section.

#### 4.4.2.9 [APPLICATIONS] Section

LinuxCNC can start other applications before the specified GUI is started. The applications can be started after a specified delay to allow for GUI-dependent actions (like creating GUI-specific HAL pins).

- `DELAY =` *value* - seconds to wait before starting other applications. A delay may be needed if an application has dependencies on `[HAL]POSTGUI_HALFILE` actions or GUI-created HAL pins (default `DELAY=0`).

- ‚APP =` *appname* [arg1 [arg2 ...]]‘ - Application to be started. This specification can be included multiple times. The appname can be explicitly named as an absolute or tilde specified filename (first character is / or ~), a relative filename (first characters of filename are ./), or as a file in the INI file directory. If no executable file is found using these names, then the user search PATH is used to find the application.
  Examples:
  - 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 [TRAJ] Section

**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` - Turn on new TP. If set to 0 TP uses parabolic blending (1 segment look ahead) (Default: 1).
- `ARC_BLEND_FALLBACK_ENABLE = 0` - Optionally fall back to parabolic blends if the estimated speed is faster. However, this estimate is rough, and it seems that just disabling it gives better performance (Default: 0).
- `ARC_BLEND_OPTIMIZATION_DEPTH = 50` - Look ahead depth in number of segments.
  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, a machine with a maximum axis velocity of 10 IPS, a max acceleration of 100 $IPS^2$, and a servo period of 0.001 s would need:

10 / (2,0 * 100 * 0,001) = 50 Segmente, um immer die maximale Geschwindigkeit entlang der schnellsten Achse zu erreichen.

In practice, this number isn't that important to tune, since the look ahead rarely needs the full depth unless you have lots of very short segments. If during testing, you notice strange slowdowns and can't figure out where they come from, first try increasing this depth using the formula above.

If you still see strange slowdowns, it may be because you have short segments in the program. If this is the case, try adding a small tolerance for Naive CAM detection. A good rule of thumb is this:

```
# min_length ~= v_req * t_c
# wobei:
# v_req = gewünschte Geschwindigkeit in UU / sec
# t_c = Servoperiode (Sekunden)
```

If you want to travel along a path at 1 IPS = 60 IPM, and your servo period is 0.001 s, then any segments shorter than min_length will slow the path down. If you set Naive CAM tolerance to around this min length, overly short segments will be combined together to eliminate this bottleneck. Of course, setting the tolerance too high means big path deviations, so you have to play with it a bit to find a good value. I'd start at 1/2 of the min_length, then work up as needed. * `ARC_BLEND_GAP_CYCLES = 4` How short the previous segment must be before the trajectory planner *consumes* it.

Often, a circular arc blend will leave short line segments in between the blends. Since the geometry has to be circular, we can't blend over all of a line if the next one is a little shorter. Since the trajectory planner has to touch each segment at least once, it means that very tiny segments will slow things down significantly. My fix to this way to "consume" the short segment by making it a part of the blend arc. Since the line+blend is one segment, we don't have to slow down to hit the very short segment. Likely, you won't need to touch this setting. * `ARC_BLEND_RAMP_FREQ = 20` - This is a *cutoff* frequency for using ramped velocity.

*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.

Basically, if a segment will complete in less time than 1 / ARC_BLEND_RAMP_FREQ, we don't bother with a trapezoidal velocity profile on that segment, and use constant acceleration. (Setting `ARC_BLEND_RAMP_FREQ = 1000` is equivalent to always using trapezoidal acceleration, if the servo loop is 1 kHz).

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
```

For the aforementioned machine, the ripple for a 20 Hz cutoff frequency is 100 / (4 * 20) = 1.25 IPS. This seems high, but keep in mind that it is only a worst-case estimate. In reality, the trapezoidal motion profile is limited by other factors, such as normal acceleration or requested velocity, and so

the actual performance loss should be much smaller. Increasing the cutoff frequency can squeeze out more performance, but make the motion rougher due to acceleration discontinuities. A value in the range 20 Hz to 200 Hz should be reasonable to start.

Und schließlich können Sie einen Werkzeugweg mit vielen kleinen, engen Kurven nicht beschleunigen, da Sie durch die Kurvenbeschleunigung eingeschränkt sind.

- `SPINDLES = 3` - The number of spindles to support. It is imperative that this number matches the "num_spindles" parameter passed to the motion module.
- `COORDINATES = X Y Z` - The names of the axes being controlled. Only X, Y, Z, A, B, C, U, V, W are valid. Only axes named in `COORDINATES` are accepted in G-code. It is permitted to write an axis name more than once (e.g., X Y Y Z for a gantry machine). For the common *trivkins kinematics*, joint numbers are assigned in sequence according to the trivkins parameter *coordinates=*. So, for trivkins *coordinates=xz*, joint0 corresponds to X and joint1 corresponds to Z. See the kinematics man page (*$ man kins*) for information on trivkins and other kinematics modules.
- `LINEAR_UNITS = <units>_` - Specifies the *machine units* for linear axes. Possible choices are mm or inch. This does not affect the linear units in NC code (the G20 and G21 words do this).
- `ANGULAR_UNITS = <units>` - Specifies the *machine units* for rotational axes. Possible choices are *deg*, *degree* (360 per circle), *rad*, *radian* (2*π per circle), *grad*, or *gon* (400 per circle). This does not affect the angular units of NC code. In RS274NGC, A-, B- and C- words are always expressed in degrees.
- `DEFAULT_LINEAR_VELOCITY = 0.0167` - The initial rate for jogs of linear axes, in machine units per second. The value shown in *AXIS* equals machine units per minute.
- `DEFAULT_LINEAR_ACCELERATION = 2.0` - In machines with nontrivial kinematics, the acceleration used for "teleop" (Cartesian space) jogs, in *machine units* per second per second.
- `MAX_LINEAR_VELOCITY = 5.0` - The maximum velocity for any axis or coordinated move, in *machine units* per second. The value shown equals 300 units per minute.
- `MAX_LINEAR_ACCELERATION = 20.0` - The maximum acceleration for any axis or coordinated axis move, in *machine units* per second per second.
- `POSITION_FILE = position.txt` - If set to a non-empty value, the joint positions are stored between runs in this file. This allows the machine to start with the same coordinates it had on shutdown. This assumes there was no movement of the machine while powered off. If unset, joint positions are not stored and will begin at 0 each time LinuxCNC is started. This can help on smaller machines without home switches. If using the Mesa resolver interface this file can be used to emulate absolute encoders and eliminate the need for homing (with no loss of accuracy). See the hostmot2 manpage for more details.
- `NO_FORCE_HOMING = 1` - The default behavior is for LinuxCNC to force the user to home the machine before any MDI command or a program is run. Normally, only jogging is allowed before homing. For configurations using identity kinematics, setting `NO_FORCE_HOMING = 1` allows the user to make MDI moves and run programs without homing the machine first. Interfaces using identity kinematics without homing ability will need to have this option set to 1.

**Warnung**
LinuxCNC will not know your joint travel limits when using `NO_FORCE_HOMING = 1`.

- `HOME = 0 0 0 0 0 0 0 0 0` - World home position needed for kinematics modules that compute world coordinates using kinematicsForward() when switching from joint to teleop mode. Up to nine

coordinate values (X Y Z A B C U V W) may be specified, unused trailing items may be omitted. This value is only used for machines with nontrivial kinematics. On machines with trivial kinematics (mill, lathe, gantry types) this value is ignored. Note: The sim hexapod config requires a non-zero value for the Z coordinate.

- `TPMOD` = *alternate_trajectory_planning module* [tp_parms=value]
  The `TPMOD` variable is optional. If specified, use a specified (user-built) module instead of the default (tpmod). Module parameters (tp_parms) may be included if supported by the named module. The setting may be overridden from the command line using the -t option ($ linuxcnc -h).
- `NO_PROBE_JOG_ERROR = 0` - Allow to bypass probe tripped check when you jog manually.
- `NO_PROBE_HOME_ERROR = 0` - Allow to bypass probe tripped check when homing is in progress.

#### 4.4.2.11 [KINS] Section

* `JOINTS = 3` - Specifies the number of joints (motors) in the system. For example, a trivkins XYZ machine with a single motor for each axis has 3 joints. A gantry machine with one motor on each of two of the axes, and two motors on the third axis, has 4 joints. (This config variable may be used by a GUI to set the number of joints (num_joints) specified to the motion module (motmod).) * `KINEMATICS = trivkins` - Specify a kinematics module for the motion module. GUIs may use this variable to specify the `loadrt`-line in HAL files for the motmod module. For more information on kinematics modules see the manpage: `$ man kins`.

#### 4.4.2.12 [AXIS_<letter>] Section

The <*letter*> specifies one of: X Y Z A B C U V W

- `TYPE = LINEAR` - The type of this axis, either `LINEAR` or `ANGULAR`. Required if this axis is not a default axis type. The default axis types are X,Y,Z,U,V,W = LINEAR and A,B,C = ANGULAR. This setting is effective with the AXIS GUI but note that other GUI's may handle things differently.
- `MAX_VELOCITY = 1.2` - Maximum velocity for this axis in machine units per second.
- `MAX_ACCELERATION = 20.0` - Maximum acceleration for this axis in machine units per second squared.
- `MIN_LIMIT = -1000` - The minimum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before `MIN_LIMIT` is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MIN_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of -1e99 is used.
- `MAX_LIMIT = 1000` - The maximum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before MAX_LIMIT is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MAX_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of 1e99 is used.
- `WRAPPED_ROTARY = 1` - When this is set to 1 for an ANGULAR axis the axis will move 0-359.999 degrees. Positive Numbers will move the axis in a positive direction and negative numbers will move the axis in the negative direction.
- `LOCKING_INDEXER_JOINT = 4` - This value selects a joint to use for a locking indexer for the specified axis <*letter*>. In this example, the joint is 4 which would correspond to the B axis for a XYZAB system with trivkins (identity) kinematics. When set, a G0 move for this axis will initiate an unlock with the `joint.4.unlock pin` then wait for the `joint.4.is-unlocked` pin then move the joint at the rapid rate for that joint. After the move the `joint.4.unlock` will be false and motion will wait for `joint.4.is-unlocked` to go false. Moving with other joints is not allowed when moving a locked rotary joint. To create the unlock pins, use the motmod parameter:

```
unlock_joints_mask=jointmask
```

Die Bits der Jointmaske sind: (LSB)0:joint0, 1:joint1, 2:joint2, ...

Example: `loadrt motmod ... unlock_joints_mask=0x38` creates unlock-pins for joints 3,4,5.

- `OFFSET_AV_RATIO = 0.1` - If nonzero, this item enables the use of HAL input pins for external axis offsets:

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

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

#### 4.4.2.13 [JOINT_<num>] Sections

The *<num>* specifies the joint number 0 ... (num_joints-1) The value of *num_joints* is set by `[KINS]JOINTS=`.

The `[JOINT_0]`, `[JOINT_1]`, etc. sections contains general parameters for the individual components in the joint control module. The joint section names begin numbering at 0, and run through the number of joints specified in the `[KINS]JOINTS` entry 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

Other kinematics modules with identity kinematics are available to support configurations with partial sets of axes. For example, using trivkins with `coordinates=XZ`, the joint-axes relationships are:

- JOINT_0 = X
- JOINT_1 = Z

For more information on kinematics modules see the manpage *kins* (on the UNIX terminal type `man kins`).

- `TYPE = LINEAR` - The type of joint, either `LINEAR` or `ANGULAR`.
- `UNITS = INCH` - If specified, this setting overrides the related `[TRAJ] UNITS` setting, e.g., `[TRAJ]LINEAR_U` if the `TYPE` of this joint is `LINEAR`, `[TRAJ]ANGULAR_UNITS` if the `TYPE` of this joint is `ANGULAR`.
- `MAX_VELOCITY = 1.2` - Maximum velocity for this joint in machine units per second.

- `MAX_ACCELERATION = 20.0` - Maximum acceleration for this joint in machine units per second squared.
- `BACKLASH = 0.0000` - Backlash in machine units. Backlash compensation value can be used to make up for small deficiencies in the hardware used to drive an joint. If backlash is added to an joint and you are using steppers the `STEPGEN_MAXACCEL` must be increased to 1.5 to 2 times the `MAX_ACCELERATION` for the joint. Excessive backlash compensation can cause an joint to jerk as it changes direction. If a COMP_FILE is specified for a joint BACKLASH is not used.
- `COMP_FILE` = *file.extension* - The compensation file consists of map of position information for the joint. Compensation file values are in machine units. Each set of values are are on one line separated by a space. The first value is the nominal value (the commanded position). The second and third values depend on the setting of `COMP_FILE_TYPE`. Points in between nominal values are interpolated between the two nominals. Compensation files must start with the smallest nominal and be in ascending order to the largest value of nominals. File names are case sensitive and can contain letters and/or numbers. Currently the limit inside LinuxCNC is for 256 triplets per joint.

  If `COMP_FILE` is specified for a joint, `BACKLASH` is not used.
- `COMP_FILE_TYPE = 0` or `1` - Specifies the type of compensation file. The first value is the nominal (commanded) position for both types.
  A `COMP_FILE_TYPE` must be specified for each `COMP_FILE`.
  - *Type 0:* The second value specifies the actual position as the joint is moving in the positive direction (increasing value). The third value specifies the actual position as the joint is moving in the negative direction (decreasing value).

    **Typ 0 Beispiel**

    ```
    -1.000 -1.005 -0.995
    0.000 0.002 -0.003
    1.000 1.003 0.998
    ```
  - *Type 1:* The second value specifies positive offset from nominal while traveling in the positive direction. The third value specifies the negative offset from nominal while traveling in a negative direction.

    **Typ 1 Beispiel**

    ```
    -1.000 0.005 -0.005
    0.000 0.002 -0.003
    1.000 0.003 -0.004
    ```
- `MIN_LIMIT = -1000` - The minimum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MIN_LIMIT` for that joint in the `[JOINT_N]` section a the value -1e99 is used.
- `MAX_LIMIT = 1000` - The maximum limit for joint motion, in machine units. When this limit is reached, the controller aborts joint motion. For a rotary joint with unlimited rotation having no `MAX_LIMIT` for that joint in the `[JOINT_N]` section a the value 1e99 is used.

---

**Anmerkung**
For **identity** kinematics, the `[JOINT_N]MIN_LIMIT/MAX_LIMIT` settings must equal or exceed the corresponding (one-to-one identity) `[AXIS_L]` limits. These settings are verified at startup when the trivkins kinematics modules is specified.

---

**Anmerkung**
The [JOINT_N]MIN_LIMIT/MAX_LIMIT settings are enforced while jogging in joint mode prior to homing. After homing, [AXIS_L]MIN_LIMIT/MAX_LIMIT coordinate limits are used as constraints for axis (coordinate letter) jogging and by the trajectory planning used for G-code moves (programs and MDI commands). The trajectory planner works in Cartesian space (XYZABCUVW) and has no information about the motion of joints implemented by **any** kinematics module. It is possible for joint limit violations to occur for G-code that obeys trajectory planning position limits when non identity kinematics are used. The motion module always detects joint position limit violations and faults if they occur during the execution of G-code commands. See also related GitHub issue #97.

- MIN_FERROR = 0.010 - This is the value in machine units by which the joint is permitted to deviate from commanded position at very low speeds. If MIN_FERROR is smaller than FERROR, the two produce a ramp of error trip points. You could think of this as a graph where one dimension is speed and the other is permitted following error. As speed increases the amount of following error also increases toward the FERROR value.
- FERROR = 1.0 - FERROR is the maximum allowable following error, in machine units. If the difference between commanded and sensed position exceeds this amount, the controller disables servo calculations, sets all the outputs to 0.0, and disables the amplifiers. If MIN_FERROR is present in the INI file, velocity-proportional following errors are used. Here, the maximum allowable following error is proportional to the speed, with FERROR applying to the rapid rate set by [TRAJ]MAX_VELOCITY, and proportionally smaller following errors for slower speeds. The maximum allowable following error will always be greater than MIN_FERROR. This prevents small following errors for stationary axes from inadvertently aborting motion. Small following errors will always be present due to vibration, etc.
- LOCKING_INDEXER = 1 - Indicates the joint is used as a locking indexer.

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

- HOME = 0.0 - The position that the joint will go to upon completion of the homing sequence.
- HOME_OFFSET = 0.0 - The joint position of the home switch or index pulse, in machine units. When the home point is found during the homing process, this is the position that is assigned to that point. When sharing home and limit switches and using a home sequence that will leave the home/limit switch in the toggled state, the home offset can be used define the home switch position to be other than 0 if your HOME position is desired to be 0.
- HOME_SEARCH_VEL = 0.0 - Initial homing velocity in machine units per second. Sign denotes direction of travel. A value of zero means assume that the current location is the home position for the machine. If your machine has no home switches you will want to leave this value at zero.
- HOME_LATCH_VEL = 0.0 - Homing velocity in machine units per second to the home switch latch position. Sign denotes direction of travel.
- HOME_FINAL_VEL = 0.0 - Velocity in machine units per second from home latch position to home position. If left at 0 or not included in the joint rapid velocity is used. Must be a positive number.
- HOME_USE_INDEX = NO - If the encoder used for this joint has an index pulse, and the motion card has provision for this signal you may set it to yes. When it is yes, it will affect the kind of home pattern used. Currently, you can't home to index with steppers unless you're using StepGen in velocity mode and PID.
- HOME_INDEX_NO_ENCODER_RESET = NO - Use YES if the encoder used for this joint does not reset its counter when an index pulse is detected after assertion of the joint index_enable HAL pin. Applicable only for HOME_USE_INDEX = YES.

- `HOME_IGNORE_LIMITS = NO` - When you use the limit switch as a home switch and the limit switch this should be set to YES. When set to YES the limit switch for this joint is ignored when homing. You must configure your homing so that at the end of your home move the home/limit switch is not in the toggled state you will get a limit switch error after the home move.
- `HOME_IS_SHARED = <n>` - If the home input is shared by more than one joint set <*n*> to 1 to prevent homing from starting if the one of the shared switches is already closed. Set <*n*> to 0 to permit homing if a switch is closed.
- `HOME_ABSOLUTE_ENCODER = 0 | 1 | 2` - Used to indicate the joint uses an absolute encoder. At a request for homing, the current joint value is set to the `HOME_OFFSET` value. If the `HOME_ABSOLUTE_ENCODER` setting is 1, the machine makes the usual final move to the `HOME` value. If the `HOME_ABSOLUTE_ENCODER` setting is 2, no final move is made.
- `HOME_SEQUENCE = <n>` - Used to define the ”Home All” sequence. <*n*> must start at `0` or `1` or `-1`. Additional sequences may be specified with numbers increasing by 1 (in absolute value). Skipping of sequence numbers is not allowed. If a `HOME_SEQUENCE` is omitted, the joint will not be homed by the ”Home All” function. More than one joint can be homed at the same time by specifying the same sequence number for more than one joint. A negative sequence number is used to defer the final move for all joints having that (negative or positive) sequence number. For additional info, see: HOME SEQUENCE.
- `VOLATILE_HOME = 0` - When enabled (set to 1) this joint will be unhomed if the Machine Power is off or if E-Stop is on. This is useful if your machine has home switches and does not have position feedback such as a step and direction driven machine.

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` - How close is close enough to consider the motor in position, in machine units.

  Dies wird oft auf einen Abstand eingestellt, der 1, 1,5, 2 oder 3 Encoderzählungen entspricht, aber es gibt keine strengen Regeln. Lockere (größere) Einstellungen ermöglichen ein geringeres Hunting’ des Servos auf Kosten einer geringeren Genauigkeit. Engere (kleinere) Einstellungen versuchen eine höhere Genauigkeit auf Kosten von mehr Servo *Hunting*. Ist es wirklich genauer, wenn es auch unsicherer ist? Generell ist es gut, das Hunting’ der Servos zu vermeiden oder zumindest zu begrenzen, wenn Sie können.

  Be careful about going below 1 encoder count, since you may create a condition where there is no place that your servo is happy. This can go beyond *hunting* (slow) to *nervous* (rapid), and even to *squealing* which is easy to confuse with oscillation caused by improper tuning. Better to be a count or two loose here at first, until you’ve been through *gross tuning* at least.

  Example of calculating machine units per encoder pulse to use in deciding `DEADBAND` value:

$$\frac{1\,revolution}{1000\,lines} X \frac{1\,line}{4\,pulse/line} X \frac{0.2\,units}{1\,revolution} = \frac{0.200\,units}{4000\,pulses} = \frac{0.00005\,units}{1\,pulse}$$

- `BIAS = 0.000` - This is used by hm2-servo and some others. Bias is a constant amount that is added to the output. In most cases it should be left at zero. However, it can sometimes be useful to compensate for offsets in servo amplifiers, or to balance the weight of an object that moves vertically. Bias is turned off when the PID loop is disabled, just like all other components of the output.
- `P = 50` - The proportional gain for the joint servo. This value multiplies the error between commanded and actual position in machine units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the P gain are volts per machine unit, e.g., $\frac{volts}{unit}$
- `I = 0` - The integral gain for the joint servo. The value multiplies the cumulative error between commanded and actual position in machine units, resulting in a contribution to the computed voltage for the motor amplifier. The units on the I gain are volts per machine unit second, e.g., $\frac{volts}{unit\ second}$
- `D = 0` - The derivative gain for the joint servo. The value multiplies the difference between the current and previous errors, resulting in a contribution to the computed voltage for the motor amplifier. The units on the D gain are volts per machine unit per second, e.g., $\frac{volts}{unit\ second}$
- `FF0 = 0` - The $0^{th}$ order feed forward gain. This number is multiplied by the commanded position, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF0 gain are volts per machine unit, e.g., $\frac{volts}{unit}$
- `FF1 = 0` - The $1^{st}$ order feed forward gain. This number is multiplied by the change in commanded position per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF1 gain are volts per machine unit per second, e.g., $\frac{volts}{unit\ second}$
- `FF2 = 0` - The $2^{nd}$ order feed forward gain. This number is multiplied by the change in commanded position per second per second, resulting in a contribution to the computed voltage for the motor amplifier. The units on the FF2 gain are volts per machine unit per second per second, e.g., $\frac{volts}{unit\ second^2}$
- `OUTPUT_SCALE = 1.000`
- `OUTPUT_OFFSET = 0.000`

  Diese beiden Werte sind die Skalierungs- und Offset-Faktoren für den gemeinsamen Ausgang zu den Motorverstärkern.

  The second value (offset) is subtracted from the computed output (in volts), and divided by the first value (scale factor), before being written to the D/A converters. The units on the scale value are in true volts per DAC output volts. The units on the offset value are in volts. These can be used to linearize a DAC. Specifically, when writing outputs, the LinuxCNC first converts the desired output in quasi-SI units to raw actuator values, e.g., Volts for an amplifier DAC. This scaling looks like: $raw = \frac{output - offset}{scale}$

  The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [output SI units]/[actuator units]. For example, on a machine with a velocity mode amplifier such that 1 V results in 250 mm/s velocity.

$$amplifier[volts] = (output[\frac{mm}{sec}] - offset[\frac{mm}{sec}])/250\frac{mm}{secvolt}$$

Note that the units of the offset are in machine units, e.g. mm/s, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of your output which yields 0.0 for the actuator output. If the DAC is linearized, this offset is normally 0.0.

Skalierung und Offset können auch zur Linearisierung des DAC verwendet werden. Diese Werte spigeln dann die kombinierten Auswirkungen von Verstärkung, Nicht-Linearität des DAC, DAC-Einheiten usw. wider.

Gehen Sie dazu folgendermaßen vor.

1. Erstellen Sie eine Kalibrierungstabelle für den Ausgang, indem Sie den DAC mit einer gewünschten Spannung betreiben und das Ergebnis messen.
2. Führen Sie eine lineare Anpassung nach dem Prinzip der kleinsten Quadrate durch, um die Koeffizienten a und b so zu ermitteln, dass $measured = a * raw + b$
3. Beachten Sie, dass wir eine Rohausgabe wünschen, bei der das gemessene Ergebnis mit der befohlenen Ausgabe identisch ist. Das bedeutet
   a. $command = a * raw + b$
   b. $raw = (command - b) / a$
4. Folglich können die Koeffizienten a und b aus der linearen Anpassung direkt als Skala und Offset für den Regler verwendet werden.

In der folgenden Tabelle finden Sie ein Beispiel für Spannungsmessungen.

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` - The maximum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units. The value is applied symmetrically to both the plus and the minus side.
- `INPUT_SCALE = 20000` - in Sample configs
- `ENCODER_SCALE = 20000` - in PnCconf built configs

Specifies the number of pulses that corresponds to a move of one machine unit as set in the `[TRAJ]` section. For a linear joint one machine unit will be equal to the setting of `LINEAR_UNITS`. For an angular joint one unit is equal to the setting in `ANGULAR_UNITS`. A second number, if specified, is ignored. For example, on a 2000 counts per rev encoder, and 10 revs/inch gearing, and desired units of inch, we have:

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

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

**Warnung**
The following are custom INI file entries that you may find in a sample INI file or a wizard generated file. These are not used by the LinuxCNC software and meant only to put all the settings in one place. For more information on custom INI file entries see the Custom Sections and Variables subsection.

Die folgenden Elemente können von einer Schrittgenerator (engl. StepGen)-Komponente verwendet werden.

- `SCALE = 4000` - in Sample configs
- `STEP_SCALE = 4000` - in PnCconf built configs

Specifies the number of pulses that corresponds to a move of one machine unit as set in the `[TRAJ]` section. For stepper systems, this is the number of step pulses issued per machine unit. For a linear joint one machine unit will be equal to the setting of `LINEAR_UNITS`. For an angular joint one unit is equal to the setting in `ANGULAR_UNITS`. For servo systems, this is the number of feedback pulses per machine unit. A second number, if specified, is ignored.

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` (Optionally used in PnCconf built configs) - Specifies the number of pulses that corresponds to a move of one machine unit as set in the `[TRAJ]` section. For a linear joint one machine unit will be equal to the setting of `LINEAR_UNITS`. For an angular joint one unit is equal to the setting in `ANGULAR_UNITS`. A second number, if specified, is ignored. For example, on a 2000 counts per rev encoder, and 10 revs/inch gearing, and desired units of inch, we have:

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

- `STEPGEN_MAXACCEL = 21.0` - Acceleration limit for the step generator. This should be 1% to 10% larger than the joint `MAX_ACCELERATION`. This value improves the tuning of StepGen's "position loop". If you have added backlash compensation to an joint then this should be 1.5 to 2 times greater than `MAX_ACCELERATION`.
- `STEPGEN_MAXVEL = 1.4` - Older configuration files have a velocity limit for the step generator as well. If specified, it should also be 1% to 10% larger than the joint `MAX_VELOCITY`. Subsequent testing has shown that use of `STEPGEN_MAXVEL` does not improve the tuning of StepGen's position loop.

#### 4.4.2.14 [SPINDLE_<num>] Section(s)

The <*num*> specifies the spindle number 0 ... (num_spindles-1)
The value of *num_spindles* is set by `[TRAJ]SPINDLES=` .
By default maximum velocity of the spindle in forward and reverse is approximately 2147483000 RPM.
By default minimum velocity of the spindle in forward and reverse is 0 RPM.
By default the increment is 100 RPM.
You change these default by setting the following INI variables:

**Anmerkung**
These settings are for the motion controller component. Control screens can limit these settings further.

- `MAX_FORWARD_VELOCITY = 20000` The maximum spindle speed (in rpm) for the specified spindle. Optional. This will also set MAX_REVERSE_VELOCITY to the negative value unless overridden.
- `MIN_FORWARD_VELOCITY = 3000` The minimum spindle speed (in rpm) for the specified spindle. Optional. Many spindles have a minimum speed below which they should not be run. Any spindle speed command below this limit will be /increased/ to this limit.
- `MAX_REVERSE_VELOCITY = 20000` This setting will default to `MAX_FORWARD_VELOCITY` if omitted. It can be used in cases where the spindle speed is limited in reverse. Set to zero for spindles which must not be run in reverse. In this context ”max” refers to the absolute magnitude of the spindle speed.
- `MIN_REVERSE_VELOCITY = 3000` This setting is equivalent to `MIN_FORWARD_VELOCITY` but for reverse spindle rotation. It will default to the MIN_FORWARD_VELOCITY if omitted.
- `INCREMENT = 200` Sets the step size for spindle speed increment / decrement commands. This can have a different value for each spindle. This setting is effective with AXIS and Touchy but note that some control screens may handle things differently.
- `HOME_SEARCH_VELOCITY = 100` - FIXME: Spindle homing not yet working. Sets the homing speed (rpm) for the spindle. The spindle will rotate at this velocity during the homing sequence until the spindle index is located, at which point the spindle position will be set to zero. Note that it makes no sense for the spindle home position to be any value other than zero, and so there is no provision to do so.
- `HOME_SEQUENCE = 0` - FIXME: Spindle homing not yet working Controls where in the general homing sequence the spindle homing rotations occur. Set the `HOME_SEARCH_VELOCITY` to zero to avoid spindle rotation during the homing sequence.

#### 4.4.2.15 [EMCIO] Section

* `EMCIO = io` - Name of IO controller program. * `CYCLE_TIME = 0.100` - The period, in seconds, at which EMCIO will run. Making it 0.0 or a negative number will tell EMCIO not to sleep at all. There is usually no need to change this number. * `TOOL_TABLE = tool.tbl` - The file which contains tool information, described in the User Manual. * `DB_PROGRAM = db_program` - Path to an executable program that manages tool data. When a DB_PROGRAM is specified, a TOOL_TABLE entry is ignored. * `TOOL_CHANGE_POSITION = 0 0 2` - Specifies the XYZ location to move to when performing a tool change if three digits are used. Specifies the XYZABC location when 6 digits are used. Specifies the XYZABCUVW location when 9 digits are used. Tool Changes can be combined. For example if you combine the quill up with change position you can move the Z first then the X and Y. * `TOOL_CHANGE_WITH_SPINDLE_ON = 1` - The spindle will be left on during the tool change when the value is 1. Useful for lathes or machines where the material is in the spindle, not the tool. * `TOOL_CHANGE_QUILL_UP = 1` - The Z axis will be moved to machine zero prior to the tool change when the value is 1. This is the same as issuing a `G0 G53 Z0`. * `TOOL_CHANGE_AT_G30 = 1` - The machine is moved to reference point defined by parameters 5181-5186 for G30 if the value is 1. For more information see G-code Parameters and G-code G30-G30.1. * `RANDOM_TOOLCHANGER = 1` - This is for machines that cannot place the tool back into the pocket it came from. For example, machines that exchange the tool in the active pocket with the tool in the spindle.

# 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.

- The negative and positive directions are based on Tool Movement which can be different from the actual machine movement. I.e., on a mill typically the table moves rather then the tool.
- 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.
- If using a separate homing switch, it is possible to start homing on the wrong side of the home switch, which combined with HOME_IGNORE_LIMITS option will lead to a hard crash. You can avoid this by making the home switch toggle its state when the trip dog is on a particular side until it returns passed the trip point again. Said another way, the home switch state must represent the position of the dog relative to the switch (ie *before* or *after* the switch), and must stay that way even if the dog coasts past the switch in the same direction.

**Anmerkung**
While it is possible to use LinuxCNC with the G53 machine origin outside the soft machine limits, if you use G28 or G30 without setting the parameters it goes to the origin by default. This would trip the limit switches before getting to position.

### 4.5.3 Separater Home-Schalter Beispiel-Layout

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

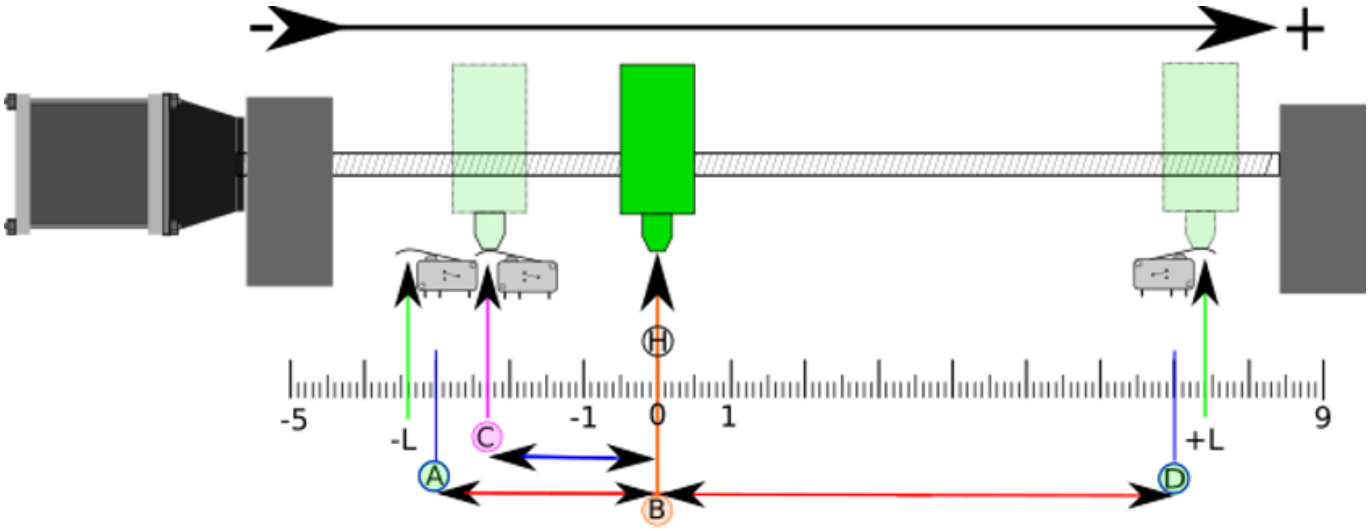



Abbildung 4.6: Demonstratives separates Schalterlayout

- 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

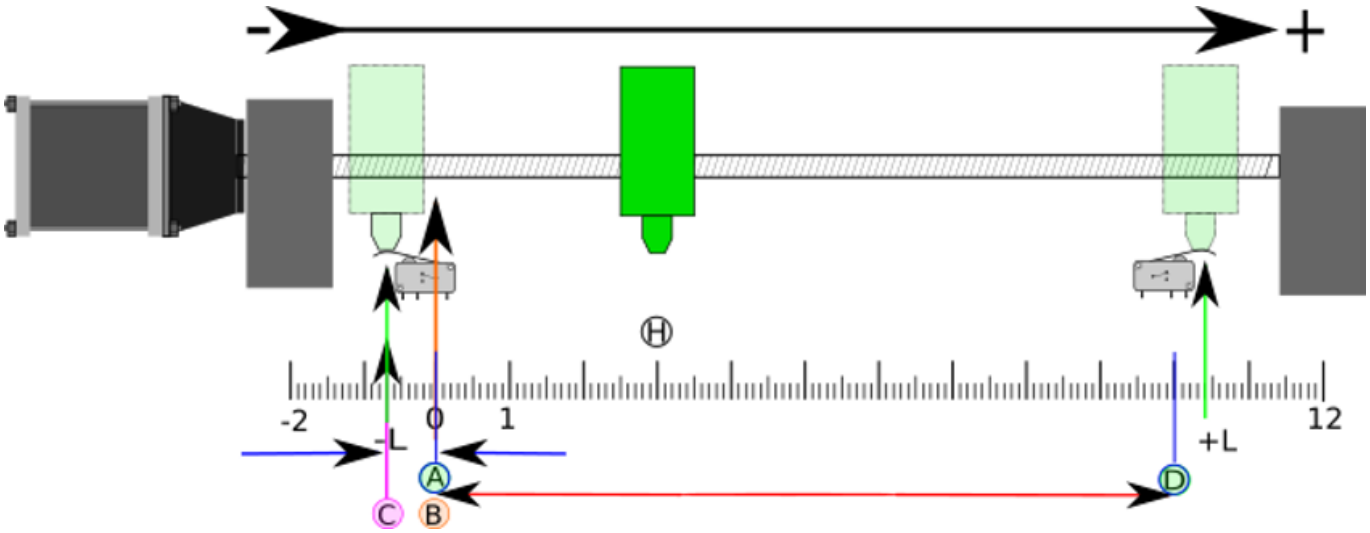



Abbildung 4.7: Beispiel für ein Layout mit geteilten Schaltern

- 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 Grenzwert-auslöser 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

There are four possible homing sequences defined by the sign of HOME_SEARCH_VEL and HOME_LATCH_V along with the associated configuration parameters as shown in the following table. Two basic conditions exist, HOME_SEARCH_VEL and HOME_LATCH_VEL are the same sign or they are opposite signs. For a more detailed description of what each configuration parameter does, see the following section.

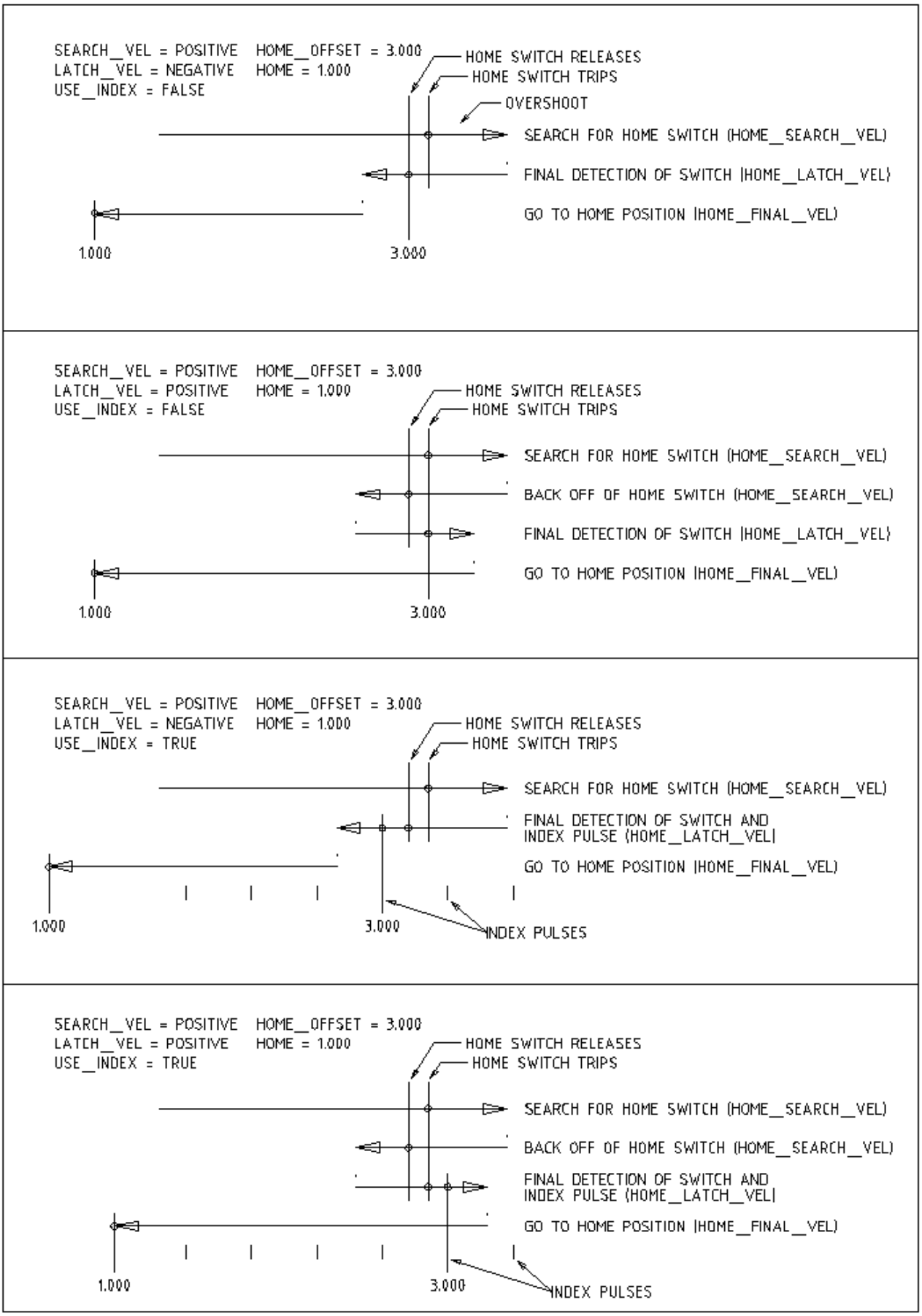



Abbildung 4.8: Referenzfahrt-Abläufe

## 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

This variable has units of machine-units per second.

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.

If HOME_SEARCH_VEL is non-zero, then LinuxCNC assumes that there is a home switch. It begins by checking whether the home switch is already tripped. If tripped it backs off the switch at HOME_SEARCH_VEL. The direction of the back-off is opposite the sign of HOME_SEARCH_VEL. Then it searches for the home switch by moving in the direction specified by the sign of HOME_SEARCH_VEL, at a speed determined by its absolute value. When the home switch is detected, the joint will stop as fast as possible, but there will always be some overshoot. The amount of overshoot depends on the speed. If it is too high, the joint might overshoot enough to hit a limit switch or crash into the end of travel. On the other hand, if HOME_SEARCH_VEL is too low, homing can take a long time.

### 4.5.6.2 HOME_LATCH_VEL

This variable has units of machine-units per second.

Specifies the speed and direction that LinuxCNC uses when it makes its final accurate determination of the home switch (if present) and index pulse location (if present). It will usually be slower than the search velocity to maximize accuracy. If HOME_SEARCH_VEL and HOME_LATCH_VEL have the same sign, then the latch phase is done while moving in the same direction as the search phase. (In that case, LinuxCNC first backs off the switch, before moving towards it again at the latch velocity.) If HOME_SEARCH_VEL and HOME_LATCH_VEL have opposite signs, the latch phase is done while moving in the opposite direction from the search phase. That means LinuxCNC will latch the first pulse after it moves off the switch. If HOME_SEARCH_VEL is zero (meaning there is no home switch), and this parameter is nonzero, LinuxCNC goes ahead to the index pulse search. If HOME_SEARCH_VEL is non-zero and this parameter is zero, it is an error and the homing operation will fail. The default value is zero.

### 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

Can hold the values YES / NO. The default value for this parameter is NO. This flag determines whether LinuxCNC will ignore the limit switch input for this joint while homing. This setting will not ignore limit inputs for other joints. If you do not have a separate home switch set this to YES and connect the limit switch signal to the joint home switch input in HAL. LinuxCNC will ignore the limit switch input for this joint while homing. To use only one input for all homing and limits you will have to block the limit signals of the joints not homing in HAL and home one joint at a time.

#### 4.5.6.5 HOME_USE_INDEX

Specifies whether or not there is an index pulse. If the flag is true (HOME_USE_INDEX = YES), LinuxCNC will latch on the rising edge of the index pulse. If false, LinuxCNC will latch on either the rising or falling edge of the home switch (depending on the signs of HOME_SEARCH_VEL and HOME_LATCH_VEL). The default value is NO.

---

**Anmerkung**
HOME_USE_INDEX requires connections in your HAL file to `joint.n.index-enable` from the `encoder.n.index-enable`.

---

#### 4.5.6.6 HOME_INDEX_NO_ENCODER_RESET

Default is NO. Use YES if the encoder used for this joint does not reset its counter when an index pulse is detected after assertion of the joint index_enable HAL pin. Applicable only for HOME_USE_INDEX = YES.

#### 4.5.6.7 HOME_OFFSET

This defines the location of the origin zero point of the G53 machine coordinate system. It is the distance (offset), in joint units, from the machine origin to the home switch trip point or index pulse. After detecting the switch trip point/index pulse, LinuxCNC sets the joint coordinate position to HOME_OFFSET, thus defining the origin, which the soft limits references from. The default value is zero.

---

**Anmerkung**
The home switch location, as indicated by the HOME_OFFSET variable, can be inside or outside the soft limits. They will be shared with or inside the hard limit switches.

---

#### 4.5.6.8 HOME

The position that the joint will go to upon completion of the homing sequence. After detecting the home switch or home switch then index pulse (depending on configuration), and setting the coordinate of that point to HOME_OFFSET, LinuxCNC makes a move to HOME as the final step of the homing process. The default value is zero. Note that even if this parameter is the same as HOME_OFFSET, the joint will slightly overshoot the latched position as it stops. Therefore there will always be a small move at this time (unless HOME_SEARCH_VEL is zero, and the entire search/latch stage was skipped). This final move will be made at the joint's maximum velocity unless HOME_FINAL_VEL has been set.

**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

If there is not a separate home switch input for this joint, but a number of momentary switches wired to the same pin, set this value to 1 to prevent homing from starting if one of the shared switches is already closed. Set this value to 0 to permit homing even if the switch is already closed.

#### 4.5.6.10 HOME_ABSOLUTE_ENCODER

Use for absolute encoders. When a request is made to home the joint, the current joint position is set to the *[JOINT_n]HOME_OFFSET* value.

Die abschließende Bewegung zur *[JOINT_n]HOME* Position ist entsprechend der *HOME_ABSOLUTE_ENCOD* 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

Used to define a multi-joint homing sequence **HOME ALL** and enforce homing order (e.g., Z may not be homed if X is not yet homed). A joint may be homed after all joints with a lower (absolute value) HOME_SEQUENCE have already been homed and are at the HOME_OFFSET. If two joints have the same HOME_SEQUENCE, they may be homed at the same time.

**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 values indicate that joints in the sequence should **synchronize the final move** to [JOINT_n]HOME by waiting until all joints in the sequence are ready. If any joint has a

**negative** HOME_SEQUENCE value, then all joints with the same absolute value (positive or negative) of the HOME_SEQUENCE item value will synchronize the final move.

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

If this setting is true, this joint becomes unhomed whenever the machine transitions into the OFF state. This is appropriate for any joint that does not maintain position when the joint drive is off. Some stepper drives, especially microstep drives, may need this.

#### 4.5.6.13 LOCKING_INDEXER

If this joint is a locking rotary indexer, it will unlock before homing, and lock afterward.

#### 4.5.6.14 Immediate Homing

If a joint does not have home switches or does not have a logical home position like a rotary joint and you want that joint to home at the current position when the ”Home All” button is pressed in the AXIS GUI, then the following INI entries for that joint are needed.

```
HOME_SEARCH_VEL = 0
HOME_LATCH_VEL = 0
HOME_USE_INDEX = NO
HOME_OFFSET = 0 (Or the home position offset (HOME))
HOME_SEQUENCE = 0 (or other valid sequence number)
```

---

**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 Inhibiting Homing

A HAL pin (motion.homing-inhibit) is provided to disallow homing initiation for both ”Home All” and individual joint homing.

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**
Realtime synchronization of joint jogging for multiple joints requires additional HAL connections for the Manual-Pulse-Generator (MPG) type jog pins (`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

I/O control handles I/O tasks like coolant, toolchange, E-stop and lube. The signals are turned on and off with G-code or in the case of E-stop 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**
You need to explicitly enable the start-change protocol by adding the -support-start-change option; otherwise the start-change pin remains low and start-change-ack is ignored. The reason for this is better backwards compatibility.

[TASK] Abschnitt

**IO_ERROR**
Printf-style template for operator error display (negative toolchanger fault codes). No quoting needed. Example: `IO_ERROR = Toolchanger fault %d`. Default: toolchanger error %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 when flood coolant is requested.
- *iocontrol.0.coolant-mist* (bit, out) TRUE when mist coolant is requested.
- *iocontrol.0.emc-enable-in* (bit, in) Should be driven FALSE when an external E-stop condition exists.
- *iocontrol.0.lube* (bit, out) TRUE when lube is requested. This pin gets driven TRUE when the controller comes out of E-stop, and when the ”Lube On” command gets sent to the controller. It gets driven FALSE when the controller goes into E-stop, and when the ”Lube Off” command gets sent to the controller.
- *iocontrol.0.lube_level* (bit, in) Should be driven FALSE when lubrication tank is empty.
- *iocontrol.0.tool-change* (bit, out) TRUE when a tool change is requested
- *iocontrol.0.tool-changed* (bit, in) Should be driven TRUE when a tool change is completed.
- *iocontrol.0.tool-number* (s32, out) Current tool number
- *iocontrol.0.tool-prep-number* (s32, out) The number of the next tool, from the RS274NGC T-word
- *iocontrol.0.tool-prep-pocket* (s32, out) This is the pocket number (location in the tool storage mechanism) of the tool requested by the most recent T-word.
- *iocontrol.0.tool-prepare* (bit, out) TRUE when a T*n* tool prepare is requested.
- *iocontrol.0.tool-prepared* (bit, in) Should be driven TRUE when a tool prepare is completed.
- *iocontrol.0.user-enable-out* (bit, out) FALSE when an internal E-stop condition exists

- *iocontrol.0.user-request-enable* (bit, out) TRUE when the user has requested that E-stop be cleared

Zusätzliche Pins hinzugefügt durch I/O Control V2

- emc-abort: (bit, out) signals emc-originated abort to toolchanger.
- emc-abort-ack: (bit, in) Acknowledge line from toolchanger for previous signal, or jumpered to abort-tool-change if not used in toolchanger. NB: after signaling an emc-abort, iov2 will block until emc-abort-ack is raised.
- emc-reason: (S32, out) convey cause for EMC-originated abort to toolchanger. Usage: UI informational. Valid during emc-abort TRUE.
- start-change: (bit, out) asserted at the very beginning of an M6 operation, before any spindle-off, quill-up, or move-to-toolchange-position operations are executed.
- start-change-ack: (bit, in) acknowledgment line for start-change.
- toolchanger-fault: (bit, in) toolchanger signals fault. This line is contionuously monitored. A fault toggles a flag in iocontrol which is reflected in the toolchanger-faulted pin.
- toolchanger-fault-ack: (bit, out) handshake line for above signal. will be set by iov2 after above fault line TRUE is recognized and deasserted when toolchanger-fault drops. Toolchanger is free to interpret the ack; reading the -ack lines assures fault has been received and acted upon.
- toolchanger-reason: (S32, in) convey reason code for toolchanger-originated fault to iov2. Usage: signal whether to continue or abort the program, plus UI informational if negative. Read during toolchanger-fault TRUE. Non-zero values will cause an Axis operator operator message or error message, see below.
- toolchanger-faulted: (bit, out) signals toolchanger-notify line has toggled and toolchanger-reason-code was in the fault range. Next M6 will abort.
- toolchanger-clear-fault: (bit, in) resets TC fault condition. Deasserts toolchanger-faulted if toolchanger-notify is line FALSE. Usage: UI - e.g., clear fault condition button.

### 4.6.4 Parameter

- iocontrol.0.tool-prep-index (s32, RO) IO's internal array index of the prepped tool requested by the most recent T-word. 0 if no tool is prepped. On random toolchanger machines this is tool's pocket number (i.e., the same as the tool-prep-pocket pin), on non-random toolchanger machines this is a small integer corresponding to the tool's location in the internal representation of the tool table. This parameter returns to 0 after a successful tool change M6.

### 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.

The usage of the toolchanger-fault-ack pin is optional. It will become TRUE when toolchanger-fault is raised and the toolchanger-reason pin has been read by iov2.

## 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

This section assumes you have done a standard install from the Live CD. After installation it is recommended that you connect the computer to the Internet and wait for the update manager to pop up and get the latest updates for LinuxCNC and Ubuntu before continuing.

### 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

If you have a Sherline several predefined configurations are provided. This is on the main menu CNC/EMC then pick the Sherline configuration that matches yours and save a copy.

### 4.8.3 Xylotex

If you have a Xylotex you can skip the following sections and go straight to the Stepper Config Wizard. LinuxCNC has provided quick setup for the Xylotex machines.

### 4.8.4 Maschineninformationen

Sammeln Sie die Informationen über jede Achse Ihrer Maschine.

Drive timing is in nano seconds. If you're unsure about the timing many popular drives are included in the stepper configuration wizard. Note some newer Gecko drives have different timing than the original one. A list is also on the user maintained LinuxCNC wiki site of more drives.

| 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 | |

| Ausgangs-Pin | Typ. Funktion | Wenn Unterschiedlich | Input Pin | Typ. Funktion | Wenn Unterschiedlich |
|---|---|---|---|---|---|
| 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 | | | | |

Note any pins not used should be set to Unused in the drop down box. These can always be changed later by running StepConf again.

### 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 Stepper Configuration

### 4.9.1 Einführung

The preferred way to set up a standard stepper machine is with the Step Configuration Wizard. See the Stepper Configuration Wizard Chapter.

This chapter describes some of the more common settings for manually setting up a stepper based system. These systems are using stepper motors with drives that accept step & direction signals.

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.

Most of this chapter is based on a sample config released along with LinuxCNC. The config is called stepper_inch, and can be found by running the Configuration Picker.

### 4.9.2 Maximale Schrittgeschwindigkeit

With software step generation, the maximum step rate is one step per two BASE_PERIODs for step-and-direction output. The maximum requested step rate is the product of an axis' MAX_VELOCITY and its INPUT_SCALE. If the requested step rate is not attainable, following errors will occur, particularly during fast jogs and G0 moves.

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

One of the major flaws in EMC was that you couldn't specify the pinout without recompiling the source code. EMC2 was far more flexible, and thus now in LinuxCNC (thanks to the Hardware Abstraction Layer) you can easily specify which signal goes where. See the HAL Basics for more information on 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
```

Depending on what you have chosen in your INI file you are using either standard_pinout.hal or xylotex_pinout.hal. These are two files that instruct the HAL how to link the various signals & pins. Further on we'll investigate the standard_pinout.hal.

#### 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).
- The read & write functions of the parport driver get assigned to the base thread [4].
- 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

If you want to change the standard_pinout.hal file, all you need is a text editor. Open the file and locate the parts you want to change.

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

If external hardware expects an ”active low” signal, set the corresponding *-invert* parameter. For instance, to invert the spindle control signal:

```
setp parport.0.pin-09-invert TRUE
```

[4] The fastest thread in the LinuxCNC setup, usually the code gets executed every few tens of microseconds.

#### 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
```

This assumes that the spindle controller’s response to PWM is simple: 0% PWM gives 0 RPM, 10% PWM gives 180 RPM, etc. If there is a minimum PWM required to get the spindle to turn, follow the example in the *nist-lathe* sample configuration to use a *scale* component.

#### 4.9.3.6 Hinzufügen eines Aktivierungssignals (engl. enable)

Some amplifiers (drives) require an enable signal before they accept and command movement of the motors. For this reason there are already defined signals called *Xen*, *Yen*, *Zen*.

Um sie zu verbinden, verwenden Sie das folgende Beispiel:

```
net Xen parport.0.pin-08-out
```

You can either have one single pin that enables all drives; or several, depending on the setup you have. Note, however, that usually when one axis faults, all the other drives will be disabled as well, so having only one enable signal / pin for all drives is a common practice.

#### 4.9.3.7 Externe NOTAUS (engl, ESTOP)-Taste

The standard_pinout.hal file assumes no external ESTOP button. For more information on an external E-Stop see the estop_latch man page.

## 4.10 Stepper Diagnostics

If what you get is not what you expect many times you just got some experience. Learning from the experience increases your understanding of the whole. Diagnosing problems is best done by divide and conquer. By this I mean if you can remove 1/2 of the variables from the equation each time you will find the problem the fastest. In the real world this is not always the case, but it’s usually a good place to start.

### 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)

Any of the above can cause the real-time pulsing to not be able to keep up the requested step rate. This can happen if you didn't run the latency test long enough to get a good number to plug into the StepConf Wizard, or if you set the Maximum Velocity or Maximum Acceleration too high.

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)
```

This error is generated by rtapi based on an indication from RTAI that a deadline was missed. It is usually an indication that the BASE_PERIOD in the [EMCMOT] section of the ini file is set too low. You should run the Latency Test for an extended period of time to see if you have any delays that would cause this problem. If you used the StepConf Wizard, run it again, and test the Base Period Jitter again, and adjust the Base Period Maximum Jitter on the Basic Machine Information page. You might have to leave the test running for an extended period of time to find out if some hardware causes intermittent problems.

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
```

# 4.11 Filter-Programme

## 4.11.1 Einführung

Die meisten Bildschirme von LinuxCNC haben die Möglichkeit, geladene Dateien durch ein ”Filterprogramm” zu senden oder das Filterprogramm zu verwenden, um G-Code zu machen. Ein solcher Filter kann jede gewünschte Aufgabe erledigen: Etwas so Einfaches wie sicherzustellen, dass die Datei mit *M2* endet, oder etwas so Kompliziertes wie die Erzeugung von G-Code aus einem Bild.

## 4.11.2 Einrichten der INI für Programmfilter

Der Abschnitt [FILTER] der INI-Datei steuert, wie die Filter funktionieren. Schreiben Sie zunächst für jeden Dateityp eine PROGRAM_EXTENSION-Zeile. Dann geben Sie das Programm an, das für jeden Dateityp ausgeführt werden soll. Dieses Programm erhält den Namen der Eingabedatei als erstes Argument und muss rs274ngc-Code in die Standardausgabe schreiben. Diese Ausgabe ist das, was im Textbereich angezeigt wird, in der Vorschau im Anzeigebereich, und dann auch von LinuxCNC ausgeführt wird. Die folgenden Zeilen fügen Unterstützung für den in LinuxCNC enthaltenen ”image-to-gcode” (engl. für Bild zu G-Code) -Konverter hinzu:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Greyscale Depth Image
png = image-to-gcode
gif = image-to-gcode
```

Es ist auch möglich, einen Interpreter anzugeben:

```
PROGRAM_EXTENSION = .py Python Script
py = python
```

Auf diese Weise kann jedes Python-Skript geöffnet werden, und seine Ausgabe wird als G-Code behandelt. Ein solches Beispielskript ist unter ”nc_files/holecircle.py” verfügbar. Dieses Skript erzeugt G-Code für das Bohren einer Reihe von Löchern entlang des Umfangs eines Kreises.

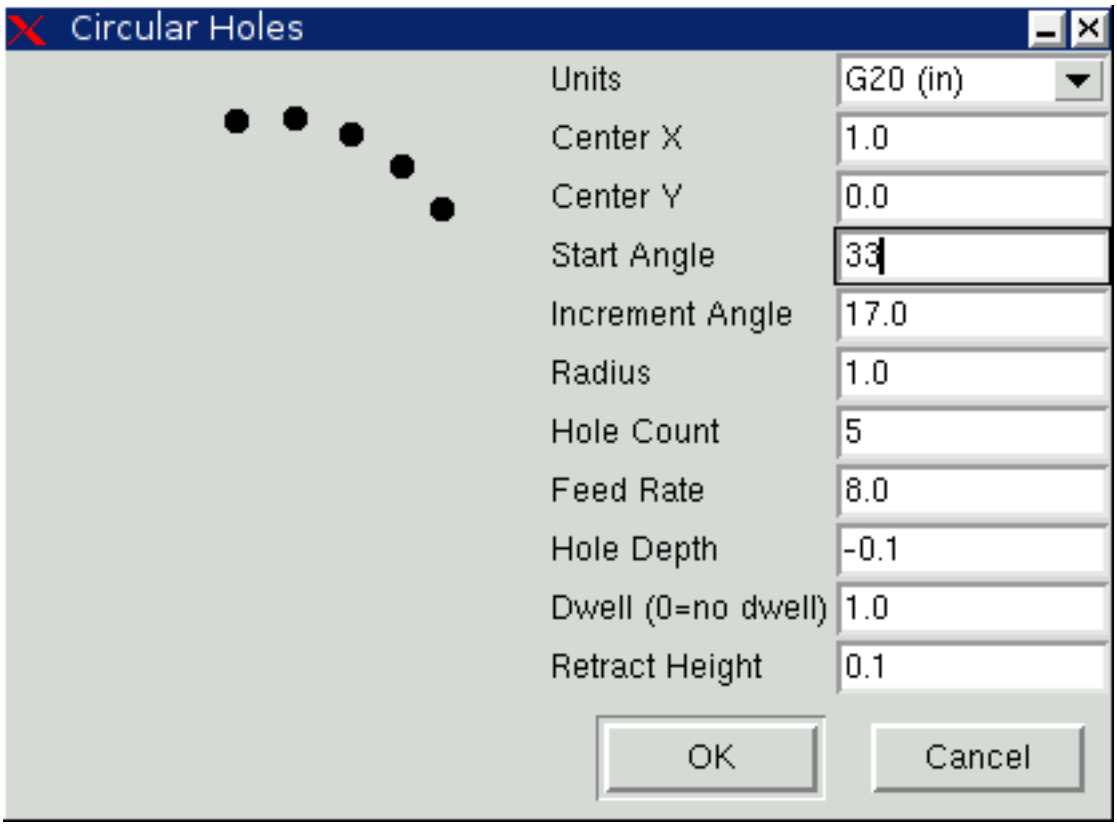



Abbildung 4.9: Kreisförmige Löcher

Wenn das Filterprogramm Zeilen in der folgenden Form an stderr sendet:

```
FILTER_PROGRESS=10
```

Sie setzt den Fortschrittsbalken des Bildschirms auf den angegebenen Prozentsatz (in diesem Fall 10). Diese Funktion sollte von jedem Filter verwendet werden, der lange läuft.

### 4.11.3 Erstellung von Filterprogrammen auf Python-Basis

Hier ist ein sehr einfaches Beispiel für die Filtermechanik: Wenn ein Linucnc-Bildschirm, der Programmfilterung bietet, durchläuft, wird jede 100stel Sekunde eine Zeile G-Code erzeugt und auf die Standardausgabe geschrieben. Außerdem sendet es eine Fortschrittsmeldung an den Unix-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_() )
```

# Kapitel 5

# HAL (Hardware Abstraction Layer)

## 5.1 HAL Introduction

LinuxCNC is about interacting with hardware. But few users have the same exact hardware specifications - similar, but not the same. And even for the exact same hardware, there may be different ways to use it, say for different materials or with different mills, which would require adaptations to the control of an already running system. An abstraction was needed to make it easier to configure LinuxCNC for a wide variety of hardware devices. At the highest level, it could simply be a way to allow a number of *building blocks* to be loaded and interconnected to assemble a complex system.

This chapter introduces to that Hardware Abstraction Layer. You will see that many of the building blocks are indeed, drivers for hardware devices. However, HAL can do more than just configure hardware drivers.

### 5.1.1 HAL Overview

The Hardware Abstraction Layer (or with a reference to the 2001 Space Odyssey movie just ”HAL”) is a software to

- provide the infrastructure for the communication with and between the many software and hardware components of the system.
- optionally process and/or override that information as it flows from component to component.

In itself, this Middleware is agnostic about its application on CNC. An Internet search, for example, found an astronomical application to control telescopes using LinuxCNC. Motors move the telescope into the right position, and it needs to know how to map motor activity with the effect of that positioning with the real world. Such a synchronisation of motor positions with real-world positions is reminiscent of what CNC machines need to do, or space craft.

Any machine controller needs to know:

- about its internal state and how this maps to the environment (machine coordinates, state of switches/regulators),
- how actuators are expected to change that state,
- how allow for updates of the internal state by sensors (encoders, probes).

The HAL layer consists of parts (referred to as ”components”) that

- are connected with each other, e.g., to update position data or have the planning algorithm tell the motors about the next step.
- may know how to communicate with hardware,
- may simply process incoming data and provide data outputs to other components,
- are always periodically executed either
  - with a very high frequency of a few microseconds (µs) execution time, called base thread, e.g., to
    1. give a stepper motor a trigger to step ahead, or to
    2. read out the position presented by an encoder.
  - with a lower frequency every millisecond (ms), e.g. to
    1. adjust the planning for the next moves to complete a G-code instruction.
  - as non-realtime ”user-space” components that run a ”main loop” just like any other software, and may be interrupted or delayed when the rest of the system is busy or overloaded.

Taken together, HAL allows

1. to program for a machine that the programmer does not know directly, but may rely on a programming interface with well-specified effect on the machine. That interface may be used to
   - tell the machine what to do
   - listen to what the machine wants to tell about the state it is in.
2. Vertical Abstractions: The human system integrator of such machine uses HAL
   - to describe what the machine is looking like and how what cable controls which motor that drives which axis.
   - The description of the machine, the programmer’s interfaces and the user’s interface somehow ”meet” in that abstract layer.
3. Horizontal Abstractions:
   - Not all machines have all kinds of features
   - Mills, Lathes and Robots share many
     - features (motors, joints, …),
     - planning algorithms for their movements.

HAL has no direct interaction with the user. But multiple interfaces have been provided that allow HAL to be manipulated

- from the command line using the ”halcmd” command.
- from Python scripts and
- from within C/C++ programs,

but none of these interfaces are *HAL itself*.

HAL itself is not a program, it consists of one or more lists of loaded programs (the components) that are periodically executed (in strict sequence), and an area of shared-memory that these components use to interchange data. The main HAL script runs only once at machine startup, setting up the real-time threads and the shared-memory locations, loading the components and setting up the data links between them (the ”signals” and ”pins”).

In principle multiple machines could share a common HAL to allow them to inter-operate, however the current implementation of LinuxCNC is limited to a single interpreter and a single Task module. Currently this is almost always a G-code interpreter and ”milltask” (which was found to also work well for lathes and adequately for robots) but these modules are selectable at load-time. With an increasing interest in the control of multiple cooperating machines, to overcome this limitation is likely one of the prime steps for the future development of LinuxCNC to address. It is a bit tricky though and the community is still organizing its thoughts on this.

HAL lies at the core of LinuxCNC and is used and/or extended by all the parts of LinuxCNC, which includes the GUIs. The G-code (or alternative language) interpreter knows how to interpret the G-code and translates it into machine operations by triggering signals in HAL. The user may query HAL in various ways to gain information about its state, which then also represents the state of the machine. Whilst writing during the development of version 2.9, the GUIs still make bit of an exception to that rule and may know something that HAL does not (need to) know.

### 5.1.2 Communication

HAL is special in that it can communicate really fast

- with other programs, but in particular
- with its components that typically run in one of the realtime threads.

And while communicating, the part of LinuxCNC that is talked to does not need to prepare for the communication: All these actions are performed asynchronously, i.e. no component is interrupting its regular execution to receive a signal and signals can be sent rightaway, i.e., an application may wait until a particular message has arrived - like an enable-signal, but it does not need to prepare for receiving that message.

The communication system

- represents and controls all the hardware attached to the system,
- starts and stops other communicating programs.

The communication with the hardware of the machine itself is performed by respective dedicated HAL components.

The HAL layer is a shared space in which all the many parts that constitute LinuxCNC are exchanging information. That space features pins that are identified by a name, though a LinuxCNC engineer may prefer the association with a pin of an electronic circuit. These pins can carry numerical and logical values, boolean, float and signed and unsigned integers. There is also a (relatively new) pin type named hal_port intended for byte streams, and a framework for exchanging more complex data called hal_stream (which uses a private shared memory area, rather than a HAL pin). These latter two types are used relatively infrequently.

With HAL you can send a signal to that named pin. Every part of HAL can read that pin that holds that value of the signal. That is until a new signal is sent to the same named pin to substitute the previous value. The core message exchange system of HAL is agnostic about CNC, but HAL ships with a large number of components that know a lot about CNC and present that information via pins. There are pins representing

- static information about the machine
- the current state of the machine
  - end switches
  - positions counted by steppers or as measured by encoders

- recipients for instructions
  - manual control of machine position (”jogging”)
  - positions that stepper motors should take next

In a analogy to electronic cables, pins can be wired, so the value changing in one pin serves as input to another pin. HAL components prepare such input and output pins and are thus automatically triggered to perform.

**HAL-Komponenten** The many ”expert” software parts of LinuxCNC are typically implemented as *components* of HAL, conceptually also referred to as *modules*. These computer-implemented experts perpetually read from HAL about a state that the machine should strive to achieve and compare that desired state with the state the machine is in at the current moment. When there is a difference between what should be and what the current state is then some action is performed to reduce that difference, while perpetually writing updates of the current states back to the HAL data space.

There are components specializing on how to talk to stepper motors, and other components know how to control servos. On a higher level, some components know how the machine’s axes are arranged in 3D and yet others know how to perform a smooth movement from one point in space to another. Lathes, mills and robots will differ in the LinuxCNC component that are active, i.e. that are loaded by a HAL configuration file for that machine. Still, two machines may be looking very different since built for very different purposes, but when they both use servo motors then they can still both use the same HAL servo component.

**Origin of the Incentive to Move** On the lowest (closest to hardware) level, e.g. for stepper motors, the description of a state of that motor is very intuitive: It is the number of steps in a particular direction. A difference between the desired position and the actual position translates into a movement. Speeds, acceleration and other parameters may be internally limited in the component itself, or may optionally be limited by upstream components. (For example, in most cases the moment-by-moment axis position values sent to the step-generator components have already been limited and shaped to suit the configured machine limits or the current feed rate.)

Any G-code line is interpreted and triggers a set of routines that in turn know how to communicate with components that are on a middle layer, e.g., to create a circle.

**Pins and Signals** HAL has a special place in the heart of its programmers for the way that the data flow between modules is represented. When traditional programmers think of variables, addresses or I/O ports, HAL refers to ”pins”. And those pins are connected or assigned values to via signals. Much like an electrical engineer would connect wires between pins of components of a mill, a HAL engineer establishes the data flow between pins of module instances.

The LinuxCNC GUIS (AXIS, GMOCCAPY, Touchy, etc.) will represent the states of some pins (such as limit switches) but other graphical tools also exist for troubleshooting and configuration: Halshow, Halmeter, Halscope and Halreport.

The remainder of this introduction presents

- the syntax of how pins of different components are connected in the HAL configuration files, and
- software to inspect the values of pins
  - at any given moment,
  - developing over time.

### 5.1.3 HAL System Design

.HAL basiert auf traditionellen Systementwurfstechniken.

HAL is based on the same principles that are used to design hardware circuits and systems, so it is useful to examine those principles first. Any system, including a CNC machine, consists of interconnected components. For the CNC machine, those components might be the main controller, servo

amps or stepper drives, motors, encoders, limit switches, pushbutton pendants, perhaps a VFD for the spindle drive, a PLC to run a toolchanger, etc. The machine builder must aselect, mount and wire these pieces together to make a complete system.

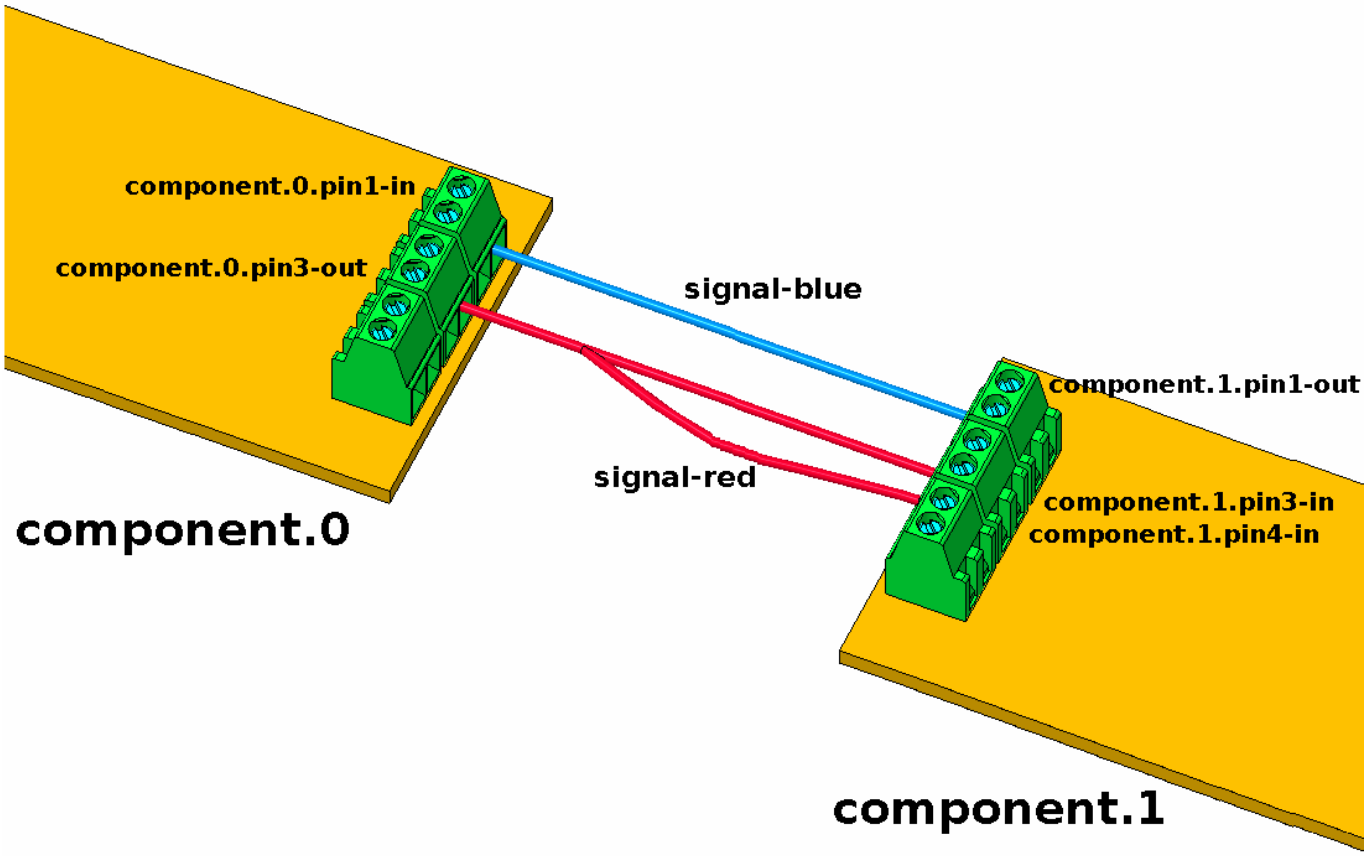



Abbildung 5.1: HAL-Konzept - Verbinden wie elektrische Schaltkreise.

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
```

#### 5.1.3.1 Part Selection

The machine builder does not need to worry about how each individual part works. He treats them as black boxes. During the design stage, he decides which parts he is going to use - steppers or servos, which brand of servo amp, what kind of limit switches and how many, etc. The integrator’s decisions about which specific components to use is based on what that component does and the specifications supplied by the manufacturer of the device. The size of a motor and the load it must drive will affect the choice of amplifier needed to run it. The choice of amplifier may affect the kinds of feedback needed by the amp and the velocity or position signals that must be sent to the amp from a control.

In the HAL world, the integrator must decide what HAL components are needed. Usually every interface card will require a driver. Additional components may be needed for software generation of step pulses, PLC functionality, and a wide variety of other tasks.

#### 5.1.3.2 Interconnection Design

The designer of a hardware system not only selects the parts, he also decides how those parts will be interconnected. Each black box has terminals, perhaps only two for a simple switch, or dozens for a servo drive or PLC. They need to be wired together. The motors connect to the servo amps, the limit switches connect to the controller, and so on. As the machine builder works on the design, he creates a large wiring diagram that shows how all the parts should be interconnected.

When using HAL, components are interconnected by signals. The designer must decide which signals are needed, and what they should connect.

#### 5.1.3.3 Implementation

Once the wiring diagram is complete it is time to build the machine. The pieces need to be acquired and mounted, and then they are interconnected according to the wiring diagram. In a physical system, each interconnection is a piece of wire that needs to be cut and connected to the appropriate terminals.

HAL provides a number of tools to help *build* a HAL system. Some of the tools allow you to *connect* (or disconnect) a single *wire*. Other tools allow you to save a complete list of all the parts, wires, and other information about the system, so that it can be *rebuilt* with a single command.

#### 5.1.3.4 Testen

Very few machines work right the first time. While testing, the builder may use a meter to see whether a limit switch is working or to measure the DC voltage going to a servo motor. He may hook up an oscilloscope to check the tuning of a drive, or to look for electrical noise. He may find a problem that requires the wiring diagram to be changed; perhaps a part needs to be connected differently or replaced with something completely different.

HAL provides the software equivalents of a voltmeter, oscilloscope, signal generator, and other tools needed for testing and tuning a system. The same commands used to build the system can be used to make changes as needed.

#### 5.1.3.5 Summary

This document is aimed at people who already know how to do this kind of hardware system integration, but who do not know how to connect the hardware to LinuxCNC. See the Remote Start Example section in the HAL UI Examples documentation.

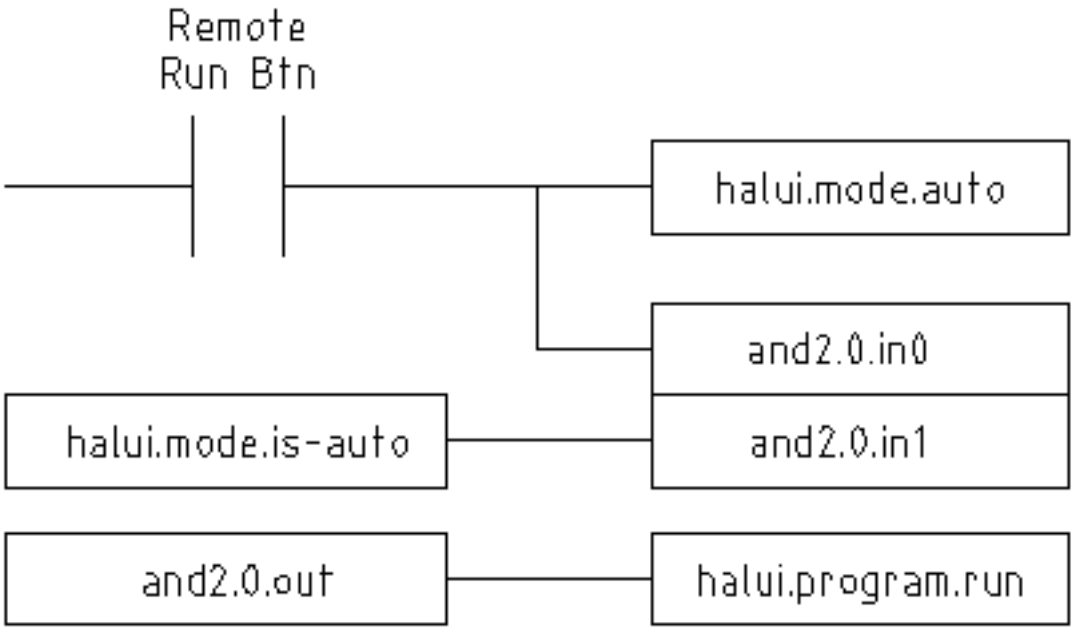



Abbildung 5.2: Remote-Start-Beispiel (Schema)

The traditional hardware design as described above ends at the edge of the main control. Outside the control are a bunch of relatively simple boxes, connected together to do whatever is needed. Inside, the control is a big mystery — one huge black box that we hope works.

HAL extends this traditional hardware design method to the inside of the big black box. It makes device drivers and even some internal part of the controller into smaller black boxes that can be interconnected and even replaced just like the external hardware. It allows the *system wiring diagram* to show part of the internal controller, rather than just a big black box. And most importantly, it allows the integrator to test and modify the controller using the same methods he would use on the rest of the hardware.

Terms like motors, amps, and encoders are familiar to most machine integrators. When we talk about using extra flexible eight conductor shielded cable to connect an encoder to the servo input board in the computer, the reader immediately understands what it is and is led to the question, *what kinds of connectors will I need to make up each end.* The same sort of thinking is essential for the HAL but the specific train of thought may take a bit to get on track. Using HAL words may seem a bit strange at first, but the concept of working from one connection to the next is the same.

This idea of extending the wiring diagram to the inside of the controller is what HAL is all about. If you are comfortable with the idea of interconnecting hardware black boxes, you will probably have little trouble using HAL to interconnect software black boxes.

### 5.1.4 HAL Concepts

This section is a glossary that defines key HAL terms but it is a bit different than a traditional glossary because these terms are not arranged in alphabetical order. They are arranged by their relationship or flow in the HAL way of things.

**Komponente**
: When we talked about hardware design, we referred to the individual pieces as *parts*, *building blocks*, *black boxes*, etc. The HAL equivalent is a *component* or *HAL component*. This document uses *HAL component* when there is likely to be confusion with other kinds of components, but normally just uses *component*. A HAL component is a piece of software with well-defined inputs,

outputs, and behavior, that can be installed and interconnected as needed. + + Many HAL Components model the behaviour of a tangible part of a machine, and a **pin** may indeed be meant to be connected to a **physical pin** on the device to communicate with it, hence the names. But most often this is not the case. Imagine a retrofit of a manual lathe/mill. What LinuxCNC implements is how the machine presents itself to the outside world, and it is secondary if the implementation how to draw a circle is implemented on the machine already or provided from LinuxCNC. And it is common to add buttons to the imaginary retrofit that **signal** an action, like an emergency stop. LinuxCNC and the machine become one. And that is through the HAL.

**Parameter**
Many hardware components have adjustments that are not connected to any other components but still need to be accessed. For example, servo amps often have trim pots to allow for tuning adjustments, and test points where a meter or scope can be attached to view the tuning results. HAL components also can have such items, which are referred to as *parameters*. There are two types of parameters: Input parameters are equivalent to trim pots - they are values that can be adjusted by the user, and remain fixed once they are set. Output parameters cannot be adjusted by the user - they are equivalent to test points that allow internal signals to be monitored.

**Pin**
Hardware components have terminals which are used to interconnect them. The HAL equivalent is a *pin* or *HAL pin*. *HAL pin* is used when needed to avoid confusion. All HAL pins are named, and the pin names are used when interconnecting them. HAL pins are software entities that exist only inside the computer.

**Physikalischer Pin (engl. physical pin)**
Many I/O devices have real physical pins or terminals that connect to external hardware, for example the pins of a parallel port connector. To avoid confusion, these are referred to as *physical pins*. These are the things that *stick out* into the real world.

---

**Anmerkung**
You may be wondering what relationship there is between the HAL_pins, physical_pins and external elements like encoders or a STG card: we are dealing here with interfaces of data translation/conversion type.

---

**Signal**
In a physical machine, the terminals of real hardware components are interconnected by wires. The HAL equivalent of a wire is a *signal* or *HAL signal*. HAL signals connect HAL pins together as required by the machine builder. HAL signals can be disconnected and reconnected at will (even while the machine is running).

**Typ**
When using real hardware, you would not connect a 24 Volt relay output to the +/-10 V analog input of a servo amp. HAL pins have the same restrictions, which are based upon their type. Both pins and signals have types, and signals can only be connected to pins of ffvthe same type. Currently there are 4 types, as follows:

- 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**
Real hardware components tend to act immediately on their inputs. For example, if the input

voltage to a servo amp changes, the output also changes automatically. However software components cannot act *automatically*. Each component has specific code that must be executed to do whatever that component is supposed to do. In some cases, that code simply runs as part of the component. However in most cases, especially in realtime components, the code must run in a specific sequence and at specific intervals. For example, inputs should be read before calculations are performed on the input data, and outputs should not be written until the calculations are done. In these cases, the code is made available to the system in the form of one or more *functions*. Each function is a block of code that performs a specific action. The system integrator can use *threads* to schedule a series of functions to be executed in a particular order and at specific time intervals.

**Thread**
A *thread* is a list of functions that runs at specific intervals as part of a realtime task. When a thread is first created, it has a specific time interval (period), but no functions. Functions can be added to the thread, and will be executed in order every time the thread runs.

As an example, suppose we have a parport component named hal_parport. That component defines one or more HAL pins for each physical pin. The pins are described in that component's doc section: Their names, how each pin relates to the physical pin, are they inverted, can you change polarity, etc. But that alone doesn't get the data from the HAL pins to the physical pins. It takes code to do that, and that is where functions come into the picture. The parport component needs at least two functions: One to read the physical input pins and update the HAL pins, the other to take data from the HAL pins and write it to the physical output pins. Both of these functions are part of the parport driver.

### 5.1.5 HAL components

Each HAL component is a piece of software with well-defined inputs, outputs, and behavior, that can be installed and interconnected as needed. The section HAL Components List lists all available components and a brief description of what each does.

### 5.1.6 Timing Issues In HAL

Unlike the physical wiring models between black boxes that we have said that HAL is based upon, simply connecting two pins with a HAL-signal falls far short of the action of the physical case.

True relay logic consists of relays connected together, and when a contact opens or closes, current flows (or stops) immediately. Other coils may change state, etc., and it all just *happens*. But in PLC style ladder logic, it doesn't work that way. Usually in a single pass through the ladder, each rung is evaluated in the order in which it appears, and only once per pass. A perfect example is a single rung ladder, with a NC contact in series with a coil. The contact and coil belong to the same relay.

If this were a conventional relay, as soon as the coil is energized, the contacts begin to open and de-energize it. That means the contacts close again, etc., etc. The relay becomes a buzzer.

With a PLC, if the coil is OFF and the contact is closed when the PLC begins to evaluate the rung, then when it finishes that pass, the coil is ON. The fact that turning on the coil opens the contact feeding it is ignored until the next pass. On the next pass, the PLC sees that the contact is open, and de-energizes the coil. So the relay still switches rapidly between on and off, but at a rate determined by how often the PLC evaluates the rung.

In HAL, the function is the code that evaluates the rung(s). In fact, the HAL-aware realtime version of ClassicLadder exports a function to do exactly that. Meanwhile, a thread is the thing that runs the function at specific time intervals. Just like you can choose to have a PLC evaluate all its rungs every 10 ms, or every second, you can define HAL threads with different periods.

What distinguishes one thread from another is *not* what the thread does - that is determined by which functions are connected to it. The real distinction is simply how often a thread runs.

In LinuxCNC you might have a 50 µs thread and a 1 ms thread. These would be created based on BASE_PERIOD and SERVO_PERIOD, the actual times depend on the values in your INI file.

The next step is to decide what each thread needs to do. Some of those decisions are the same in (nearly) any LinuxCNC system. For instance, motion-command-handler is always added to servo-thread.

Other connections would be made by the integrator. These might include hooking the STG driver's encoder read and DAC write functions to the servo thread, or hooking StepGen's function to the base-thread, along with the parport function(s) to write the steps to the port.

## 5.2 HAL Basics

This document provides a reference to the basics of HAL.

### 5.2.1 HAL Commands

More detailed information can be found in the man page for halcmd: run *man halcmd* in a terminal window.

To see the HAL configuration and check the status of pins and parameters use the HAL Configuration window on the Machine menu in AXIS. To watch a pin status open the Watch tab and click on each pin you wish to watch and it will be added to the watch window.

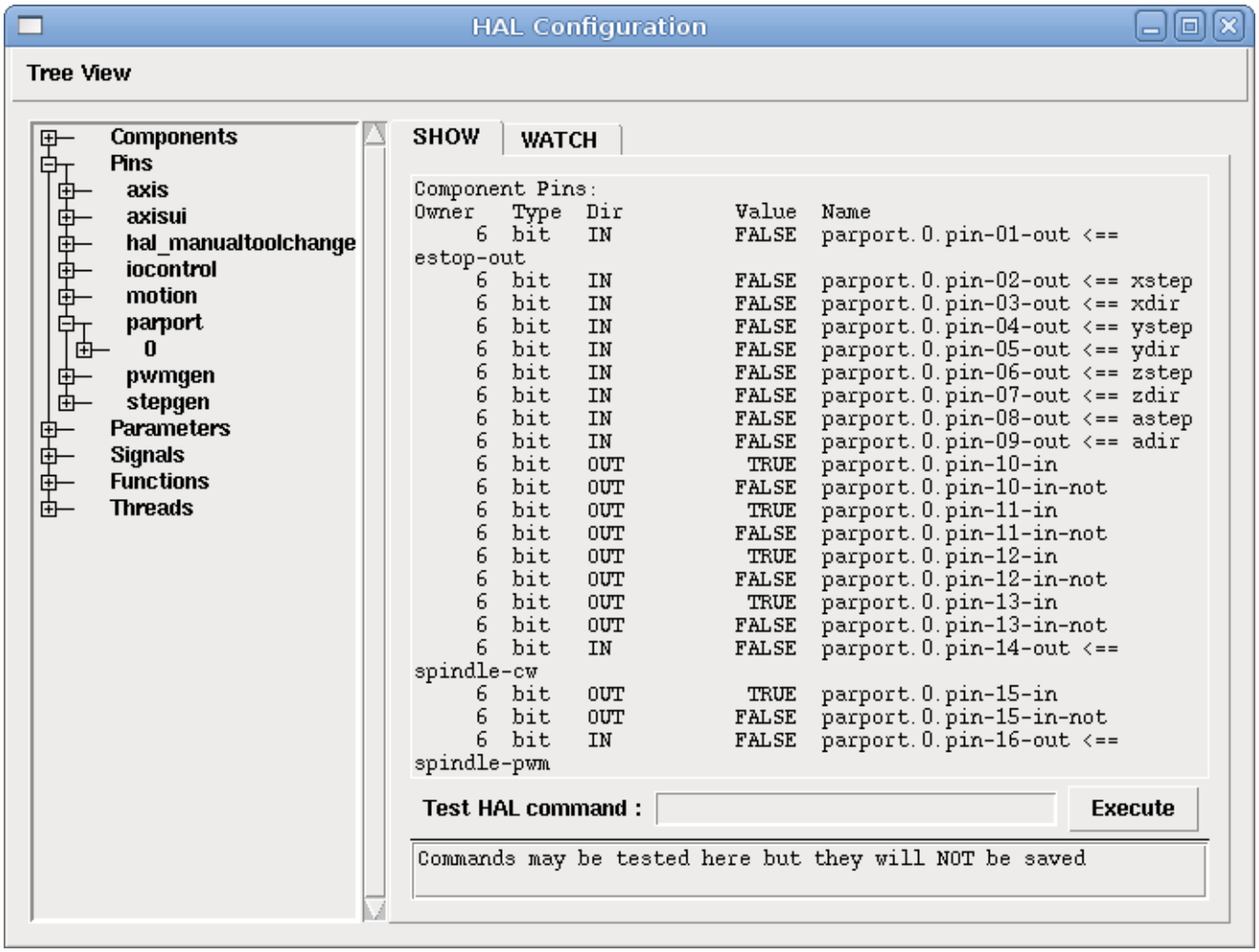



Abbildung 5.3: HAL-Konfigurationsfenster

#### 5.2.1.1 loadrt

The command `loadrt` loads a real time HAL component. Real time component functions need to be added to a thread to be updated at the rate of the thread. You cannot load a non-realtime component into the realtime space.

**loadrt Syntax und Beispiel**

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

#### 5.2.1.2 addf

The `addf` command adds a function to a real-time thread. If the StepConf wizard was used to create the configuration, two threads have been created (``base-thread`` and ``servo-thread``).

`addf` adds function *functname* to thread *threadname*. Default is to add the function in the order they are in the file. If *position* is specified, adds the function to that spot in the thread. A negative *position* indicates the position with respect to the end of the thread. For example *1* is start of thread, *-1* is the end of the thread, *-3* is third from the end.

For some functions it is important to load them in a certain order, like the parport read and write functions. The function name is usually the component name plus a number. In the following example the component *or2* is loaded and `show function` shows the name of the or2 function.

```
$ halrun
halcmd: loadrt or2
halcmd: show function
Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f8bc5000  f8f950c8  NO       0   or2.0
```

You have to add a function from a HAL real time component to a thread to get the function to update at the rate of the thread. Usually there are two threads as shown in this example. Some components use floating point math and must be added to a thread that supports floating point math. The FP indicates if floating point math is supported in that thread.

```
$ 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 )
```

- base-thread (the high-speed thread): This thread handles items that need a fast response, like making step pulses, and reading and writing the parallel port. Does not support floating point math.
- servo-thread (the slow-speed thread): This thread handles items that can tolerate a slower response, like the motion controller, ClassicLadder, and the motion command handler and supports floating point math.

**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

The command `loadusr` loads a non-realtime HAL component. Non-realtime programs are their own separate processes, which optionally talk to other HAL components via pins and parameters. You cannot load realtime components into non-realtime space.

Flags können eine oder mehrere der folgenden sein:

| | |
|---|---|
| -W | um auf die Bereitschaft der Komponente zu warten. Es wird davon ausgegangen, dass die Komponente denselben Namen hat wie das erste Argument des Befehls. |
| -Wn <Name> | um auf die Komponente zu warten, die den angegebenen <Name> haben wird. Dies gilt nur, wenn die Komponente eine Namensoption hat. |

| | |
|---|---|
| -w | um zu warten, bis das Programm beendet wird |
| -i | um den Rückgabewert des Programms zu ignorieren (mit -w) |
| -n | Benennt eine Komponente, sofern dies eine zulässige Option für diese Komponente ist. |

**Syntax and Examples of loadusr**

```
loadusr <component> <options>
loadusr halui
loadusr -Wn spindle gs2_vfd -n spindle
```

Auf Deutsch bedeutet es *loadusr wartet auf Name Spindel Komponente gs2_vfd mit Namen Spindel*.

#### 5.2.1.4 net

The command `net` creates a *connection* between a signal and one or more pins. If the signal does not exist net creates the new signal. This replaces the need to use the command newsig. The optional direction arrows `<=`, `=>` and `<=>` make it easier to follow the logic when reading a `net` command line and are not used by the net command. The direction arrows must be separated by a space from the pin names.

**Syntax and Examples of net**

```
net signal-name pin-name <optional arrow> <optional second pin-name>
net home-x joint.0.home-sw-in <= parport.0.pin-11-in
```

In the above example `home-x` is the signal name, `joint.0.home-sw-in` is a *Direction IN* pin, `<=` is the optional direction arrow, and `parport.0.pin-11-in` is a *Direction OUT* pin. This may seem confusing but the in and out labels for a parallel port pin indicates the physical way the pin works not how it is handled in HAL.

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.

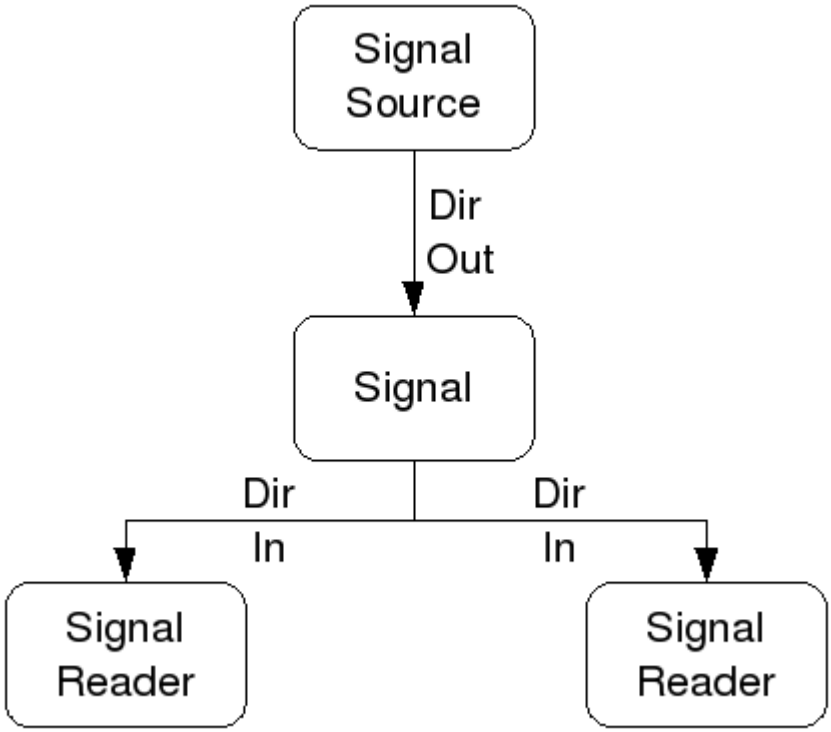



Abbildung 5.4: Signalrichtung (engl. signal direction)

This example shows the signal xStep with the source being `stepgen.0.out` and with two readers, `parport.0.pin-02-out` and `parport.0.pin-08-out`. Basically the value of `stepgen.0.out` is sent to the signal xStep and that value is then sent to `parport.0.pin-02-out` and `parport.0.pin-08-out`.

```
#   Signal    Ursprung           Destination          Destination
net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out
```

Since the signal xStep contains the value of `stepgen.0.out` (the source) you can use the same signal again to send the value to another reader. To do this just use the signal with the readers on another line.

```
#   Signal       Destination2
net xStep => parport.0.pin-06-out
```

**E/A-Pins (engl. I/O Pins)** An I/O pin like encoder.*N*.index-enable can be read or set as allowed by the component.

#### 5.2.1.5 setp

The command `setp` sets the value of a pin or parameter. The valid values will depend on the type of the pin or parameter. It is an error if the data types do not match.

Some components have parameters that need to be set before use. Parameters can be set before use or while running as needed. You cannot use setp on a pin that is connected to a signal.

**Syntax and Examples of setp**

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

#### 5.2.1.6 sets

The command `sets` sets the value of a signal.

**Syntax and Examples of sets**

```
sets <signal-name> <value>
net mysignal and2.0.in0 pyvcp.my-led
sets mysignal 1
```

Es ist ein Fehler, wenn:

- Der Signal-Name existiert nicht
- Wenn das Signal bereits einen Schreiber (engl. writer) hat
- Wenn Wert nicht der richtige Typ für das Signal ist

#### 5.2.1.7 unlinkp

The command `unlinkp` unlinks a pin from the connected signal. If no signal was connected to the pin prior running the command, nothing happens. The `unlinkp` command is useful for trouble shooting.

**Syntax and Examples of unlinkp**

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

#### 5.2.1.8 Veraltete Befehle

The following commands are depreciated and may be removed from future versions. Any new configuration should use the net command. These commands are included so older configurations will still work.

The command `linksp` creates a *connection* between a signal and one pin.

**Syntax and Examples of linksp**

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

**The `linksp` command has been superseded by the `net` command.**

The command `linkps` creates a *connection* between one pin and one signal. It is the same as `linksp` but the arguments are reversed.

**Syntax and Examples of linkps**

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

**The `linkps` command has been superseded by the `net` command.**

the command `newsig` creates a new HAL signal by the name <*signame*> and the data type of <*type*>. Type must be *bit*, *s32*, *u32* or *float*. Error if <*signame*> already exists.

**Syntax and Examples of newsig**

```
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

A bit value is an on or off.

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

### 5.2.2.2 Gleitkommazahl (engl. float)

A *float* is a floating point number. In other words the decimal point can move as needed.

- 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:

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

### 5.2.2.3 s32

An *s32* number is a whole number that can have a negative or positive value.

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

### 5.2.2.4 u32

A *u32* number is a whole number that is positive only.

- u32-Werte = Ganzzahlige Zahlen von 0 bis 4294967295

## 5.2.3 HAL Files

If you used the Stepper Config Wizard to generate your config you will have up to three HAL files in your config directory.

- *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* This file is loaded next and before the GUI loads. This is where you put your custom HAL commands that you want loaded before the GUI is loaded.
- *custom_postgui.hal* This file is loaded after the GUI loads. This is where you put your custom HAL commands that you want loaded after the GUI is loaded. Any HAL commands that use PyVCP widgets need to be placed here.

### 5.2.4 HAL Parameter

Two parameters are automatically added to each HAL component when it is created. These parameters allow you to scope the execution time of a component.

`.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 Basic Logic Components

HAL contains several real time logic components. Logic components follow a *Truth Table* that states what the output is for any given input. Typically these are bit manipulators and follow electrical logic gate truth tables.

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

### 5.2.5.1 and2

The `and2` component is a two input and-gate. The truth table below shows the output based on each combination of input.

**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: Truth Table of `and2`

| in0 | in1 | out |
|---|---|---|
| False | False | False |
| True | False | False |
| False | True | False |
| True | True | True |

### 5.2.5.2 not

The `not` component is a 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: Truth Table of `not`

| in | out |
|---|---|
| True | False |
| False | True |

#### 5.2.5.3 or2

The `or2` component is a two input or-gate.

**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

The `xor2` component is a two input xor (exclusive or)-gate.

**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 Logic Examples

.Example using `and2`

```
loadrt and2 count=1
addf and2.0 servo-thread
net my-sigin1 and2.0.in0 <= parport.0.pin-11-in
net my-sigin2 and2.0.in1 <= parport.0.pin-12-in
net both-on parport.0.pin-14-out <= and2.0.out
```

In the above example one copy of `and2` is loaded into real time space and added to the servo thread. Next `pin-11` of the parallel port is connected to the `in0` bit of the and gate. Next `pin-12` is connected to the `in1` bit of the and gate. Last we connect the `and2` out bit to the parallel port `pin-14`. So following the truth table for `and2` if pin 11 and pin 12 are on then the output pin 14 will be on.

### 5.2.7 Conversion Components

#### 5.2.7.1 weighted_sum

The weighted sum converts a group of bits into an integer. The conversion is the sum of the *weights* of the bits present plus any offset. It’s similar to *binary coded decimal* but with more options. The *hold* bit interrupts the input processing, so that the *sum* value no longer changes.

**Syntax for loading component weighted_sum**

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

Creates groups of ``weighted_sum``s, each with the given number of input bits (size).

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

**Add process_wsums to servo thread**

```
addf process_wsums servo-thread
```

Which updates the `weighted_sum` component.

In the following example, a copy of the AXIS HAL configuration window, bits *0* and *2* are TRUE, they have no offset. The weight (*weight*) of bit 0 is 1, that of bit 2 is 4, so the sum is 5.

Tabelle 5.5: Component pins of `weighted_sum`

| Owner | Typ | Dir | 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

This section describes an option to have multiple load-commands for multiple instances of the same component at different positions in the file or among different files. Internally, this requires to read the HAL file twice, hence the name TWOPASS. Supported since LinuxCNC version 2.5, the TWOPASS processing of LinuxCNC configuration files helps with their modularization and readability. To recall, LinuxCNC configuration files are specified in a LinuxCNC INI file as `[HAL]HALFILE=filename`.

Normally, a set of one or more LinuxCNC configuration files must use a single, unique `loadrt` line to load a realtime component, which may create multiple instances of the component. For example, if you use a two-input AND gate component (and2) in three different places in your setup, you would need to have a single line somewhere to specify:

**Example resulting in real-time components with default names and2.0, and2.1, and2.2.**

```
loadrt and2 count=3
```

Configurations are more readable if you specify with the `names=` option for components where it is supported, e.g.:

**Example load command resulting in explicitly named components aa, ab, ac.**

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

It can be a maintenance problem to keep track of the components and their names, since when you add (or remove) a component, you must find and update the single loadrt directive applicable to the component.

**TWOPASS processing is enabled by including an INI file parameter in the [HAL] section, where ”anystring” can be any non-null string.**

```
[HAL]

TWOPASS = anystring
```

With TWOPASS enabled, you can have multiple specifications like:

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

These commands can appear in different HAL files. The HAL files are processed in the order of their appearance in the INI file, in multiple HALFILE assignments.

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
```

With TWOPASS processing, all [HAL]HALFILES are first read and multiple appearances of loadrt directives for each module are accumulated. Non-realtime components (loadusr) are loaded in order but no other LinuxCNC commands are executed in the initial pass.

---

**Anmerkung**
Non-realtime components should use the wait (-W) option to ensure the component is ready before other commands are executed.

---

After the initial pass, the realtime modules are loaded (loadrt) automatically

- with a number equal to the total number when using the *count*= option or
- with all of the individual names specified when using the *names*= option.

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.

Many comps supplied with the distribution are created with the halcompile utility and support the *names*= option. These include the common logic components that are the glue of many LinuxCNC configurations.

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.

Two-step processing occurs before the GUI is loaded. When using a [HAL]POSTGUI_HALFILE, it is convenient to place all the [HAL]POSTGUI_HALFILE loadrt declarations for the necessary components in a preloaded HAL file.

**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)

Some GUIs support HAL files that are processed after the GUI is started in order to connect LinuxCNC pins that are created by the GUI. When using a postgui HAL file with TWOPASS processing, include all loadrt items for components added by postgui HAL files in a separate HAL file that is processed before the GUI. The `addf` commands can also be included in the file.

**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

TWOPASS processing converts *.hal* files to equivalent *.tcl* files and uses haltcl to find loadrt and addf commands in order to accumulate and consolidate their usage. Loadrt parameters that conform to the simple `names=` (or `count=`) parameters accepted by the HAL Component Generator (`halcompile`) are expected. More complex parameter items included in specialized LinuxCNC components may not be handled properly.

A *.hal* file may be excluded from TWOPASS processing by including a magic comment line anywhere in the *.hal* file. The magic comment line must begin with the string: `#NOTWOPASS`. Files specified with this magic comment are sourced by halcmd using the `-k` (keep going if failure) and `-v` (verbose) options.

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.

**Excluded HAL file example**

```
$ 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**
Case and whitespace within the magic comment are ignored. The loading of components that use `names=` or `count=` parameters (typically built by halcompile) should not be used in excluded files, as that would eliminate the benefits of TWOPASS processing. The LinuxCNC commands that create signals (`net`) and commands that establish execution order (`addf`) should not be placed in excluded files. This is especially true for addf commands since their ordering may be important.

### 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

Configuration moves from theory to device — HAL device that is. For those who have had just a bit of computer programming, this section is the *Hello World* of the 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` is a command line tool for manipulating HAL. A more complete man page exists for halcmd and installed together with LinuxCNC, from source or from a package. If LinuxCNC has been compiled as *run-in-place*, the man page is not installed but is accessible in the LinuxCNC main directory with the following command:

```
$ 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. 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 non-realtime 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 )
```

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

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:

```
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

Das meiste, was wir bisher mit `halcmd` gemacht haben, war einfach das Anzeigen von Dingen mit dem *show*-Befehl. Zwei der Befehle haben jedoch tatsächlich Dinge verändert. Wenn wir komplexere Systeme mit HAL entwerfen, werden wir viele Befehle verwenden, um die Dinge genau so zu konfigurieren, wie wir sie haben wollen. HAL hat ein Gedächtnis wie ein Elefant und behält diese Konfiguration bei, bis wir es abschalten. Aber was ist beim nächsten Mal? Wir wollen nicht jedes Mal, wenn wir das System benutzen wollen, eine Reihe von Befehlen manuell eingeben.

**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

Wenn eine HAL-Sitzung unerwartet beendet wird, müssen Sie möglicherweise HAL entladen, bevor eine neue Sitzung beginnen kann. Geben Sie dazu den folgenden Befehl in ein Terminalfenster ein.

**Removing HAL**

```
halrun -U
```

### 5.4.4 Halmeter

You can build very complex HAL systems without ever using a graphical interface. However there is something satisfying about seeing the result of your work. The first and simplest GUI tool for the HAL is halmeter. It is a very simple program that is the HAL equivalent of the handy multimeter (or analog meter for the old timers).

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.

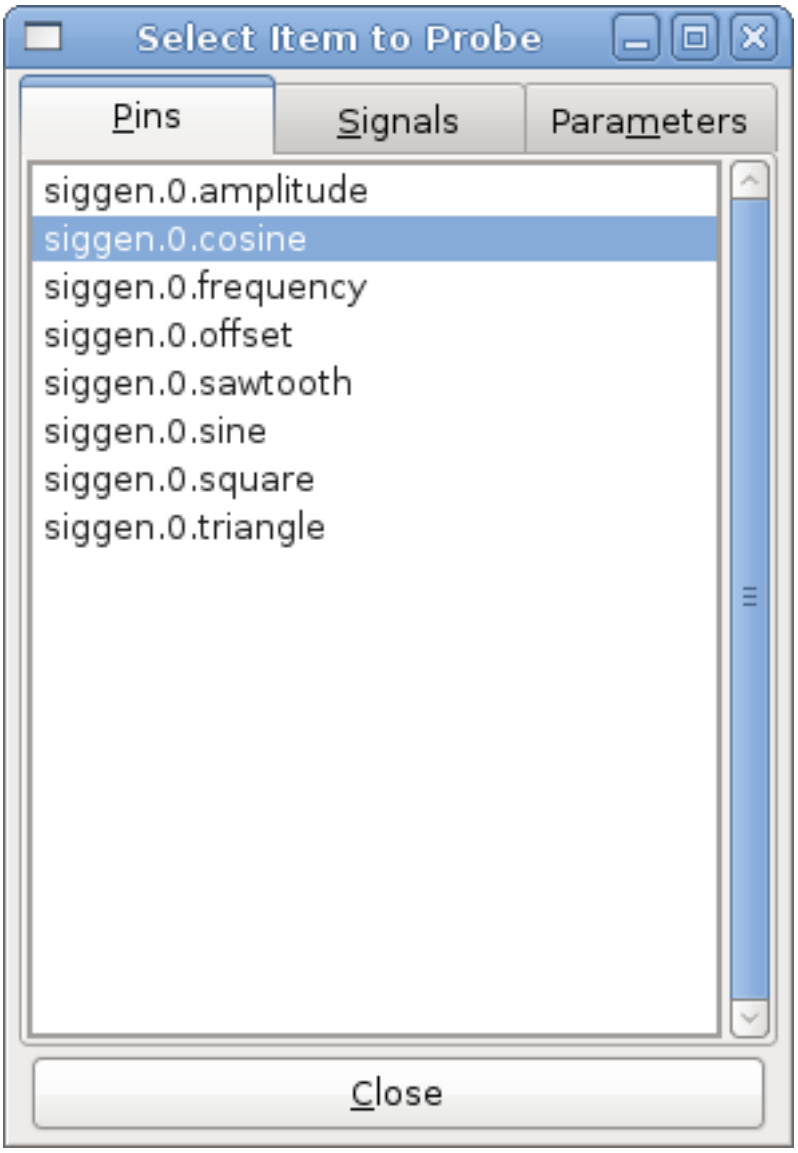



Abbildung 5.5: Halmeter Auswahlfenster

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 Example

Up till now we have only loaded one HAL component. But the whole idea behind the HAL is to allow you to load and connect a number of simple components to make up a complex system. The next example will use two components.

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.

Für unser Beispiel bedeutet dies, dass wir `siggen.0.update` mit einer moderaten Rate ausführen wollen, um die Sinus- und Kosinuswerte zu berechnen. Unmittelbar nachdem wir `siggen.0.update` ausgeführt haben, wollen wir `stepgen.update_freq` ausführen, um die neuen Werte in den Schrittimpulsgenerator zu laden. Schließlich müssen wir `stepgen.make_pulses` so schnell wie möglich ausführen, um gleichmäßige Impulse zu erhalten. Da wir keine Positionsrückmeldung verwenden, brauchen wir `stepgen.capture_position` überhaupt nicht auszuführen.

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

The previous example generates some very interesting signals. But much of what happens is far too fast to see with halmeter. To take a closer look at what is going on inside the HAL, we want an oscilloscope. Fortunately HAL has one, called halscope.

Halscope has two parts - a realtime part that reads the HAL signals, and a non-realtime part that provides the GUI and display. However, you don't need to worry about this because the non-realtime part will automatically load the realtime part when needed.

With LinuxCNC running in a terminal you can start halscope with the following command.

**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.

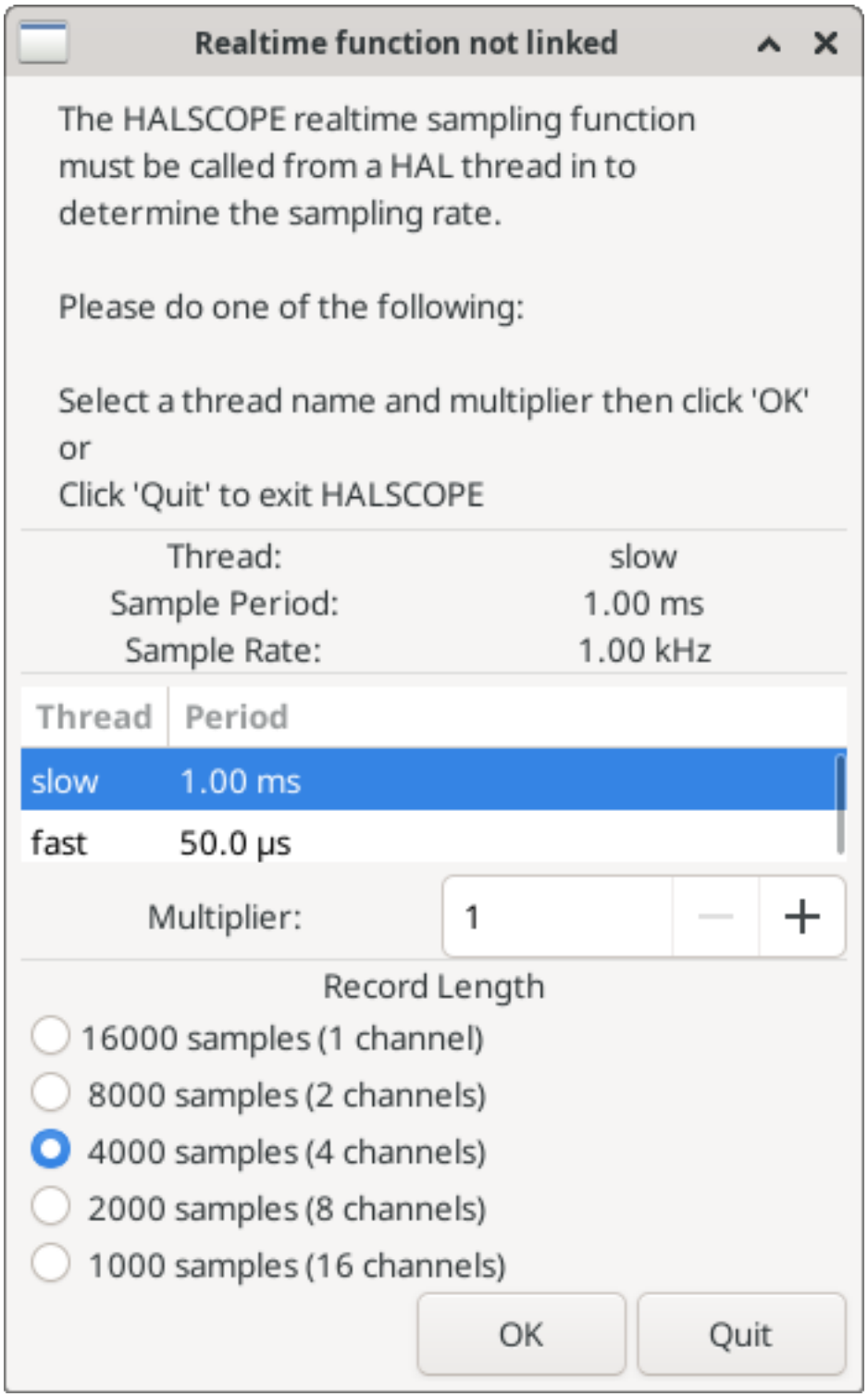



Abbildung 5.7: Dialog *Echtzeitfunktion nicht verknüpft*

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 1.00 ms 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.

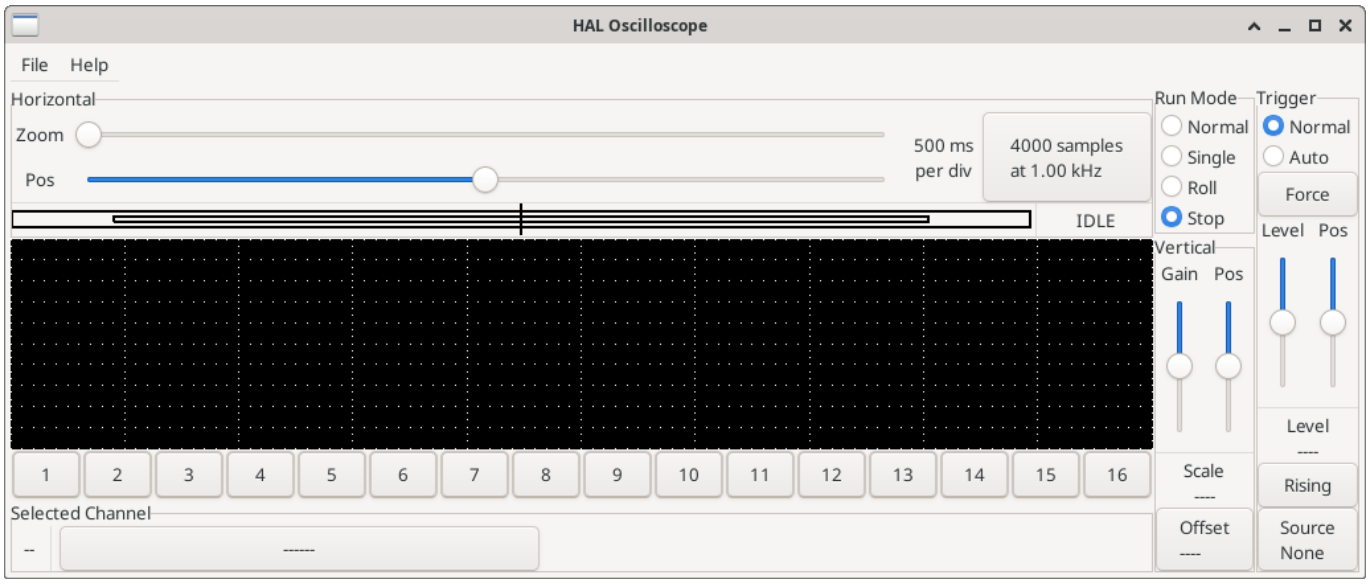



Abbildung 5.8: Fenster für den anfänglichen Geltungsbereich

#### 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).

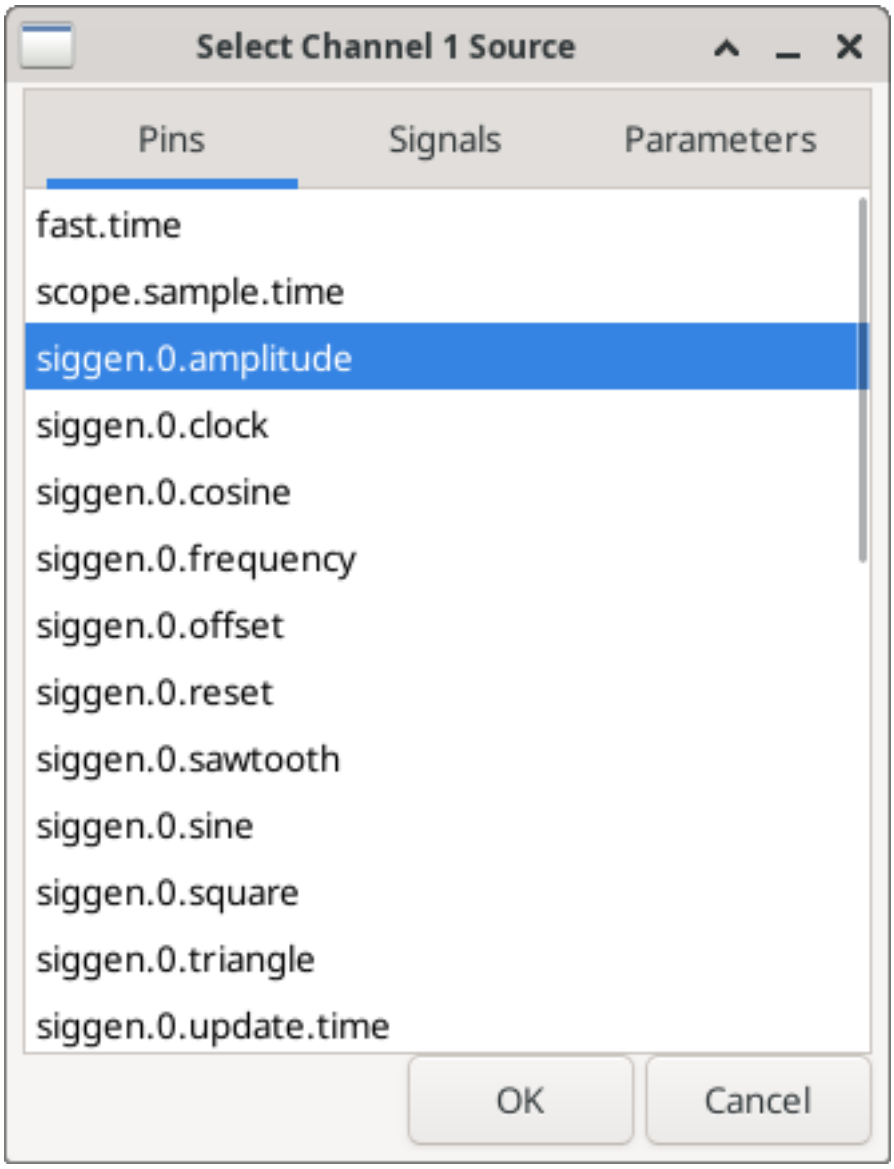



Abbildung 5.9: Kanalquelle auswählen

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.

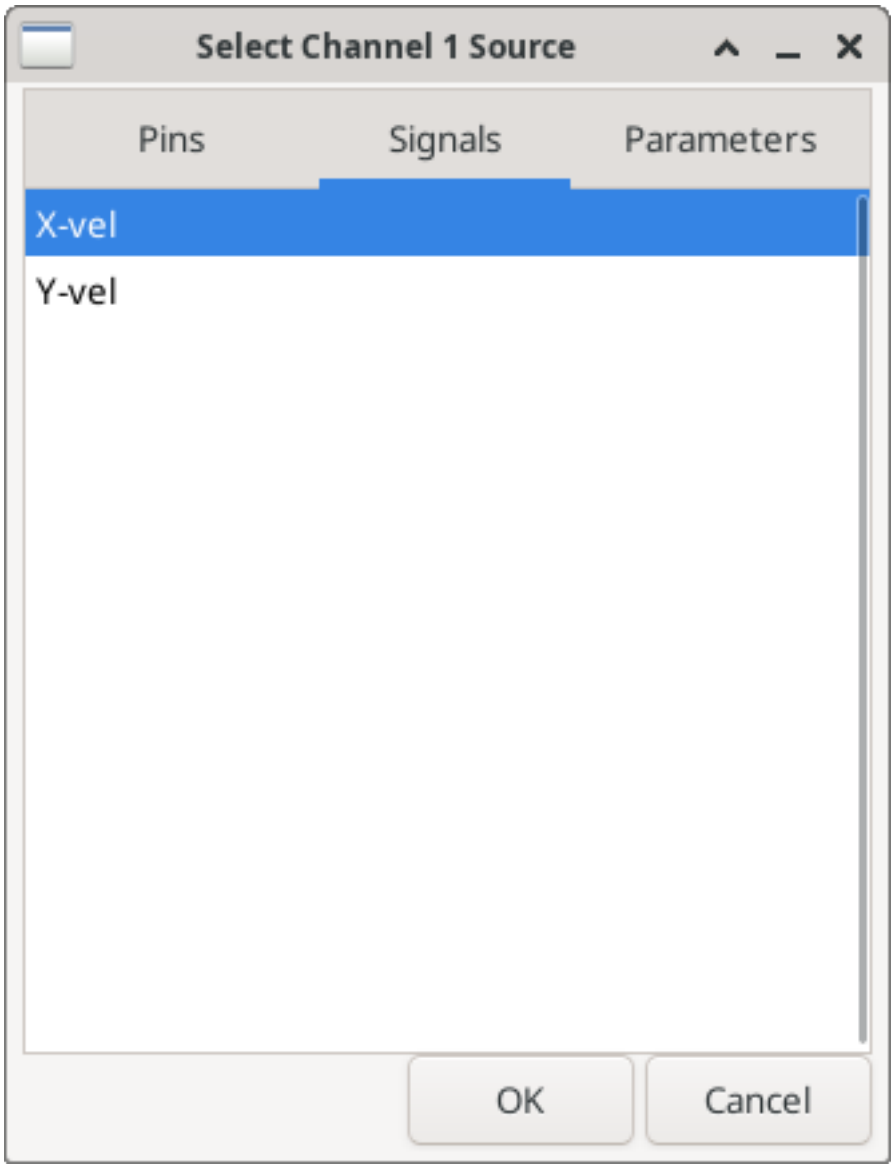



Abbildung 5.10: Signal auswählen

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.

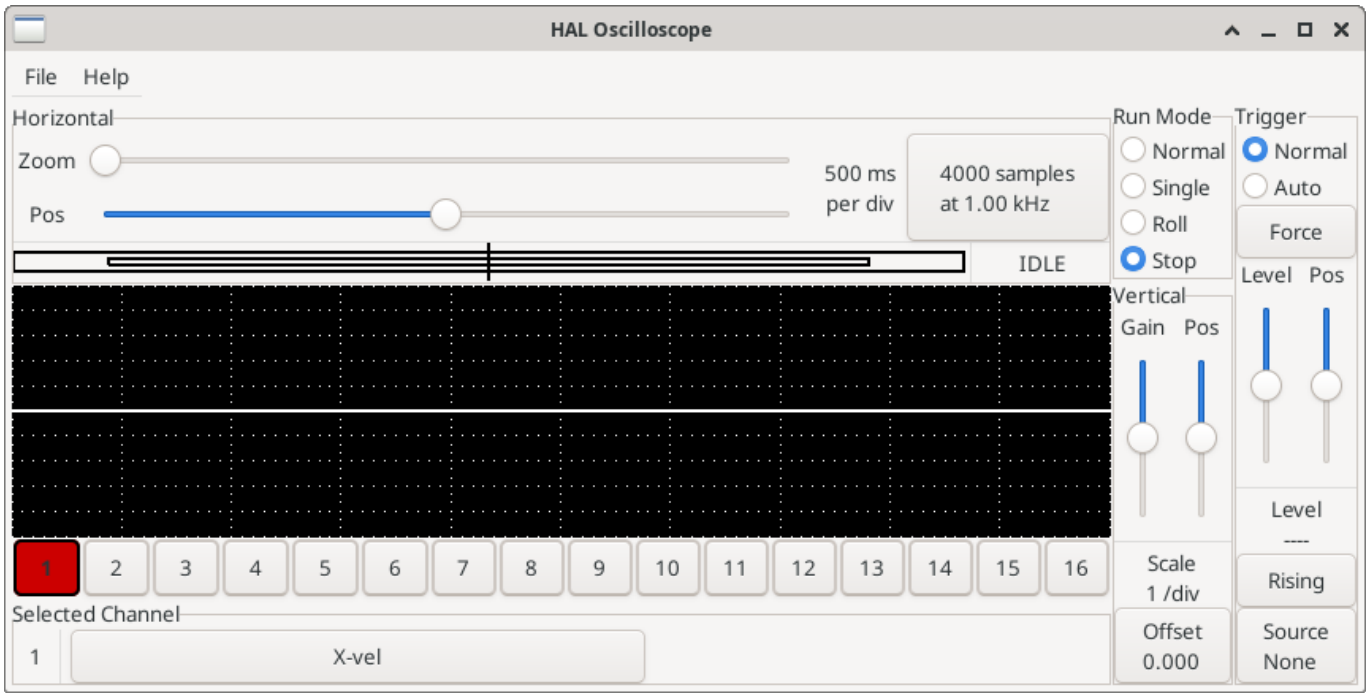



Abbildung 5.11: Halscope

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.

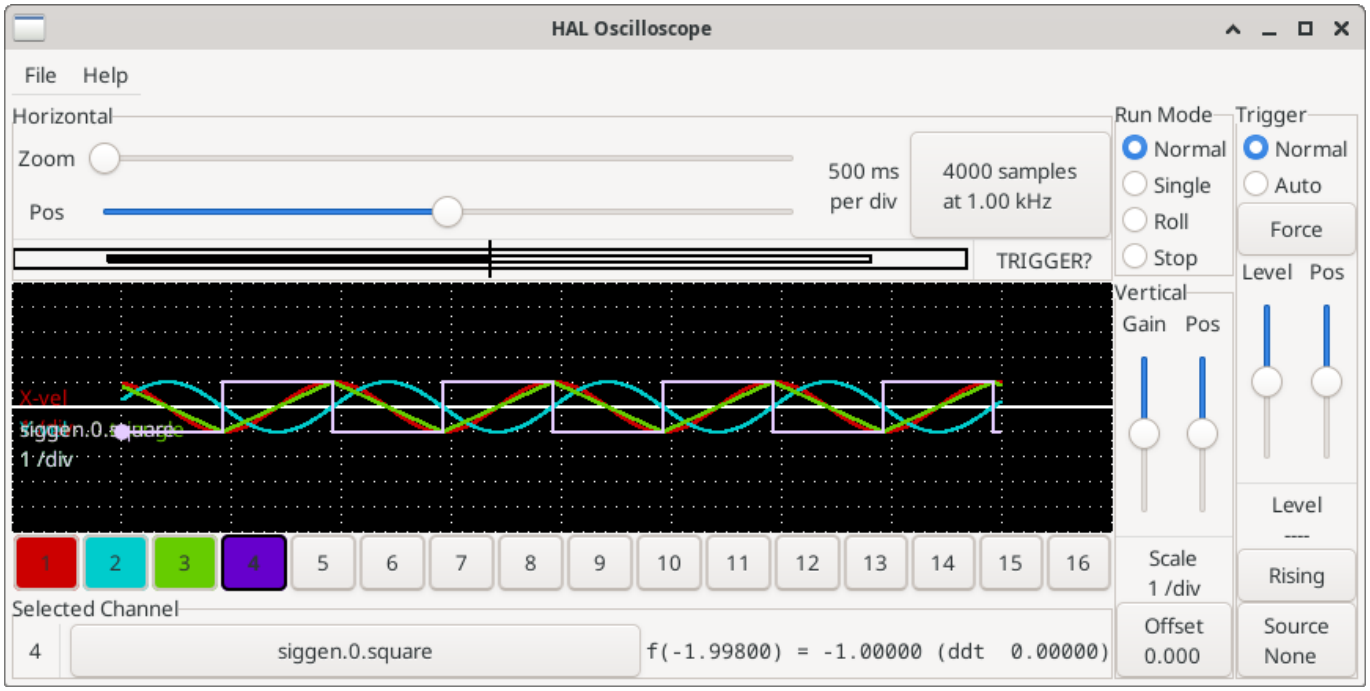



Abbildung 5.12: Erfasste Wellenformen

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.

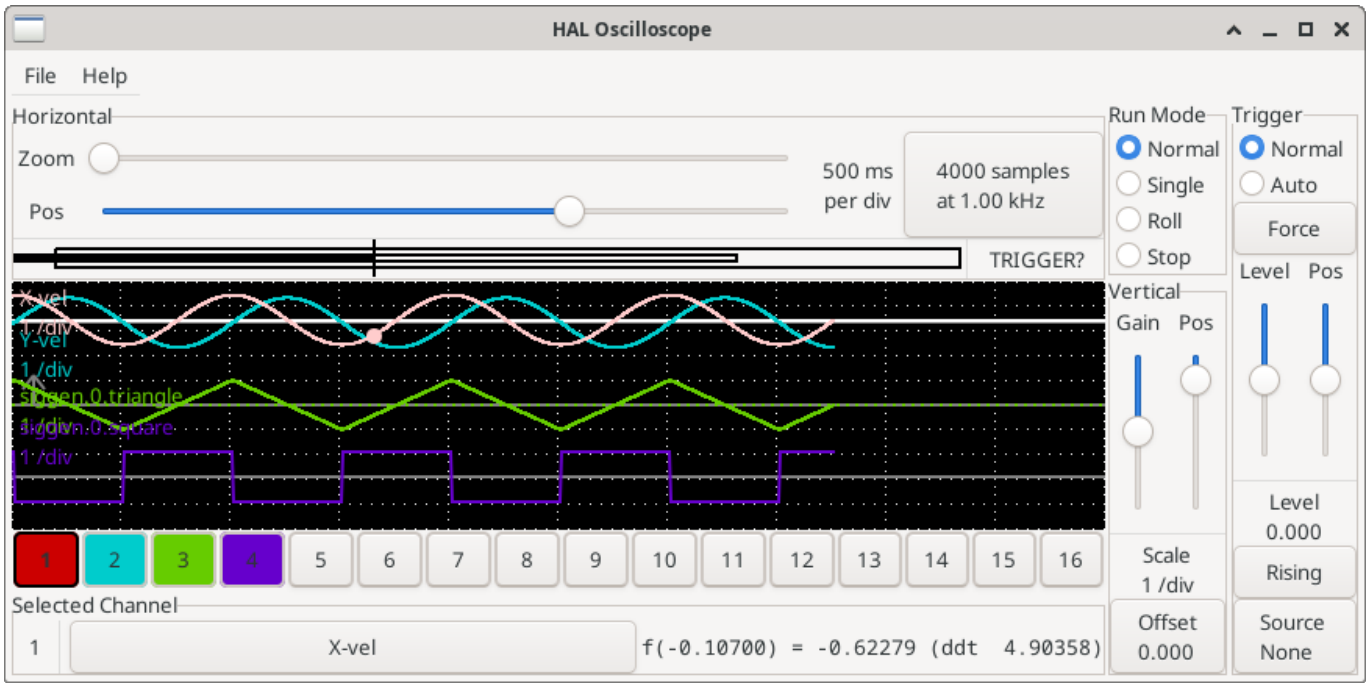



Abbildung 5.13: Vertikale Einstellung

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.

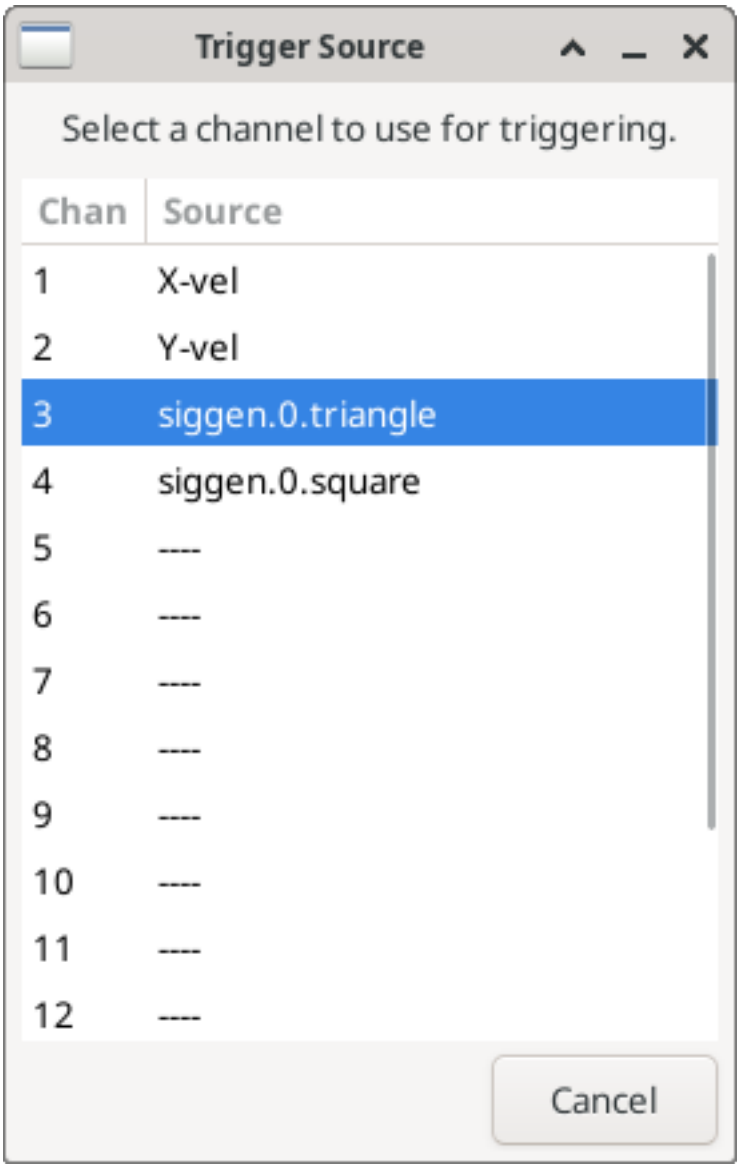



Abbildung 5.14: Dialogfeld *Triggerquelle* (engl. trigger source)

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.

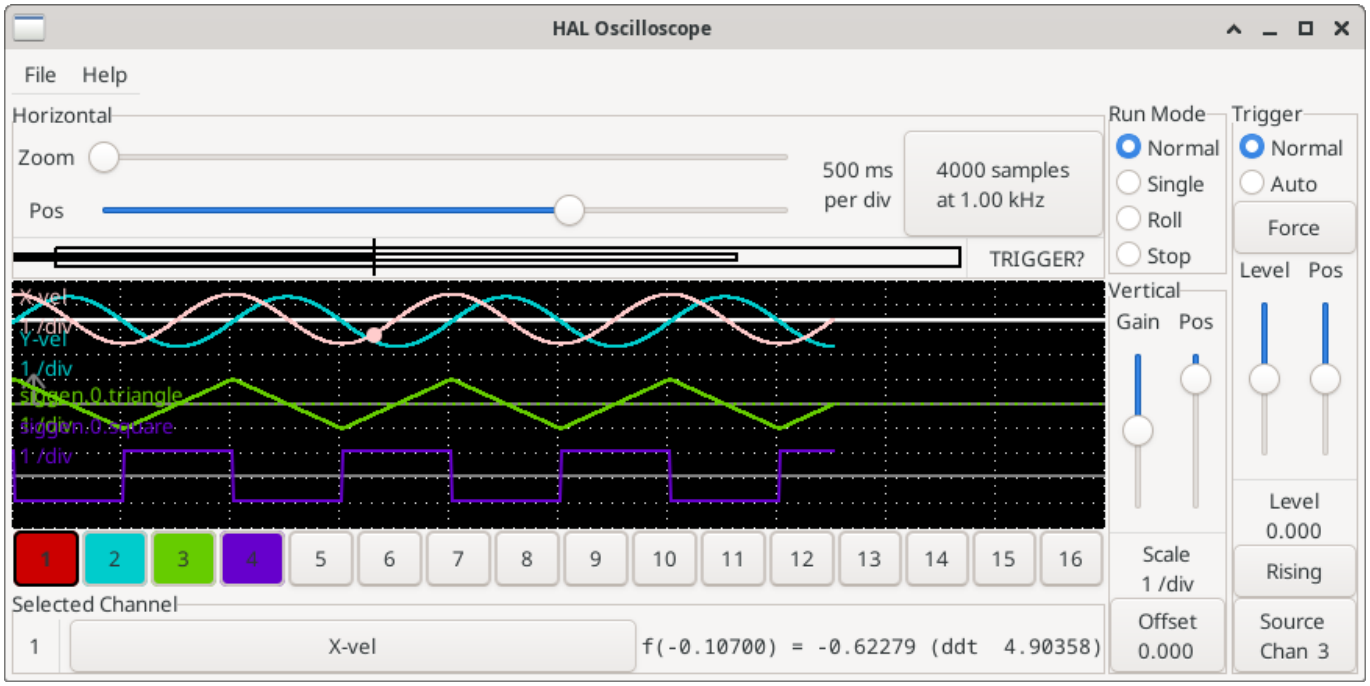



Abbildung 5.15: Wellenformen mit Triggerung

#### 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.

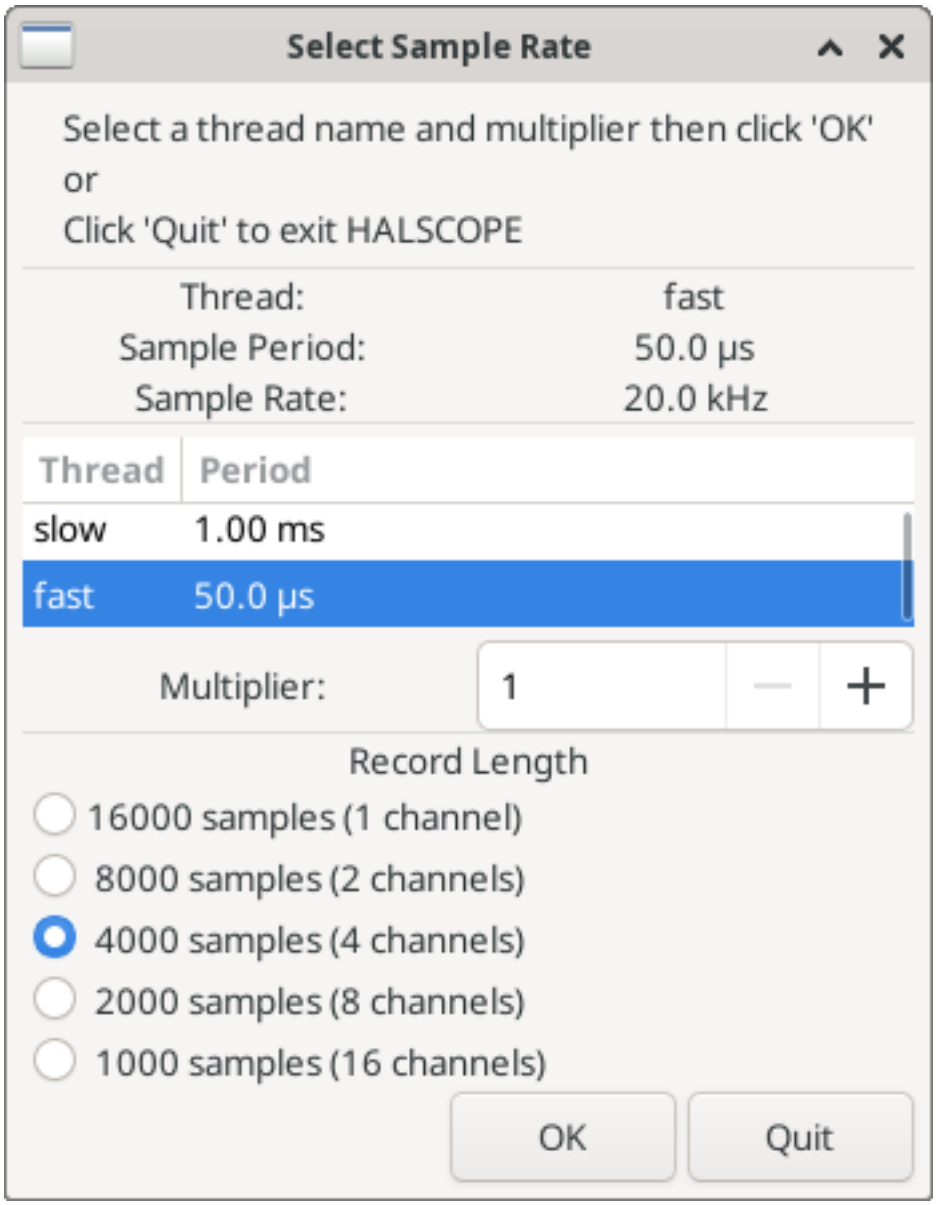



Abbildung 5.16: Dialogfeld für Abtastrate

#### 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. Clicking on a selected channel button (black border) will turn the channel off. So click on the channel 2 button, then click again on this button and the channel will turn off. Then click twice on channel 3 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.

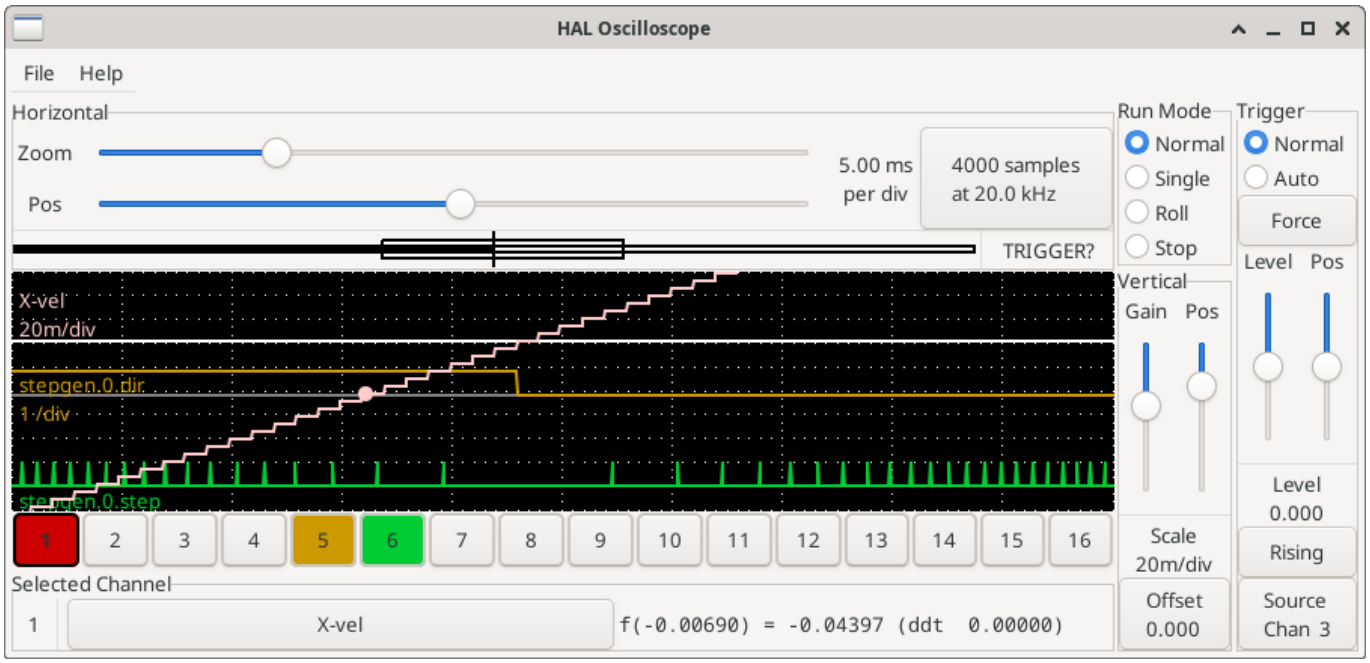



Abbildung 5.17: Schrittimpulse

#### 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 Examples

All of these examples assume you are starting with a StepConf-based configuration and have two threads *base-thread* and *servo-thread*. The StepConf wizard will create an empty custom.hal and a custom_postgui.hal file. The custom.hal file will be loaded after the configuration HAL file and the custom_postgui.hal file is loaded after the GUI has been loaded.

### 5.5.1 Verbinden von zwei Ausgängen

To connect two outputs to an input you can use the or2 component. The or2 works like this, if either input to or2 is on then the or2 output is on. If neither input to or2 is on the or2 output is off.

For example to have two PyVCP buttons both connected to one LED.

**The .xml file to instruct PyVCP to prepare a GUI that features two buttons (named ”button-1” and ”button-2”) and an LED (named ”led-1”).**

```
<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, read after the GUI is set up and ports ready to accept the logic described in HAL.**

```
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
```

When you run this example in an axis simulator created with the StepConf Wizard, you can open a terminal and see the pins created with *loadrt or2* by typing in `halcmd show pin or2` in the terminal.

**Running halcmd on the UNIX command line to show the pins crafted with module or2.**

```
$ 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
```

You can see from the HAL command `show pin or2` that the `button-1` pin is connected to the `or2.0.in0` pin. From the direction arrow you can see that the button is and output and the `or2.0.in0` is an input. The output from or2 goes to the input of the LED.

### 5.5.2 Manueller Werkzeugwechsel

In this example it is assumed that you’re *rolling your own* configuration and wish to add the HAL Manual Toolchange window. The HAL Manual Toolchange is primarily useful if you have presettable tools and you store the offsets in the tool table. If you need to touch off for each tool change then it is best just to split up your G-code. To use the HAL Manual Toolchange window you basically have to

1. load the hal_manualtoolchange component,
2. then send the iocontrol *tool change* to the hal_manualtoolchange *change* and
3. send the hal_manualtoolchange *changed* back to the iocontrol *tool changed*.

A pin is provided for an external input to indicate the tool change is complete.

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

This example uses *ddt*, *mult2* and *abs* to compute the velocity of a single axis. For more information on the real time components see the man pages or the HAL Components List (Abschnitt 5.1.5).

The first thing is to check your configuration to make sure you are not using any of the real time components all ready. You can do this by opening up the HAL Configuration window and look for the components in the pin section. If you are then find the HAL file that they are being loaded in and increase the counts and adjust the instance to the correct value. Add the following to your custom.hal file.

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 this last section we are setting the `mult2.0.in1` to 60 to convert the inch per second to inch per minute (IPM) that we get from the `ddt.0.out`.

The `xpos-cmd` sends the commanded position to the `ddt.0.in`. The ddt computes the derivative of the change of the input.

The `ddt2.0.out` is multiplied by 60 to give IPM.

The `mult2.0.out` is sent to the `abs` to get the absolute value.

The following figure shows the result when the X axis is moving at 15 IPM in the minus direction. Notice that we can get the absolute value from either the `abs.0.out` pin or the `X-IPM` signal.

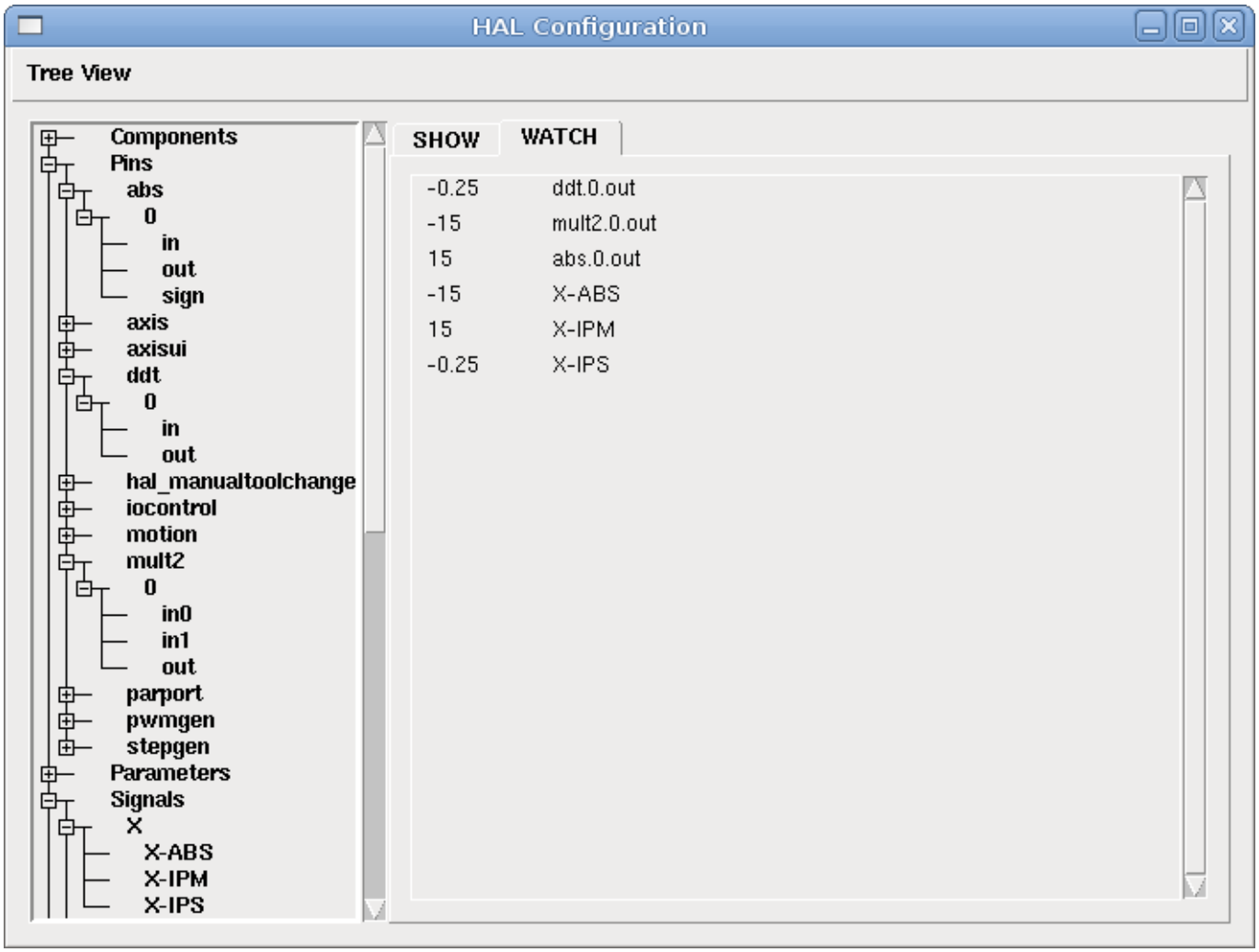



Abbildung 5.18: HAL: Geschwindigkeitsbeispiel

### 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 this example we have a servo motor driving a lathe spindle. If we just used the commanded spindle speeds on the servo it will try to go from present speed to commanded speed as fast as it can. This could cause a problem or damage the drive. To slow the rate of change we can send the `spindle.N.speed-out` through a limiter before the PID, so that the PID command value changes to new settings more slowly.

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.

Place the following in a text file called softstart.hal. If you’re not familiar with Linux place the file in your home directory.

```
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
```

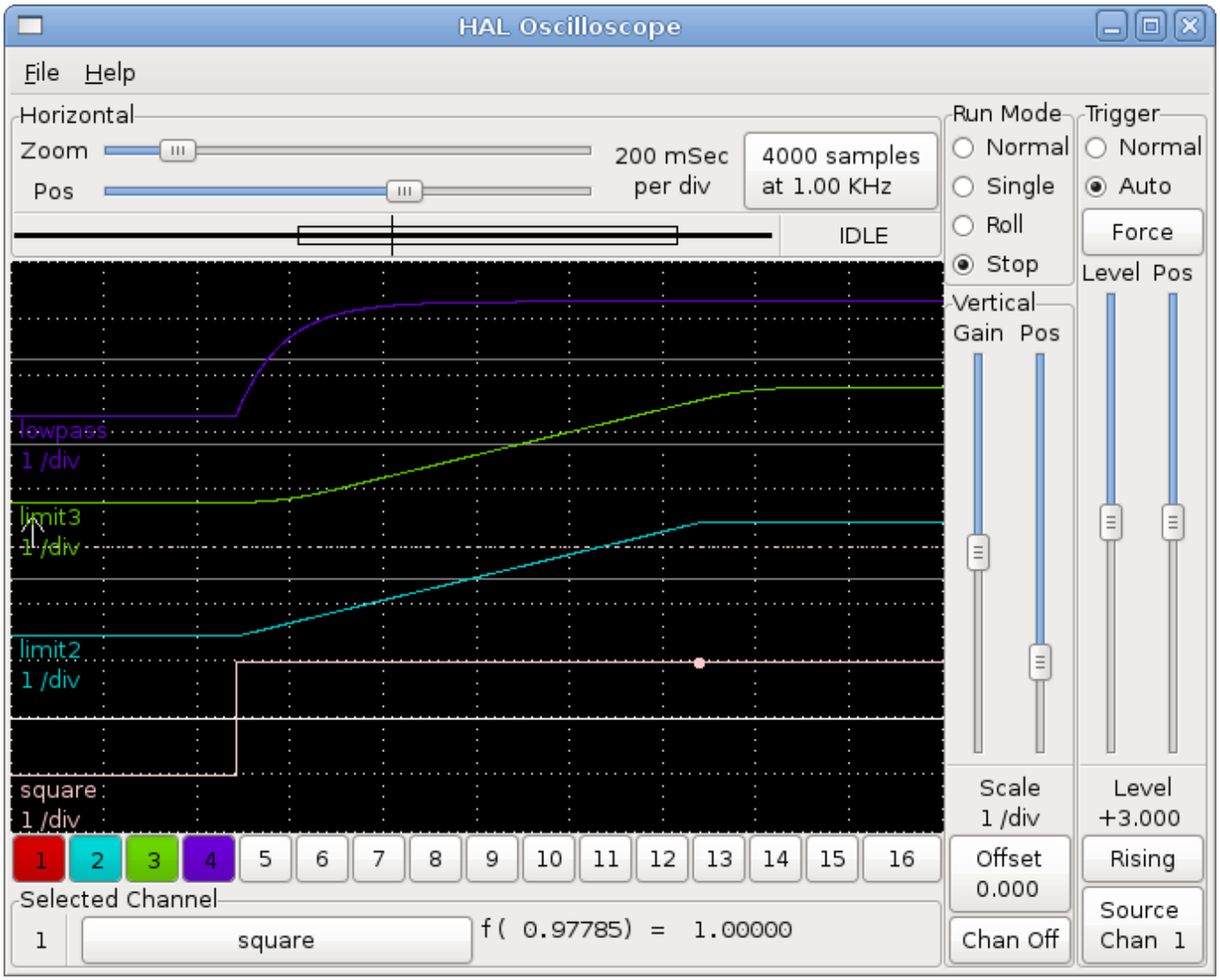
Wenn das HAL Oszilloskop zum ersten Mal startet, klicken Sie auf *OK*, um den Standardfaden zu akzeptieren.

Next you have to add the signals to the channels. Click on channel 1 then select *square* from the Signals tab. Repeat for channels 2-4 and add lowpass, limit2, and limit3.

Next to set up a trigger signal click on the Source None button and select square. The button will change to Source Chan 1.

Next click on Single in the Run Mode radio buttons box. This will start a run and when it finishes you will see your 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.




To see the effect of changing the set point values of any of the components you can change them in the terminal window. To see what different gain settings do for lowpass just type the following in the terminal window and try different settings.

```
setp lowpass.0.gain *.01
```

Nachdem Sie eine Einstellung geändert haben, lassen Sie das Oszilloskop erneut laufen, um die Änderung zu sehen.

When you're finished type *exit* in the terminal window to shut down halrun and close the halscope. Don't close the terminal window with halrun running as it might leave some things in memory that could prevent LinuxCNC from loading.

Weitere Informationen zu Halscope finden Sie im HAL-Handbuch und im Tutorial.

### 5.5.5 Stand-Alone HAL

In some cases you might want to run a GladeVCP screen with just HAL. For example say you had a stepper driven device that all you need is to run a stepper motor. A simple *Start/Stop* interface is all you need for your application so no need to load up and configure a full blown CNC application.

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 Core Components

See also the man pages *motion(9)*.

### 5.6.1 Bewegung (engl. motion)

These pins and parameters are created by the realtime *motmod* module.

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

These pins, parameters, and functions are created by the realtime *motmod* module.

- *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

#### 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* - Receives and processes motion commands
- *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 - accepts non-realtime I/O commands via NML, interacts with HAL .

iocontrol’s HAL pins are turned on and off in non-realtime context. If you have strict timing requirements or simply need more I/O, consider using the realtime synchronized I/O provided by motion instead.

#### 5.6.4.1 Pins

* *iocontrol.0.coolant-flood* (bit, out) TRUE when flood coolant is requested. * *iocontrol.0.coolant-mist* (bit, out) TRUE when mist coolant is requested. * *iocontrol.0.emc-enable-in* (bit, in) Should be driven FALSE when an external E-Stop condition exists. * *iocontrol.0.lube* (bit, out) TRUE when lube is commanded. * *iocontrol.0.lube_level* (bit, in) Should be driven TRUE when lube level is high enough. * *iocontrol.0.tool-change* (bit, out) TRUE when a tool change is requested. * *iocontrol.0.tool-changed* (bit, in) Should be driven TRUE when a tool change is completed. * *iocontrol.0.tool-number* (s32, out) The current tool number. * *iocontrol.0.tool-prep-number* (s32, out) The number of the next tool, from the RS274NGC T-word. * *iocontrol.0.tool-prepare* (bit, out) TRUE when a tool prepare is requested. * *iocontrol.0.tool-prepared* (bit, in) Should be driven TRUE when a tool prepare is completed. * *iocontrol.0.user-enable-out* (bit, out) FALSE when an internal E-Stop condition exists. * *iocontrol.0.user-request-enable* (bit, out) TRUE when the user has requested that E-Stop be cleared.

## 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

Most of the commands in the following list have their own dedicated man pages. Some will have expanded descriptions, some will have limited descriptions. From this list you know what components exist, and you can use `man` *name* on your UNIX command line to get additional information. To view the information in the man page, in a terminal window type:

```
man axis
```

The one or other setup of a UNIX system may require to explicitly specify the section of the man page. If you do not find the man page or the name of the man page is already taken by another UNIX tool with the LinuxCNC man page residing in another section, then try to explicitly specify the section, as in `man _sectionno_ axis`, with *sectionno* = 1 for non-realtime and 9 for realtime components.

---

**Anmerkung**
See also the *Man Pages* section of the docs main page or the directory listing. To search in the man pages, use the UNIX tool `apropos`.

---

### 5.7.1.1 User Interfaces (non-realtime)

| | |
|---|---|
| **axis** | AXIS LinuxCNC (ehemals ”Enhanced Machine Controller”) GUI |
| **axis-remote** | AXIS Fernzugriff (engl. Remote Interface) |
| **gmoccapy** | Touchy LinuxCNC Grafische Benutzeroberfläche |
| **gscreen** | Touchy LinuxCNC Grafische Benutzeroberfläche |
| **halui** | Beobachten Sie die HAL-Pins und befehlen Sie LinuxCNC über NML |
| **mdro** | nur manuell Digital Read Out (DRO) |
| **ngcgui** | Framework zur dialogorientierten G-Code-Generierung auf dem Controller |
| **panelui** | 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 Motion (non-realtime)

| | |
|---|---|
| **io** | iocontrol - interacts with HAL or G-code in non-realtime |
| **iocontrol** | Interacts with HAL or G-code in non-realtime |
| **iov2** | Interacts with HAL or G-code in non-realtime |
| **mdi** | Senden von G-Code-Befehlen vom Terminal an die laufende LinuxCNC-Instanz |
| **milltask** | Non-realtime task controller for LinuxCNC |

#### 5.7.1.3 Hardware-Treiber

| | |
|---|---|
| **elbpcom** | Kommunikation mit Mesa-Ethernet-Karten |
| **gs2_vfd** | HAL non-realtime component for Automation Direct GS2 VFDs |
| **hy_gt_vfd** | HAL non-realtime component for Huanyang GT-series VFDs |
| **hy_vfd** | HAL non-realtime component for Huanyang VFDs |
| **mb2hal** | MB2HAL is a generic non-realtime HAL component to communicate with one or more Modbus devices. Modbus RTU and Modbus TCP are supported. |
| **mitsub_vfd** | HAL non-realtime component for Mitsubishi A500 F500 E500 A500 D700 E700 F700-series VFDs (others may work) |
| **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 non-realtime component for SVD-P(S) VFDs |
| **vfdb_vfd** | HAL non-realtime component for Delta VFD-B Variable Frequency Drives |
| **vfs11_vfd** | HAL non-realtime component for Toshiba-Schneider VF-S11 Variable Frequency Drives |
| **wj200_vfd** | Hitachi WJ200 Modbus-Treiber |
| **xhc-hb04** | Non-realtime HAL component for the xhc-hb04 pendant |
| **xhc-hb04-accels** | Obsolete script for jogging wheel |
| **xhc-whb04b-6** | Non-realtime 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 |
| **hal_parport** | Realtime HAL component to communicate with one or more PC parallel ports |
| **hm2_7i43** | Mesa Electronics driver for the 7I43 EPP Anything IO board with HostMot2. (See the man page for more information) |

| | |
|---|---|
| **hm2_7i90** | LinuxCNC HAL driver for the Mesa Electronics 7I90 EPP Anything IO board with HostMot2 firmware |
| **hm2_eth** | LinuxCNC HAL-Treiber für die Mesa Electronics Ethernet Anything IO-Karten, mit HostMot2-Firmware |
| **hm2_pci** | Mesa Electronics driver for the 5I20, 5I22, 5I23, 4I65, and 4I68 Anything I/O boards, with HostMot2 firmware. (See the man page for more information) |
| **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 driver for the 7I65 eight-axis servo card. (See the man page for more information) |
| **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 Utilities (non-realtime)

| | |
|---|---|
| **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** | Filter noisy digital inputs Details Description |
| **demux** | Auswahl eines von mehreren Ausgangsstiften durch Ganzzahl und/oder oder einzelne Bits |
| **edge** | Kanten-Detektor |
| **estop_latch** | Notaus-Verriegelung |
| **flipflop** | D-Typ Flip-Flop |
| **logic** | Allgemeine logische Funktionskomponente |
| **lut5** | Logikfunktion mit 5 Eingängen, die auf einer Look-up-Tabelle basiert, siehe Beschreibung |
| **match8** | 8-Bit-Binär-Match-Detektor |
| **multiclick** | Einzel-, Doppel-, Dreifach- und Vierfach-Klick-Detektor |
| **multiswitch** | Schaltet zwischen einer bestimmten Anzahl von Ausgangsbits um |
| **not** | Inverter |
| **oneshot** | One-Shot-Pulsgenerator |
| **or2** | ODER-Gatter mit zwei Eingängen |
| **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** | Counts input pulses (deprecated). Use the encoder component. |
| **ddt** | Berechnet die Ableitung der Eingangsfunktion. |
| **deadzone** | Gibt den Mittelpunkt zurück, wenn er sich innerhalb des Schwellenwerts befindet. |
| **div2** | Quotient of two floating point inputs. |
| **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** | Limits the output signal to fall between min and max. Limit its slew rate to less than maxv per second. [3] |
| **limit3** | Limit the output signal to fall between min and max. Limit its slew rate to less than maxv per second. Limit its second derivative to less than MaxA per second squared [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 |
| **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 |

[2] Wenn der Eingang eine Position ist, bedeutet dies, dass die *Position* begrenzt ist.
[3] When the input is a position, this means that *position* and *velocity* are limited.
[4] When the input is a position, this means that the *position*, *velocity*, and *acceleration* are limited.

| | |
|---|---|
| **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 Motion control (Realtime)

| | |
|---|---|
| **motion** | Akzeptiert NML-Bewegungsbefehle, interagiert mit HAL in Echtzeit |

#### 5.7.1.9 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. |

#### 5.7.1.10 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** | Creates a square-wave for the *charge pump* input of some controller boards. The *Charge Pump* should be added to the base thread function. When enabled, the output is on for one period and off for one period. To calculate the frequency [Hz] of the output: 1/(period time in seconds x 2). For example, if you have a base period of 100,000 ns that is 0.0001 seconds and the formula would be 1/(0.0001 x 2) = 5,000 Hz or 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** | Used by StepConf to allow testing of acceleration and velocity values for an axis. |
| **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

This chapter provides details on core functionalities of LinuxCNC that demand exact timing for

- the generation of signals that is interpreted by hardware (like motors) or
- for the interpretation of signals sent by the hardware (like encoders).

### 5.8.1 StepGen

This component provides software based generation of step pulses in response to position or velocity commands. In position mode, it has a built in pre-tuned position loop, so PID tuning is not required. In velocity mode, it drives a motor at the commanded speed, while obeying velocity and acceleration limits. It is a realtime component only, and depending on CPU speed, etc., is capable of maximum step rates of 10 kHz to perhaps 50 kHz. The step pulse generator block diagram shows three block diagrams, each is a single step pulse generator. The first diagram is for step type *0*, (step and direction). The second is for step type *1* (up/down, or pseudo-PWM), and the third is for step types 2 through 14 (various stepping patterns). The first two diagrams show position mode control, and the third one shows velocity mode. Control mode and step type are set independently, and any combination can be selected.

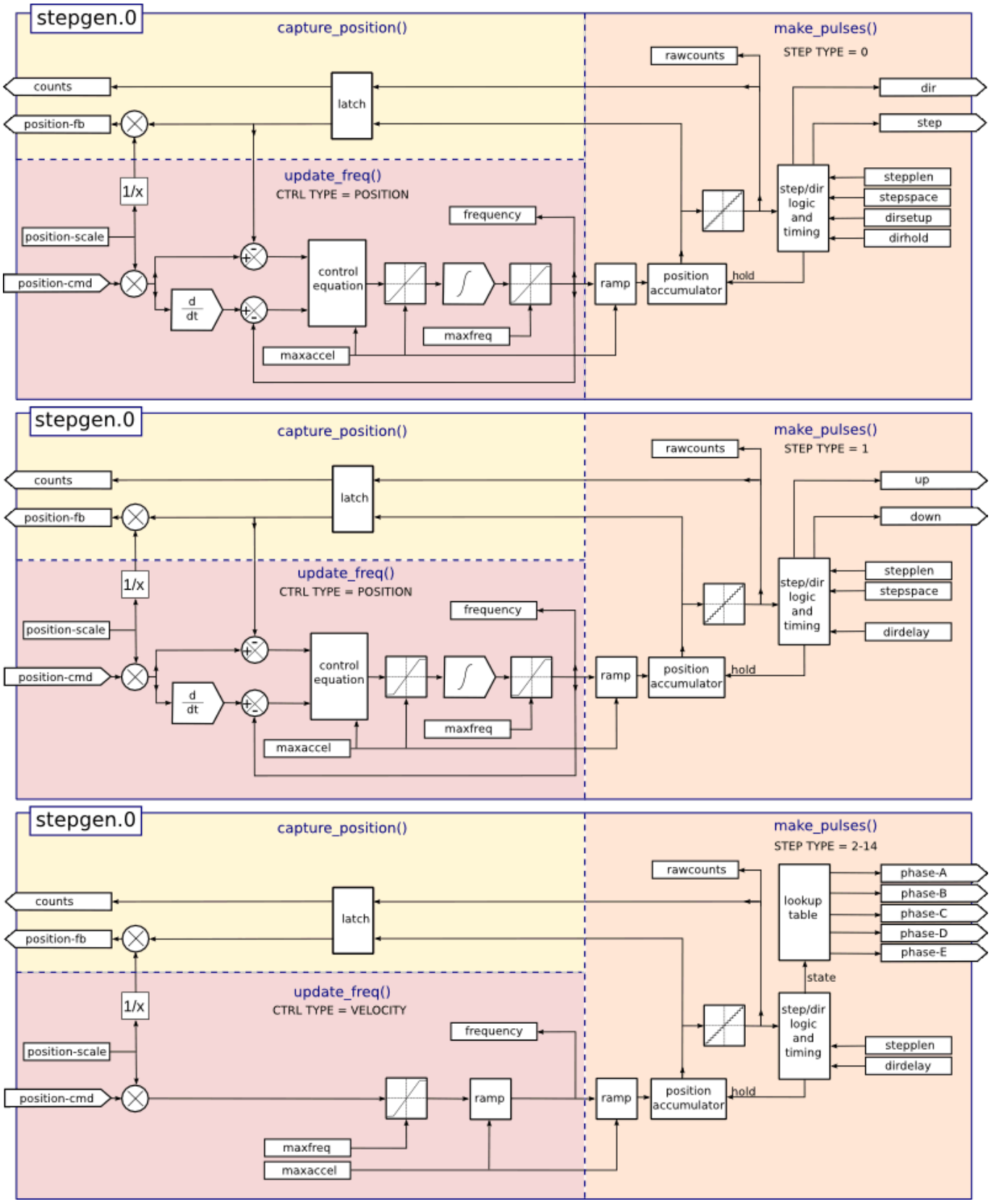



Abbildung 5.19: Step Pulse Generator Block Diagram position mode

**Laden der Komponente stepgen**

```
halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

**<type-array>**
ist eine Reihe von durch Kommata getrennten Dezimalzahlen. Jede Zahl bewirkt, dass ein Einzelschritt-Impulsgenerator geladen wird; der Wert der Zahl bestimmt die Schrittart.

**<ctrl_array>**
ist eine durch Komma getrennte Folge von *p*- oder *v*-Zeichen, um den Positions- oder Geschwindigkeitsmodus anzugeben.

**ctrl_type**
ist optional, wenn sie weggelassen wird, werden alle Schrittgeneratoren im Positionsmodus arbeiten.

Zum Beispiel:

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

Will install three step generators. The first two use step type *0* (step and direction) and run in position mode. The last one uses step type *2* (quadrature) and runs in velocity mode. The default value for *<config-array>* is *0,0,0* which will install three type *0* (step/dir) generators. The maximum number of step generators is 8 (as defined by MAX_CHAN in stepgen.c). Each generator is independent, but all are updated by the same function(s) at the same time. In the following descriptions, *<chan>* is the number of a specific generator. The first generator is number 0.

**Komponente ”stepgen” entfernen (engl. unload)**

```
halcmd: unloadrt stepgen
```

#### 5.8.1.1 Pins

On the step type and control type selected.

- (float) `stepgen.__<chan>__.position-cmd` - Desired motor position, in position units (position mode only).
- (float) `stepgen.__<chan>__.velocity-cmd` - Desired motor velocity, in position units per second (velocity mode only).
- s32) `stepgen.__<chan>__.counts` - Feedback position in counts, updated by *capture_position()*.
- (float) `stepgen.__<chan>__.position-fb` - Feedback position in position units, updated by *capture_position()*.
- (bit) `stepgen.__<chan>__.enable` - Enables output steps - when false, no steps are generated.
- (bit) `stepgen.__<chan>__.step` - Step pulse output (step type 0 only).
- (bit) `stepgen.__<chan>__.dir` - Direction output (step type 0 only).
- (bit) `stepgen.__<chan>__.up` - UP pseudo-PWM output (step type 1 only).
- (bit) `stepgen.__<chan>__.down` - DOWN pseudo-PWM output (step type 1 only).
- (bit) `stepgen.__<chan>__.phase-A` - Phase A output (step types 2-14 only).
- (bit) `stepgen.__<chan>__.phase-B` - Phase B output (step types 2-14 only).
- (bit) `stepgen.__<chan>__.phase-C` - Phase C output (step types 3-14 only).
- (bit) `stepgen.__<chan>__.phase-D` - Phase D output (step types 5-14 only).
- (bit) `stepgen.__<chan>__.phase-E` - Phase E output (step types 11-14 only).

#### 5.8.1.2 Parameter

* (float) `stepgen.__<chan>__.position-scale` - Steps per position unit. This parameter is used for both output and feedback. * (float) `stepgen.__<chan>__.maxvel` - Maximum velocity, in position units per second. If 0.0, has no effect. * (float) `stepgen.__<chan>__.maxaccel` - Maximum accel/decel rate, in positions units per second squared. If 0.0, has no effect. * (float) `stepgen.__<chan>__.frequency` - The current step rate, in steps per second. * (float) `stepgen.__<chan>__.steplen` - Length of a step pulse (step type 0 and 1) or minimum time in a given state (step types 2-14), in nano-seconds. * (float) `stepgen.__<chan>__.stepspace` - Minimum spacing between two step pulses (step types 0 and 1 only), in nano-seconds. Set to 0 to enable the stepgen *doublefreq* function. To use *doublefreq* the parport reset function must be enabled. * (float) `stepgen.__<chan>__.dirsetup` - Minimum time from a direction change to the beginning of the next step pulse (step type 0 only), in nano-seconds. * (float) `stepgen.__<chan>__.dirhold` - Minimum time from the end of a step pulse to a direction change (step type 0 only), in nanoseconds. * (float) `stepgen.__<chan>__.dirdelay` - Minimum time any step to a step in the opposite direction (step types 1-14 only), in nano-seconds. * (s32) `stepgen.__<chan>__.rawcounts` - The raw feedback count, updated by *make_pulses()*.

In position mode, the values of maxvel and maxaccel are used by the internal position loop to avoid generating step pulse trains that the motor cannot follow. When set to values that are appropriate for the motor, even a large instantaneous change in commanded position will result in a smooth trapezoidal move to the new location. The algorithm works by measuring both position error and velocity error, and calculating an acceleration that attempts to reduce both to zero at the same time. For more details, including the contents of the *control equation* box, consult the code.

In velocity mode, maxvel is a simple limit that is applied to the commanded velocity, and maxaccel is used to ramp the actual frequency if the commanded velocity changes abruptly. As in position mode, proper values for these parameters ensure that the motor can follow the generated pulse train.

#### 5.8.1.3 Step Types

Step generator supports 15 different *step sequences*:

**Schritttyp 0 (engl. step type 0)** Step type 0 is the standard step and direction type. When configured for step type 0, there are four extra parameters that determine the exact timing of the step and direction signals. In the following figure the meaning of these parameters is shown. The parameters are in nanoseconds, but will be rounded up to an integer multiple of the thread period for the threaed that calls *make_pulses()*. For example, if *make_pulses()* is called every 16 µs, and steplen is 20000, then the step pulses will be 2 x 16 = 32 µs long. The default value for all four of the parameters is 1 ns, but the automatic rounding takes effect the first time the code runs. Since one step requires *steplen* ns high and *stepspace* ns low, the maximum frequency is 1,000,000,000 divided by (*steplen* + *stepspace*)'. If *maxfreq* is set higher than that limit, it will be lowered automatically. If *maxfreq* is zero, it will remain zero, but the output frequency will still be limited.

When using the parallel port driver the step frequency can be doubled using the parport reset function together with StepGen's *doublefreq* setting.

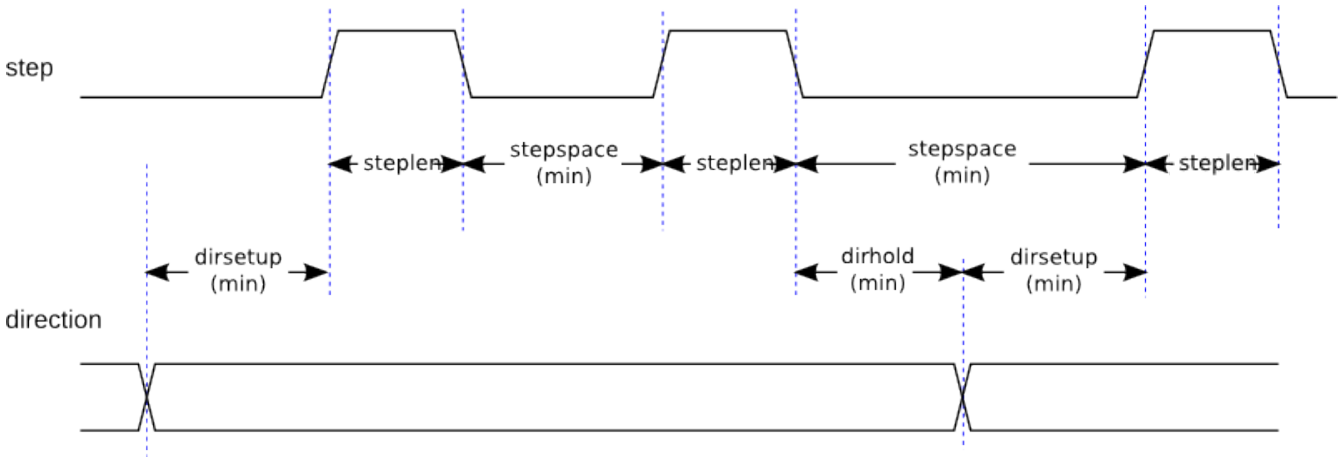



Abbildung 5.20: Schritt- und Richtungs-Timing (engl. step and direction timing)

**Schritt Typ 1 (step type 1)** Step type 1 has two outputs, up and down. Pulses appear on one or the other, depending on the direction of travel. Each pulse is *steplen* ns long, and the pulses are separated by at least *stepspace* ns. The maximum frequency is the same as for step type 0. If *maxfreq* is set higher than the limit it will be lowered. If *maxfreq* is zero, it will remain zero but the output frequency will still be limited.

**Warnung**
Do not use the parport reset function with step types 2 - 14. Unexpected results can happen.

**Schritt Typen 2 - 14 (engl. step type 2-14)** Step types 2 through 14 are state based, and have from two to five outputs. On each step, a state counter is incremented or decremented. The Two-and-Three-Phase, Four-Phase, and Five-Phase show the output patterns as a function of the state counter. The maximum frequency is 1,000,000,000 divided by *steplen*, and as in the other modes, *maxfreq* will be lowered if it is above the limit.

.Two-and-Three-Phase Step Types Schritttypen: Zwei-und-Drei-Phase

.Four-Phase Step Types Schritttypen: Vierphasig

.Five-Phase Step Types Schritttypen: Fünfphasig

#### 5.8.1.4 Funktionen

The component exports three functions. Each function acts on all of the step pulse generators - running different generators in different threads is not supported.

- (funct) `stepgen.make-pulses` - High speed function to generate and count pulses (no floating point).
- (funct) `stepgen.update-freq` - Low speed function does position to velocity conversion, scaling and limiting.
- (funct) `stepgen.capture-position` - Low speed function for feedback, updates latches and scales position.

The high speed function *stepgen.make-pulses* should be run in a very fast thread, from 10 to 50 µs depending on the capabilities of the computer. That thread's period determines the maximum step frequency, since *steplen*, *stepspace*, *dirsetup*, *dirhold*, and *dirdelay* are all rounded up to a integer multiple of the thread periond in nanoseconds. The other two functions can be called at a much lower rate.

## 5.8.2 PWMgen

This component provides software based generation of PWM (Pulse Width Modulation) and PDM (Pulse Density Modulation) waveforms. It is a realtime component only, and depending on CPU speed, etc., is capable of PWM frequencies from a few hundred Hertz at pretty good resolution, to perhaps 10 kHz with limited resolution.

**Laden von PWMgen**

```
loadrt pwmgen output_type=<config-array>
```

The <*config-array*> is a series of comma separated decimal integers. Each number causes a single PWM generator to be loaded, the value of the number determines the output type. The following example will install three PWM generators. There is no default value, if <*config-array*> is not specified, no PWM generators will be installed. The maximum number of frequency generators is 8 (as defined by MAX_CHAN in pwmgen.c). Each generator is independent, but all are updated by the same function(s) at the same time. In the following descriptions, <chan> is the number of a specific generator. The first generator is number 0.

**Beispiel für das Laden von PWMgen**

```
loadrt pwmgen output_type=0,1,2
```

Will install three PWM generators. The first will use an output of type *0* (PWM only), the next one will use a type 1 output (PWM and direction) and the third will use a type 2 output (UP and DOWN). There is no default value, if <*config-array*> is not not specified, no PWM generator will be installed. The maximum number of frequency generators is 8 (as defined by MAX_CHAN in pwmgen.c). Each generator is independent, but all are updated by the same function(s), at the same time. In the descriptions that follow, <*chan*> is the number of specific generators. The numbering of PWM generators starts at 0.

**Entfernen (engl. hier unloading) von PWMgen**

```
unloadrt pwmgen
```

### 5.8.2.1 Ausgangstypen (engl. output types)

Der PWM-Generator unterstützt drei verschiedene ”Ausgangstypen”.

- *Output type 0* - PWM output pin only. Only positive commands are accepted, negative values are treated as zero (and will be affected by the parameter *min-dc* if it is non-zero).
- *Output type 1* - PWM/PDM and direction pins. Positive and negative inputs will be output as positive and negative PWM. The direction pin is false for positive commands, and true for negative commands. If your control needs positive PWM for both CW and CCW use the abs component to convert your PWM signal to positive value, when a negative input is input.
- *Output type 2* - UP and DOWN pins. For positive commands, the PWM signal appears on the up output, and the down output remains false. For negative commands, the PWM signal appears on the down output, and the up output remains false. Output type 2 is suitable for driving most H-bridges.

### 5.8.2.2 Pins

Jeder PWM-Generator hat die folgenden Pins:

- (float) `pwmgen.__<chan>__.value` - Command value, in arbitrary units. Will be scaled by the *scale* parameter (see below).

- (bit) `pwmgen.__<chan>__.enable` - Enables or disables the PWM generator outputs.

Jeder PWM-Generator verfügt über einige dieser Pins, je nach gewähltem Ausgangstyp:

- (bit) `pwmgen.__<chan>__.pwm` - PWM (or PDM) output, (output types 0 and 1 only).
- (bit) `pwmgen.__<chan>__.dir` - Direction output (output type 1 only).
- (bit) `pwmgen.__<chan>__.up` - PWM/PDM output for positive input value (output type 2 only).
- (bit) `pwmgen.__<chan>__.down` - PWM/PDM output for negative input value (output type 2 only).

#### 5.8.2.3 Parameter

- (float) `pwmgen.__<chan>__.scale` - Scaling factor to convert `value` from arbitrary units to duty cycle. For example if scale is set to 4000 and the input value passed to the `pwmgen.__<chan>__.value` is 4000 then it will be 100% duty-cycle (always on). If the value is 2000 then it will be a 50% 25 Hz square wave.
- (float) `pwmgen.__<chan>__.pwm-freq` - Desired PWM frequency, in Hz. If 0.0, generates PDM instead of PWM. If set higher than internal limits, next call of *update_freq()* will set it to the internal limit. If non-zero, and *dither* is false, next call of *update_freq()* will set it to the nearest integer multiple of the *make_pulses()* function period.
- (bit) `pwmgen.__<chan>__.dither-pwm` - If true, enables dithering to achieve average PWM frequencies or duty cycles that are unobtainable with pure PWM. If false, both the PWM frequency and the duty cycle will be rounded to values that can be achieved exactly.
- (float) `pwmgen.__<chan>__.min-dc` - Minimum duty cycle, between 0.0 and 1.0 (duty cycle will go to zero when disabled, regardless of this setting).
- (float) `pwmgen.__<chan>__.max-dc` - Maximum duty cycle, between 0.0 and 1.0.
- (float) `pwmgen.__<chan>__.curr-dc` - Current duty cycle - after all limiting and rounding (read only).

#### 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` - High speed function to generate PWM waveforms (no floating point). The high speed function `pwmgen.make-pulses` should be run in the base (fastest) thread, from 10 to 50 µs depending on the capabilities of the computer. That thread's period determines the maximum PWM carrier frequency, as well as the resolution of the PWM or PDM signals. If the base thread is 50,000 ns then every 50 µs the module decides if it is time to change the state of the output. At 50% duty cycle and 25 Hz PWM frequency this means that the output changes state every (1/25) s / 50 µs * 50% = 400 iterations. This also means that you have a 800 possible duty cycle values (without dithering).
- (funct) `pwmgen.update` - Low speed function to scale and limit value and handle other parameters. This is the function of the module that does the more complicated mathematics to work out how many base-periods the output should be high for, and how many it should be low for.

### 5.8.3 Encoder

This component provides software based counting of signals from quadrature (or single-pulse) encoders. It is a realtime component only, and depending on CPU speed, latency, etc., is capable of maximum count rates of 10 kHz to perhaps up to 50 kHz.

The base thread should be 1/2 count speed to allow for noise and timing variation. For example if you have a 100 pulse per revolution encoder on the spindle and your maximum RPM is 3000 the maximum base thread should be 25 µs. A 100 pulse per revolution encoder will have 400 counts. The spindle speed of 3000 RPM = 50 RPS (revolutions per second). 400 * 50 = 20,000 counts per second or 50 µs between counts.

Das Blockdiagramm des Encoderzählers ist ein Blockdiagramm eines Kanals eines Encoderzählers.

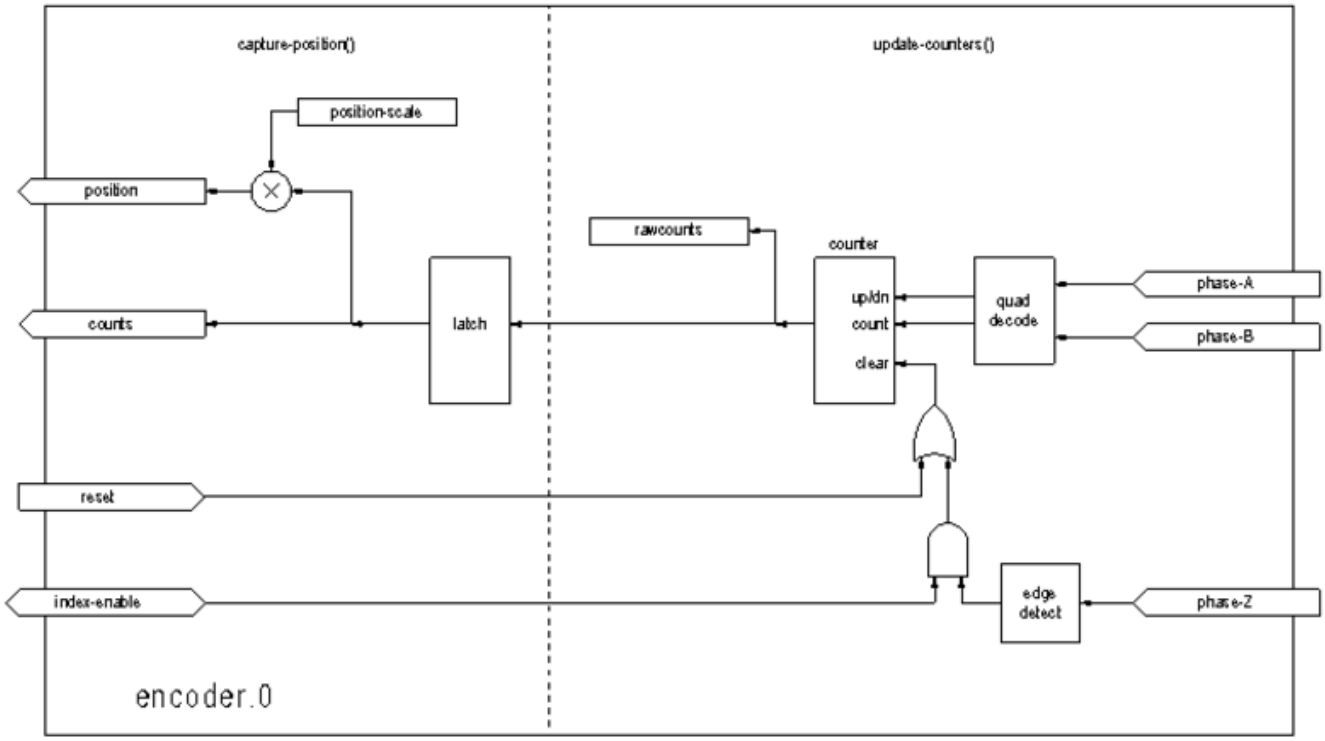



Abbildung 5.21: Encoder Counter Block Diagram

#### Laden des Encoders

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<*counters*> is the number of encoder counters that you want to install. If *num_chan* is not specified, three counters will be installed. The maximum number of counters is 8 (as defined by MAX_CHAN in encoder.c). Each counter is independent, but all are updated by the same function(s) at the same

time. In the following descriptions, <*chan*> is the number of a specific counter. The first counter is number 0.

**Encoder entfernen (engl. unload)**

```
halcmd: unloadrt encoder
```

#### 5.8.3.1 Pins

- `encoder._<chan>_.counter-mode` (bit, I/O) (default: FALSE) - Enables counter mode. When true, the counter counts each rising edge of the phase-A input, ignoring the value on phase-B. This is useful for counting the output of a single channel (non-quadrature) sensor. When false, it counts in quadrature mode.
- `encoder._<chan>_.missing-teeth` (s32, In) (default: 0) - Enables the use of missing-tooth index. This allows a single IO pin to provide both position and index information. If the encoder wheel has 58 teeth with two missing, spaced as if there were 60(common for automotive crank sensors) then the position-scale should be set to 60 and missing-teeth to 2. To use this mode counter-mode should be set true. This mode will work for lathe threading but not for rigid tapping.
- `encoder._<chan>_.counts` (s32, Out) - Position in encoder counts.
- `encoder._<chan>_.counts-latched` (s32, Out) - Not used at this time.
- `encoder._<chan>_.index-enable` (bit, I/O) - When True, `counts` and `position` are reset to zero on next rising edge of Phase Z.
  At the same time, `index-enable` is reset to zero to indicate that the rising edge has occurred. The `index-enable` pin is bi-directional. If `index-enable` is False, the Phase Z channel of the encoder will be ignored, and the counter will count normally. The encoder driver will never set `index-enable` True. However, some other component may do so.
- `encoder._<chan>_.latch-falling` (bit, In) (default: TRUE) - Not used at this time.
- `encoder._<chan>_.latch-input` (bit, In) (default: TRUE) - Not used at this time.
- `encoder._<chan>_.latch-rising` (bit, In) - Not used at this time.
- `encoder._<chan>_.min-speed-estimate` (float, in) - Determine the minimum true velocity magnitude, at which velocity will be estimated as nonzero and position-interpolated will be interpolated. The units of `min-speed-estimate` are the same as the units of `velocity`. Scale factor, in counts per length unit. Setting this parameter too low will cause it to take a long time for velocity to go to 0 after encoder pulses have stopped arriving.
- `encoder._<chan>_.phase-A` (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 (index pulse) of the quadrature encoder signal.
- `encoder._<chan>_.position` (float, Out) - Position in scaled units (see `position-scale`).
- `encoder._<chan>_.position-interpolated` (float, Out) - Position in scaled units, interpolated between encoder counts.
  The `position-interpolated` attempts to interpolate between encoder counts, based on the most recently measured velocity. Only valid when velocity is approximately constant and above `min-speed-estima`
  Do not use for position control, since its value is incorrect at low speeds, during direction reversals, and during speed changes.
  However, it allows a low ppr encoder (including a one pulse per revolution *encoder*) to be used for lathe threading, and may have other uses as well.
- `encoder._<chan>_.position-latched` (float, Out) - Not used at this time.

- encoder._<chan>_.position-scale (float, I/O) - Scale factor, in counts per length unit. For example, if position-scale is 500, then 1000 counts of the encoder will be reported as a position of 2.0 units.
- encoder._<chan>_.rawcounts (s32, In) - The raw count, as determined by update-counters. This value is updated more frequently than counts and position. It is also unaffected by reset or the index pulse.
- encoder._<chan>_.reset (bit, In) - When True, force *counts* and *position* to zero immediately.
- encoder._<chan>_.velocity (float, Out) - Velocity in scaled units per second. encoder uses an algorithm that greatly reduces quantization noise as compared to simply differentiating the *position* output. When the magnitude of the true velocity is below min-speed-estimate, the velocity output is 0.
- encoder._<chan>_.x4-mode (bit, I/O) (default: TRUE) - Enables times-4 mode. When true, the counter counts each edge of the quadrature waveform (four counts per full cycle). When false, it only counts once per full cycle. In counter-mode, this parameter is ignored. The 1x mode is useful for some jogwheels.

#### 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

The component exports two functions. Each function acts on all of the encoder counters - running different counters in different threads is not supported.

- (funct) encoder.update-counters - High speed function to count pulses (no floating point).
- (funct) encoder.capture-position - Low speed function to update latches and scale position.

### 5.8.4 PID

This component provides Proportional/Integral/Derivative control loops. It is a realtime component only. For simplicity, this discussion assumes that we are talking about position loops, however this component can be used to implement other feedback loops such as speed, torch height, temperature, etc. The PID Loop Block Diagram is a block diagram of a single PID loop.

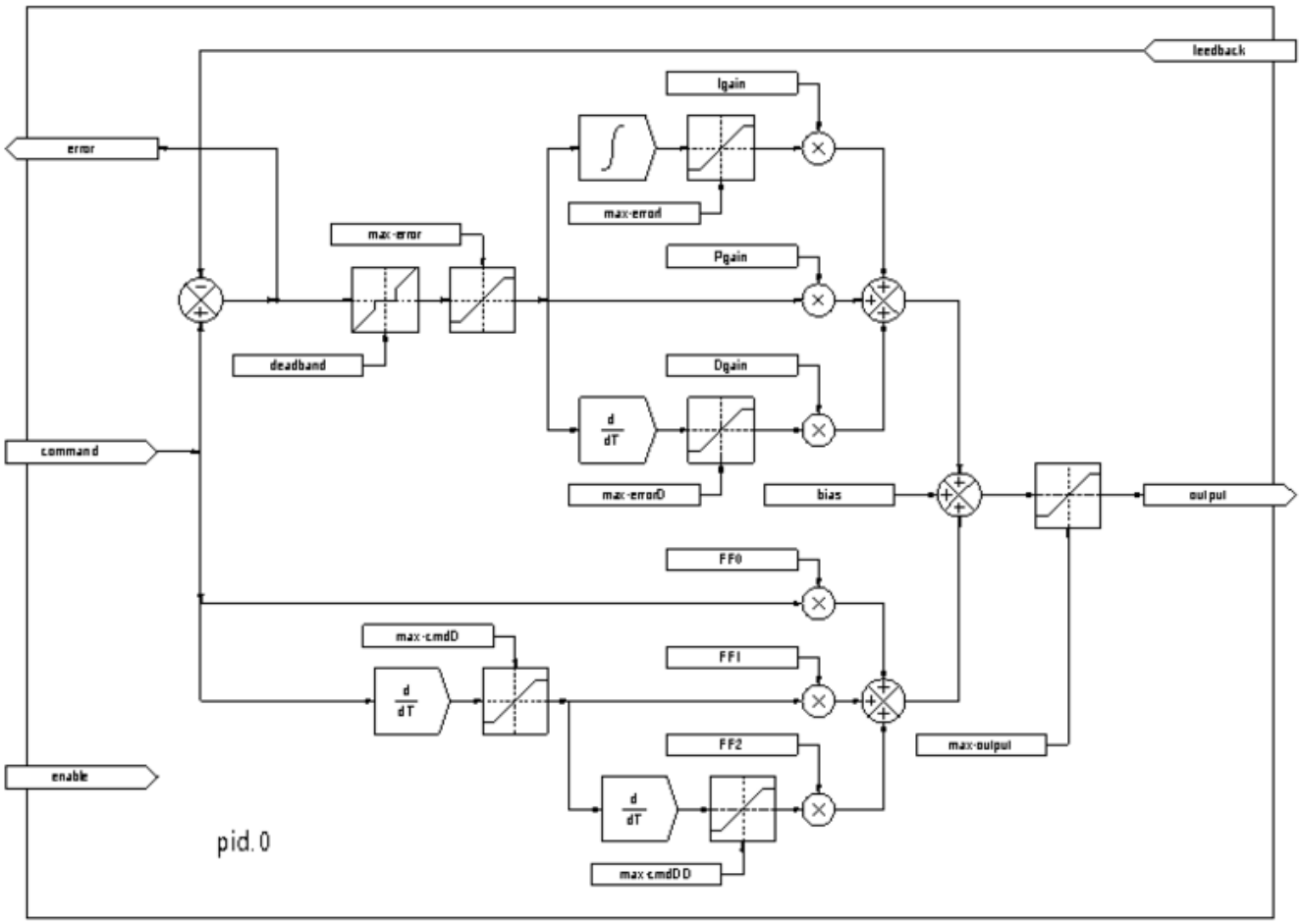



Abbildung 5.22: PID Loop Block Diagram

**PID laden**

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

*<loops>* is the number of PID loops that you want to install. If *num_chan* is not specified, one loop will be installed. The maximum number of loops is 16 (as defined by MAX_CHAN in pid.c). Each loop is completely independent. In the following descriptions, *<loopnum>* is the loop number of a specific loop. The first loop is number 0.

If `debug=1` is specified, the component will export a few extra pins that may be useful during debugging and tuning. By default, the extra pins are not exported, to save shared memory space and avoid cluttering the pin list.

**PID entfernen (engl. unload)**

```
halcmd: unloadrt pid
```

#### 5.8.4.1 Pins

Die drei wichtigsten Pins sind

- (float) `pid. `__<loopnum>__.command` - The desired position, as commanded by another system component.
- (float) `pid. `__<loopnum>__.feedback` - The present position, as measured by a feedback device such as an encoder.
- (float) `pid. `__<loopnum>__.output` - A velocity command that attempts to move from the present position to the desired position.

For a position loop, `.command` and `.feedback` are in position units. For a linear axis, this could be inches, mm, meters, or whatever is relevant. Likewise, for an angular axis, it could be degrees, radians, etc. The units of the `.output` pin represent the change needed to make the feedback match the command. As such, for a position loop `.output` is a velocity, in inches/s, mm/s, degrees/s, etc. Time units are always seconds, and the velocity units match the position units. If command and feedback are in meters, then output is in meters per second.

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* - A bit that enables the loop. If *.enable* is false, all integrators are reset, and the output is forced to zero. If *.enable* is true, the loop operates normally.

Pins zur Meldung der Sättigung. Eine Sättigung ist gegeben, wenn der Ausgang des PID-Blocks an seinem maximalen oder minimalen Grenzwert liegt.

- *(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

The component exports one function for each PID loop. This function performs all the calculations needed for the loop. Since each loop has its own function, individual loops can be included in different threads and execute at different rates.

- *(funct) pid.*<loopnum>*.do_pid_calcs* - Führt alle Berechnungen für eine einzelne PID-Schleife durch.

If you want to understand the exact algorithm used to compute the output of the PID loop, refer to - figure PID Loop Block Diagram, - the comments at the beginning of *emc2/src/hal/components/pid.c*, and of course to - the code itself. The loop calculations are in the C function *calc_pid()*.

### 5.8.5 Simulated Encoder

The simulated encoder is exactly that. It produces quadrature pulses with an index pulse, at a speed controlled by a HAL pin. Mostly useful for testing.

**Sim-Encoder laden**

```
halcmd: loadrt sim-encoder num_chan=<number>
```

*<number>* is the number of encoders that you want to simulate If not specified, one encoder will be installed. The maximum number is 8 (as defined by MAX_CHAN in sim_encoder.c).

**Abladen des sim-encoder**

```
halcmd: unloadrt sim-encoder
```

#### 5.8.5.1 Pins

- (float) `sim-encoder.` `__<chan-num>__.speed` - The speed command for the simulated shaft.
- (bit) `sim-encoder.` `__<chan-num>__.phase-A` - Quadrature output.
- (bit) `sim-encoder.` `__<chan-num>__.phase-B` - Quadrature output.
- (bit) `sim-encoder.` `__<chan-num>__.phase-Z` - Index pulse output.

When `.speed` is positive, `.phase-A` leads `.phase-B`.

#### 5.8.5.2 Parameter

- (u32) `sim-encoder.`__<chan-num>__.ppr` - Pulses Per Revolution.
- (float) `sim-encoder.`__<chan-num>__.scale` - Scale Factor for `.speed`. The default is 1.0, which means that `.speed` is in revolutions per second. Change to 60 for RPM, to 360 for degrees per second, 6.283185 (= 2*π) for radians per second, etc.

Note that pulses per revolution is not the same as counts per revolution. A pulse is a complete quadrature cycle. Most encoder counters will count four times during one complete cycle.

#### 5.8.5.3 Funktionen

Die Komponente exportiert zwei Funktionen. Jede Funktion wirkt auf alle simulierten Geber.

- (funct) `sim-encoder.make-pulses` - High speed function to generate quadrature pulses (no floating point).
- (funct) `sim-encoder.update-speed` - Low speed function to read `.speed`, do scaling, and set up `.make-pulses`.

### 5.8.6 Debounce

Debounce is a realtime component that can filter the glitches created by mechanical switch contacts. It may also be useful in other applications where short pulses are to be rejected.

**Debounce wird geladen**

```
halcmd: loadrt debounce cfg=<config-string>
```

***<config-string>***
: Is a series of comma separated decimal integers. Each number install a group of identical debounce filters, the number determines how many filters are in the group.

**Beispiel zum Laden von Debounce**

```
halcmd: loadrt debounce cfg=1,4,2
```

will install three groups of filters. Group 0 contains one filter, group 1 contains four, and group 2 contains two filters. The default value for <*config-string*> is ”1” which will install a single group containing a single filter. The maximum number of groups 8 (as defined by MAX_GROUPS in debounce.c). The maximum number of filters in a group is limited only by shared memory space. Each group is completely independent. All filters in a single group are identical, and they are all updated by the same function at the same time. In the following descriptions, <*G*> is the group number and <*F*> is the filter number within the group. The first filter is group 0, filter 0.

**Entladen der Entprellung**

```
halcmd: unloadrt debounce
```

#### 5.8.6.1 Pins

Jeder einzelne Filter hat zwei Pins.

- (bit) `debounce.`__<G>__.__<F>__.in` - Input of filter <*F*> in group <*G*>.
- (bit) `debounce.`__<G>__.__<F>__.out` - Output of filter <*F*> in group <*G*>.

#### 5.8.6.2 Parameter

Each group of filters has one parameter[5].

- (s32) `debounce.`__<G>__.delay` - Filter delay for all filters in group <*G*>.

The filter delay is in units of thread periods. The minimum delay is zero. The output of a zero delay filter exactly follows its input - it doesn't filter anything. As `.delay` increases, longer and longer glitches are rejected. If `.delay` is 4, all glitches less than or equal to four thread periods will be rejected.

#### 5.8.6.3 Funktionen

Each group of filters has one function, which updates all the filters in that group *simultaneously*. Different groups of filters can be updated from different threads at different periods.

- (funct) `debounce.`<*G*> - Updates all filters in group <*G*>.

### 5.8.7 SigGen

SigGen is a realtime component that generates square, triangle, and sine waves. It is primarily used for testing.

**Laden von siggen**

```
halcmd: loadrt siggen [num_chan=<chans>]
```

**<chans>**
: is the number of signal generators that you want to install. If *numchan* is not specified, one signal generator will be installed. The maximum number of generators is 16 (as defined by MAX_CHAN in siggen.c). Each generator is completely independent. In the following descriptions is

***<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` - Sine wave output.
- (float) `siggen.`__<chan>__.cosine` - Cosine output.
- (float) `siggen.`__<chan>__.sawtooth` - Sawtooth output.
- (float) `siggen.`__<chan>__.triangle` - Triangle wave output.
- (float) `siggen.`__<chan>__.square` - Square wave output.

[5] Each individual filter also has an internal state variable. There is a compile time switch that can export that variable as a parameter. This is intended for testing, and simply wastes shared memory under normal circumstances.

Alle fünf Ausgänge haben die gleiche Frequenz, Amplitude und Offset.

Zusätzlich zu den Ausgangspins gibt es drei Steuerpins:

- (float) `siggen. `__<chan>__.frequency` - Sets the frequency in Hertz, default value is 1 Hz.
- (float) `siggen. `__<chan>__.amplitude` - Sets the peak amplitude of the output waveforms, default is 1.
- (float) `siggen. `__<chan>__.offset` - Sets DC offset of the output waveforms, default is 0.

For example, if `siggen.0.amplitude` is 1.0 and `siggen.0.offset` is 0.0, the outputs will swing from -1.0 to +1.0. If `siggen.0.amplitude` is 2.5 and `siggen.0.offset` is 10.0, then the outputs will swing from 7.5 to 12.5.

#### 5.8.7.2 Parameter

None. [6]

#### 5.8.7.3 Funktionen

- (funct) `siggen. `__<chan>__.update` - Calculates new values for all five outputs.

### 5.8.8 `lut5`

The `lut5` component is a 5 input logic component based on a look up table.

- `lut5` does not require a floating point thread.

**Loading lut5**

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

**lut5 Computing Function** To compute the hexadecimal number for the function starting from the top put a 1 or 0 to indicate if that row would be true or false. Next write down every number in the output column starting from the top and writing them from right to left. This will be the binary number. Using a calculator with a program view like the one in Ubuntu enter the binary number and then convert it to hexadecimal and that will be the value for function.

Tabelle 5.6: `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 | |

[6] Prior to version 2.1, frequency, amplitude, and offset were parameters. They were changed to pins to allow control by other components.

Tabelle 5.6: (continued)

| Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Ausgabe |
|---|---|---|---|---|---|
| 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 Two Inputs Example** In der folgenden Tabelle haben wir für jede Zeile den Ausgangszustand ausgewählt, den wir für wahr halten wollen.

Tabelle 5.7: `lut5` Two Inputs Example 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 |

Looking at the output column of our example we want the output to be on when Bit 0 or Bit 0 and Bit1 is on and nothing else. The binary number is *b1010* (rotate the output 90 degrees CW). Enter this number into the calculator then change the display to hexadecimal and the number needed for function is *0xa*. The hexadecimal prefix is *0x*.

## 5.9 HAL Component Generator

This section introduces to the compilation HAL components, i.e. the addition of some machinists' knowledge on how to deal with the machine. It should be noted that such components do not necessarily deal with the hardware directly. They often do, but not necessarily, e.g. there could be a

component to convert between imperial and metric scales, so this section does not require to dive into the interaction with hardware.

Writing a HAL component can be a tedious process, most of it in setup calls to *rtapi_* and *hal_* functions and associated error checking. *halcompile* will write all this code for you, automatically. Compiling a HAL component is also much easier when using *halcompile*, whether the component is part of the LinuxCNC source tree, or outside it.

For instance, when coded in C, a simple component such as ”ddt” is around 80 lines of code. The equivalent component is very short when written using the *halcompile* preprocessor:

**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;
```

## 5.9.1 Installation

To compile a component, if a packaged version of LinuxCNC is used, development packages have to be installed using either Synaptic from the main menu *System -> Administration -> Synaptic package manager* or by running one of the following commands in a terminal window:

**Installation of Development packages for LinuxCNC**

```
sudo apt install linuxcnc-dev
# oder
sudo apt install linuxcnc-uspace-dev
```

Eine andere Methode ist die Verwendung des Synaptic-Paketmanagers aus dem Anwendungsmenü, um die Pakete `linuxcnc-dev` oder `linuxcnc-uspace-dev` zu installieren.

## 5.9.2 Compiling

### 5.9.2.1 Inside the source tree

Place the `.comp` file in the source directory `linuxcnc/src/hal/components` and re-run `make`. *Comp* files are automatically detected by the build system.

If a `.comp` file is a driver for hardware, it may be placed in `linuxcnc/src/hal/drivers` and will be built unless LinuxCNC is configured as a non-realtime simulator.

### 5.9.2.2 Realtime components outside the source tree

*halcompile* can process, compile, and install a realtime component in a single step, placing `rtexample.ko` in the LinuxCNC realtime module directory:

```
[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.

Or, it can process and compile in one step, leaving `example.ko` (or `example.so` for the simulator) in the current directory:

```
halcompile --compile rtexample.comp
```

Or it can simply process, leaving `example.c` in the current directory:

```
halcompile rtexample.comp
```

*halcompile* can also compile and install a component written in C, using the `--install` and `--compile` options shown above:

```
[sudo] halcompile --install rtexample2.c
```

Die Dokumentation im man-Format kann auch aus den Informationen im Deklarationsabschnitt erstellt werden:

```
halcompile --document -o example.9 rtexample.comp
```

Die resultierende Manpage „example.9“ kann angezeigt werden mit

```
man ./example.9
```

oder an einen Standardspeicherort für UNIX man pages kopiert.

#### 5.9.2.3 Non-realtime components outside the source tree

*halcompile* can process, compile, install, and document non-realtime components:

```
halcompile non-rt-example.comp
halcompile --compile non-rt-example.comp
[sudo] halcompile --install non-rt-example.comp
halcompile --document non-rt-example.comp
```

For some libraries (for example modbus) it might be necessary to add extra compiler and linker arguments to enable the compiler to find and link the libraries. In the case of .comp files this can be done via ”option” statements in the `.comp` file. For `.c` files this is not possible so the `--extra-compile-args` and `--extra-link-args` parameters can be used instead. As an example, this command line can be used to compile the `vfdb_vfd.c` component out-of-tree.

```
halcompile --userspace --install --extra-compile-args=”-I/usr/include/modbus” --extra-link- ↩
    args=”-lm -lmodbus -llinuxcncini” vfdb_vfd.c
```

**Anmerkung**
The effect of using both command-line and in-file extra-args is undefined.

### 5.9.3 Verwendung einer Komponente

Components need to be loaded and added to a thread before it can be employed. The provided functionality can then be invoked directly and repeatedly by one of the threads or it is called by other components that have their own respective triggers.

**Example HAL script for installing a component (ddt) and executing it every millisecond.**

```
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 Basics.

Um Ihre Komponente zu testen, können Sie den Beispielen im HAL Tutorial folgen.

### 5.9.4 Definitionen

- **component** - A component is a single real-time module, which is loaded with `Halcmd loadrt`. One `.comp` file specifies one component. The component name and file name must match.
- **instance** - A component can have zero or more instances. Each instance of a component is created equal (they all have the same pins, parameters, functions, and data) but behave independently when their pins, parameters, and data have different values.
- **singleton** - It is possible for a component to be a ”singleton”, in which case exactly one instance is created. It seldom makes sense to write a *singleton* component, unless there can literally only be a single object of that kind in the system (for instance, a component whose purpose is to provide a pin with the current UNIX time, or a hardware driver for the internal PC speaker).

### 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

A `.comp` file consists of a number of declarations, followed by `;;` on a line of its own, followed by C code implementing the module’s functions.

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;*

Parentheses indicate optional items. A vertical bar indicates alternatives. Words in *CAPITALS* indicate variable text, as follows:

- *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.

In the HAL identifier for a pin or parameter, # denotes an array item, and must be used in conjunction with a *[SIZE]* declaration. The hash marks are replaced with a 0-padded number with the same length as the number of # characters.

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* - An expression involving the variable *personality* which is nonzero when the pin or parameter should be created.
- *SIZE* - Eine Zahl, um die Größe eines Arrays anzugeben. Die Array-Elemente sind von 0 bis *SIZE*-1 nummeriert.
- *MAXSIZE : CONDSIZE* - A number that gives the maximum size of the array, followed by an expression involving the variable *personality* and which always evaluates to less than *MAXSIZE*. When the array is created its size will be *CONDSIZE*.
- *DOC* - Eine Zeichenfolge, die das Element dokumentiert. Die Zeichenfolge kann eine ”doppelt in Anführungszeichen” gesetzte Zeichenfolge im C-Stil sein, z. B.:

```
”Wählt die gewünschte Flanke aus: TRUE bedeutet fallend, FALSE bedeutet steigend”
```

  oder eine ”dreifach in Anführungszeichen” gesetzte Zeichenfolge im Python-Stil, die eingebettete Zeilenumbrüche und Anführungszeichen enthalten kann, z. B.:

```
”””Die Wirkung dieses Parameters, auch bekannt als ”der Orb von Zot”,
ist in mindestens zwei Absätzen zu erklären.

Hoffentlich haben Ihnen diese Absätze geholfen, ”zot” besser
zu verstehen.”””
```

  Einer Zeichenkette kann auch das Literalzeichen *r* vorangestellt werden; in diesem Fall wird die Zeichenkette wie eine Python-Rohzeichenkette interpretiert.

  The documentation string is in ”groff -man” format. For more information on this markup format, see *groff_man(7)*. Remember that *halcompile* interprets backslash escapes in strings, so for instance to set the italic font for the word *example*, write:

```
”\\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* - One of the HAL types: *bit*, *signed*, *unsigned*, or *float*. The old names *s32* and *u32* may also be used, but *signed* and *unsigned* are preferred.
- *PINDIRECTION* - One of the following: *in*, *out*, or *io*. A component sets a value for an *out* pin, it reads a value from an *in* pin, and it may read or set the value of an *io* pin.
- *PARAMDIRECTION* - 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* - Specifies the initial value of a pin or parameter. If it is not specified, then the default is *0* or *FALSE*, depending on the type of the item.
- *HEADERFILE* - Der Name einer Headerdatei, entweder in doppelten Anführungszeichen (*include ”myfile.h”;*) oder in spitzen Klammern (*include <systemfile.h>;*). Die Header-Datei wird (unter Verwendung der #include von C) am Anfang der Datei vor Pin- und Parameterdeklarationen eingefügt.

#### 5.9.7.1 HAL-Funktionen

- *fp* - Gibt an, dass die Funktion Gleitkommaberechnungen durchführt.
- *nofp* - Indicates that it only performs integer calculations. If neither is specified, *fp* is assumed. Neither *halcompile* nor gcc can detect the use of floating-point calculations in functions that are tagged *nofp*, but the use of such operations results in undefined behavior.

#### 5.9.7.2 Optionen

Die derzeit definierten Optionen sind:

- *option singleton yes* - (default: no) Do not create a *count* module parameter, and always create a single instance. With *singleton*, items are named *component-name.item-name* and without *singleton*, items for numbered instances are named *component-name.<num>.item-name*.
- 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* - (default: yes) Normally, the functions `rtapi_app_main()` and `rtapi_app_exit()` are automatically defined. With *option rtapi_app no*, they are not, and must be provided in the C code. Use the following prototypes:

  ```
  'int rtapi_app_main(void);'

  'void rtapi_app_exit(void);'
  ```

  Wenn Sie Ihre eigene `rtapi_app_main()` implementieren, rufen Sie die Funktion `int export(char *prefix, long extra_arg)` auf, um die Pins, Parameter und Funktionen für `prefix` zu registrieren.
- *option data TYPE* - (default: none) **deprecated**
  If specified, each instance of the component will have an associated data block of type *TYPE* (which can be a simple type like *float* or the name of a type created with *typedef*). In new components, *variable* should be used instead.
- *option extra_setup yes* - (default: no)
  If specified, call the function defined by *EXTRA_SETUP* for each instance. If using the automatically defined *rtapi_app_main*, *extra_arg* is the number of this instance.
- *option extra_cleanup yes* - (default: no)
  If specified, call the function defined by *EXTRA_CLEANUP* from the automatically defined *rtapi_app_exit* or, in case of a detected error, in the automatically defined *rtapi_app_main*.
- *option userspace yes* - (default: no)
  If specified, this file describes a non-realtime (formerly known as "userspace") component, rather than a regular (i.e., realtime) one. A non-realtime component may not have functions defined by the *function* directive. Instead, after all the instances are constructed, the C function `void user_mainloop(void);` is called. When this function returns, the component exits. Typically, *user_mainloop()* will use *FOR_ALL_INSTS()* to perform the update action for each instance, then sleep for a short time. Another common action in *user_mainloop()* may be to call the event handler loop of a GUI toolkit.

- *option userinit yes* - (default: no)
  This option is ignored if the option *userspace* (see above) is set to *no*. If *userinit* is specified, the function *userinit(argc,argv)* is called before *rtapi_app_main()* (and thus before the call to *hal_init()* ). This function may process the commandline arguments or take other actions. Its return type is *void*; it may call *exit()* if it wishes to terminate rather than create a HAL component (e.g., because the commandline arguments were invalid).
- *option extra_link_args "..."* - (default: "") This option is ignored if the option *userspace* (see above) is set to *no*. When linking a non-realtime component, the arguments given are inserted in the link line. Note that because compilation takes place in a temporary directory, "-L." refers to the temporary directory and not the directory where the .comp source file resides. This option can be set in the halcompile command-line with -extra-link-args="-L.....". This alternative provides a way to set extra flags in cases where the input file is a .c file rather than a .comp file.
- *option extra_compile_args "..."* - (default: "") This option is ignored if the option *userspace* (see above) is set to *no*. When compiling a non-realtime component, the arguments given are inserted in the compiler command line. If the input file is a .c file this option can be set in the halcompile command-line with --extra-compile-args="-I.....". This alternative provides a way to set extra flags in cases where the input file is a .c file rather than a .comp file.
- *option homemod yes* - (default: no)
  Module is a custom Homing module loaded using `[EMCMOT]HOMEMOD=`*modulename* .
- *option tpmod yes* - (default: no)
  Module is a custom Trajectory Planning (tp) module loaded using `[TRAJ]TPMOD=`*modulename* .

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` - Specify the license of the module for the documentation and for the MODULE_LICENSE() module declaration. For example, to specify that the module's license is GPL v2 or later:

  ```
  'license "GPL"; // indicates GPL v2 or later'
  ```

  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` - Specify the author of the module for the documentation.

#### 5.9.7.4 Datenspeicherung pro Instanz

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARRED = DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Declare a per-instance variable *STARREDNAME* of type *CTYPE*, optionally as an array of *SIZE* items, and optionally with a default value *DEFAULT*. Items with no *DEFAULT* are initialized to all-bits-zero. *CTYPE* is a simple one-word C type, such as `float`, `u32`, `s32`, `int`, etc. Access to array variables uses square brackets.

If a variable is to be of a pointer type, there may not be any space between the "*" and the variable name. Therefore, the following is acceptable:

```
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

Based on the items in the declaration section, *halcompile* creates a C structure called `struct __comp_state`. However, instead of referring to the members of this structure (e.g., `*(inst->name)`), they will generally be referred to using the macros below. The details of `struct __comp_state` and these macros may change from one version of *halcompile* to the next.

- `FUNCTION( __name__)` - Use this macro to begin the definition of a realtime function, which was previously declared with *function NAME*. The function includes a parameter *period* which is the integer number of nanoseconds between calls to the function.
- `EXTRA_SETUP()` - Use this macro to begin the definition of the function called to perform extra setup of this instance. Return a negative Unix *errno* value to indicate failure (e.g., *return -EBUSY* on failure to reserve an I/O port), or 0 to indicate success.
- `EXTRA_CLEANUP()` - Use this macro to begin the definition of the function called to perform extra cleanup of the component. Note that this function must clean up all instances of the component, not just one. The ”pin_name”, ”parameter_name”, and ”data” macros may not be used here.

- *pin_name* or *parameter_name* - For each pin *pin_name* or param *parameter_name* there is a macro which allows the name to be used on its own to refer to the pin or parameter. When *pin_name* or *parameter_name* is an array, the macro is of the form *pin_name(idx)* or *param_name(idx)*, where *idx* is the index into the pin array. When the array is a variable-sized array, it is only legal to refer to items up to its *condsize*.

  Wenn es sich um eine bedingte Position handelt, kann nur auf sie verwiesen werden, wenn ihre "Bedingung" einen Wert ungleich Null ergibt.

- *variable_name* - For each variable *variable_name* there is a macro which allows the name to be used on its own to refer to the variable. When *variable_name* is an array, the normal C-style subscript is used: *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() {...}` - For non-realtime components. This macro iterates over all the defined instances. Inside the body of the loop, the *pin_name*, *parameter_name*, and *data* macros work as they do in realtime functions.

### 5.9.10 Komponenten mit einer Funktion

If a component has only one function and the string `"FUNCTION"` does not appear anywhere after `;;`, then the portion after `;;` is all taken to be the body of the component's single function. See the Simple Comp for an example of this.

### 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 Beispiele

### 5.9.12.1 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; }
```

### 5.9.12.2 sincos

This component computes the sine and cosine of an input angle in radians. It has different capabilities than the ”sine” and ”cosine” outputs of siggen, because the input is an angle, rather than running freely based on a ”frequency” parameter.

The pins are declared with the names *sin_* and *cos_* in the source code so that they do not interfere with the functions *sin()* and *cos()*. The HAL pins are still called *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); }
```

### 5.9.12.3 out8

This component is a driver for a *fictional* card called ”out8”, which has 8 pins of digital output which are treated as a single 8-bit value. There can be a varying number of such cards in the system, and they can be at various addresses. The pin is called *out_* because *out* is an identifier used in *<asm/io.h>*. It illustrates the use of *EXTRA_SETUP* and *EXTRA_CLEANUP* to request an I/O region and then free it in case of error or when the module is unloaded.

```
component out8;
pin out 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); }
```

#### 5.9.12.4 hal_loop

```
component hal_loop;
pin out float example;
```

This fragment of a component illustrates the use of the *hal_* prefix in a component name.

`loop` is a common name, and the `hal_` prefix avoids potential name collisions with other unrelated software. For example, on RTAI realtime systems realtime code runs in the kernel, so if the component were named just `loop` it could easily conflict with the standard `loop` kernel module.

When loaded, *halcmd show comp* will show a component called *hal_loop*. However, the pin shown by *halcmd show pin* will be *loop.0.example*, not *hal-loop.0.example*.

#### 5.9.12.5 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;
```

#### 5.9.12.6 rand

This non-realtime component changes the value on its output pin to a new random value in the range (0,1) about once every 1 ms.

```
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();
    }
}
```

#### 5.9.12.7 logic (using personality)

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`

#### 5.9.12.8 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.

### 5.9.13 Verwendung der Kommandozeile

The halcompile man page gives details for invoking `halcompile`.

```
$ man halcompile
```

Eine kurze Zusammenfassung der Verwendung von halcompile finden Sie hier:

```
$ halcompile --help
```

# 5.10 HALTCL-Dateien

halcmd excels in specifying components and connections but these scripts offer no computational capabilities. As a result, INI files are limited in the clarity and brevity that is possible with higher level languages.

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.

The .tcl extension is understood by the main script (`linuxcnc`) that processes INI files. Haltcl files are identified in the the HAL section of INI files (just like HAL files).

**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

INI file variables are accessible by both `halcmd` and `haltcl` but with differing syntax. LinuxCNC INI files use SECTION and ITEM specifiers to identify configuration items:

```
[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.

When the value could contain special characters (quote characters, curly-brace characters, embedded whitespace, and other characters that have special meaning in Tcl) then it is necessary to distinguish between the list of values and the initial (and possibly only) value in the list.

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

Consider the topic of *stepgen headroom*. Software `stepgen` runs best with an acceleration constraint that is "a bit higher" than the one used by the motion planner. So, when using `halcmd` files, we force INI files to have a manually calculated value.

```
[JOINT_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

With `haltcl`, you can use Tcl commands to do the computation and eliminate the `STEPGEN_MAXACCEL` INI file item altogether:

```
setp stepgen.0.maxaccel $::JOINT_0(MAXACCEL)*1.05
```

Another `haltcl` feature is looping and testing. For example, many simulator configurations use "core_sim.hal" or "core_sim9.hal" HAL files. These differ because of the requirement to connect more or fewer axes. The following haltcl code would work for any combination of axes in a trivkins machine.

```
# 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

The `halrun` command recognizes haltcl files. With the -T option, haltcl can be run interaactively as a Tcl interpreter. This capability is useful for testing and for standalone HAL applications.

**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 HAL-Benutzeroberfläche

### 5.11.1 Einführung

Halui ist eine HAL-basierte Benutzeroberfläche für LinuxCNC, es verbindet HAL-Pins mit NML-Befehlen. Die meisten Funktionen (Schaltflächen, Anzeigen usw.), wie von einer traditionellen GUI (AXIS, GMOCCAPY, QtDragon, etc.) zur Verfügung gestellt, werden von HAL-Pins in Halui übernommen.

Der einfachste Weg, halui hinzuzufügen, besteht darin, das Folgende in den [HAL]-Abschnitt der INI-Datei einzufügen:

```
[HAL]
HALUI = halui
```

Ein alternativer Weg, es aufzurufen (besonders, wenn Sie die Konfiguration mit StepConf erzeugen), ist, das Folgende in Ihre `custom.hal`-Datei aufzunehmen.
Stellen Sie sicher, dass Sie den richtigen Pfad zu Ihrer INI-Datei verwenden.

```
loadusr halui -ini /path/to/inifile.ini
```

### 5.11.2 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 5.1** 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).

### 5.11.3 Beispiel-Konfiguration

Eine Beispiel-Sim-Konfiguration (`configs/sim/axis/halui_pyvcp/halui.ini`) ist in der Distribution enthalten.

### 5.11.4 Halui-Pin-Referenz

Alle halui-Pins sind auch in der halui-Manualseite dokumentiert:

```
$ man halui
```

Or see http://linuxcnc.org/docs/stable/html/man/man1/halui.1.html

#### 5.11.4.1 Abbrechen

- halui.abort' (bit, in) - Pin zum Senden einer Abbruchmeldung (löscht die meisten Fehler)

#### 5.11.4.2 Notaus (engl. 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

#### 5.11.4.3 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

#### 5.11.4.4 Nebel (engl. 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

#### 5.11.4.5 Flut-Kühlmittel (engl. 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

#### 5.11.4.6 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.

#### 5.11.4.7 Schmiermittel (engl. 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

#### 5.11.4.8 Maschine

- 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

#### 5.11.4.9 Max. Geschwindigkeit

Die maximale lineare Geschwindigkeit kann zwischen 0 und der MAX_VELOCITY eingestellt werden, die im Abschnitt [TRAJ] der INI-Datei festgelegt ist.

- *halui.max-velocity.count-enable* (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.

#### 5.11.4.10 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.

#### 5.11.4.11 Gelenk

*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

#### 5.11.4.12 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

#### 5.11.4.13 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

#### 5.11.4.14 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

#### 5.11.4.15 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

#### 5.11.4.16 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

#### 5.11.4.17 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)

#### 5.11.4.18 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)

#### 5.11.4.19 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

#### 5.11.4.20 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

## 5.12 Halui Examples

For any Halui examples to work you need to add the following line to the [HAL] section of the ini file.

```
HALUI = halui
```

### 5.12.1 Ferngesteuerter Start

To connect a remote program start button to LinuxCNC you use the `halui.program.run` pin and the `halui.mode.auto` pin. You have to ensure that it is OK to run first by using the `halui.mode.is-auto` pin. You do this with an `and2` component. The following figure shows how this is done. When the Remote Run Button is pressed it is connected to both `halui.mode.auto` and `and2.0.in0`. If it is OK for auto mode the pin `halui.mode.is-auto` will be on. If both the inputs to the `and2.0` component are on the `and2.0.out` will be on and this will start the program.

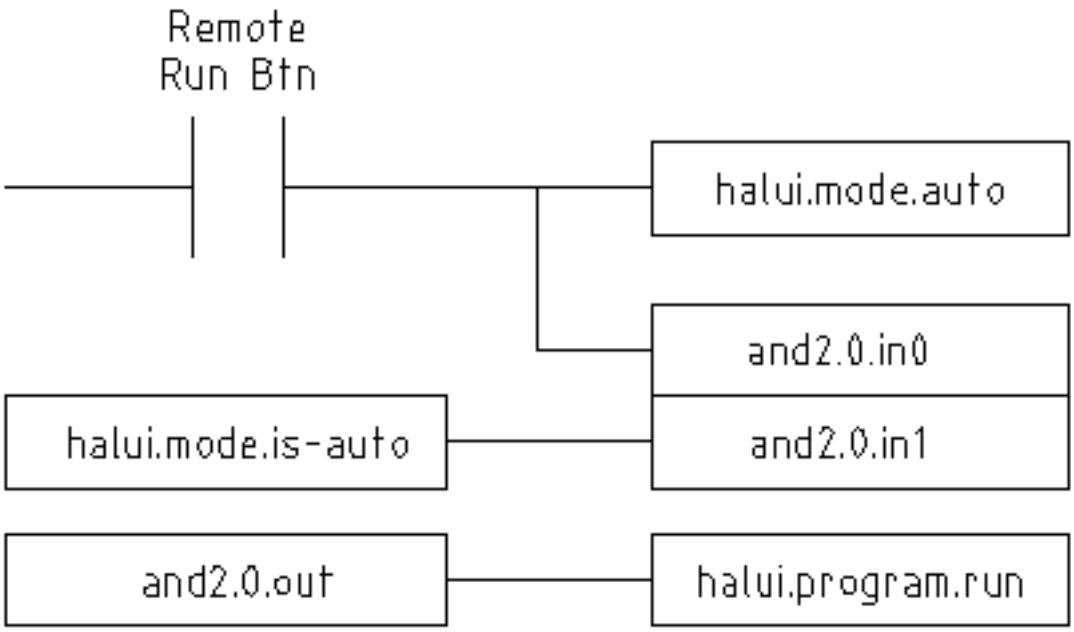



Abbildung 5.23: Beispiel für Fernstart

Die für das Vorstehende erforderlichen HAL-Befehle sind:

```
net program-start-btn halui.mode.auto and2.0.in0 <= <your input pin>
net program-run-ok and2.0.in1 <= halui.mode.is-auto
net remote-program-run halui.program.run <= and2.0.out
```

Beachten Sie, dass es in der ersten Zeile zwei Leser-Pins gibt, die auch in zwei Zeilen aufgeteilt werden können:

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

### 5.12.2 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 your customized HAL file, add the following two lines that will be connected to your I/O to turn on the program pause or to resume when the external system wants LinuxCNC to continue.

```
net ispaused halui.program.is paused => ”Dein output (Ausgabe) Pin”
net resume halui.program.resume <= ”your input (Eingabe) Pin”
```

Ihre Eingangs- und Ausgangspins sind mit den Pins verbunden, die mit dem anderen Controller verdrahtet sind. Dabei kann es sich um Pins des Parallelports oder andere E/A-Pins handeln, auf die Sie Zugriff haben.

Dieses System funktioniert auf folgende Weise. Wrid ein M0 in Ihrem G-Code erreicht, so wird das Signal ”halui.program.is-paused” wahr. Dies schaltet auf Ihrem Ausgangspin, so dass die externe Steuerung weiß, dass LinuxCNC pausiert ist.

Um die LinuxCNC G-Code-Programm fortzusetzen, wenn die externe Steuerung bereit ist, wird es seine Ausgabe wahr zu machen. Dies wird LinuxCNC signalisieren, dass es die Ausführung von G-Code fortsetzen sollte.

Schwierigkeiten beim Timing

- Das ”Resume”-Eingangsrücksignal sollte nicht länger sein als die Zeit, die benötigt wird, um den G-Code wieder zum Laufen zu bringen.
- Der Ausgang ”is-paused” sollte nicht mehr aktiv sein, wenn das ”resume”-Signal endet.

Diese Timing-Probleme könnten vermieden werden, indem ClassicLadder verwendet wird, um den ”is-paused”-Ausgang über einen monostabilen Timer zu aktivieren und einen schmalen Ausgangsimpuls zu liefern. Der ”Resume”-Puls könnte ebenfalls über einen monostabilen Timer empfangen werden.

## 5.13 Creating Non-realtime Python Components

This section explains principles behind the implementation of HAL components with the Python programming language.

### 5.13.1 Basic usage example

A non-realtime component begins by creating its pins and parameters, then enters a loop which will periodically drive all the outputs from the inputs. The following component copies the value seen on its input pin (*passthrough.in*) to its output pin (*passthrough.out*) approximately once per second.

```
#!/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
```

Copy the above listing into a file named ”passthrough”, make it executable (*chmod +x*), and place it on your *$PATH*. Then try it out:

**Screen copy with details on the execution of the newly created passthrough HAL module.**

```
$ 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
```

### 5.13.2 Non-realtime components and delays

If you typed ”show pin” quickly, you may see that *passthrough.out* still had its old value of 0. This is because of the call to *time.sleep(1)*, which makes the assignment to the output pin occur at most once per second. Because this is a non-realtime component, the actual delay between assignments can be much longer if the memory used by the passthrough component is swapped to disk, as the assignment could be delayed until that memory is swapped back in.

Thus, non-realtime components are suitable for user-interactive elements such as control panels (delays in the range of milliseconds are not noticed, and longer delays are acceptable), but not for sending step pulses to a stepper driver board (delays must always be in the range of microseconds, no matter what).

### 5.13.3 Pins und Parameter erstellen

```
h = hal.component(”passthrough”)
```

The component itself is created by a call to the constructor *hal.component*. The arguments are the HAL component name and (optionally) the prefix used for pin and parameter names. If the prefix is not specified, the component name is used.

```
h.newpin(”in”, hal.HAL_FLOAT, hal.HAL_IN)
```

Then pins are created by calls to methods on the component object. The arguments are: pin name suffix, pin type, and pin direction. For parameters, the arguments are: parameter name suffix, parameter type, and parameter direction.

Tabelle 5.8: 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.

#### 5.13.3.1 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.

### 5.13.4 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}
```

#### 5.13.4.1 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.

#### 5.13.4.2 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.

### 5.13.5 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.

### 5.13.6 Hilfreiche Funktionen

See Python HAL Interface for an overview of available functions.

### 5.13.7 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

### 5.13.8 System-Informationen

Lesen Sie diese, um Informationen über das Echtzeitsystem zu erhalten.

- is_kernelspace
- is_rt
- is_sim
- is_userspace

## 5.14 Canonical Device Interfaces

The following sections show the pins, parameters, and functions that are supplied by ”canonical devices”. All HAL device drivers should supply the same pins and parameters, and implement the same behavior.

Note that only the `_<io-type>_` and `_<specific-name>_` fields are defined for a canonical device. The `_<device-name>`, `_<device-num>_`, and `_<chan-num>_` fields are set based on the characteristics of the real device.

## 5.14.1 Digital Input

The canonical digital input (I/O type field: `digin`) is quite simple.

### 5.14.1.1 Pins

**(bit) in**
Zustand des Hardware-Eingangs.

**(bit) in-not**
Invertierter Zustand des Eingangs.

### 5.14.1.2 Parameter

None

### 5.14.1.3 Funktionen

(funct) **read**:: Lesen der Hardware und Setzen der HAL-Pins ”in” und ”in-not”.

## 5.14.2 Digital Output

The canonical digital output (I/O type field: `digout`) is also very simple.

### 5.14.2.1 Pins

(bit) **out**:: Wert, der (eventuell invertiert) an den Hardware-Ausgang geschrieben werden soll.

### 5.14.2.2 Parameter

(bit) **invert**:: Wenn TRUE, wird **out** vor dem Schreiben in die Hardware invertiert.

### 5.14.2.3 Funktionen

(funct) **write**:: Lesen Sie **out** und **invert** und stellen Sie die Hardwareausgabe entsprechend ein.

## 5.14.3 Analog Input

The canonical analog input (I/O type: `adcin`). This is expected to be used for analog to digital converters, which convert e.g. voltage to a continuous range of values.

### 5.14.3.1 Pins

(float) **Wert**:: Der Hardware-Messwert, skaliert gemäß den Parametern **Skala** und **Offset**.
**Wert** = ((Eingangsmesswert, in hardwareabhängigen Einheiten) * **Skala**) - **Offset**

#### 5.14.3.2 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**:: The value of one least significant bit (LSB). This is effectively the granularity of the input reading. (float) **hw_offset**:: The value present on the input when 0 Volts is applied to the input pin(s).

#### 5.14.3.3 Funktionen

(funct) **read**:: Read the values of this analog input channel. This may be used for individual channel reads, or it may cause all channels to be read.

### 5.14.4 Analog Output

The canonical analog output (I/O Type: **adcout**). This is intended for any kind of hardware that can output a more-or-less continuous range of values. Examples are digital to analog converters or PWM generators.

#### 5.14.4.1 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.

#### 5.14.4.2 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):: The actual voltage (or current) that will be output if 0 is written to the hardware.

#### 5.14.4.3 Funktionen

(funct) **write**:: This causes the calculated value to be output to the hardware. If enable is false, then the output will be 0, regardless of **value**, **scale**, and **offset**. The meaning of ”0” is dependent on the hardware. For example, a bipolar 12-bit A/D may need to write 0x1FF (mid scale) to the D/A get 0 volts from the hardware pin. If enable is true, read scale, offset and value and output to the adc (**scale** * **value**) + **offset**. If enable is false, then output 0.

# 5.15 HAL Tools

## 5.15.1 Halcmd

`halcmd` is a command line tool for manipulating the HAL. There is a rather complete man page for `halcmd`, which will be installed if you have installed LinuxCNC from either source or a package. The manpage provides usage info:

```
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`.

## 5.15.2 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. 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 ``halmeter``s can be open at the same time. If you use a script to open multiple ``halmeter``s you can set the position of each one with `-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.

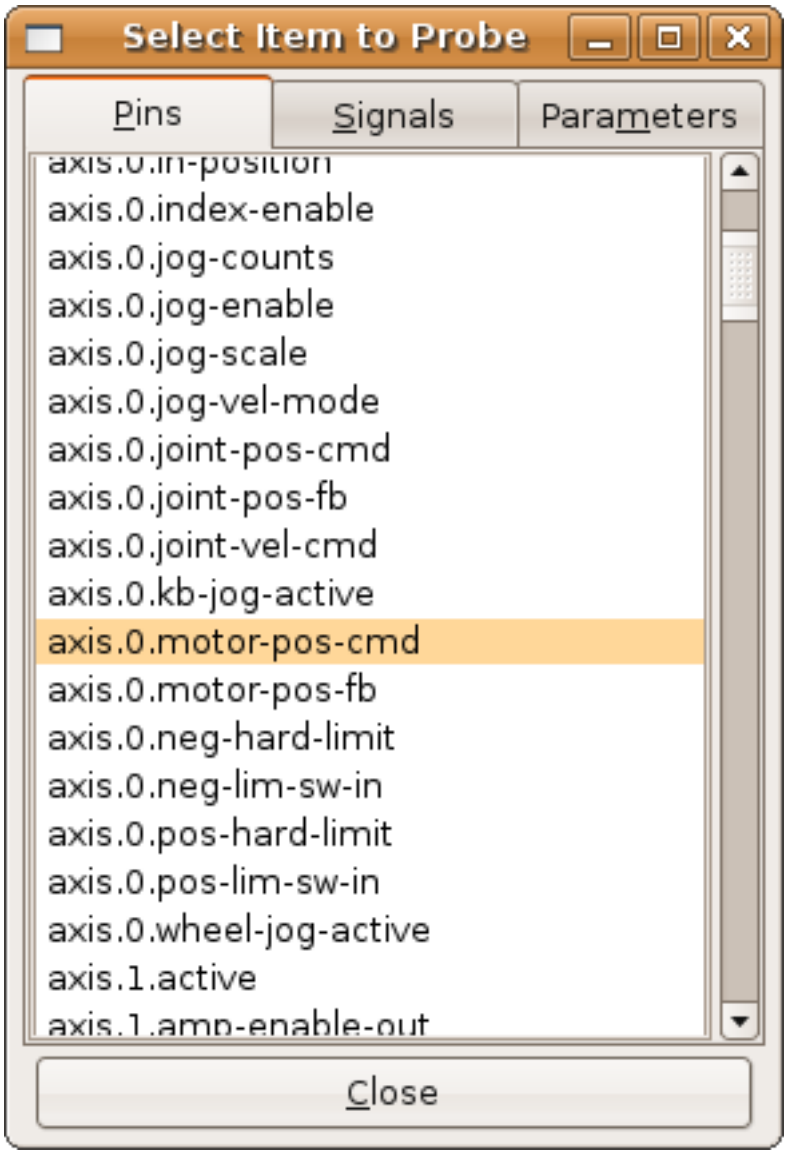



Abbildung 5.24: Halmeter-Auswahlfenster

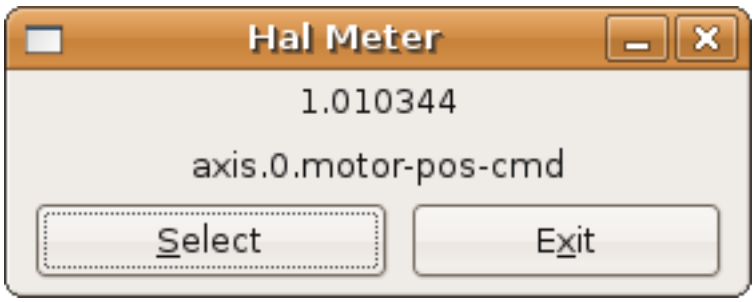



Abbildung 5.25: Halmeter-Watch Fenster

### 5.15.3 Halshow

`halshow` (complete usage description) can be started from the command line to show details for selected components, pins, parameters, signals, functions, and threads of a running HAL. The WATCH tab provides a continuous display of selected pin, parameters, and signal items. The File menu provides buttons to save the watch items to a watch list and to load an existing watch list. The watch list items can also be loaded automatically on startup. For command line usage:

```
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.
```

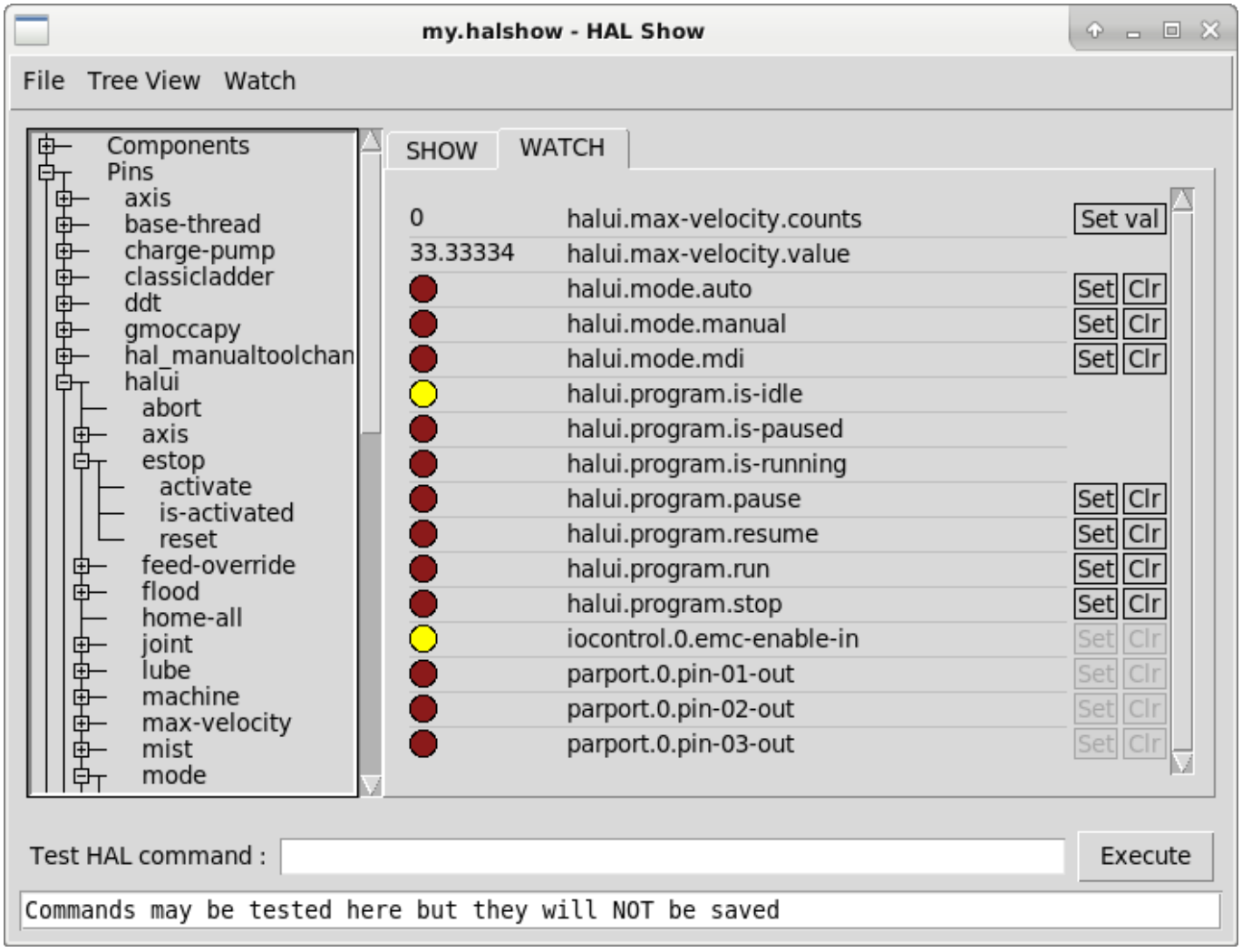



Abbildung 5.26: Halshow Watch Tab

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.

### 5.15.4 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 Halscope 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.

Configuration files may also be specified when starting halscope from the commandline. Commandline help (`-h`) usage:

```
halscope -h
Usage:
  halscope [-h] [-i infile] [-o outfile] [num_samples]
```

### 5.15.5 Sim-Pin

sim_pin ist ein Kommandozeilenprogramm zur Anzeige und Aktualisierung einer beliebigen Anzahl von beschreibbaren Pins, Parametern oder Signalen.

**sim_pin Usage**

```
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 item types bit,s32,u32,float are supported.

        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 Example (with LinuxCNC running)**

```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0
sim_pin example.sel example.in1 sig_example &
```

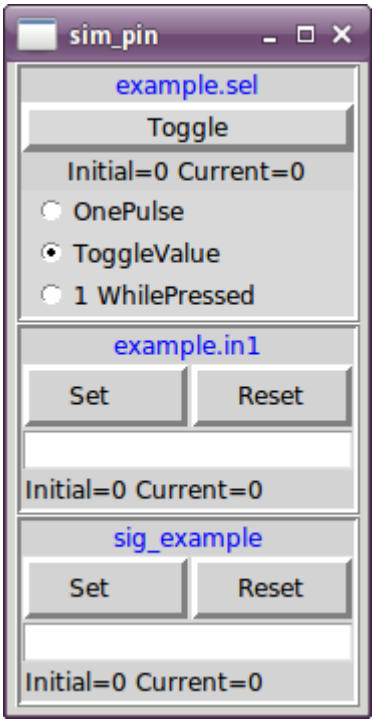



Abbildung 5.27: sim_pin Fenster

## 5.15.6 simulate_probe (Sonde simulieren)

`simulate_probe` ist ein einfaches GUI, um die Aktivierung des Pins motion.probe-input zu simulieren. Verwendung:

```
simulate_probe &
```

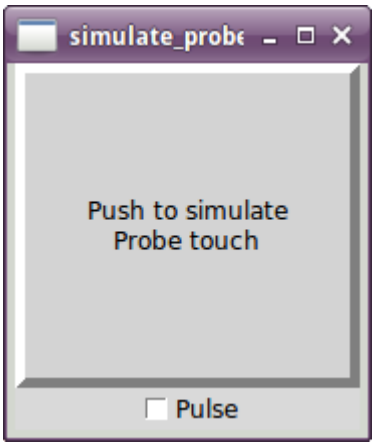



Abbildung 5.28: simulieren_probe Fenster

### 5.15.7 HAL Histogramm

`hal-histogram` ist ein Kommandozeilenprogramm zur Anzeige von Histogrammen für HAL-Pins.

**Nutzung:**

```
   hal-histogram --help | -?
or
   hal-histogram [Optionen] [Pinname]
```

Tabelle 5.9: Optionen:

| 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. The pin must be associated with a thread supporting floating point. For a base thread, this may require using `loadrt motmod ... base_thread_fp=1` .

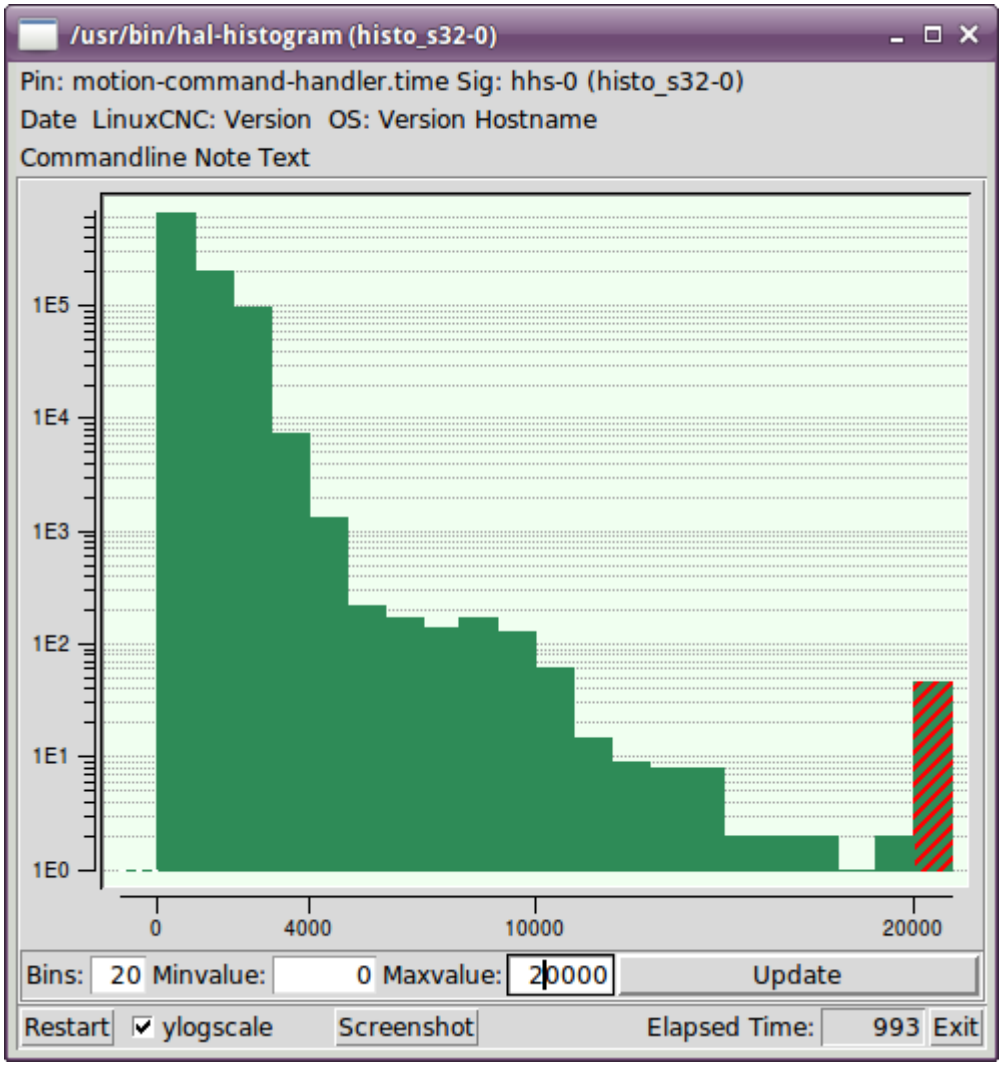



Abbildung 5.29: hal-histogram Fenster

### 5.15.8 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. Non-realtime component pins having non-ordered functions.
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 Usage**

```
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 Example**

```
[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. Read input pins,
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.

For routine signal paths that handle switch inputs, non-realtime pins, etc., the addf-ordering is often not critical. Moreover, the timing of non-realtime pin value changes cannot be controlled or guaranteed at the intervals typically employed for HAL threads.

Example report file excerpts showing a pid loop for a hostmot2 stepgen operated in velocity mode on a trivkins machine with `joint.0` corresponding to the X axis coordinate:

```
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-created non-realtime components that lack distinguishing characteristics such as a prefix based on the GUI program name.

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

The hal_parport component is a driver for the traditional PC parallel port. The port has a total of 17 physical pins. The original parallel port divided those pins into three groups: data, control, and status. The data group consists of 8 output pins, the control group consists of 4 pins, and the status group consists of 5 input pins.

In the early 1990s, the bidirectional parallel port was introduced, which allows the data group to be used for output or input. The HAL driver supports the bidirectional port, and allows the user to set the data group as either input or output. If configured as *out*, a port provides a total of 12 outputs and 5 inputs. If configured as *in*, it provides 4 outputs and 13 inputs.

In some parallel ports, the control group pins are open collectors, which may also be driven low by an external gate. On a board with open collector control pins. If configured as *x*, it provides 8 outputs, and 9 inputs.

Bei einigen parallelen Anschlüssen verfügt die Steuergruppe über Push-Pull-Treiber und kann nicht als Eingang verwendet werden.

---

**HAL und Open Collectors**
HAL cannot automatically determine if the *x* mode bidirectional pins are actually open collectors (OC). If they are not, they cannot be used as inputs, and attempting to drive them LOW from an external source can damage the hardware.
To determine whether your port has *open collector* pins, load hal_parport in *x* mode. With no device attached, HAL should read the pin as TRUE. Next, insert a 470 Ω resistor from one of the control pins to GND. If the resulting voltage on the control pin is close to 0 V, and HAL now reads the pin as FALSE, then you have an OC port. If the resulting voltage is far from 0 V, or HAL does not read the pin as FALSE, then your port cannot be used in *x* mode.
The external hardware that drives the control pins should also use open collector gates, e.g. 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

The hal_parport driver is a real time component so it must be loaded into the real time thread with *loadrt*. The configuration string describes the parallel ports to be used, and (optionally) their types. If the configuration string does not describe at least one port, it is an error.

```
loadrt hal_parport cfg="Anschluss [Typ] [Anschluss [Typ] ...]"
```

**Angabe des Ports** Numbers below 16 refer to parallel ports detected by the system. This is the simplest way to configure the `hal_parport` driver and cooperates with the Linux parport_pc driver if it is loaded. A port of 0 is the first parallel port detected on the system, 1 is the next and so on.

**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.

The config string represents the hexadecimal address of the port, optionally followed by a direction, all repeated for each port. The directions are *in*, *out*, or *x*, and determine the direction of the physical pins 2 to 9 of the D-Sub 25 connector. If the direction is not specified, the data group will by default be configured as outputs. For example:

**Command to load the real-time module hal_partport with the additional <config-string> to specify the port at which the parallel-port card is expected.**

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

This example installs the drivers for a port 0x0278, with pins 2 to 9 as outputs (by default, since neither *in* nor *out* is specified), a port 0x0378, with pins 2 to 9 as inputs and a 0x20A0 port, with pins 2 to 9 explicitly specified as outputs. Note that you must know the base address of the parallel ports to configure the drivers correctly. For ISA bus ports, this is usually not a problem, since the ports are almost always at a well-known address, such as 0x278 or 0x378 which are typically configured in the BIOS. The addresses of PCI bus cards are usually found with `lspci -v` in an *I/O ports* line, or in a kernel message after running `sudo modprobe -a parport_pc`. There is no default address, so if `<config-string>` does not contain at least one address, it is an error.

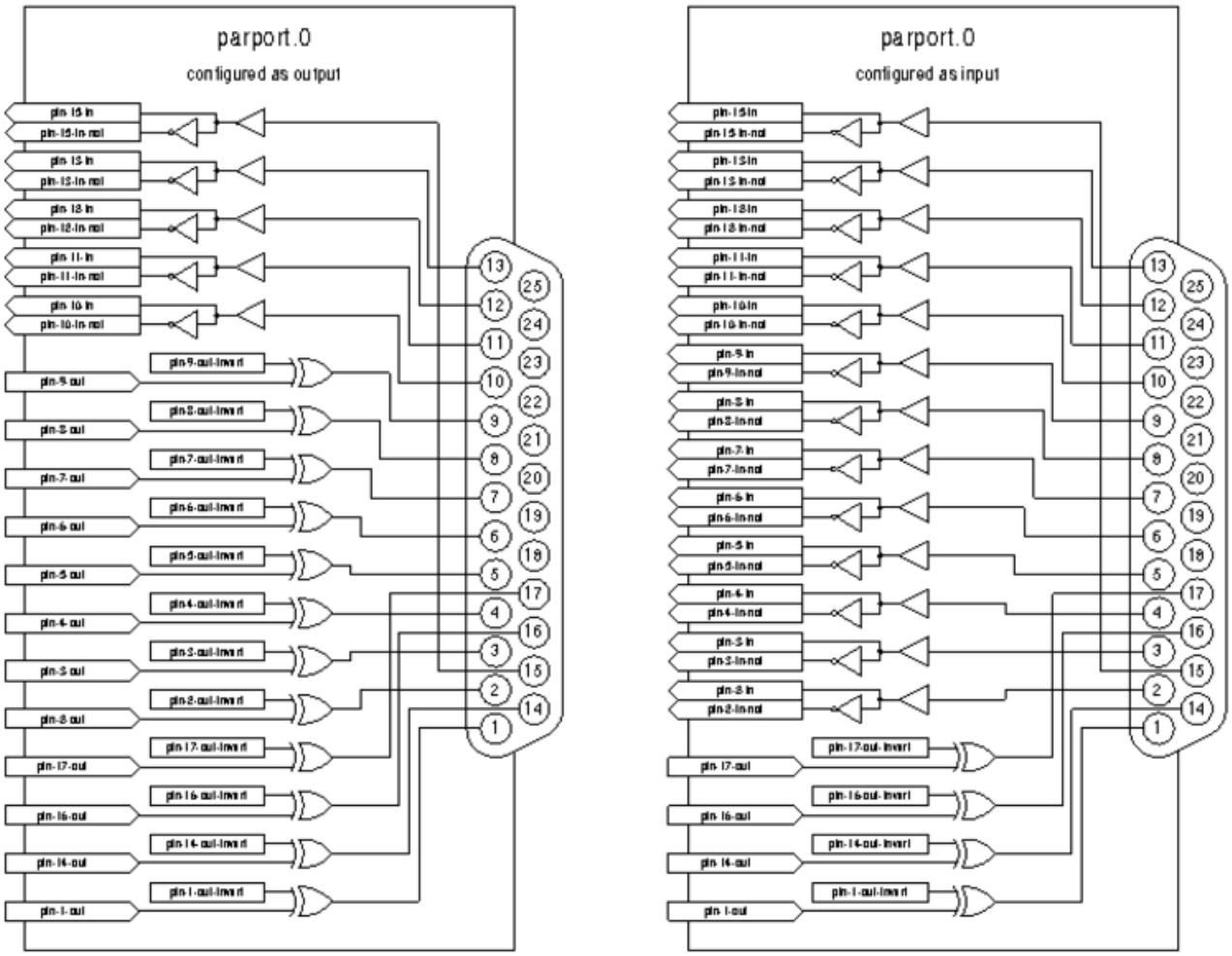



Abbildung 6.1: Parport-Blockdiagramm

**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

One good PCI parport card is made with the Netmos 9815 chipset. It has good +5 V signals, and can come in a single or dual ports.

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]
```

From experimentation, I’ve found the first port (the on-card port) uses the third address listed (c000), and the second port (the one that attaches with a ribbon cable) uses the first address listed (b800). The following example shows the onboard parallel port and a PCI parallel port using the default out direction.

```
loadrt hal_parport cfg="0x378 0xc000"
```

Please note that your values will differ. The Netmos cards are Plug-N-Play, and might change their settings depending on which slot you put them into, so if you like to *get under the hood* and re-arrange things, be sure to check these values before you start LinuxCNC.

### 6.1.3 Pins

- parport.<p>.pin-__<n>__-out (bit) Drives a physical output pin.
- parport.<p>.pin-__<n>__-in (bit) Tracks a physical input pin.
- parport.<p>.pin-__<n>__-in-not (bit) Tracks a physical input pin, but inverted.

For each pin, <*p*> is the port number, and <*n*> is the physical pin number in the 25 pin D-shell connector.

For each physical output pin, the driver creates a single HAL pin, for example: `parport.0.pin-14-out`.

For each physical input pin, the driver creates two HAL pins, for example: `parport.0.pin-12-in` and `parport.0.pin-12-in-not`.

The `-in` HAL pin is TRUE if the physical pin is high, and FALSE if the physical pin is low. The `-in-not` HAL pin is inverted and is FALSE if the physical pin is high.

### 6.1.4 Parameter

- parport.__<p>__.pin-__<n>__-out-invert (bit) Inverts an output pin.
- parport.__<p>__.pin-__<n>__-out-reset (bit) (only for `-out` pins) TRUE if this pin should be reset when the `-reset` function is executed.
- parport.__<p>__.reset-time (U32) The time (in nanoseconds) between a pin is set by `-write` and reset by the `-reset` function if it is enabled.

The `-invert` parameter determines whether an output pin is active high or active low. If `-invert` is FALSE, setting the HAL `-out` pin TRUE drives the physical pin high, and FALSE drives it low. If `-invert` is TRUE, then setting the HAL `-out` pin TRUE will drive the physical pin low.

### 6.1.5 Funktionen

- parport.__<p>__.read (funct) Reads physical input pins of port number <*p*> and updates HAL `-in` and `-in-not` pins.
- `parport.read-all` (funct) Reads physical input pins of all ports and updates HAL `-in` and `-in-not` pins.
- parport.__<p>__.write (funct) Reads HAL `-out` pins of port number <*p*> and updates that port's physical output pins.
- `parport.write-all` (funct) Reads HAL `-out` pins of all ports and updates all physical output pins.
- parport.__<p>__.reset (funct) Waits until `reset-time` has elapsed since the associated `write`, then resets pins to values indicated by `-out-invert` and `-out-invert` settings. `reset` must be later in the same thread as `write`. If `-reset` is TRUE, then the `reset` function will set the pin to the value of `-out-invert`. This can be used in conjunction with stepgen's *doublefreq* to produce one step per period. The stepgen stepspace for that pin must be set to 0 to enable doublefreq.

The individual functions are provided for situations where one port needs to be updated in a very fast thread, but other ports can be updated in a slower thread to save CPU time. It is probably not a good idea to use both an `-all` function and an individual function at the same time.

### 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

To setup DoubleStep on the parallel port you must add the function parport.n.reset after parport.n.write and configure stepspace to 0 and the reset time wanted. So that step can be asserted on every period in HAL and then toggled off by parport after being asserted for time specified by `parport. `__n__.reset-time`.

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
```

More information on DoubleStep can be found on the wiki.

### 6.1.8 probe_parport

In today's PCs, parallel ports may require a plug and play (PNP) configuration before they can be used. The kernel module *probe_parport* configures all PNP ports present. It must be loaded before *hal_parport*. On machines without a PNP port, it can be loaded but will have no effect.

#### 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 erkannt.
```

Dann wird die Verwendung dieses Moduls wahrscheinlich notwendig sein.

Schließlich sollten die HAL-Parport-Komponenten geladen werden:

```
loadrt probe_parport
loadrt hal_parport ...
```

## 6.2 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.

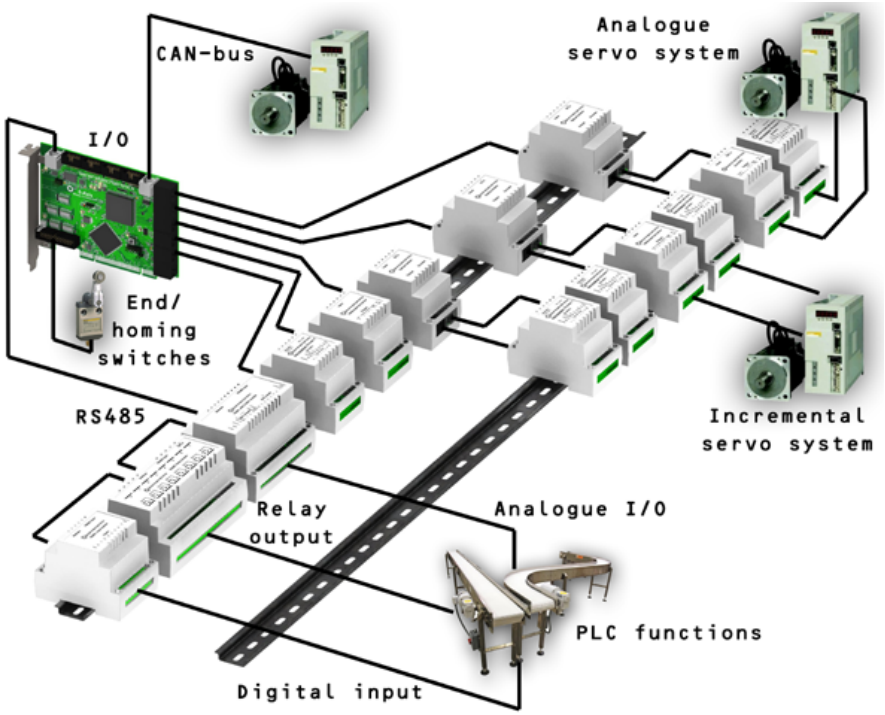
## 6.3 General Mechatronics Treiber

General Mechatronics GM6-PCI-Kartenbasiertes Bewegungssteuerungssystem (engl. Motion-Control-System)

For detailed description, please refer to the System integration manual.

Die GM6-PCI-Bewegungssteuerungskarte basiert auf einem FPGA und einem PCI-Bridge-Interface-ASIC. Eine kleine automatisierte Fertigungszelle kann mit einem kurzen Systemintegrationsverfahren gesteuert werden. Die folgende Abbildung zeigt den typischen Anschluss von Geräten im Zusammenhang mit dem Steuerungssystem:

- Er kann bis zu sechs Achsen steuern, jede davon kann eine Schrittmotor- oder CAN-Bus-Schnittstelle oder ein analoger Servo sein.
- GPIO: Vier bzw. acht E/A-Pins sind auf Standard-Flachkabelsteckern untergebracht.
- RS485 E/A-Erweiterungsmodule: Der RS485-Bus wurde für den Anschluss von kompakten Erweiterungsmodulen für die DIN-Schienenmontage entwickelt. Ein 8-Kanal-Digitaleingang, ein 8-Kanal-Relaisausgang und ein analoges I/O-Modul (4x +/-10 Volt Ausgang und 8x +/-5 Volt Eingang) sind jetzt verfügbar. Insgesamt können bis zu 16 Module an den Bus angeschlossen werden.
- 20 optisch isolierte Eingangspins: Sechs mal drei für den direkten Anschluss von zwei Endschaltern und einem Referenzierungssensor für jedes Gelenk. Und zusätzlich zwei optisch isolierte Notaus-Eingänge.




Installation:

```
loadrt hal_gm
```

Während des Ladens (oder versuchten Ladens) gibt der Treiber einige nützliche Debugging-Meldungen in das Kernel-Protokoll aus, die mit dmesg eingesehen werden können.

Es können bis zu 3 Karten in einem System verwendet werden.

Die folgenden Anschlüsse befinden sich auf der GM6-PCI-Karte:

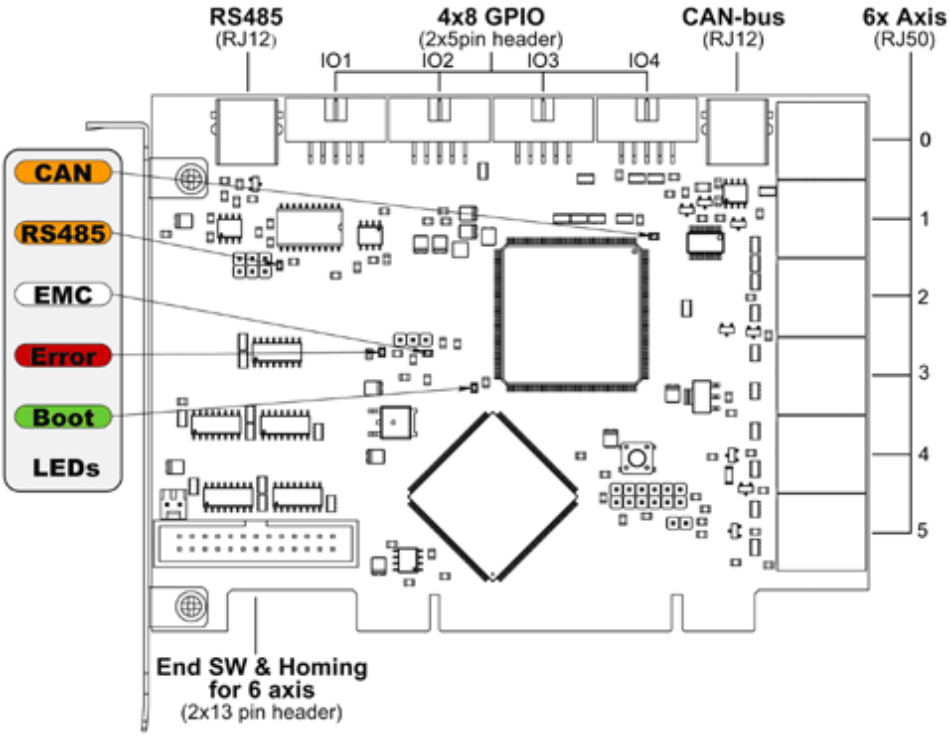



Abbildung 6.2: GM6-PCI-Kartenanschlüsse und LEDs

### 6.3.1 I/O-Anschlüsse

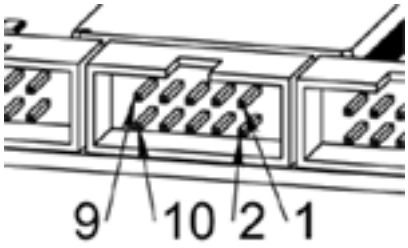



Abbildung 6.3: Pin-Nummerierung der GPIO-Anschlüsse

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>
```

wobei <*gpio_con_no*> zwischen 0 und 3 liegt.

**Zustand des ersten Pins des ersten GPIO-Anschlusses auf der GM6-PCI-Karte an.**

```
gm.0.gpio.0.in-0
```

HAL Pins werden aktualisiert durch Funktion

```
gm.<card_no>.read
```

#### 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

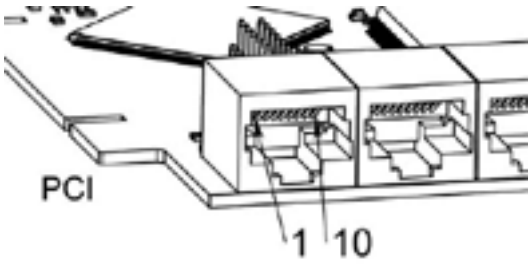



Abbildung 6.4: Pin-Nummerierung der Achsenverbinder

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

Small sized DIN rail mounted interface modules gives easy way of connecting different types of servo modules to the axis connectors. Seven different system configurations are presented in the System integration manual for evaluating typical applications. Also the detailed description of the Axis modules can be found in the System integration manual.

Zur Ermittlung der geeigneten Servoantriebsstruktur sind die Module wie im folgenden Blockschaltbild dargestellt zu verbinden:

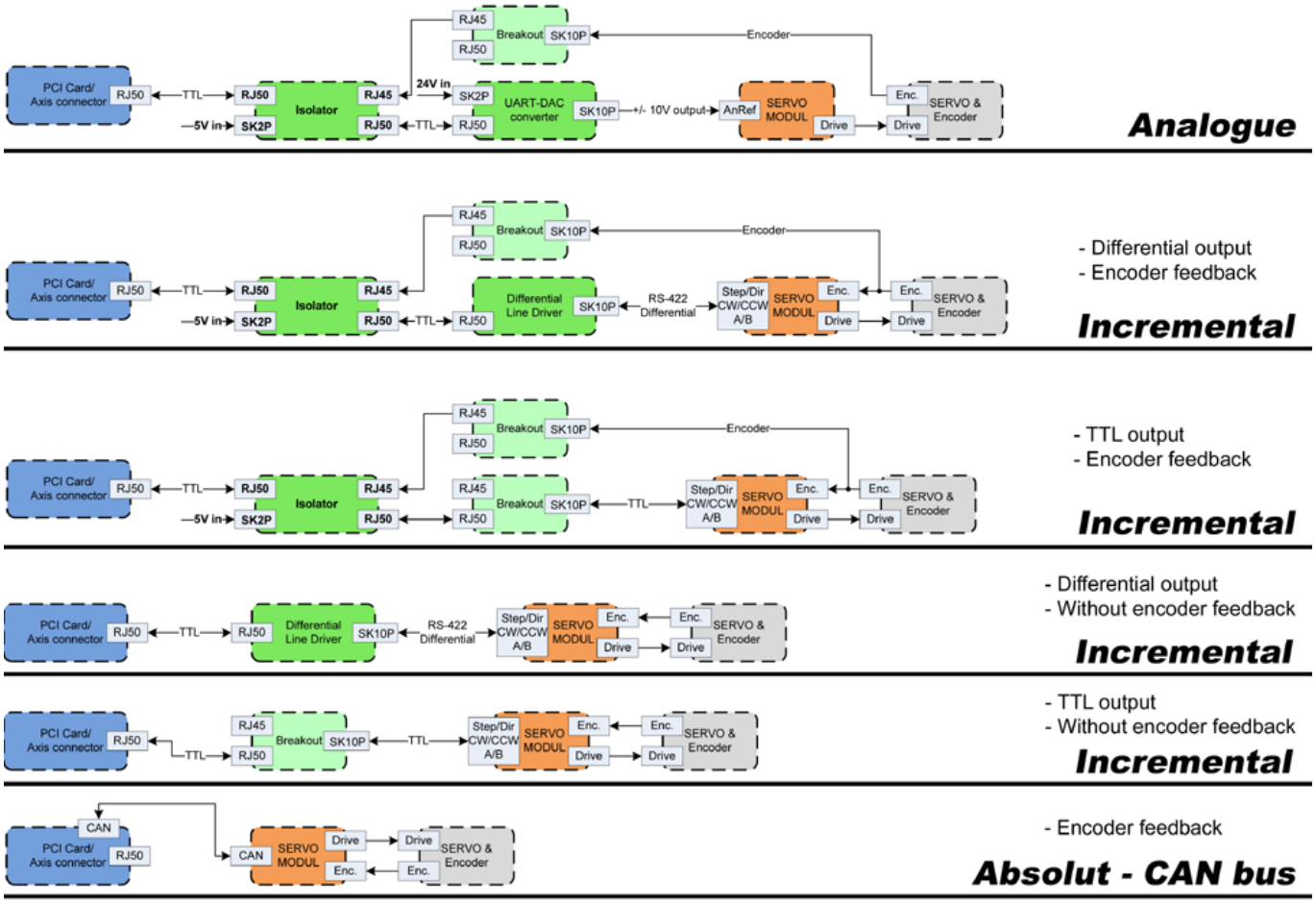



Abbildung 6.5: Servo-Achsen-Schnittstellen

#### 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>
```

wobei <*axis_no*> zwischen 0 und 5 liegt. Beispielsweise bezieht sich „gm.0.encoder.0.position“ auf die Position des Encodermoduls von Achse 0.

Die GM6-PCI-Karte zählt das Gebersignal unabhängig von LinuxCNC. HAL-Pins werden nach Funktion aktualisiert:

```
gm.<card_no>.read
```

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) | Wenn True, werden Zählerstände und Position bei der nächsten steigenden Flanke von Kanal-I gerundet oder zurückgesetzt (abhängig vom Index-Modus). Jedes Mal, wenn die Position aufgrund von Index zurückgesetzt wird, dann wird der `index-enabled`-Pin auf 0 gesetzt und bleibt 0, bis der angeschlossene HAL-Pin ihn nicht mehr setzt. |
| `.velocity` | (float, Out) | Geschwindigkeit in skalierten Einheiten pro Sekunde. Der GM-Encoder verwendet einen Hochfrequenz-Hardware-Timer zur Messung der Zeit zwischen den Encoderimpulsen, um die Geschwindigkeit zu berechnen. Dadurch wird das Quantisierungsrauschen im Vergleich zur einfachen Differenzierung des Positionsausgangs erheblich reduziert. Wenn die gemessene Geschwindigkeit unter der geschätzten Mindestgeschwindigkeit liegt, ist der Geschwindigkeitsausgang 0. |

Tabelle 6.7: Encoder-Parameter

| **Parameter** | **Typ und Lesen/-Schreiben** | **Parameterbeschreibung** |
|---|---|---|
| `.counter-mode` | (bit, R/W) | Bei True zählt der Zähler jede ansteigende Flanke des Kanal-A-Eingangs in die durch Kanal-B bestimmte Richtung. Dies ist nützlich für die Zählung des Ausgangs eines Einkanal- (Nicht-Quadratur-) oder Schritt-/Differenzsignalsensors. Wenn False, zählt er im Quadraturmodus. |

Tabelle 6.7: (continued)

| **Parameter** | **Typ und Lesen/-Schreiben** | **Parameterbeschreibung** |
|---|---|---|
| `.index-mode` | (bit, R/W) | Wenn True und .index-enabled ebenfalls True ist, werden .counts und .position gerundet (basierend auf .counts-per-rev) bei der steigenden Flanke von Kanal-I. Dies ist nützlich, um durch Rauschen verursachte Fehler bei wenigen Impulsen zu korrigieren. Im Rundungsmodus ist es wichtig, dass der Parameter .counts-per-rev korrekt eingestellt ist. Wenn .index-mode auf False und .index-enabled auf True steht, werden .counts und .position beim Kanal-I-Impuls zurückgesetzt. |
| `.counts-per-rev` | (s32, R/V) | Bestimmt, wie viele Zählungen zwischen zwei Indeximpulsen liegen. Er wird nur im runden Modus verwendet, d.h. wenn die Parameter .index-enabled und .index-mode beide auf True stehen. GM-Encoder verarbeiten das Encodersignal im 4x-Modus, d.h. im Falle eines 500 CPR Encoders sollte er auf 2000 gesetzt werden. Dieser Parameter kann leicht gemessen werden, indem man .index-enabled True und .index-mode False setzt (so dass .counts bei Kanal-I-Impuls zurücksetzt), dann die Achse von Hand bewegen und die maximale Größe des .counts-Pins im Halmeter sehen. |
| `.index-invert` | (bit, R/W) | Wenn True, tritt das Kanal-I-Ereignis (Reset oder Round) bei der fallenden Flanke des Kanal-I-Signals ein, andernfalls bei der steigenden Flanke. |
| `.min-speed-estimate` | (float, R/W) | Legen Sie den minimalen gemessenen Geschwindigkeitswert fest, bei dem .velocity als ungleich Null gesetzt wird. Wird dieser Parameter zu niedrig eingestellt, dauert es sehr lange, bis die Geschwindigkeit auf Null geht, nachdem keine Encoderimpulse mehr ankommen. |
| `.position-scale` | (float, R/W) | Skala in Zählungen pro Längeneinheit. .position=.counts/.position-scale. Wenn z. B. der Positionsmaßstab 2000 ist, dann ergeben 1000 Zählungen des Encoders eine Position von 0,5 Einheiten. |

**Einstellung des Gebermoduls der Achse 0 für den Empfang des 500 CPR Quadraturgebersignals und Verwendung der Rückstellung zum Runden der Position.**

```
setp gm.0.encoder.0.counter-mode 0 # 0: quad, 1: stepDir
setp gm.0.encoder.0.index-mode 1 # 0: reset pos at index, 1:round pos at index
setp gm.0.encoder.0.counts-per-rev 2000 # GM-Prozess-Encoder im 4x-Modus, 4x500=2000
setp gm.0.encoder.0.index-invert 0 #
setp gm.0.encoder.0.min-speed-estimate 0.1 # in Positionseinheit/s
setp gm.0.encoder.0.position-scale 20000 # 10 Geberumdrehungen bewirken, dass sich die ↩
    Maschine um eine Positionseinheit bewegt (10x2000)
```

**Verbinden der Encoder Position mit LinuxCNC Gelenk-Position Feedback**

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

#### 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>
```

wobei <*axis_no*> zwischen 0 und 5 liegt. Zum Beispiel bezieht sich *gm.0.stepgen.0.position-cmd* auf den Positionsbefehl des StepGen-Moduls der Achse 0 auf Karte 0.

Die GM6-PCI-Karte erzeugt unabhängig von LinuxCNC Schrittimpulse. Die HAL-Pins werden geupdatet durch die Funktion

```
gm.<card_no>.write
```

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) | Wenn True, wird .velocity-cmd als Referenz verwendet und velocityvcontrol berechnet die Pulsfrequenzausgabe. Bei False wird .position-cmd als Referenz verwendet und die Lageregelung berechnet den Impulsratenausgang. |
| `.invert-step1` | (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) | Maximale Geschwindigkeit in Positionseinheiten pro Sekunde. Wenn er auf 0,0 gesetzt ist, wird der Parameter .maxvel ignoriert. |
| `.maxaccel` | (float, R/W) | Maximale Beschleunigung in Positionseinheiten pro Sekunde zum Quadrat. Wenn er auf 0,0 gesetzt ist, wird der Parameter .maxaccel ignoriert. |
| `.position-scale` | (float, R/W) | Skala in Schritten pro Längeneinheit. |
| `.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) | Mindestzeit zwischen Schrittimpuls und Richtungswechsel in Nanosekunden. |

Zur Ermittlung der entsprechenden Werte siehe die nachstehenden Zeitdiagramme:

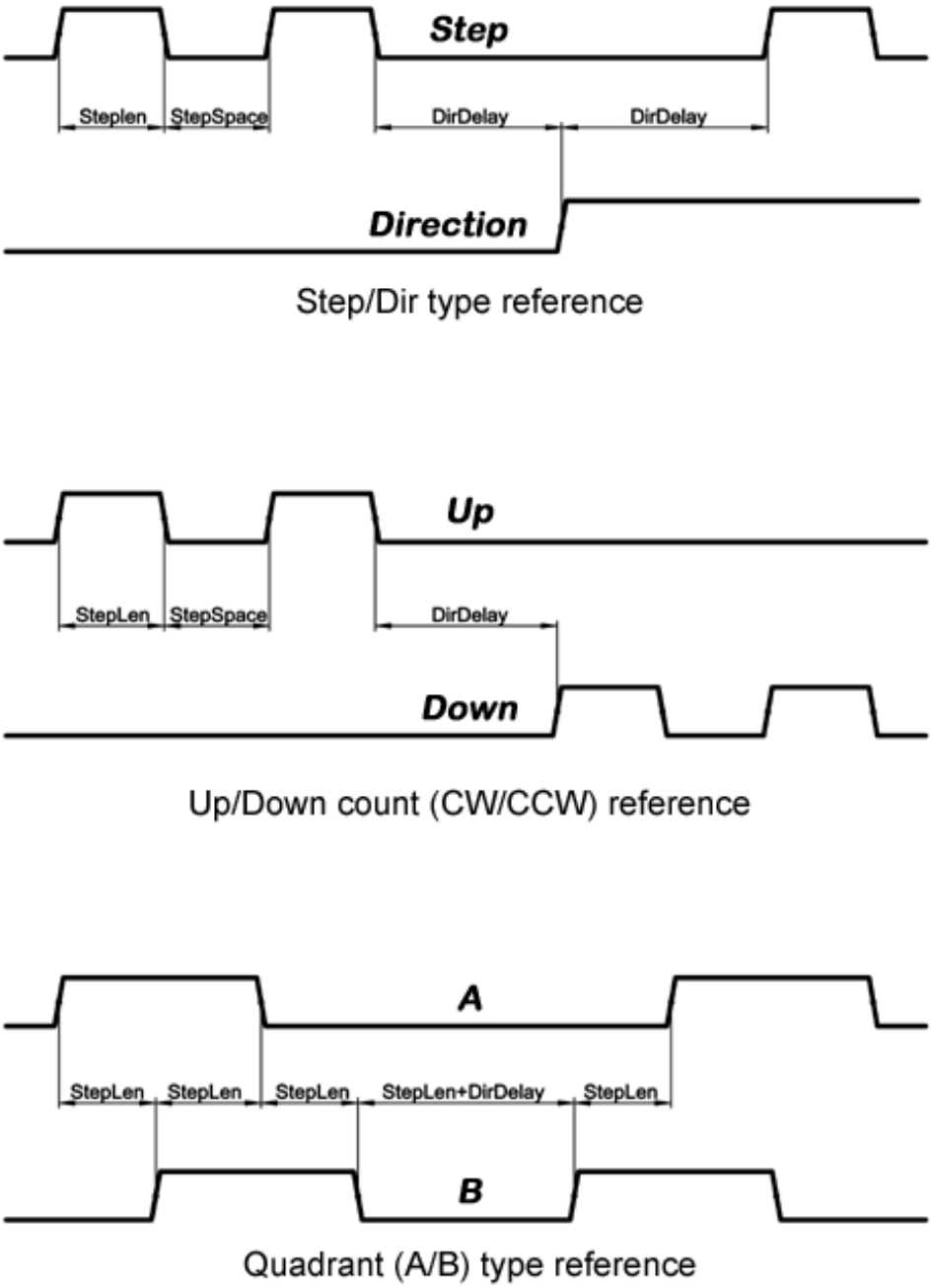



Abbildung 6.6: Referenzsignal-Zeitdiagramme

**Einstellung des StepGen-Moduls der Achse 0 zur Erzeugung von 1000 Schrittimpulsen pro Positionseinheit**

```
setp gm.0.stepgen.0.step-type 0         # 0:stepDir, 1:UpDown, 2:Quad
setp gm.0.stepgen.0.control-type 0      # 0:Pos. 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
```

**Verbinden Sie StepGen mit den Positionsreferenz- und Freigabepins der Achse 0**

```
net Xpos-cmd joint.0.motor-pos-cmd => gm.0.stepgen.0.position-cmd
net Xen joint.0.amp-enable-out => gm.0.stepgen.0.enable
```

#### 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) | Wenn dieser Pin True ist,<br>* und der Watch Dog Timer nicht abgelaufen ist<br>* und es liegt kein Stromfehler vor<br>dann werden die Power-Enable-Pins der Achsen- und CAN-Anschlüsse auf High gesetzt, ansonsten auf 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>
```

wobei <*axis_no*> zwischen 0 und 5 liegt. Zum Beispiel bezieht sich *gm.0.dac.0.value* auf die Ausgangsspannung des DAC-Moduls der Achse 0.

HAL Pins werden durch Funktion aktualisiert:

```
gm.<card_no>.write
```

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) | Maximale Ausgangsspannung der Hardware in Volt. |
| .Untergrenze | (float, R/W) | Minimale Ausgangsspannung der Hardware in Volt. |
| .invert-seriell | (float, R/W) | Die GM6-PCI-Karte kommuniziert mit der DAC-Hardware über schnelle serielle Kommunikation, um die Zeitverzögerung im Vergleich zur PWM stark zu reduzieren. Es wird empfohlen, das DAC-Modul zu isolieren, wodurch die serielle Kommunikationsleitung negiert wird. Im Falle einer Isolierung lassen Sie diesen Parameter auf dem Standardwert (0), während Sie diesen Parameter im Falle einer Nicht-Isolierung auf 1 setzen. |

### 6.3.3 CAN-Bus-Servoverstärker

Die GM6-PCI Motion Control Karte verfügt über ein CAN-Modul zur Ansteuerung von CAN-Servoverstärkern Die Implementierung von Protokollen auf höherer Ebene wie CANopen ist für eine zukünftige Entwicklung. Derzeit haben die von GM hergestellten Leistungsverstärker Treiber der oberen Ebene, die Pins und Parameter nach HAL exportieren. Sie empfangen Positionsreferenzen und liefern Encoder-Feedback über den CAN-Bus.

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>
```

wobei < *axis_no* > zwischen 0 und 5 liegt. Zum Beispiel bezieht sich `gm.0.can-gm.0.position` auf die Ausgangsposition der Achse 0 in Positionseinheiten.

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) | Aktiviert den Watchdog-Timer.<br>Es wird dringend empfohlen, den Watchdog-Timer zu aktivieren, weil er im Falle eines PC-Fehlers alle Servoverstärker durch Herunterziehen aller Freigabesignale deaktivieren kann. |
| gm.<*card_no*>.watchdog-timeout | (float, R/W) | Zeitintervall, in dem die Funktion gm.<*card_no*>.read ausgeführt werden muss. Die gm.<*card_no*>.read wird typischerweise dem Servo-Thread hinzugefügt, daher wird der Watch-Timeout typischerweise auf das 3-fache der Servoperiode gesetzt. |

### 6.3.5 End-, Referenzpunkt- und Notaus-Schalter

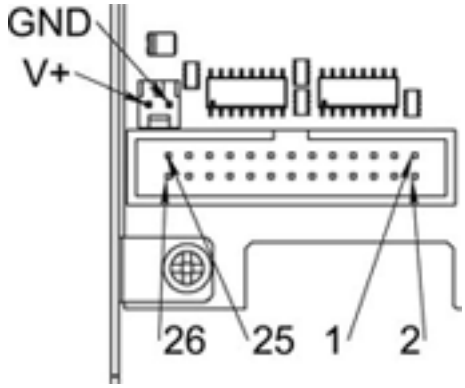



Abbildung 6.7: Pin-Nummerierung des Anschlusses für Referenzfahrt und Endschalter

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/Hom-ing | 3/End- | 4/End+ | 4/Hom-ing | 5/End- | 6/End+ | 6/Hom-ing | Notaus 2 | V+ (Ext.) |

| 26 | 24 | 22 | 20 | 18 | 16 | 14 | 12 | 10 | 8 | 6 | 4 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GND | | 1/End+ | 1/Hom-ing | 2/End- | 3/End+ | 3/Hom-ing | 4/End- | 5/End+ | 5/Hom-ing | 6/End- | Notaus 1 | V+ (Ext.) |

Die GM6-PCI-Bewegungssteuerungskarte hat zwei Endschalter- und einen Referenzschaltereingang für jedes Gelenk. Die Namen dieser Pins beginnen wie folgt:

```
gm.<card_no>.joint.<axis_no>
```

wobei *<axis_no>* zwischen 0 bis 5 liegt. Beispiel: ”gm.0.joint.0.home-sw-in” gibt den Zustand des Home-Schalters der Achse 0 an.

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.

**Automatische Knotenerkennung** Jeder an den Bus angeschlossene Knoten wurde von der GM6-PCI-Karte automatisch erkannt. Beim Start von LinuxCNC exportiert der Treiber automatisch Pins und Parameter aller verfügbaren Module.

**Fehlerbehandlung** Wenn ein Modul nicht regelmäßig antwortet, fährt die GM6-PCI-Karte das Modul herunter. Wenn ein Modul mit Ausgang nicht regelmäßig Daten mit korrektem CRC erhält, schaltet das Modul in den Fehlerzustand (grüne LED blinkt), und schaltet alle Ausgänge in den Fehlerzustand.

**Verbinden der Knoten** Die Module auf dem Bus müssen in serieller Topologie angeschlossen werden, mit Abschlusswiderständen am Ende. Der Anfang der Topologie ist die PCI-Karte, und das Ende ist das letzte Modul.

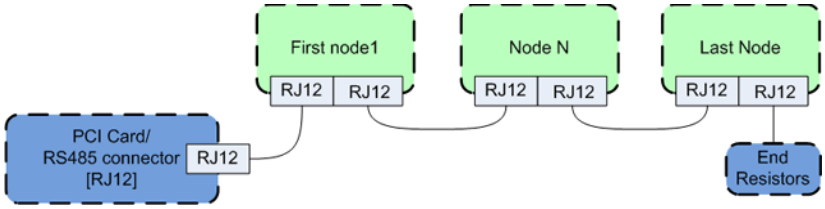



Abbildung 6.8: Anschließen der RS485-Knoten an die GM6-PCI-Karte

**Adressierung** Jeder Knoten am Bus hat eine eindeutige 4-Bit-Adresse, die mit einem roten DIP-Schalter eingestellt werden kann.

**Status-LED** Eine grüne LED zeigt den Status des Moduls an:

- Blinkt, wenn das Modul nur mit Strom versorgt, aber noch nicht erkannt wird, oder wenn das Modul fallengelassen wird.
- Aus, während der Identifizierung (Computer ist eingeschaltet, aber LinuxCNC nicht gestartet)
- Ein, wenn es kontinuierlich kommuniziert.

#### 6.3.7.1 Relais-Ausgangsmodul

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

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.

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 # Erstes Relais des Knotens.
#gm.0                    # Bedeutet die erste GM6-PCI-Bewegungssteuerungskarte (PCI- ↩
    Kartenadresse = 0)
#.      .rs485.0            # Auswahl des Knotens mit der Adresse 0 auf dem RS485-Bus
#.              .relay-0    # Auswahl des ersten Relais
```

### 6.3.7.2 Digitales Eingangsmodul

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

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.

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 # Erster Eingang des Knotens.
# gm.0                  # Bedeutet die erste GM6-PCI-Bewegungssteuerungskarte (PCI- ↩
    Kartenadresse = 0)
```

```
#      .rs485.0         # Auswahl des Knotens mit der Adresse 0 auf dem RS485-Bus
#              .in-0    # Wählt das erste digitale Eingangsmodul
```

#### 6.3.7.3 DAC & ADC-Modul

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

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.

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) | Die Eingangsspannung wird mit der Skalierung multipliziert, bevor sie an den .adc-Pin ausgegeben wird. |
| .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) | Maximale Ausgangsspannung der Hardware in Volt. |
| .dac-low-limit-<*0-3*> | (float, R/W) | Minimale Ausgangsspannung der Hardware in Volt. |

**HAL-Beispiel**

```
gm.0.rs485.0.adc-0 # Erster Analogkanal des Knotens.
# gm.0                  # Bedeutet die erste GM6-PCI-Bewegungssteuerungskarte (PCI- ↩
    Kartenadresse = 0)
```

```
#      .rs485.0          # Auswahl des Knotens mit der Adresse 0 auf dem RS485-Bus
#              .adc-0    # Wählt den ersten Analogeingang des Moduls
```

#### 6.3.7.4 Teach Pendant Modul

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

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 # Erster Analogkanal des Knotens.
# gm.0                 # Bedeutet die erste GM6-PCI-Bewegungssteuerungskarte (PCI- ↩
    Kartenadresse = 0)
#     .rs485.0         # Auswahl des Knotens mit der Adresse 0 auf dem RS485-Bus
#             .adc-0  # Wählt den ersten Analogeingang des Moduls
```

### 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 Driver

This is a non-realtime HAL program for the GS2 series of VFDs at Automation Direct. [1]

Diese Komponente wird mit dem halcmd-Befehl ”loadusr” geladen:

```
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd
```

Der obige Befehl lautet: loadusr, wait for named to load, component gs2_vfd, named spindle-vfd. Der HAL-Befehl ”loadusr” ist im Kapitel loadusr beschrieben.

### 6.4.1 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* or *--debug* Turn on debugging messages. This will also set the verbose flag. Debug mode will cause all modbus messages to be printed in hex on the terminal.
- *-n* or *--name <string>* (default: gs2_vfd) Set the name of the HAL module. The HAL comp name will be set to <string>, and all pin and parameter names will begin with <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* or *--rate* <n> (default: 38400) Set baud rate to *n*. It is an error if the rate is not one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

[1] In Europe the equivalent can be found under the brand name Omron.

- *-s* oder *--stopbits {1,2}* (Voreinstellung: 1) Setzt die Anzahl von Stopbits auf 1 oder 2
- *-t* or *--target* <n> (default: 1) Set MODBUS target (slave) number. This must match the device number you set on the GS2.
- *-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* or *--decel-seconds <n>* (default: 0.0) Seconds to decelerate the spindle from max. RPM to 0. If set to 0.0 the spindle will be allowed to coast to a stop without controlled deceleration.
- *-R* or *--braking-resistor* This argument should be used when a braking resistor is installed on the GS2 VFD (see Appendix A of the GS2 manual). It disables deceleration over-voltage stall prevention (see GS2 modbus Parameter 6.05), allowing the VFD to keep braking even in situations where the motor is regenerating high voltage. The regenerated voltage gets safely dumped into the braking resistor.

---

**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**
The status value is a sum of all the bits that are on. So a 163 which means the drive is in the run mode is the sum of 3 (run) + 32 (freq set by serial) + 128 (operation set by serial).

### 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 HAL Driver for Raspberry Pi GPIO pins

Note: This driver will not be compiled into images aimed at non-ARM CPUS. It is only really intended to work on the Raspberry Pi. It may, or may not, work on similar boards or direct clones.

### 6.5.1 Purpose

This driver allows the use of the Rapberry Pi GPIO pins in a way analagous to the parallel port driver on x86 PCs. It can use the same step generators, encoder counters and similar components.

### 6.5.2 Anwendung

```
loadrt hal_pi_gpio dir=0x13407 exclude=0x1F64BF8
```

The ”dir” mask determines whether the pins are inputs and outputs, the exclude mask prevents the driver from using the pins (and so allows them to be used for their normal RPi purposes such as SPI or UART).

The mask can be in decimal or hexadecimal (hex may be easier as there will be no carries).

To determine the value of the masks, add up the hex/decimal values for all pins that should be configured as output, and analogously for all pins that should be excluded according to the following table.

Tabelle 6.27: GPIO masks - mapping of GPIO numbers (leftmost column) to physical pin numbers as printed on the Raspberry Pi board (rightmost column) and the decimal/hexadecimal values that contribute to the value of the mask.

| GPIO Num | Decimal | Hex | Pin Num |
|---|---|---|---|
| 2 | 1 | 0x00000001 | 3 |
| 3 | 2 | 0x00000002 | 5 |
| 4 | 4 | 0x00000004 | 7 |
| 5 | 8 | 0x00000008 | 29 |
| 6 | 16 | 0x00000010 | 31 |
| 7 | 32 | 0x00000020 | 26 |
| 8 | 64 | 0x00000040 | 24 |
| 9 | 128 | 0x00000080 | 21 |
| 10 | 256 | 0x00000100 | 19 |
| 11 | 512 | 0x00000200 | 23 |
| 12 | 1024 | 0x00000400 | 32 |
| 13 | 2048 | 0x00000800 | 33 |
| 14 | 4096 | 0x00001000 | 8 |
| 15 | 8192 | 0x00002000 | 10 |
| 16 | 16384 | 0x00004000 | 36 |
| 17 | 32768 | 0x00008000 | 11 |
| 18 | 65536 | 0x00010000 | 12 |
| 19 | 131072 | 0x00020000 | 35 |
| 20 | 262144 | 0x00040000 | 38 |
| 21 | 524288 | 0x00080000 | 40 |
| 22 | 1048576 | 0x00100000 | 15 |
| 23 | 2097152 | 0x00200000 | 16 |
| 24 | 4194304 | 0x00400000 | 18 |
| 25 | 8388608 | 0x00800000 | 22 |
| 26 | 16777216 | 0x01000000 | 37 |
| 27 | 33554432 | 0x02000000 | 13 |

Note: In the calculation of the individual pin's mask value its GPIO numbers are used, the value being derived as 2^(*GPIO number* - 2), whereas in the naming of the HAL pins it is the Raspberry Pi header pin numbers.

So, for example, if you enable GPIO 17 as an output (dir=0x8000) then that output will be controlled by the hal pin **hal_pi_gpio.pin-11-out**.

### 6.5.3 Pins

- hal_pi_gpio.pin-*NN*-out
- hal_pi_gpio.pin-*NN*-in

Depending on the dir and exclude masks.

### 6.5.4 Parameter

Only the standard timing parameters which are created for all components exist.

*hal_pi_gpio.read.tmax *hal_pi_gpio.read.tmax-increased *hal_pi_gpio.write.tmax *hal_pi_gpio.write.tmax-increased

For unknown reasons the driver also creates HAL *pins* to indicate timing

*hal_pi_gpio.read.time *hal_pi_gpio.write.time

### 6.5.5 Funktionen

- *hal_pi_gpio.read* - Add this to the base thread to update the HAL pin values to match the physical input values.
- *hal_pi_gpio.write* - Add this to the base thread to update the physical pins to match the HAL values.

Typically the *read* function will be early in the call list, before any encoder counters and the *write* function will be later in the call list, after stepgen.make-pulses.

### 6.5.6 Pin-Nummerierung

The GPIO connector and the pinout has been consistent since around 2015. These older Pi models are probably a poor choice for LinuxCNC anyway. However, this driver is designed to work with them, and will detect and correctly configure for the two alternative pinouts.

The current pinout mapping between GPIO numbers and connector pin numbers is included in the table above.

Note that the config string uses GPIO numbers, but once the driver is loaded the HAL pin names refer to connector pin numbers.

This may be more logical than it first appears. When setting up you need to configure enough pins of each type, whilst avoiding overwriting any other functions that your system needs. Then once the driver is loaded, in the HAL layer you just want to know where to connect the wires for each HAL pin.

### 6.5.7 Known Bugs

At the moment (2023-07-16) this driver only seems to work on Raspbian as the generic Debian image does not set up the correct interfaces in /dev/gpiomem and restricts access to the /sys/mem interface.

## 6.6 Generic driver for any GPIO supported by gpiod.

This driver has been tested on the Raspberry Pi, and should also work on Banana Pi, BeagleBone, Pine64 (et al.) and other single board computers, and potentially on other platforms.

### 6.6.1 Purpose

This driver allows the use of GPIO pins in a way analogous to the parallel port driver on x86 PCs. It can use the same step generators, encoder counters and similar components.

### 6.6.2 Anwendung

```
loadrt hal_gpio inputs=GPIO5,GPIO6,GPIO12,GPIO13,GPIO16,GPIO17,GPIO18,GPIO19 \
                                 outputs=GPIO20,GPIO21,GPIO22,GPIO23,GPIO24,GPIO25,GPIO26, ↩
                                     GPIO27 \
                                 invert=GPIO20,GPIO27 \
                                 reset=GPIO21,GPIO22
```

This driver relies on the libgpiod-dev library and the gpiod package, which contains a number of utilities for configuring and querying GPIO. The GPIO pin names in the ”loadrt” line of the HAL given above should be the names given by the `gpioinfo` command.

Sample output (truncated):

```
$ gpioinfo
gpiochip0 - 54 lines:
        line   0:     ”ID_SDA”       unused   input  active-high
        line   1:     ”ID_SCL”       unused   input  active-high
        line   2:       ”SDA1”       unused   input  active-high
        line   3:       ”SCL1”       unused   input  active-high
        line   4:  ”GPIO_GCLK”       unused   input  active-high
        line   5:      ”GPIO5”       unused   input  active-high
        line   6:      ”GPIO6”       unused   input  active-high
        line   7:  ”SPI_CE1_N”       unused   input  active-high
        line   8:  ”SPI_CE0_N”       unused   input  active-high
        line   9:   ”SPI_MISO”       unused   input  active-high
        line  10:   ”SPI_MOSI”       unused   input  active-high
        line  11:   ”SPI_SCLK”       unused   input  active-high
        line  12:     ”GPIO12”       unused   input  active-high
        line  13:     ”GPIO13”       unused   input  active-high
        line  14:       ”TXD1”       unused   input  active-high
        line  15:       ”RXD1”       unused   input  active-high
        line  16:     ”GPIO16”       unused   input  active-high
        line  17:     ”GPIO17”       unused   input  active-high
        line  18:     ”GPIO18”       unused   input  active-high
        line  19:     ”GPIO19”       unused   input  active-high
        line  20:     ”GPIO20”       unused  output  active-high
...
```

A list of input and/or output pins should be specified as shown in the sample above. The \ character is used for line continuation in HAL, and is used to improve readability. The pin names are case-sensitive and there must be no spaces in the strings, neither between the comma-separated pins lists nor the ”=” signs.

**Additional modifiers are**

**invert**
(valid for outputs only). Inverts the sense of the physical pin relative to the value in HAL.

**reset**
(valid for outputs only). If any pins are allocated to the ”reset” list then a HAL parameter **hal_gpio.reset_ns** will be created. This will have no effect unless the **hal_gpio.reset** function is added to a realtime thread. This should be placed after the **hal_gpio.write** function and must be in the same thread. The behaviour of this function is equivalent to the same function in the **hal_parport** driver, and it allows a step pulse every thread cycle. If the **hal_gpio.reset_ns** time is set longer than 1/4 of the period of the thread that it is added to, then the value will be reduced to 1/4 the thread period. There is a lower limit to how long the pulse can be. With 8 pins in the output list the pulse width can not reduce lower than 5000 ns on an RPi4, for example.

The following functions are accepted in all versions, but are only effective if a version of libgpiod_dev >= 1.6 is installed. They should be used in the same way as the parameters described above, and will alter the electrical parameters of the GPIO pins **if** this is supported by the hardware.

**opendrain**

**opensource**

**biasdisable**

**pulldown**

**pullup**

The version of libgpiod-dev installed can be determined by the command `gpioinfo -v`

### 6.6.3 Pins

- hal_gpio.NAME-in - HAL_OUT The value of an input pin presented in to HAL
- hal_gpio.NAME-in-not - HAL_OUT An inverted version of the above, for convenience
- hal_gpio.NAME-out - HAL_IN use this pin to transfer a HAL bit value to a physical output

### 6.6.4 Parameter

- hal_gpio.reset_ns - HAL_RW - ”setp” this parameter to control the pulse length of pins added to the ”reset” list. The value will be limited between 0 and thread-period / 4.

### 6.6.5 Funktionen

- *hal_gpio.read* - Add this to the base thread to update the HAL pin values to match the physical input values.
- *hal_gpio.write* - Add this to the base thread to update the physical pins to match the HAL values.
- *hal_gpio.reset* - Only exported if there are pins defined in the reset list. This should be placed after the ”write” function, and should be in the same thread.

Typically, the *read* function will be early in the call list, before any encoder counters and the *write* function will be later in the call list, after `stepgen.make-pulses`.

### 6.6.6 Pin Identification

Use the pin names returned by the `gpioinfo` utility. This uses the device-tree data. If the installed OS does not have a device-tree database then the pins will all be called ”unnamed” (or similar) and this driver can not be used.

A further update to this driver might allow access by index number, but this is not currently supported.

### 6.6.7 Troubleshooting permissions problems.

If ”access denied” messages are returned on loading the driver, try the following recipe: (Should not be needed for Raspbian, and will need to be modified to match the actual GPIO chip name on non-Pi platforms)

1. Create a new group `gpio` with the command

```
sudo groupadd gpio
```

2. Then to setup permissions for the ”gpio” group, create a file called `90-gpio-access` in the `/etc/udev/rules.d/` directory with the following contents (this is copied from the Raspbian install)

```
SUBSYSTEM==”bcm2835-gpiomem”, GROUP=”gpio”, MODE=”0660”
SUBSYSTEM==”gpio”, GROUP=”gpio”, MODE=”0660”
SUBSYSTEM==”gpio*”, PROGRAM=”/bin/sh -c ’\
        chown -R root:gpio /sys/class/gpio && chmod -R 770 /sys/class/gpio;\
        chown -R root:gpio /sys/devices/virtual/gpio && chmod -R 770 /sys/devices/ ↩
            virtual/gpio;\
        chown -R root:gpio /sys$devpath && chmod -R 770 /sys$devpath\
’”

SUBSYSTEM==”pwm*”, PROGRAM=”/bin/sh -c ’\
        chown -R root:gpio /sys/class/pwm && chmod -R 770 /sys/class/pwm;\
        chown -R root:gpio /sys/devices/platform/soc/*.pwm/pwm/pwmchip* && chmod -R 770 ↩
             /sys/devices/platform/soc/*.pwm/pwm/pwmchip*\
’”
```

3. Add the user who runs LinuxCNC to the `gpio` group with

```
sudo usermod -aG gpio <username>
```

### 6.6.8 Author

Andy Pugh

### 6.6.9 Known Bugs

None at this time.

## 6.7 Mesa HostMot2-Treiber

### 6.7.1 Einführung

HostMot2 is an FPGA configuration developed by Mesa Electronics for their line of *Anything I/O* motion control cards. The firmware is open source, portable and flexible. It can be configured (at compile-time) with zero or more instances (an object created at runtime) of each of several Modules: encoders (quadrature counters), PWM generators, and step/dir generators. The firmware can be configured (at run-time) to connect each of these instances to pins on the I/O headers. I/O pins not driven by a Module instance revert to general-purpose bi-directional digital I/O.

### 6.7.2 Firmware-Binärdateien

**50 Pin Header FPGA-Karten** Several pre-compiled HostMot2 firmware binaries are available for the different Anything I/O boards. This list is incomplete, check the hostmot2-firmware distribution for up-to-date firmware lists.

- 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 I/O pins): using hm2_pci module
  - 16-Kanal-Servo
  - 8-Kanal-Servo plus 24 Step/Dir-Generatoren
- 5I20, 5I23, 4I65, 4I68 (72 I/O pins): using hm2_pci module
  - 12-Kanal-Servo
  - 8-Kanal-Servo plus 4 Step/Dir-Generatoren
  - 4-Kanal-Servo plus 8 Step/Dir-Generatoren
- 7I43 (48 I/O pins): using hm2_7i43 module
  - 8-Kanal-Servo (8 PWM-Generatoren und 8 Encoder)
  - 4-Kanal-Servo plus 4 Step/Dir-Generatoren

**DB25 FPGA-Karten** The 5I25 Superport FPGA card is preprogrammed when purchased and does not need a firmware binary.

### 6.7.3 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.

### 6.7.4 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.

### 6.7.5 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:

#### 6.7.5.1 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.

#### 6.7.5.2 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.

### 6.7.6 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* - Read the GPIO input pins only. (This function is not available on the 7I43 due to limitations of the EPP bus.)
- *hm2_<BoardType>.<BoardNum>.write_gpio* - Write the GPIO control registers and output pins only. (This function is not available on the 7I43 due to limitations of the EPP bus.)

---

**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.

---

### 6.7.7 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.

An example of a 5I20 configuration:

```
[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.

### 6.7.8 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>/
```

### 6.7.9 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*.

### 6.7.10 HAL-Pins

The HAL pins for each configuration can be seen by opening up *Show HAL Configuration* from the Machine menu. All the HAL pins and parameters can be found there. The following figure is of the 5I20 configuration used above.

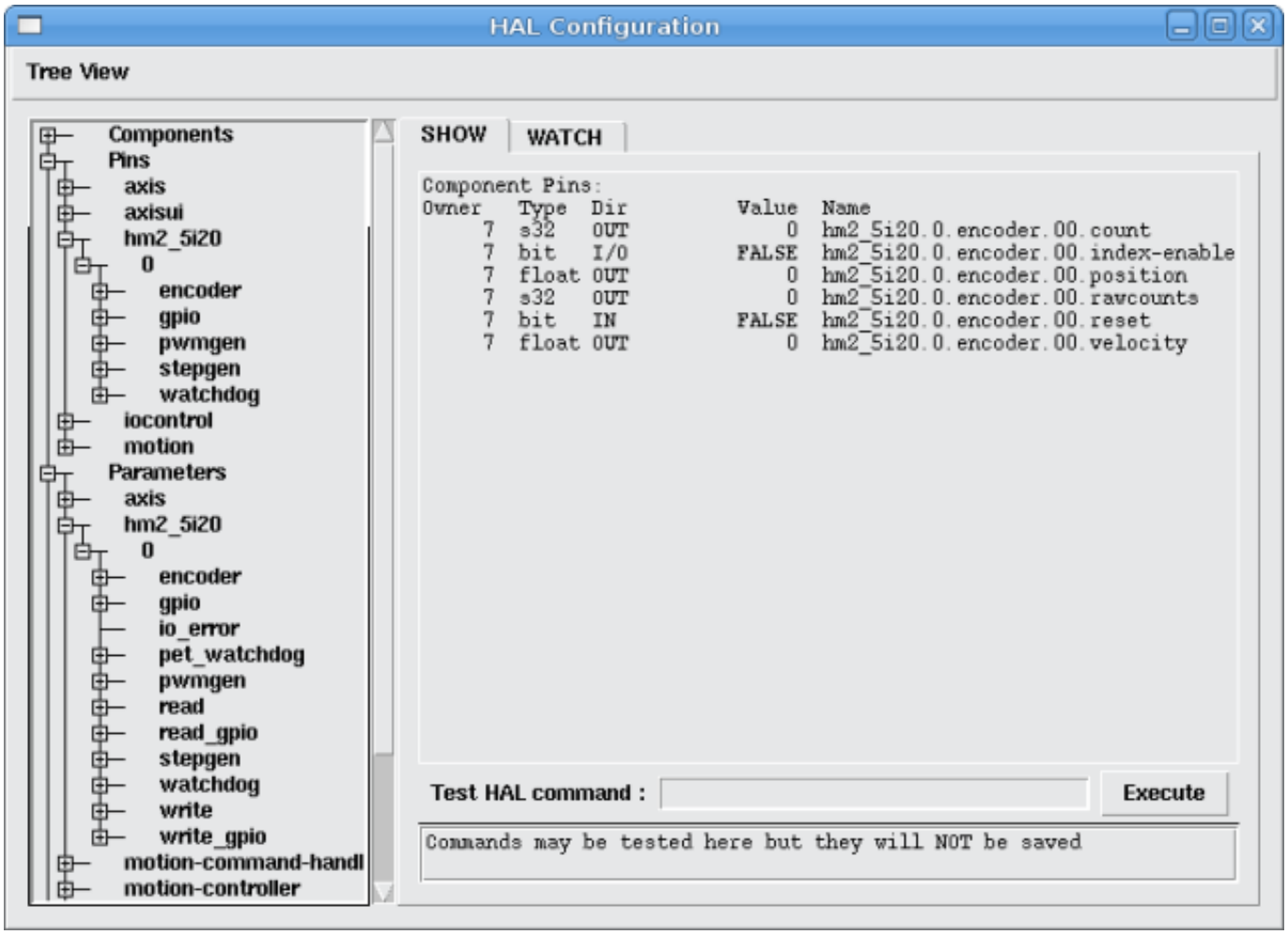



Abbildung 6.9: 5i20 HAL-Pins

### 6.7.11 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 the 7I43 (two ports), SV8 (*Servo 8*) would be for having 8 servos or fewer, using the *classic* 7I33 4-axis (per port) servo board. So 8 servos would use up all 48 signals in the two ports. But if you only needed 3 servos, you could say *num_encoders=3* and *num_pwmgens=3* and recover 5 servos at 6 signals each, thus gaining 30 bits of GPIO.

Or, in the 5I22 (four ports), SVST8_24 (*Servo 8, Stepper 24*) would be for having 8 servos or fewer (7I33 x2 again), and 24 steppers or fewer (7I47 x2). This would use up all four ports. If you only needed 4 servos you could say *num_encoders=4* and *num_pwmgens=4* and recover 1 port (and save a 7I33). And if you only needed 12 steppers you could say *num_stepgens=12* and free up one port (and save a 7I47). So in this way we can save two ports (48 bits) for GPIO.

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) Default Configurations (The 5I22 comes in 1M and 1.5M gate versions. So far, all firmware is available in all gate sizes.)

| 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) Default Configurations (The 5I23 has 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) Default Configurations (The 5I20 has 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) Default Configurations (The 4I68 has 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) Default Configurations (The 4I65 has 200k gates.)

| Firmware | Encoder | PWM | StepGen | GPIO |
|---|---|---|---|---|
| SV12 | 12 | 12 | 0 | 0 |
| SVST8_4 | 8 | 8 | 4 | 0 |

| Firmware | Encoder | PWM | StepGen | GPIO |
|---|---|---|---|---|
| SVST8_4IM2 | 8 (+IM) | 8 | 4 | 8 |

7I43 (2-port parallel) 400k gate versions, Default Configurations

| 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-port parallel) 200k gate versions, Default Configurations

| 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.

## 6.7.12 GPIO

Allgemeine E/A-Pins auf der Karte, wenn nicht von einer Modulinstanz verwendet, werden als *volle* GPIO-Pins an HAL exportiert. Full-GPIO-Pins können zur Laufzeit als Eingänge, Ausgänge oder Open Drains konfiguriert werden und verfügen über eine HAL-Schnittstelle, die diese Flexibilität offenlegt. E/A-Pins einer aktiven Modulinstanz sind durch die Anforderungen dieses sie besitzenden Moduls eingeschränkt und haben eine eingeschränkte HAL-Schnittstelle.

GPIOs haben Namen wie *hm2_<BoardType>.<BoardNum>.gpio.<IONum>.* IONum ist eine dreistellige Zahl. Die Zuordnung von IONum zu Stecker und Pin-auf-dem-Stecker wird in das Syslog geschrieben, wenn der Treiber geladen wird, und sie ist im Mesa-Handbuch für die Anything I/O-Boards dokumentiert.

Die hm2-GPIO-Darstellung ist den digitalen Eingängen und digitalen Ausgängen nachempfunden, die in der kanonischen Geräteschnittstelle (Teil des Dokuments HAL General Reference) beschrieben sind.

GPIO-Pins sind standardmäßig auf Eingang eingestellt.

### 6.7.12.1 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.

#### 6.7.12.2 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.

### 6.7.13 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:

#### 6.7.13.1 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.

#### 6.7.13.2 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.

#### 6.7.13.3 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.

## 6.7.14 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:

### 6.7.14.1 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.

### 6.7.14.2 Parameter

- *output-type* - (s32, RW) This emulates the output_type load-time argument to the software pwmgen component. This parameter may be changed at runtime, but most of the time you probably want to set it at startup and then leave it alone. Accepted values are 1 (PWM on Out0 and Direction on Out1), 2 (Up on Out0 and Down on Out1), 3 (PDM mode, PDM on Out0 and Dir on Out1), and 4 (Direction on Out0 and PWM on Out1, *for locked antiphase*).
- *scale* - (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, then 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, then it will be clipped to the max supported frequency of the board. Frequencies below about 5 Hz are not terribly accurate, but above 5 Hz they are pretty close.

#### 6.7.14.3 Ausgangsparameter

Die Ausgangspins der einzelnen PWMGen haben zwei zusätzliche Parameter. Um herauszufinden, welcher E/A-Pin zu welchem Ausgang gehört, führen Sie `dmesg` wie oben beschrieben aus.

- `invert_output` - (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.

### 6.7.15 Encoder

Encoder haben Namen wie `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.. Instanz` ist eine zweistellige Zahl, die der Instanznummer des HostMot2-Encoders entspricht. Es gibt *num_encoders* Instanzen, beginnend mit 00.

Jeder Encoder verwendet drei oder vier Eingangs-I/O-Pins, je nachdem, wie die Firmware kompiliert wurde. Dreipolige Encoder verwenden A, B und Index (manchmal auch als Z bekannt). Vierpolige Encoder verwenden A, B, Index und Index-Maske.

Die hm2-Encoder-Darstellung ähnelt derjenigen, die von der kanonischen Geräteschnittstelle (im Dokument HAL General Reference) beschrieben wird, und der Software-Encoder-Komponente. Jede Encoder-Instanz hat die folgenden Pins und Parameter:

#### 6.7.15.1 Pins

- `count` - (s32, Out) Anzahl der Encoderzählungen seit dem letzten Reset.
- `index-enable` - (Bit, I/O) Wenn dieser Pin auf True gesetzt ist, wird der Zählerstand (und damit auch die Position) beim nächsten Indeximpuls (Phase-Z) auf Null zurückgesetzt. Gleichzeitig wird index-enable auf Null zurückgesetzt, um anzuzeigen, dass der Impuls stattgefunden hat.
- `position` - (Float, Out) Encode-Position in Positionseinheiten (Count / Scale).
- `rawcounts` - (s32, Out) Gesamtzahl der Encoder-Zählungen seit dem Start, nicht für Index oder Reset angepasst.
- `reset` - (Bit, In) Wenn dieser Pin TRUE ist, werden die Zähl- und Positions-Pins auf 0 gesetzt. Der Wert des Geschwindigkeits-Pins wird davon nicht beeinflusst. Der Treiber setzt diesen Pin nicht auf FALSE zurück, nachdem er die Zählung auf 0 gesetzt hat, das ist Aufgabe des Benutzers.
- `velocity` - (Float, Out) Geschätzte Encoder-Geschwindigkeit in Positionseinheiten pro Sekunde.

#### 6.7.15.2 Parameter

- `counter-mode` - (Bit, RW) Auf False (Standard) für Quadratur gesetzt. Auf True gesetzt für Up/Down oder für einen einzelnen Eingang an Phase A. Kann für einen Frequenz/Geschwindigkeits-Wandler mit einem einzelnen Eingang an Phase A verwendet werden, wenn auf True gesetzt.

- `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) Wenn auf True gesetzt, löst die steigende Flanke des Index-Eingangspins das Index-Ereignis aus (wenn index-enable True ist). Wenn auf False gesetzt, löst die fallende Flanke aus.
- `index-mask` - (Bit, RW) Wenn auf True gesetzt, hat der Index-Eingangs-Pin nur dann eine Wirkung, wenn der Index-Mask-Eingangs-Pin True ist (oder False, abhängig vom `index-mask-invert`-Pin unten).
- `index-mask-invert` - (Bit, RW) Wenn auf True gesetzt, muss Index-Mask False sein, damit Index eine Wirkung hat. Bei False muss der `index-mask`-Pin auf True gesetzt werden.
- `scale` - (Float, RW) Konvertiert von *count* Einheiten in *position* Einheiten. Ein Quadratur-Drehgeber hat normalerweise 4 Zählungen pro Impuls, so dass ein 100 PPR-Drehgeber 400 Zählungen pro Umdrehung aufweisen würde. Im `.counter-mode` würde ein 100 PPR Encoder 100 Zählungen pro Umdrehung haben, da er nur die ansteigende Flanke von A verwendet und die Richtung B ist.
- `vel-timeout` - (Float, RW) Wenn sich der Encoder langsamer bewegt als ein Impuls für jedes Mal, wenn der Treiber den Zählerstand aus dem FPGA liest (in der Funktion hm2_read()), ist die Geschwindigkeit schwieriger zu schätzen. Der Treiber kann mehrere Iterationen abwarten, bis der nächste Impuls eintrifft, während er gleichzeitig die obere Grenze der Encoder-Geschwindigkeit meldet, die genau abgeschätzt werden kann. Dieser Parameter gibt an, wie lange auf den nächsten Impuls gewartet werden soll, bevor der Encoder gestoppt wird. Dieser Parameter wird in Sekunden angegeben.

## 6.7.16 5I25 Konfiguration

### 6.7.16.1 Firmware

Die 5I25-Firmware wird für die Tochterkarte, mit der sie gekauft wurde, vorinstalliert. Die `firmware=xxx.BI` ist also nicht Teil des hm2_pci-Konfigurationsstrings, wenn eine 5I25 verwendet wird.

### 6.7.16.2 Konfiguration

Beispielkonfigurationen der Karten 5I25/7I76 und 5I25/7I77 sind im Konfigurations-Auswahl enthalten.

Wenn Sie Ihre eigene Konfiguration erstellen möchten, zeigen die folgenden Beispiele, wie Sie die Treiber in die HAL-Datei laden.

**5I25 + 7I76 Karte**

```
# den generischen Treiber laden
loadrt hostmot2

# PCI-Treiber laden und konfigurieren
loadrt hm2_pci config=”num_encoders=1 num_stepgens=5 sserial_port_0=0XXX”
```

**5I25 + 7I77 Karte**

```
# den generischen Treiber laden
loadrt hostmot2

# Laden Sie den PCI-Treiber und konfigurieren Sie ihn
loadrt hm2_pci config=”num_encoders=6 num_pwmgens=6 sserial_port_0=0XXX”
```

#### 6.7.16.3 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)].

#### 6.7.16.4 7I77 Grenzwerte

Minlimit und Maxlimit sind Begrenzungen des Pin-Wertes (in diesem Fall des Analogausgangswertes), Fullscalemax ist der Skalierungsfaktor.

Diese sind standardmäßig auf den analogen Eingang oder den analogen Bereich (meist in Volt) eingestellt.

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)

### 6.7.17 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.

## 6.8 MB2HAL

### 6.8.1 Einführung

MB2HAL is a generic non-realtime HAL component to communicate with one or more Modbus devices. So far, there are two options to communicate with a Modbus device:

1. One option is to create a HAL component as a driver see VFD Modbus.

2. Another option is to use Classic Ladder which has Modbus built in, see ClassicLadder.
3. Jetzt gibt es eine dritte Option, die aus einem ”generischen” Treiber besteht, der per Textdatei konfiguriert wird, dieser heißt MB2HAL.

Why MB2HAL? Consider using MB2HAL if:

- Sie einen neuen Treiber schreiben müssenund keine Ahnung haben vom Programmieren.
- Sie müssen Classic Ladder ”nur” verwenden, um die Modbus-Verbindungen zu verwalten.
- You have to discover and configure first time the Modbus transactions. MB2HAL have debug levels to facilitate the low level protocol debug.
- You have more than one device to connect. MB2HAL is very efficiently managing multiple devices, transactions and links. Currently I am monitoring two axis drivers using a Rs232 port, a VFD driver using another Rs232 port, and a remote I/O using TCP/IP.
- You want a protocol to connect your Arduino to HAL. Look the included sample configuration file, sketch and library for Arduino Modbus.

## 6.8.2 Anwendung

a. Erstellen Sie eine Konfigurationsdatei nach folgendem Beispiel

   1. Komponentenname festlegen (optional)
      Set `HAL_MODULE_NAME=mymodule` (Voreinstellung `HAL_MODULE_NAME=mb2hal`)
   2. Load the modbus HAL non-realtime component

b. Standard-Komponentenname: `loadusr -W mb2hal config=config_file.ini`

c. Benutzerdefinierter Komponentenname: `loadusr -Wn mymodule mb2hal config=config_file.ini`

## 6.8.3 Optionen

### 6.8.3.1 Init-Abschnitt

`[MB2HAL_INIT]`

| Wert | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| `INIT_DEBUG` | Integer | Nein | Debug-Ebene der Init- und INI-Datei-Analyse.<br>0 = stumm<br>1 = Fehlermeldungen (Standard)<br>2 = OK-Bestätigungsmeldungen<br>3 = Debugging-Meldungen<br>4 = maximale Fehlersuchmeldungen (nur in Transaktionen) |
| `VERSION` | Zeichenfolge (engl. string) | Nein | Version number in the format N.N[NN]. Defaults to 1.0. |
| `HAL_MODULE_NAME` | Zeichenfolge (engl. string) | Nein | HAL module (component) name. Defaults to ”mb2hal”. |

| Wert | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| 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. |

#### 6.8.3.2 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 | You must specify either a ”serial” or ”tcp” link for the first transaction. Later transactions will use the previous transaction link if not specified. |
| TCP_IP | IP Adresse | If LINK_TYPE=tcp | The Modbus slave device IP address. Ignored if LINK_TYPE=serial. |
| TCP_PORT | Integer | Nein | The Modbus slave device TCP port. Defaults to 502. Ignored if LINK_TYPE=serial. |
| SERIAL_PORT | Zeichenfolge (engl. string) | If LINK_TYPE=serial | The serial port. For example ”/dev/ttyS0”. Ignored if LINK_TYPE=tcp. |
| SERIAL_BAUD | Integer | If LINK_TYPE=serial | The baud rate. Ignored if LINK_TYPE=tcp. |
| SERIAL_BITS | Integer | If LINK_TYPE=serial | Data bits. One of 5, 6, 7, 8. Ignored if LINK_TYPE=tcp. |
| SERIAL_PARITY | Zeichenfolge (engl. string) | If LINK_TYPE=serial | Data parity. One of: even, odd, none. Ignored if LINK_TYPE=tcp. |
| SERIAL_STOP | Integer | If LINK_TYPE=serial | Stop bits. One of 1, 2. Ignored if LINK_TYPE=tcp. |
| SERIAL_DELAY_MS | Integer | If LINK_TYPE=serial | Serial port delay between transactions of this section only. In ms. Defaults to 0. Ignored if 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 | 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. |
| PIN_NAMES | Liste | Sofern nicht NELEMENTS angegeben ist | 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. Example: PIN_NAMES=cycle_start,stop,feed_hold |

| Wert | Typ | Erforderlich | Beschreibung |
|---|---|---|---|
| MB_TX_CODE | Zeichenfolge (engl. string) | Ja | Modbus Transaction Functions-Code (siehe Spezifikationen):<br>• fnct_01_read_coils<br>• fnct_02_read_discrete_inputs<br>• fnct_03_read_holding_registers<br>• fnct_04_read_input_registers<br>• fnct_05_write_single_coil<br>• fnct_06_write_single_register<br>• fnct_15_write_multiple_coils<br>• fnct_16_write_multiple_registers |
| MB_RESPONSE_TIMEOUT_MS | Integer | Nein | Response timeout for this transaction. In ms. Defaults to 500 ms. This is how much to wait for 1st byte before raise an error. |
| MB_BYTE_TIMEOUT_MS | Integer | Nein | Byte timeout for this transaction. In ms. Defaults to 500 ms. This is how much to wait from byte to byte before raise an error. |
| HAL_TX_NAME | Zeichenfolge (engl. string) | Nein | 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 don't end up with two transactions using the same name. |
| MAX_UPDATE_RATE | Gleitkommazahl (engl. float) | Nein | 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. |
| DEBUG | Zeichenfolge (engl. string) | Nein | Debug level for this transaction only. See `INIT_DEBUG` parameter above. |

#### 6.8.3.3 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

### 6.8.4 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: Set to 1.1 to enable the new functions:
# - fnct_01_read_coils
# - fnct_05_write_single_coil
# - changed pin names (see https://linuxcnc.org/docs/2.9/html/drivers/mb2hal.html#_pins).
VERSION=1.1

#OPTIONAL: HAL module (component) name. Defaults to ”mb2hal”.
HAL_MODULE_NAME=mb2hal

#OPTIONAL: Insert a delay of ”FLOAT seconds” between transactions in order
#to not to have a lot of logging and facilitate the debugging.
```

```
#Useful when using DEBUG=3 (NOT INIT_DEBUG=3)
#It affects ALL transactions.
#Use ”0.0” for normal activity.
SLOWDOWN=0.0

#REQUIRED: The number of total Modbus transactions. There is no maximum.
TOTAL_TRANSACTIONS=9

# ++++++++++++++++++++++++
# Transactions
# ++++++++++++++++++++++++
#One transaction section is required per transaction, starting at 00 and counting up  ↩
    sequentially.
#If there is a new link (not transaction), you must provide the REQUIRED parameters 1st  ↩
    time.
#Warning: Any OPTIONAL parameter not specified are copied from the previous transaction.
[TRANSACTION_00]

#REQUIRED: You must specify either a ”serial” or ”tcp” link for the first transaction.
#Later transaction will use the previous transaction link if not specified.
LINK_TYPE=tcp

#if LINK_TYPE=tcp then REQUIRED (only 1st time): The Modbus slave device ip address.
#if LINK_TYPE=serial then IGNORED
TCP_IP=192.168.2.10

#if LINK_TYPE=tcp then OPTIONAL.
#if LINK_TYPE=serial then IGNORED
#The Modbus slave device tcp port. Defaults to 502.
TCP_PORT=502

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The serial port.
SERIAL_PORT=/dev/ttyS0

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The baud rate.
SERIAL_BAUD=115200

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data bits. One of 5,6,7,8.
SERIAL_BITS=8

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data parity. One of: even, odd, none.
SERIAL_PARITY=none

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Stop bits. One of 1, 2.
SERIAL_STOP=2

#if LINK_TYPE=serial then OPTIONAL:
#if LINK_TYPE=tcp then IGNORED
#Serial port delay between for this transaction only.
#In ms. Defaults to 0.
SERIAL_DELAY_MS=10
```

```
#REQUIRED (only 1st time).
#Modbus slave number.
MB_SLAVE_ID=1

#REQUIRED: The first element address (decimal integer).
FIRST_ELEMENT=0

#REQUIRED unless PIN_NAMES is specified: The number of elements.
#It is an error to specify both NELEMENTS and PIN_NAMES
#The pin names will be sequential numbers e.g mb2hal.plcin.01
#NELEMENTS=4

#REQUIRED unless NELEMENTS is specified: A list of element names.
#these names will be used for the pin names, e.g mb2hal.plcin.cycle_start
#NOTE: there must be no white space characters in the list
PIN_NAMES=cycle_start,stop,feed_hold

#REQUIRED: Modbus transaction function code (see www.modbus.org specifications).
#    fnct_01_read_coils               (01 = 0x01) (new in 1.1)
#    fnct_02_read_discrete_inputs     (02 = 0x02)
#    fnct_03_read_holding_registers   (03 = 0x03)
#    fnct_04_read_input_registers     (04 = 0x04)
#    fnct_05_write_single_coil        (05 = 0x05) (new in 1.1)
#    fnct_06_write_single_register    (06 = 0x06)
#    fnct_15_write_multiple_coils     (15 = 0x0F)
#    fnct_16_write_multiple_registers (16 = 0x10)
#
# Created pins:
# fnct_01_read_coils:
# fnct_02_read_discrete_inputs:
#     mb2hal.m.n.bit     (output)
#     mb2hal.m.n.bit-inv (output)
# fnct_03_read_holding_registers:
# fnct_04_read_input_registers:
#     mb2hal.m.n.float   (output)
#     mb2hal.m.n.int     (output)
# fnct_05_write_single_coil:
#     mb2hal.m.n.bit     (input)
#         NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
# fnct_06_write_single_register:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
# fnct_15_write_multiple_coils:
#     mb2hal.m.n.bit     (input)
# fnct_16_write_multiple_registers:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
#
# m = HAL_TX_NAME or transaction number if not set, n = element number (NELEMENTS) or name  ↩
   from PIN_NAMES
# Example: mb2hal.00.01.<type> (transaction=00, second register=01 (00 is the first one))
#          mb2hal.TxName.01.<type> (HAL_TX_NAME=TxName, second register=01 (00 is the first ↩
    one))
MB_TX_CODE=fnct_03_read_holding_registers

#OPTIONAL: Response timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait for 1st byte before raise an error.
```

```
MB_RESPONSE_TIMEOUT_MS=500

#OPTIONAL: Byte timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait from byte to byte before raise an error.
MB_BYTE_TIMEOUT_MS=500

#OPTIONAL: Instead of giving the transaction number, use a name.
#Example: mb2hal.00.01 could become mb2hal.plcin.01
#The name must not exceed 28 characters.
#NOTE: when using names be careful that you dont end up with two transactions
#using the same name.
HAL_TX_NAME=remoteIOcfg

#OPTIONAL: Maximum update rate in HZ. Defaults to 0.0 (0.0 = as soon as available =  ↩
    infinite).
#NOTE: This is a maximum rate and the actual rate may be lower.
#If you want to calculate it in ms use (1000 / required_ms).
#Example: 100 ms = MAX_UPDATE_RATE=10.0, because 1000.0 ms / 100.0 ms = 10.0 Hz
MAX_UPDATE_RATE=0.0

#OPTIONAL: Debug level for this transaction only.
#See INIT_DEBUG parameter above.
DEBUG=2

#While DEBUGGING transactions note the returned ”ret[]” value correspond to:
#/* Modbus protocol exceptions */
#ILLEGAL_FUNCTION        -0x01 the FUNCTION code received in the query is not allowed or  ↩
    invalid.
#ILLEGAL_DATA_ADDRESS    -0x02 the DATA ADDRESS received in the query is not an allowable  ↩
    address for the slave or is invalid.
#ILLEGAL_DATA_VALUE      -0x03 a VALUE contained in the data query field is not an  ↩
    allowable value or is invalid.
#SLAVE_DEVICE_FAILURE    -0x04 SLAVE (or MASTER) device unrecoverable FAILURE while  ↩
    attempting to perform the requested action.
#SERVER_FAILURE          -0x04 (see above).
#ACKNOWLEDGE             -0x05 This response is returned to PREVENT A TIMEOUT in the master ↩
    .
#                              A long duration of time is required to process the request  ↩
    in the slave.
#SLAVE_DEVICE_BUSY       -0x06 The slave (or server) is BUSY. Retrasmit the request later.
#SERVER_BUSY             -0x06 (see above).
#NEGATIVE_ACKNOWLEDGE    -0x07 Unsuccessful programming request using function code 13 or  ↩
    14.
#MEMORY_PARITY_ERROR     -0x08 SLAVE parity error in MEMORY.
#GATEWAY_PROBLEM_PATH    -0x0A (-10) Gateway path(s) not available.
#GATEWAY_PROBLEM_TARGET  -0x0B (-11) The target device failed to 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
```

### 6.8.5 Pins

**Anmerkung**
Yellow = New in MB2HAL 1.1 (LinuxCNC 2.9) To use these new features you have to set `VERSION = 1.1`.
*m* = Value of `HAL_TX_NAME` if set or transaction number
*n* = Element number (`NELEMENTS`) or name from `PIN_NAMES`
Beispiel:

- `mb2hal.00.01.int` (TRANSACTION_00, second register)
- `mb2hal.readStatus.01.bit` (HAL_TX_NAME=readStatus, first bit)

#### 6.8.5.1 fnct_01_read_coils

- mb2hal.*m*.*n*.bit *bit out*
- mb2hal.*m*.*n*.bit-inv *bit out*

#### 6.8.5.2 fnct_02_read_discrete_inputs

- mb2hal.*m*.*n*.bit *bit out*
- mb2hal.*m*.*n*.bit-inv *bit out*

#### 6.8.5.3 fnct_03_read_holding_registers

- mb2hal.*m*.*n*.float *float out*
- mb2hal.*m*.*n*.int *s32 out*

#### 6.8.5.4 fnct_04_read_input_registers

- mb2hal.*m*.*n*.float *float out*
- mb2hal.*m*.*n*.int *s32 out*

#### 6.8.5.5 fnct_05_write_single_coil

- mb2hal.*m*.*n*.bit *bit in*

`NELEMENTS` muss 1 sein oder `PIN_NAMES` darf nur einen Namen enthalten.

#### 6.8.5.6 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).

#### 6.8.5.7 fnct_15_write_multiple_coils

- mb2hal.*m*.*n*.bit *bit in*

#### 6.8.5.8 fnct_16_write_multiple_registers

- mb2hal.*m*.*n*.float *float in*
- mb2hal.*m*.*n*.int *s32 in*

Both pin values are added and limited to 65535 (UINT16_MAX). Use one and let the other open (read as 0).

## 6.9 Mitsub VFD-Treiber

This is a non-realtime HAL program, written in Python, to control VFDs from Mitsubishi.
Specifically the A500 F500 E500 A500 D700 E700 F700 series - others may work.
mitsub_vfd supports serial control using the RS485 protocol.
Conversion from USB or serial port to RS485 requires special hardware.

---

**Anmerkung**
Since this is a non-realtime program it can be affected by computer loading and latency. It is possible to lose control of the VFDs. It is optional to set the VFD to stop if it loses communication if that is desirable. One should always have an Estop circuit that kills the power to the unit in case of emergency.

---

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)

### 6.9.1 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.

---

### 6.9.2 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) some models (eg E500) cannot monitor status - set the monitor pin to false in this case pins such as up-to-speed, amps, alarm and status bits are not updated.
- *<n>.motor-cmd* (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 to 7, status bits are user configurable on the VFD. Bit 3 should be set to at speed and bit 7 should be set to alarm. Others are free to be set as required.

### 6.9.3 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
```

## 6.9.4 Konfigurieren des Mitsubishi VFD für die serielle Nutzung

### 6.9.4.1 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.

### 6.9.4.2 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* set to 1-10 *-if 10 (maximum) COM errors then VFD faults.*
- *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.*

# 6.10 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.

## 6.10.1 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.

### 6.10.2 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.

### 6.10.3 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.

## 6.11 Opto22 Treiber

**PCI AC5 ADAPTER KARTE / HAL TREIBER**

### 6.11.1 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!

### 6.11.2 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.

### 6.11.3 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.

### 6.11.4 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.

### 6.11.5 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.

### 6.11.6 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.

### 6.11.7 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.

## 6.12 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.

### 6.12.1 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.

### 6.12.2 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.

### 6.12.3 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 Notaus (engl. E-stop) auf falsch geht, 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.

### 6.12.4 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.

## 6.13 Pluto P-Treiber

### 6.13.1 Allgemeine Informationen

Das Pluto-P ist ein FPGA-Board mit dem ACEX1K-Chip von Altera.

#### 6.13.1.1 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.

---

For more information on PCI EPP compatible parallel port cards see the LinuxCNC Supported Hardware page on the wiki.

#### 6.13.1.2 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.

#### 6.13.1.3 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.

#### 6.13.1.4 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*).

#### 6.13.1.5 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.

#### 6.13.1.6 PC-Schnittstelle

- Es wird nur eine einzige pluto_servo oder pluto_step Karte unterstützt.

#### 6.13.1.7 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.

#### 6.13.1.8 Für weitere Informationen

Some additional information about it is available from KNJC LLC and from the developer’s blog.

### 6.13.2 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.

#### 6.13.2.1 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)

.Pluto-Servo-Pinbelegung Pluto-Servo-Pinbelegung

Tabelle 6.28: 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 |
| **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 |

Tabelle 6.28: (continued)

| **Primäre Funktion** | **Alternative Funktion** | **Verhalten bei Verwendung beider Funktionen** |
|---|---|---|
| | 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 |

#### 6.13.2.2 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.

#### 6.13.2.3 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*.

#### 6.13.2.4 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.

### 6.13.3 Pluto Step

Pluto-step is suitable for control of a 3- or 4-axis CNC mill with stepper motors. The large number of inputs allows for a full set of limit switches.

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

#### 6.13.3.1 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.

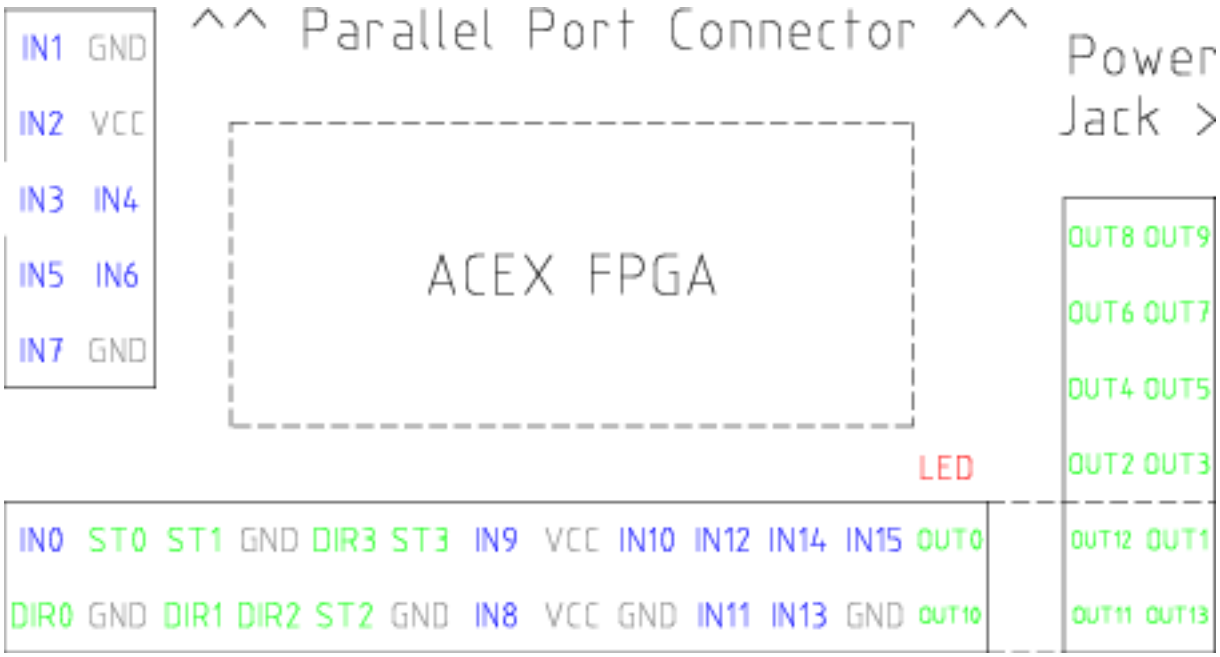



Abbildung 6.10: Pluto-Step Pinout

#### 6.13.3.2 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.

#### 6.13.3.3 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.

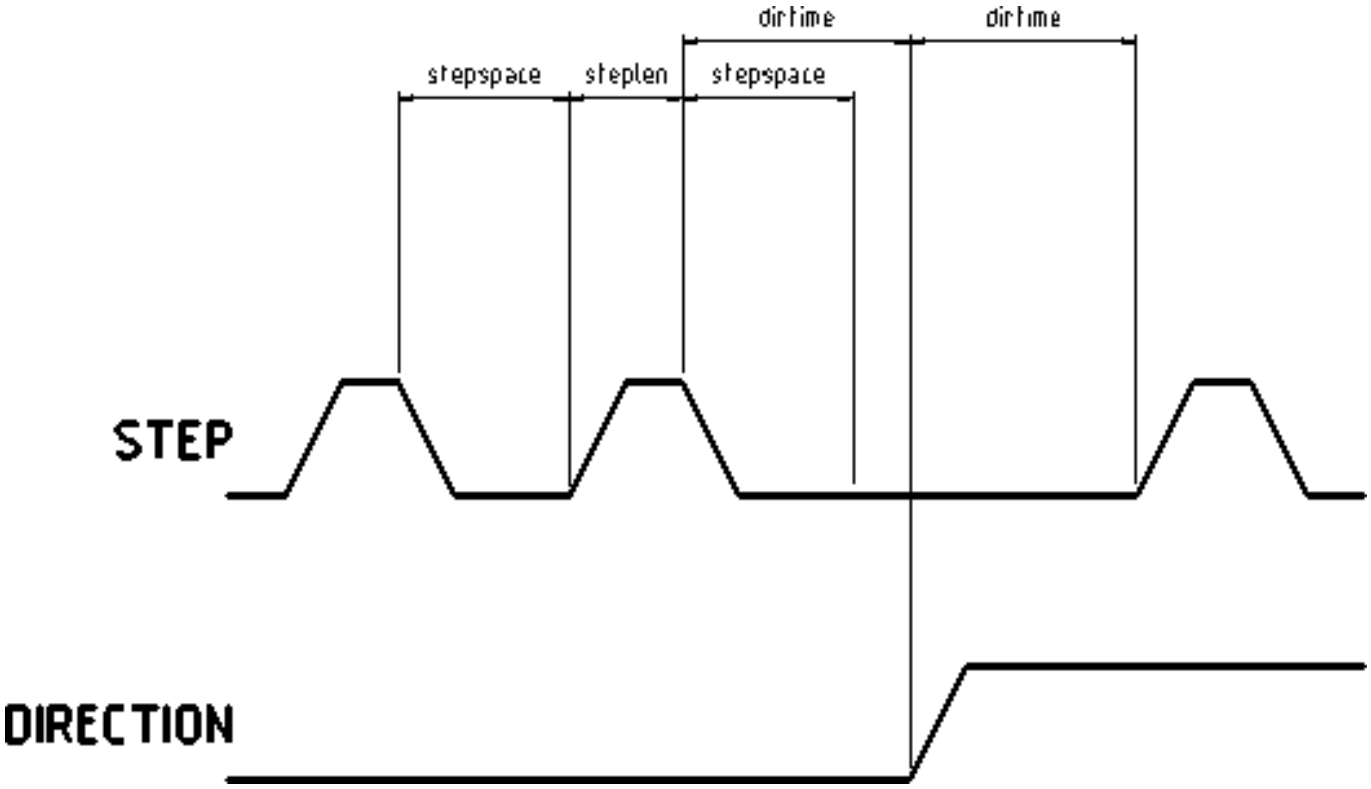



Abbildung 6.11: Pluto-Step-Timings

#### 6.13.3.4 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*.

## 6.14 Powermax Modbus-Treiber

This is a non-realtime HAL program, written in python, to control Hypetherm Powermax plasma cutters using the Modbus ASCII protocol over RS485.

---

**Anmerkung**
Since this is a non-realtime program it can be affected by computer loading and latency. It is possible to lose communications which will be indicated by a change in the status output. One should always have an Estop circuit that kills the power to the unit in case of emergency.

---

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.

### 6.14.1 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

### 6.14.2 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.

### 6.14.3 Referenz:

- Hypertherm Anwendungshinweis #807220
  ”Powermax45 XP/65/85/105/125® Serielles Kommunikationsprotokoll”

## 6.15 Servo To Go-Treiber

The Servo-To-Go (STG) is one of the first PC motion control cards supported by LinuxCNC. It is an ISA card and it exists in different flavors (all supported by this driver). The board includes up to 8 channels of quadrature encoder input, 8 channels of analog input and output, 32 bits digital I/O, an interval timer with interrupt and a watchdog.

---

**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.

---

### 6.15.1 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.

### 6.15.2 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.

### 6.15.3 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.

### 6.15.4 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.

# 6.16 Shuttle

## 6.16.1 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**
The Shuttle devices have an internal 8-bit counter for the current jog-wheel position. The shuttle driver can not know this value until the Shuttles device sends its first event. When the first event comes into the driver, the driver uses the device's reported jog-wheel position to initialize counts to 0.
Das heißt, wenn das erste Ereignis durch eine Jogwheel-Bewegung erzeugt wird, geht diese erste Bewegung verloren.
Any user interaction with the Shuttle device will generate an event, informing the driver of the jog-wheel position. So if you (for example) push one of the buttons at startup, the jog-wheel will work fine and notice the first click.

## 6.16.2 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.

## 6.16.3 Pins

All HAL pin names are prefixed with `shuttle` followed by the index of the device (the order in which the driver found them), for example `shuttle.0` or `shuttle.2`.

**<*Prefix*>.button-<ButtonNumber> (bit out)**
Diese Pins sind wahr (1) wenn die Schaltfläche gedrückt wird.

**<*Prefix*>.button-<ButtonNumber>-not (bit out)**
Diese Pins haben den umgekehrten Zustand des Buttons, also sind sie True (1) wenn der Button nicht gedrückt wird.

**<*Prefix*>.counts (s32 out)**
Kumulierte Zählungen des Jogwheels (das innere Rad).

**<*Prefix*>.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.

**<*Prefix*>.spring-wheel-f (float out)**
The current deflection of the spring-wheel (the outer wheel). It's 0.0 at rest, -1.0 at the counter-clockwise extreme, and +1.0 at the clockwise extreme. The Shuttle devices report the spring-wheel position as an integer from -7 to +7, so this pin reports only 15 discrete values in it's range.

## 6.17 VFS11 VFD-Treiber

This is a non-realtime HAL program to control the S11 series of VFDs from Toshiba.

vfs11_vfd supports serial and TCP connections. Serial connections may be RS232 or RS485. RS485 is supported in full- and half-duplex mode. TCP connections may be passive (wait for incoming connection), or active outgoing connections, which may be useful to connect to TCP-based devices or through a terminal server.

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

### 6.17.1 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

Debugging can be toggled by sending a USR1 signal to the vfs11_vfd process. Modbus debugging can be toggled by sending a USR2 signal to vfs11_vfd process (example: `kill -USR1 $(pidof vfs11_vfd)`).

---

**Anmerkung**
Bei seriellen Konfigurationsfehlern kann das Einschalten von verbose zu einer Flut von Timeout-Fehlern führen.

---

### 6.17.2 Pins

Dabei steht <n> für `vfs11_vfd` oder den beim Laden mit der Option -n angegebenen Namen.

- *<n>.acceleration-pattern* (bit, in) when true, set acceleration and deceleration times as defined in registers F500 and F501 respectively. Used in PID loops to choose shorter ramp times to avoid oscillation.
- *<n>.alarm-code* (s32, out) non-zero if drive is in alarmed state. Bitmap describing alarm information (see register FC91 description). Use err-reset (see below) to clear the alarm.
- *<n>.at-speed* (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) enable the VFD. If false, all operating parameters are still read but control is released and panel control is enabled (subject to VFD setup).
- *<n>.err-reset* (bit, in) reset errors (alarms a.k.a Trip and e-stop status). Resetting the VFD may cause a 2-second delay until it’s rebooted and Modbus is up again.
- *<n>.estop* (bit, in) put the VFD into emergency-stopped status. No operation possible until cleared with err-reset or powercycling.
- *<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) the VFD is in emergency stop status (blinking ”E” on panel). Use err-reset to reboot the VFD and clear the e- stop status.
- *<n>.is-stopped* (bit, out) true wenn der VFD 0 Hz Ausgang meldet
- *<n>.max-rpm* (float, R) actual RPM limit based on maximum frequency the VFD may generate, and the motors nameplate values. For instance, if nameplate-HZ is 50, and nameplate-RPM_ is 1410, but the VFD may generate up to 80 Hz, then max-rpm would read as 2256 (80*1410/50). The frequency limit is read from the VFD at startup. To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. See the VF-S11 manual for instructions how to set the maximum frequency.
- *<n>.modbus-ok* (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 for ON and 0 for OFF, enables the VF-S11 *jog mode*. Speed control is disabled, and the output frequency is determined by register F262 (preset to 5 Hz). This might be useful for spindle orientation. In normal mode, the VFD shuts off if the frequency drops below 12 Hz.
- *<n>.status* (s32, out) Drive Status of the VFD (see the TOSVERT VF-S11 Communications Function Instruction Manual, register FD01). A 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) ignore the loop-time parameter and run Modbus at maximum speed, at the expense of higher CPU usage. Suggested use during spindle positioning.

### 6.17.3 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) speed tolerance (default 0.01) for determining whether spindle is at speed (0.01 meaning: Output frequency is within 1% of target frequency)

### 6.17.4 INI-Datei-Konfiguration

This lists all options understood by vfs11_vfd. Typical setups for RS-232, RS-485 and TCP can be found 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
```

### 6.17.5 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
```

### 6.17.6 Bedienung des Panels

The vfs11_vfd driver takes precedence over panel control while it is enabled (see *enable* pin), effectively disabling the panel. Clearing the *enable* pin re-enables the panel. Pins and parameters can still be set, but will not be written to the VFD untile the *enable* pin is set. Operating parameters are still read while bus control is disabled. Exiting the vfs11_vfd driver in a controlled way will release the VFD from the bus and restore panel control.

See the LinuxCNC Integrators Manual for more information. For a detailed register description of the Toshiba VFDs, see the ”TOSVERT VF-S11 Communications Function Instruction Manual” (Toshiba document number E6581222) and the ”TOSVERT VF-S11 Instruction manual” (Toshiba document number E6581158).

### 6.17.7 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.

### 6.17.8 Konfigurieren des VFS11 VFD für die Modbus-Nutzung

#### 6.17.8.1 Anschließen der seriellen Schnittstelle

The VF-S11 has an RJ-45 jack for serial communication. Unfortunately, it does not have a standard RS-232 plug and logic levels. The Toshiba-recommended way is: connect the USB001Z USB-to-serial conversion unit to the drive, and plug the USB port into the PC. A cheaper alternative is a homebrew interface ( hints from Toshiba support, circuit diagram).

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”.

#### 6.17.8.2 Modbus-Einrichtung

Several parameters need setting before the VF-S11 will talk to this module. This can either be done manually with the control panel, or over the serial link - Toshiba supplies a Windows application called *PCM001Z* which can read/set parameters in the VFD. Note - PCM001Z only talks the Toshiba inverter protocol. So the last parameter which you’d want to change is the protocol - set from Toshiba Inverter Protocol to Modbus; thereafter, the Windows app is useless.

To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. I increased them from 50 to 80.

See dump-params.mio for a description of non-standard VF-S11 parameters of my setup. This file is for the modio Modbus interactive utility.

### 6.17.9 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.

The Ubuntu `libmodbus5` and `libmodbus-dev` packages are only available starting from Ubuntu 12 (*Precise Pengolin*). Moreover, these packages lack support for the MODBUS_RTS_MODE_* flags. Therefore, building vfs11_vfd using this library might generate a warning if RTS_MODE= is specified in the INI file.

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 Spindle Speed

If your spindle speed is controlled by an analog signal, (for example, by a VFD with a 0 V to 10 V signal) and you're using a DAC card like the m5i20 to output the control signal:

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 Spindle Speed

If your spindle can be controlled by a PWM signal, use the *pwmgen* component to create the signal:

```
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 Spindle Enable

If you need a spindle enable signal, link your output pin to *spindle.0.on*. To link these pins to a parallel port pin put something like the following in your .hal file, making sure you pick the pin that is connected to your control device.

```
net spindle-enable spindle.0.on => parport.0.pin-14-out
```

### 7.2.4 Spindle Direction

If you have direction control of your spindle, then the HAL pins *spindle*.N.*forward* and *spindle*.N.*reverse* are controlled by the G-codes M3 and M4. Spindle speed S*n* must be set to a positive non-zero value for M3/M4 to turn on spindle motion.

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 Spindle Soft Start

If you need to ramp your spindle speed command and your control does not have that feature it can be done in HAL. Basically you need to hijack the output of *spindle*.N.*speed-out* and run it through a *limit2* component with the scale set so it will ramp the rpm from *spindle*.N.*speed-out* to your device that receives the rpm. The second part is to let LinuxCNC know when the spindle is at speed so motion can begin.

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

Spindle feedback is needed by LinuxCNC to perform any spindle coordinated motions like threading and constant surface speed. LinuxCNC can perform synchronized motion and CSS with any of up to 8 spindles. Which spindles are used is controlled from G-code. CSS is possible with several spindles simultaneously.

Der StepConf Wizard kann die Verbindungen für eine Einspindelkonfiguration für Sie durchführen, wenn Sie Encoder Phase A und Encoder Index als Eingänge auswählen.

Hardware-Annahmen für dieses Beispiel:

- An der Spindel ist ein Drehgeber angeschlossen, der auf der Phase A 100 Impulse pro Umdrehung ausgibt.
- Die A-Phase des Encoders wird an den Pin 10 des Parallelports angeschlossen.
- Der Indeximpuls des Encoders wird an den Parallelport Pin 11 angeschlossen.

Grundlegende Schritte, um die Komponenten hinzuzufügen und zu konfigurieren: [1] [2] [3]

```
# Fügen Sie den Encoder zu HAL hinzu und verbinden Sie ihn mit Threads.
loadrt encoder num_chan=4
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread

# Den HAL-Geber auf 100 Impulse pro Umdrehung einstellen.
setp encoder.3.position-scale 100

# Stellen Sie den HAL-Encoder auf einfache Zählung ohne Quadratur nur auf A ein.
setp encoder.3.counter-mode true

# Verbinden Sie die HAL-Geberausgänge mit LinuxCNC.
net spindle-position encoder.3.position => spindle.0.revs
net spindle-velocity encoder.3.velocity => spindle.0.speed-in
net spindle-index-enable encoder.3.index-enable <=> spindle.0.index-enable

# Verbinden Sie die HAL-Encodereingänge mit dem realen Encoder.
```

[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-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 Spindle At Speed

To enable LinuxCNC to wait for the spindle to be at speed before executing a series of moves, the spindle.*N*.at-speed needs to turn true at the moment the spindle is at the commanded speed. To achieve this you need spindle feedback from an encoder. Since the feedback and the commanded speed are not usually *exactly* the same you should to use the *near* component to determine that the two numbers are close enough.

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.

Using the GS2 component involves very little to set up. We start with a StepConf Wizard generated config. Make sure the pins with ”Spindle CW” and ”Spindle PWM” are set to unused in the parallel port setup screen.

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**

```
# load the non-realtime component for the Automation Direct GS2 VFDs
loadusr -Wn spindle-vfd gs2_vfd -r 9600 -p none -s 2 -n spindle-vfd

# connect the spindle direction pin to the GS2
net gs2-fwd spindle-vfd.spindle-fwd <= spindle.N.forward

# connect the spindle on pin to the GS2
net gs2-run spindle-vfd.spindle-on <= spindle.N.on

# connect the GS2 at speed to the motion at speed
net gs2-at-speed spindle.N.at-speed <= spindle-vfd.at-speed

# connect the spindle RPM to the GS2
net gs2-RPM spindle-vfd.speed-command <= spindle.N.speed-out
```

---

**Anmerkung**
The transmission speed might be able to be faster depending on the exact environment. Both the drive and the command line options must match. To check for transmission errors add the *-v* command line option and run from a terminal.

---

On the GS2 drive itself you need to set a couple of things before the modbus communications will work. Other parameters might need to be set based on your physical requirements but these are beyond the scope of this manual. Refer to the GS2 manual that came with the drive for more information on the drive parameters.

- 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

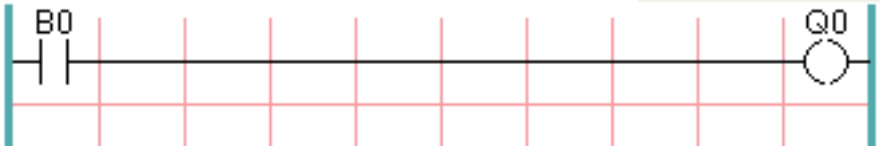
Die gebräuchlichsten Bestandteile von Leitern sind Kontakte (Eingänge), die in der Regel entweder Öffner (NC) oder Schließer (NO) sind, und Spulen (Ausgänge).

- 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:



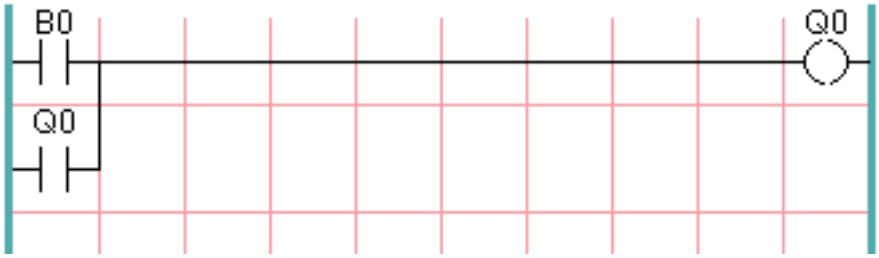

Der Eingang auf der linken Seite, B0, ein Schließer, ist mit der Spule (Ausgang) auf der rechten Seite, Q0, verbunden. Stellen Sie sich nun vor, dass am linken Ende eine Spannung angelegt wird, weil der Eingang B0 aktiv wird (z. B. weil der Eingang aktiviert ist oder der Benutzer den Schließer betätigt 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:

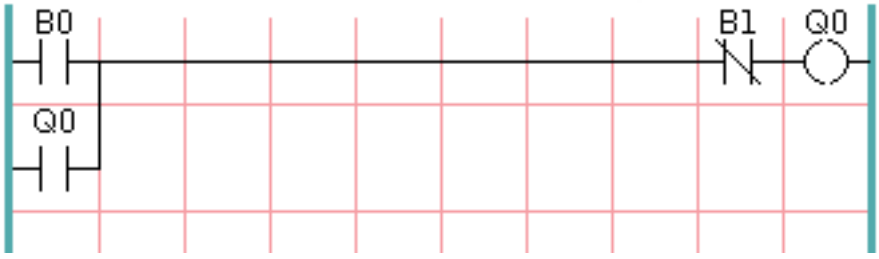



Wenn der Benutzer den Taster B0 drückt, schaltet sich wie zuvor die Spule Q0 ein. Und wenn die Spule Q0 eingeschaltet wird, dann wird der Schalter Q0 eingeschaltet. Jetzt kommt der interessante Teil. Wenn der Benutzer den Druckknopf B0 loslässt, bleibt die Spule Q0 nicht wie zuvor stehen. Das liegt daran, 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 Programming

### 8.2.1 SPS / Kontaktplan Konzepte

ClassicLadder is a type of programming language originally implemented on industrial PLCs (it’s called Ladder Programming). It is based on the concept of relay contacts and coils, and can be used to construct logic checks and functions in a manner that is familiar to many systems integrators. Ladder consists of rungs that may have branches and resembles an electrical circuit. It is important to know how ladder programs are evaluated when running.

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

This can be confusing at first if the output of one line is read by the input of a another rung. There will be one scan before the second input becomes true after the output is set.

Another gotcha with ladder programming is the ”Last One Wins” rule. If you have the same output in different locations of your ladder the state of the last one will be what the output is set to.

### 8.2.2 Sprachen

The most common language used when working with ClassicLadder is *ladder*. ClassicLadder also supports Sequential Function Chart (Grafcet).

### 8.2.3 Komponenten

ClassicLadder besteht aus zwei Komponenten.

- The realtime module classicladder_rt
- The non-realtime module (including a GUI) classicladder

#### 8.2.3.1 Dateien

Typically ClassicLadder components are placed in the custom.hal file if your working from a StepConf generated configuration. These must not be placed in the custom_postgui.hal file or the Ladder Editor menu will be grayed out.

---

**Anmerkung**
Kontaktplan-Dateien (.clp) dürfen keine Leerzeichen im Namen enthalten.

---

#### 8.2.3.2 Echtzeit-Modul

Loading the ClassicLadder real time module (classicladder_rt) is possible from a HAL file, or directly using a halcmd instruction. The first line loads real time the ClassicLadder module. The second line adds the function classicladder.0.refresh to the servo thread. This line makes ClassicLadder update at the servo thread rate.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

The speed of the thread that ClassicLadder is running in directly affects the responsiveness to inputs and outputs. If you can turn a switch on and off faster than ClassicLadder can notice it then you may need to speed up the thread. The fastest that ClassicLadder can update the rungs is one millisecond. You can put it in a faster thread but it will not update any faster. If you put it in a slower than one millisecond thread then ClassicLadder will update the rungs slower. The current scan time will be displayed on the section display, it is rounded to microseconds. If the scan time is longer than one millisecond you may want to shorten the ladder or put it in a slower thread.

#### 8.2.3.3 Variablen

It is possible to configure the number of each type of ladder object while loading the ClassicLadder real time module. If you do not configure the number of ladder objects ClassicLadder will use the default values.

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.

Changing these numbers will change the number of HAL bit pins available. numPhysInputs and numPhysOutputs control how many HAL bit (on/off) pins are available. numS32in and numS32out control how many HAL signed integers (+- integer range) pins are available.

Zum Beispiel (Sie brauchen nicht alle, nur einige wenige zu ändern):

```
loadrt classicladder_rt numRungs=12 numBits=100 numWords=10
numTimers=10 numMonostables=10 numCounters=10 numPhysInputs=10
numPhysOutputs=10 numArithmExpr=100 numSections=4 numSymbols=200
```

```
numS32in=5 numS32out=5
```

Um die Standardanzahl von Objekten zu laden:

```
loadrt classicladder_rt
```

### 8.2.4 Loading the ClassicLadder non-realtime module

ClassicLadder HAL commands must executed before the GUI loads or the menu item Ladder Editor will not function. If you used the Stepper Config Wizard place any ClassicLadder HAL commands in the custom.hal file.

To load the non-realtime module:

```
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

If you load ClassicLadder with the GUI it will display two windows: Section display, and section manager.

#### 8.2.5.1 Sektions-Manager

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Fenster des Abschnittsmanagers.

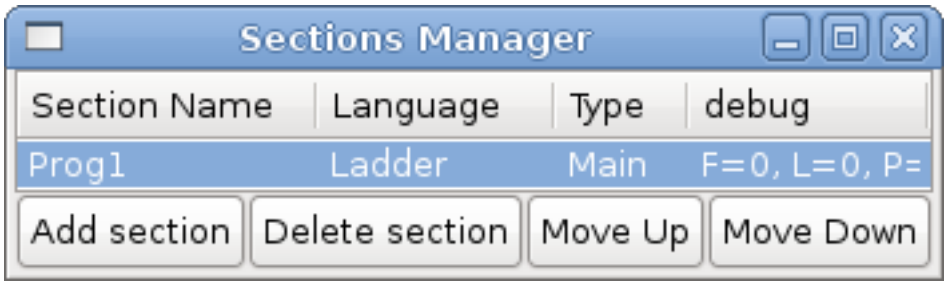



Abbildung 8.1: Sections Manager Standardfenster

This window allows you to name, create or delete sections and choose what language that section uses. This is also how you name a subroutine for call coils.

#### 8.2.5.2 Abschnittsanzeige

Wenn Sie ClassicLadder zum ersten Mal starten, sehen Sie ein leeres Abschnittsanzeigefenster. Es wird eine leere Sprosse angezeigt.

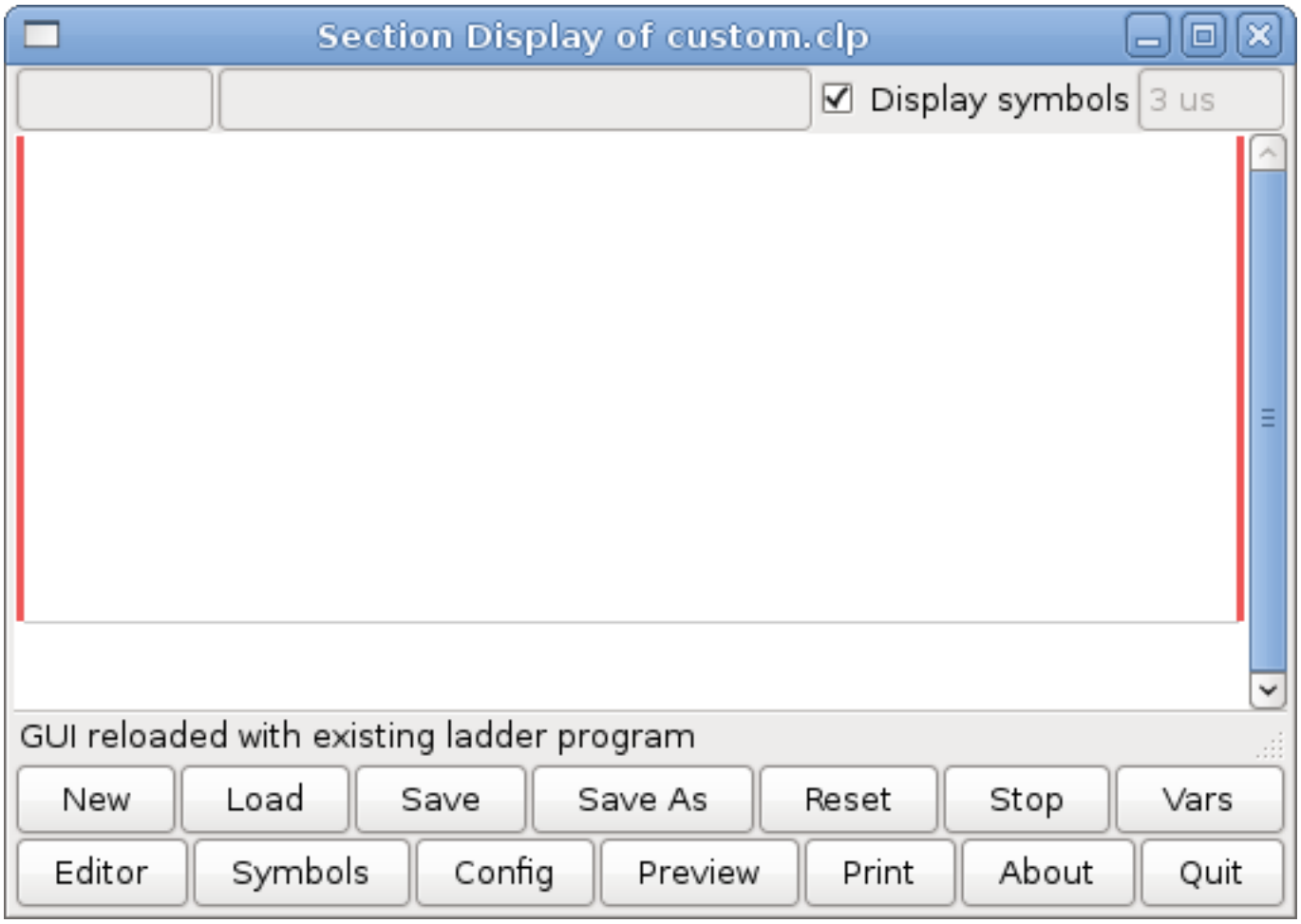



Abbildung 8.2: Standardfenster der Abschnittsanzeige

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).

The Quit button will shut down the non-realtime program, i.e. Modbus and the display. The realtime ladder program will still run in the background.

Über das Kontrollkästchen oben rechts können Sie auswählen, ob Variablennamen oder Symbolnamen angezeigt werden

You might notice that there is a line under the ladder program display that reads ”Project failed to load...”. That is the status bar that gives you info about elements of the ladder program that you click on in the display window. This status line will now display HAL signal names for variables %I, %Q and the first %W (in an equation). You might see some funny labels, such as (103) in the rungs. This is displayed (on purpose) because of an old bug- when erasing elements older versions sometimes didn’t erase the object with the right code. You might have noticed that the long horizontal connection button sometimes did not work in the older versions. This was because it looked for the *free* code but found something else. The number in the brackets is the unrecognized code. The ladder program will still work properly, to fix it erase the codes with the editor and save the program.

#### 8.2.5.3 Die Variablenfenster

This are two variable windows: the Bit Status Window (boolean) and the Watch Window (signed integer). The Vars button is in the Section Display Window, toggle the Vars button to display one, the other, both, then none of the variable windows.

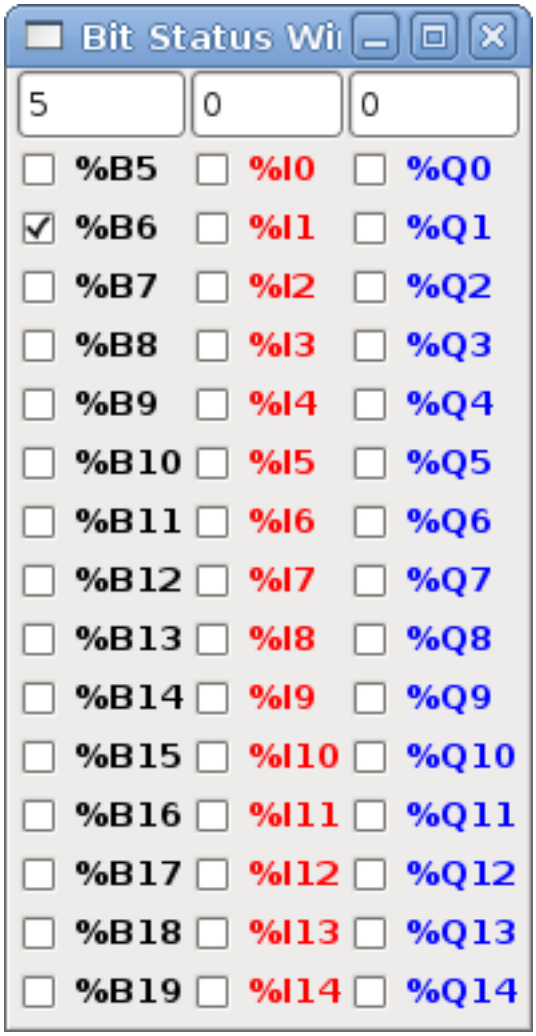



Abbildung 8.3: Bit-Status-Fenster

The Bit Status Window displays some of the boolean (on/off) variable data. Notice all variables start with the % sign. The %I variables represent HAL input bit pins. The %Q represents the relay coil and HAL output bit pins. The %B represents an internal relay coil or internal contact. The three edit areas at the top allow you to select what 15 variables will be displayed in each column. For instance, if the %B Variable column were 15 entries high, and you entered 5 at the top of the column, variables %B5 to %B19 would be displayed. The check boxes allow you to set and unset %B variables manually as long as the ladder program isn’t setting them as outputs. Any Bits that are set as outputs by the program when ClassicLadder is running can not be changed and will be displayed as checked if on and unchecked if off.

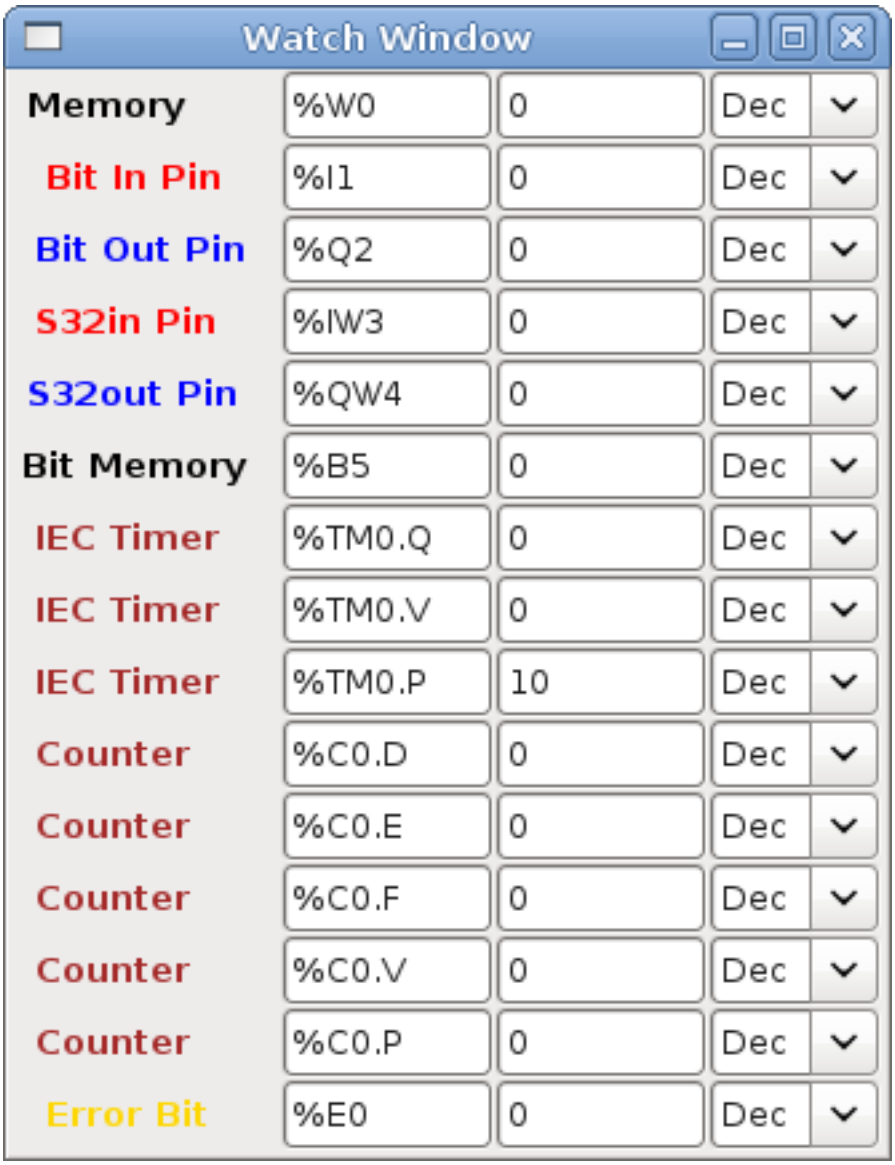



Abbildung 8.4: Schaufenster

The Watch Window displays variable status. The edit box beside it is the number stored in the variable and the drop-down box beside that allow you to choose whether the number to be displayed in hex, decimal or binary. If there are symbol names defined in the symbols window for the word variables showing and the *display symbols* checkbox is checked in the section display window, symbol names will be displayed. To change the variable displayed, type the variable number, e.g. %W2 (if the display symbols check box is not checked) or type the symbol name (if the display symbols checkbox is checked) over an existing variable number/name and press the Enter Key.

#### 8.2.5.4 Symbol-Fenster

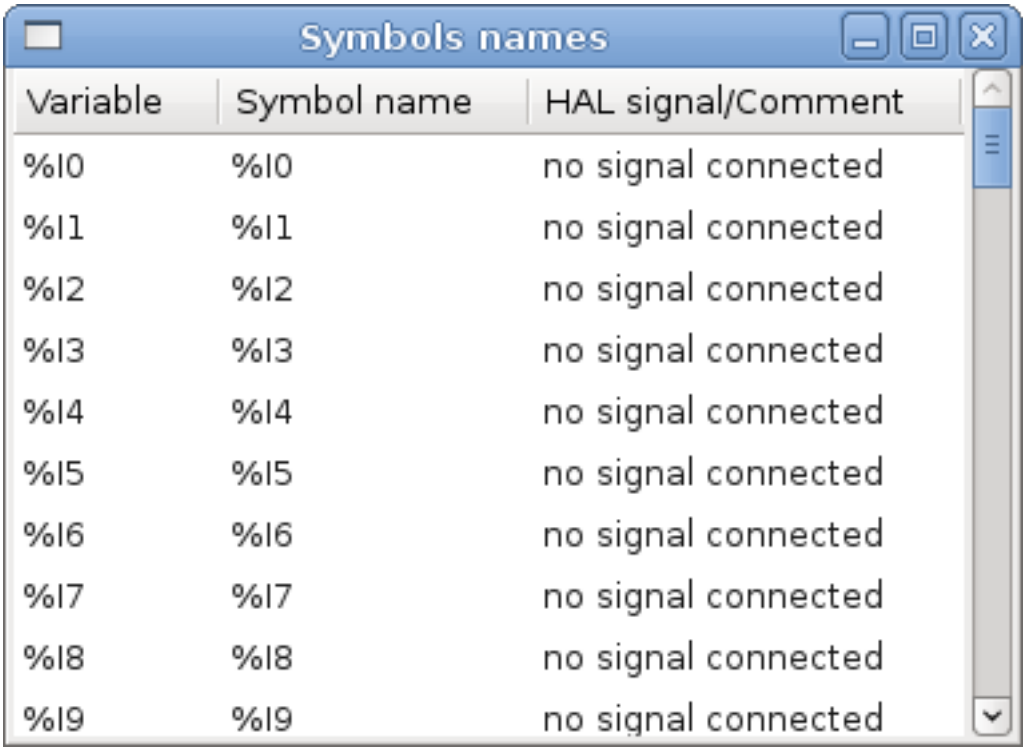



Abbildung 8.5: Fenster mit Symbolnamen

This is a list of *symbol* names to use instead of variable names to be displayed in the section window when the *display symbols* check box is checked. You add the variable name (remember the % symbol and capital letters), symbol name. If the variable can have a HAL signal connected to it (%I, %Q, and %W-if you have loaded s32 pin with the real time module) then the comment section will show the current HAL signal name or lack thereof. Symbol names should be kept short to display better. Keep in mind that you can display the longer HAL signal names of %I, %Q and %W variable by clicking on them in the section window. Between the two, one should be able to keep track of what the ladder program is connected to!

#### 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. Input, N.C. Input, Rising Edge Input, Falling Edge Input

- Horizontal Connection, Vertical Connection, Long Horizontal Connection
- 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* - creates a normally open contact. It can be an external HAL-pin (%I) input contact, an internal-bit coil (%B) contact or a external coil (%Q) contact. The HAL-pin input contact is closed when the HAL-pin is true. The coil contacts are closed when the corresponding coil is active (%Q2 contact closes when %Q2 coil is active).
- *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* - creates a compare block to compare variable to values or other variables, e.g. `%W1<=5` or `%W1=%W2`. Compare cannot be placed in the right most side of the section display.
- *Variable Assignment* - creates an assignment block so you to assign values to variables, e.g. `%W2=7` or `%W1=%W2`. ASSIGNMENT functions can only be placed at the right most side of the section display.

#### 8.2.5.6 Konfigurationsfenster

Das Konfigurationsfenster zeigt den aktuellen Projektstatus und enthält die Registerkarten für die Modbus-Einrichtung.

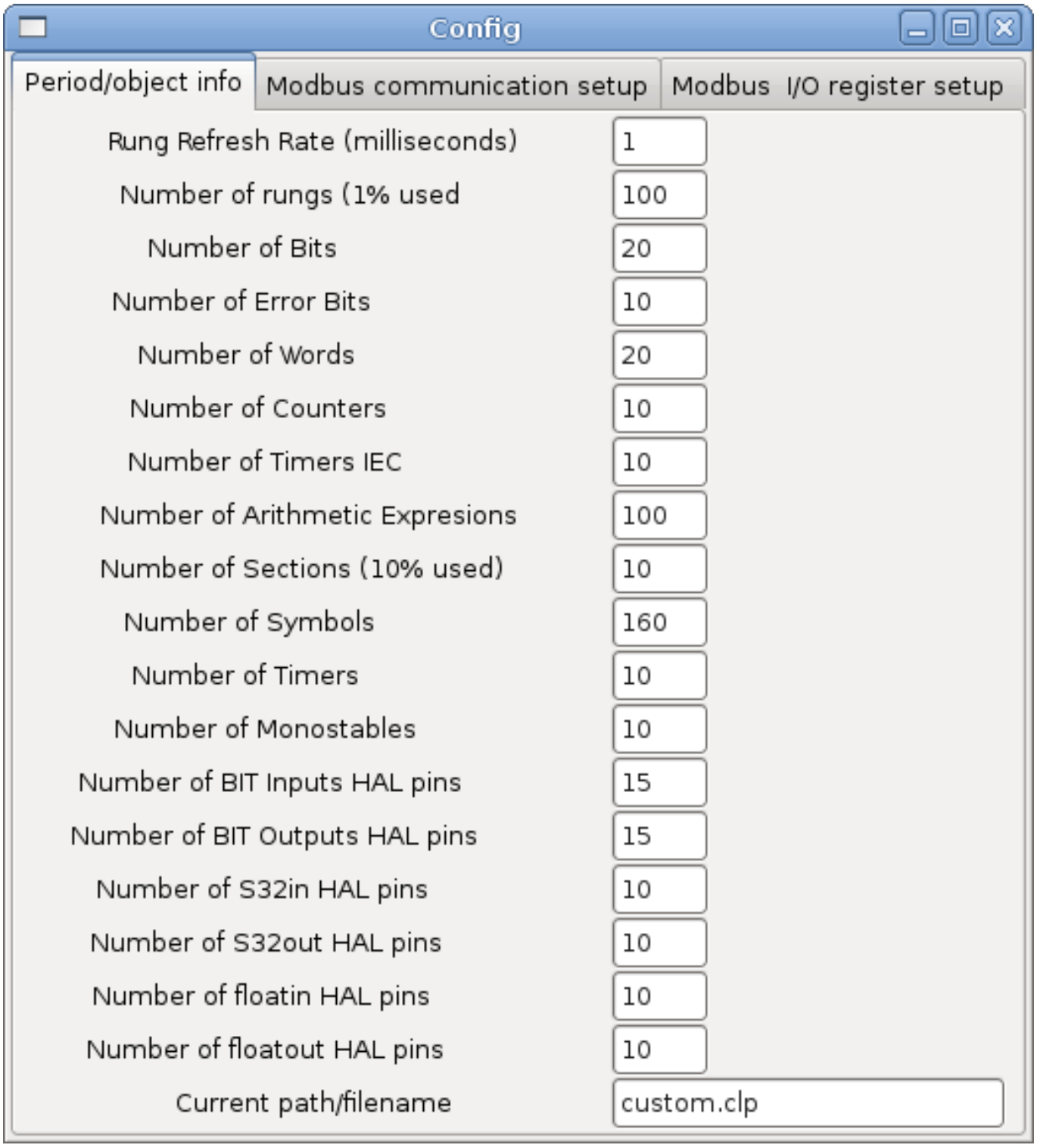



Abbildung 8.7: Konfigurationsfenster

## 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, e.g. `%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* - (Normally Closed) When the variable is false the switch is on.
- *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* - When timer input is true countdown begins and continues as long as input remains true. After countdown is done and as long as timer input is still true the output will be true.
- *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)

The time intervals can be set in multiples of 100&8239;ms, seconds, or minutes.

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.

The I contact is rising edge sensitive meaning it starts the timer only when changing from false to true (or off to on). While the timer is running the I contact can change with no effect to the running timer. R will be true and stay true till the timer finishes counting to zero. The timer base can be multiples of milliseconds, seconds, or minutes.

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.

- '%C'*xx*`.D` - Counter *xx* done (*Boolean*, *read only*)
- '%C'*xx*`.E` - Counter *xx* empty overflow (*Boolean*, *read only_*)
- '%C'*xx*`.F` - Counter *xx* full overflow (*Boolean*, *read only*)
- '%C'*xx*`.V` - Counter *xx* current value (*integer*, *read* or *write*)
- '%C'*xx*`.P` - Counter *xx* preset (*integer*, *read* or *write*)

#### 8.2.6.6 VERGLEICHEN

Für den arithmetischen Vergleich. Ist die Variable %XXX = zu dieser Zahl (oder der ausgewerteten Zahl)

The compare block will be true when comparison is true. You can use most math symbols:

- `+`, `-`, *, `/`, = (standard math symbols)
- < (less than), > (greater than), <= (less or equal), >= (greater or equal), <> (not equal)
- `(`, `)` separate into groups example `%IF1=2,&%IF2<5` in pseudo code translates to if %IF1 is equal to 2 and %IF2 is less than 5 then the comparison is true. Note the comma separating the two groups of comparisons.
- ^ (exponent), `%` (modulus), & (and), `|` (or),. -
- `ABS` (absolute), `MOY` (French for average), `AVG` (average)

Zum Beispiel ABS(%W2)=1, MOY(%W1,%W2)<3.

No spaces are allowed in the comparison equation. For example `%C0.V>%C0.P` is a valid comparison expression while `%C0.V > %CO.P` is not a valid expression.

There is a list of Variables down the page that can be used for reading from and writing to ladder objects. When a new compare block is opened be sure and delete the # symbol when you enter a compare.

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

For variable assignment, e.g. assign this number (or evaluated number) to this variable %xxx, there are two math functions MINI and MAXI that check a variable for maximum (0x80000000) and minimum values (0x07FFFFFF) (think signed values) and keeps them from going beyond.

When a new variable assignment block is opened be sure to delete the # symbol when you enter an assignment.

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**
When you assign a value to a variable with the variable assignment block the value is retained until you assign a new value using the variable assignment block. The last value assigned will be restored when LinuxCNC is started.

The following figure shows an Assignment and a Comparison Example. %QW0 is a S32out bit and %IW0 is a S32in bit. In this case the HAL pin `classicladder.0.s32out-00` will be set to a value of 5 and when the HAL pin `classicladder.0.s32in-00` is 0 the HAL pin `classicladder.0.out-00` will be set to True.

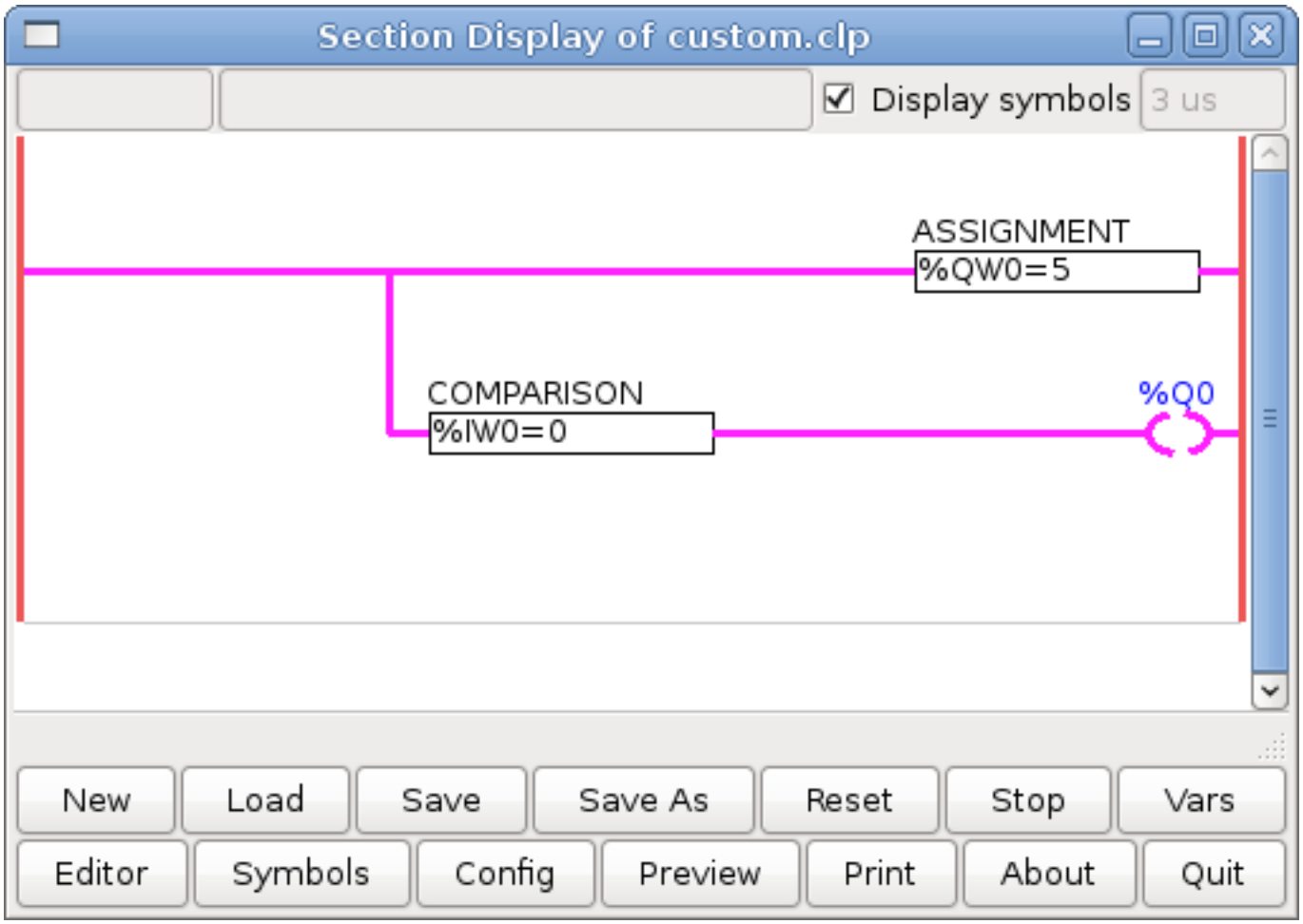



Abbildung 8.8: Beispiel für Zuordnen/Vergleichen von mit SPS

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* - A *goto* coil: When coil is energized then the ladder program jumps to a rung (in the CURRENT section) - jump points are designated by a rung label. (Add rung labels in the section display, top left label box.)
- *CALL COIL* - A *gosub* coil: When coil is energized then the program jumps to a subroutine section designated by a subroutine number - subroutines are designated SR0 to SR9 (designate them in the section manager).

---

**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.

If you look at the top left of the sections display window you will see a small label box and a longer comment box beside it. Now go to Editor→Modify then go back to the little box, type in a name.

Go ahead and add a comment in the comment section. This label name is the name of this rung only and is used by the JUMP COIL to identify where to go.

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.

Select a subroutine number (SR0 for example). An empty section will be displayed and you can build your subroutine.

Wenn Sie das getan haben, gehen Sie zurück zum Abschnittsmanager und klicken Sie auf Ihren Hauptabschnitt (Standardname prog1).

Jetzt können Sie eine CALL COIL in Ihr Programm einfügen. CALL COILs sind an der äußersten rechten Position im Strompfad zu platzieren.

Vergessen Sie nicht, die Beschriftung in die Nummer des Unterprogramms zu ändern, die Sie zuvor gewählt haben.

### 8.2.7 ClassicLadder Variablen

Diese Variablen werden in COMPARE oder OPERATE verwendet, um Informationen über Kontaktplanobjekte zu erhalten oder deren Spezifikationen zu ändern, z. B. um einen Zähler zu ändern oder um zu sehen, ob ein Timer fertig ist.

Liste der Variablen :

- `%B`*xxx* - Bit memory *xxx* (Boolean)
- `%W`*xxx* - Word memory *xxx* (32 bits signed integer)
- `%IW`*xxx* - Word memory *xxx* (S32 in pin)
- `%QW`*xxx* - Word memory *xxx* (S32 out pin)
- `%IF`*xx* - Word memory *xx* (Float in pin) (**converted to S32 in ClassicLadder**)
- `%QF`*xx* - Word memory *xx* (Float out pin) (**converted to S32 in ClassicLadder**)
- `%T _xx_.R` - Timer *xx* running (Boolean, user read only)
- `%T _xx_.D` - Timer *xx* done (Boolean, user read only)
- `%T _xx_.V` - Timer *xx* current value (integer, user read only)
- `%T _xx_.P` - Timer *xx* preset (integer)
- `%TM _xxx_.Q` - Timer *xxx* done (Boolean, read write)
- `%TM _xxx_.P` - Timer *xxx* preset (integer, read write)
- `%TM _xxx_.V` - Timer *xxx* value (integer, read write)
- `%M _xx_.R` - Monostable *xx* running (Boolean)
- `%M _xx_.V` - Monostable *xx* current value (integer, user read only)

- %M `__xx__.P` - Monostable *xx* preset (integer)
- %C `__xx__.D` - Counter *xx* done (Boolean, user read only)
- %C `__xx__.E` - Counter *xx* empty overflow (Boolean, user read only)
- %C `__xx__.F` - Counter *xx* full overflow (Boolean, user read only)
- %C `__xx__.V` - Counter *xx* current value (integer)
- %C `__xx__.P` - Counter *xx* preset (integer)
- %I*xxx* - Physical input *xxx* (Boolean) (HAL input bit)
- %Q*xxx* - Physical output *xxx* (Boolean) (HAL output bit)
- %X*xxx* - Activity of step *xxx* (sequential language)
- %X `__xxx__.V` - Time of activity in seconds of step *xxx* (sequential language)
- %E*xx* - Errors (Boolean, read write(will be overwritten))
- *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**
This is probably the least used and most poorly understood feature of ClassicLadder. Sequential programming is used to make sure a series of ladder events always happen in a prescribed order. Sequential programs do not work alone. There is always a ladder program as well that controls the variables. Here are the basic rules governing sequential programs:

- Rule 1 : Initial situation - The initial situation is characterized by the initial steps which are by definition in the active state at the beginning of the operation. There shall be at least one initial step.
- Rule 2 : R2, Clearing of a transition - A transition is either enabled or disabled. It is said to be enabled when all immediately preceding steps linked to its corresponding transition symbol are active, otherwise it is disabled. A transition cannot be cleared unless it is enabled, and its associated transition condition is true.
- 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.

This is the SEQUENTIAL editor window. (Starting from the top left):
Selector arrow, Eraser
Ordinary step, Initial (Starting) step
Transition, Step and Transition
Transition Link-Downside, Transition Link-Upside
Pass-through Link-Downside, Pass-through Link-Upside Jump

Link, Comment Box

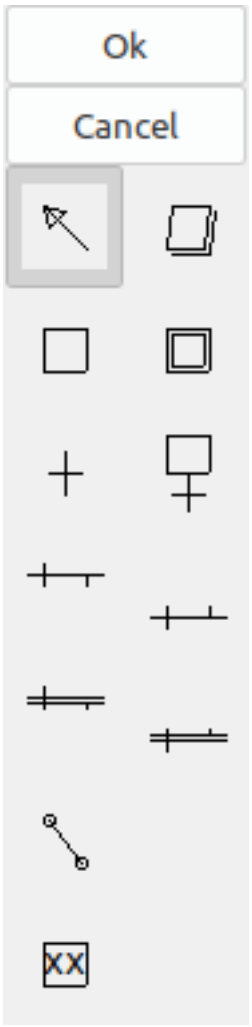



Abbildung 8.11: Fenster des Sequenzeditors

- *ORDINARY STEP* - hat für jeden eine eindeutige Nummer
- *STARTING STEP* - a sequential program must have one. This is where the program will start.
- *TRANSITION* - shows the variable that must be true for control to pass through to the next step.
- *STEP AND TRANSITION* - combined for convenience
- *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

To use links, you must have steps already placed. Select the type of link, then select the two steps or transactions one at a time. It takes practice!

With sequential programming: The variable `%X __xxx__ (e.g., %X5)` is used to see if a step is active. The variable `%X __xxx__.V` (e.g., `%X5.V`) is used to see how long the step has been active. The %X and %X.v variables are use in LADDER logic. The variables assigned to the transitions (e.g., %B) control whether the logic will pass to the next step. After a step has become active the transition variable that caused it to become active has no control of it anymore. The last step has to JUMP LINK back only to the beginning step.

### 8.2.9 Modbus

Zu beachtende Punkte:

- Modbus is a non-realtime program so it might have latency issues on a heavily laden computer.
- 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.

To get MODBUS to initialize you must specify that when loading the ClassicLadder non-realtime program.

**Laden von Modbus**

```
loadusr -w classicladder --modmaster myprogram.clp
```

The `-w` makes HAL wait until you close ClassicLadder before closing realtime session. ClassicLadder also loads a TCP modbus slave if you add `--modserver` on command line.

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

If you do not specify a *--modmaster* when loading the ClassicLadder non-realtime program this page will not be displayed.

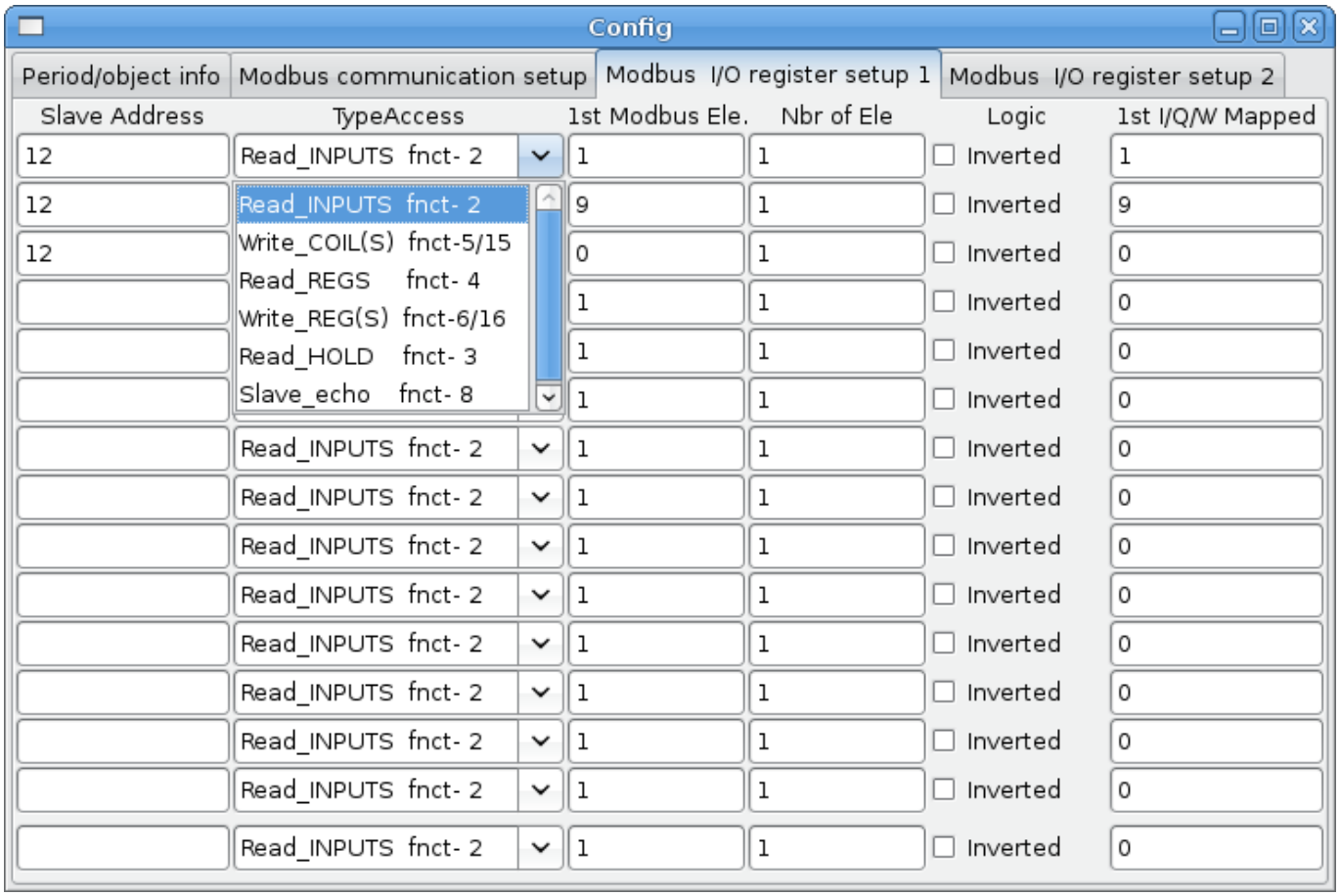



Abbildung 8.12: Modbus I/O-Konfiguration

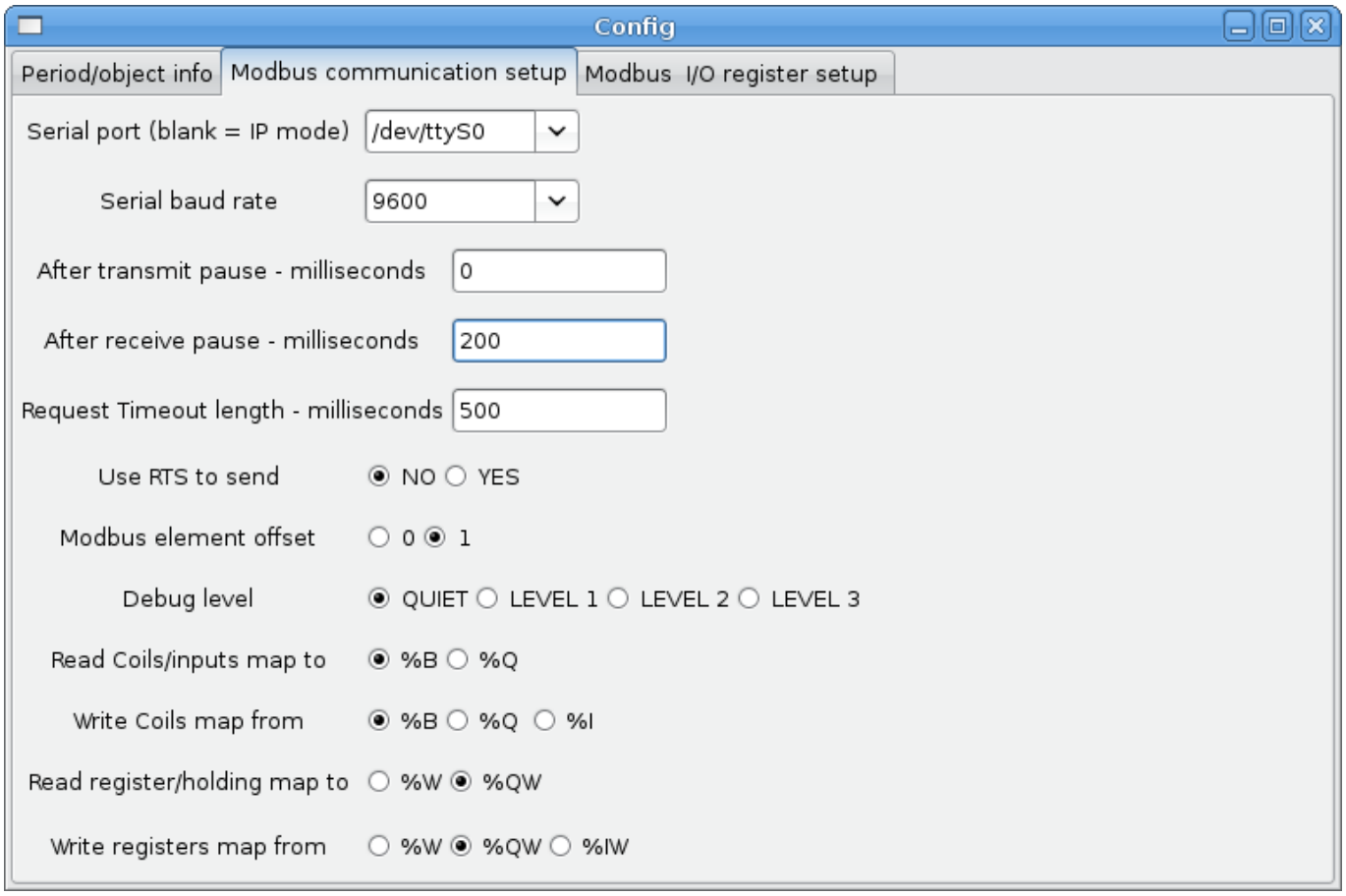



Abbildung 8.13: Modbus-Kommunikationskonfiguration

- *SERIAL PORT* - For IP blank. For serial the location/name of serial driver, e.g., /dev/ttyS0 ( or /dev/ttyUSB0 for a USB-to-serial converter).
- *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 ADDRESS* - For serial the slaves ID number usually settable on the slave device (usually 1-256). For IP the slave IP address plus optionally the port number.
- *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).

The first line is set up for 8 input bits starting at the first register number (register 1). So register numbers 1-8 are mapped onto ClassicLadder's %B variables starting at %B1 and ending at %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.

The third line is set to write 2 registers (16 bits each) starting at the $0^{th}$ register number (register 0), so register numbers 0-1 are mapped onto ClassicLadder's %W variables starting at %W0 ending at %W1.

It's easy to make an off-by-one error as sometimes the modbus elements are referenced starting at one rather then 0 (actually by the standard that is the way it's supposed to be!). You can use the modbus element offset radio button to help with this.

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).

To use the echo function select the echo function and add the slave number you wish to test. You don't need to specify any variables.

The number 257 will be sent to the slave number you specified and the slave should send it back. You will need to have ClassicLadder running in a terminal to see the message.

### 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.
- Baud rate must be the same for slave and master. ClassicLadder can only have one baud rate so all the slaves must be set to the same rate.
- 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 can use distributed inputs/outputs on modules using the Modbus protocol (”master”: polling 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.
- 2 products have been used for tests: a Modbus/TCP one (Adam-6051, https://www.advantech.com) and a serial Modbus/RTU one (http://www.ipac.ws).
- Siehe Beispiele: adam-6051 und modbus_rtu_serial.
- Web links: https://www.modbus.org and this interesting one: 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.

https://www.modbus.org/

#### 8.2.10.2 Kommunikationsfehler

If there is a communication error, a warning window will pop up (if the GUI is running) and %E0 will be true. Modbus will continue to try to communicate. The %E0 could be used to make a decision based on the error. A timer could be used to stop the machine if timed out, etc.

### 8.2.11 Fehlersuche bei Modbus-Problemen

A good reference for the protocol: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. If you run linuxcnc/classicladder from a terminal, it will print the Modbus commands and slave responses.

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.

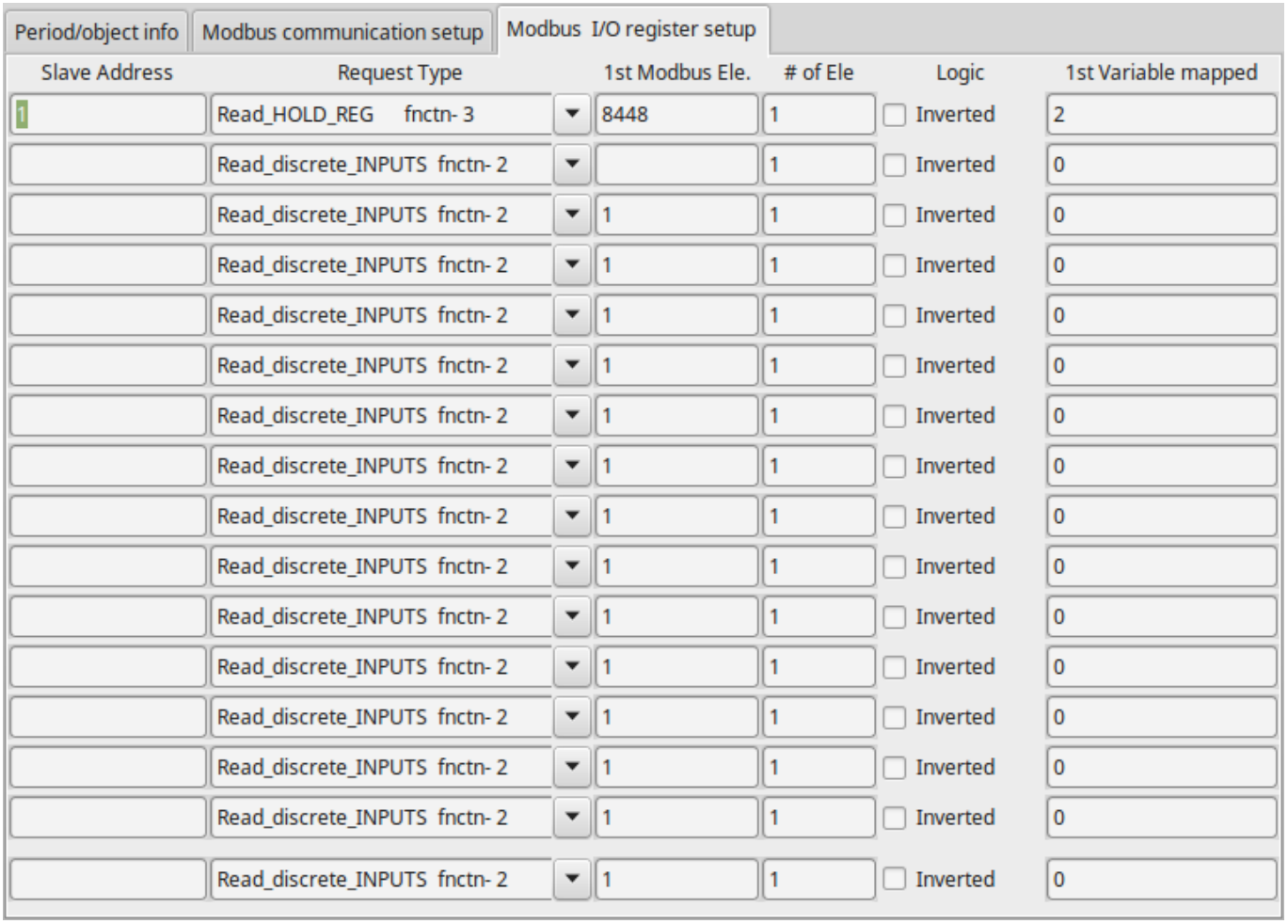



Abbildung 8.14: Modbus I/O-Register-Setup

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.

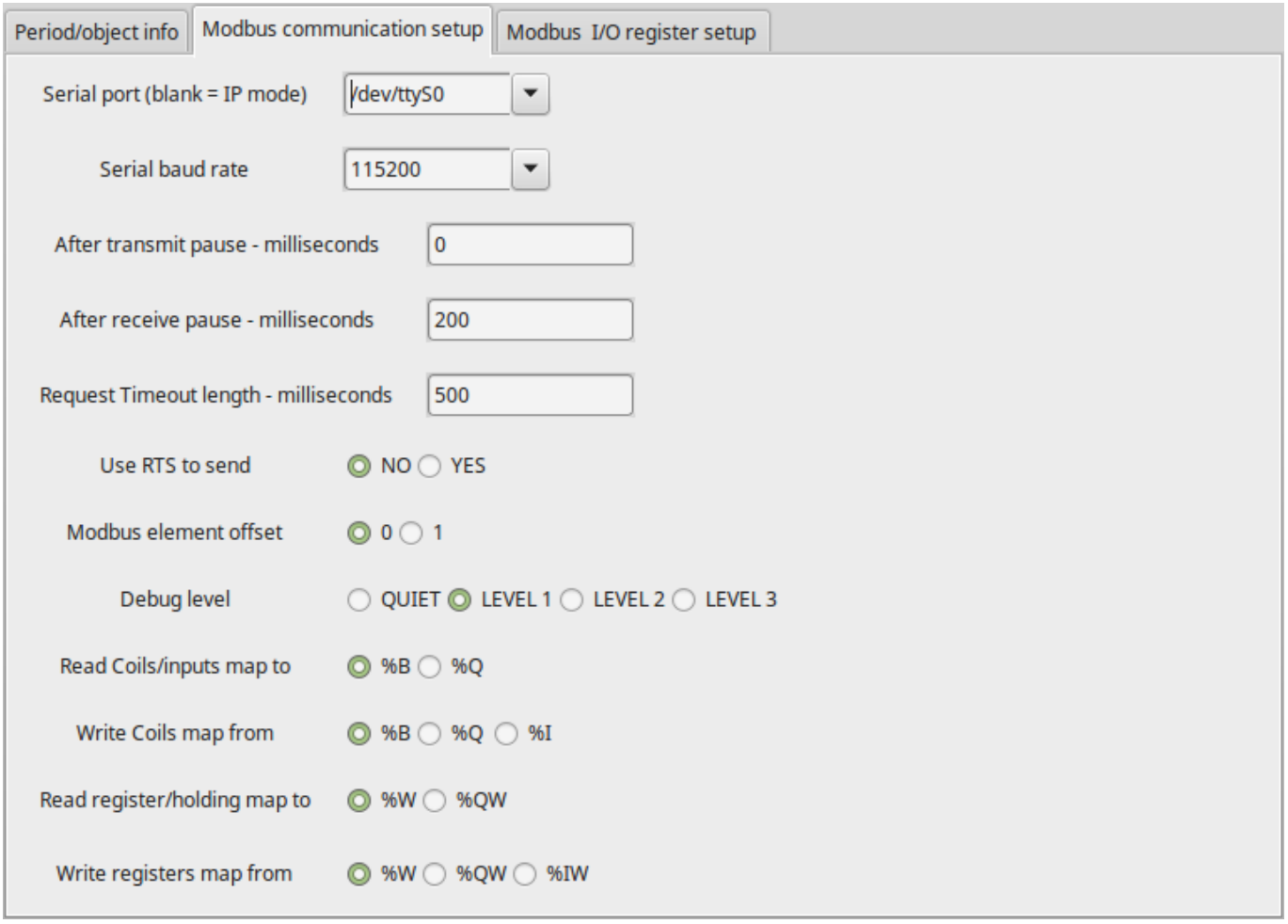



Abbildung 8.15: Einrichtung der Modbus-Kommunikation

#### 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.
- When loading a ladder program it will load Modbus info but will not tell ClassicLadder to initialize Modbus. You must initialize Modbus when you first load the GUI by adding *--modmaster*.
- If the section manager is placed on top of the section display, across the scroll bar and exit is clicked the non-realtime program crashes.
- 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 this section we will cover the steps needed to add ClassicLadder to a StepConf Wizard generated config. On the advanced Configuration Options page of StepConf Wizard check off ”Include ClassicLadder PLC”.

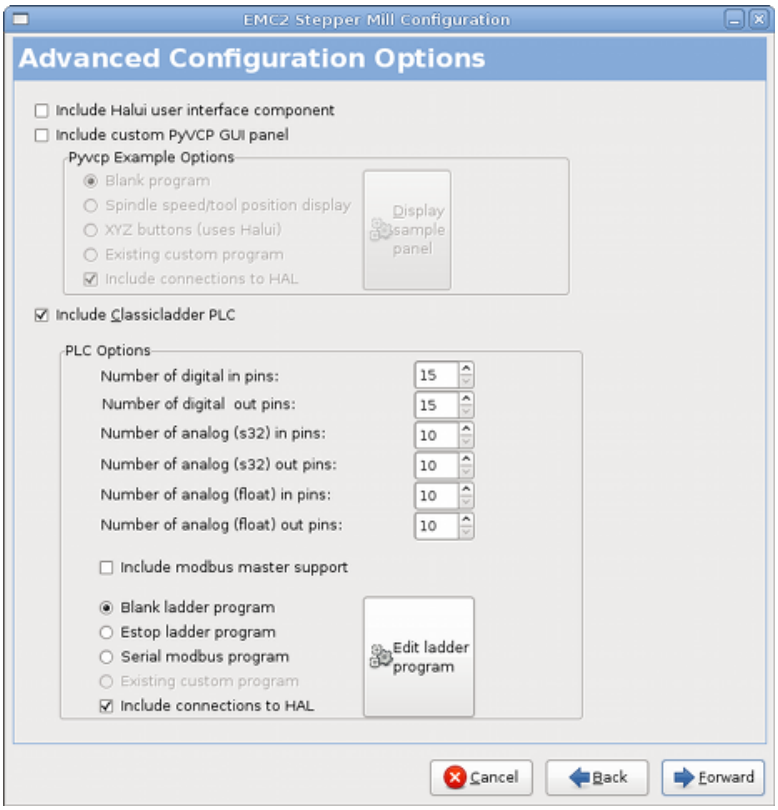



Abbildung 8.16: StepConf ClassicLadder

#### 8.2.12.1 Hinzufügen der Module

If you used the StepConf Wizard to add ClassicLadder you can skip this step.

To manually add ClassicLadder you must first add the modules. This is done by adding a couple of lines to the custom.hal file.

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

Now start up your config and select ”File/Ladder Editor” to open up the ClassicLadder GUI. You should see a blank Section Display and Sections Manager window as shown above. In the Section Display window open the Editor. In the Editor window select Modify. Now a Properties window pops up and the Section Display shows a grid. The grid is one rung of ladder. The rung can contain branches. A simple rung has one input, a connector line and one output. A rung can have up to six horizontal branches. While it is possible to have more than one circuit in a run the results are not predictable.

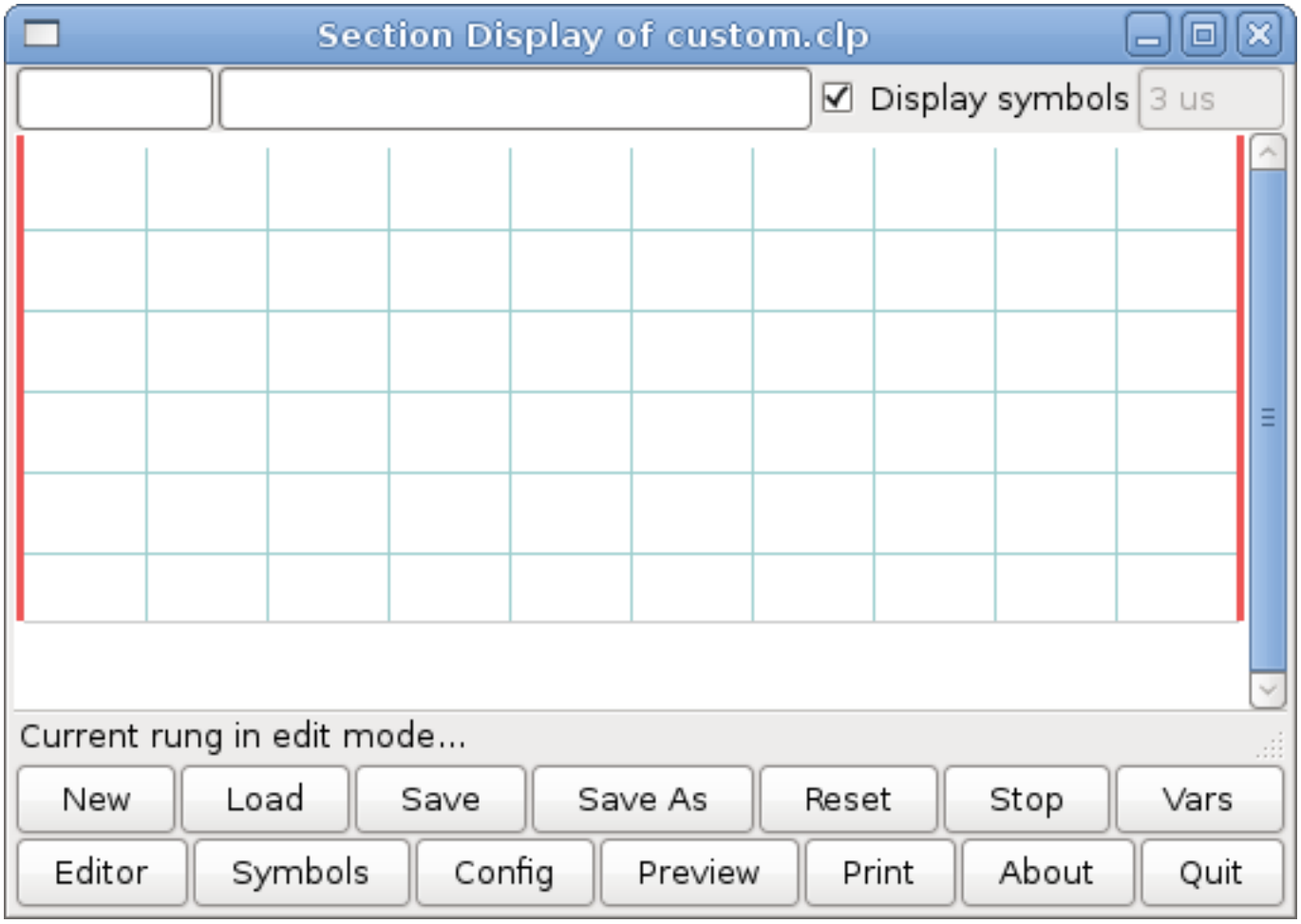



Abbildung 8.17: Abschnitt Display mit Grid

Now click on the N.O. input in the Editor Window.

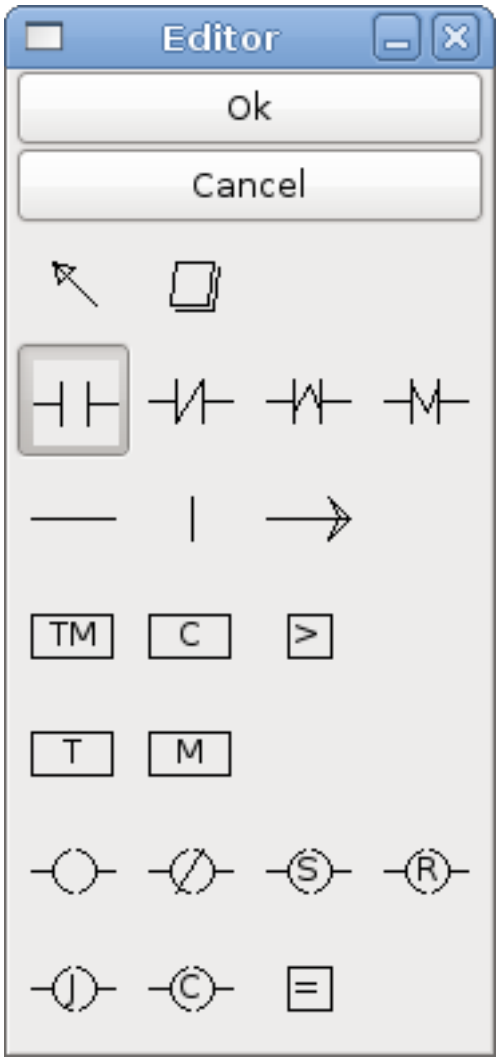



Abbildung 8.18: Editor-Fenster

Klicken Sie nun in das obere linke Raster, um den N.O.-Eingang in der Leiter zu platzieren.

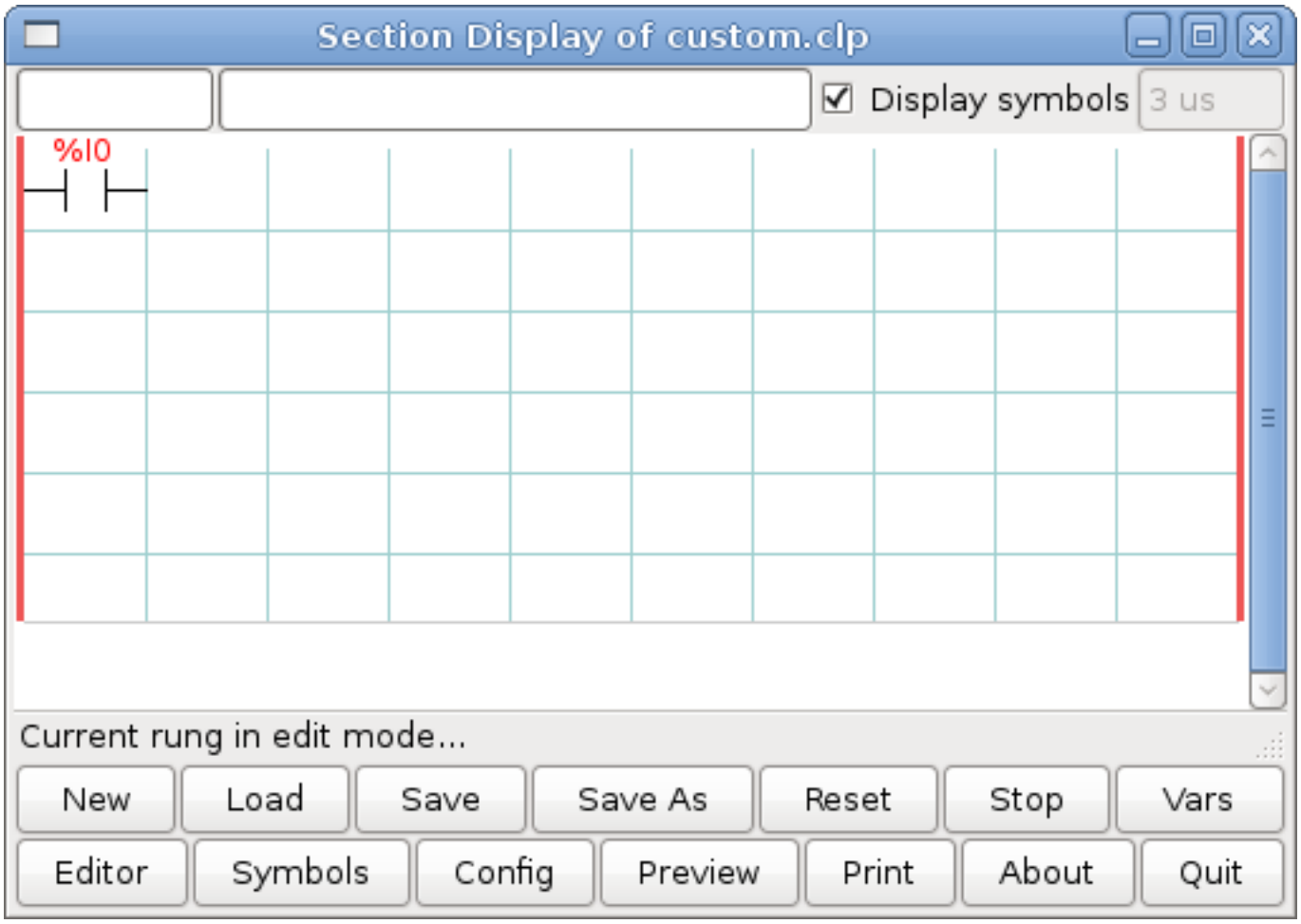



Abbildung 8.19: Abschnitt Display mit Input

Repeat the above steps to add a N.O. output to the upper right grid and use the Horizontal Connection to connect the two. It should look like the following. If not, use the Eraser to remove unwanted sections.

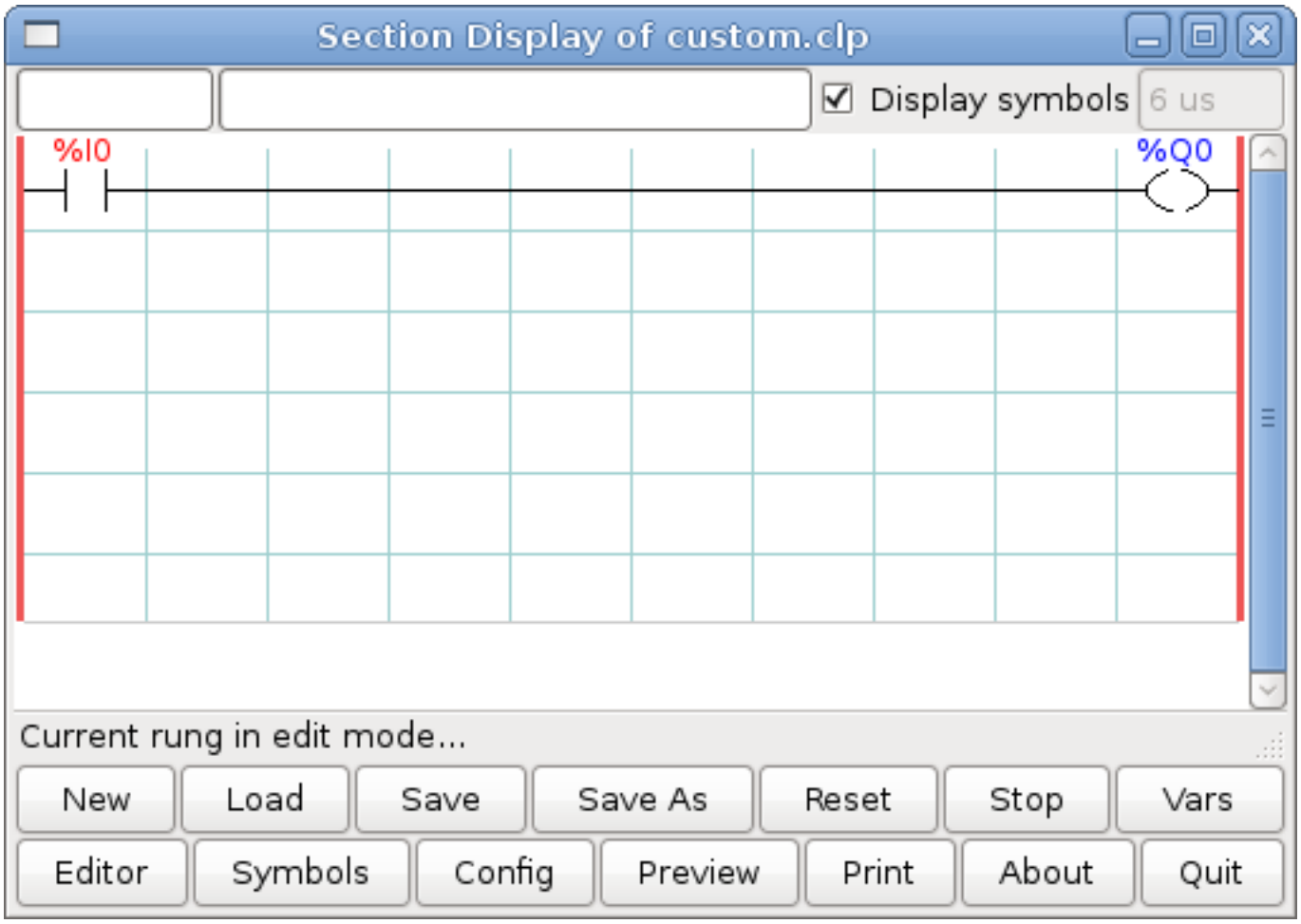



Abbildung 8.20: Abschnittsanzeige mit Sprosse

Klicken Sie nun im Editor-Fenster auf die Schaltfläche OK. Jetzt sollte Ihre Abschnittsanzeige wie folgt aussehen:

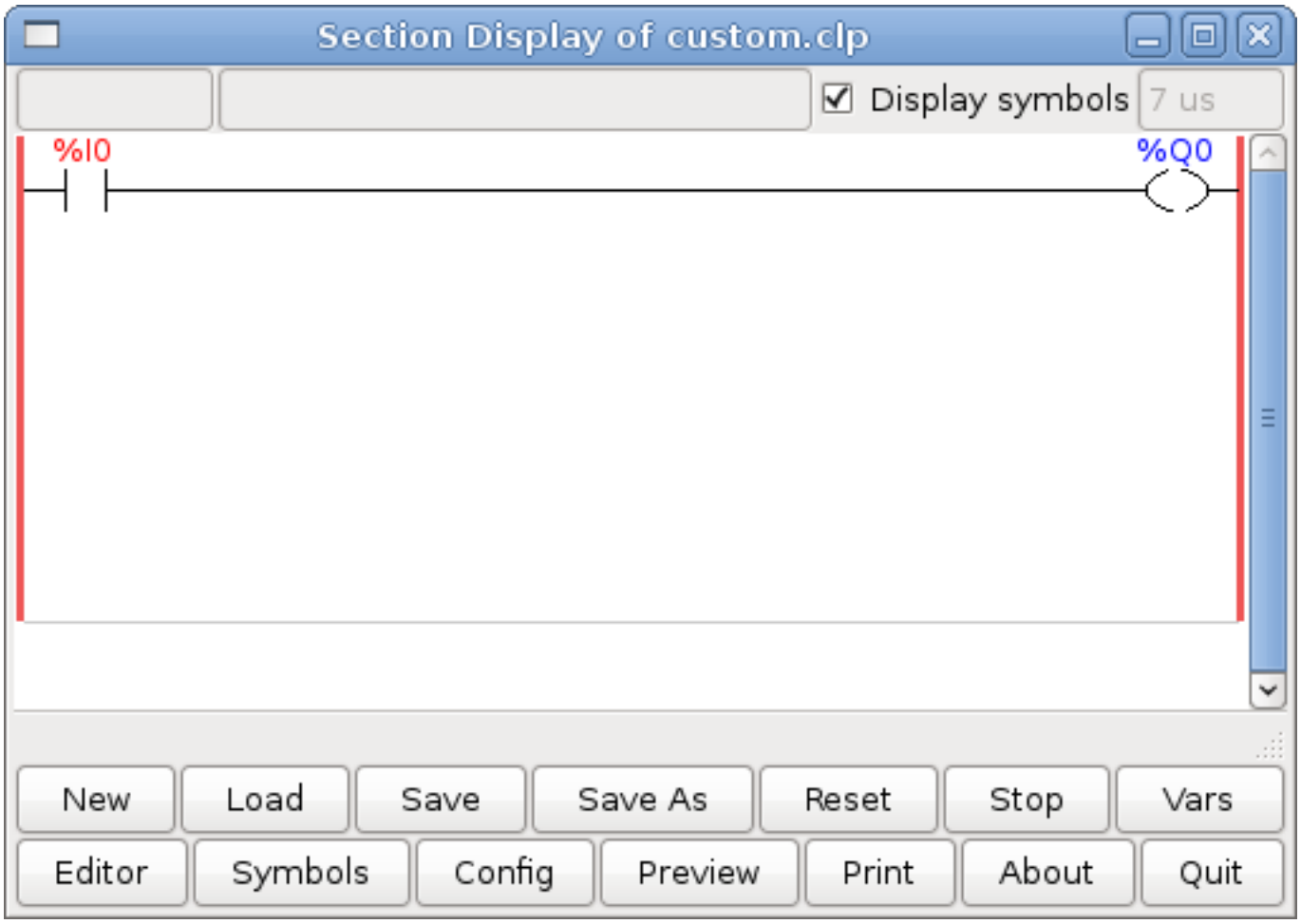



Abbildung 8.21: Abschnitt Anzeige Beendet

To save the new file select *Save As* and give it a name. The .clp extension will be added automatically. It should default to the running config directory as the place to save it.

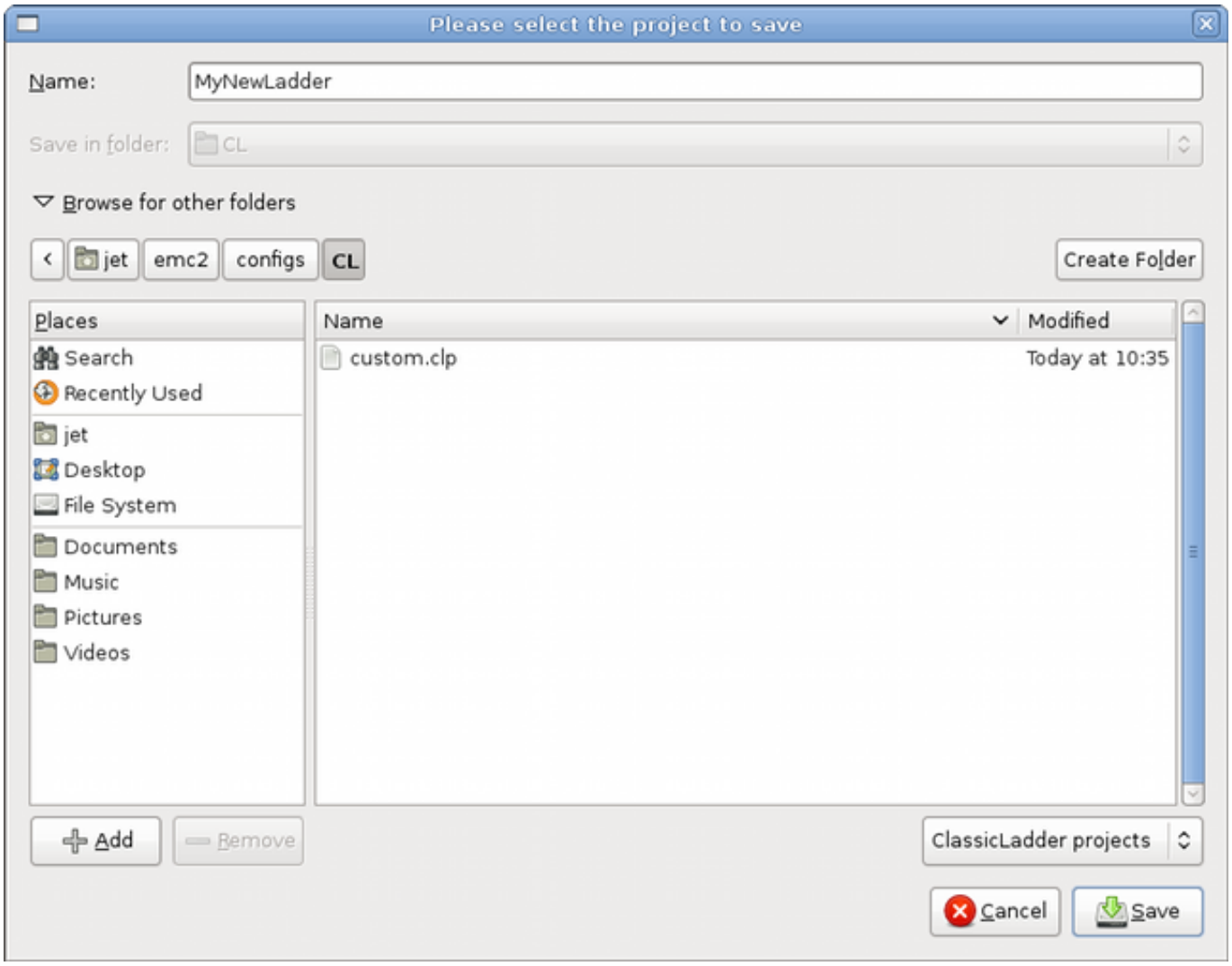



Abbildung 8.22: Speichern unter Dialog

Again if you used the StepConf Wizard to add ClassicLadder you can skip this step.

To manually add a ladder you need to add add a line to your custom.hal file that will load your ladder file. Close your LinuxCNC session and add this line to your custom.hal file.

```
loadusr -w classicladder --nogui MyLadder.clp
```

Now if you start up your LinuxCNC config your ladder program will be running as well. If you select ”File/Ladder Editor”, the program you created will show up in the Section Display window.

## 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.

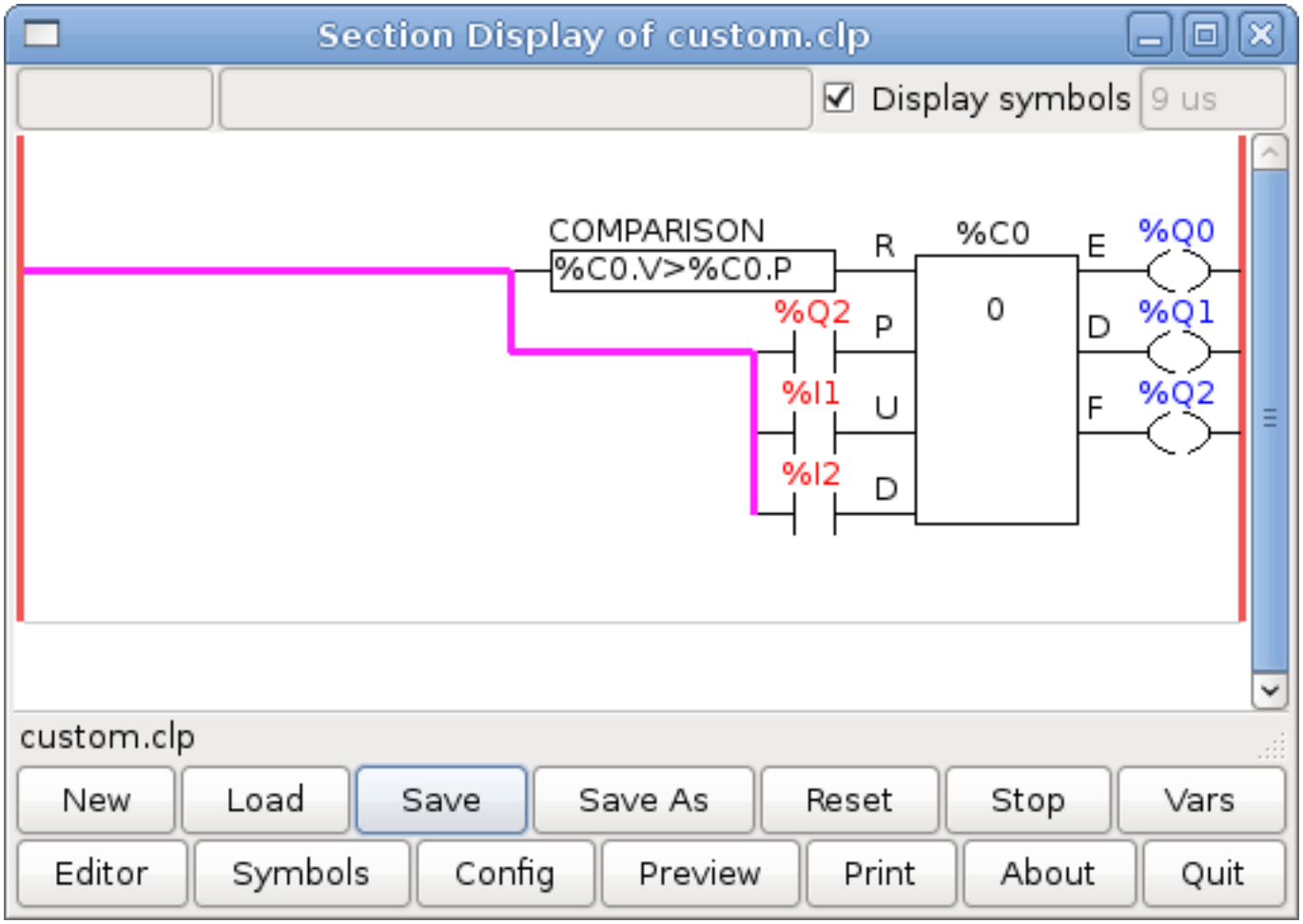



Abbildung 8.23: Umlaufender (engl. wrapping) Zähler

### 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.

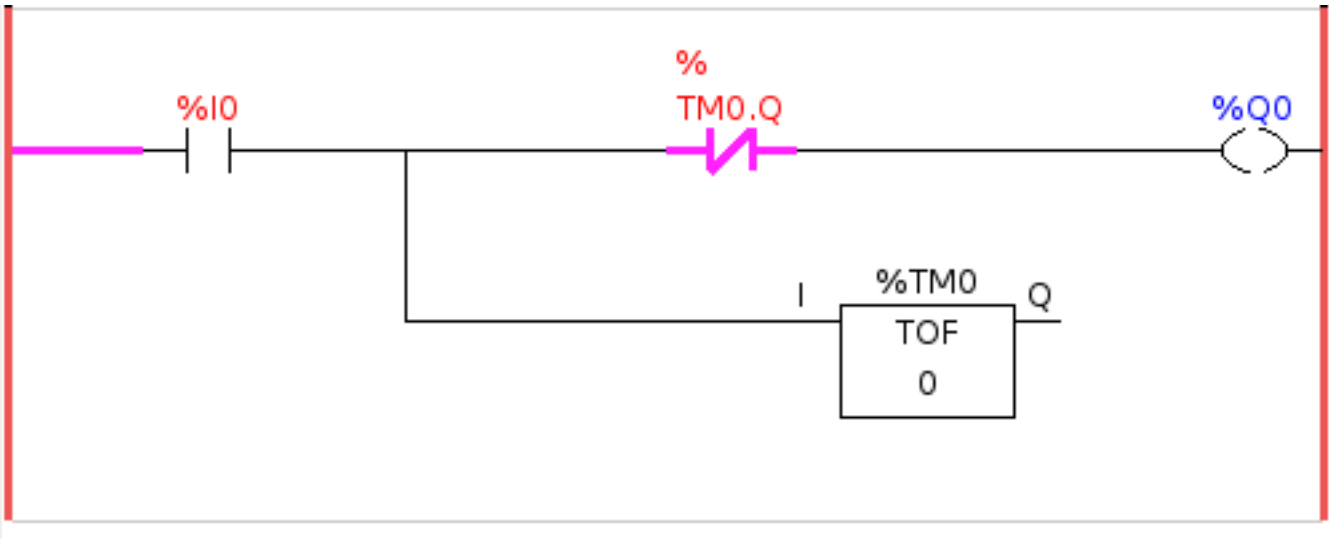



Abbildung 8.24: Extra-Impuls ablehnen

### 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.

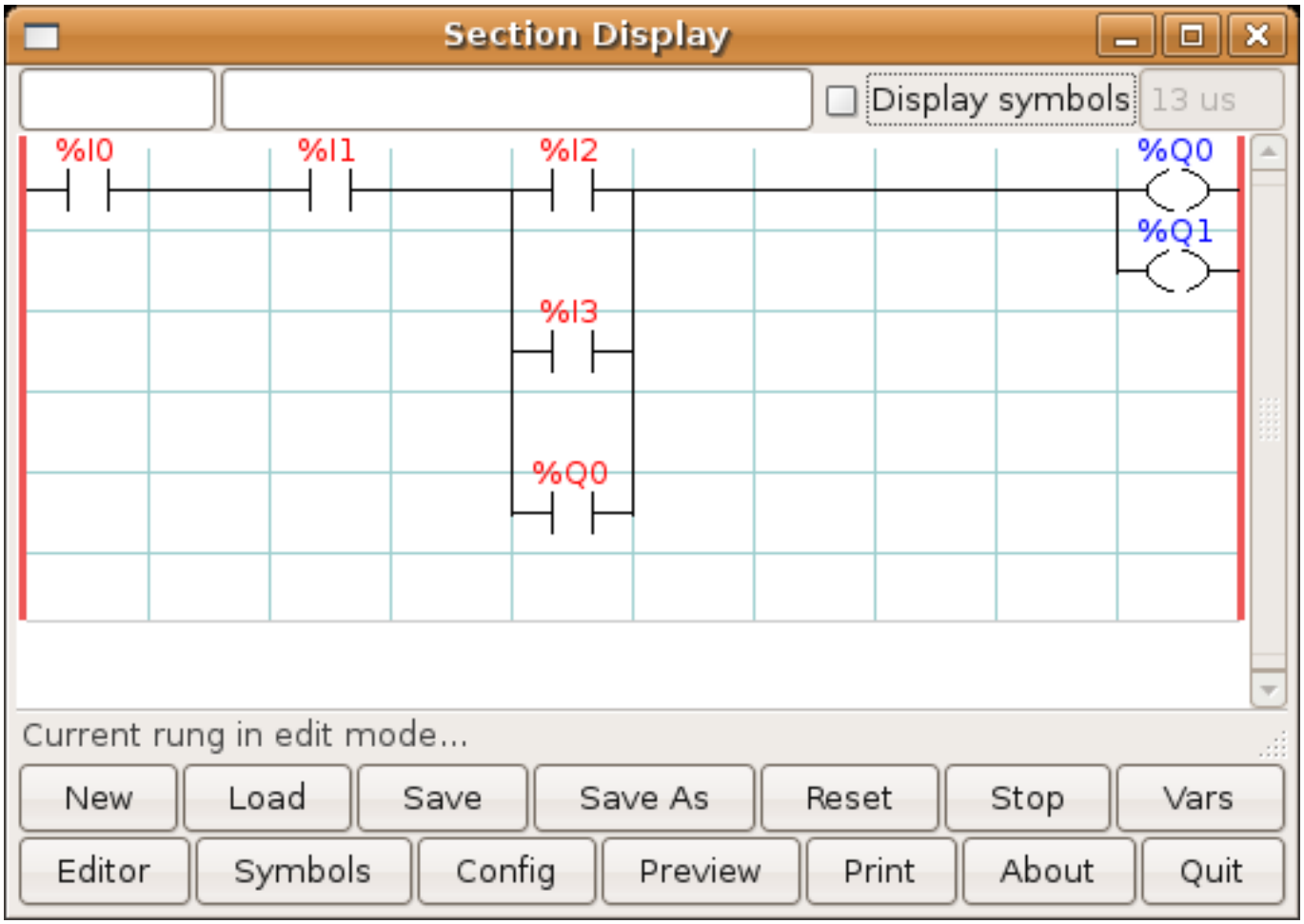



Abbildung 8.25: Anzeige des Notaus-Bereichs

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.

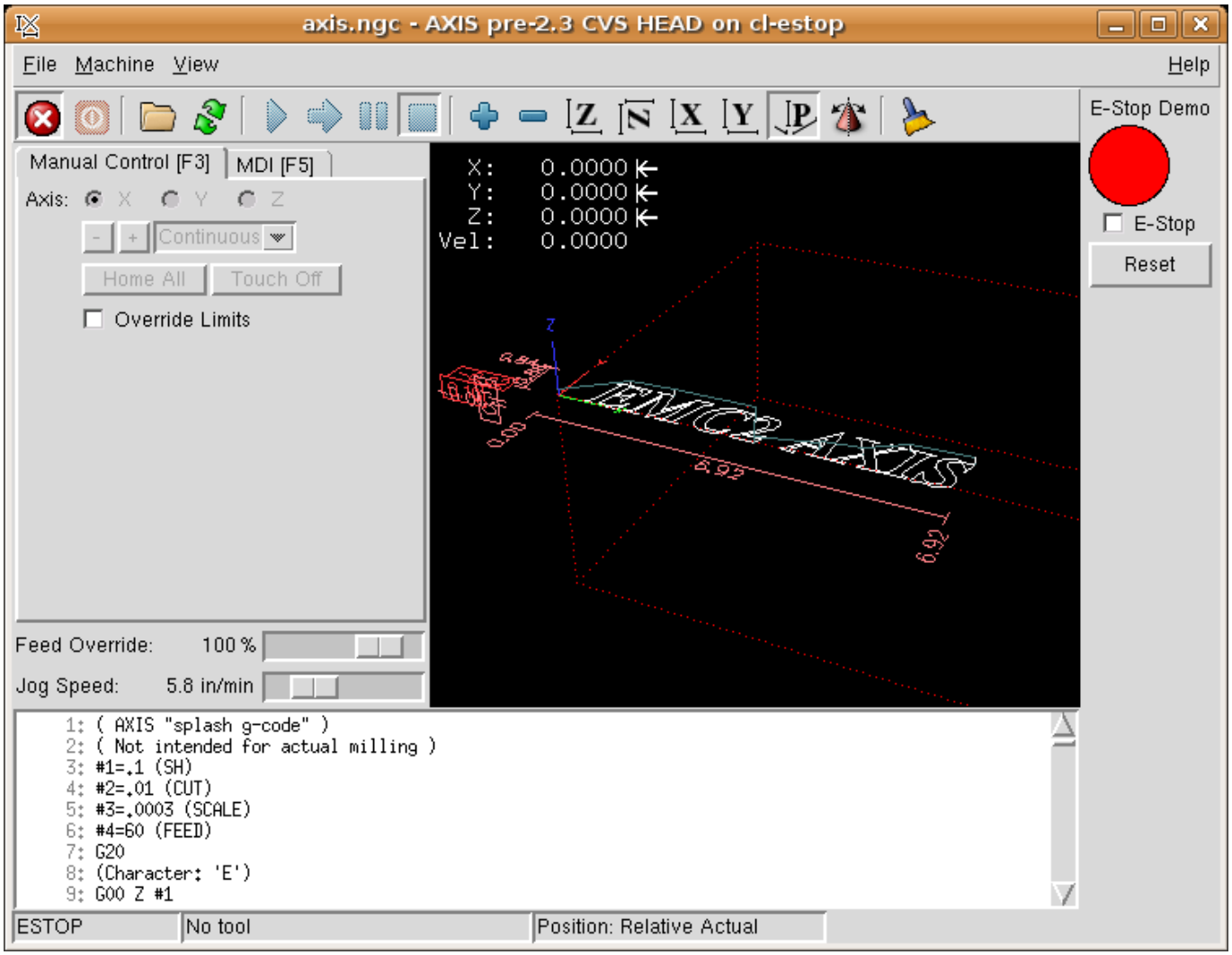



Abbildung 8.26: AXIS Notaus

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.

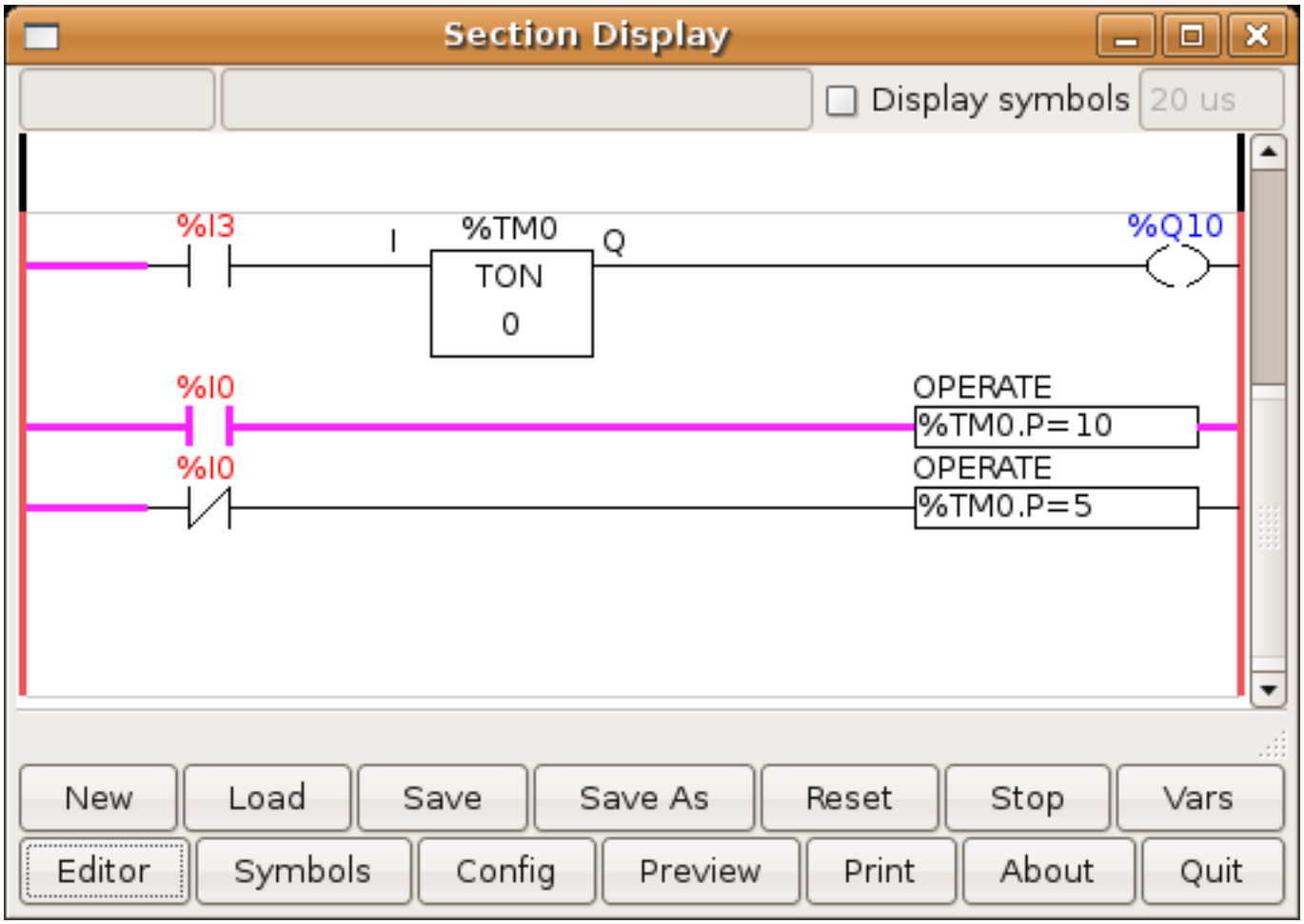



Abbildung 8.27: Beispiel für Timer/Bedienung

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 Kinematik

### 9.1.1 Einführung

When we talk about CNC machines, we usually think about machines that are commanded to move to certain locations and perform various tasks. In order to have an unified view of the machine space, and to make it fit the human point of view over 3D space, most of the machines (if not all) use a common coordinate system called the Cartesian Coordinate System.

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.].

When we talk about a G-code program (RS274/NGC) we talk about a number of commands (G0, G1, etc.) which have positions as parameters (X- Y- Z-). These positions refer exactly to Cartesian positions. Part of the LinuxCNC motion controller is responsible for translating those positions into positions which correspond to the machine kinematics [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] Kinematics: a two way function to transform from Cartesian space to joint space.

### 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.

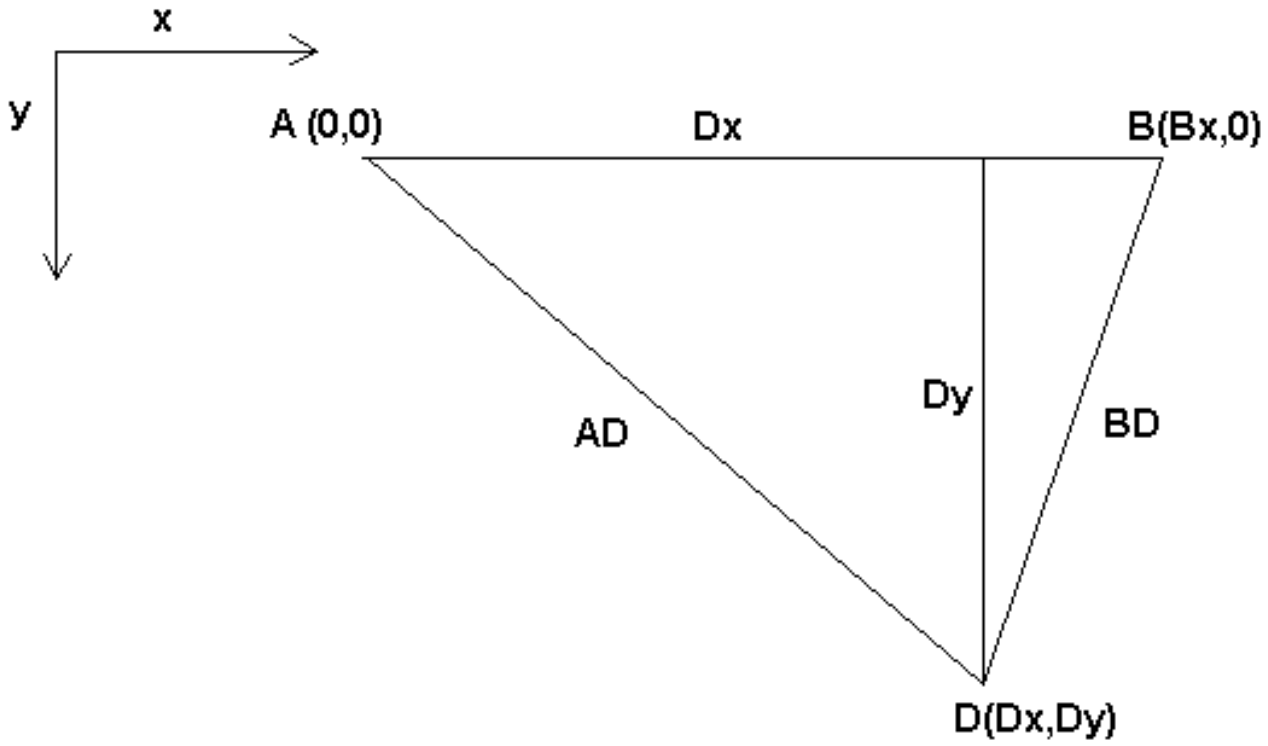



Abbildung 9.1: Zweibein-Einrichtung

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 Setting up ”modified” Denavit-Hartenberg (DH) parameters for *genserkins*

### 9.2.1 Vorspiel

LinuxCNC supports a number of kinematics modules including one that supports a generalized set of serial kinematics commonly specified via Denavit-Hartenberg parameters.

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).

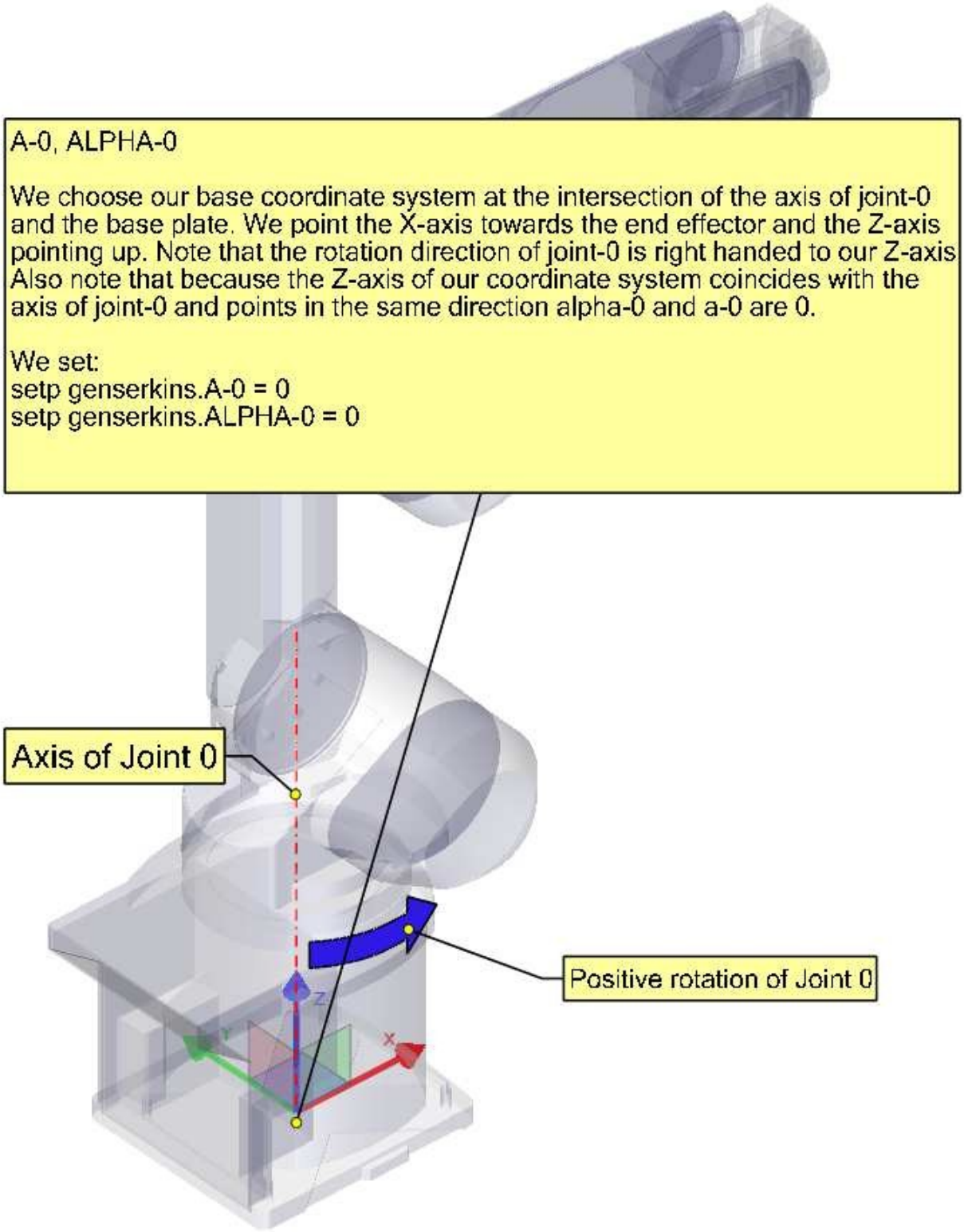
Und so sitzt der TCP-CS des Endeffektors in der Mitte des Handflansches.

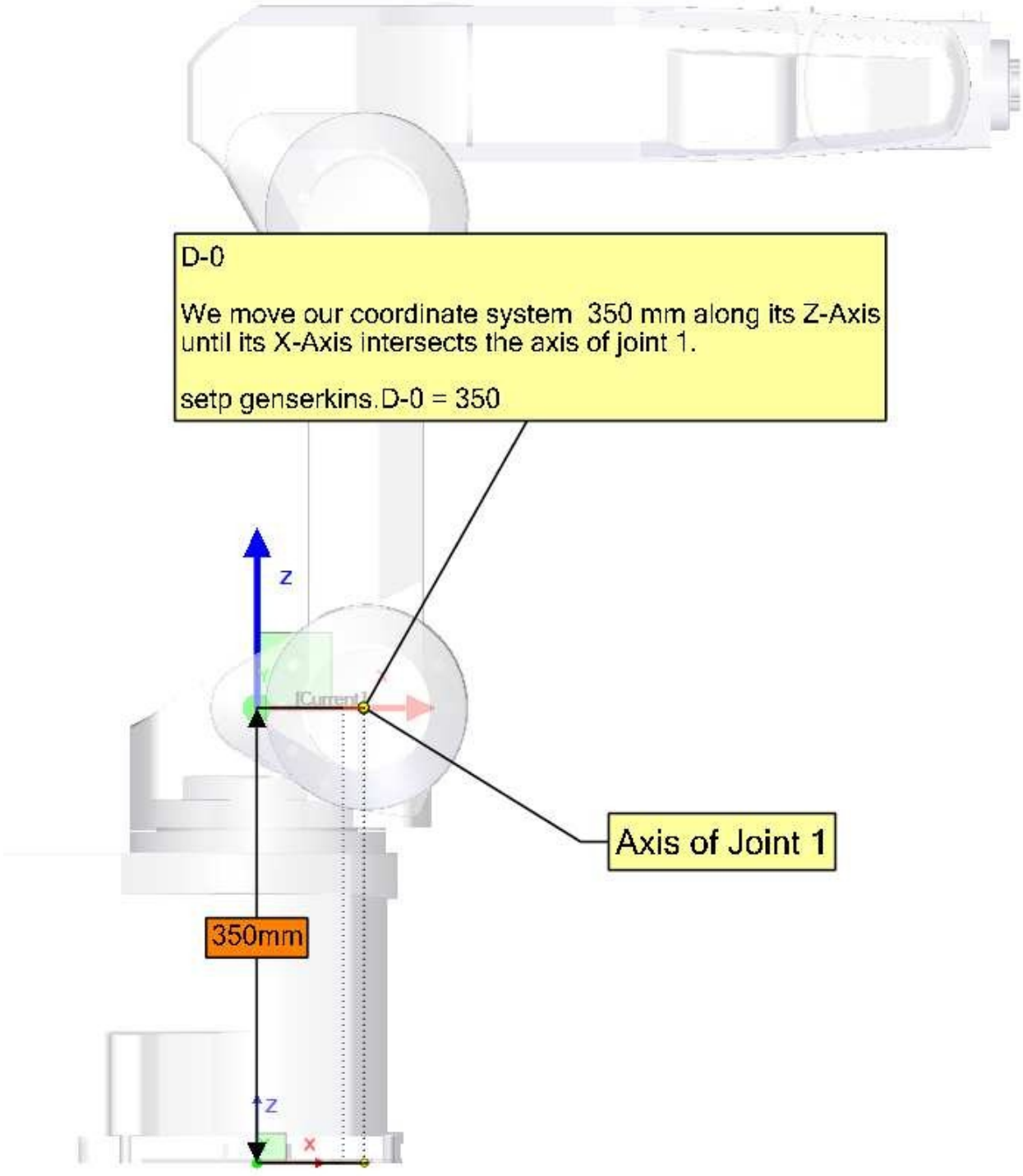
### 9.2.7 Sonderfälle

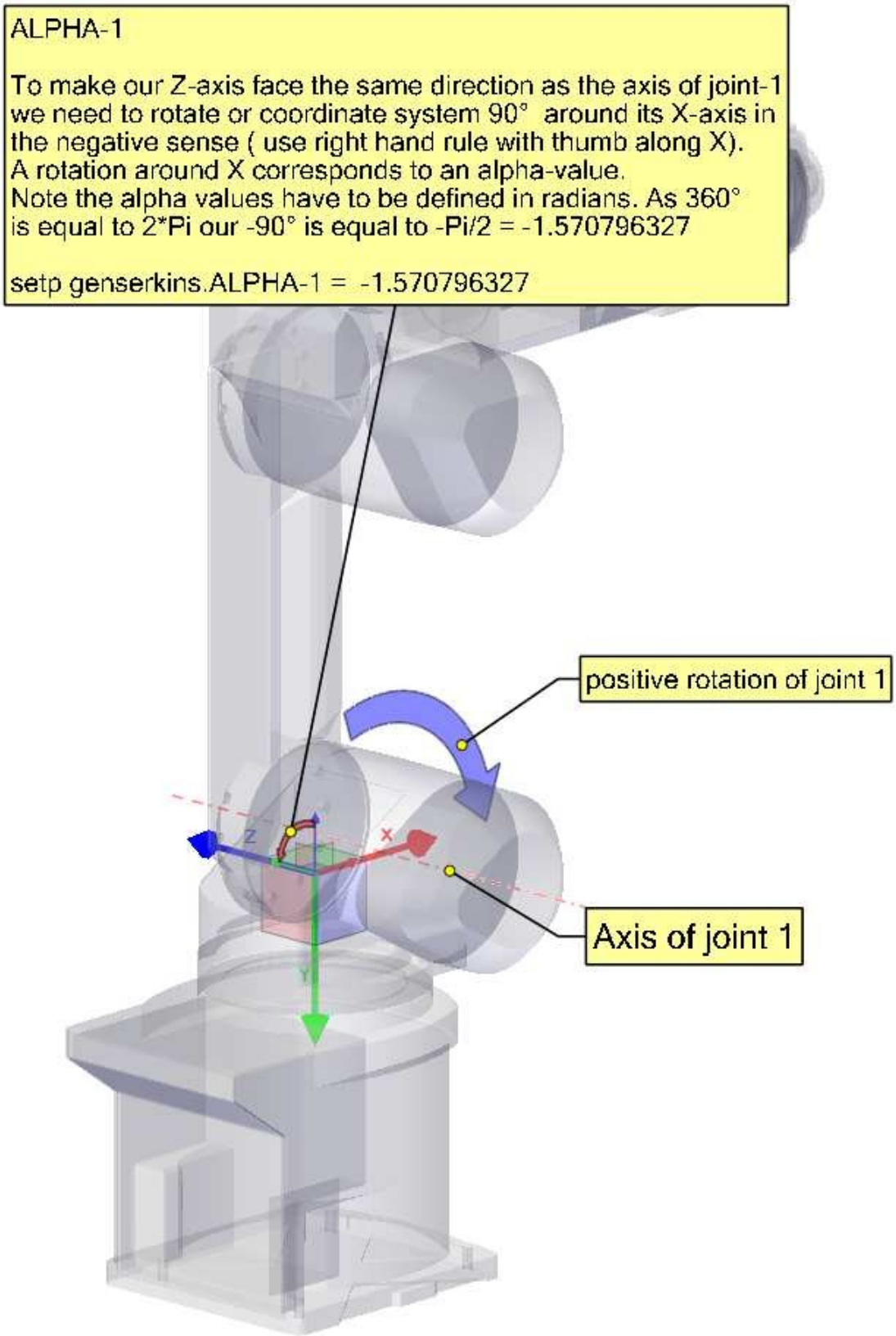
Wenn die nächste Gelenkachse parallel zur letzten ist, kann man den Wert für den d-Parameter beliebig wählen, aber es hat keinen Sinn, ihn anders als 0 zu setzen.

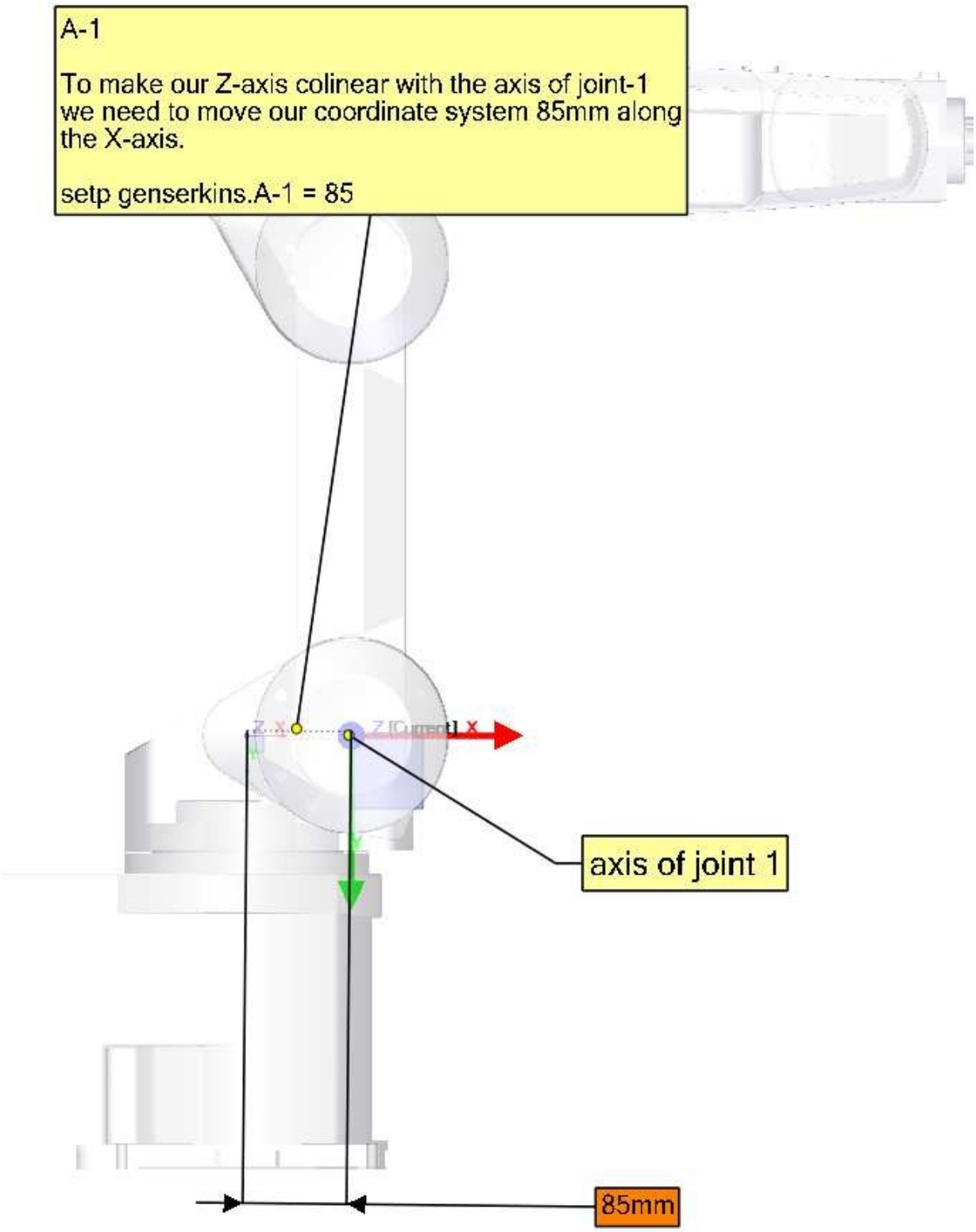
### 9.2.8 Detailliertes Beispiel (RV-6SL)

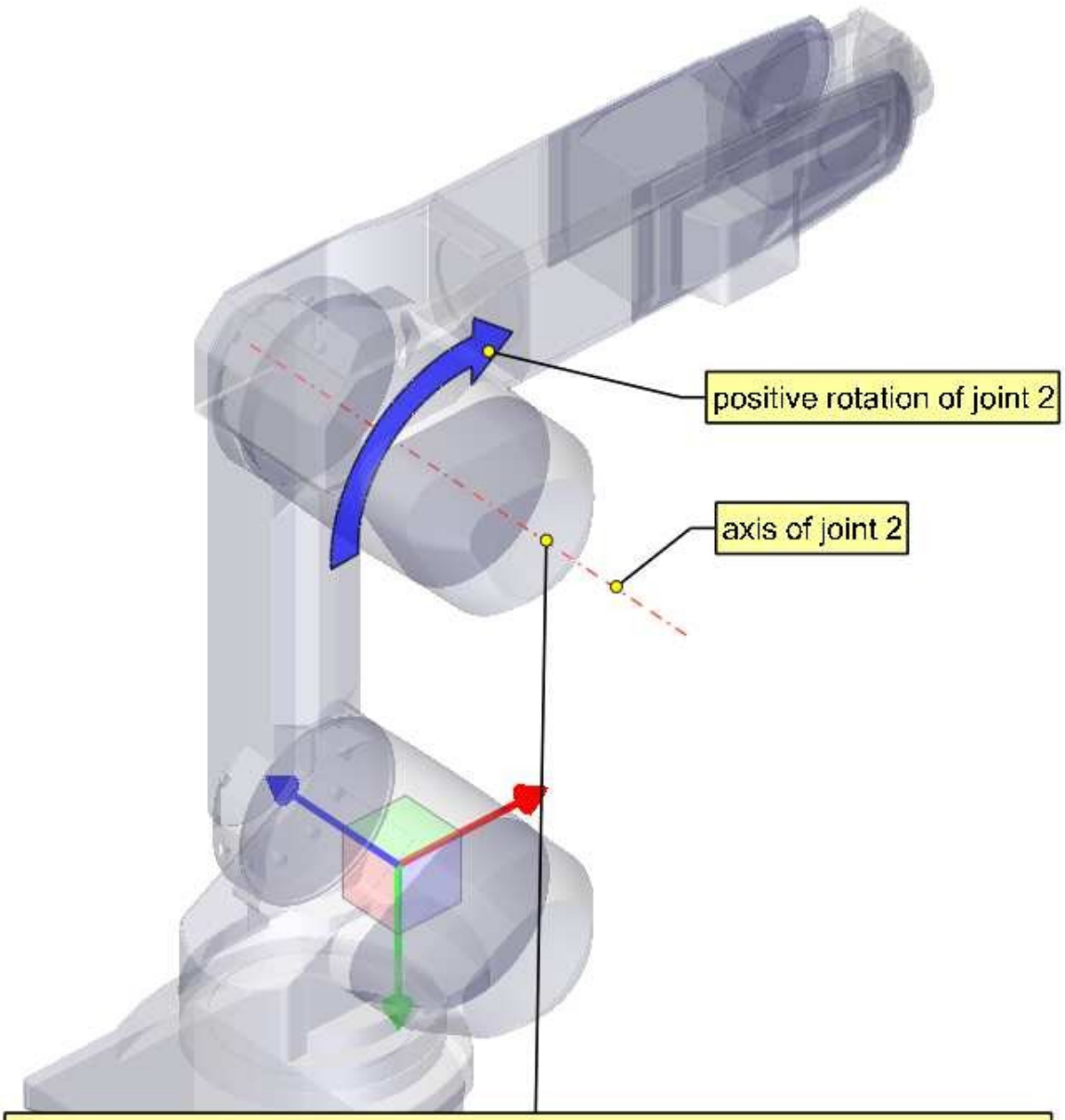
Im Folgenden wird eine Methode beschrieben, wie man die erforderlichen ”modifizierten DH-Parameter” für einen Mitsubishi RV-6SDL ableitet und wie man die Parameter in der HAL-Datei einstellt, um sie mit der ”genserkins”-Kinematik in LinuxCNC zu verwenden. Die erforderlichen Abmessungen werden am besten 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

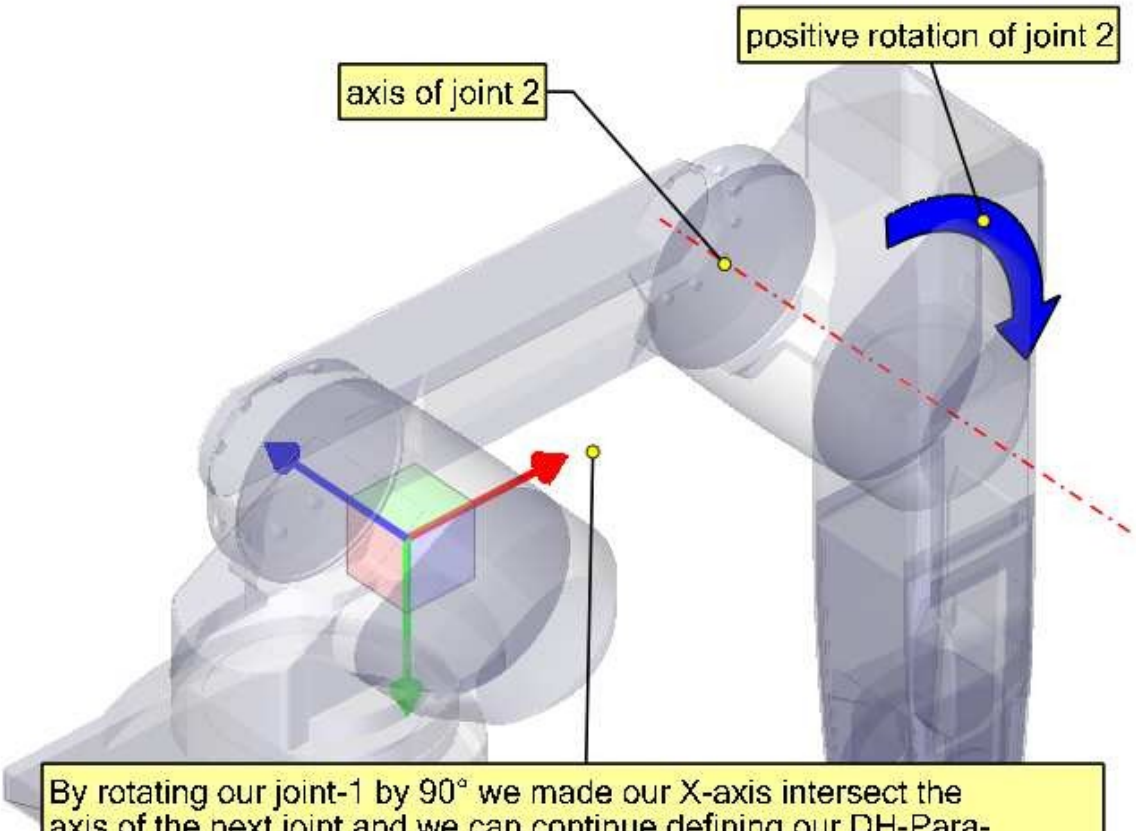

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


ALPHA-1
To make our Z-axis face the same direction as the axis of joint-1 we need to rotate or coordinate system 90° around its X-axis in the negative sense ( use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327
setp genserkins.ALPHA-1 = -1.570796327
positive rotation of joint 1
Axis of joint 1
Z
X
Y


A-1
To make our Z-axis colinear with the axis of joint-1 we need to move our coordinate system 85mm along the X-axis.
setp genserkins.A-1 = 85
axis of joint 1
85mm




Note:
In order to make our X-axis intersect the axis of joint-2 we would need to rotate our coordinate system around its Z-axis. To do this we could, in theory, define theta-1 equal to -90°.
However genserkins does not allow the definition of theta values. In genserkins.c we see that the theta values for all joints are set to 0.
Now theta of course is the rotation of the joint itself and so is variable in an angular joint. Theta values are only used to define the home pose of a robot in the way of an offset.
So if we could define theta-1 equal to -90° we could define joint-1 to be oriented this way for 0°. Since we cannot define it we need to rotate joint-1 in a way so that our X-axis intersects with the axis of the next joint.




By rotating our joint-1 by 90° we made our X-axis intersect the axis of the next joint and we can continue defining our DH-Parameters.

D-1

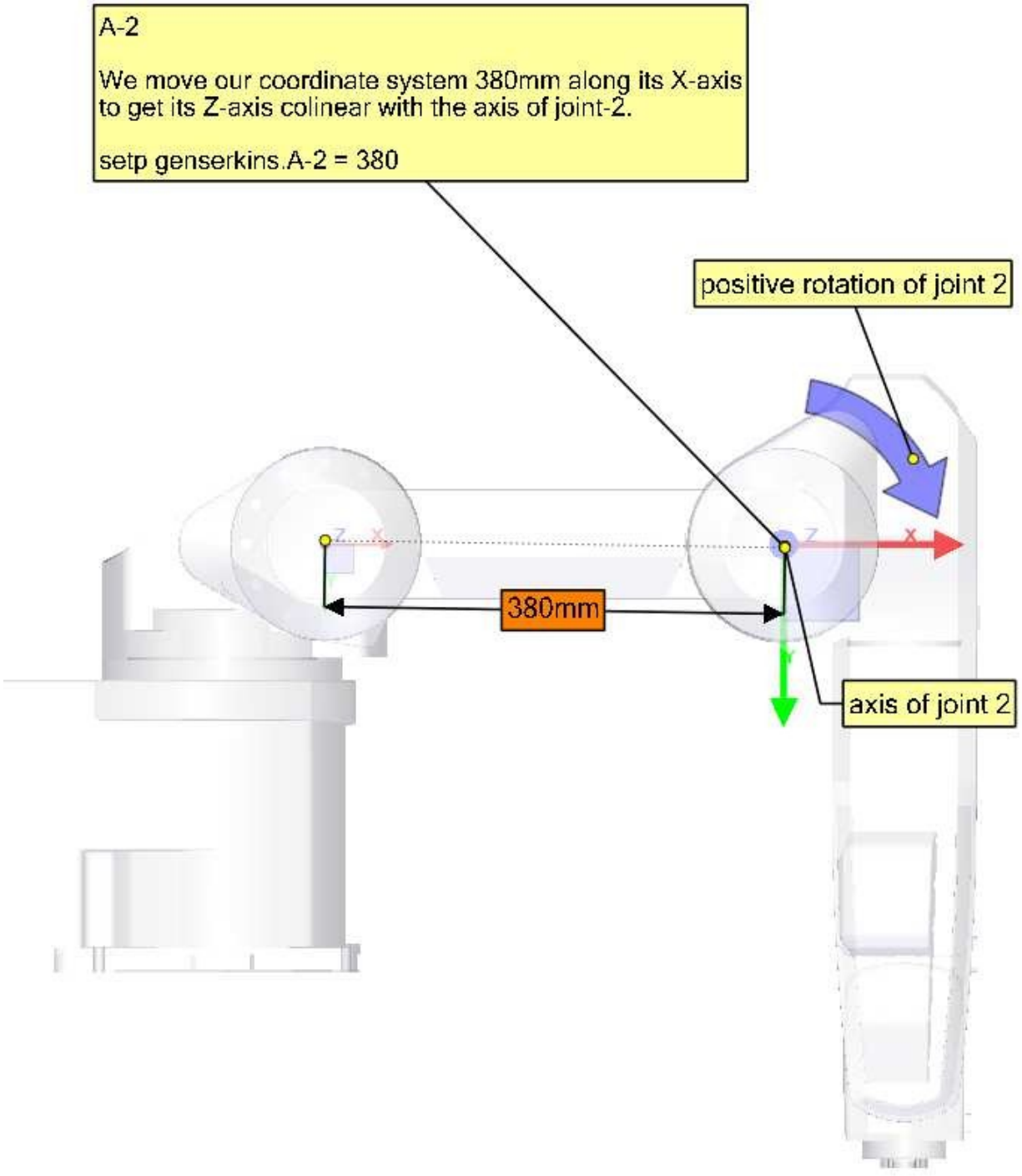
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

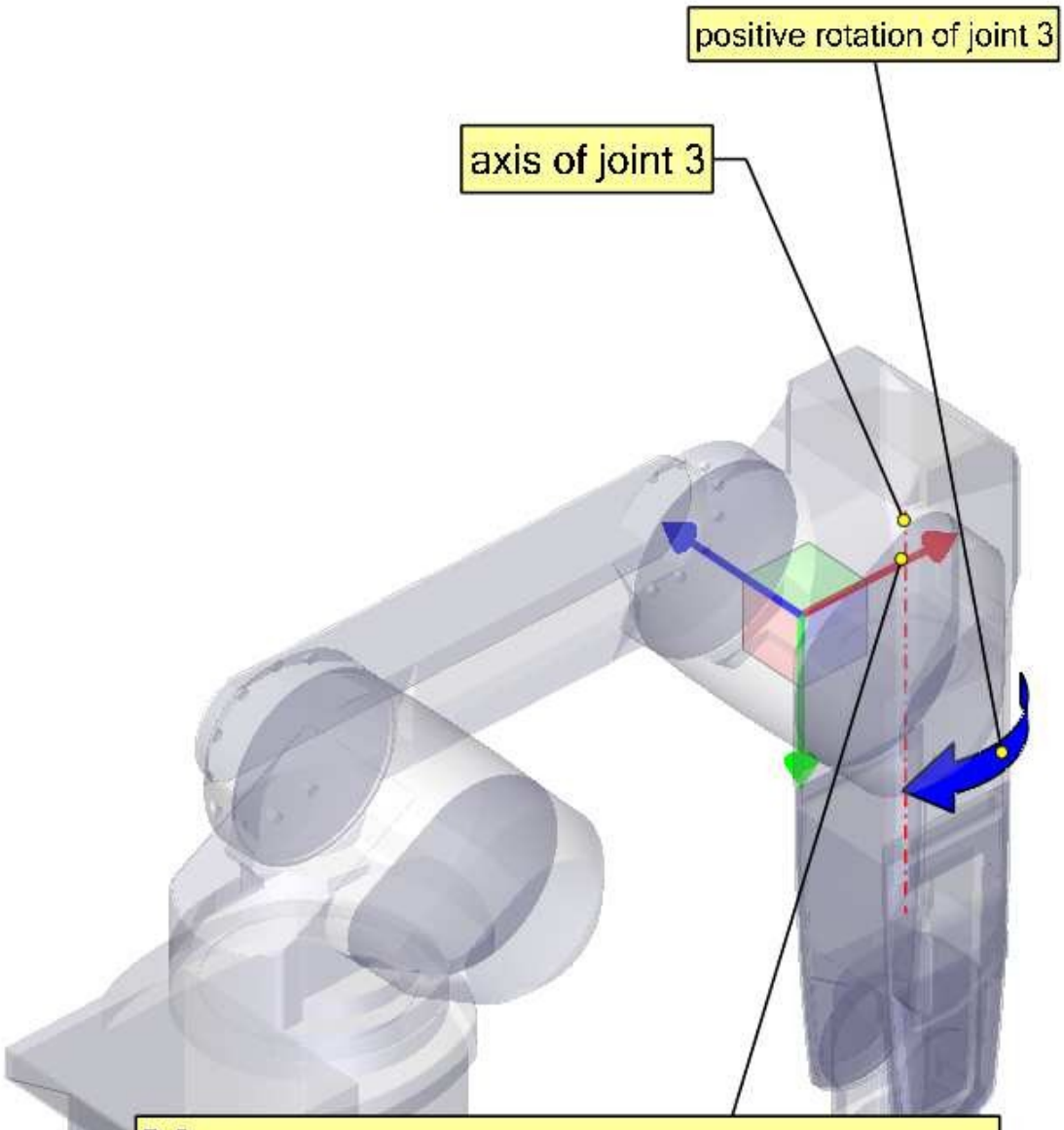
setp genserkins.D-1 = 0

ALPHA-2

The axis of joint -2 is parallel to the axis of joint -1 and points in the same direction. Thus we do not need to rotate our Z-axis.

setp genserkins.ALPHA-2 = 0


A-2
We move our coordinate system 380mm along its X-axis to get its Z-axis colinear with the axis of joint-2.
setp genserkins.A-2 = 380
positive rotation of joint 2
380mm
axis of joint 2




D-2

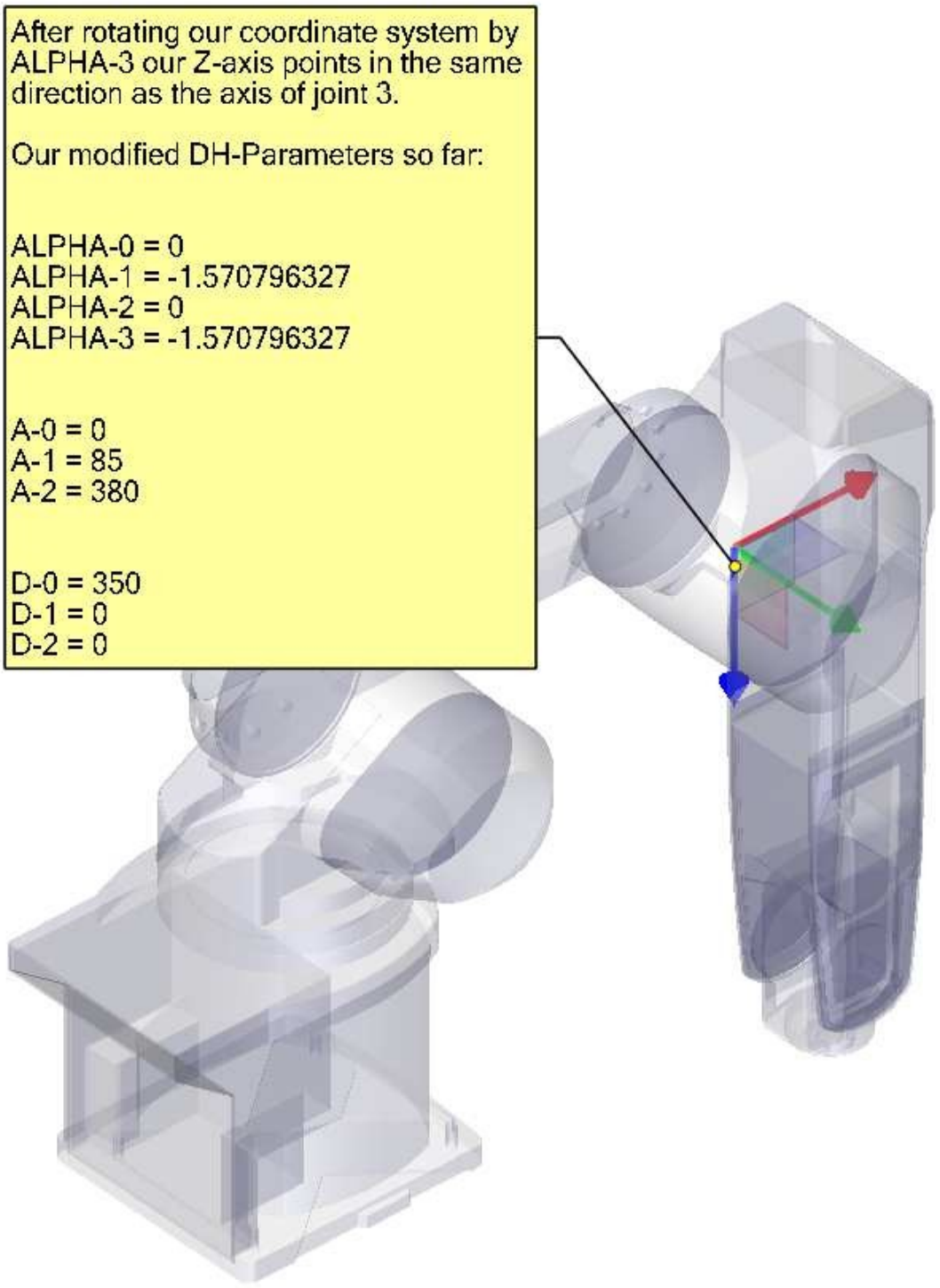
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

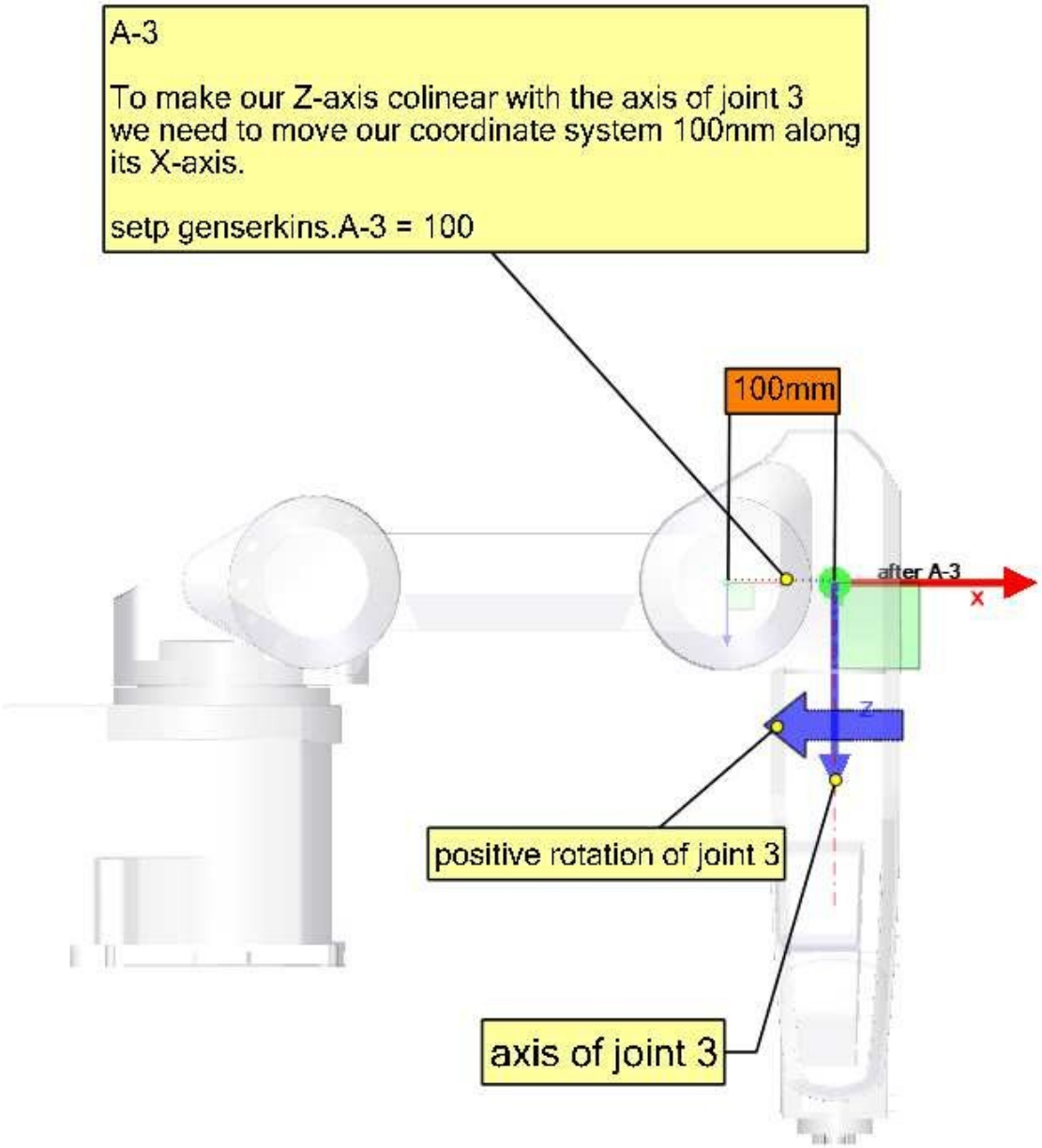
setp genserkins.D-2 = 0

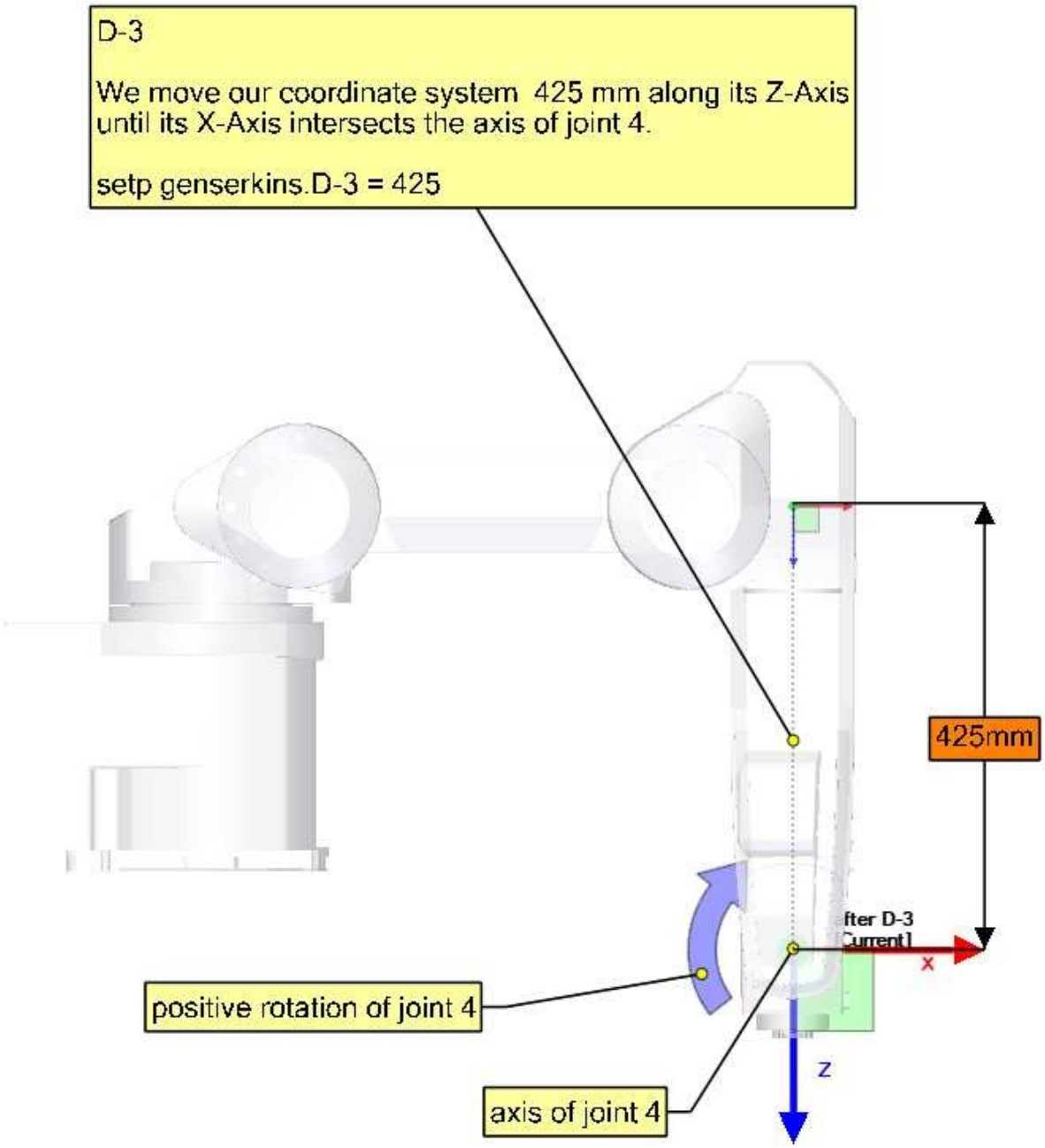
ALPHA-3

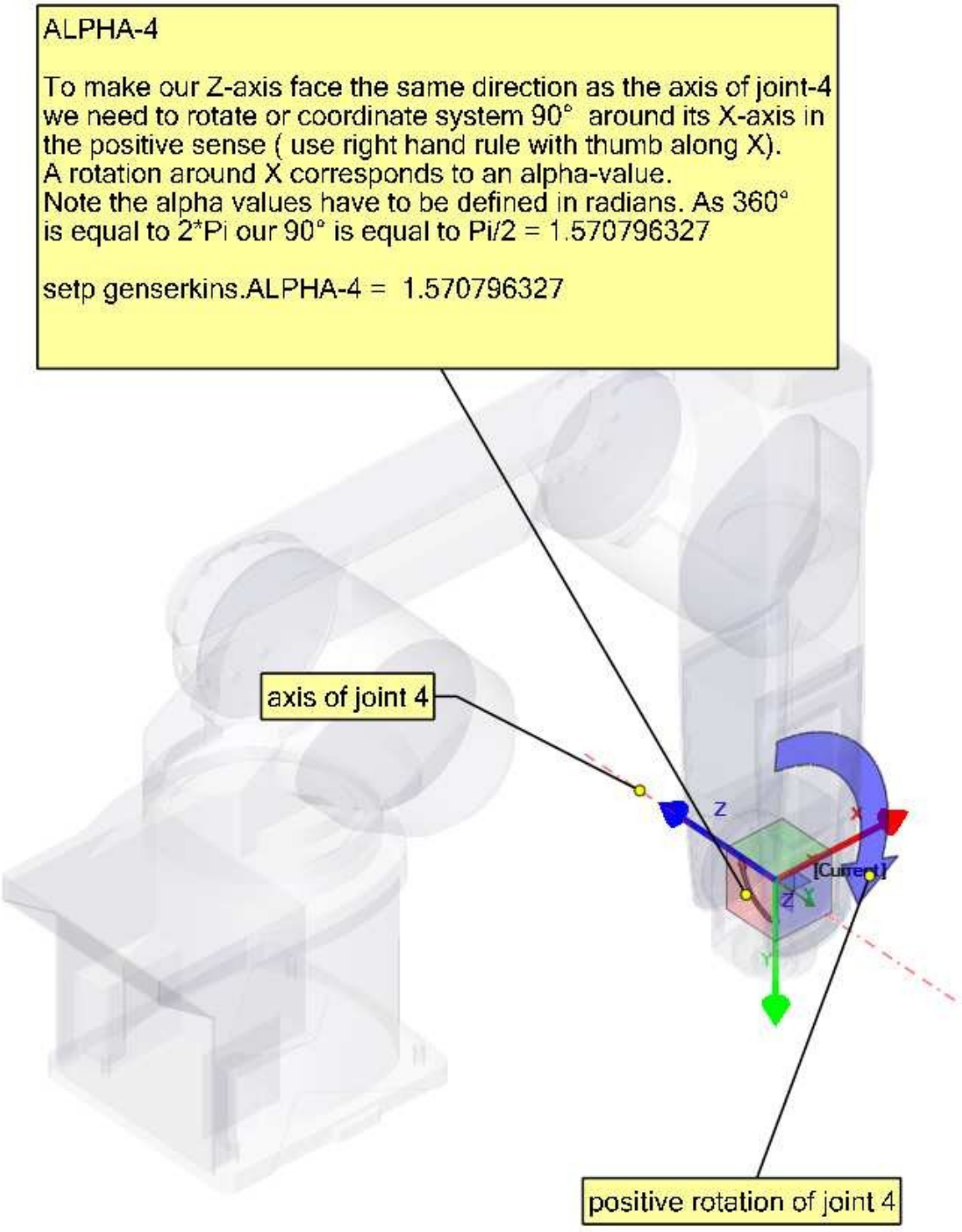
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense ( use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

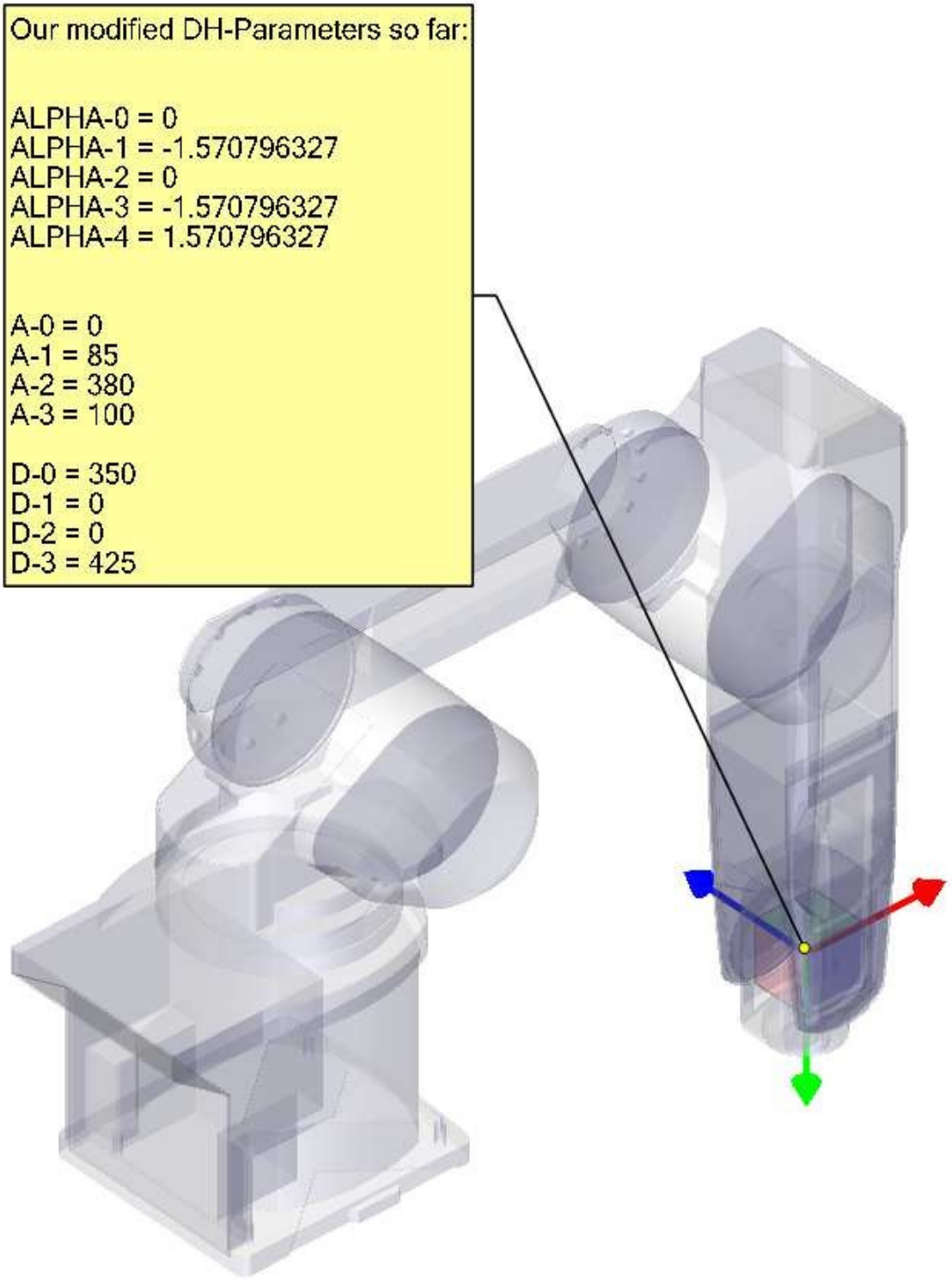
setp genserkins.ALPHA-3 = -1.570796327

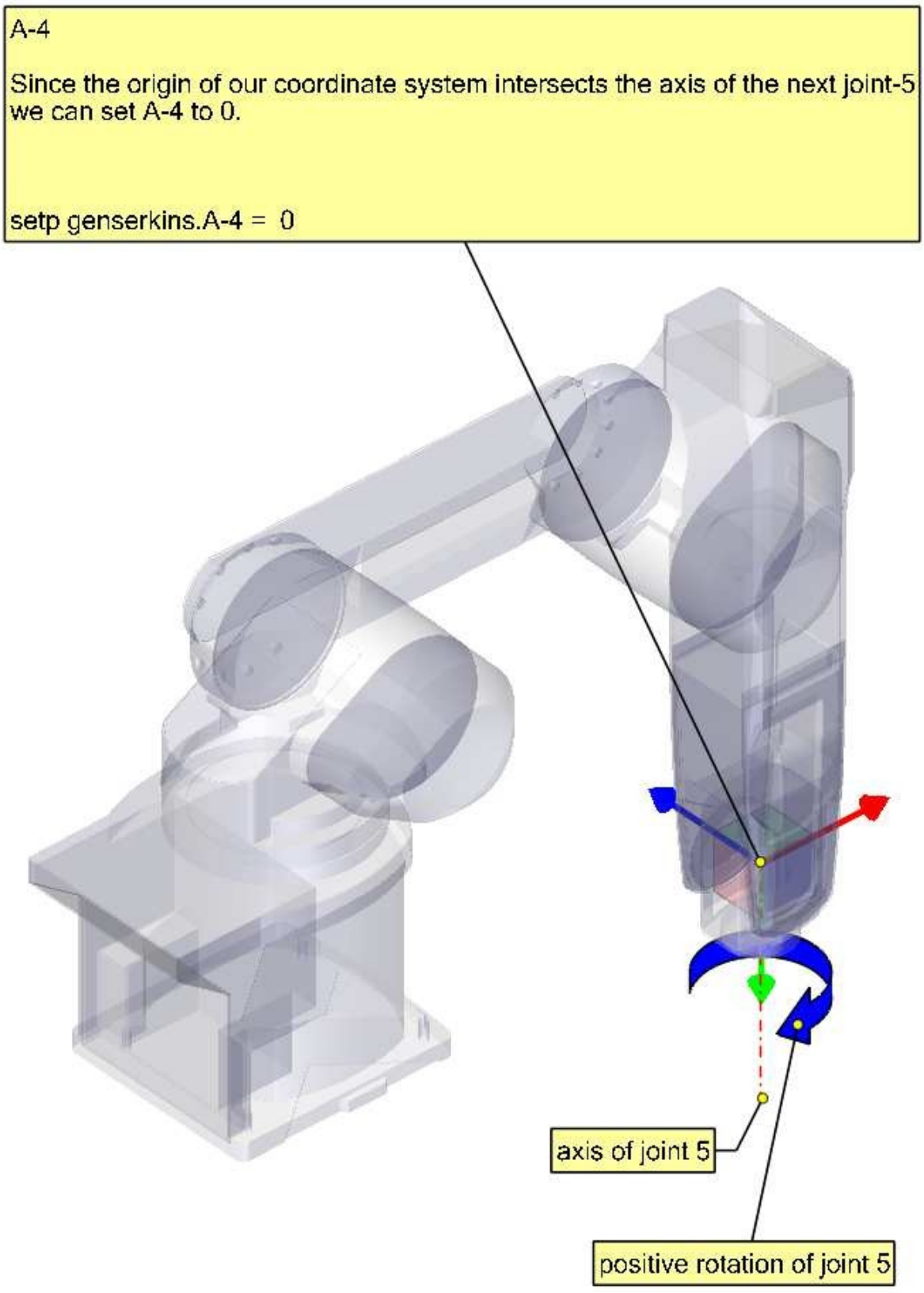

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


A-3
To make our Z-axis colinear with the axis of joint 3 we need to move our coordinate system 100mm along its X-axis.
setp genserkins.A-3 = 100
100mm
after A-3
X
Z
positive rotation of joint 3
axis of joint 3


D-3
We move our coordinate system 425 mm along its Z-Axis until its X-Axis intersects the axis of joint 4.
setp genserkins.D-3 = 425
425mm
positive rotation of joint 4
axis of joint 4
X
Z


ALPHA-4
To make our Z-axis face the same direction as the axis of joint-4 we need to rotate or coordinate system 90° around its X-axis in the positive sense ( use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our 90° is equal to Pi/2 = 1.570796327
setp genserkins.ALPHA-4 = 1.570796327
axis of joint 4
positive rotation of joint 4
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

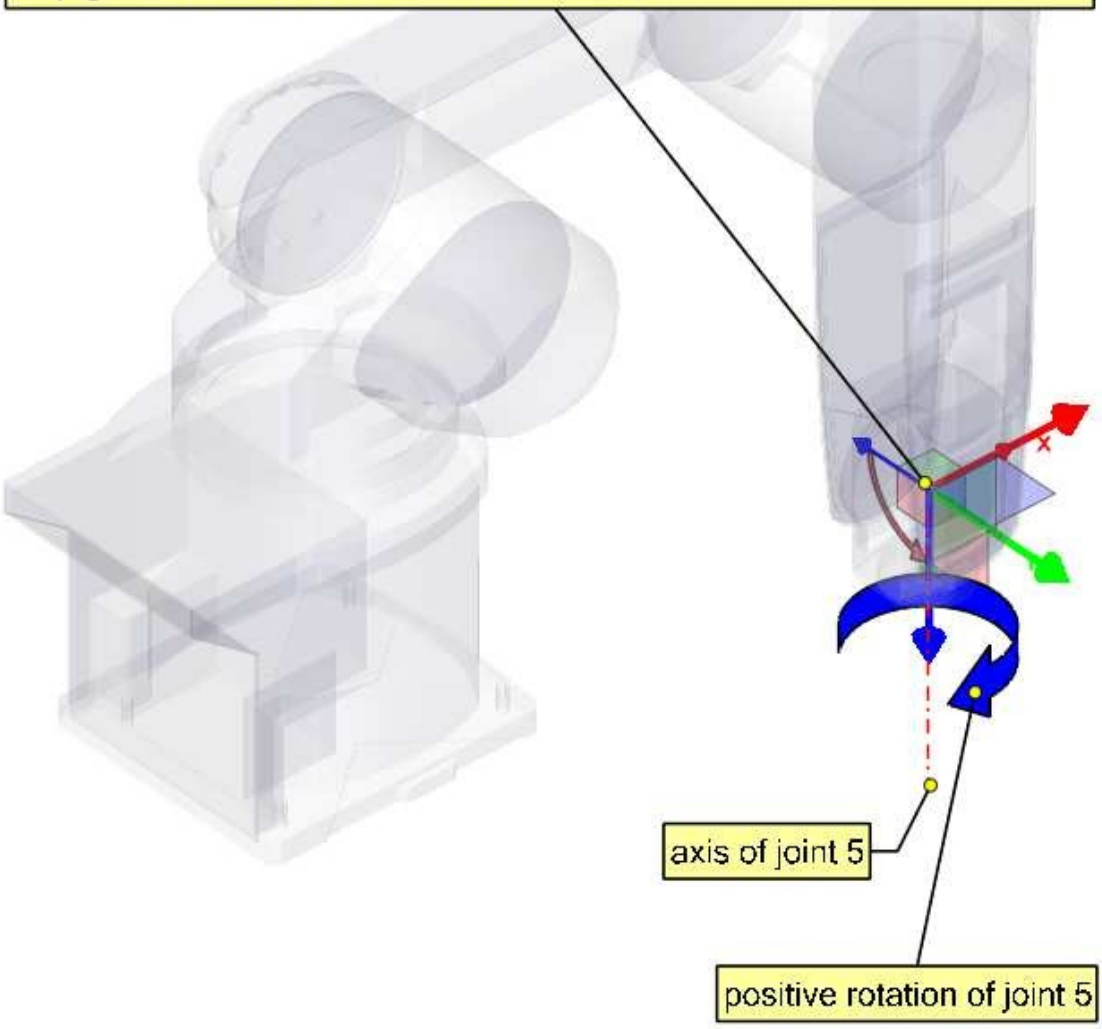
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

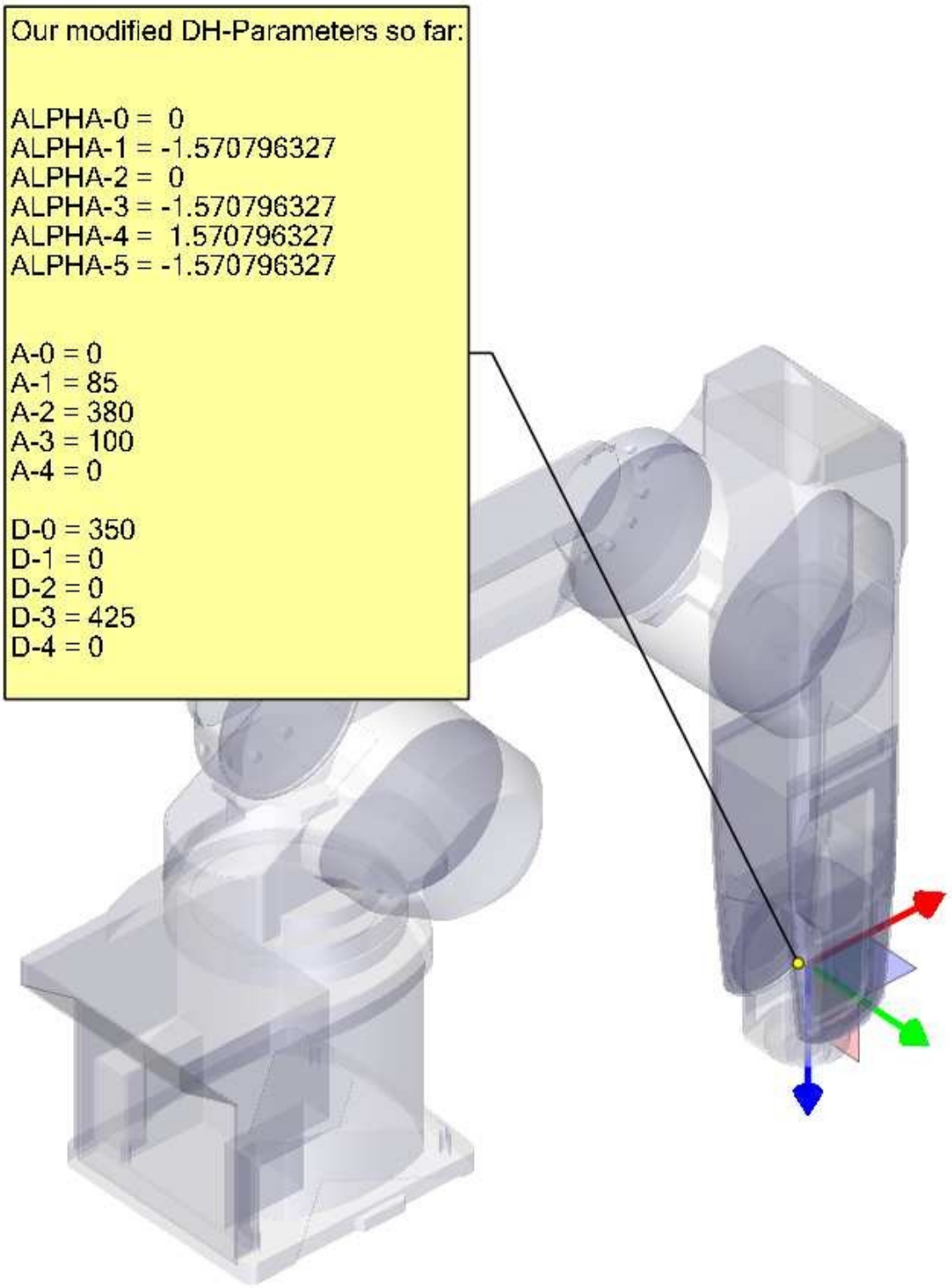
setp genserkins.D-4 = 0

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



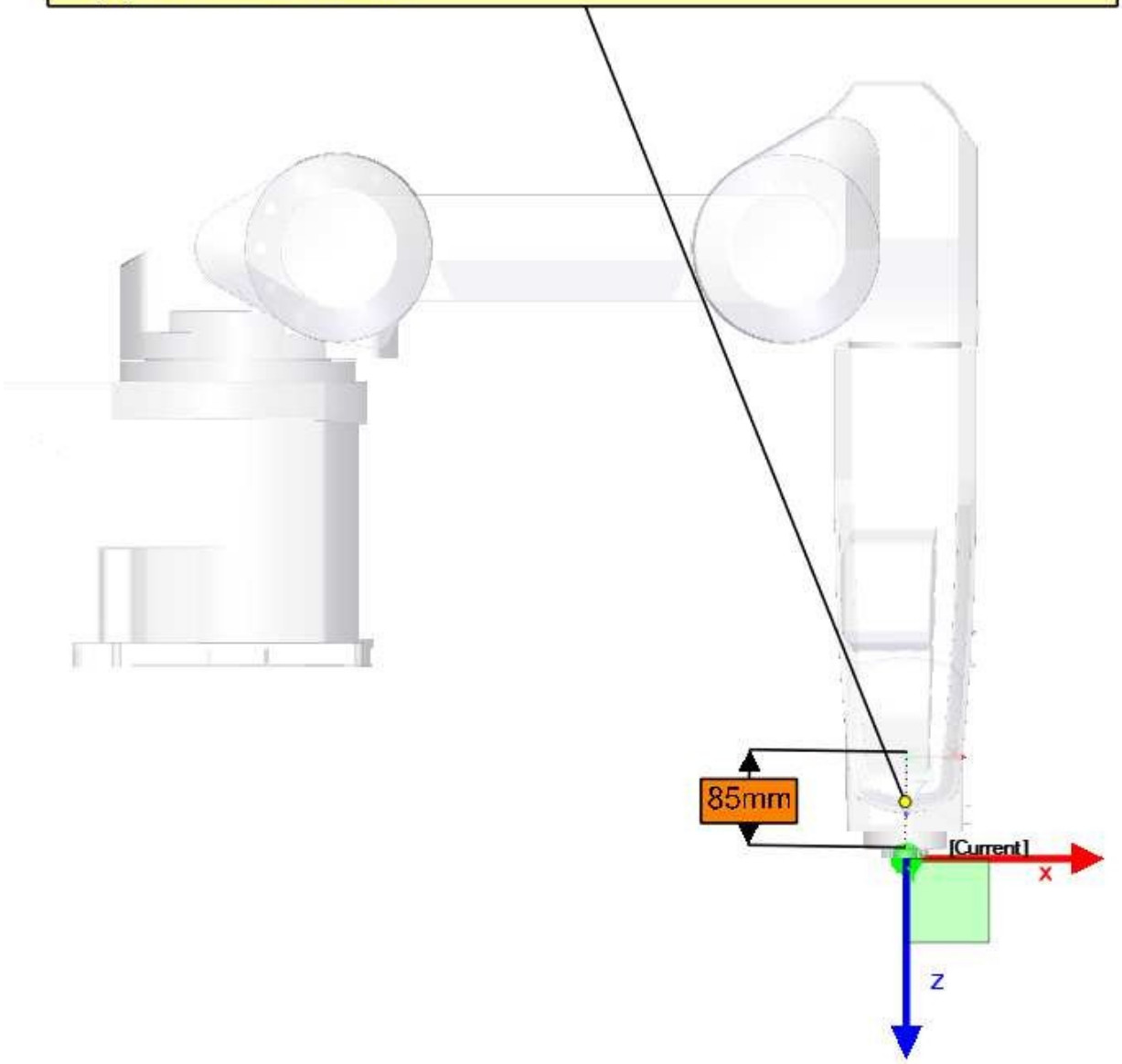

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0

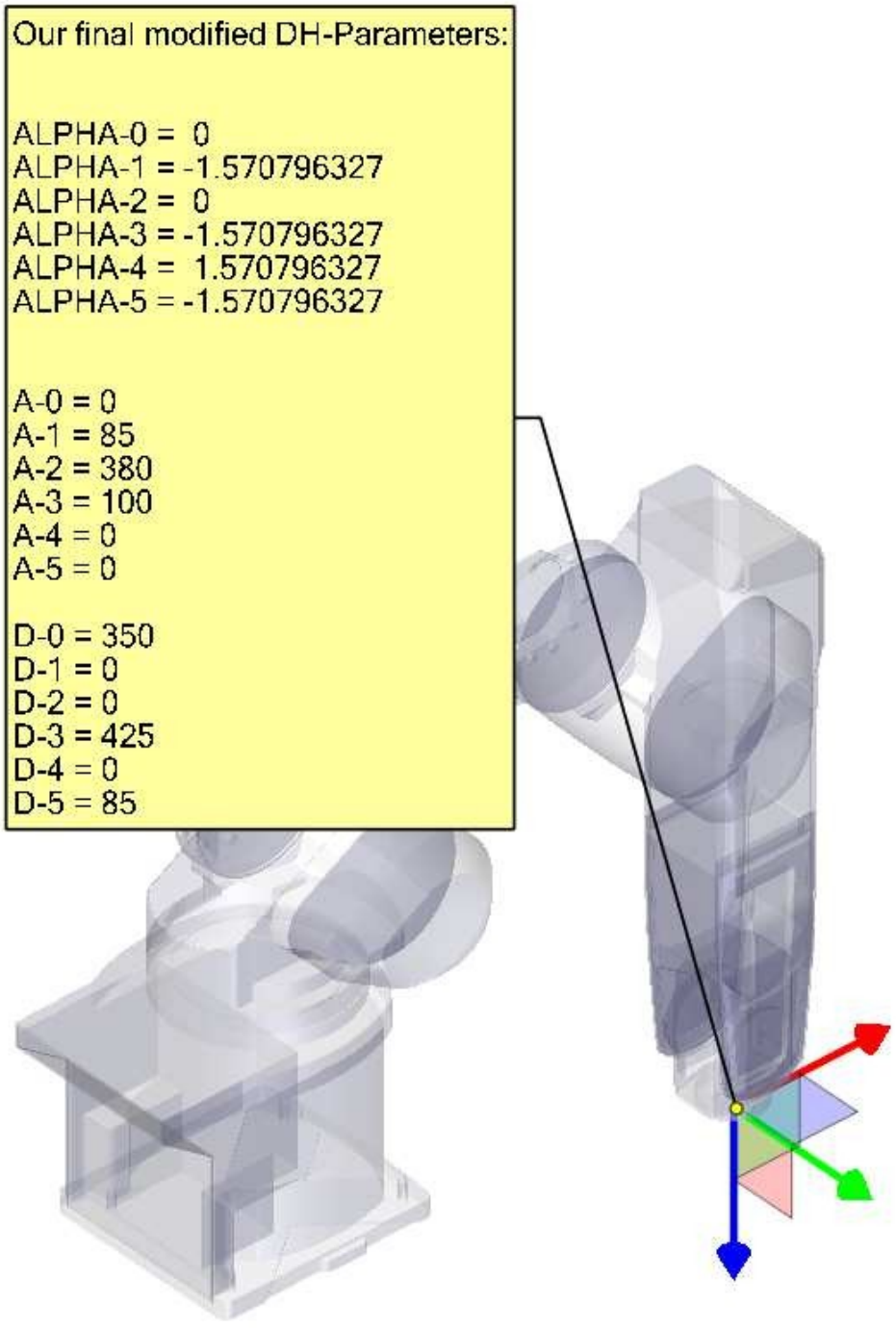
D-5

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-Axis Kinematics

### 9.3.1 Einführung

Coordinated multi-axis CNC machine tools controlled with LinuxCNC, require a special kinematics component for each type of machine. This chapter describes some of the most popular 5-axis machine configurations and then develops the forward (from work to joint coordinates) and inverse (from joint to work) transformations in a general mathematical process for two types of machine.

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.

Note that with these kinematics, the rotational axes move in the opposite direction of what the convention is. See section [”rotational axes”](https://linuxcnc.org/docs/html/gcode/machining-center.html#_rotatio for details.

### 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.

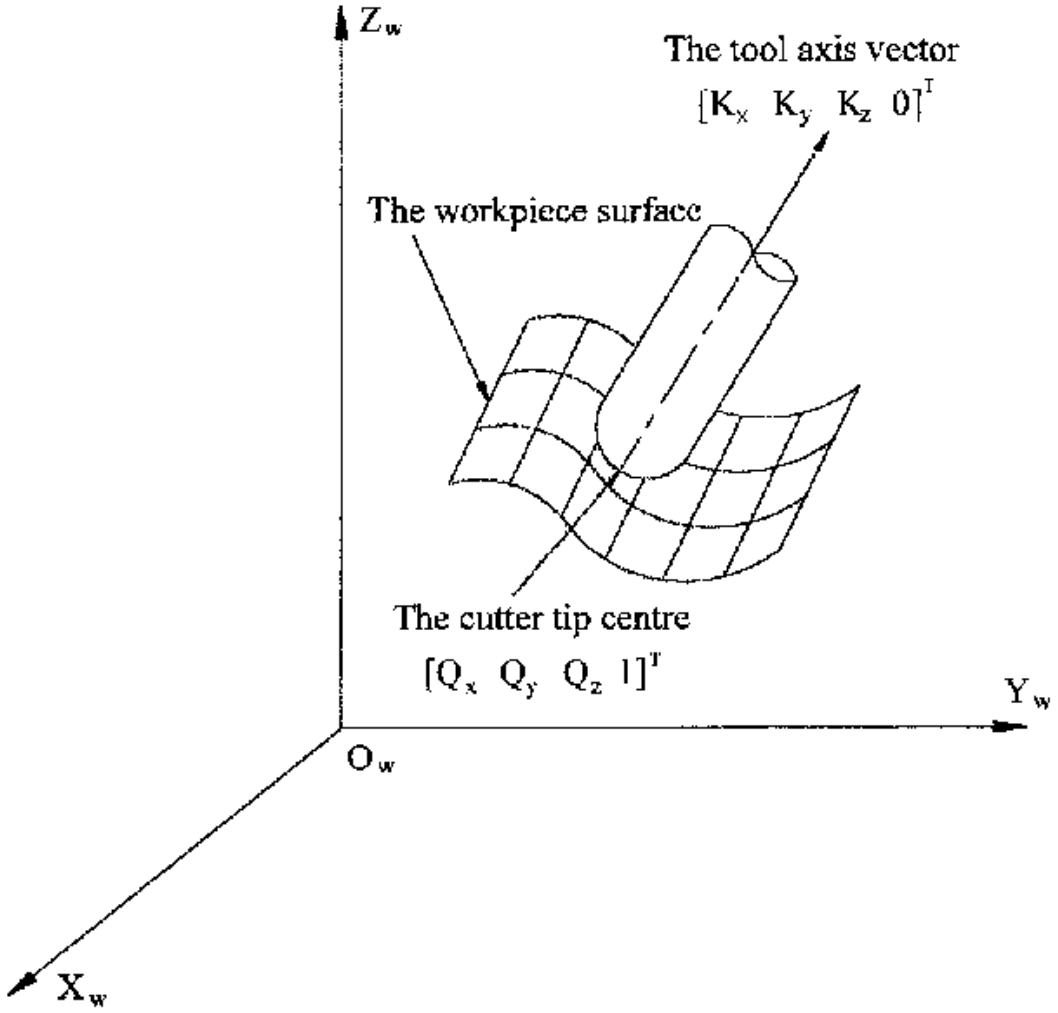



Abbildung 9.2: Standortdaten des Fräsers

### 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.

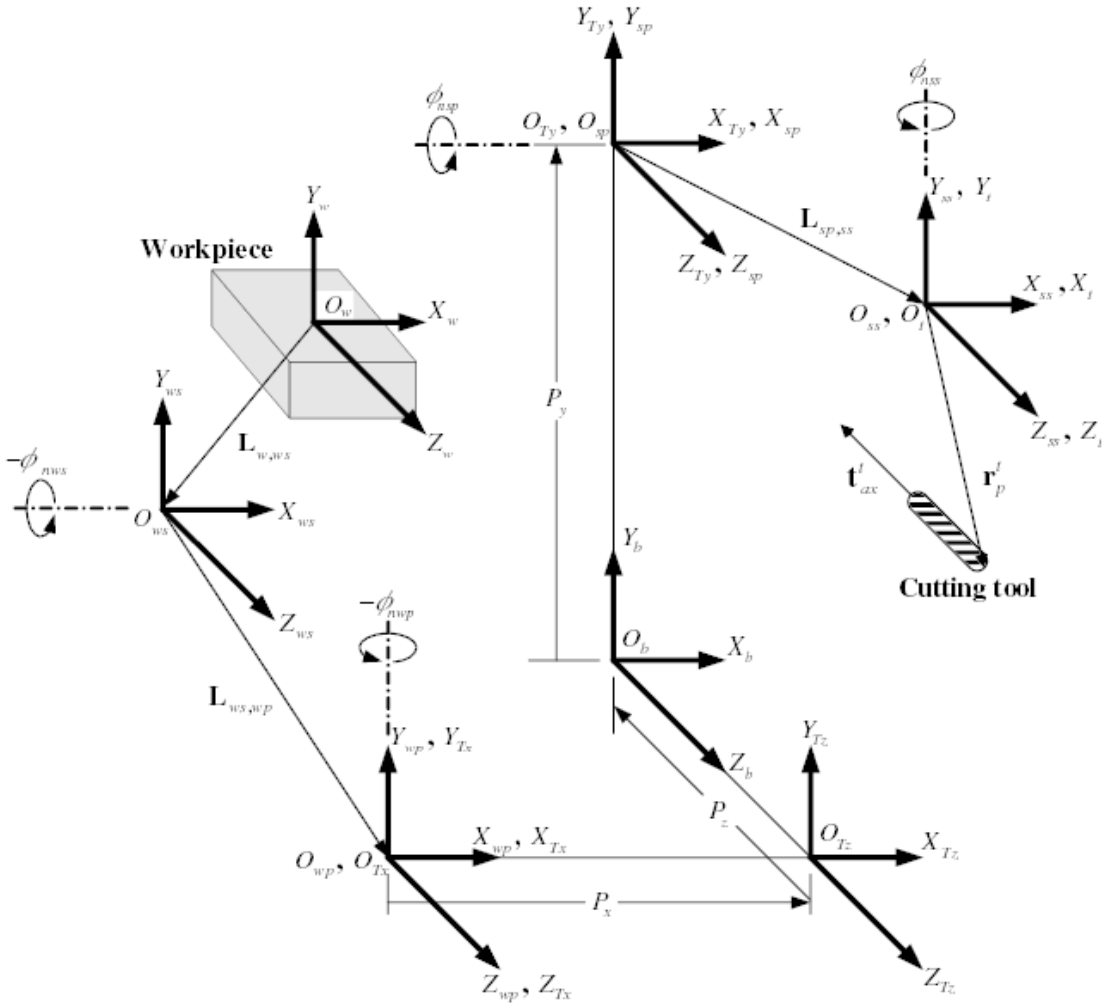



Abbildung 9.3: Allgemeine Konfiguration und Koordinatensysteme

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

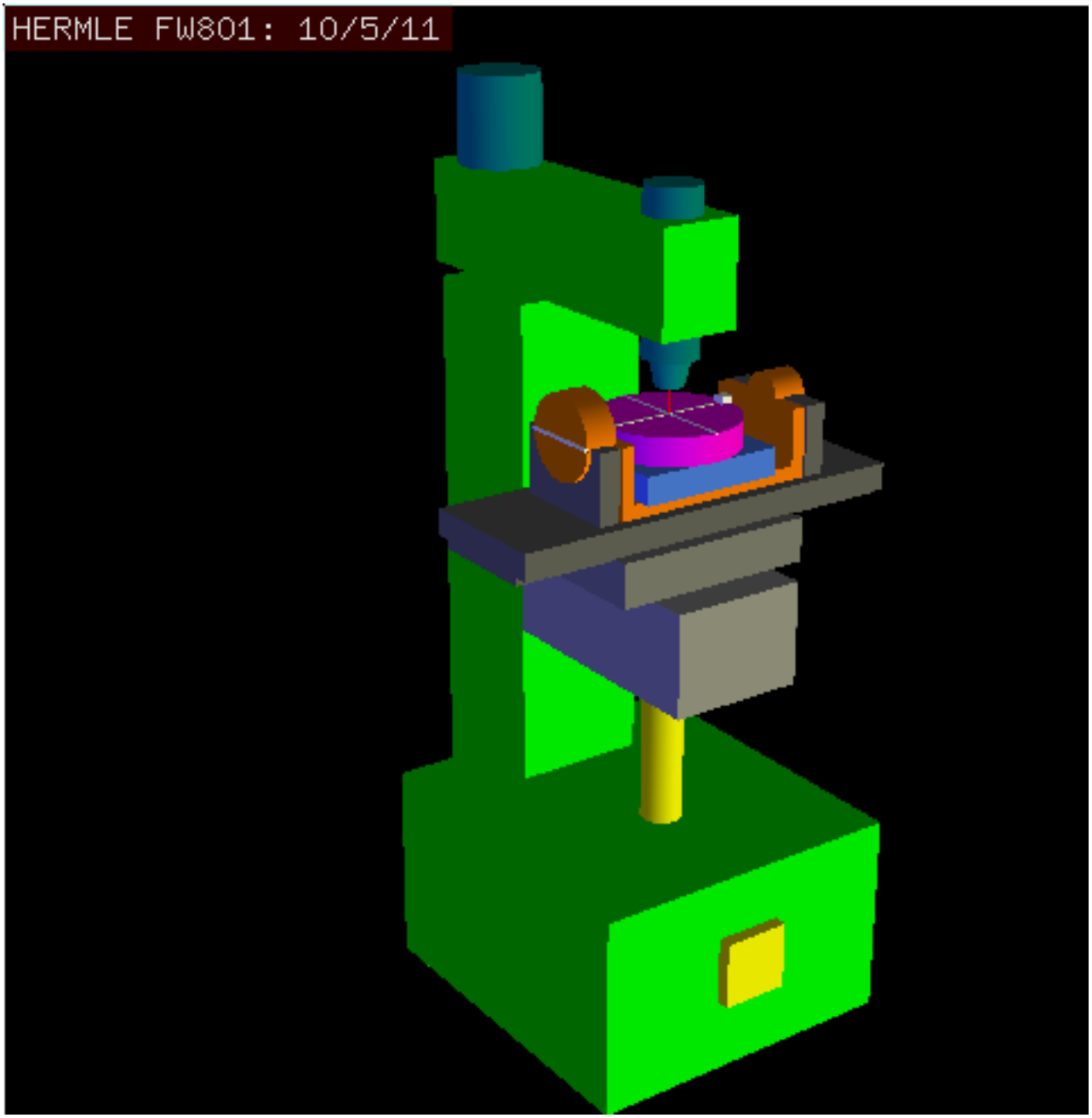



Abbildung 9.4: vismach-Modell von xyzac-trt mit übereinstimmenden Drehachsen

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.

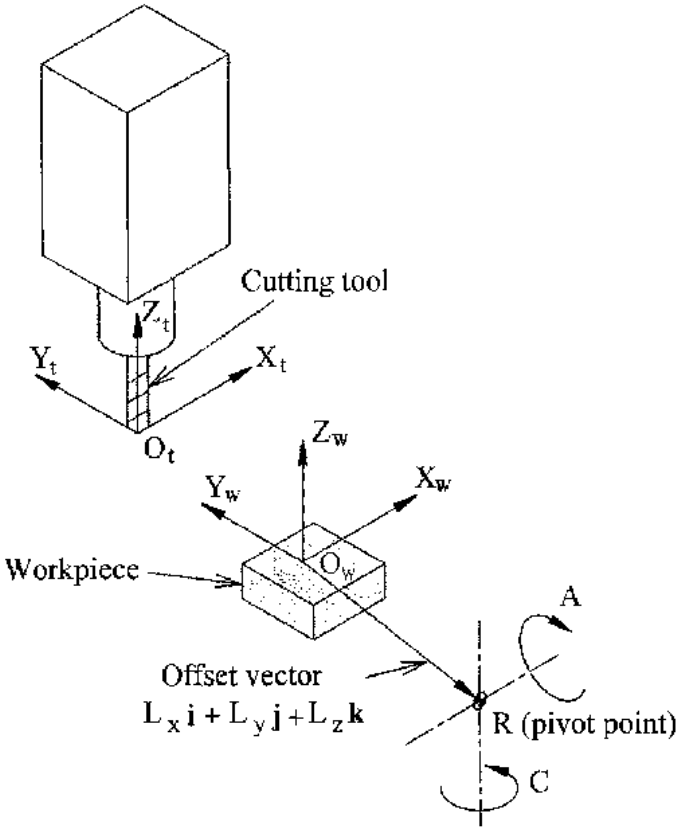



Abbildung 9.5: Kipp-/Drehkonfiguration des Tisches

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

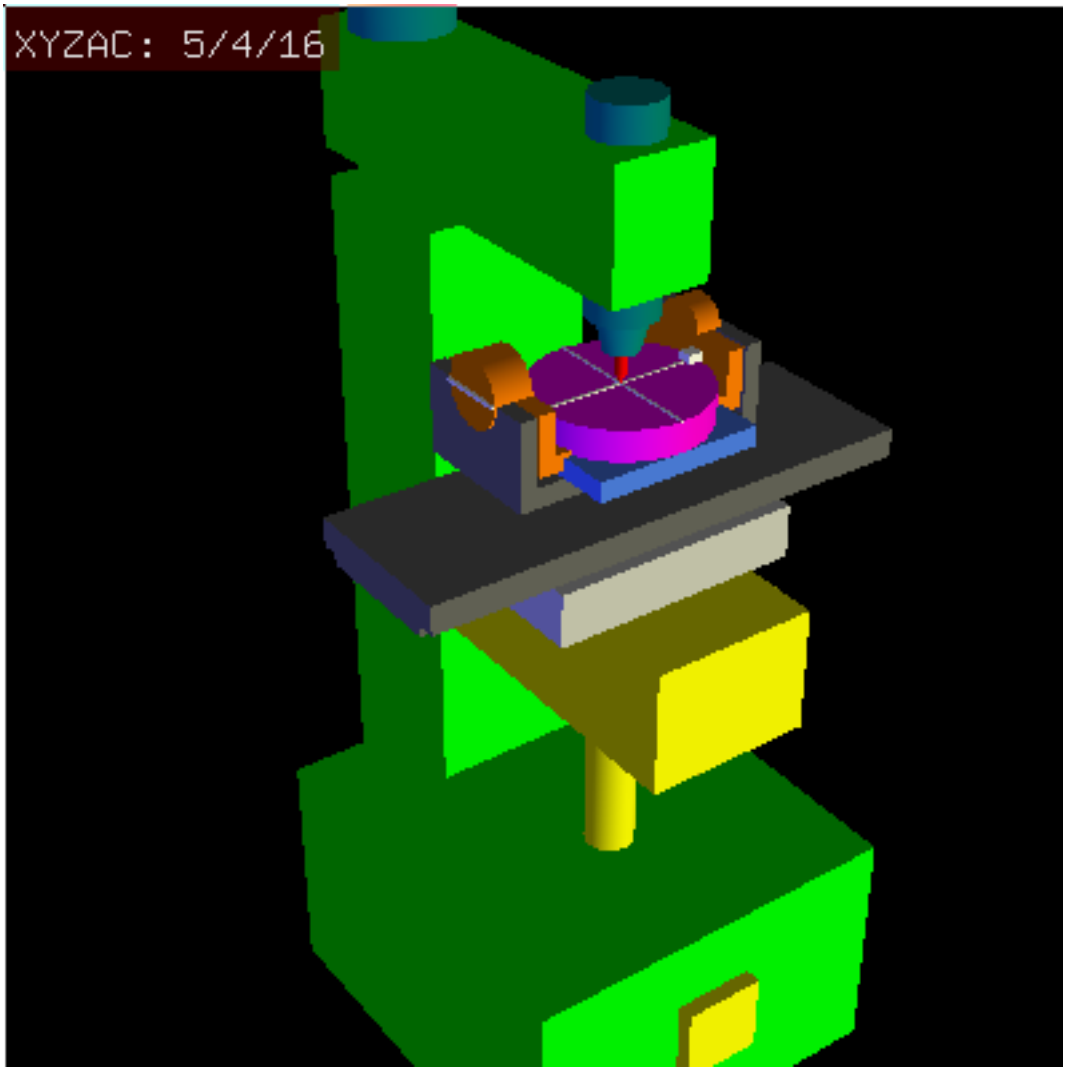



Abbildung 9.6: Vismach-Modell von xyzac-trt mit Rotationsachsenversatz (positiv)

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.

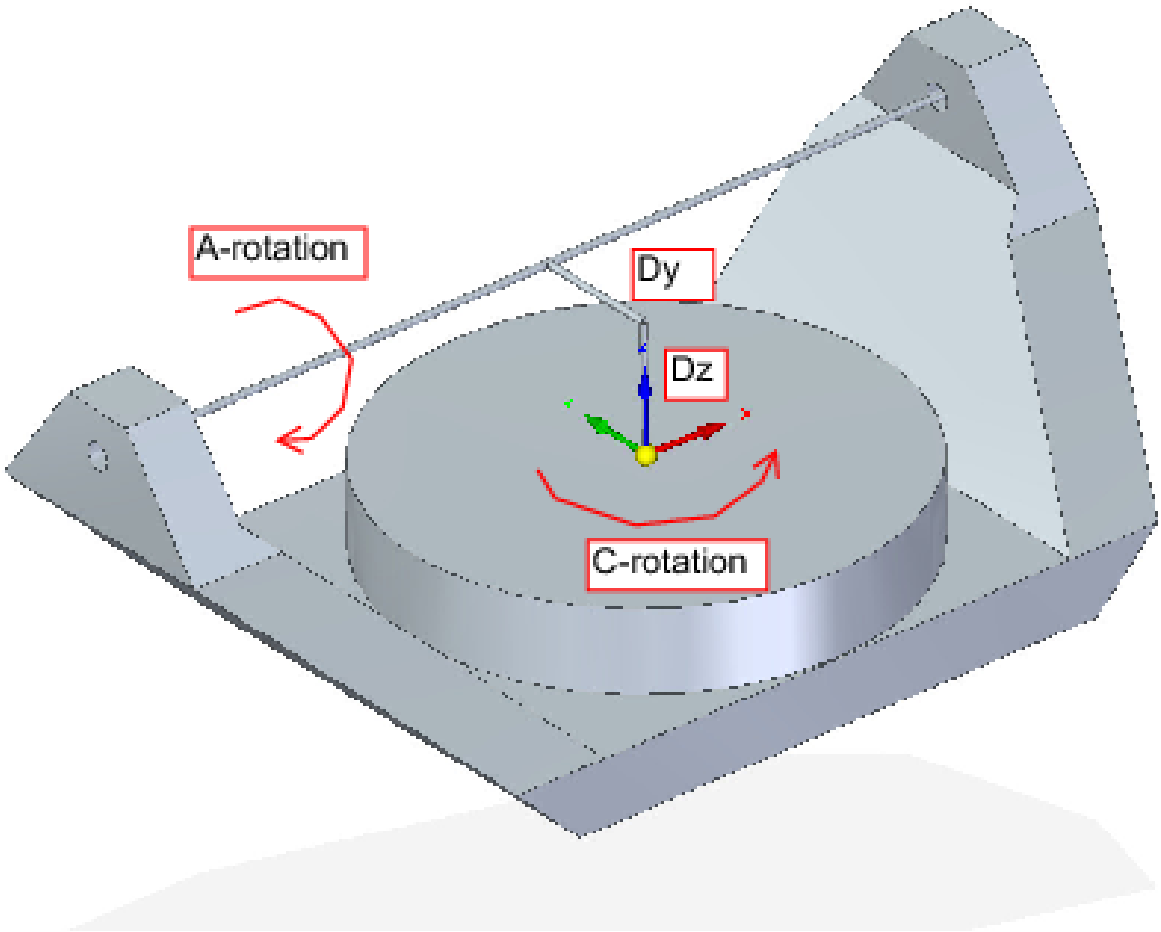



Abbildung 9.7: Kipp-/Drehkonfiguration des Tisches xyzac-trt, mit Achsenversatz

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

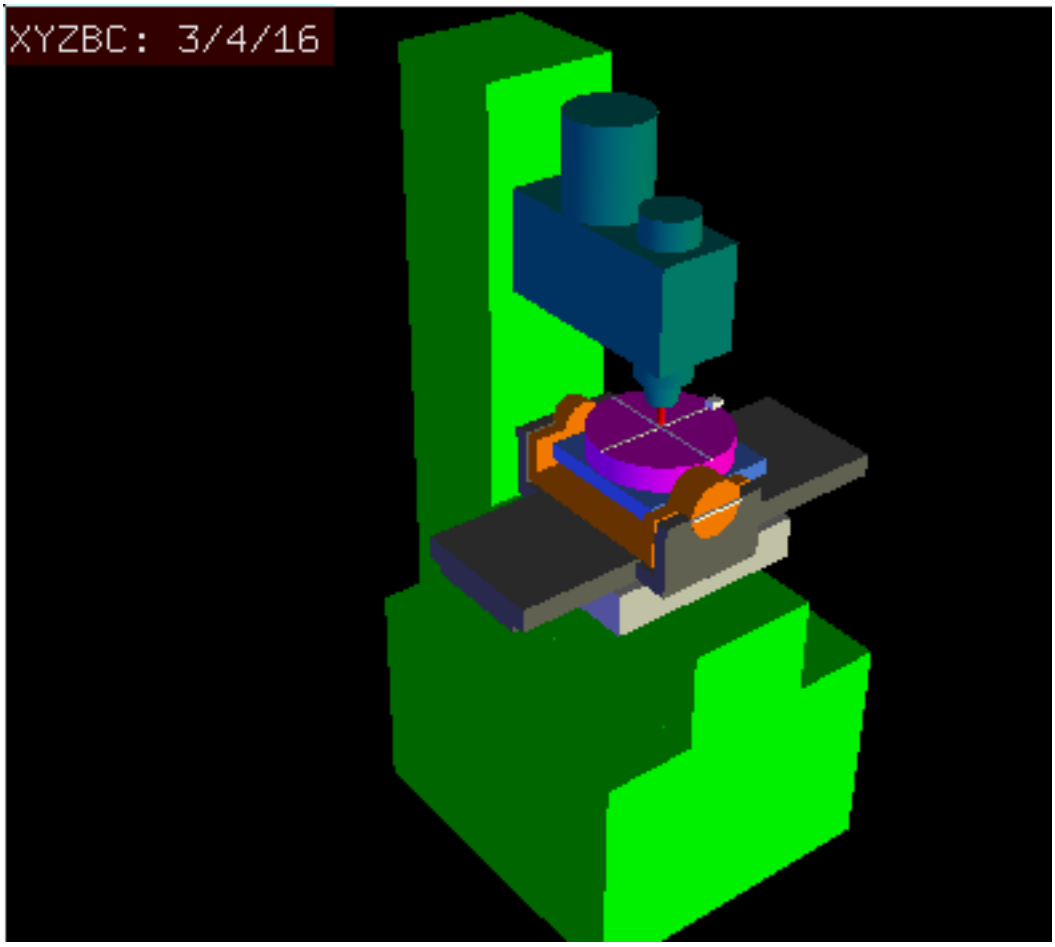



Abbildung 9.8: Vismach-Modell von xyzbc-trt mit Rotationsachsenversatz (negativ)

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.

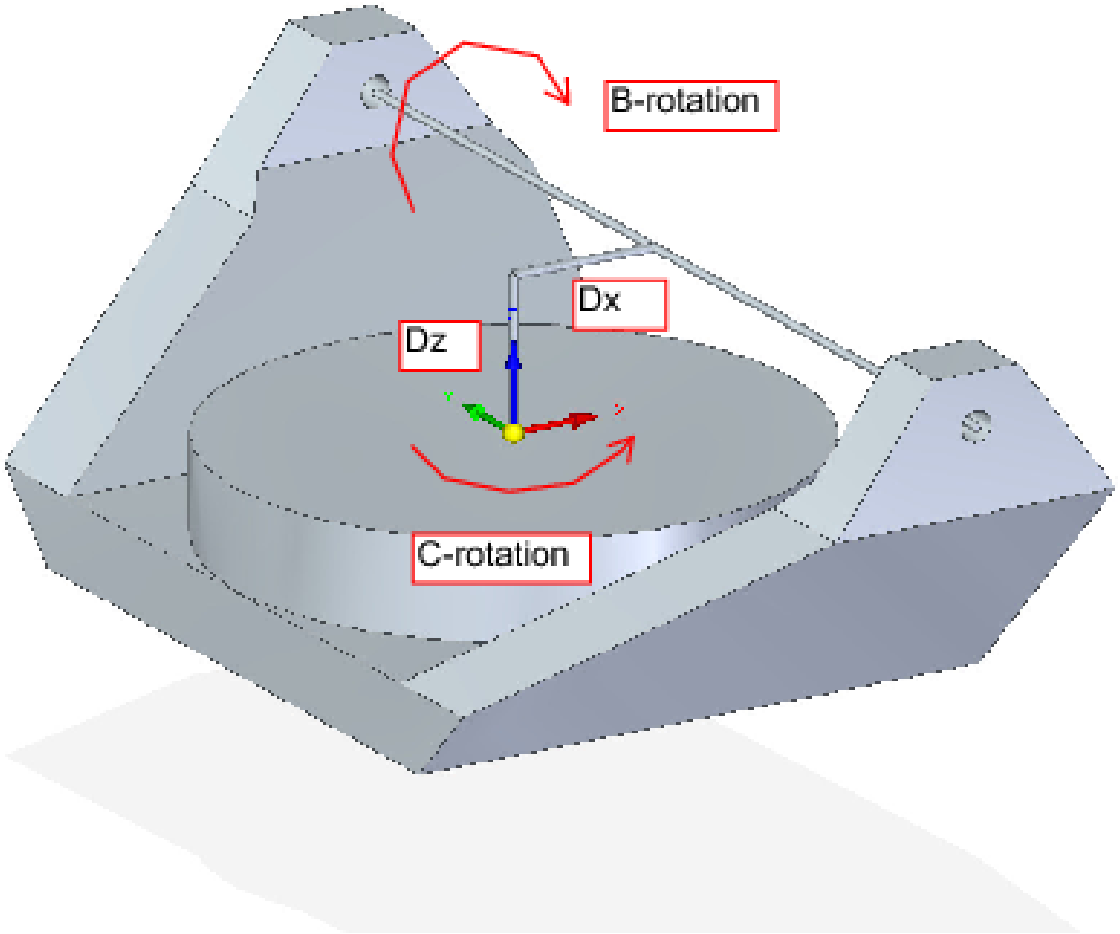



Abbildung 9.9: Kipp-/Drehkonfiguration des Tisches xyzbc-trt, mit Achsenversatz

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 non-realtime 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

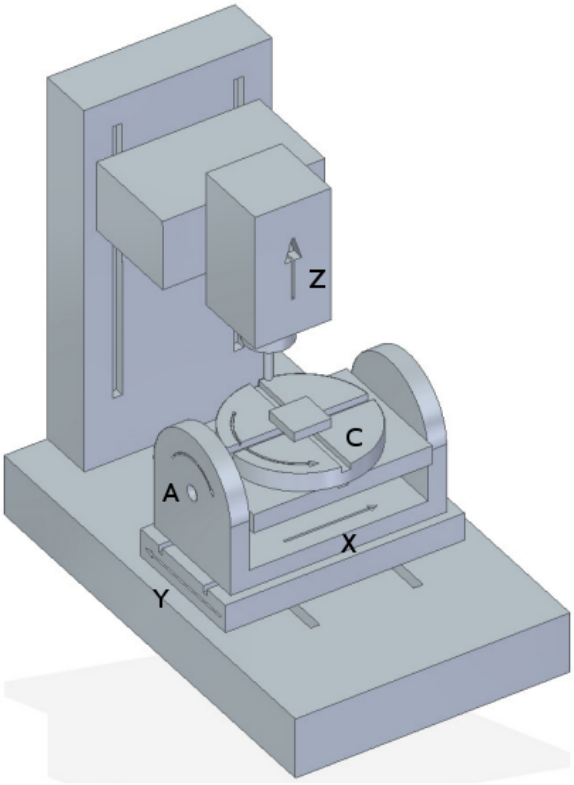



Abbildung 9.10: Kipp-/Drehkonfiguration des Tisches

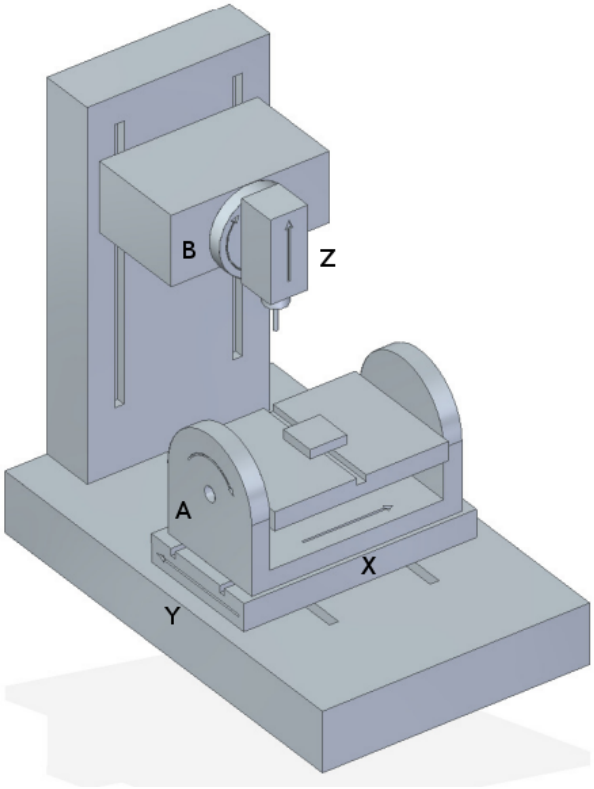



Abbildung 9.11: Spindel-/Tischkippkonfiguration

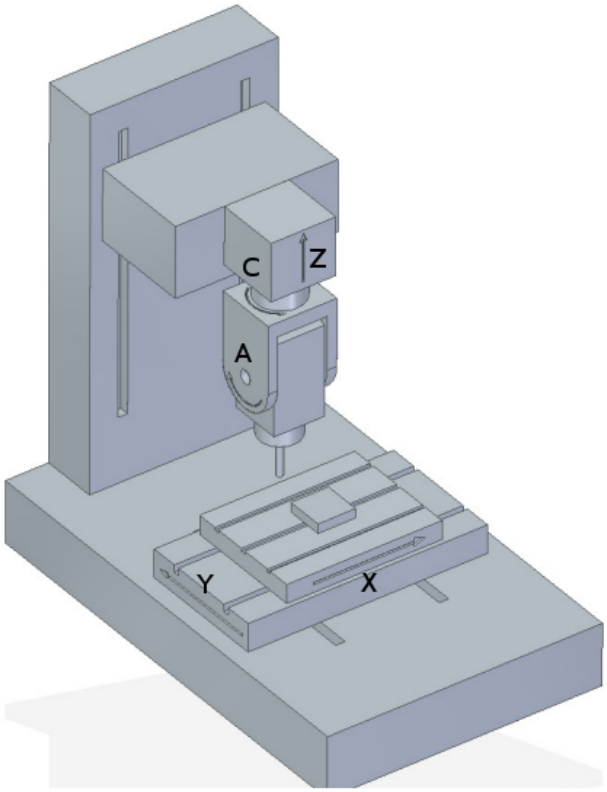



Abbildung 9.12: Kipp-/Drehkonfiguration der Spindel

### 9.3.9 VERWEISE

1. AXIS MACHINE TOOLS: Kinematics and Vismach Implementation in LinuxCNC, RJ du Preez, SA-CNC-CLUB, April 7, 2016.
2. A Postprocessor Based on the Kinematics Model for General Five-Axis machine Tools: C-H She, R-S Lee, J Manufacturing Processes, V2 N2, 2000.
3. NC Post-processor for 5-axis milling of table-rotating/tilting type: YH Jung, DW Lee, JS Kim, HS Mok, J Materials Processing Technology,130-131 (2002) 641-646.
4. 3D 6-DOF Serial Arm Robot Kinematics, RJ du Preez, SA-CNC-CLUB, Dec. 5, 2013.
5. Design of a generic five-axis postprocessor based on generalized kinematics model of machine tool: C-H She, C-C Chang, Int. J Machine Tools & Manufacture, 47 (2007) 537-545.

## 9.4 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 Tuning

## 9.5.1 PID-Regler (engl. PID controller)

A proportional-integral-derivative controller (PID controller) is a common feedback loop component in industrial control systems. [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.

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

[3] This Subsection is taken from an much more extensive article found at https://en.wikipedia.org/wiki/PID_controller

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.

In particular, some of these codes implement a fixed sequence of steps to be executed. While some of these, like M6, can be moderately configured by activating or skipping some of these steps through INI file options, overall the behavior is fairly rigid. So - if your are happy with this situation, then this manual section is not for you.

In many cases, this means that supporting a non *out of the box* configuration or machine is either cumbersome or impossible, or requires resorting to changes at the *C/C+\+* language level. The latter is unpopular for good reasons - changing internals requires in-depth understanding of interpreter internals, and moreover brings its own set of support issues. While it is conceivable that certain patches might make their way into the main LinuxCNC distribution, the result of this approach is a hodgepodge of special-case solutions.

A good example for this deficiency is tool change support in LinuxCNC: While random tool changers are well supported, it is next to impossible to reasonably define a configuration for a manual-tool change machine with, for example, an automatic tool length offset switch being visited after a tool change, and offsets set accordingly. Also, while a patch for a very specific rack tool changer exists, it has not found its way back into the main code base.

However, many of these things may be fixed by using an O-word procedure instead of a built in code - whenever the - insufficient - built in code is to be executed, call the O-word procedure instead. While possible, it is cumbersome - it requires source-editing of NGC programs, replacing all calls to the deficient code by a an O-word procedure call.

In its simplest form, a remapped code isn't much more than a spontaneous call to an O-word procedure. This happens behind the scenes - the procedure is visible at the configuration level, but not at the NGC program level.

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]
```

whereas M- or G-codes typically take required or optional *word* parameters. For instance, G76 (threading) requires the P,Z,I,J and K words, and optionally takes the R,Q,H, E and L words.

So it isn't simply enough to say *whenever you encounter code X, please call procedure Y* - at least some checking and conversion of parameters needs to happen. This calls for some *glue code* between the new code, and its corresponding NGC procedure to execute before passing control to the NGC procedure.

This glue code is impossible to write as an O-word procedure itself, since the RS274NGC language lacks the introspective capabilities and access into interpreter internal data structures to achieve the required effect. Doing the glue code in - again - *C/C+\+* would be an inflexible and therefore unsatisfactory solution.

**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.
- For remapping T,M6,M61,S,F there is some standard Python glue which should cover most situations, see Standard Glue.
- For more complex situations, one can write your own Python glue to implement new behavior.

Embedded Python functions in the Interpreter started out as glue code, but turned out very useful well beyond that. Users familiar with Python will likely find it easier to write remapped codes, glue, O-word procedures, etc. in pure Python, without resorting to the somewhat cumbersome RS274NGC language at all.

**Ein Wort zu eingebettetem Python** Many people are familiar with *extending* the Python interpreter by *C/C++* modules, and this is heavily used in LinuxCNC to access Task, HAL and Interpreter internals from Python scripts. *Extending Python* basically means: Your Python script executes as *it is in the driver seat*, and may access non-Python code by importing and using extension modules written in *C/C+\+*. Examples for this are the LinuxCNC `hal`, `gcode` and `emc` modules.

Embedded Python is a bit different and less commonly known: The main program is written in C/C++ and may use Python like a subroutine. This is powerful extension mechanism and the basis for the *scripting extensions* found in many successful software packages. Embedded Python code may access *C/C+\+* variables and functions through a similar extension module method.

## 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

Please note that currently only some existing codes can be redefined, while there are many *free* codes that may be available for remapping. When developing redefined existing code, it is a good idea to start with an unassigned G- or M- code, so that you can use both an existing behavior as well as a new one. When you're done, redefine the existing code to use your remapping configuration.

- The current set of unused M-codes, available for user definition, can be found in the unallocated M-codes section.
- Informationen zu unbelegten G-Codes finden Sie hier.
- Existing codes that can be reassigned are listed in the remappable codes section.

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.

This remap also adds wear offsets to the tool offset, e.g. T201 would index to tool 2 (with tool 2's tool offset) and adds wear offset 1. In the tool table, tools numbers above 10000 are wear offsets, e.g. in the tool table, tool 10001 would be wear offset 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

Note that currently only a few existing codes may be redefined, whereas there are many *free* codes which might be made available by remapping. When developing a redefined existing code, it might be a good idea to start with an unallocated G- or M-code, so both the existing and new behavior can be exercised. When done, redefine the existing code to use your remapping setup.

- The current set of unused M-codes open to user definition can be found here.
- Unallocated G-codes are listed here.
- Vorhandene Codes, die neu zugeordnet werden können, sind in dieser Liste aufgeführt.

#### 9.6.2.3 Handhabung der Parameter

Let's assume the new code will be defined by an NGC procedure, and needs some parameters, some of which might be required, others might be optional. We have the following options to feed values to the procedure:

1. Extracting words from the current block and pass them to the procedure as parameters (like `X22.34` or `P47`),
2. referring to INI file variables,
3. referring to global variables (like `#2200 = 47.11` or `#<_global_param> = 315.2`).

The first method is preferred for parameters of dynamic nature, like positions. You need to define which words on the current block have any meaning for your new code, and specify how that is passed to the NGC procedure. Any easy way is to use the argspec statement. A custom prolog might provide better error messages.

Using to INI file variables is most useful for referring to setup information for your machine, for instance a fixed position like a tool-length sensor position. The advantage of this method is that the parameters are fixed for your configuration, regardless which NGC file you're currently executing.

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

Your new code might succeed or fail, for instance if passed an invalid parameter combination. Or you might choose to *just execute* the procedure and disregard results, in which case there isn't much work to do.

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.

When you define a new executable code, the interpreter does not yet know where your code fits into this scheme. For this reason, you need to choose an appropriate modal group for your code to execute in.

#### 9.6.2.6 Ein minimales Beispiel für neu zugeordneten Code

To give you an idea how the pieces fit together, let's explore a fairly minimal but complete remapped code definition. We choose an unallocated M-code and add the following option to the INI file:

```
[RS274NGC]
REMAP=M400  modalgroup=10 argspec=Pq ngc=myprocedure
```

Zusammengefasst bedeutet dies:

- The `M400` code takes a required parameter `P` and an optional parameter `Q`. Other words in the current block are ignored with respect to the `M400` code. If the `P` word is not present, fail execution with an error.
- When an `M400` code is encountered, execute `myprocedure.ngc` along the other modal group 10 M-codes as per order of execution.
- The value of `P`, and `Q` are available in the procedure as local named parameters. The may be referred to as `#<P>` and `#<Q>`. The procedure may test whether the `Q` word was present with the EXISTS built in function.

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

To remap a code, define it using the `REMAP` option in `RS274NG` section of your INI file. Use one `REMAP` line per remapped code.

Die Syntax von *REMAP* lautet:

**`REMAP=<`*code*`> <`*options*`>`**
: where <*code*> may be one of `T,M6,M61,S,F` (existing codes) or any of the unallocated M-codes or G-codes.

It is an error to omit the <*code*> parameter.

The options of the REMAP statement are separated by whitespace. The options are keyword-value pairs and currently are:

**`modalgroup=<`*modal group*`>`**

**G-Codes**
: the only currently supported modal group is 1, which is also the default value if no group is given. Group 1 means *execute alongside other G-codes*.

**M-Codes**
: Currently supported modal groups are: 5,6,7,8,9,10. If no modalgroup is give, it defaults to 10 (*execute after all other words in the block*).

**T,S,F**
: für diese ist die modale Gruppe festgelegt und die Option `modalgroup=` wird ignoriert.

**`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 is 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 do not 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 (e.g., `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 `F`, `S` and `T` (short `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 is uppercase in the argspec)
(debug, P word=#<P>)
(the q argspec is optional since its lowercase in the argspec. Use as follows:)
o100 if [EXISTS[#<q>]]
    (debug, Q word set: #<q>)
o100 endif
o<m400> endsub
M2
```

- 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)

You'll notice the gradual introduction of the embedded Python environment - see here for details. Note that with Python remapping functions, it make no sense to have Python prolog or epilog functions since it is executing a Python function in the first place.

**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.

The following example is based on the `nc_files/involute.py` script - but canned as a G-code with some parameter extraction and checking. It also demonstrates calling the interpreter recursively (see `self.execute()`).

Angenommen, die Definition lautet wie folgt (Anmerkung: Hier wird argspec nicht verwendet):

`REMAP=G88.1 modalgroup=1 py=involute`

The `involute` function in `python/remap.py` listed below does all word extraction from the current block directly. Note that interpreter errors can be translated to Python exceptions. Remember this is *readahead time* - execution time errors cannot be trapped this way.

```
import sys
import traceback
from math import sin,cos
```

```
from interpreter import *
from emccanon import MESSAGE
from util import lineno, call_pydevd
# raises InterpreterException if execute() or read() fails
throw_exceptions = 1

def involute(self, **words):
    """ remap-Funktion mit Rohzugriff auf Interpreter-Interna """

    if self.debugmask & 0x20000000: call_pydevd() # USER2 debug flag

    if equal(self.feed_rate,0.0):
        return "feedrate > 0 required"

    if equal(self.speed[0], 0.0):
        return "spindle speed > 0 required"

    plunge = 0.1 # if Z word was given, plunge - with reduced feed

    # Kontrollblock auf relevante Wörter untersuchen
    c = self.blocks[self.remap_level]
    x0 = c.x_number if c.x_flag else 0
    y0 = c.y_number if c.y_flag else 0
    a  = c.p_number if c.p_flag else 10
    old_z = self.current_z

    if self.debugmask & 0x10000000:
        print("x0=%f y0=%f a=%f old_z=%f" % (x0,y0,a,old_z))

    try:
        #self.execute("G3456")  # would raise InterpreterException
        self.execute("G21",lineno())
        self.execute("G64 P0.001",lineno())
        self.execute("G0 X%f Y%f" % (x0,y0),lineno())

        if c.z_flag:
            feed = self.feed_rate
            self.execute("F%f G1 Z%f" % (feed * plunge, c.z_number),lineno())
            self.execute("F%f" % (feed),lineno())

        for i in range(100):
            t = i/10.
            x = x0 + a * (cos(t) + t * sin(t))
            y = y0 + a * (sin(t) - t * cos(t))
            self.execute("G1 X%f Y%f" % (x,y),lineno())

        if c.z_flag: # retract to starting height
            self.execute("G0 Z%f" % (old_z),lineno())

    except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
  return msg

    return INTERP_OK
```

Die bisher beschriebenen Beispiele finden Sie in "configs/sim/axis/remap/getting-started" mit vollständigen Arbeitskonfigurationen.

### 9.6.4 Aktualisieren einer bestehenden Konfiguration für die Neuzuordnung

Die Mindestvoraussetzungen für die Verwendung von ”REMAP”-Anweisungen sind wie folgt:

- The Python plug in must be activated by specifying a `[PYTHON]TOPLEVEL=<path-to-toplevel-script>` in the INI file.
- The toplevel script needs to import the `remap` module, which can be initially empty, but the import needs to be in place.
- 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
- Recommended: import the `stdglue` handlers in the `remap` module. In this case Python also needs to find `stdglue.py` - we just copy it from the distribution so you can make local changes as needed. Depending on your installation the path to `stdglue.py` might vary.

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
```

Now edit ``/home/user/`*xxx*``/`*xxx*``.ini`` and add the following:

```
[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).

Note than when remapping an existing code, we completely disable this codes' built-in functionality of the interpreter.

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.

However, this does **not** affect the processing of tool change-related commands in `task` and `iocontrol`. This means when we execute step 6b this will still cause iocontrol to do its thing.

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:

- When using an O-word procedure, we need prolog and epilog functions.
- If using all Python code and no O-word procedure, a Python function is enough.
- When using the `iocontrol` pins, our O-word procedure or Python code will contain mostly moves.
- When we need a more complex interaction than offered by `iocontrol`, we need to completely define our own interaction, using `motion.digital*` and `motion.analog*` pins, and essentially ignore the `iocontrol` pins by looping them.

---

**Anmerkung**
If you hate O-word procedures and love Python, you are 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 numbered 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**
Die inzwischen nicht mehr verwendeten Namen **selected_pocket** und **current_pocket** verweisen auf einen sequentiellen Werkzeugdatenindex für Werkzeugelemente, die aus einer Werkzeugtabelle ([EMCIO]TOOL_TABLE) oder über eine Werkzeugdatenbank ([EMCIO]DB_PROGRAM) geladen werden.

---

```
def change_prolog(self, **words):
    try:
        if self.selected_pocket < 0:
            return "M6: no tool prepared"

        if self.cutter_comp_side:
            return "Cannot change tools with cutter radius compensation on"

        self.params["tool_in_spindle"] = self.current_tool
        self.params["selected_tool"] = self.selected_tool
        self.params["current_pocket"] = self.current_pocket
        self.params["selected_pocket"] = self.selected_pocket
        return INTERP_OK
    except Exception as e:
        return "M6/change_prolog: {}".format(e)
```

You will find that most prolog functions look very similar:

1. First test that all preconditions for executing the code hold, then
2. prepare the environment - inject variables and/or do any preparatory processing steps which cannot easily be done in NGC code;
3. 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, e.g.:
; 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:

1. First, determine whether things went right in the remap procedure,
2. 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 `Tx` (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)
```

The functions *prepare_prolog* and *prepare_epilog* are part of the *standard glue* provided by *nc_files/remap_li*
*stdglue/stdglue.py*. This module is intended to cover most standard remapping situations in a common way.

#### 9.6.5.8 Fehlerbehandlung: Umgang mit Abbrüchen

The built in tool change procedure has some precautions for dealing with a program abort, e.g., by hitting escape in AXIS during a change. Your remapped function has none of this, therefore some explicit cleanup might be needed if a remapped code is aborted. In particular, a remap procedure might establish modal settings which are undesirable to have active after an abort. For instance, if your remap procedure has motion codes (G0,G1,G38..) and the remap is aborted, then the last modal code will remain active. However, you very likely want to have any modal motion canceled when the remap is aborted.

The way to do this is by using the `[RS274NGC]ON_ABORT_COMMAND` feature. This INI option specifies a O-word procedure call which is executed if `task` for some reason aborts program execution. `on_abort` receives a single parameter indicating the cause for calling the abort procedure, which might be used for conditional cleanup.

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**
Never use an M2 in a O-word subroutine, including this one. It will cause hard-to-find errors. For instance, using an M2 in a subroutine will not end the subroutine properly and will leave the subroutine NGC file open, not your main program.

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).

Statements in that procedure typically would assure that post-abort any state has been cleaned up, like HAL pins properly reset. For an example, see 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:

If displaying an operator error message and stopping the current program is good enough, use the (abort, `__<message>__)` feature to terminate the handler with an error message. Note that you can substitute numbered, named, INI and HAL parameters in the text like in this example (see also 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:

- Define an epilog function, even if it is just to signal an error condition,
- pass a negative value from the handler to signal the error,
- inspect the return value in the epilog function,
- take any recovery action needed,
- return the error message string from the handler, which will set the interpreter error message and abort the program (pretty much like `abort, message=`).

Diese Fehlermeldung wird in der Benutzeroberfläche angezeigt, und wenn INTERP_ERROR zurückgegeben wird, dann wird dieser Fehler wie jeder andere Laufzeitfehler behandelt.

Note that both (`abort,` <*msg*> ) and returning INTERP_ERROR from an epilog will cause any ON_ABORT handler to be called as well if defined (see previous section).

## 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.

Besides creating new cycles, this provides an easy method for repackaging existing G-codes which do not behave as cycles. For instance, the G33.1 Rigid Tapping code does not behave as a cycle. With such a wrapper, a new code can be easily created which uses G33.1 but behaves as a cycle.

See *configs/sim/axis/remap/cycle* for a complete example of this feature. It contains two cycles, one with an NGC procedure like above, and a cycle example using just Python.

## 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 = <*filename*>**
Dateiname des anfänglichen Python-Skripts, das beim Starten ausgeführt wird. Dieses Skript ist für die Einrichtung der Paketnamensstruktur verantwortlich, siehe unten.

**PATH_PREPEND = <*directory*>**
Dieses Verzeichnis dem PYTHON_PATH voranstellen. Eine sich wiederholende Gruppe.

**PATH_APPEND = <*directory*>**
Dieses Verzeichnis an PYTHON_PATH anhängen. Eine sich wiederholende Gruppe.

**LOG_LEVEL = <*integer*>**
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 for 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 functions in 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

The interpreter is an existing C++ class (*Interp*) defined in *src/emc/rs274ngc*. Conceptually all `oword.<funct` and `remap.<function>` Python calls are methods of this Interp class, although there is no explicit Python definition of this class (it is a *Boost.Python* wrapper instance) and hence receive the as the first parameter `self` which can be used to access internals.

#### 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).

Wenn das Modul `TOPLEVEL` eine Funktion `__delete__` definiert, wird sie einmal aufgerufen, bevor der Interpreter heruntergefahren wird und nachdem die persistenten Parameter in der `PARAMETER_FILE` gespeichert worden sind.

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 O-word 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/topl 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>**
: Value returned by the last procedure called. Initialized to 0.0 on startup. Exposed in Interp as `self.return_value` (float).

**#<value_returned>**
: Indicates the last procedure called did `return` or `endsub` with an explicit value. 1.0 if true. Set to 0.0 on each `call`. Exposed in Interp was `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**
: Keyword parameter dictionary. If an argspec was present, words are collected from the current block accordingly and passed in the dictionary for convenience (the words could as well be retrieved directly from the calling block, but this requires more knowledge of interpreter internals). If no argspec was passed, or only optional values were specified and none of these was present in the calling block, this dict is empty. Word names are converted to lowercase.

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**
: Return this on success. You need to import this from `interpreter`.

***a message text***
: Returning a string from a handler means *this is an error message, abort the program*. Works like `(abort, <msg> )`.

**Aufrufkonventionen für python=-Unterroutinen** Argumente sind:

**self**
: The interpreter instance.

**words**
Keyword parameter dictionary. The same kwargs dictionary as prologs and epilogs (see above).

Das minimale *python*=-Funktionsbeispiel:

```
def useless(self,  **words): # in remap module
    return INTERP_OK
```

Rückgabewerte:

**INTERP_OK**
Return this on success

***a message text***
Returning a string from a handler means *this is an error message, abort the program*. Works like `(abort, <msg> )`.

If the handler needs to execute a *queuebuster operation* (tool change, probe, HAL pin reading) then it is supposed to suspend execution with the following statement:

**yield INTERP_EXECUTE_FINISH**
This signals `task` to stop read ahead, execute all queued operations, execute the *queue-buster* operation, synchronize interpreter state with machine state, and then signal the interpreter to continue. At this point the function is resumed at the statement following the `yield ..` statement.

**Umgang mit Queue-Buster: Sonde, Werkzeugwechsel und Warten auf einen HAL-Pin** Queue busters interrupt a procedure at the point where such an operation is called, hence the procedure needs to be restarted after the interpreter synch(). When this happens the procedure needs to know if it is restarted, and where to continue. The Python generator method is used to deal with procedure restart.

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 following a yield may not recursively call the interpreter, like with `self.execute("<mdi command>")`. This is an architectural restriction of the interpreter and is not fixable without a major redesign.

#### 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 are using lots of execute() statements, it is probably easier to trap InterpreterException as shown below.

CAUTION:

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
- Die Übergabe von lokalen benannten Parametern ist nicht erforderlich.

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**
: Exposes internals of the Interp class. See `src/emc/rs274ngc/interpmodule.cc`, and the `tests/remap/` regression test.

**emccanon**
: Exposes most calls of `src/emc/task/emccanon.cc`.

**emctask**
: Exposes the `emcStatus` class instance. See `src/emc/task/taskmodule.cc`. Not present when using the `gcode` module used for user interfaces - only present in the milltask instance of the interpreter.

### 9.6.10 Hinzufügen vordefinierter benannter Parameter

The interpreter comes with a set of predefined named parameters for accessing internal state from the NGC language level. These parameters are read-only and global, and hence cannot be assigned to.

Additional parameters may be added by defining a function in the `namedparams` module. The name of the function defines the name of the new predefined named parameter, which now can be referenced in arbitrary expressions.

Um einen benannten Parameter hinzuzufügen oder neu zu definieren:

- Add a `namedparams` module so it can be found by the interpreter,
- define new parameters by functions (see below). These functions receive `self` (the interpreter instance) as parameter and so can access arbitrary state. Arbitrary Python capabilities can be used to return a value.
- Import that module from the `TOPLEVEL` script.

```
# namedparams.py
# trivial example
def _pi(self):
    return 3.1415926535
```

```
#<Umfang> = [2 * #<Radius> * #<_pi>]
```

Functions in `namedparams.py` are expected to return a float or int value. If a string is returned, this sets the interpreter error message and aborts execution.

Es werden nur Funktionen mit führendem Unterstrich als Parameter hinzugefügt, da dies die RS274NGC-Konvention für Globals ist.

It is possible to redefine an existing predefined parameter by adding a Python function of the same name to the `namedparams` module. In this case, a warning is generated during startup.

While the above example isn't terribly useful, note that pretty much all of the interpreter internal state is accessible from Python, so arbitrary predicates may be defined this way. For a slightly more advanced example, see `tests/remap/predefined-named-params`.

### 9.6.11 Standardmäßige Glue (Programmierer-Slang für verbindende)-Routinen

Since many remapping tasks are very similar, I've started collecting working prolog and epilog routines in a single Python module. These can currently be found in *ncfiles/remap_lib/python-stdglue/stdglue.py* and provide the following routines:

#### 9.6.11.1 T: `prepare_prolog` and `prepare_epilog`

These wrap a NGC procedure for T*x* Tool Prepare.

**Aktionen von prepare_prolog** Die folgenden Parameter werden für das NGC-Verfahren sichtbar gemacht:

- `#<tool>` - der Parameter des T-Wortes
- `#<pocket>` - die entsprechende Tasche

Wenn die Werkzeugnummer Null angefordert wird (d.h. Werkzeug entladen), wird die entsprechende Tasche als -1 übergeben.

Es ist ein Fehler, wenn:

- No tool number is given as T parameter,
- das Werkzeug nicht in der Werkzeugtabelle gefunden werden kann.

Note that unless you set the `[EMCIO] RANDOM_TOOLCHANGER=1` parameter, tool and pocket number are identical, and the pocket number from the tool table is ignored. This is currently a restriction.

Actions of `prepare_epilog`

- The NGC procedure is expected to return a positive value, otherwise an error message containing the return value is given and the interpreter aborts.
- In case the NGC procedure executed the T command (which then refers to the built in T behavior), no further action is taken. This can be used for instance to minimally adjust the built in behavior be preceding or following it with some other statements.
- Otherwise, the `#<tool>` and `#<pocket>` parameters are extracted from the subroutine's parameter space. This means that the NGC procedure could change these values, and the epilog takes the changed values in account.
- Then, the Canon command `SELECT_TOOL(#<tool>)` is executed.

#### 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.
  - if parameter 5601 (error code) is greater equal zero, this indicates a soft fault. An informational message is displayed and the prolog continues.
- 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

- The NGC procedure is expected to return a positive value, otherwise an error message containing the return value is given and the interpreter aborts.
- 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.
- In case the NGC procedure executed the M6 command (which then refers to the built in M6 behavior), no further action is taken. This can be used for instance to minimally adjust the built in behavior be preceding or following it with some other statements.
- Otherwise, the #<selected_pocket> parameter is extracted from the subroutine’s parameter space, and used to set the interpreter’s current_pocket variable. Again, the procedure could change this value, and the epilog takes the changed value in account.
- 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

These wrap a NGC procedure so it can act as a cycle, meaning the motion code is retained after finishing execution. If the next line just contains parameter words (e.g. new X,Y values), the code is executed again with the new parameter words merged into the set of the parameters given in the first invocation.

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 G-code/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 |

*or* these flags into the `[EMC]DEBUG` variable as needed. For a current list of debug flags see *src/emc/nml_intf/d*

#### 9.6.12.5 Fehlersuche in eingebettetem Python-Code

Debugging of embedded Python code is harder than debugging normal Python scripts, and only a limited supply of debuggers exists. A working open-source based solution is to use the Eclipse IDE, and the PydDev Eclipse plug in and its remote debugging feature.

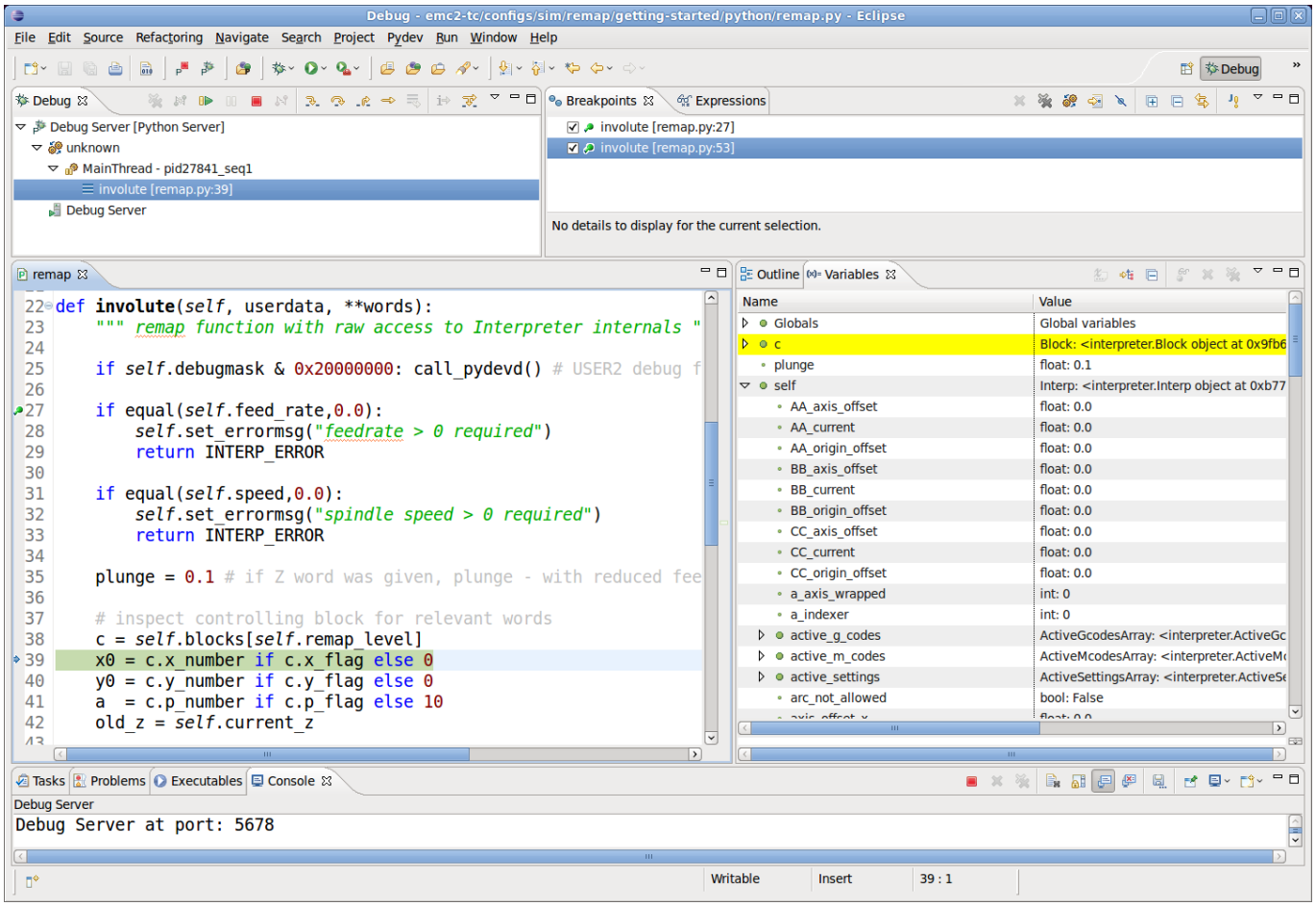
Um diesen Ansatz zu verwenden:

- Installieren Sie Eclipse über das *Ubuntu Software Center* (wählen Sie die erste Option).
- Install the PyDev plug in from the 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 is 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 is 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 | | | | | | | | |
| *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 | | | | | | | |

Tabelle 9.8: (continued)

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *72* | G72 | G72.1 | G72.2 | | | | | | | |
| *73* | | | | | | | | | | |
| *74* | | | | | | | | | | |
| *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:

These M-codes are currently undefined in the current implementation of LinuxCNC and may be used to define new M-codes. (Developers who define new M-codes in LinuxCNC are encouraged to remove them from this table.)

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 is 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)

Once a line is read (in either MDI mode, or from the current NGC file), it is parsed and flags and parameters are set in a *struct block* (struct _setup, member block1). This struct holds all information about the current source line, but independent of different ordering of codes on the current line: As long as several codes are compatible, any source ordering will result in the same variables set in the struct block. Right after parsing, all codes on a block are checked for compatibility.

#### 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.

If a "queue buster" is found, a corresponding flag is set in the interpreter state (toolchange_flag, input_flag, probe_flag) and the interpreter returns an INTERP_EXECUTE_FINISH return value, signaling *stop readahead for now, and resynch* to the caller (*`task`*). If no queue busters are found after all items are executed, INTERP_OK is returned, signalling that read-ahead may continue.

When read ahead continues after the synch, `task` starts executing interpreter read() operations again. During the next read operation, the above mentioned flags are checked and corresponding variables are set (because the a synch() was just executed, the values are now current). This means that the next command already executes in the properly set variable context.

#### 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

The actions happening in LinuxCNC are a bit involved, but it is necessary to get the overall idea what currently happens, before you set out to adapt those workings to your own needs.

Note that remapping an existing code completely disables all internal processing for that code. That means that beyond your desired behavior (probably described through an NGC O-word or Python procedure), you need to replicate those internal actions of the interpreter, which together result in a complete replacement of the existing code. The prolog and epilog code is the place to do this.

**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.

Tool information is held in the *emcStatus* structure, which is shared by all parties. One of its fields is the *toolTable* array, which holds the description as loaded from the tool table file (tool number, diameter, frontangle, backangle and orientation for lathe, tool offset information).

The authoritative source and only process actually *setting* tool information in this structure is the *iocontrol* process. All others processes just consult this structure. The interpreter holds actually a local copy of the tool table.

For the curious, the current emcStatus structure can be accessed by Python statements. The interpreter’s perception of the tool currently loaded for instance is accessed by:

```
;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 How T*x* (Prepare Tool) works

**Interpreter action on a Tx command**

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.

When the changer responds by asserting ”tool-prepared”, it considers the prepare phase to be completed and signals `task` to continue. Again, this *wait* is not strictly necessary IMO.

**Building the prolog and epilog for Tx** See the Python functions `prepare_prolog` and `prepare_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.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.

Auch wenn Sie mit C nicht vertraut sind, empfehle ich Ihnen, sich den Code von ”Interp::convert_tool_change in ”src/emc/rs274/interp_convert.cc” anzusehen.

**Interpreteraktion bei einem M6-Befehl**

Wenn der Interpreter ein M6 sieht, wird er:

1. 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. prüfen, ob die Fräserradiuskompensation aktiv ist, und meldet in diesem Fall ”Werkzeugwechsel mit eingeschalteter Fräserradiuskompensation nicht möglich”.
3. die Spindel anhalten, außer wenn die INI-Option ”TOOL_CHANGE_WITH_SPINDLE_ON” gesetzt ist.
4. eine schnelle Z-Aufwärtsbewegung erzeugen, wenn die INI-Option ”TOOL_CHANGE_QUILL_UP” gesetzt ist.
5. if TOOL_CHANGE_AT_G30 was set:
   a. Indexer A, B und C verschieben wenn durchführbar
   b. eine schnelle Bewegung zur G30-Position erzeugen
6. execute a CHANGE_TOOL canon command, with the selected pocket as a parameter. CHANGE_TOOL will:
   a. eine schnelle Bewegung zur TOOL_CHANGE_POSITION erzeugen, wenn dies in der INI eingestellt ist
   b. eine EMC_TOOL_LOAD-NML-Nachricht in die Warteschlange für `task` stellen.
7. 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.

**Building the prolog and epilog for M6** See the Python functions `change_prolog` and `change_epilog` in `nc_files/remap_lib/python-stdglue/stdglue.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 `nc_files/remap_lib/python-stdglue/stdglue.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 O-word 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 Component

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.

Waypoints are recorded when the moveoff component is enabled. Waypoints are managed with the waypoint-sample-secs and waypoint-threshold pins. When the backtrack-enable pin is TRUE, the autoreturn path follows the recorded waypoints. When the memory available for waypoints is exhausted, offsets are frozen and the waypoint-limit pin is asserted. This restriction applies regardless of the state of the backtrack-enable pin. An enabling pin must be deasserted to allow a return to the original (non-offset position).

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 External Axis Offsets

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.eoffset-limited* aktiviert und der Achsenwert nominal auf der weichen Grenze gehalten. Dieser HAL-Pin kann von der zugehörigen HAL-Logik verwendet werden, um weitere E-Offset-Zählungen abzuschneiden oder die Maschine anzuhalten (z. B. durch Anschluss an *halui.machine.off*). Wird die Achse innerhalb des Softlimits bewegt, so wird der *motion.eoffset-limited*-Pin zurückgesetzt.

Beim Betrieb an einer weichen Grenze während einer koordinierten Bewegung, die den geplanten Achsenwert weiter verändert, zeigt der HAL-Ausgangspin *axis.L.eoffset* den aktuellen Offset an - die Distanz, die benötigt wird, um die Grenze zu erreichen, anstatt der berechneten Offset-Anforderung. Dieser angezeigte Wert ändert sich, wenn sich der geplante Achsenwert ändert.

Der HAL-Pin *axis.L.eoffset-request* zeigt den aktuellen angeforderten Offset an, der das Produkt aus dem internen Zählregister und der eoffset-Skala ist. Im Allgemeinen hinkt der Wert des Pins *axis.L.eoffset* dem Wert von *axis.L.eoffset-request* hinterher, da der externe Offset einer Beschleunigungsgrenze unterliegt. Beim Betrieb an einer weichen Grenze wirken sich zusätzliche Aktualisierungen der *axis.L.eoffset-counts* weiterhin auf den angeforderten externen Offset aus, wie er im *axis.L.eoffset-request*-HAL-Pin reflektiert wird.

Beim Teleop-Jogging mit aktivierten externen Offsets **und** angewandten Werten ungleich Null wird bei Erreichen eines Soft-Limits die Bewegung in der betreffenden Achse **ohne Verzögerungsintervall** angehalten. In ähnlicher Weise wird bei einer koordinierten Bewegung mit aktivierten externen Offsets das Erreichen eines Soft-Limits zum Anhalten der Bewegung ohne Verzögerungsphase führen. In diesem Fall spielt es keine Rolle, ob die Offsets Null sind.

Wenn die Bewegung ohne Verzögerungsphase gestoppt wird, können die **Beschleunigungsgrenzen des Systems verletzt werden**, was zu Folgefehlern führt: 1) einem Schleppfehler (und/oder einem Klopfen) bei einem Servomotor-System, 2) einem Verlust von Schritten bei einem Schrittmotor-System. Im Allgemeinen wird empfohlen, externe Offsets so zu verwenden, dass eine Annäherung an die weichen Grenzen vermieden wird.

#### 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. Upon a successful acknowledgement, LinuxCNC issues a *g* (**get**) command to request all tools. The **db_program** must respond with a sequence of replies to identify each available tool. The textual reply format is identical to the text line format used in conventional tool table files. A final response of ”FINI” terminates the reply.
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 loading until interp is 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

AXIS is a graphical front-end for LinuxCNC which features a live preview and backplot. It is written in Python and uses Tk and OpenGL to display its user interface.

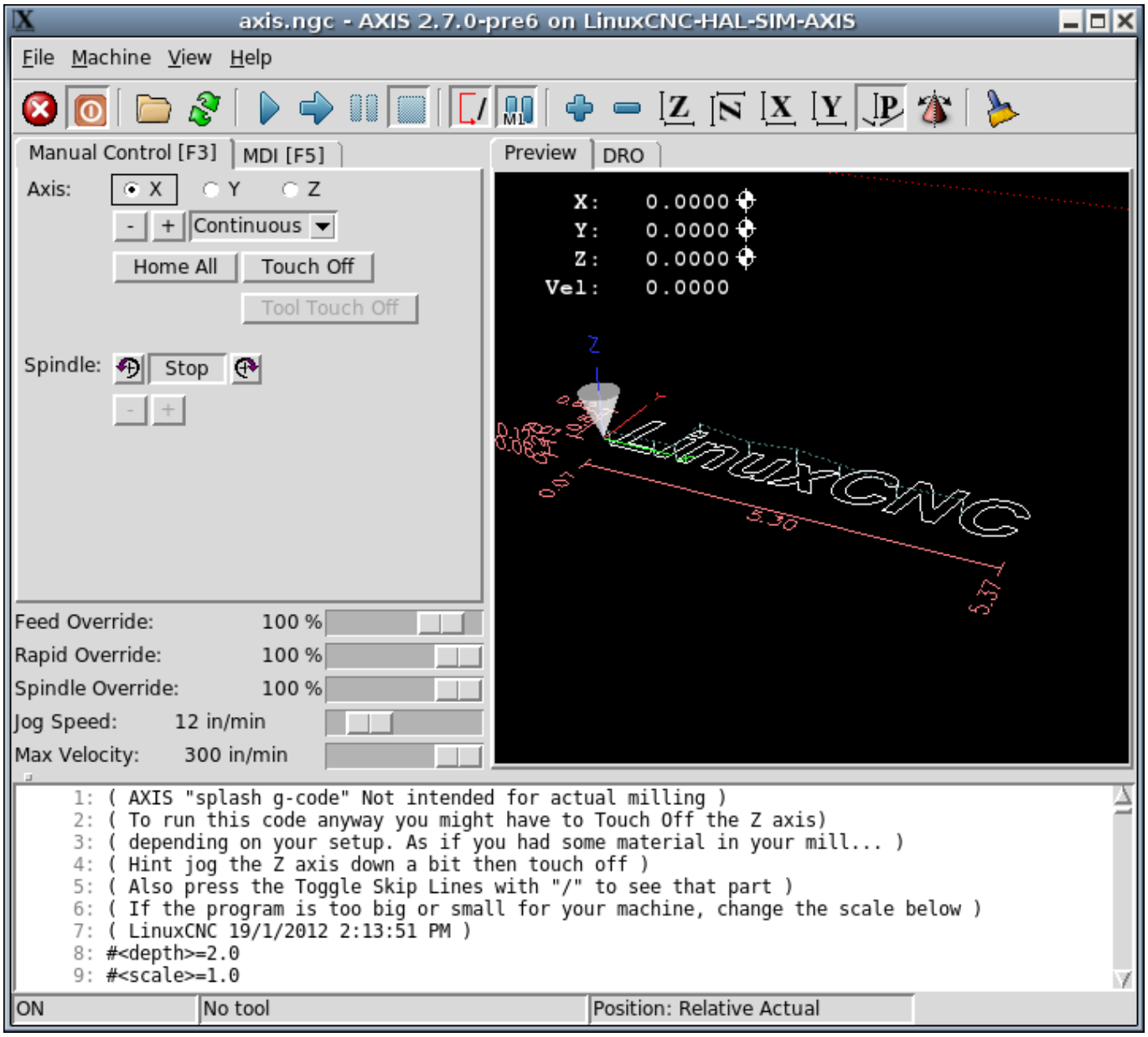



Abbildung 10.1: Das AXIS-Fenster

### 10.1.1 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.

#### 10.1.1.1 INI-Einstellungen

For more information on INI file settings that can change how AXIS works see the Display Section and the Axis Section of the INI Configuration Chapter.

- *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
```

#### 10.1.1.2 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.

---

### 10.1.2 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).

#### 10.1.2.1 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.
- Tool 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.

#### 10.1.2.2 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]

#### 10.1.2.3 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

**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.

#### 10.1.2.4 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.

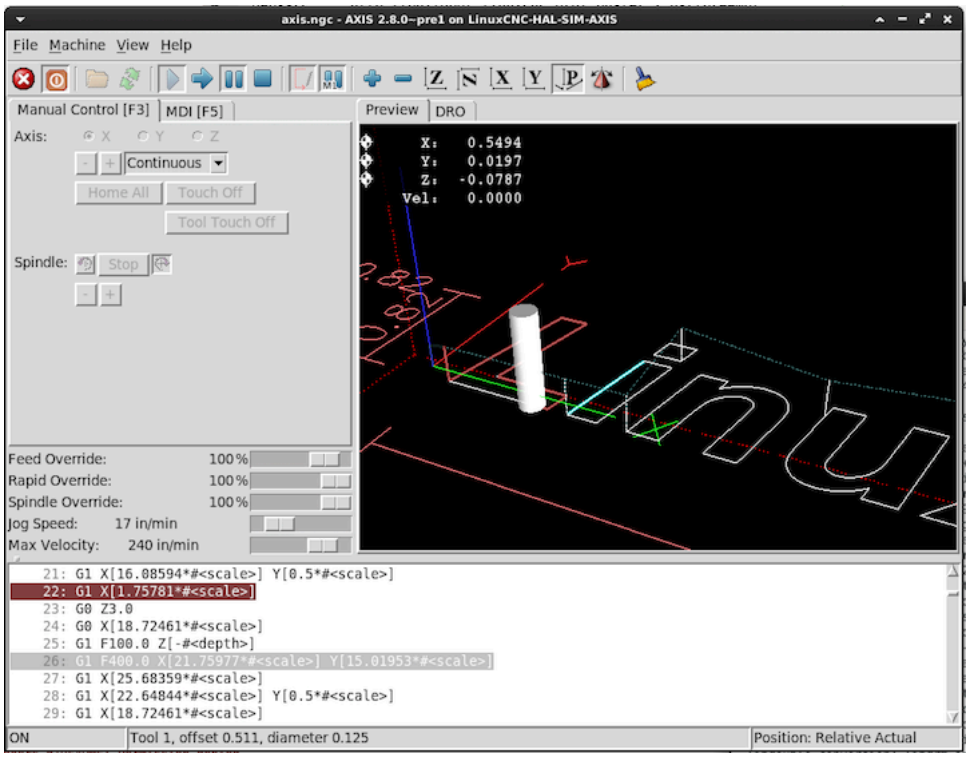



Abbildung 10.3: Aktuelle und ausgewählte Zeilen

#### 10.1.2.5 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.

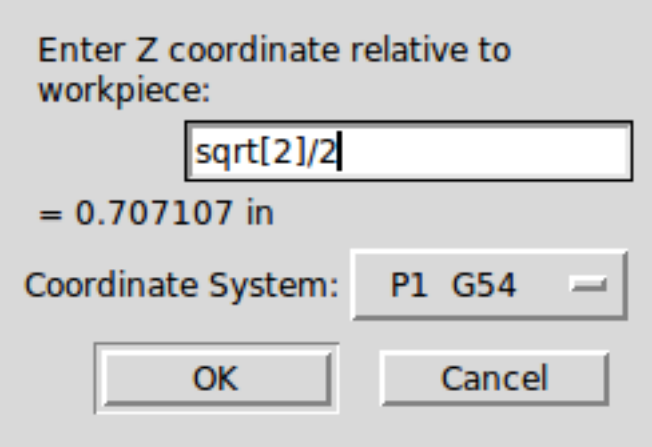



Abbildung 10.4: Touch Off Fenster

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.

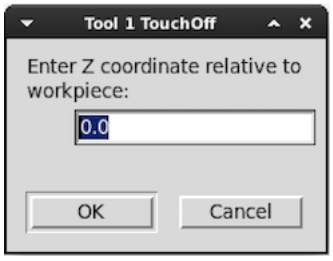



Abbildung 10.5: Werkzeug Touch Off Fenster

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.

#### 10.1.2.6 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.

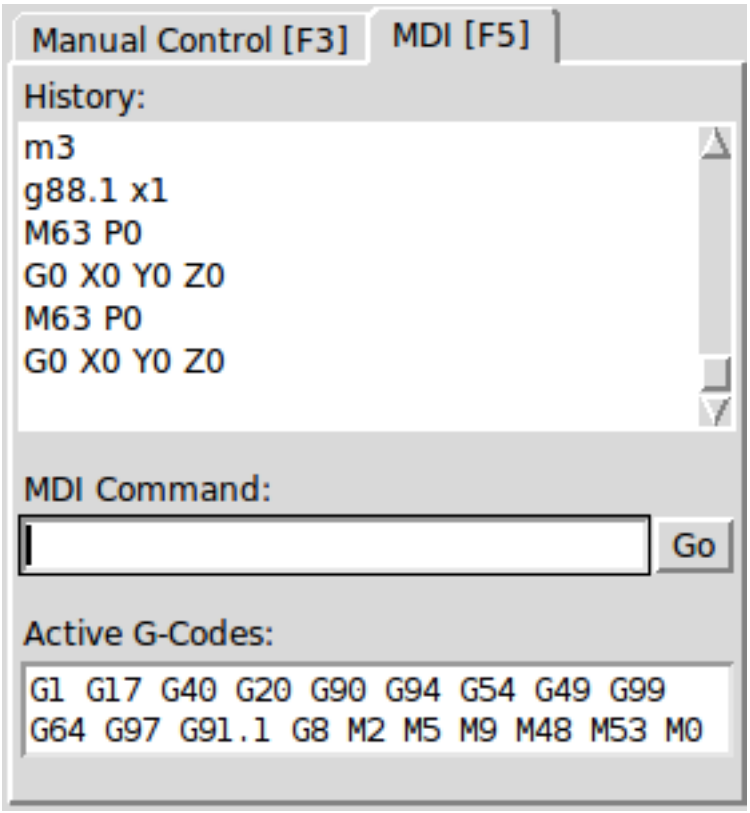



Abbildung 10.6: Die MDI-Registerkarte

- *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.

#### 10.1.2.7 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.

#### 10.1.2.8 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.

#### 10.1.2.9 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.

#### 10.1.2.10 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.

### 10.1.3 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.

#### 10.1.3.1 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: Most Common Keyboard Shortcuts

| Tastenkombination | Ergriffene Maßnahmen | Modus |
|---|---|---|
| F1 | Notaus ein-/ausschalten | Jede (engl. any) |
| F2 | Maschine ein-/ausschalten | Jede (engl. 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 | Jede (engl. any) |
| # | Umschalten Relativ/Maschine | Jede (engl. any) |

### 10.1.4 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

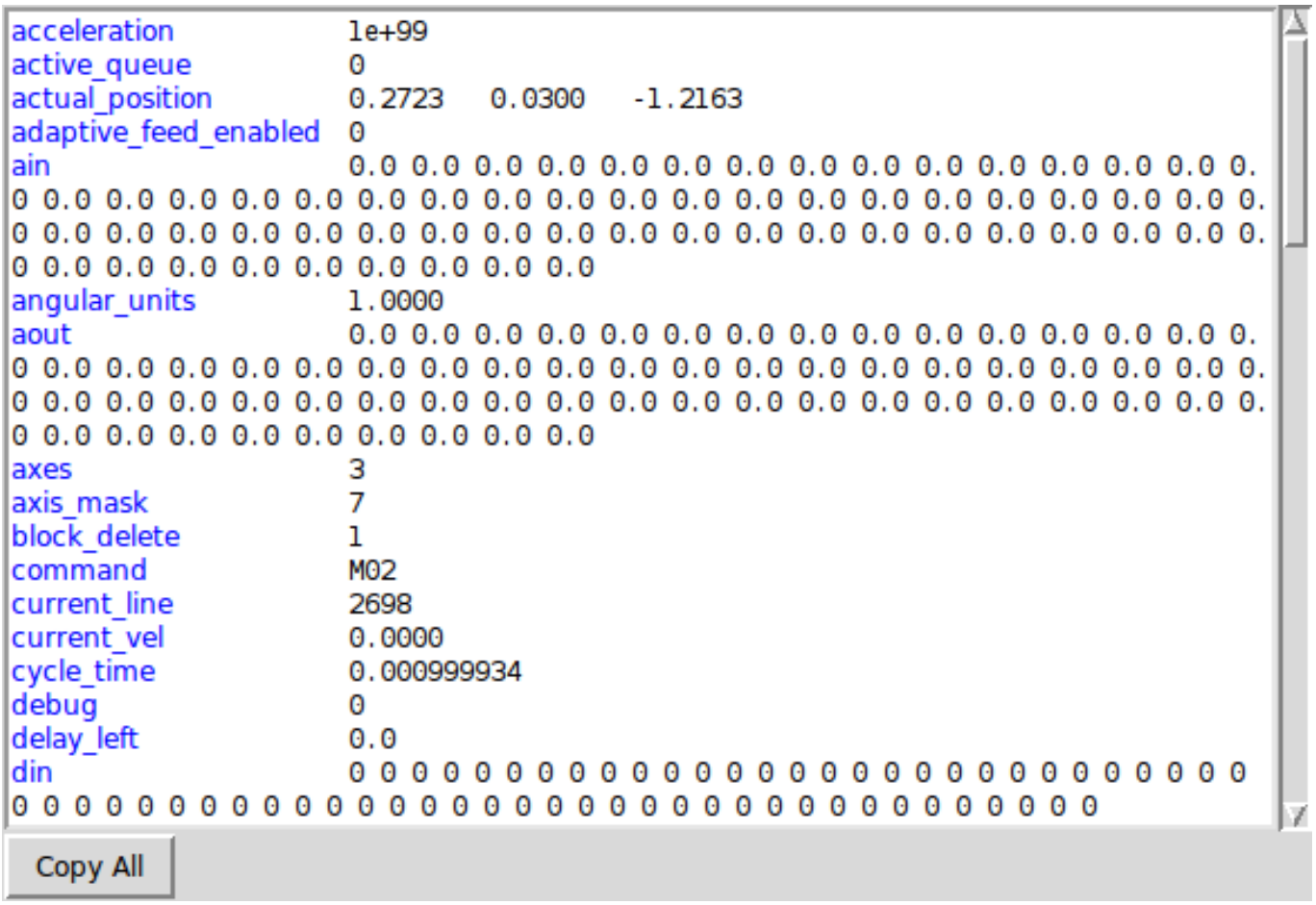



Abbildung 10.7: LinuxCNC-Statusfenster

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.

### 10.1.5 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)
```

### 10.1.6 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*).

### 10.1.7 Manueller Werkzeugwechsel

LinuxCNC includes a non-realtime HAL component called *hal_manualtoolchange*, which shows a window prompt telling you what tool is expected when a *M6* command is issued. After the OK button is pressed, execution of the program will continue.

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>.

---

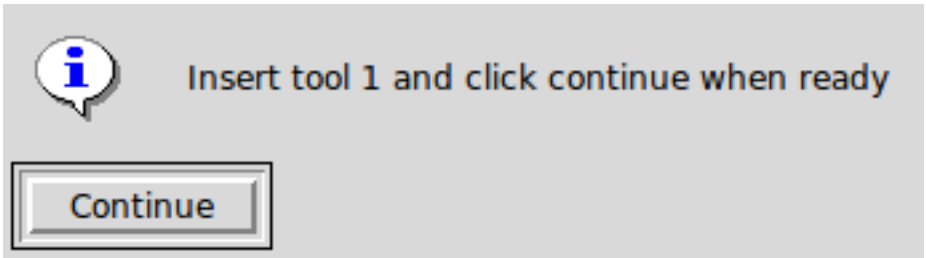



Abbildung 10.8: Fenster für Manuellen Werkzeugwechsel

### 10.1.8 Python modules

AXIS includes several Python modules which may be useful to others. For more information on one of these modules, use *pydoc <module name>* or read the source code. These modules include:

- 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* allows the creation of non-realtime HAL components written in Python
- *_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.

### 10.1.9 Using AXIS in Lathe Mode

By including the line *LATHE = 1* in the [DISPLAY] section of the INI file, AXIS selects lathe mode. The *Y* axis is not shown in coordinate readouts, the view is changed to show the Z axis extending to the right and the X axis extending towards the bottom of the screen, and several controls (such as those for preset views) are removed. The coordinate readouts for X are replaced with diameter and radius.

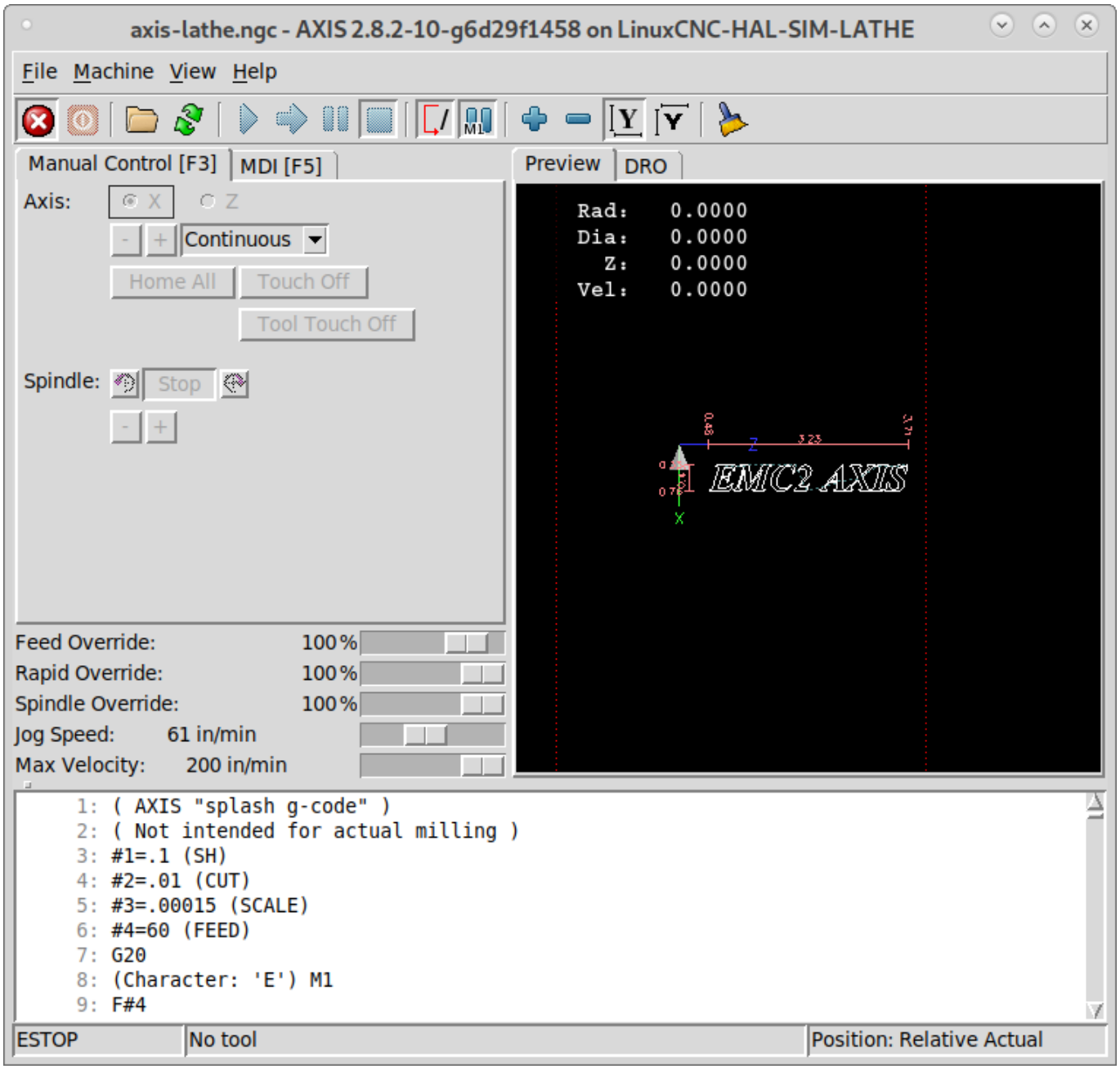



Abbildung 10.9: AXIS-Drehmaschinenmodus

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.

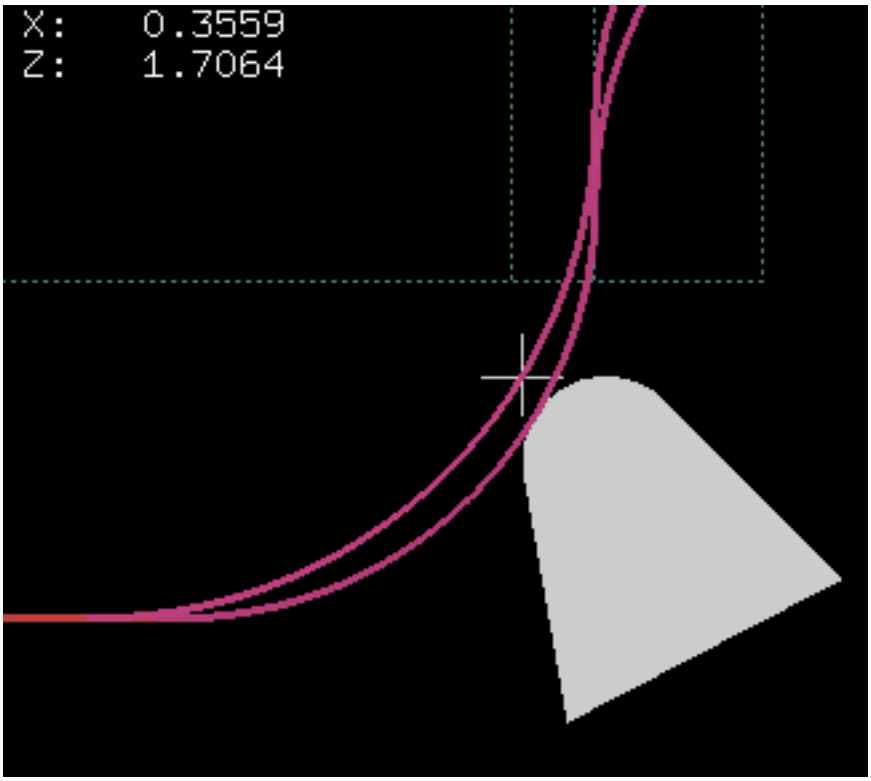



Abbildung 10.10: Drehwerkzeug-Form

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.

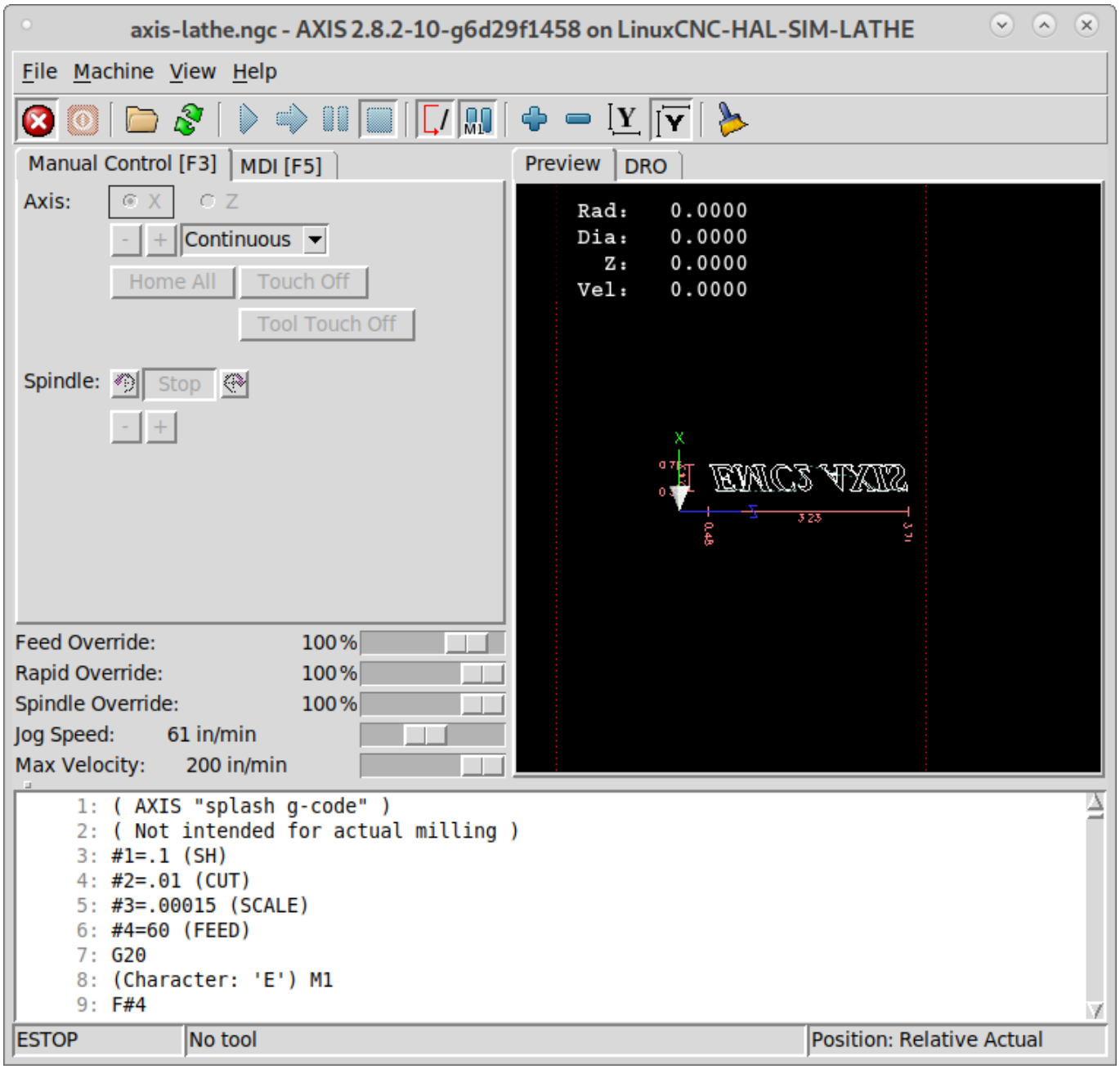



Abbildung 10.11: Lathe Back Tool Shape

### 10.1.10 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.

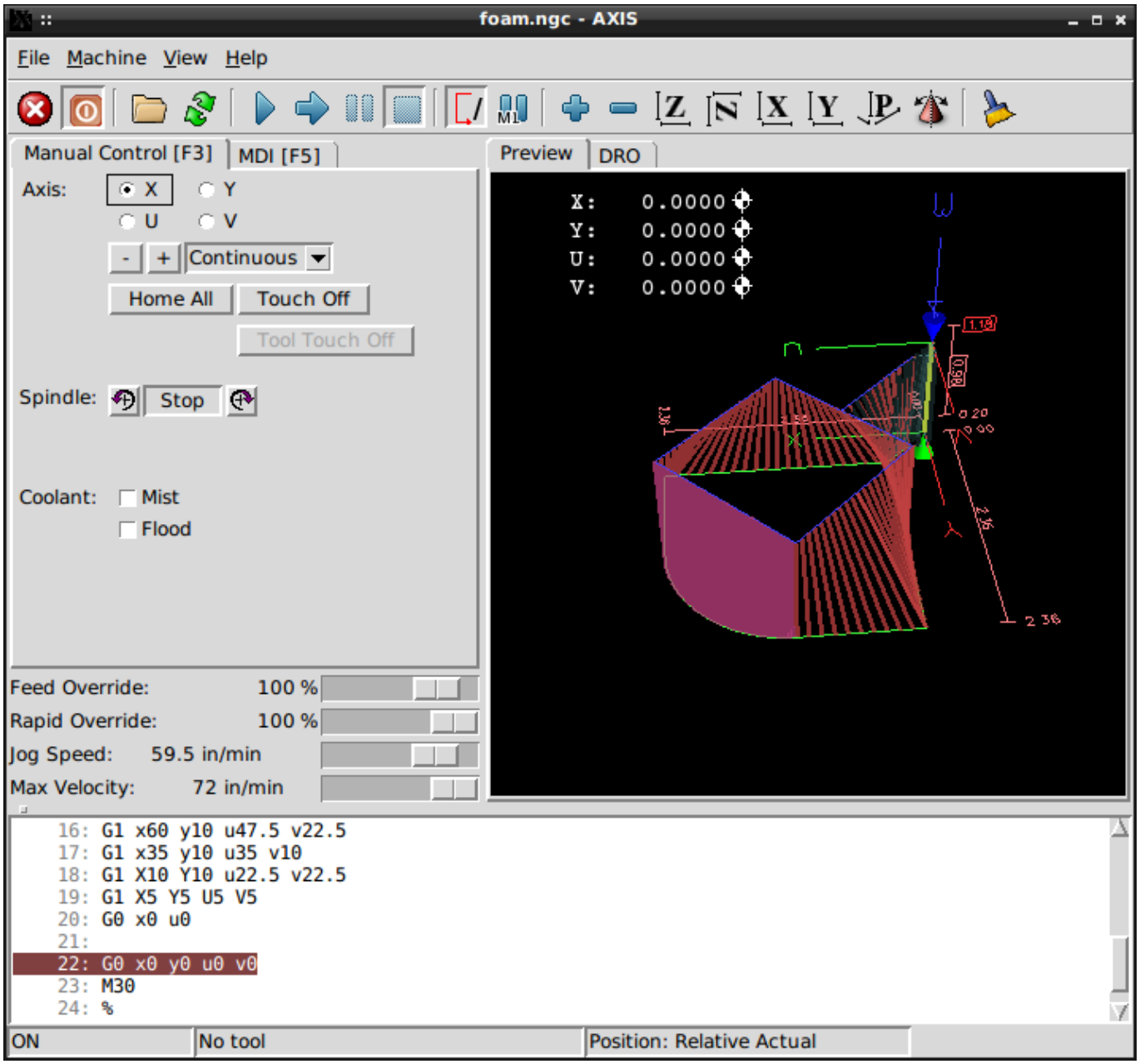



Abbildung 10.12: Modus Schaumstoffschneiden

### 10.1.11 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.

#### 10.1.11.1 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.

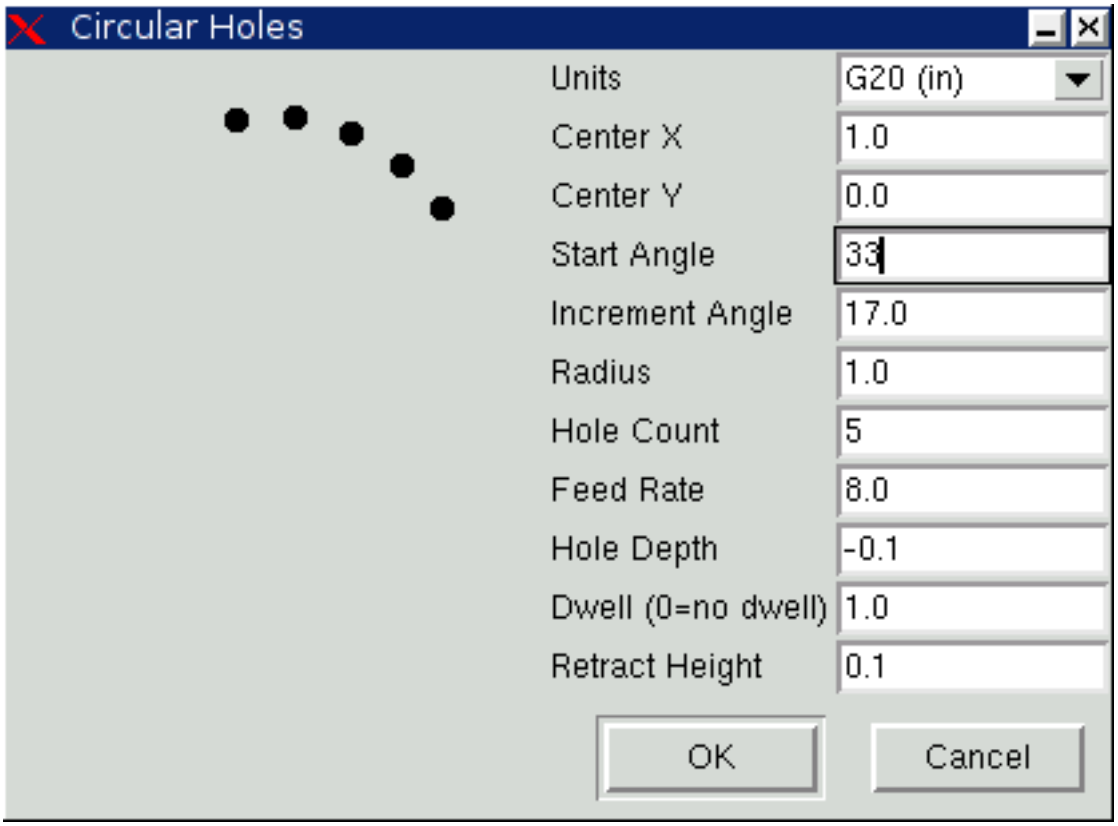



Abbildung 10.13: Kreisförmige Löcher

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.

#### 10.1.11.2 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.

#### 10.1.11.3 Jogwheel

To improve the interaction of AXIS with a physical jogwheel, the current active axis selected in the GUI is also reported on a *HAL pin* with a name like *axisui.jog.x*. Except for a short time after the current axis has changed, only one of these pins at a time is *TRUE*, the others remain *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.

#### 10.1.11.4 ~/.axisrc

If it exists, the contents of `~/.axisrc` are executed as Python source code just before the AXIS GUI is displayed. The details of what may be written in the `~/.axisrc` are subject to change during the development cycle.

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 .”)
```

#### 10.1.11.5 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.

#### 10.1.11.6 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.

#### 10.1.11.7 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.

#### 10.1.11.8 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.

#### 10.1.11.9 Virtual Control Panel

AXIS can display a custom virtual control panel in either the right side column or the bottom row. Additionally one or more panels may be displayed as embedded tabs. You can program buttons, indicators, data displays and more. For more information, see the PyVCP and the GladeVCP chapters.

#### 10.1.11.10 Vorschau Kontrolle

Special comments can be inserted into the G-code file to control how the preview of AXIS behaves. In the case where you want to limit the drawing of the preview use these special comments. Anything between the (AXIS,hide) and (AXIS,show) will not be drawn during the preview. The (AXIS,hide) and (AXIS,show) must be used in pairs with the (AXIS,hide) being first. Anything after a (AXIS,stop) will not be drawn during the preview.

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.

### 10.1.12 Axisui

To improve the interaction of AXIS with physical jog wheels, the axis currently selected in the GUI is also reported on a pin with a name like *axisui.jog.x*. One of these pins is *TRUE* at one time, and the rest are *FALSE*. These are meant to control motion's jog-enable pins.

**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
```

### 10.1.13 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.

#### 10.1.13.1 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...')
```

#### 10.1.13.2 Deaktivieren des Schließen-Dialogs

```
# Deaktivieren Sie den "Do you want to close"-Dialog
root_window.tk.call("wm", "protocol",".", "WM_DELETE_WINDOW", "destroy .")
```

#### 10.1.13.3 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')
```

#### 10.1.13.4 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))
```

#### 10.1.13.5 Lesen der INI-Datei

```
# INI-Dateielement lesen
machine = inifile.find(’EMC’,’MACHINE’)
print(’machine name =’,machine)
```

#### 10.1.13.6 Read LinuxCNC Status

```
# LinuxCNC status can be read from s.
print(s.actual_position)
print(s.paused)
```

#### 10.1.13.7 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()
```

#### 10.1.13.8 Erstellen neuer AXISUI HAL-Pins

```
def user_hal_pins():
    comp.newpin(’my-new-in-pin’, hal.HAL_BIT, hal.HAL_IN)
    comp.ready()
```

#### 10.1.13.9 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)
```

#### 10.1.13.10 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
```

#### 10.1.13.11 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)
```

#### 10.1.13.12 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)
```

### 10.1.13.13 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
```

#### 10.1.13.14 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')
```

#### 10.1.13.15 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')
```

#### 10.1.13.16 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)
```

#### 10.1.13.17 Ändern Sie die DRO-Farbe

```
# dro-Bildschirm ändern
root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','-
    background','black')
```

#### 10.1.13.18 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)
```

#### 10.1.13.19 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

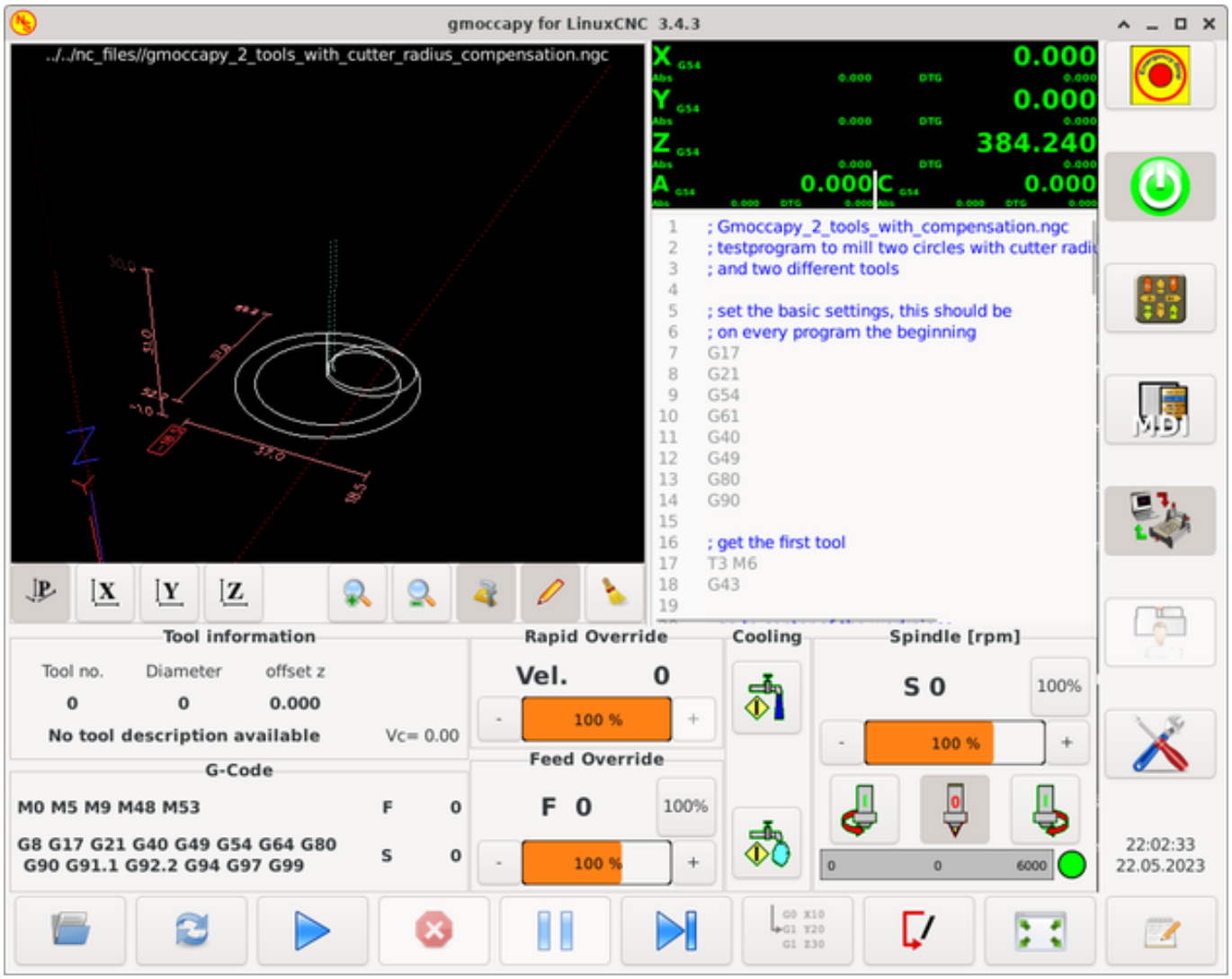
GMOCCAPY is a GUI for LinuxCNC, designed to be used with a touch screen, but can also be used on normal screens with a mouse or hardware buttons and MPG wheels, as it presents HAL Pins for the most common needs. Please find more information in the following.

It offers the possibility to display up to 9 axes, support a lathe mode for normal and back tool lathe and can be adapted to nearly every need, because GMOCCAPY supports embedded tabs and side panels. As a good example for that see gmoccapy_plasma.

GMOCCAPY 3 does support up to 9 axes and 9 joints. As GMOCCAPY 3 has been changed in code to support the joint / axis changes in LinuxCNC it does not work on 2.7 or 2.6 branch!

It has support for integrated virtual keyboard (onboard or matchbox-keyboard), so there is no need for a hardware keyboard or mouse, but it can also be used with that hardware. GMOCCAPY offers a separate settings page to configure most settings of the GUI without editing files.

GMOCCAPY can be localized very easy, because the corresponding files are separated from the linuxcnc.po files, so there is no need to translate unneeded stuff. The files are placed in **/src/po/gmoccapy**. You could just copy the gmoccapy.pot file to something like it.po and translate that file with gtranslator or poedit. After rebuilding, you'd get the GUI in your preference language. To facilitate the sharing of the translation, GMOCCAPY is available on the Weblate web interface. GMOCCAPY is currently available in English, German, Spanish, Polish, Serbian and Hungarian. Feel free to help me to introduce more languages, be it locally or via the web. If you need help, don't hesitate to contact me on **nieson@web.de**.




## 10.2.2 Anforderungen

GMOCCAPY 3 has been tested on Debian Jessie, Debian Stretch and MINT 18 with LinuxCNC master and 2.8 release. It fully support joint / axis changes of LinuxCNC, making it suitable as GUI for Scara, Robots or any other config with more joints than axes. So it supports also gantry configs. If you use other versions, please inform about problems and / or solutions on the LinuxCNC forum or the German CNC Ecke Forum or LinuxCNC users mailing list.

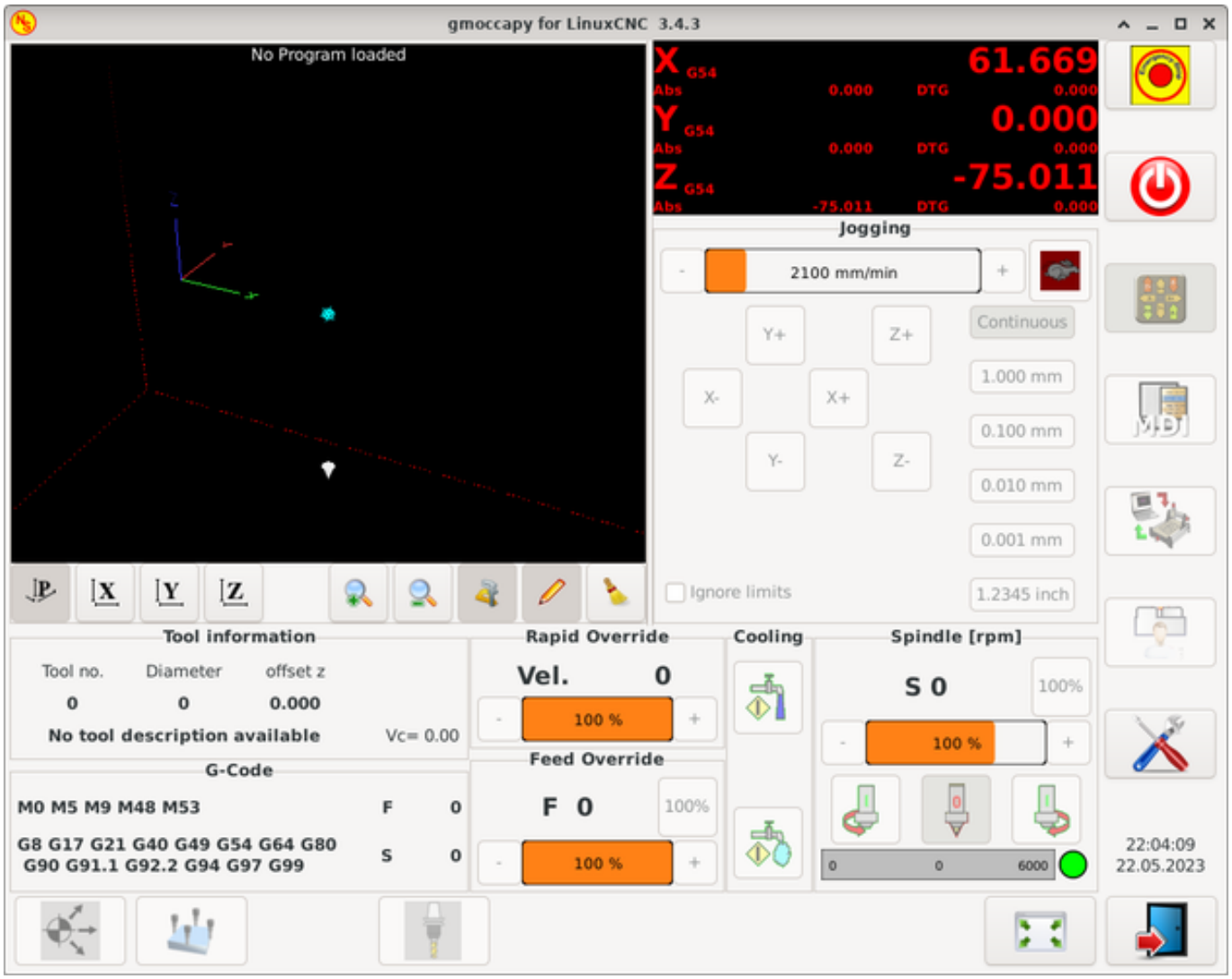
The minimum screen resolution for GMOCCAPY for the normal layout (without side panels) is **980 x 750 Pixel**, so it should fit to every standard screen. It is recommended to use screens with minimum resolution of 1024x768. There is also a configuration which fits for 800x600 screens (introduced in GMOCCAPY 3.4.8).

## 10.2.3 How to get GMOCCAPY

GMOCCAPY 3 is included in the standard distribution of LinuxCNC since release 2.7. 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.

In the **release notes** aka changelist you can track the latest bugfixes and features.

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.

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
DEFAULT_SPINDLE_SPEED = 500
LATHE = 1
BACK_TOOL_LATHE = 1
PROGRAM_PREFIX = ../../nc_files/
```

- *DISPLAY = gmoccapy* - Damit wird LinuxCNC angewiesen, GMOCCAPY zu verwenden.
- *PREFERENCE_FILE_PATH* - 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.

- *DEFAULT_SPINDLE_SPEED* - Start value for ”Starting RPM” if value not present in preferences file or file is not present. Will have no effect with valid preferences file.
- *MIN_ANGULAR_VELOCITY* - Sets the minimal jog velocity of the machine for rotary axes.
- *MAX_ANGULAR_VELOCITY* - Sets the maximal jog velocity of the machine for rotary axes.
- *DEFAULT_ANGULAR_VELOCITY* - Sets the default jog velocity of the machine for rotary axes.

#### 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. This value will also be the maximum linear jog velocity.

  **Anmerkung**
  Der Standardwert ist 600, falls nicht festgelegt.

#### 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 separated ":".

Dann müssen Sie nur noch einen Abschnitt wie diesen hinzufügen:

**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.
- Die Dateien müssen in einem Ordner abgelegt werden, der in Ihrer INI-Datei durch *SUBROUTINE_PATH* im Abschnitt RS274NGC angegeben ist

Der Code zwischen sub und endsub wird durch Betätigen der entsprechenden Makrotaste ausgeführt.

**Anmerkung**
A maximum of 16 macros will be shown in the GUI. Due to space reasons you may need to click on an arrow to switch the page and display hidden macro buttons. The macro buttons will be displayed in the order of the INI entries. It is no error placing more than 16 macros in your INI file, they will just not be shown.

**Anmerkung**
You will find the sample macros in a folder named *macros* placed in the GMOCCAPY sim folder. If you have given several subroutine paths, they will be searched in the order of the given paths. The first file found will be used.

GMOCCAPY akzeptiert auch Makros, die nach Parametern wie den folgenden fragen:

```
[MACROS]
MACRO = go_to_position X-pos Y-pos Z-pos
```

Die Parameter müssen durch Leerzeichen getrennt werden. Dieses Beiaspiel ruft eine Datei ”go_to_position.n
mit dem folgenden Inhalt auf:

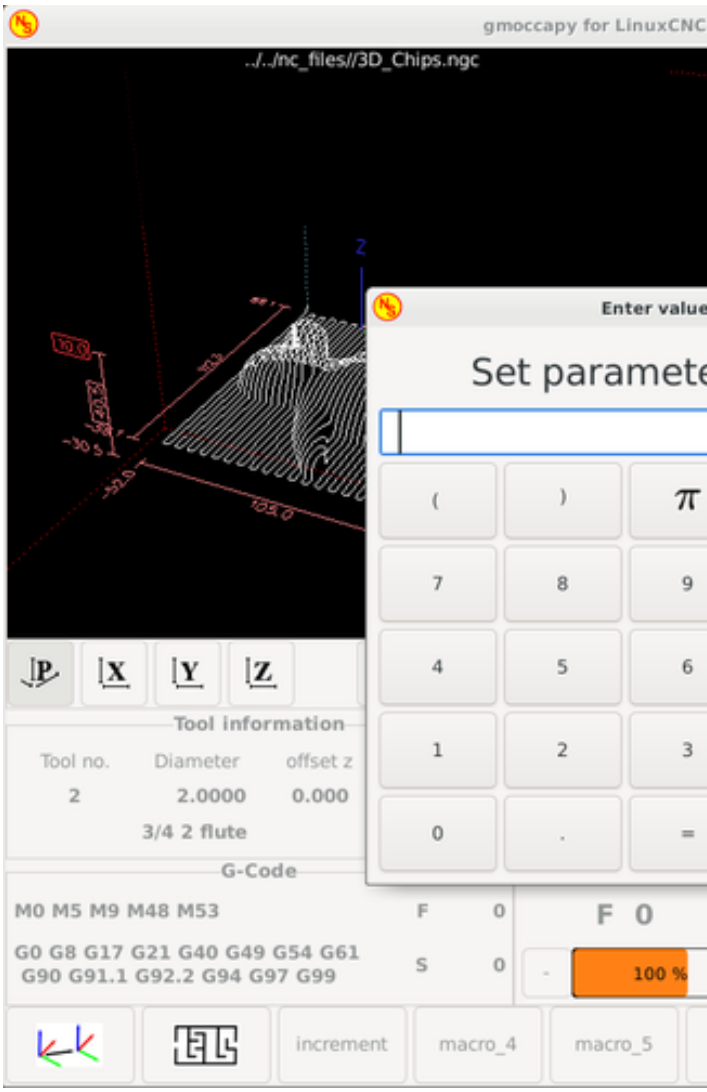
```
; Testdatei ”go to position” (engl. für geh’ zur Position)
; fährt die Maschine zu einer bestimmten Position

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.

---




**Makrobeispiel mit dem ”Gehe zu Position”-Makro**

### 10.2.4.4 Embedded Tabs and Panels

You can add embedded programs to GMOCCAPY like you can do in AXIS, Touchy and Gscreen. All is done by GMOCCAPY automatically if you include a few lines in your INI file in the DISPLAY section.

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)
  - **box_dro_side** (will introduce your glade file right of the DRO)

---

**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
```

includes a custom glade file called dro.glade in the mentioned location. The file must be placed in the config folder of your machine.

```
gladevcp h_buttonlist.glade
```

will just open a new user window called h_buttonlist.glade note the difference. This one is stand alone, and can be moved around independent from GMOCCAPY window.

```
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.

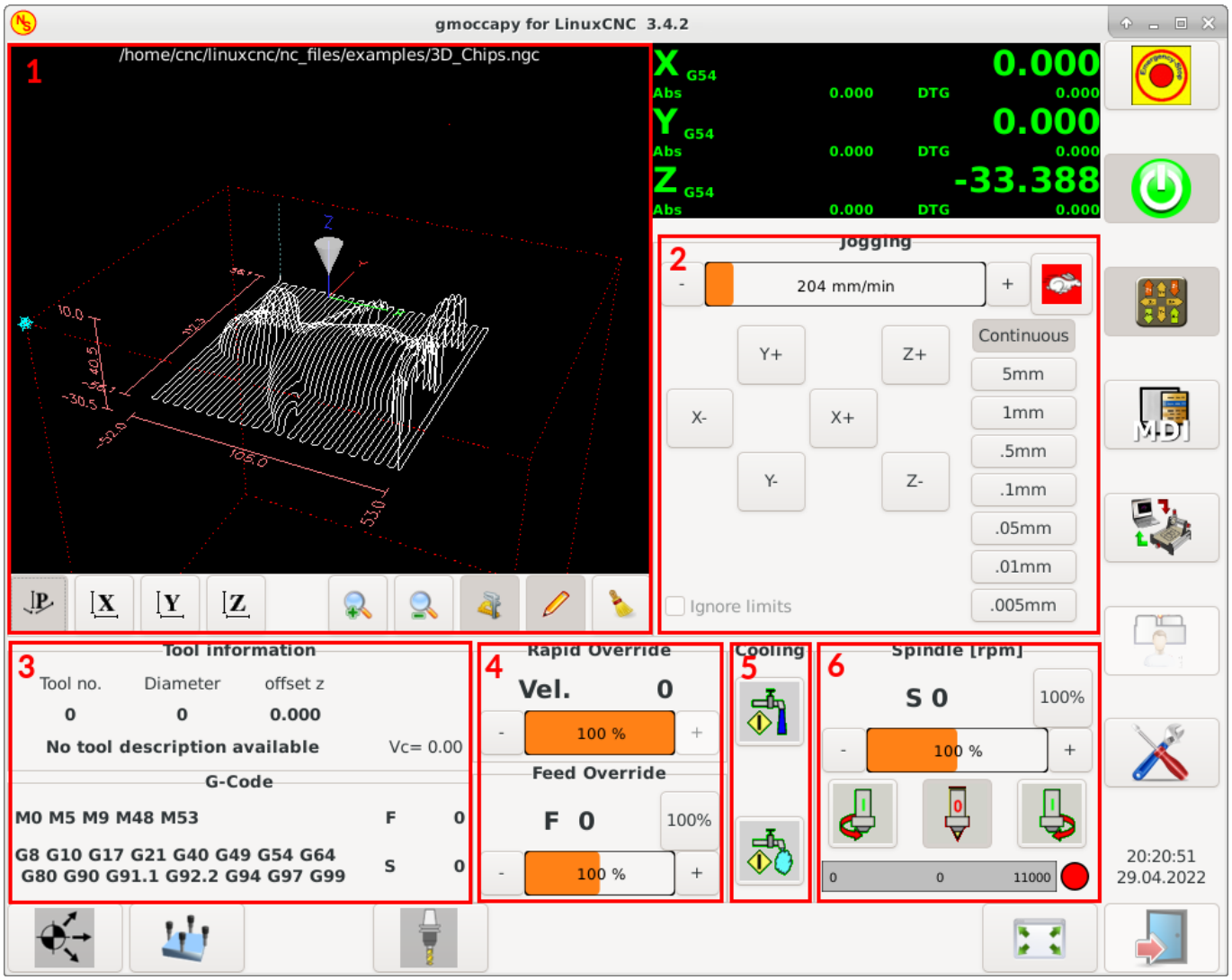
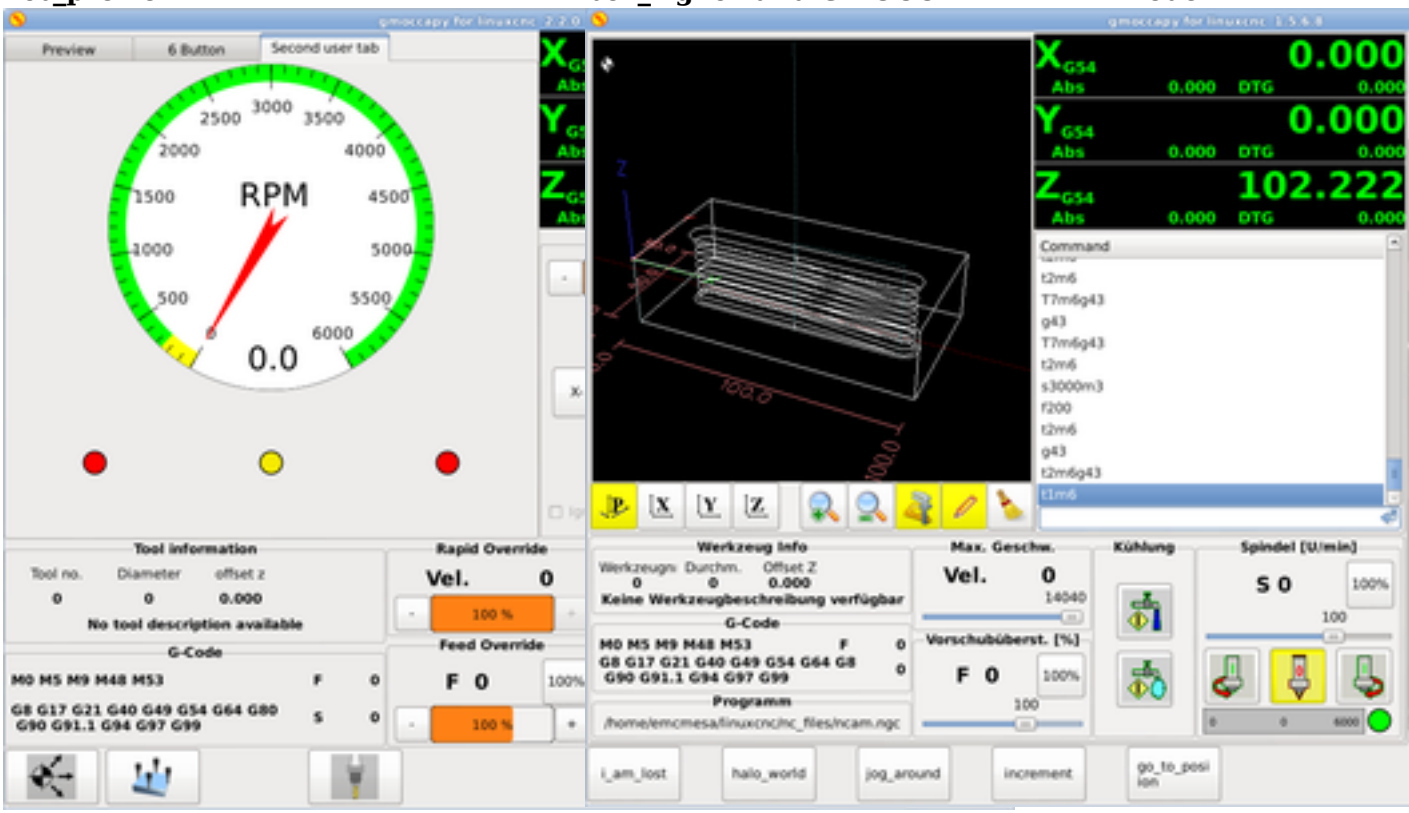



Abbildung 10.14: Eingebettete Registerkartenpositionen

**Anmerkung**
If you make any HAL connections to your custom glade panel, you need to do that in the HAL file specified in the EMBED_TAB_COMMAND line, otherwise you may get an error that the HAL pin does not exist — this is because of race conditions loading the HAL files. Connections to GMOCCAPY HAL pins need to be made in the postgui HAL file specified in your INI file, because these pins do not exist prior of realizing the GUI.

Hier sind einige Beispiele:

**ntb_preview** **box_right - and GMOCCAPY in MDI mode**




### 10.2.4.5 User Created Messages

GMOCCAPY has the ability to create HAL driven user messages. To use them you need to introduce some lines in the [DISPLAY] section of the INI file.

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** - Will just display a message as pop up window, using the messaging system of GMOCCAPY.
- **okdialog** - Will hold focus on the message dialog and will activate a `-waiting` HAL pin.
- **yesnodialog** - Will hold focus on the message dialog and will activate a `-waiting` HAL pin and provide a `-response` 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 G-code:

```
(PREVIEW,hide)
<G-code to be hidden>
(PREVIEW,show)
```

#### 10.2.4.7 User Command File

If a file `~/.gmoccapyrc` exists, its contents are executed as Python source code just after the GUI is displayed. The details of what may be written in the `~/.gmoccapyrc` are subject to change during the development cycle.

A configuration-specific Python file may be specified with an INI file setting

```
[DISPLAY]
USER_COMMAND_FILE=filename.py
```

If this file is specified, this file is sourced just after the GMOCCAPY GUI is displayed **instead** of `~/.gmoccapyrc`.

The following example changes the size of the vertical buttons: .Example of .gmoccapyrc file

```
self.widgets.vbtb_main.set_size_request(85,-1)
BB_SIZE = (70, 70) # default = (90, 56)
self.widgets.tbtn_estop.set_size_request(*BB_SIZE)
self.widgets.tbtn_on.set_size_request(*BB_SIZE)
self.widgets.rbt_manual.set_size_request(*BB_SIZE)
self.widgets.rbt_mdi.set_size_request(*BB_SIZE)
self.widgets.rbt_auto.set_size_request(*BB_SIZE)
self.widgets.tbtn_setup.set_size_request(*BB_SIZE)
self.widgets.tbtn_user_tabs.set_size_request(*BB_SIZE)
self.widgets.btn_exit.set_size_request(*BB_SIZE)
```

The widget names can the looked up in the /usr/share/gmoccapy.glade file

#### 10.2.4.8 User CSS File

Similar to the User command file it's possible to influence the appearance by cascading style sheets (CSS). If a file `~/.gmoccapy_css` exists, its contents are loaded into the stylesheet provider and are so being applied to the GUI.

A configuration-specific CSS file may be specified with an INI file setting

```
[DISPLAY]
USER_CSS_FILE=filename.css
```

If this file is specified, this file is used **instead** of `~/.gmoccapy_css`.

Information what can be controlled by CSS can be found here: Overview of CSS in GTK

Here an example how the color of checked buttons can be set to yellow: .Example Yellow color for checked buttons

```
button:checked {
    background: rgba(250,230,0,0.8);
}
```

#### 10.2.4.9 Protokollierung (engl. logging)

GMOCCAPY supports specifying the level of information (log level) that will be printed to the console and to the log file.

The order is *VERBOSE*, *DEBUG*, *INFO*, *WARNING*, *ERROR*, *CRITICAL*. Default is *WARNING*, that means *WARNING*, *ERROR* and *CRITICAL* are printed.

You can specify the log level in the INI file like this:

```
[DISPLAY]
DISPLAY = gmoccapy <log_level_param>
```

using these parameters:

```
Log level   <log_level_param>
DEBUG       -d
INFO        -i
VERBOSE     -v
ERROR       -q
```

**Example: Configure logging to print only errors**

```
[DISPLAY]
DISPLAY = gmoccapy -q
```

You can specify where to save the log file:

```
[DISPLAY]
LOG_FILE = gmoccapy.log
```

If `LOG_FILE` is not set, logging happens to `$HOME/<base_log_name>.log`.

### 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**
Sie müssen alle Verbindungen zu GMOCCAPY-Pins in Ihrer postgui.hal-Datei herstellen. Wenn GMOCCAPY gestartet wird, erstellt es die HAL-Pins für die GUI und führt dann die in der INI-Datei genannte post-GUI-HAL-Datei aus:

```
[HAL]
POSTGUI_HALFILE=<Dateiname>
```

Typischerweise wäre `<Dateiname>` der Basisname der Konfiguration + `_postgui.hal`, z.B. `lathe_postgui.hal`, aber es kann jeder legale Dateiname sein.
Diese Befehle werden nach der Erstellung des Bildschirms ausgeführt und garantieren, dass die HAL-Pins des Widgets verfügbar sind.
Sie können mehrere Zeilen mit `POSTGUI_HALFILE=<Dateiname>` in der INI-Datei haben. Sie werden nacheinander in der Reihenfolge ausgeführt, in der sie erscheinen.

---

#### 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!

Bei einer dreiachsigen XYZ-Maschine reagieren die HAL-Pins wie in den folgenden drei Tabellen dargestellt:

Tabelle 10.2: Funktionelle Zuordnung der horizontalen Buttons (1)

| Pin | Manual Mode | MDI Mode | Auto Mode |
|---|---|---|---|
| gmoccapy.h-button.button-0 | open homing button | macro_0 oder nichts | open file |
| gmoccapy.h-button.button-1 | Öffnen des Touch-Off-Krims | macro_1 oder nichts | reload program |
| gmoccapy.h-button.button-2 | | macro_2 oder nichts | run |
| gmoccapy.h-button.button-3 | open tool dialogs | macro_3 oder nichts | stop |
| gmoccapy.h-button.button-4 | | macro_4 oder nichts | pause |
| gmoccapy.h-button.button-5 | | macro_5 oder nichts | step by step |
| gmoccapy.h-button.button-6 | | macro_6 oder nichts | run from line if enabled in settings, otherwise Nothing |
| gmoccapy.h-button.button-7 | | macro_7 oder nichts | 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 | Code bearbeiten |

Tabelle 10.3: Funktionszuweisung der horizontalen Buttons (2)

| Pin | Settings Mode | Homing Mode | Touch-Off-Modus |
|---|---|---|---|
| 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 | öffnen von HAL-scope | home z | |
| gmoccapy.h-button.button-6 | HAL-Status öffnen | | zero G92 |
| gmoccapy.h-button.button-7 | open HAL meter | | |
| gmoccapy.h-button.button-8 | HAL-Kalibrierung öffnen | unhome all | set selected |
| gmoccapy.h-button.button-9 | öffnen von HAL-show | back | back |

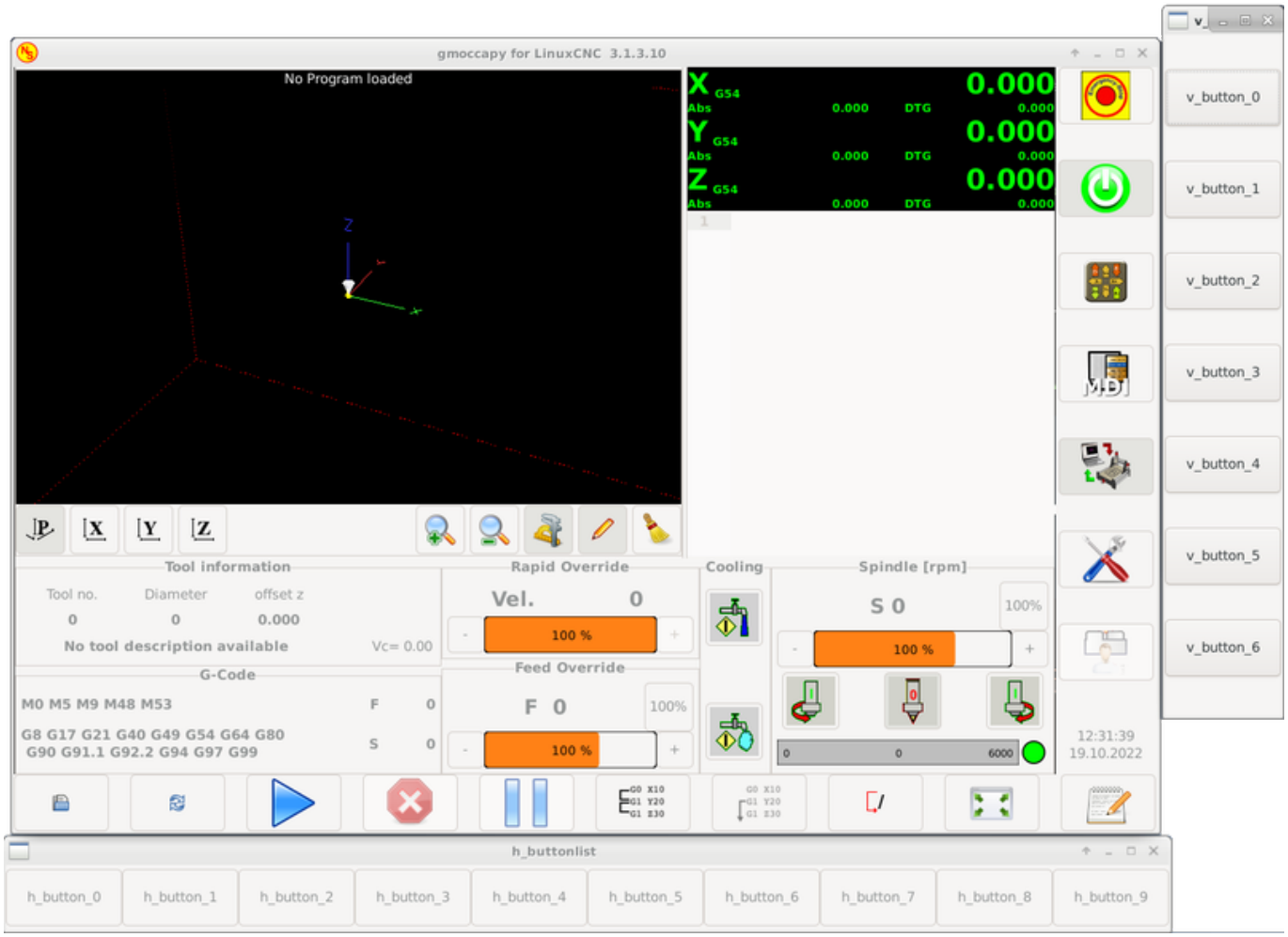
Tabelle 10.4: Funktionelle Zuordnung der horizontalen Buttons (3)

| Pin | Tool Mode | Edit Mode | Select File |
|---|---|---|---|
| gmoccapy.h-button.button-0 | delete tool(s) | | go to home directory |
| gmoccapy.h-button.button-1 | new tool | reload file | one directory level up |
| gmoccapy.h-button.button-2 | Werkzeugtabelle neu laden | speichern | |
| gmoccapy.h-button.button-3 | Änderungen übernehmen | speichern unter | move selection left |
| gmoccapy.h-button.button-4 | change tool by number T? M6 | | move selection right |
| gmoccapy.h-button.button-5 | set tool by number without change M61 Q? | | jump to directory as set in settings |
| gmoccapy.h-button.button-6 | change tool to the selected one | new file | |
| gmoccapy.h-button.button-7 | | | select / ENTER |
| gmoccapy.h-button.button-8 | Werkzeugberührung in Z | show keyboard | |
| gmoccapy.h-button.button-9 | back | back | back |

Wir haben also 67 Reaktionen mit nur 10 HAL-Pins!

These pins are made available to be able to use the screen without a 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.

**Sample configuration ”gmoccapy_sim_hardware_button” showing the side buttons**




### 10.2.5.2 Velocities and 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 Geschwindigkeit
- **gmoccapy.jog.jog-velocity.count-enable** *(bit IN)* - Must be True, to enable counts
- **gmoccapy.feed.feed-override.counts** *(s32 IN)* - Vorschub Override
- **gmoccapy.feed.feed-override.count-enable** *(bit IN)* - Muss True sein, um Zählungen zu aktivieren
- **gmoccapy.feed.reset-feed-override** *(bit IN)* - setzt den Vorschub-Override auf *0% zurück
- **gmoccapy.spindle.spindle-override.counts** *(s32 IN)* - Spindel override

- **gmoccapy.spindle.spindle-override.count-enable** *(bit IN)* - Muss True sein, um Zählungen zu aktivieren
- **gmoccapy.spindle.reset-spindle-override** *(bit IN)* - setzt den Spindel-Override auf *0% zurück
- **gmoccapy.rapid.rapid-override.counts** *(s32 IN)* - Maximale Geschwindigkeit der *Chine
- **gmoccapy.rapid.rapid-override.count-enable** *(bit IN)* - Muss True sein, um Zählungen zu aktivieren

To connect potentiometers, use the following pins:

- **gmoccapy.jog.jog-velocity.direct-value** *(float IN)* - So stellen Sie den Schieberegler für die Jog-Geschwindigkeit ein
- **gmoccapy.jog.jog-velocity.analog-enable** *(bit IN)* - Muss True sein, um analoge Eingänge zuzulassen
- **gmoccupy.feed.feed-override. direct-value** *(float IN)* So passen Sie den Schieberegler für die Vorschubüberschreitung an (engl. feed override)
- **gmoccapy.feed.feed-override.analog-enable** *(bit IN)* - Muss True sein, um analoge Eingänge zuzulassen
- **gmoccapy.spindle.spindle-override.direct-value** *(float IN)* - Zum Einstellen des Spindel-Override-Schiebereglers
- **gmoccapy.spindle.spindle-override.analog-enable** *(bit IN)* - Muss True sein, um Analogeingänge zuzulassen
- **gmoccapy.rapid.rapid-override.direct-value** *(float)* - Zum Einstellen des Reglers für die maximale Geschwindigkeit
- **gmoccapy.rapid.rapid-override.analog-enable** *(bit IN)* - Muss True sein, um Analogeingänge zuzulassen

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)* - Solange True, wird der Wert des Schiebereglers erhöht
- **gmoccapy.spc_jog_vel.decrease** *(bit IN)* - Solange True, wird der Wert des Schiebereglers verringert
- **gmoccapy.spc_jog_vel.scale** *(float IN)* - Ein Wert zur Skalierung des Ausgabewerts (praktisch, um Einheiten/Min in Einheiten/Sek zu ändern)
- **gmoccapy.spc_jog_vel.value** *(float OUT)* - Wert des Widgets
- **gmoccapy.spc_jog_vel.scaled-value** *(float OUT)* - Skalierter Wert des Widgets .FEED
- **gmoccapy.spc_feed.increase** *(bit IN)* - Solange True wird der Wert des Schiebereglers erhöht
- **gmoccapy.spc_feed.decrease** *(bit IN)* - Solange True wird der Wert des Schiebereglers verringert
- **gmoccapy.spc_feed.scale** *(float IN)* - Ein Wert zur Skalierung des Ausgabewerts (praktisch, um Einheiten/Min in Einheiten/Sek zu ändern)
- **gmoccapy.spc_feed.value** *(float OUT)* - Wert des Widgets

- **gmoccapy.spc_feed.scaled-value** *(float OUT)* - Skalierter Wert des Widgets .SPINDLE (engl. für Spindel)
- **gmoccapy.spc_spindle.increase** *(bit IN)* - Solange True, wird der Wert des Schiebereglers erhöht
- **gmoccapy.spc_spindle.decrease** *(bit IN)* - Solange True, wird der Wert des Schiebereglers verringert
- **gmoccapy.spc_spindle.scale** *(float IN)* - Ein Wert zur Skalierung des Ausgabewerts (praktisch, um Einheiten/min in Einheiten/sec zu ändern)
- **gmoccapy.spc_spindle.value** *(float OUT)* - Wert des Widgets
- **gmoccapy.spc_spindle.scaled-value** *(float OUT)* - Skalierter Wert des Widgets .RAPIDS
- **gmoccapy.spc_rapid.increase** *(bit IN)* - Solange True wird der Wert des Schiebereglers erhöht
- **gmoccapy.spc_rapid.decrease** *(bit IN)* - Solange True, wird der Wert des Schiebereglers verringert
- **gmoccapy.spc_rapid.scale** *(float IN)* - Ein Wert zur Skalierung des Ausgabewerts (praktisch, um Einheiten/Min in Einheiten/Sek zu ändern)
- **gmoccapy.spc_rapid.value** *(float OUT)* - Wert des Widgets
- **gmoccapy.spc_rapid.scaled-value** *(float OUT)* - Skalierter Wert des Widgets

The float pins do accept values from 0.0 to 1.0, being the percentage value you want to set the slider value.

**Warnung**
If you use both connection types, do not connect the same slider to both pin as the influences between the two has not been tested! Different sliders may be connected to the one or other HAL connection type.

**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**
Die Benennung dieser HAL-Pins hat sich in GMOCCAPY 2 geändert.

---

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)*

Wenn Sie eine 4-Achsen-Konfiguration verwenden, gibt es zwei zusätzliche Pins:

- **gmoccapy.jog.jog-<Ihr Buchstabe der vierten Achse >-plus** *(bit IN)*
- **gmoccapy.jog.jog-<Ihr Buchstabe der vierten Achse >-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

The jog velocity can be selected with the corresponding slider. The scale of the slider will be modified if the turtle button (the one showing a rabbit or a turtle) has been toggled. If the button is not visible, it might have been disabled on the settings page. If the button shows the rabbit-icon, the scale is from min to max machine velocity. If it shows the turtle, the scale will reach only 1/20 of max velocity by default. The used divider can be set on the settings page.

Mit einem Touchscreen ist es also viel einfacher, kleinere Geschwindigkeiten auszuwählen.

GMOCCAPY bietet diesen HAL-Pin an, um zwischen Hasen- und Igel-Joggen zu wechseln:

- **gmoccapy.jog.turtle-jog** *(bit IN)*

#### 10.2.5.5 Jog Increment HALPins

The jog increments given in the 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.

Wenn Sie 6 Inkremente in Ihrer INI-Datei haben, wie im obigen Beispiel, erhalten Sie **7** Pins:

- **gmoccapy.jog.jog-inc-0** *(bit IN)* - Dieser Wert ist fest und steht für kontinuierliches Joggen.
- **gmoccapy.jog.jog-inc-1** *(bit IN)* - Erste in der INI-Datei angegebene Schrittweite.
- **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

Um einen Schlüsselschalter zum Entsperren der Einstellungsseite verwenden zu können, wird der folgende Pin exportiert:

- **gmoccapy.unlock-settings** *(bit IN)* - Die Einstellungsseite wird entsperrt, wenn der Pin high ist. Um diesen Pin zu verwenden, müssen Sie ihn auf der Einstellungsseite aktivieren.

#### 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)* - Löscht den ersten Fehler und setzt den Pin `gmoccapy.error` auf false, nachdem der letzte Fehler gelöscht wurde.
- **gmoccapy.warning-confirm** *(bit IN)* - Bestätigt den Warnungsdialog mit Klick auf 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)* - Löst den Dialog aus.

okdialog

- **gmoccapy.messages.okdialog** *(bit IN)* - Löst den Dialog aus.
- **gmoccapy.messages.okdialog-waiting** *(bit OUT)* - Wird *1* sein, solange der Dialog geöffnet ist. Beim Schließen der Nachricht wird dieser Pin zurückgesetzt.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bit IN)* - Löst den Dialog aus.
- **gmoccapy.messages.yesnodialog-waiting** *(bit OUT)* - Wird *1* sein, solange der Dialog geöffnet ist. Beim Schließen der Nachricht wird dieser Pin zurückgesetzt.
- **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.

To add a user created message you need to add the message to the INI file in the DISPLAY section. See Configuration of User Created Messages.

**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 zur Anzeige der Spindeldrehzahl auf der Spindelleiste.
- **gmoccapy.spindle_at_speed_led** *(bit IN)* - Pin zum Beleuchten der is-at-speed-led.

#### 10.2.5.10 Pins to Indicate Program Progress Information

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 Tool Related Pins

**Werkzeugwechsel-Pins** These pins are provided to use GMOCCAPY's internal tool change dialog, similar to the one known from AXIS, but with several modifications. So you will not only get the message to change to *tool number 3*, but also the description of that tool like *7.5 mm 3 flute cutter*. The information is taken from the tool table, so it is up to you what to display.

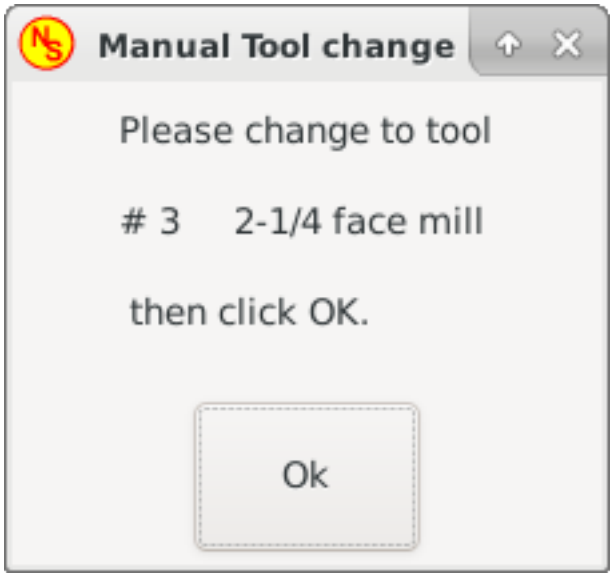



Abbildung 10.15: GMOCCAPY tool change dialog

- **gmoccapy.toolchange-number** *(s32 IN)* - Die Nummer des zu wechselnden Werkzeugs
- **gmoccapy.toolchange-change** *(bit IN)* - Zeigt an, dass ein Werkzeug gewechselt werden muss
- **gmoccapy.toolchange-changed** *(bit OUT)* - Zeigt an, dass das Werkzeug gewechselt wurde
- **gmoccapy.toolchange-confirm** *(bit IN)* - Bestätigt den Werkzeugwechsel

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**
Bitte beachten Sie, dass diese Verbindungen in der postgui HAL Datei vorgenommen werden müssen.

---

**Werkzeug-Offset Pins** These pins allow you to show the active tool offset values for X and Z in the tool information frame. You should know that they are only active after G43 has been sent.

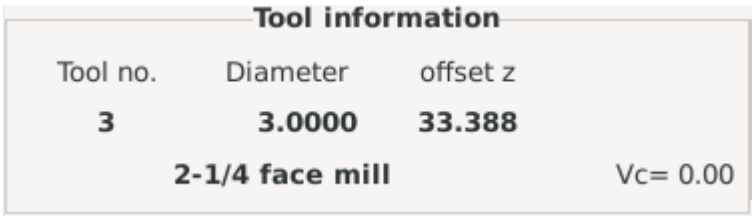



Abbildung 10.16: Tool information area

- **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.

---

Um die aktuellen Offsets anzuzeigen, müssen die Pins in der postgui-HAL-Datei wie folgt verbunden werden:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

---

**Wichtig**
Bitte beachten Sie, dass GMOCCAPY selbst dafür sorgt, die Offsets zu aktualisieren, indem es nach jedem Werkzeugwechsel ein G43 sendet, **aber nicht im Auto-Modus!**
Wenn Sie also ein Programm schreiben, sind Sie dafür verantwortlich, nach jedem Werkzeugwechsel ein G43 zu senden!

---

### 10.2.6 Automatische Werkzeugmessung

GMOCCAPY offers an integrated auto tool measurement. To use this feature, you will need to do some additional settings and you may want to use the offered HAL pin to get values in your own NGC remap procedure.

---

**Wichtig**
Before starting the first test, do not forget to enter the probe height and probe velocities on the settings page! See Settings Page Tool Measurement.

---

It might be also a good idea to take a look at the tool measurement video, see tool measurement related videos.

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:

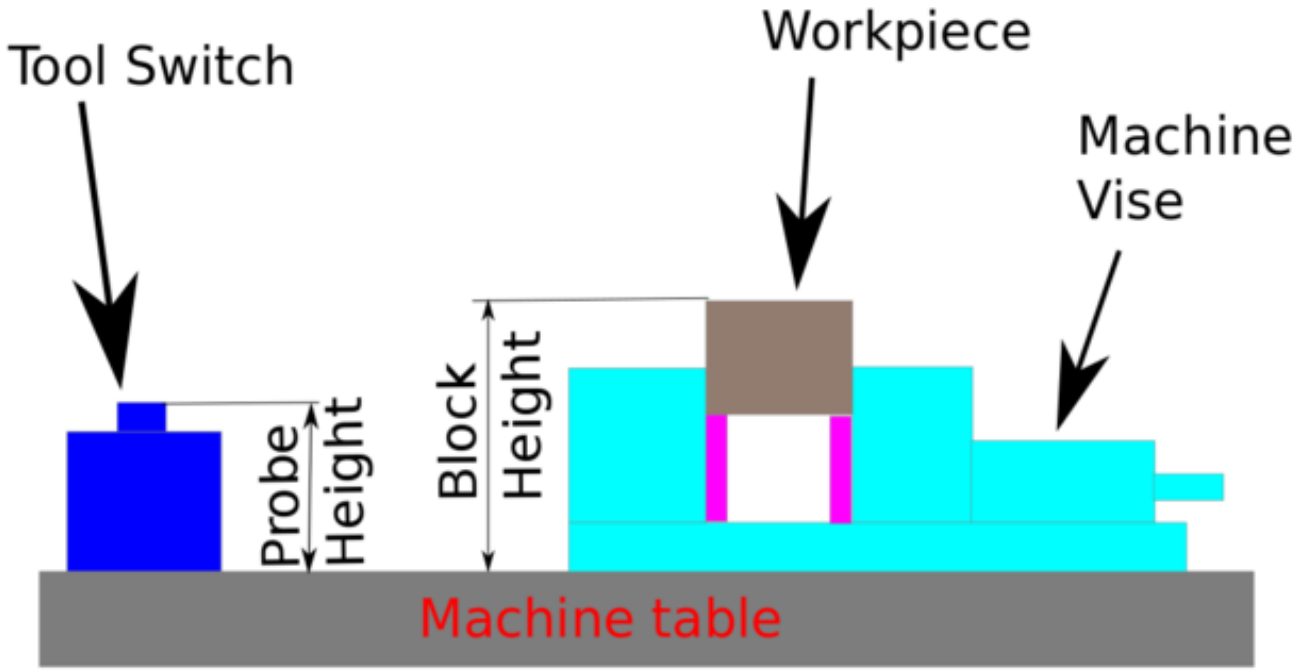



Abbildung 10.17: Werkzeugmessung Daten

With the first given tool change the tool will be measured and the offset will be set automatically to fit the block height. The advantage of the GMOCCAPY way is, that you do not need a reference tool.

**Anmerkung**
Your program must contain a tool change at the beginning! The tool will be measured, even it has been used before, so there is no danger, if the block height has changed. There are several videos showing the way to do that on YouTube.

#### 10.2.6.1 Provided 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)* - Werkzeugmessung einschalten oder nicht
- **gmoccapy.blockheight** *(float OUT)* - Der gemessene Wert der oberen Fläche des Werkstücks
- **gmoccapy.probeheight** *(float OUT)* - The probe switch height
- **gmoccapy.searchvel** *(float OUT)* - Die Geschwindigkeit, mit der nach dem Schalter des Werkzeugmesstasters gesucht wird
- **gmoccapy.probevel** *(float OUT)* - Die Geschwindigkeit zum Antasten der Werkzeuglänge

#### 10.2.6.2 INI File Modifications

Ändern Sie Ihre INI-Datei so, dass sie die folgenden Abschnitte enthält.

**Der RS274NGC-Abschnitt**

```
[RS274NGC]
# Unterfunktion wird aufgerufen, wenn ein Fehler beim Werkzeugwechsel auftritt, wird nicht ↩
    bei jeder Maschinenkonfiguration benötigt
ON_ABORT_COMMAND=O <on_abort> call

# The remap code
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

---

**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

Erstellen Sie zunächst ein Verzeichnis "python" in Ihrem Konfigurationsverzeichnis. Kopieren Sie von `<Ihr_linuxcnc-dev_verzeichnis>/configs/sim/gmoccapy/python` die folgenden Dateien in den gerade erstellten Ordner `config_dir/python`:

- `toplevel.py`
- `remap.py`
- `stdglue.py`

Von `<Ihr_linuxcnc-dev_Verzeichnis>/configs/sim/gmoccapy/macros` kopieren

- `on_abort.ngc`
- `change.ngc`

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

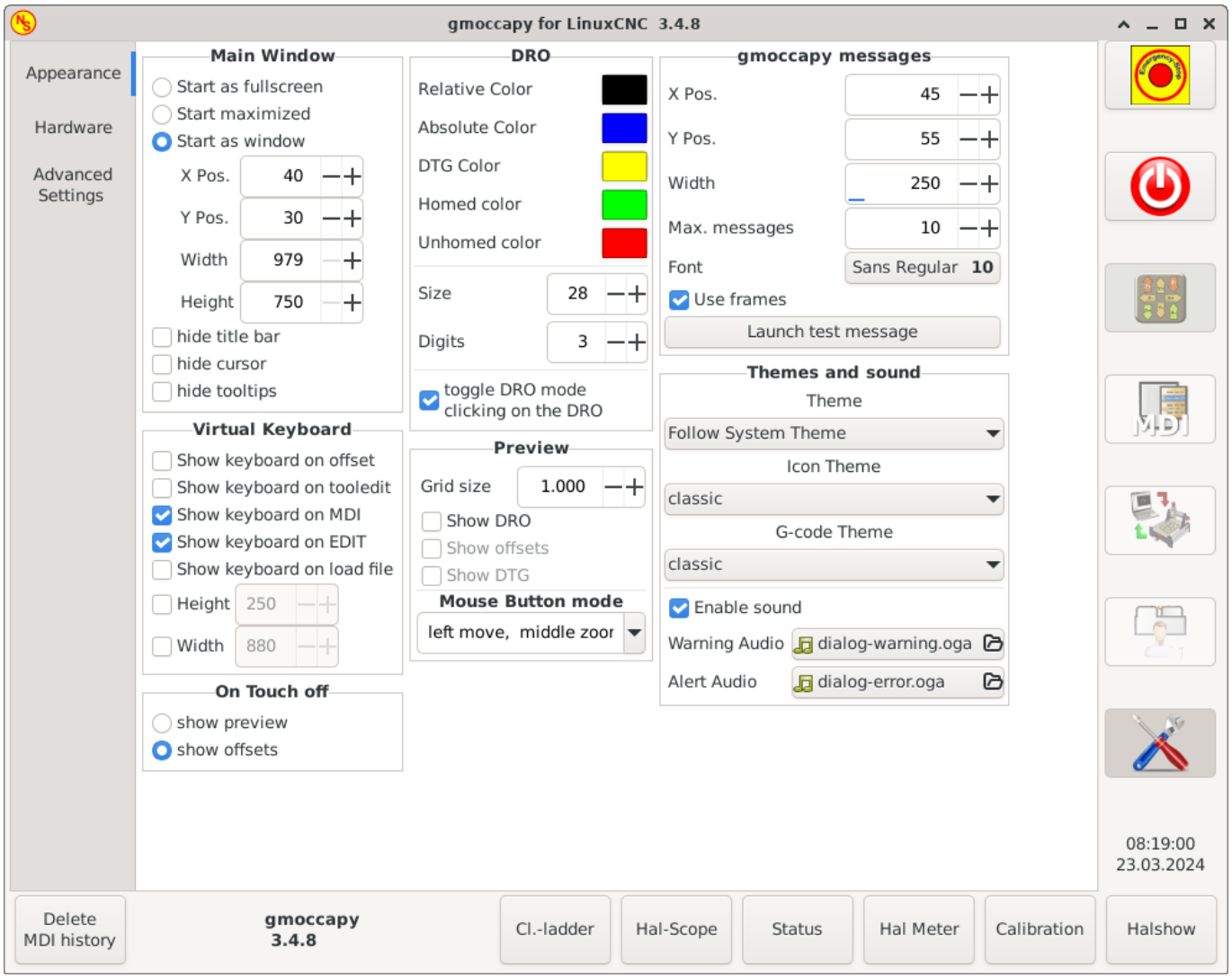



Abbildung 10.18: GMOCCAPY Einstellungsseite Erscheinungsbild

Auf dieser Registerkarte finden Sie die folgenden Optionen:

**Hauptfenster** Here you can select how you wish the GUI to start. The main reason for this was the wish to get an easy way for the user to set the starting options without the need to touch code. You have three options:

- *Start as full screen*
- *Start maximized*
- *Start als Fenster* - Wenn Sie ”Start als Fenster” wählen, werden die Spinboxen zum Einstellen der Position und Größe aktiv. Einmal eingestellt, startet die GUI jedes Mal an der gewählten Stelle und mit der gewählten Größe. Dennoch kann der Benutzer die Größe und Position mit der Maus verändern, was aber keinen Einfluss auf die Einstellungen hat.
- *hide title bar* - Allows the title bar to be hidden. (default: title bar visible).
- *hide cursor* - Does allow to hide the cursor, what is very useful if you use a touch screen.

- *hide tooltips* - Hides the tool tips.

**Virtuelle Tastatur** The checkboxes allow the user to select if he wants the on board keyboard to be shown immediately when entering the MDI Mode, 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 these settings, so you are able to show or hide the keyboard by pressing the button.

The default setting is:

- *Show keyboard on offset* = False
- *Show keyboard on tooledit* = False
- *Show keyboard on MDI* = True
- *Show keyboard on EDIT* = True
- *Show keyboard on load file* = False

---

**Anmerkung**
If this section is not sensitive, you have not installed a virtual keyboard, supported ones are *onboard* and *matchbox-keyboard*.

---

---

**Anmerkung**
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:

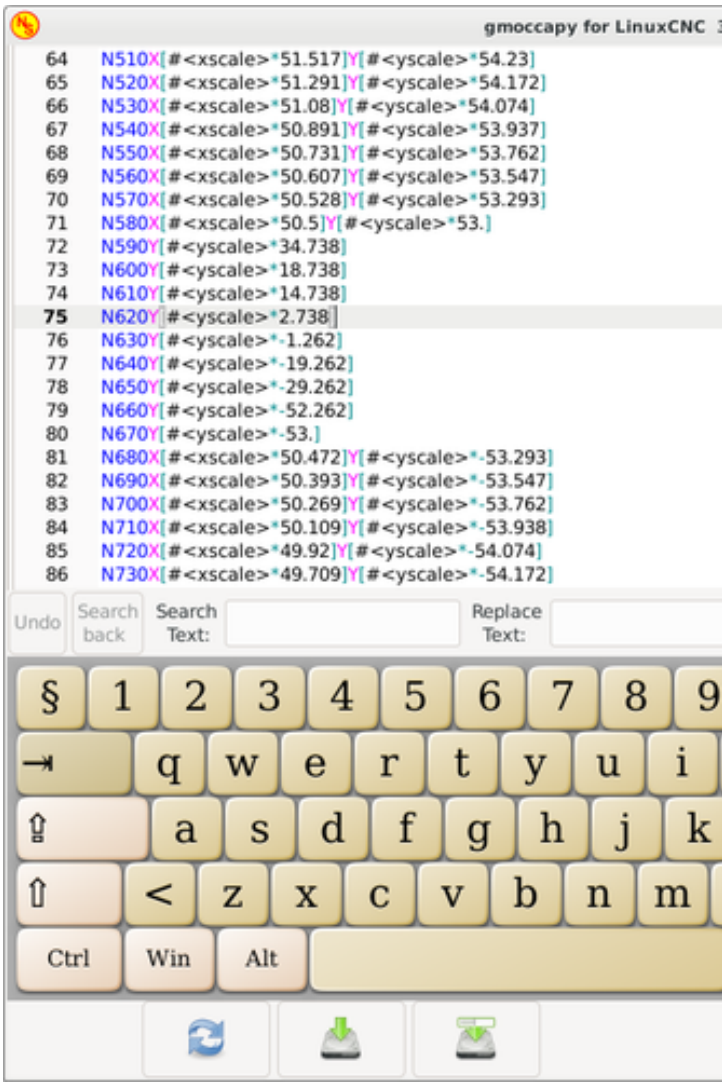
```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

The letters ”de” are for German, you will have to set them according to your locale settings. Just execute this file before starting LinuxCNC, it can be done also adding a starter to your local folder.

```
./config/autostart
```

Damit das Layout beim Start automatisch eingestellt wird.
Für matchbox-keyboard müssen Sie Ihr eigenes Layout erstellen, für ein deutsches Layout fragen Sie im Forum.

---




**GMOCCAPY 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:

- *Relative Color* = black
- *Absolute Color* = blue
- *DTG Color* = yellow

Die Vordergrundfarbe der DRO kann ausgewählt werden mit:

- *Homed Color* = green
- *Unhomed Color* = red

**Anmerkung**
Sie können durch die DRO-Modi (absolut, relativ, Entfernung zu gehen) wechseln, indem Sie auf die Zahl auf dem DRO klicken! Wenn Sie auf den linken Seitenbuchstaben des DRO klicken, können Sie in einem Popup-Fenster den Wert der Achsen festlegen, was das Einstellen des Werts erleichtert, da Sie nicht über die untere Touch-Off-Taste gehen müssen.

- *size* - Allows to set the size of the DRO font, default is 28, if you use a bigger screen you may want to increase the size up to 56. If you do use 4 axes, the DRO font size will be 3/4 of the value, because of space reason.
- *digits* - 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.

- *toggle DRO mode* - If not active, a mouse click on the DRO will not take any action.
  By default this checkbox is active, so every click on any DRO will toggle the DRO readout from actual to relative to DTG (distance to go).
  Neverthereless a click on the axis letter will open the popup dialog to set the axis value.

Vorschau

- *Grid Size* - 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* - Zeigt die DRO auch im Vorschaufenster an, sie wird immer in der Vollbildvorschau angezeigt.
- *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

Default is left move, middle zoom, right rotate.
The mouse wheel will still zoom the preview in every mode.

**Tipp**
Wenn Sie ein Element in der Vorschau auswählen, wird das ausgewählte Element als Rotationsmittelpunkt genommen und im Automodus wird die entsprechende Codezeile hervorgehoben.

**Gmoccapy Messages**

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* - Die Breite des Meldungsfelds.
- *Max. messages* - The maximum number of messages you want to see at once. If you set this to 10, the $11^{th}$ 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.

**Themen und Klänge** This lets the user select what desktop theme to apply and what error and messages sounds should be played.

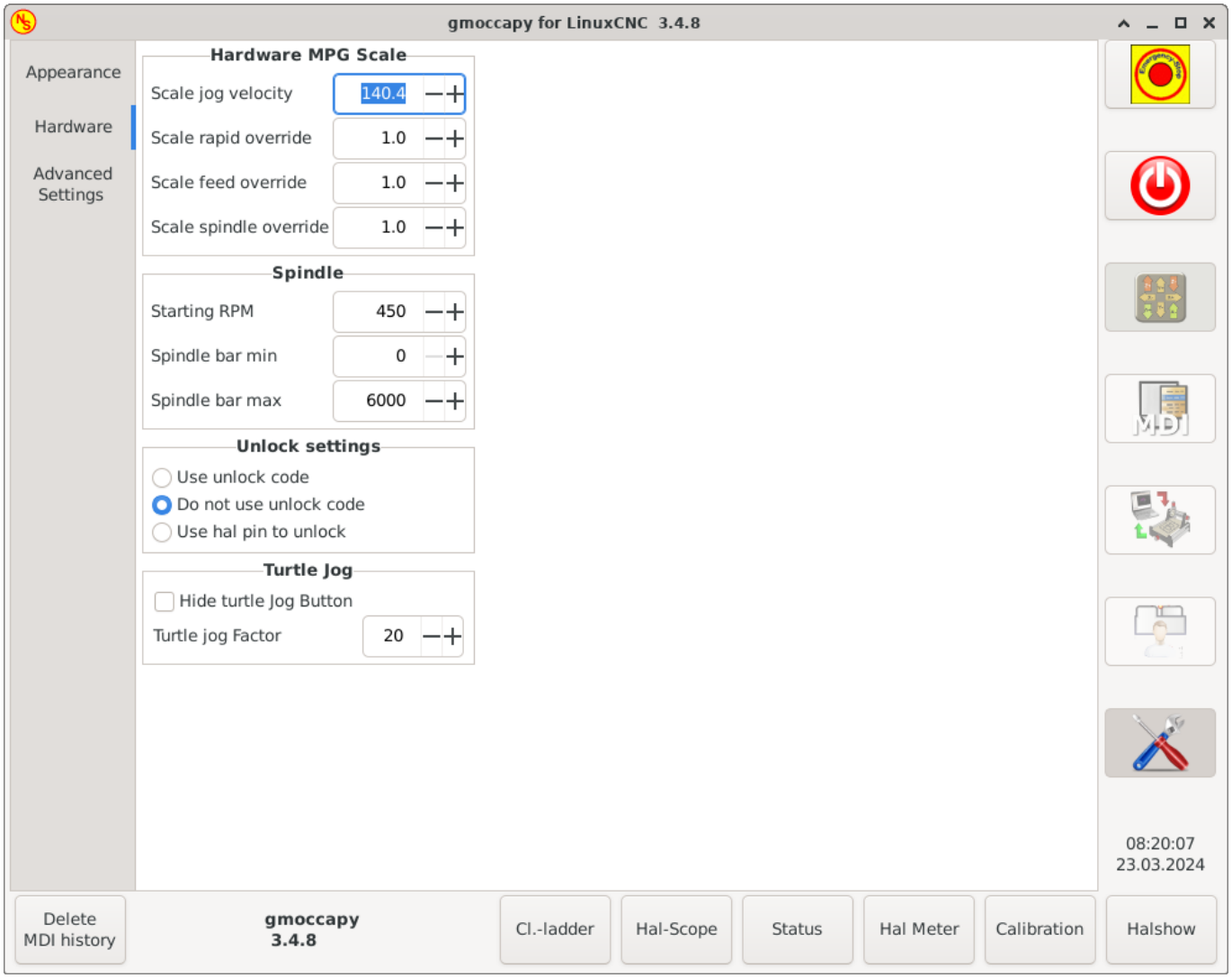
By default ”Follow System Theme” is set.

Es ermöglicht weiterhin, das Symbolthema zu ändern. Derzeit stehen drei Themen zur Verfügung:

- 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 ppr and he wants to slow down the max. vel. from 14000 to 2000 mm/min, that needs 12000 pulses, 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 which has limits from 50 to 120 % so he goes from min to max within 70 pulses, meaning not even 1/4 turn.

Standardmäßig werden alle Skalen anhand der Berechnung festgelegt:

```
(MAX - MIN)/100
```

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.

- *Spindelstange min* und *Spindelstange max* - Legt die Grenzen der Spindelstange fest, die im INFO-Rahmen auf dem Hauptbildschirm angezeigt werden.

  Standardwerte sind
  MIN = 0
  MAX = 6000

  **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.

**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 code is **123**).

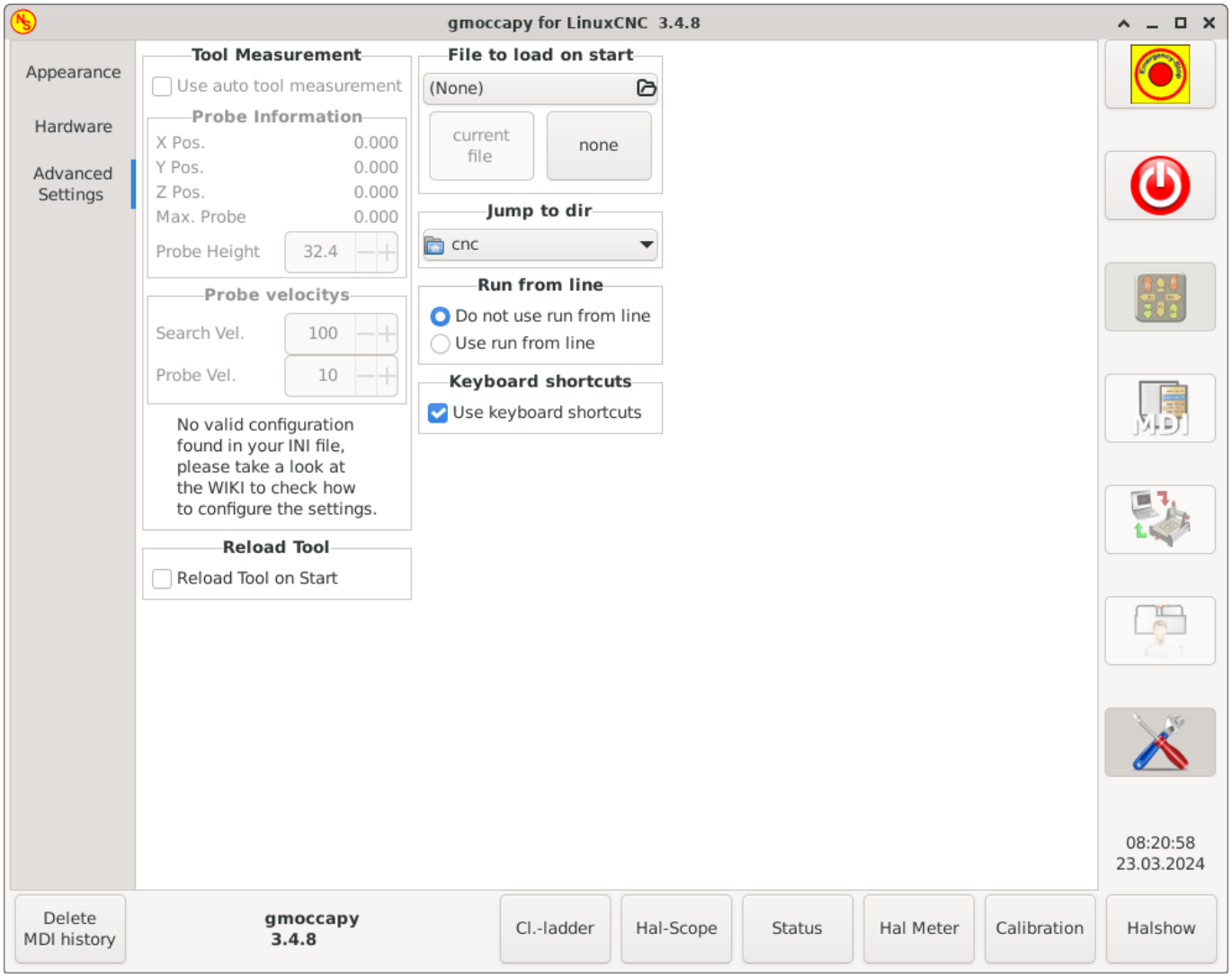
**Schildkröten Jog**

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**
  The following information are taken from your INI file and must be given in absolute coordinates:

  - *X Pos.* - Die X-Position des Werkzeugschalters.
  - *Y Pos.* - Die Y-Position des Werkzeugschalters.
  - *Z Pos.* - Die Z-Position des Werkzeugschalters, wir fahren im Eilgang zu dieser Koordinate.

- *Max. Probe* - 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* - Lädt das letzte Werkzeug beim Start nach der Referenzfahrt.

Wenn diese Option aktiviert ist, wird das Werkzeug in der Spindel bei jeder Änderung der Voreinstellungsdatei gespeichert, so dass das zuletzt montierte Werkzeug beim Start wieder geladen werden kann. Das Werkzeug wird geladen, nachdem alle Achsen referenziert wurden, da es vorher nicht erlaubt ist, MDI-Befehle auszuführen. Wenn Sie NO_FORCE_HOMING verwenden, können Sie diese Funktion nicht nutzen, da das benötigte all_homed_signal nie ausgegeben wird.

**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.

**Ausführen von gegebener Zeile** You can allow or disallow the run from line. This will set the corresponding button insensitive (grayed out), so the user will not be able to use this option. The default is disable run from line.

**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.

**Tastatürkürzel** Some users want to jog their 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. They can be activated by the checkbox

• *Use keyboard shortcuts*

**Warnung**
Es wird nicht empfohlen, Tastatur-Jogging zu verwenden, da dies ein ernsthaftes Risiko für Bediener und Maschine darstellt.

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

**Im manuellen Modus**

- *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

**Im Auto-Modus**

- *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

### 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.

Start by creating an empty directory with the name of the icon theme. Place the directory in one of GMOCCAPY's icon theme directories. Then we need a file called index.theme in the root folder of our icon theme which contains the required metadata for the theme. That's a simple text file with at least the following sections:

- [Icon Theme]

```
[Icon Theme]
Name=Ihr_Theme-Name
Comment=Eine Beschreibung des Themes
Inherits=hicolor
Directories=16x16/actions,24x24/actions,32x32/actions,48x48/actions,scalable/actions
```

  - Name: Der Name Ihres Icon-Designs.
  - Comment: Eine Beschreibung des Themas Ihres Symbols.
  - Inherits: Ein Icon-Thema kann von einem anderen Icon-Thema abgeleitet werden, der Standard ist hicolor.
  - Verzeichnisse: Eine durch Kommata getrennte Liste aller Verzeichnisse Ihres Icon-Themas. Jedes Verzeichnis enthält in der Regel alle Icons des Themas in einer bestimmten Größe, z.B. 16x16/actions sollte alle Icons mit der Kategorie "actions" in der Größe 16x16 Pixel als Pixelgrafiken (z.B. png-Dateien) enthalten. Ein Sonderfall ist das Verzeichnis "scalable/actions", dieses enthält skalierbare Icons, die nicht an eine bestimmte Größe gebunden sind (z.B. svg-Dateien). Durch die Bereitstellung unterschiedlich großer Versionen der Icons können wir ein schön aussehendes Icon in verschiedenen Größen garantieren, und wir haben auch die Möglichkeit, das Icon entsprechend seiner Größe zu verändern, zum Beispiel kann ein 64x64 px großes Icon mehr Details enthalten als seine 16x16 px Version.

- Für jedes Verzeichnis müssen wir auch einen Abschnitt in die Datei index.theme schreiben:

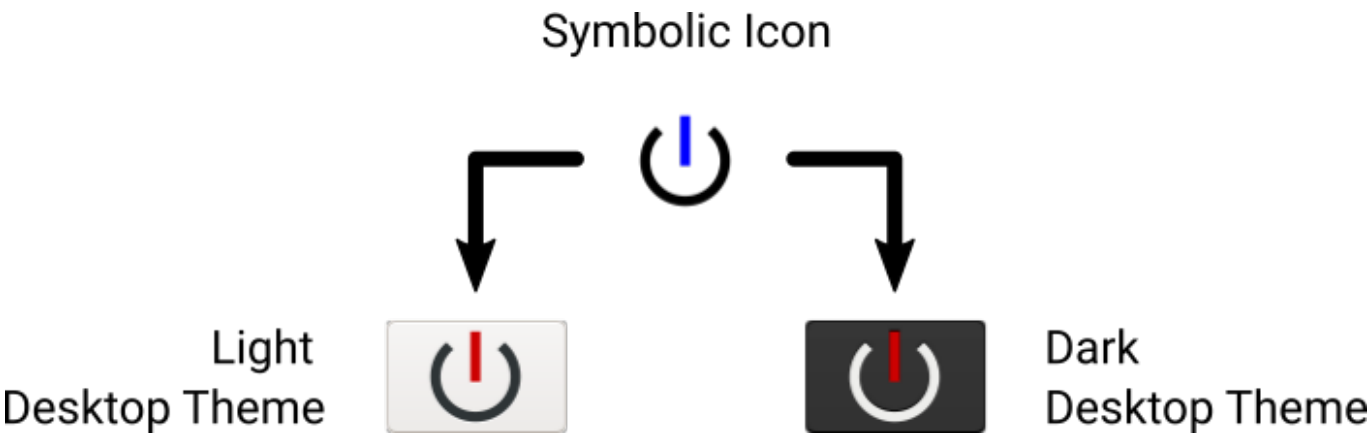
```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

[scalable/actions]
Size=48
Type=Scalable
Context=Actions
```

  - SIze: Nominale Symbolgröße in diesem Verzeichnis
  - Type: Fixed (engl. für festgelegt), Threshold (engl. für Schwellwert) or Scalable (engl. für skalierbar)
  - Context: Beabsichtigte "Kategorie" von Icons

Das ist im Grunde alles, was man braucht, um ein eigenes Icon Theme zu erstellen.

#### 10.2.8.2 Symbolische Icons

Symbolic icons are a special type of icon, usually a monochrome image. The special feature of symbolic icons is that the icons are automatically colored at runtime to match the desktop theme. That way, icon themes can be created that work well with dark and also light desktop themes (in fact, that's not always the best option, that's why a dedicated "material-light" theme exists).




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.

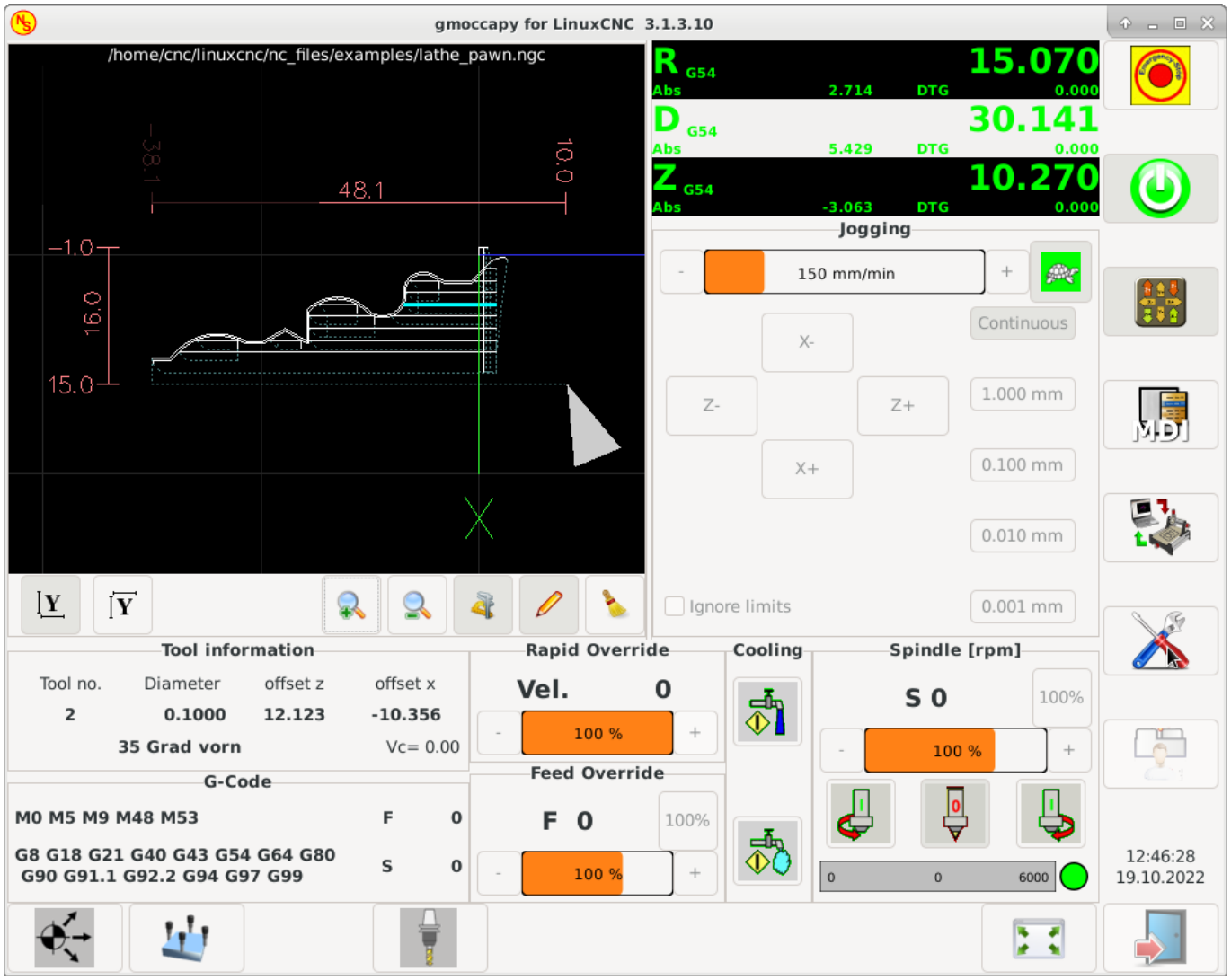



Abbildung 10.19: Normale Drehmaschine

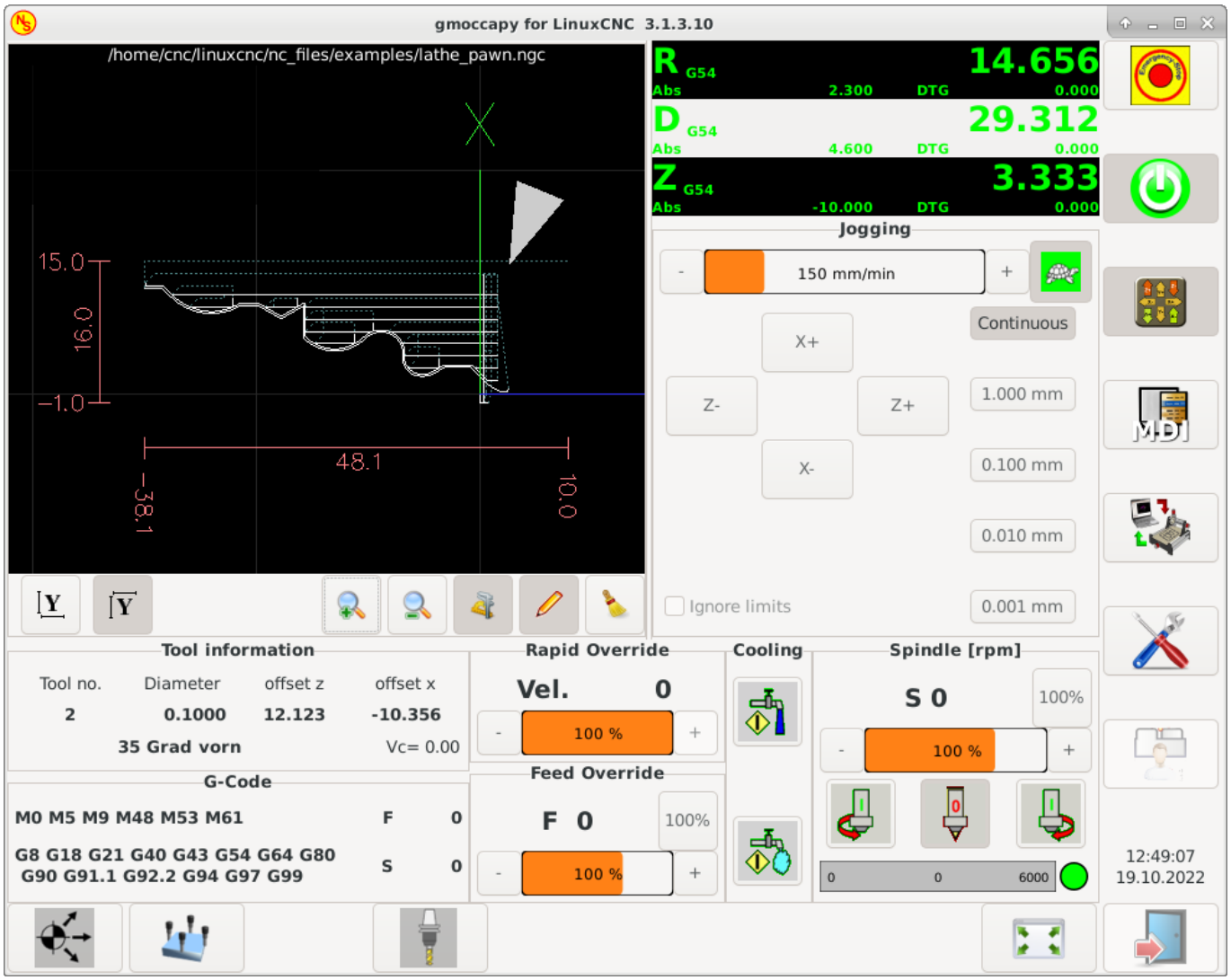



Abbildung 10.20: Back Tool Lathe

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:

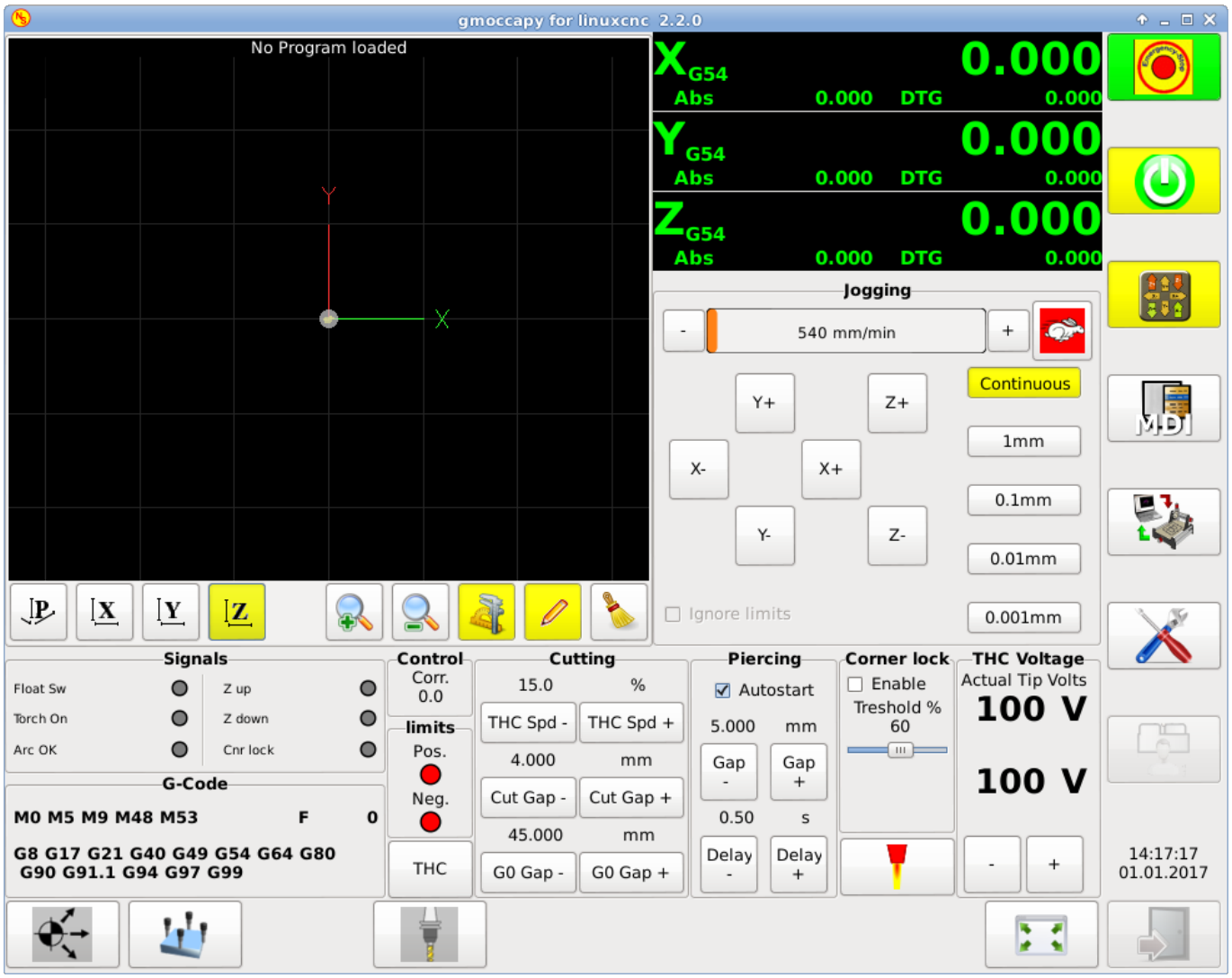
- *Arrow_Left* or *NumPad_Left* - Jog Z minus
- *Arrow_Right* or *NumPad_Right* - Jog Z plus
- *Arrow_up* or *NumPad_Up* - Jog X minus
- *Arrow_Down* or *NumPad_Down* - Jog X plus

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




There is a very good WIKI, which is actually growing, maintained by Marius, see Plasma wiki page.

### 10.2.11 Videos on 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://youtu.be/O5B-s3uiI6g

#### 10.2.11.2 Simulierte Jog-Wheels

https://youtu.be/ag34SGxt97o

#### 10.2.11.3 Einstellungen Seite

https://youtu.be/AuwhSHRJoiI

#### 10.2.11.4 Simulierte Hardware-Taste

German: https://youtu.be/DTqhY-MfzDE
English: https://youtu.be/ItVWJBK9WFA

#### 10.2.11.5 Benutzer-Registerkarten

https://youtu.be/rG1zmeqXyZI

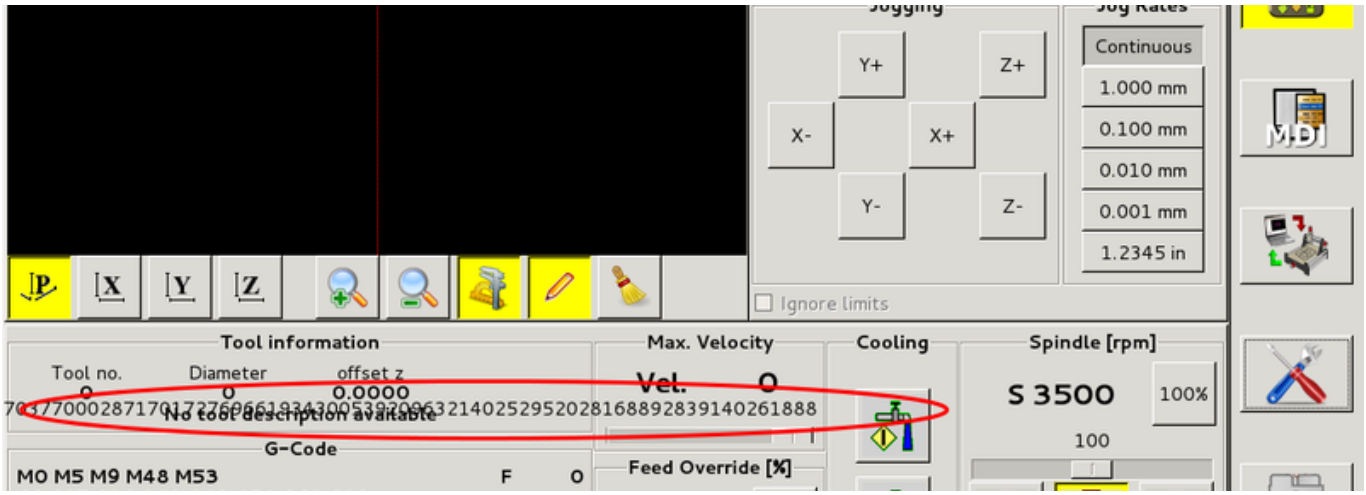
#### 10.2.11.6 Videos zur Werkzeugvermessung

Auto Tool Measurement Simulation: https://youtu.be/rrkMw6rUFdk
Auto Tool Measurement Screen: https://youtu.be/Z2ULDj9dzvk
Auto Tool Measurement Machine: https://youtu.be/1arucCaDdX4

### 10.2.12 Known Problems

#### 10.2.12.1 Seltsame Zahlen im Infobereich

Wenn Sie im Infobereich von GMOCCAPY seltsame Zahlen erhalten wie:




You have made your config file using an older version of StepConfWizard. It has made a wrong entry in the INI file under the [TRAJ] named MAX_LINEAR_VELOCITY = xxx. Change that entry to 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 will not see the end of the macro, because the interpreter needs to change its state to IDLE, but the macro does not even set the interpreter to a new state. To avoid that just add a G4 P0.1 line to get the needed signal. The correct macro would be:

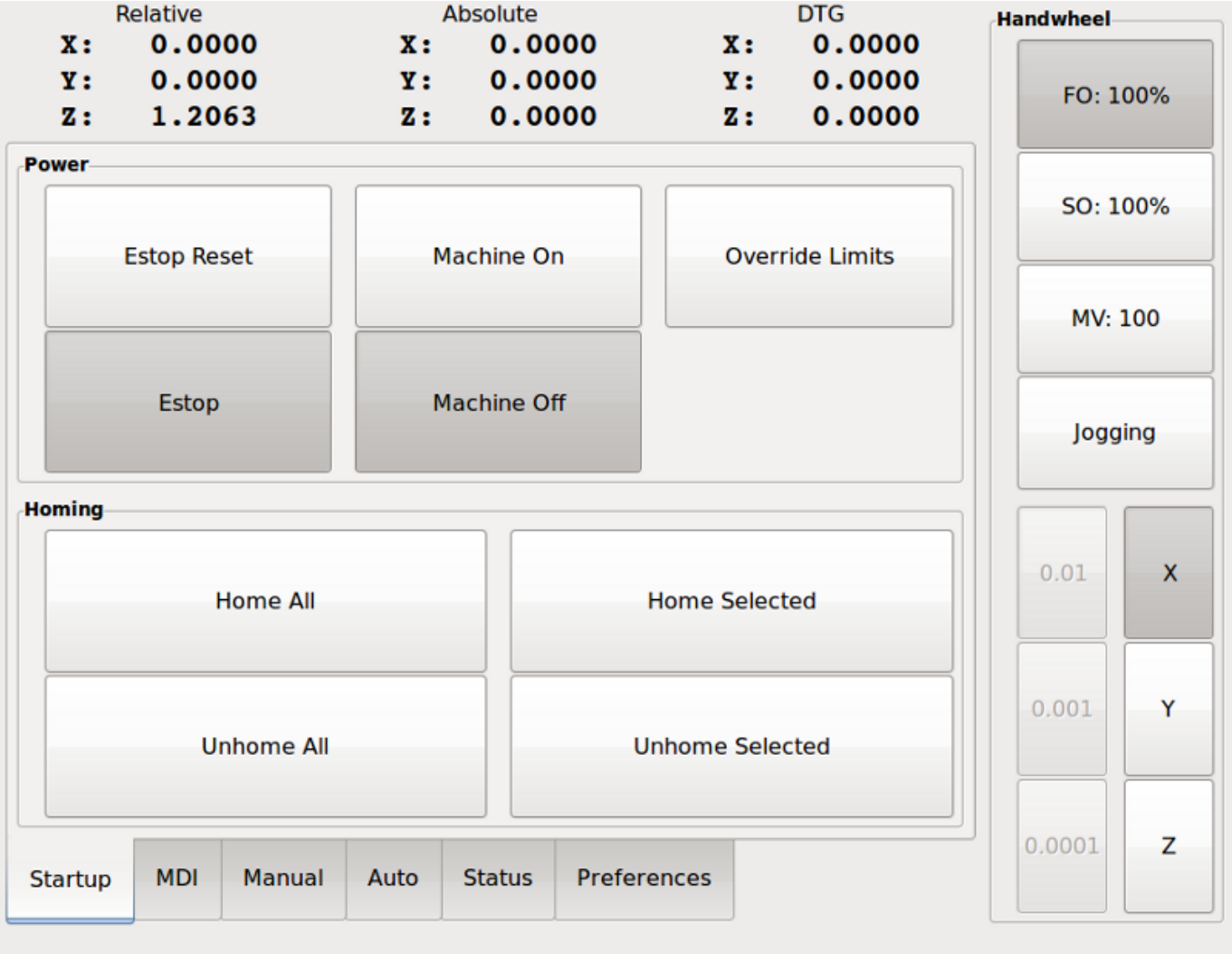
```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

G4 P0.1

o<zeroxy> endsub
m2
```

## 10.3 The Touchy Graphical User Interface

Touchy is a user interface for LinuxCNC meant for use on machine control panels, and therefore does not require keyboard or mouse.

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 is an infrastructure to display a custom screen to control LinuxCNC. Gscreen borrows heavily from GladeVCP. GladeVCP uses the GTK widget editor GLADE to build virtual control panels (VCP) by

point and click. Gscreen combines this with Python programming to create a GUI screen for running a CNC machine.

Gscreen is customizable if you want different buttons and status LEDs. Gscreen supports GladeVCP which is used to add controls and indicators. To customize Gscreen you use the Glade editor. Gscreen is not restricted to adding a custom panel on the right or a custom tab it is fully editable.

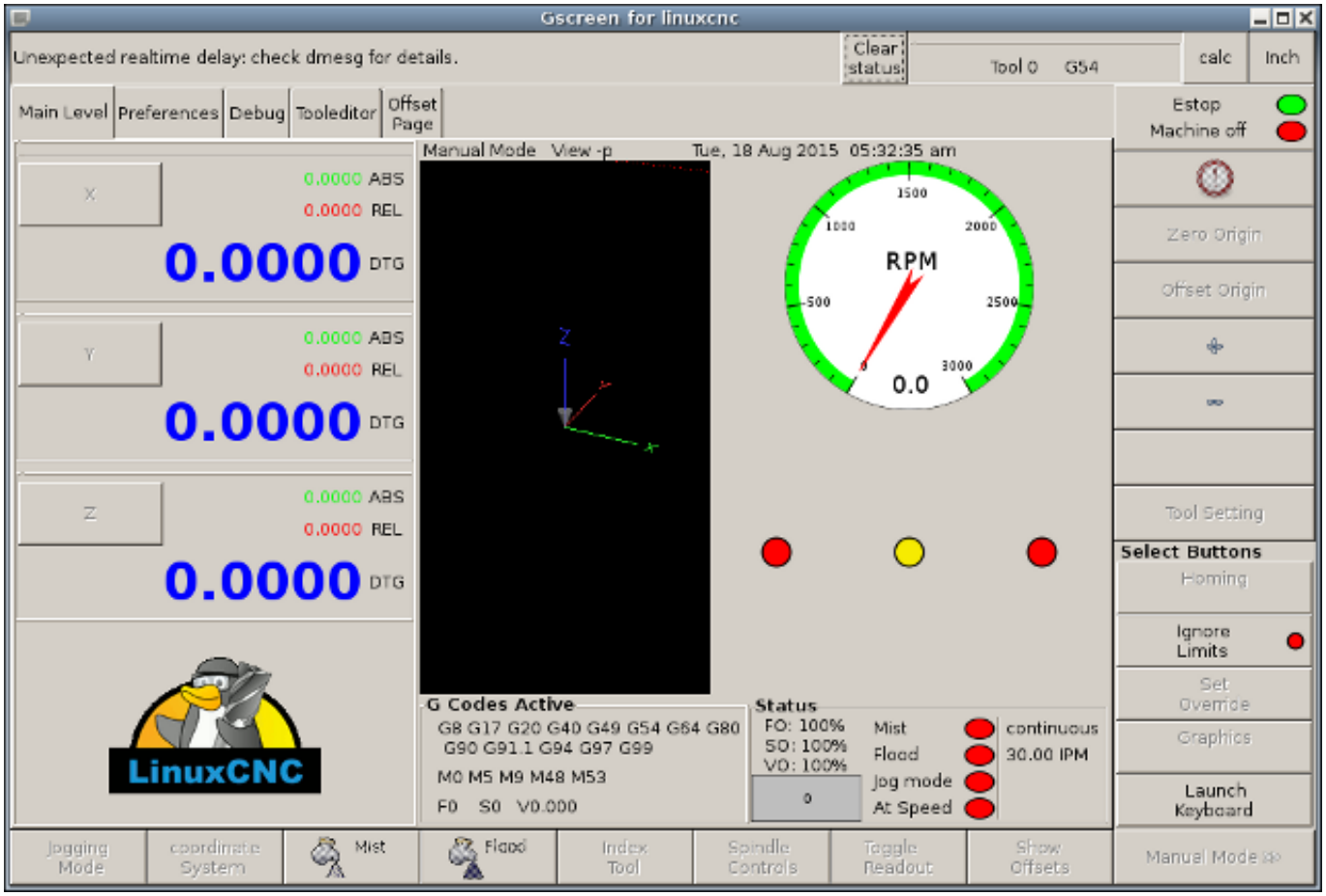



Abbildung 10.21: Gscreen Standardbildschirm

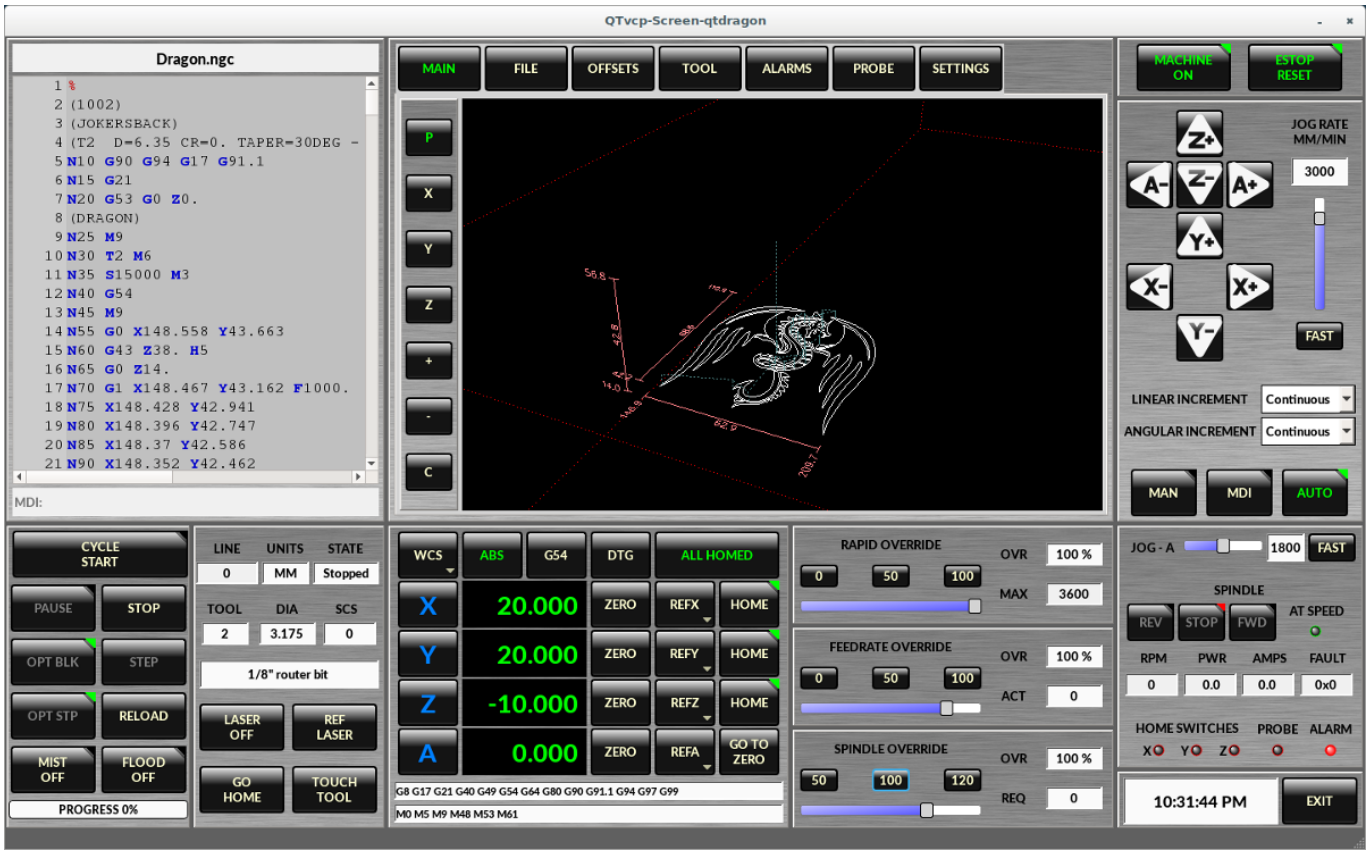



Abbildung 10.22: Gscreen Silverdragon-Bildschirm

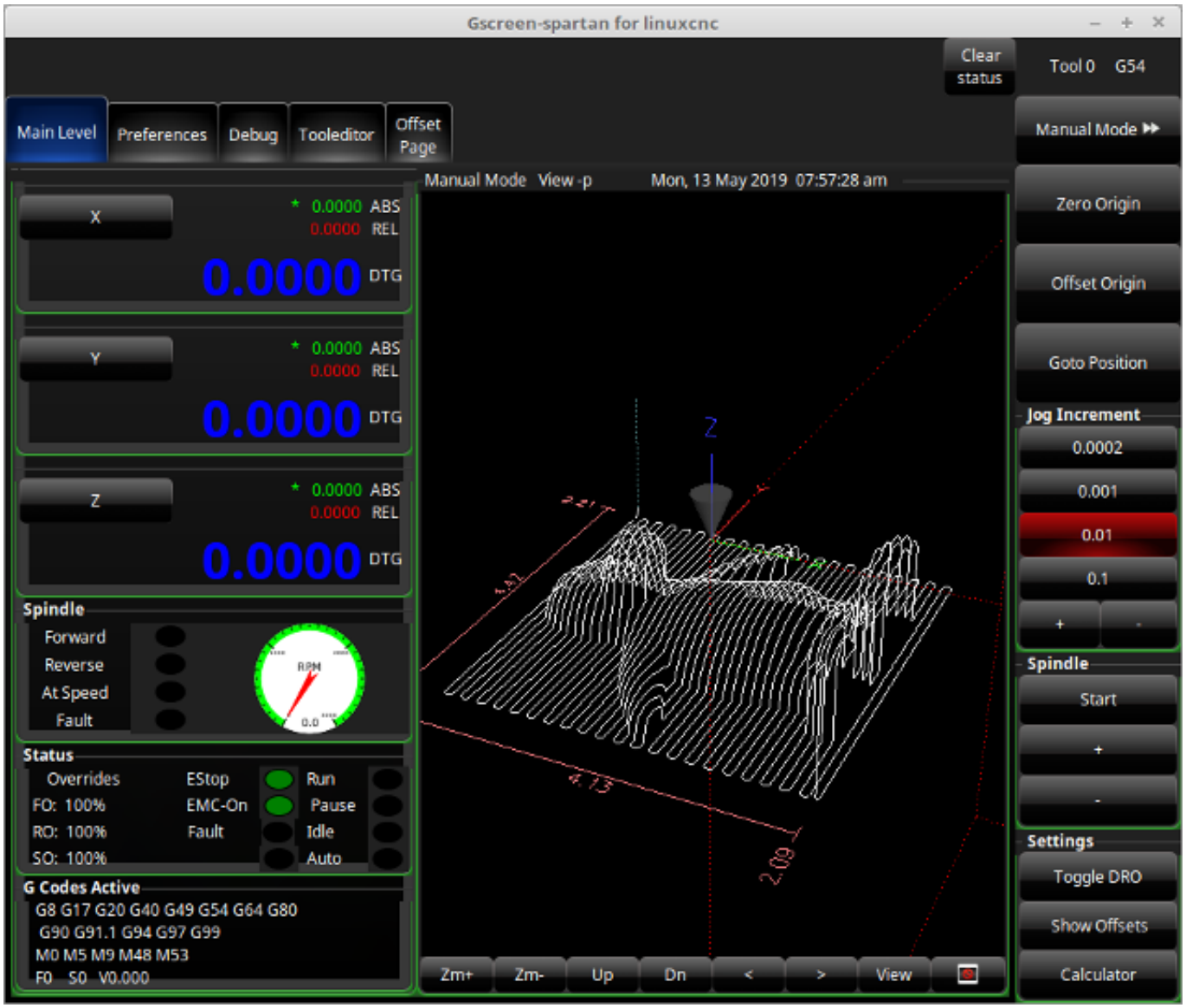



Abbildung 10.23: Gscreen Spartan-Bildschirm

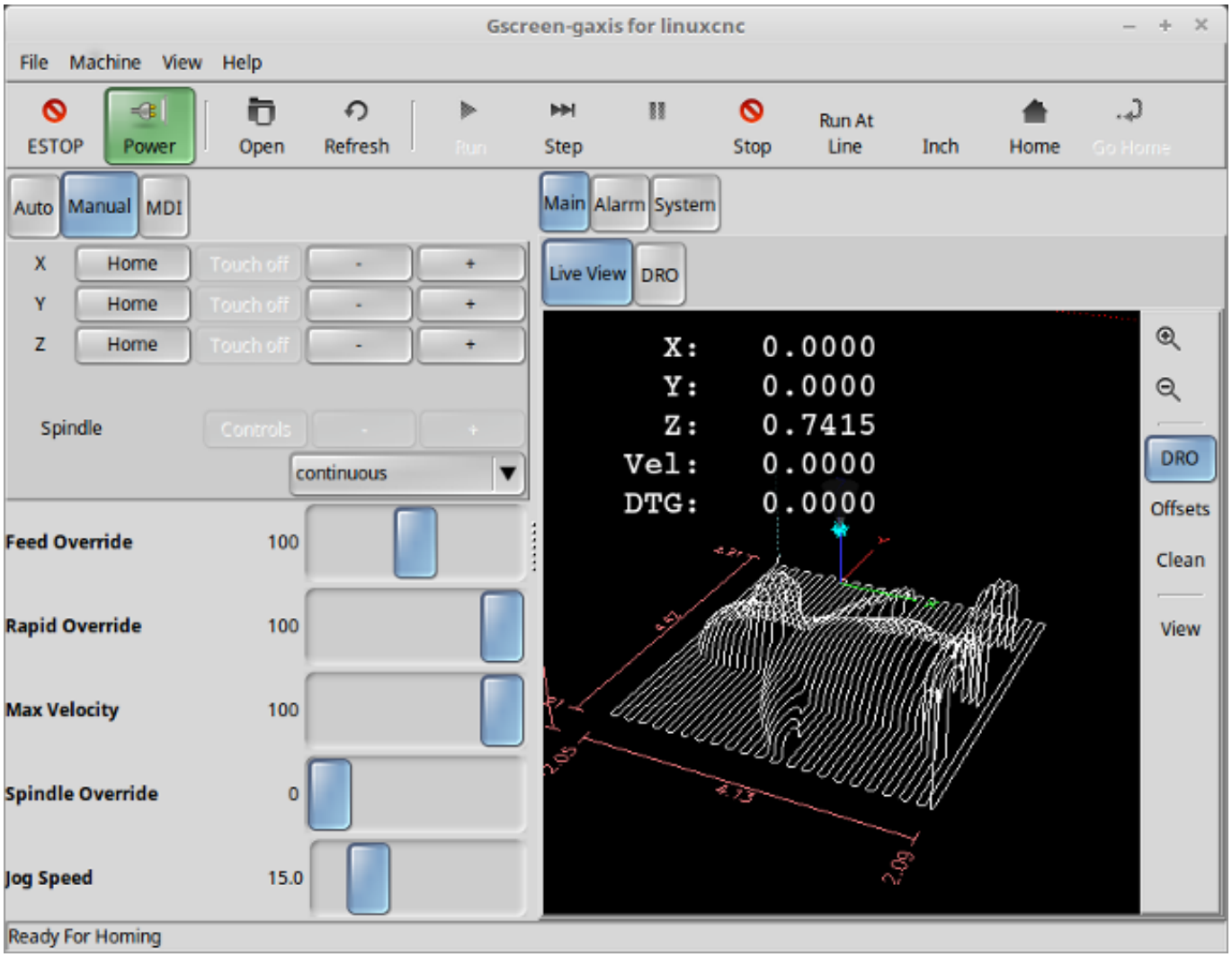



Abbildung 10.24: Gscreen Gaxis-Bildschirm

Abbildung 10.25: Gscreen Industrieller Bildschirm

Gscreen is based on *Glade* (the editor), *PyGTK* (the widget toolkit), and *GladeVCP* (LinuxCNC's connection to Glade and PyGTK). GladeVCP has some special widgets and actions added just for LinuxCNC A widget is just the generic name used for the buttons, sliders, labels etc of the PyGTK toolkit.

#### 10.4.1.1 Glade-Datei

A Glade file is a text file organized in the XML standard that describes the layout and the widgets of the screen. PyGTK uses this file to actually display and react to those widgets. The Glade editor makes it relatively easy to build and edit this file You must use the Glade 3.38.2 editor that uses the GTK3 widgets.

#### 10.4.1.2 PyGTK

PyGTK is the Python binding to GTK. GTK is the *toolkit* of visual widgets, it is programmed in C. PyGTK uses Python to *bind* with GTK.

## 10.4.2 GladeVCP

GladeVCP binds LinuxCNC, HAL, PyGTK and Glade all together. LinuxCNC requires some special widgets so GladeVCP supplies them. Many are just HAL extensions to existing PyGTK widgets. GladeVCP creates the HAL pins for the special widgets described in the Glade file. GladeVCP also allows one to add Python commands to interact with the widgets, to make them do things not available in their default form. If you can build a GladeVCP panel you can customize Gscreen!

### 10.4.2.1 Übersicht

There are two files that can be used, individually or in combination to add customizations. Local Glade files and handler files. Normally Gscreen uses the stock Glade file and possibly a handler file (if using a sample *skin*). You can specify Gscreen to use *local* Glade and handler files. Gscreen looks in the folder that holds all the configuration files for the configuration you selected.

**Lokale Glade-Dateien** If present, local Glade files in the configuration folder will be loaded instead of the stock Glade files. Local Glade files allow you to use your customized designs rather then the default screens. There is a switch in the INI file to set set the base name: -c name so Gscreen looks for MYNAME.glade and MYNAME_handler.py.

You can tell Gscreen to just load the Glade file and not connect its internal signals to it. This allows gscreen to load any GTK builder saved Glade file. This means you can display a completely custom screen, but also requires you to use a handler file. Gscreen uses the Glade file to define the widgets, so it can show and interact with them. Many of them have specific names, others have Glade given generic names. If the widget will be displayed but never changed then a generic name is fine. If one needs to control or interact with the widget then a hopefully purposeful name is given (all names must be unique). Widgets can also have signals defined for them in the GLADE editor. It defines what signal is given and what method to call.

**Ändern von Standard-Skins** If you change the name of a widget, Gscreen might not be able to find it. If this widget is referenced to from Python code, at best this makes the widget not work anymore at worst it will cause an error when loading Gscreen's default screens don't use many signals defined in the editor, it defines them in the Python code. If you move (cut and paste) a widget with signals, the signals will not be copied. You must add them again manually.

**Handler-Dateien** A handler file is a file containing Python code, which Gscreen adds to its default routines. A handler file allows one to modify defaults, or add logic to a Gscreen skin without having to modify Gscreen proper. You can combine new functions with Gscreen's function to modify behavior as you like. You can completely bypass all of Gscreen's functions and make if work completely differently. If present a handler file named gscreen_handler.py (or MYNAME_handler.py if using the INI switch) will be loaded and registered only one file is allowed Gscreen looks for the handler file, if found it will look for specific function names and call them instead of the default ones. If adding widgets you can set up signal calls from the Glade editor to call routines you have written in the handler file. In this way you can have custom behavior. Handler routines can call Gscreen's default routines, either before or after running its own code. In this way you can tack on extra behavior such as adding a sound. Please see the GladeVCP Chapter for the basics to GladeVCP handler files. Gscreen uses a very similar technique.

**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 is just a big complicated GladeVCP panel, with Python code to control it. To customize it we need the Glade file loaded in the Glade editor.

**Installiertes LinuxCNC** If you have LinuxCNC 2.6+ installed on Ubuntu 10.04 just start the Glade editor from the applications menu or from the terminal. Newer versions of Linux will require you to install Glade 3.8.0 - 3.8.6 (you may need to compile it yourself).

**RIP-kompilierte Befehle** Using a compiled from source version of LinuxCNC open a terminal and cd to the top of the LinuxCNC folder. Set up the environment by entering . *./scripts/rip-environment* now enter *glade*, you see a bunch of warnings in the terminal that you can ignore and the editor should open. The stock Gscreen Glade file is in: src/emc/usr_intf/gscreen/ sample skins are in /share/gscreen/skins/. This should be copied to a configuration folder. Or you can make a clean-sheet Glade file by saving it in a configuration folder.

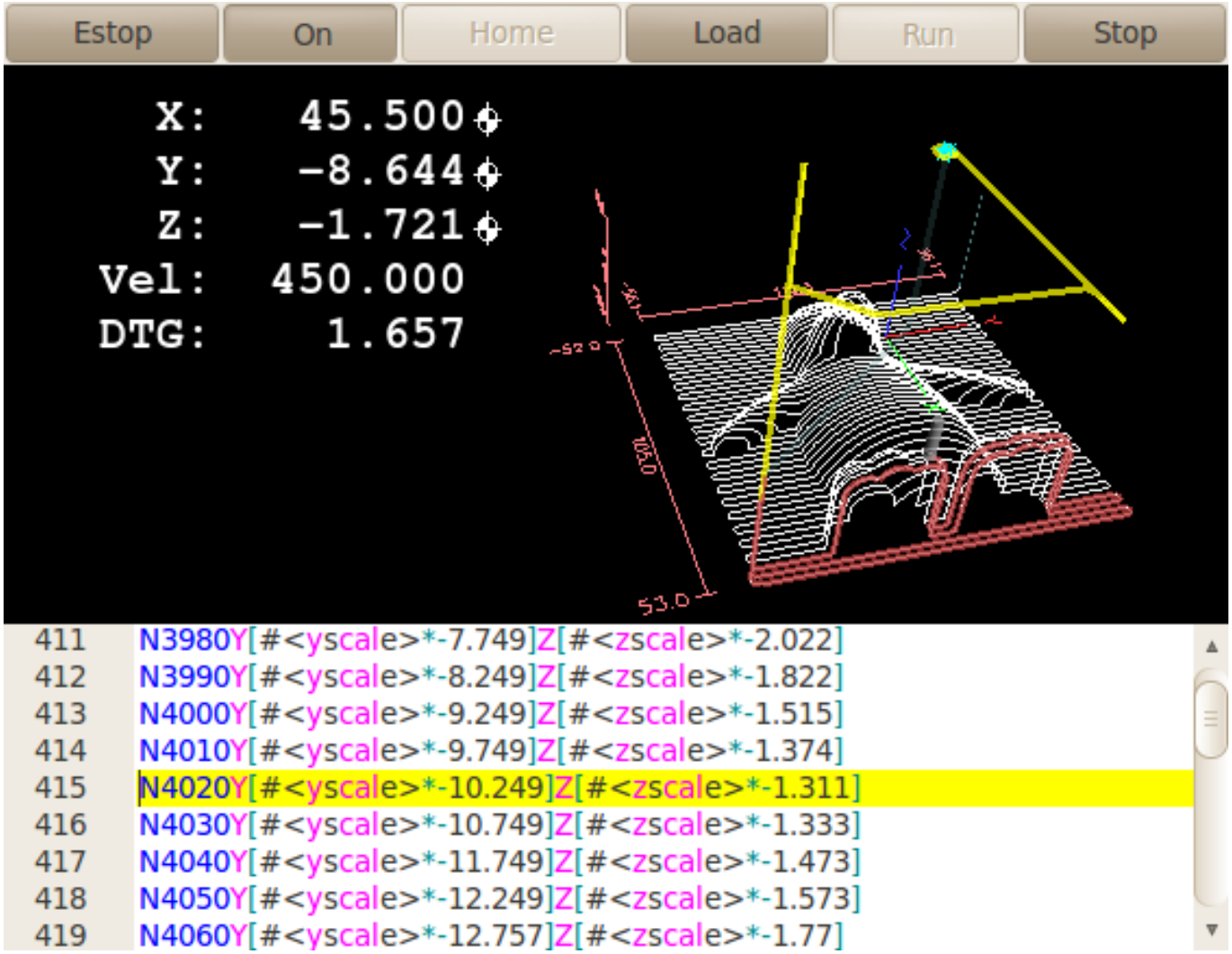
Ok you have loaded the stock Glade file and now can edit it. The first thing you notice is it does not look in the editor like what it is displayed like Gscreen uses some tricks, such as hiding all boxes of buttons except one and changing that one depending on the mode. The same goes for notebooks, some screens use notebooks with the tabs not shown. To change pages in the editor you need to temporarily show those tabs.

When making changes it is far easier to add widgets then subtract widgets and still have the screen work properly making objects *not visible* is one way to change the display without getting errors. This won’t always work some widgets will be set visible again. Changing the names of Gscreen’s regular widgets is probably not gonna work well without changing the Python code, but moving a widget while keeping the name is usually workable.

Gscreen leverages GladeVCP widgets as much as possible, to avoid adding Python code. Learning about GladeVCP widgets is a prerequisite. If the existing widgets give you the function you want or need then no Python code needs be added, just save the Glade file in your configuration folder. If you need something more custom then you must do some Python programming. The name of the parent window needs to be window1. Gscreen assumes this name.

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

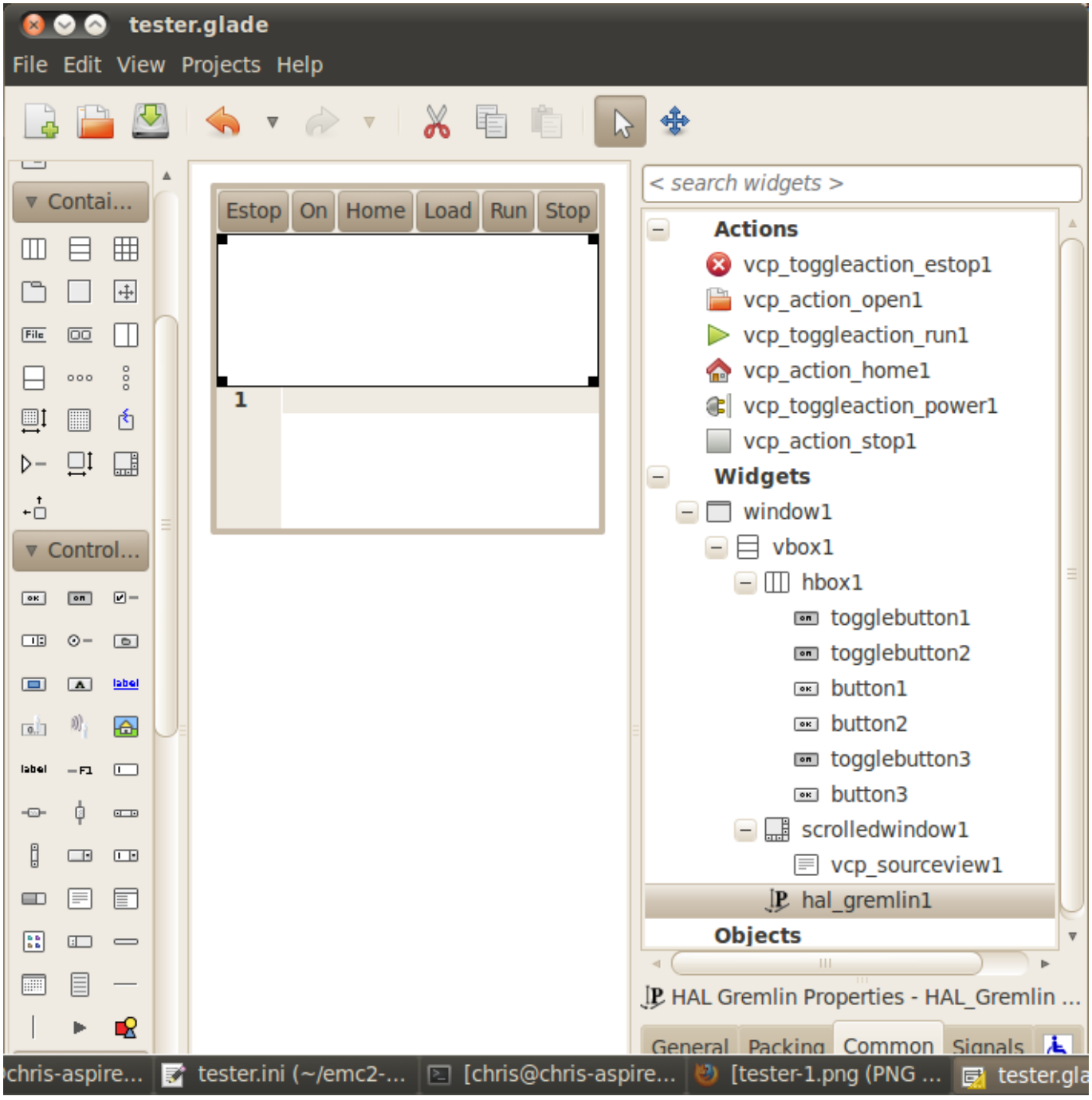



Lassen Sie uns einen einfachen brauchbaren Bildschirm erstellen. Erstellen Sie diesen im Glade-Editor (wenn Sie ein RIP-Paket verwenden, führen Sie ihn von einem Terminal aus, nachdem Sie . scripts/rip-environment verwendet haben).

Zu beachtende Punkte:

- Das Fenster der obersten Ebene muss den Standardnamen ”window1” tragen - Gscreen verlässt sich auf diesen Namen.
- Add actions by right clicking, and selecting *add as toplevel widget* they don’t add anything visual to the window but are added to the right most action list. Add all the ones you see on the top right.
- 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).
- The sourceview widget will try to use the whole window so adding it to a scrolled window will cover this. (This is already done in the example.)
- 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 this screen we are using VCP_actions to communicate to LinuxCNC the actions we want. This allows us standard functions without adding Python code in the handler file. Let’s link the estop toggle button to the estop action Select the estop toggle button and under the general tab look for *Related Action* and click the button beside it. Now select the toggle estop action. Now the button will toggle estop on and off when clicked. Under the general tab you can change the text of the button’s label to describe its function. Do this for all the buttons.

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.

Click the horizontal box that holds the buttons. Click the packing tab and click *expand* to *No*.

Save it as tester.glade and save it in sim/gscreen/gscreen_custom/ folder. Now launch LinuxCNC and click to sim/gscreen/gscreen_custom/tester and start it. If all goes well our screen will pop up and the buttons will do their job. This works because the tester.ini tells gscreen to look for and load tester.glade and tester_handler.py. The tester_handler.py file is included in that folder and is coded just show the screen and not much else. Since the special widgets directly communicate with LinuxCNC you can still do useful things. If your screen needs are covered by the available special widgets then this is as far as you need to go to build a screen. If you want something more there are still many tricks available from just adding *function calls* to get canned behaviour. To coding your own Python code to customize exactly what you want. But that means learning about handler files.

### 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) You can install new keybinding routines. In most cases you won't want to do this, you will want to override the individual keybinding calls. You can also add more keybindings that will call an arbitrary function.
- initialize_pins(self): erzeugt / initialisiert HAL-Pins
- connect_signals(self,handlers): If you are using a completely different screen the default Gscreen you must add this or gscreen will try to connect signals to widgets that are not there. Gscreen's default function is called with self.gscreen.connect_signals(handlers). If you wish to just add extra signals to your screen but still want the default ones call this first then add more signals. If you signals are simple (no user data passed) then you can also use the Glade signal selection in the Glade editor.
- initialize_widgets(self): You can use this to set up any widgets. Gscreen usually calls *self.gscreen.initialize_* which actually calls many separate functions. If you wish to incorporate some of those widgets then just call those functions directly. Or add self.gscreen.init_show_windows() so widgets are just shown. Then if desired, initialize/adjust your new widgets.
- initialize_manual_toolchange(self): Ermöglicht eine vollständige Überarbeitung des manuellen Werkzeugwechselsystems.
- set_restart_line(self.line):
- timer_interrupt(self): allows one to complete redefine the interrupt routine. This is used for calling periodic() and checking for errors from linuxcnc.status.
- 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): Use this to override any user message or manual tool change dialog. Called when the dialog is closed.
- periodic(self): This is called every (default 100) milliseconds. Use it to update your widgets/HAL pins. You can call Gscreen regular periodic afterwards too, self.gscreen.update_position() or just add pass to not update anything. Gscreen's update_position() actually calls many separate functions. If you wish to incorporate some of those widgets then just call those functions directly.

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)
```

When LinuxCNC loads a new file, Gstat will send a callback message to the function *update_filepath*. In this example we update a label with that path name (assuming there is a label named *my_path_label*) in the GLADE file.

#### 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 checks to see if there are *local* XML, handler and/or locale files in the configuration folder. It will use them instead of the default ones (in share/gscreen/skins/) (There can be two separate screens displayed).
4. The main screen is loaded and translations set up. If present the second screen will be loaded and translations set up.
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. Initializes Python's binding to HAL to build a non-realtime component with the Gscreen name.
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 checks the handler file for pins function (*initialize_pins(self)*) or else use Gscreen stock one.
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 tells HAL the Gscreen HAL component is finished making pins and is ready. If there is a terminal widget in the screen it will print all the Gscreen pins to it.
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
```

The -c switch allows one to select a *skin*. Gscreen assumes the Glade file and the handler file use this same name. The optional second screen will be the same name with a 2 (e.g., tester2.glade). There is no second handler file allowed. It will only be loaded if it is present. Gscreen will search the LinuxCNC configuration file that was launched first for the files, then in the system skin folder.

### 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.
  - have one HAL pin to launch the dialog and one to signify it is waiting for response.

- 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

If you wish to use a stock screen but modify its handler file, you need to copy the stock file to your config file folder. Gscreen will see this and use the copied file. But where is the original file? If using a RIP LinuxCNC the sample skins are in /share/gscreen/skins/*SCREENNAME* Installed versions of LinuxCNC have them in slightly different places depending on the distribution used. An easy way to find the location is to open a terminal and start the sim screen you wish to use. In the terminal the file locations will be printed. It may help to add the -d switch to the gscreen load line in the INI.

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 and QtDragon_hd are built with the QtVCP framework. It is the creative vision of forum personality Persei8. Much of it is based on the excellent work of others in the LinuxCNC community. LinuxCNC's version is adapted from Persei8's Github versions. It is primarily meant for 3/4 axes machines such as mills or routers. It works well with a touchscreen and/or mouse. QtDragon supports multiple ways to touch off tools and probing work pieces. You can use LinuxCNC's external offsets capability to automatically raise the spindle during a pause. If you the VersaProbe option and remap code you can add auto tool length probing at tool changes.

---

**Anmerkung**
QtDragon and QtVCP are relatively new programs added into LinuxCNC. Bugs and oddities are possible. Please test carefully when using a dangerous machine. Please forward reports to the forum or maillist.

---

### 10.5.1.1 QtDragon

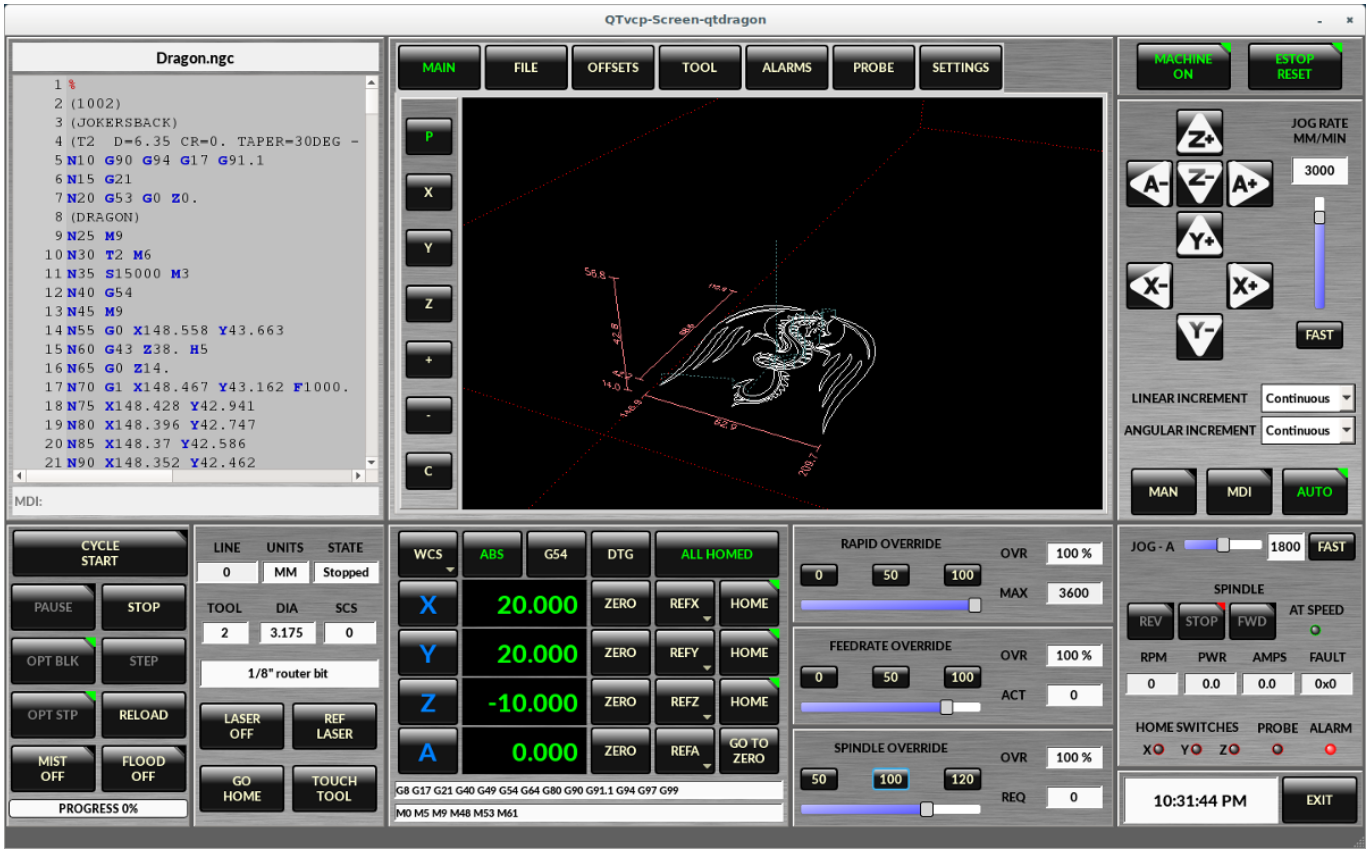



Abbildung 10.26: QtDragon - 3 or 4 axis sample (1440x860) in silver theme

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

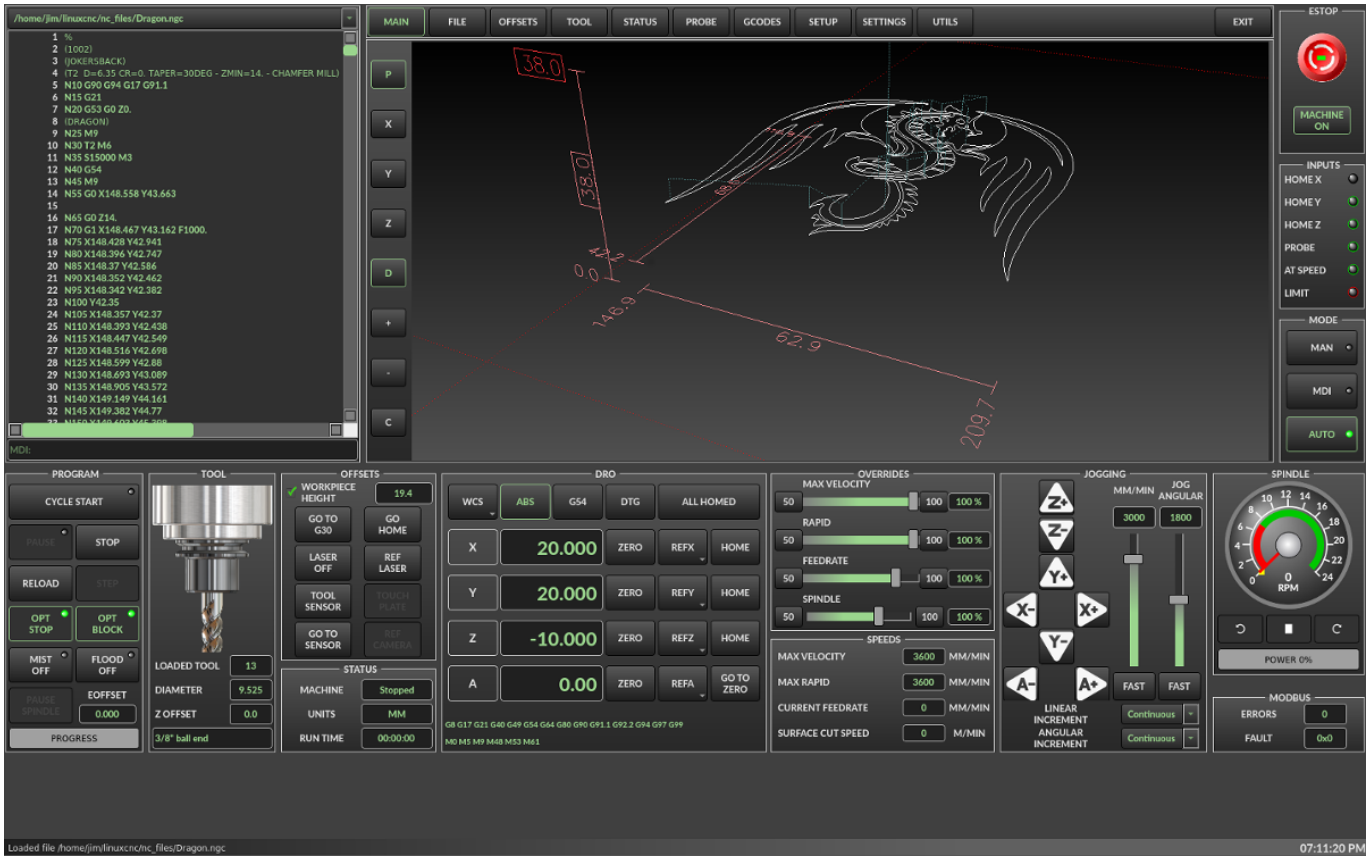



Abbildung 10.27: QtDragon_hd - 3 or 4 axis sample for larger monitors (1920x1056) in dark theme

QtDragon_hd hat ein ähnliches Design wie QtDragon, wurde aber modifiziert, um den zusätzlichen Platz auf modernen, größeren Monitoren zu nutzen. Es gibt einige kleine Unterschiede im Layout und im Nutzen.

QtDragon_hd has a resolution of 1920x1056 and is not resizeable. It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

### 10.5.2 Getting Started - The INI File

If your configuration is not currently set up to use QtDragon, you can change it by editing the INI file sections. For an exhaustive list of options, see the display section of the INI file documentation.

---

**Anmerkung**
You can only have one of each section (e.g., [HAL]) in the INI file. If you see in these docs multiple section options, place them all under the one appropriate section name.

---

#### 10.5.2.1 Anzeige (engl. display)

In the section `[DISPLAY]` change the `DISPLAY =` assignment to read:

- `qtdragon` für eine kleine Version
- `qtdradon_hd` für die große Version.

You can add `-v`, `-d`, `-i`, or `-q` for (respectably) verbose, debug, info or quiet output to the terminal.

```
[DISPLAY]
DISPLAY = qtvcp qtdragon
```

#### 10.5.2.2 Einstellungen

To keep track of preferences, QtDragon looks for a preference text file. Add the following entry under the `[DISPLAY]` heading.
It can use ~ for home directory or `WORKINGFOLDER` or `CONFIGFOLDER` to represent QtVCP's idea of those directories:
This example will save the file in the config folder of the launch screen. (Other options are possible see the QtVCP's screenoption widget docs.)

```
[DISPLAY]
PREFERENCE_FILE_PATH = WORKINGFOLDER/qtdragon.pref
```

#### 10.5.2.3 Protokollierung (engl. logging)

You can specify where to save history/logs.
These file names can be user selected.
In the section `[DISPLAY]` add:

```
[DISPLAY]
MDI_HISTORY_FILE = mdi_history.dat
MACHINE_LOG_PATH = machine_log.dat
LOG_FILE = qtdragon.log
```

#### 10.5.2.4 Override-Kontrollen

These set qtdragon's override controls (1.0 = 100 percent):

```
[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

Set selectable jogging increments.
These increments can be user changed.

```
[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

Set jog speed controls (in units per second)

```
[DISPLAY]
MIN_LINEAR_VELOCITY     = 0
MAX_LINEAR_VELOCITY     = 60.00
DEFAULT_LINEAR_VELOCITY = 50.0
DEFAULT_ANGULAR_VELOCITY = 10
MIN_ANGULAR_VELOCITY = 1
MAX_ANGULAR_VELOCITY = 360
```

#### 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.
These dialogs could be used for such things as low lube oil warnings, etc.

```
[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 optionally embed QtVCP Virtual Control Panels into the QtDragon or QtDragon_hd screen.
These panels can be either user built or builtin QtVCP Panels.
See QtVCP/VCP panels for other available builtin panels.

The `EMBED_TAB_NAME` entry will used as the title for the new tab.(must be unique)
Tab `EMBED_TAB_LOCATION` options include: `tabWidget_utilities`, `tabWidget_setup` and `stackedWidget_ma`
Tab `EMBED_TAB_COMMAND` specifies what embed-able program to run, including any of its command line options.

If using the `tabWidget_utilities` or `tabWidget_setup` locations, an extra tab will appear with the panel.
If using `stackedWidget_mainTab`, a button labelled *User* will appear.
Pressing this button will cycle through displaying all available panels (specified for this location) on the main tab area.

**Sample adding a builtin panel to the utilities tab, i.e., a graphical animated machine using the vismach library.**

```
[DISPLAY]
EMBED_TAB_NAME = Vismach Demo
EMBED_TAB_COMMAND = qtvcp vismach_mill_xyz
EMBED_TAB_LOCATION = tabWidget_utilities
```

This example panel is designed to display additional RS485 VFD data and also to configure a 4 sheave, 2 belt spindle drive via a series of buttons.

```
[DISPLAY]
EMBED_TAB_NAME = Spindle Belts
EMBED_TAB_COMMAND = qtvcp spindle_belts
EMBED_TAB_LOCATION = tabWidget_utilities
```

#### 10.5.2.10 Subroutine Paths

If using NGCGUI, remap or custom M codes routines, LinuxCNC needs to know where to look for the files.
This sample is typical of what is needed for NgcGui, Basic Probe. and Versa Probe remap code.
These paths will need to be adjusted to point to the actual files on your system. RS274NZGC Section Details

```
[RS274NGC]
SUBROUTINE_PATH = :~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ ↩
    ngcgui_lib/utilitysubs; \
~/linuxcnc/nc_files/examples/probe/basic_probe/macros:~/linuxcnc/nc_files/examples/remap- ↩
    subroutines: \
~/linuxcnc/nc_files/examples/ngcgui_lib/remap_lib
```

QtVCP's NGCGUI program also need to know where to open for subroutine selection and pre-selection. NGCGUI_SUBFILE_PATH must point to an actual path on your system and also a path described in SUBROUTINE_PATHS.

```
[DISPLAY]
# NGCGUI subroutine path.
# Thr path must also  be in [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# pre selected programs tabs
# specify filenames only, files must be in the NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

#### 10.5.2.11 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)
```

#### 10.5.2.12 Programmerweiterungen/Filter

You can control what programs are displayed in the filemanager window with program extensions. Create a line with the . endings you wish to use separated by commas, then a space and the description. You can add multiple lines for different selections in the combo box.

```
[FILTER]
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code file (*.ngc,*.nc,*.tap)
```

QtDragon has the ability to send loaded files through a *filter program*. This filter can do any desired task: Something as simple as making sure the file ends with *M2*, or something as complicated as generating G-code from an image. See Filter Programs for more information.

The *[FILTER]* section of the INI file controls how filters work. First, for each type of file, write a *PROGRAM_EXTENSION* line. Then, specify the program to execute for each type of file. This program is given the name of the input file as its first argument, and must write rs274ngc code to standard output. This output is what will be displayed in the text area, previewed in the display area, and executed by LinuxCNC when *Run*.

The following lines add support for the `image-to-gcode` converter included with LinuxCNC and running Python based filter programs:

```
[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
```

#### 10.5.2.13 Sonden-/Touchplate-/Lasereinstellungen

QtDragon has INI entries for two optional probing tab screens available. Comment/uncomment which ever you prefer.

- *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 perform similar probing routines, though Versa probe optionally handles auto tool measurement.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

#### 10.5.2.14 Abort detection

When using qtdragon's probing routines, it is important to detect a user abort request.
By default, LinuxCNC does not report an abort in a useful way for the probe routines.
You need to add a ngc file to print out an error that can be detected. Remap Abort Details

```
[RS274NGC]
# on abort, this ngc file is called. required for basic/versa probe routines. +
ON_ABORT_COMMAND=O <on_abort> call
```

This example code will send a message on abort. The probe routines can detect this sample.
According to the setting above, it would need to be saved as *on_abort.ngc* within LinuxCNC's [RS274NGC] SUBROUTINE_PATHS and [DISPLAY] PROGRAM_PREFIX search paths.

```
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
```

#### 10.5.2.15 Startup codes

You should set default M/G code for start up. These will be overridden by running a NGC file.
These are only sample codes, integrator should choose appropriate codes.

```
[RS274NGC]
# start up G/M codes when first loaded
RS274NGC_STARTUP_CODE = G17 G20 G40 G43H0 G54 G64P0.0005 G80 G90 G94 G97 M5 M9
```

#### 10.5.2.16 Makro-Buttons

QtDragon has up to ten convenience buttons for calling *macro actions*.
These could also call OWord routines if desired.
In the sample configurations they are labelled for moving between current user system origin (zero

point) and Machine system origin.
User origin is the first MDI command in the INI list, machine origin is the second.
This example shows how to move Z axis up first. The commands are separated by the *;*.

The label is set after the comma. The symbols *\n* adds a line break.

```
[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
```

#### 10.5.2.17 Post GUI HAL File

These optional HAL files will be called after QtDragon has loaded everything else.
You can add multiple line for multiple file. Each one will be called in the order they appear.
Calling HAL files after QtDragon is already loaded assures that QtDragon's HAL pins are available.

**Sample with typical entries for the specificion of HAL files to be read after the QtDragon was started. Adjust these lines to match actual requirements.**

```
[HAL]
POSTGUI_HALFILE = qtdragon_hd_postgui.hal
POSTGUI_HALFILE = qtdragon_hd_debugging.hal
```

#### 10.5.2.18 Post GUI HAL Command

These optional HAL commands will be run after QtDragon has loaded everything else.
You can add multiple line. Each one will be called in the order they appear.
Any HAL command can be used.

**Sample with typical files in INI file to load modules after the GUI is available. Adjusti these to match your actual requirements.**

```
[HAL]
POSTGUI_HALCMD = loadusr qtvcp test_probe
POSTGUI_HALCMD = loadusr qtvcp test_led
POSTGUI_HALCMD = loadusr halmeter
```

#### 10.5.2.19 Integrierte Beispielkonfigurationen

The sample configurations `sim/qtdragon/` or `sim/qtdragon_hd` are already configured to use QtDragon as the screen. There are several examples that demonstrate various machine configurations.

### 10.5.3 Tastenbelegungen

QtDragon is not intended to primarily use a keyboard for machine control.
It lacks many keyboard short cuts that for instance AXIS has - but you can use a mouse or touchscreen.
There are several key presses that will control the machine for convenience.

```
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

Buttons that are checkable will change their text colour when checked. This is controlled by the stylesheet/theme

### 10.5.5 Virtuelle Tastatur

QtDragon includes a virtual keyboard for use with touchscreens. To enable the keyboard, check the Use Virtual Keyboard checkbox in the Settings page. Clicking on any input field, such as probe parameters or tool table entries, will show the keyboard. To hide the keyboard, do one of the following:

- Klicken Sie auf den MAIN page Button
- The currently 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

These pins are specific to the QtDragon screen.
There are of course are many more HAL pins that must be connected for LinuxCNC to function.

If you need a manual tool change prompt, add these lines in your postgui file.
QtDragon emulates the hal_manualtoolchange HAL pins - don't load the separate HAL component *hal_manualtoolchange*.

```
net tool-change      hal_manualtoolchange.change   <=  iocontrol.0.tool-change
net tool-changed     hal_manualtoolchange.changed  <=  iocontrol.0.tool-changed
net tool-prep-number hal_manualtoolchange.number   <=  iocontrol.0.tool-prep-number
```

Also if you don't have an automatic tool changer make sure these pins are connected in one of the HAL files:

```
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

This input pin should be connected to indicate probe state.

```
qtdragon.led-probe
```

These pins are inputs related to spindle VFD indicating.
The volt and amp pins are used to calculate spindle power. You must also set the MAX_SPINDLE_POWER in the INI.

```
qtdragon.spindle-modbus-connection
qtdragon.spindle-modbus-errors
qtdragon.spindle-amps
qtdragon.spindle-fault
qtdragon.spindle-volts
```

This bit pin is an output to the spindle control to pause it.
You would connect it to `spindle.0.inhibit`.

```
qtdragon.spindle-inhibit
```

QtDragon spindle speed display and spindle-at-speed LED require that `spindle.0.speed-in` be connected to spindle speed feedback.
Encoder or VFD feedback could be used, as long as the feedback is in revolutions per second (RPS).
If no feedback is available you can have the display show the requested speed by connecting pins like so:

```
net spindle-speed-feedback spindle.0.speed-out-rps => spindle.0.speed-in
```

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
```

These bit/s32/float pins are related to external offsets if they are used:

```
qtdragon.eoffset-clear
qtdragon.eoffset-enable
qtdragon.eoffset-value
qtdragon.eoffset-spindle-count
qtdragon.eoffset-zlevel-count
```

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
```

These output pins are available when setting the Versa Probe INI option. They can be used for auto-tool-length-probe at tool change - with added remap code.

```
qtversaprobe.enable
qtversaprobe.blockheight
qtversaprobe.probeheight
qtversaprobe.probevel
qtversaprobe.searchvel
qtversaprobe.backoffdist
```

### 10.5.7 HAL-Dateien

The HAL files supplied are for simulation only. A real machine needs its own custom HAL files. The QtDragon screen works with 3 or 4 axes with one joint per axis or 3 or 4 axes in a gantry configuration (2 joints on 1 axis).

### 10.5.8 Manueller Werkzeugwechsel

If your machine requires manual tool changes, QtDragon can pop a message box to direct you. QtDragon emulates the hal_manualtoolchange HAL pins - don't load the separate HAL component *hal_manualtoolc*
Hereto you must connect the proper HAL pin in the postgui HAL file, for example:

```
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

The screen is intended to interface to a VFD, but will still work without it.
There are a number of VFD drivers included in the LinuxCNC distribution.
It is up to the end user to supply the appropriate driver and HAL file connections according to his own machine setup.

### 10.5.10 Automatisches Anheben der Z-Achse bei Pausieren der Spindel

QtDragon can be set up to automatically raise and lower the Z axis when the spindle is paused.
When a program is paused, then you press the *Spindle Pause* button to stop the spindle and raise it in Z.
Press the button again to start spindle and lower it, then unpause the program.
If you have the HAL pin `spindle.0.at-speed` connected to a driving pin, the spindle will not lower until the pin is true
You typically connect this to a timer or logic that detects the speed of the spindle.
If that pin is not connected to a driving pin, a dialog will pop up to warn you to wait for the spindle response.
The spindle will lower when you close that dialog.
The amount to raise is set in the *Settings* tab under the heading *SPINDLE RAISE*.
This line edit box can only be directly set when not in Auto mode.
The up/down buttons can be used to adjust the raise amount at any time, including when the spindle is already raised.
The button increments are 1 inch or 5 mm (depending on the units the machine is based on)
This optional behaviour requires additions to the INI and the qtdragon_post_gui HAL file.

In der INI, unter der Überschrift AXIS_Z.

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

In der Datei qtdragon_postgui.hal hinzufügen:

```
# Set up Z axis external offsets
net eoffset_clear    qtdragon.eoffset-clear => axis.z.eoffset-clear
net eoffset_count    qtdragon.eoffset-spindle-count => axis.z.eoffset-counts
net eoffset          qtdragon.eoffset-value <= axis.z.eoffset

# 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 level compensation is a bed levelling/distortion correction function typically used in 3D printing or engraving. It uses a HAL non-realtime component which utilizes the external offsets feature of

LinuxCNC. The component has a HAL pin that specifies an interpolation type, which must be one of cubic, linear or nearest (0, 1, 2 respectively). If none is specified or if an invalid number is specified, the default is assumed to be cubic.

When Z LEVEL COMP is enabled, the compensation component reads a probe data file, which must be called *probe_points.txt*. The file can be modified or updated at any time while compensation is disabled. When next enabled, the file will be reread and the compensation map is recalculated. This file is expected to be in the configuration directory.

The probe data file is generated by a probing program, which itself is generated by an external python program called `gcode_ripper`, which can be launched from the file manager tab using the *G-code Ripper* button.

#### 10.5.11.1 Verwendung von G-code Ripper für die Z-Ebenen-Kompensation

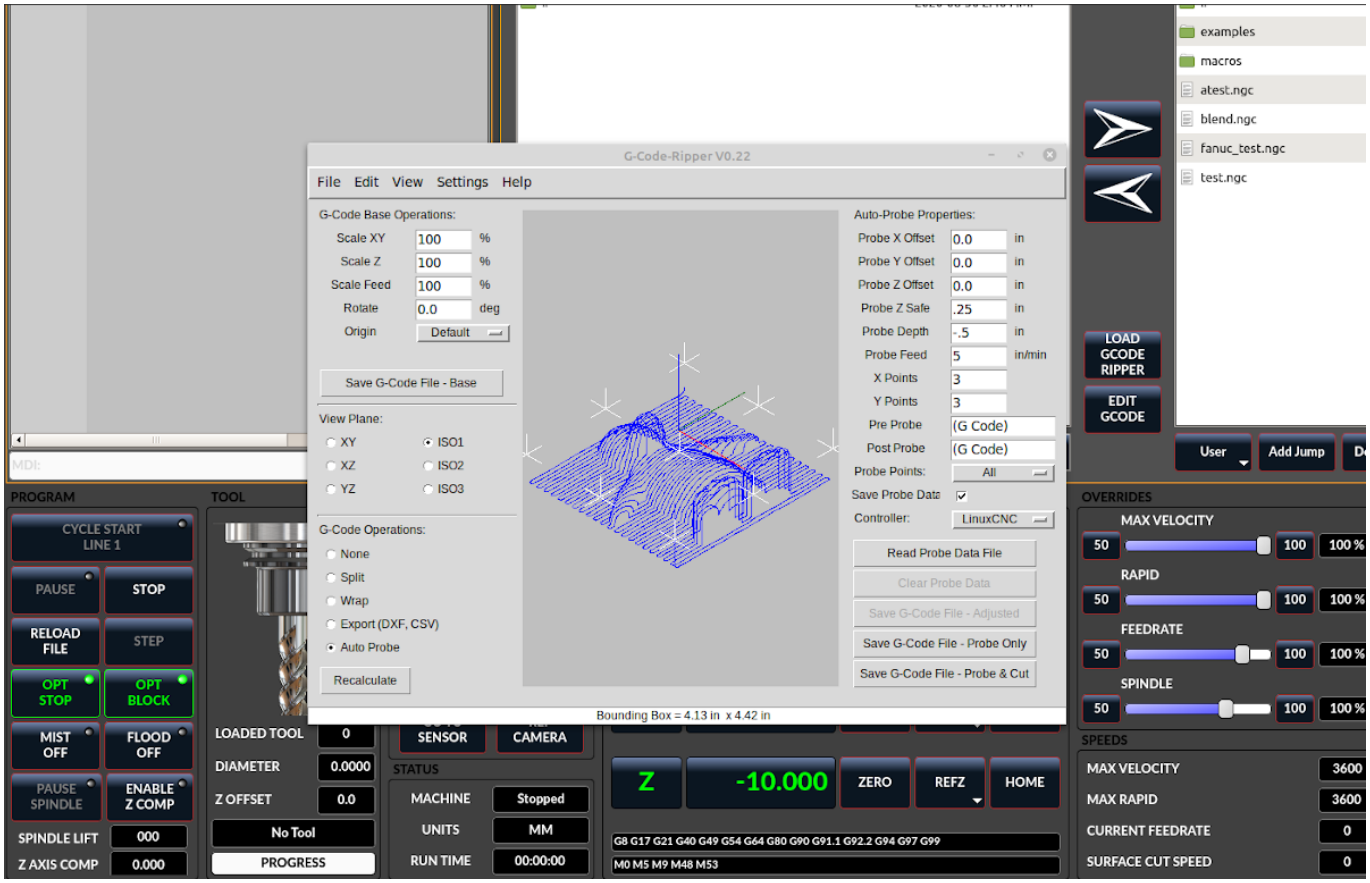



Abbildung 10.28: QtDragon_hd zeigt G-code Ripper

**Anmerkung**
G-code Ripper bietet viele Funktionen, auf die wir hier nicht näher eingehen werden. Diese sind nur in der QtDragon_hd Version verfügbar.

- In qtdragon_hd, switch to the file tab and press the load G-code Ripper button.
- Set origin to match the origin of the G-code file to be probed.
- Under G-Code Operations, check Auto Probe.

- Datei -> G-Code Datei öffnen (Die Datei, die Sie nach der Kompensation ausführen)
- If necessary, make adjustments and press Recalculate.
- Press Save G-Code File - Probe Only.
- Save the generated file to the nc_files folder.
- Exit 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*.
- In qtdragon_hd, press the *Enable Z Comp* button to enable compensation. Look at the status line for indication of success or failure. Active compensation will be displayed beside the label: *Z Level Comp* While jogging that display should change based on the compensation component.

---

**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.

---

**Sample postgui HAL file for combined spindle raise and Z Level compensation**

```
# Komponenten laden
#############################################################################

loadrt logic names=logic-and personality=0x102
addf logic-and servo-thread

# load a summing component for adding spindle lift and Z compensation
loadrt scaled_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.fade-height 0.0
setp z_level_compensation.method 1

# Signale mit der Bewegungskomponente von LinuxCNC verbinden
#############################################################################

net eoffset-clear    axis.z.eoffset-clear
net eoffset-counts   axis.z.eoffset-counts
setp axis.z.eoffset-scale .001
net eoffset-total          axis.z.eoffset
setp axis.z.eoffset-enable True

# external offsets for spindle pause function
#############################################################################
net eoffset-clear              qtdragon.eoffset-clear
net eoffset-spindle-count   <= qtdragon.eoffset-spindle-count
net spindle-pause              qtdragon.spindle-inhibit     spindle.0.inhibit

## Z level compensation
#####################################################
net eoffset-clr2            z_level_compensation.clear      => logic-and.in-01
net xpos-cmd                z_level_compensation.x-pos      <= axis.x.pos-cmd
net ypos-cmd                z_level_compensation.y-pos      <= axis.y.pos-cmd
```

```
net zpos-cmd                z_level_compensation.z-pos      <= axis.z.pos-cmd
net z_compensation_on       z_level_compensation.enable-in  <= qtdragon.comp-on
net eoffset-zlevel-count    z_level_compensation.counts     => qtdragon.eoffset-zlevel- ↩
    count

# add Z level and scaled spindle raise level values together
net eoffset-spindle-count   scaled-s32-sums.0.in0
net eoffset-zlevel-count    scaled-s32-sums.0.in1
setp scaled-s32-sums.0.scale0 1000
net eoffset-counts          scaled-s32-sums.0.out-s
```

### 10.5.12 Sondieren

The probe screen has been through basic testing but there could still be some minor bugs. When running probing routines, use extreme caution until you are familiar with how everything works. Probe routines run without blocking the main GUI. This gives the operator the opportunity to watch the DROs and stop the routine at any time.

---

**Anmerkung**
Die Sondierung ist sehr unempfindlich gegenüber Fehlern; überprüfen Sie die Einstellungen vor der Verwendung.

---

QtDragon has 2 methods for setting Z0. The first is a touchplate, where a metal plate of known thickness is placed on top of the workpiece, then the tool is lowered until it touches the plate, triggering the probe signal. The current user system's (G5x) Z0 is set to probe height - the entered plate thickness.

The second method uses a tool setter in a fixed position and a known height above the table where the probe signal will be triggered. In order to set Z0 to the top of the workpiece, it has to know

1. how far above the table the probe trigger point is (tool setter height) and
2. how far above the table the top of the workpiece is.

This operation has to be done every time the tool is changed as the tool length is not saved.

For touching off with a touch probe, whether you use the touchplate operation with thickness set to 0 or use a probing routine, the height from table to top of workpiece parameter is not taken into account and can be ignored. It is only for the tool setter.

#### 10.5.12.1 Versa Probe

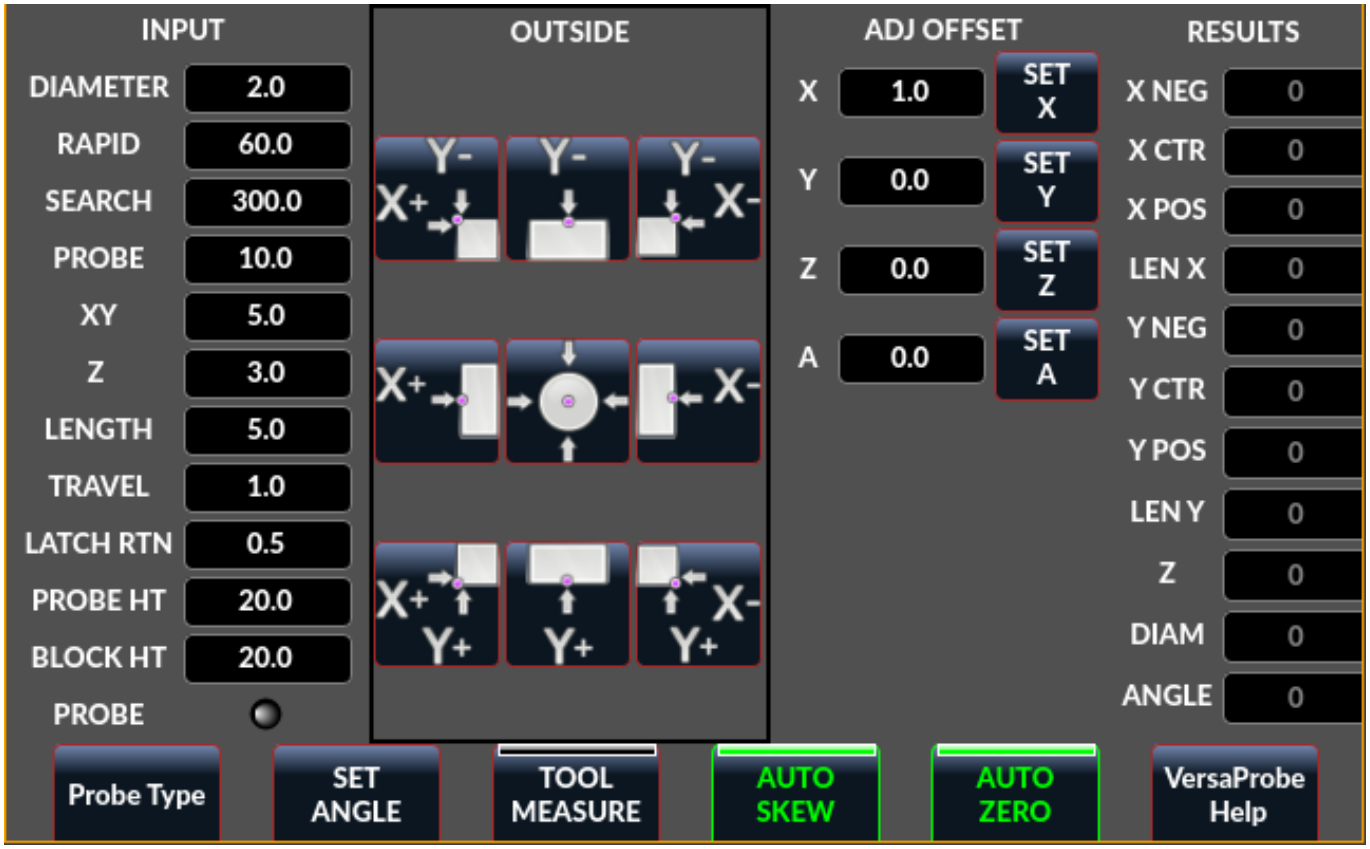



Abbildung 10.29: QtDragon - Versa-Probe-Option

Versa probe is used to semi-automatically probe work pieces to find edges, centers and angles.
It can also be sued to auto probe tool length at tool changes with added remap code.

You must carefully set the *Probing Parameters*:

- *DIAMETER*:: This is the diameter of the probe tip. The accuracy of probe measurements is directly affected by the accuracy of the probe tip diameter.
- *TRAVEL*:: The distance that the probe will travel during the initial search. If the search distance is too short, you will receive a message like ”G38 finished without making contact”. For safety reasons, it is recommended to set this parameter to 3-4 mm more than probe stylus diameter.
- *LATCH RTN*:: The distance the probe is retracted after making initial contact with the workpiece. This should be a short distance because the second approach will be at a slow speed, but large enough for the probe to break contact and bring it to the search ready state. If the Latch Rtn distance too large, you will end up spending a lot of time waiting for the search to complete. Recommendation: 1-2 mm
- *SEARCH*:: This is the feed rate at which the probe searches for the target workpiece in machine units per minute. The search speed should be slow enough to give an acceptable initial accuracy, but fast enough to not waste time waiting for movement. Recommendation: 200-500 mm/min.
- *PROBE*:: Once initial contact has been made and the probe is retracted, it will wait for 0.5 seconds before performing the search again at a lower speed, the probe velocity. This lower speed ensures the machine can stop movement as quickly as possible on contact with the workpiece.

- *RAPID*:: Axis movements not associated with searching are done at the speed defined by RAPID in machine units per minute.
- *SIDE/EDGE LENGTH*:: This is the distance the probe will move at the rapid rate to the position where it will begin a search. If measuring a corner, it will move EDGE LENGTH units away from the corner, then move away from the workpiece by XY CLEARANCE, lower by Z CLEARANCE and begin the initial search. If measuring an inner circle, then EDGE LENGTH should be set to the approximate radius of the circle. Note: NOT the diameter.
- *PROBE HT*:: The height of the tool sensor from the machine table surface. This value is used to calculate the Z zero height for the current work coordinate system when using the probe with a tool setter sensor.
- *BLOCK HT*:: The height of the top of the workpiece from the machine table surface. This value is used to calculate the Z zero height for the current work coordinate system when using the probe with a tool setter sensor.
- *XY CLEARANCE*:: The distance that the probe will move away from an edge or corner before performing a search. It should be large enough to ensure that the probe will not contact the workpiece or any other fixtures before moving down. It should be small enough to avoid excessive waiting for movement while searching.
- *Z CLEARANCE*:: The distance that the probe will move down before performing a search. If measuring an inside hole, the probe could be manually jogged to the starting Z height and then set Z CLEARANCE to 0.

There are three toggle buttons:

- *Auto Zero* This selects if after probing the relevant axis is set to zero in the current user system.
- *Auto Skew* This selects if after probing, the system will be rotated or just display the calculated rotation.
- *Tool Measure* This (if integrated) turns auto tool probing on and off.

### 10.5.12.2 Basic probe

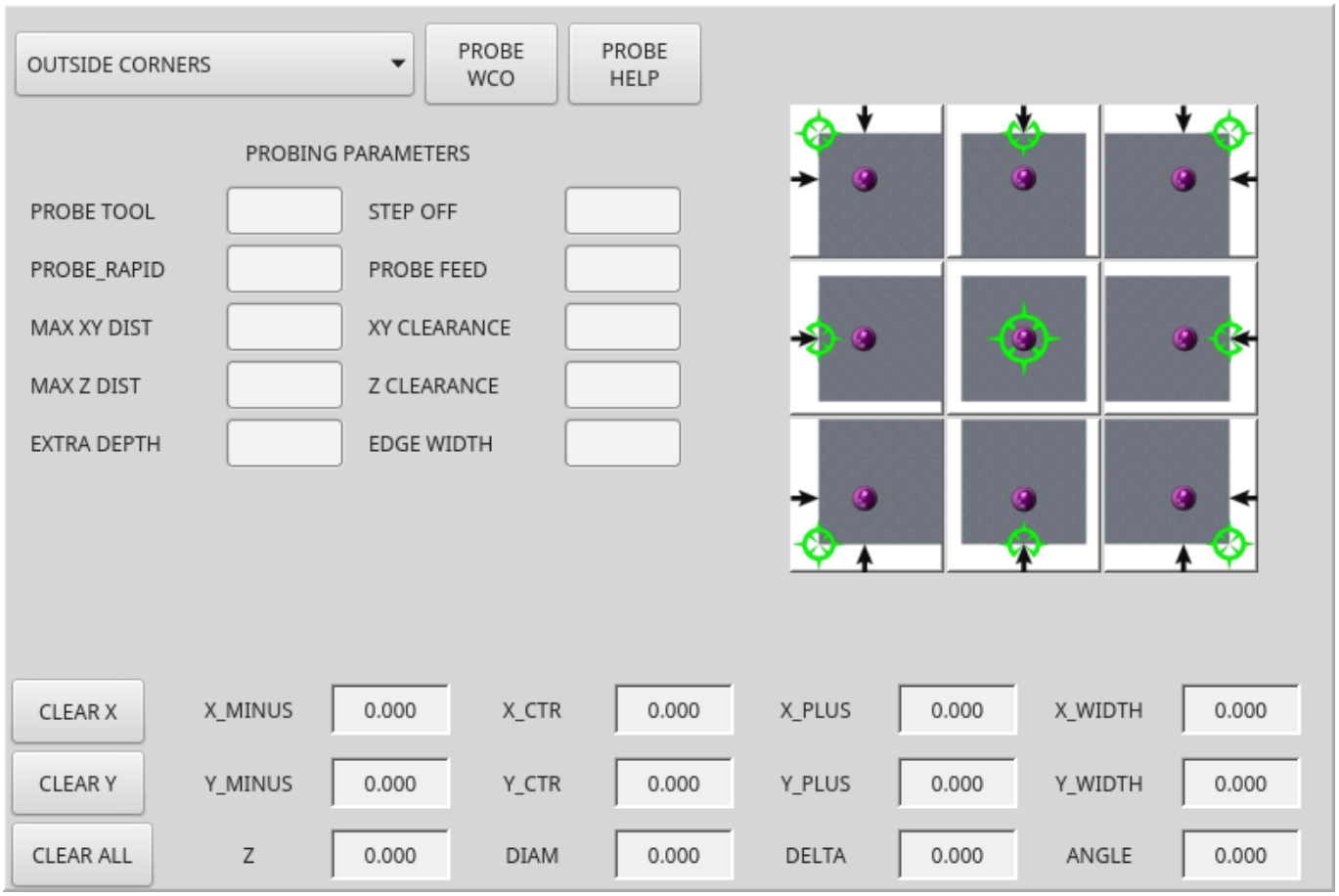



Abbildung 10.30: QtDragon - Grundlegende Sondenoption

Basic probe is used to semi-automatically probe work pieces to find edges, centers and angles. The combo box allows selecting the basic type of probing buttons shown:

- Outside Corners
- Inside Corners
- Edge Angles
- Boss and Pockets
- Ridge and Valleys
- Calibration

You must carefully set the *Probing Parameters*:

- *Probe Tool*: will only allow probing if this tool number is in the spindle
- *Probe Diameter*: the size of the probe tip
- *Probe Rapid*: the speed of rapid moves in machine units
- *Probe Search*: the speed of the first *rough* search in machine units

- *Probe Feed*: the speed of the second *fine* search in machine units
- *Step Off*: back off and re-probe distance
- *Max XY Distance*: the maximum distance the probe will search for in X and Y before failing with error
- *Max Z Distance*: the maximum distance the probe will search for in Z before failing with error
- *XY Clearance*: clearance distance from probe to wall edge before rapid traversing down in Z and *rough* probing
- *Z Clearance*: clearance distance from probed to top of material
- *Extra Depth*: distance from top of material to desired probe depth

There are also hint parameters depending on selected probing type:

- *Edge Width*: desired distance from the probe start position, along wall edge before starting to probe
- *Diameter Hint*: used by Round Boss or Round Pocket probing (start move: 1/2 diameter plus XY clearance)
- *X Hint*: used by Rectangular Boss/Pocket probing (start move: 1/2 X length plus XY clearance)
- *Y Hint*: used by Rectangular Boss/Pocket probing (start move: 1/2 Y length plus XY clearance)

After setting the parameters and hints:

- Manually move the probe to the approximate position represented by the green target on the button.
- Confirm the parameters are reasonable.
- Press the desired probing button.

The probing routine will start immediately.

---

**Anmerkung**
Pressing the stop button or the keyboard escape key, will abort the probing.

---

Lets discuss inside corner probing using the top right button in Basic Probe (back_right_inside). While all probe entries must be correct, the most important settings to change for each each probe:

- XY CLEARANCE - distance away from edge before rough probing,
- Z CLEARANCE - distance from probe to top of material,
- EXTRA DEPTH - distance to lower probe from top of material,
- EDGE WIDTH - distance along edge wall (away from corner) to start probing.

---

**Anmerkung**
These distance are always to be set in *machine units* (mm for metric machine, inch for imperial machine).

---

Preset:

- manual set probe at the intersection of the edges (ie corner) of material as described by the green bullseye on the button. Set it Z CLEARANCE above the top of material. These can be done by eye.
- set EXTRA CLEARANCE to a value that you want the probe to go below the *top* of material. (So the probe will move from its start position down Z Clearance + Extra Clearance distance.)
- set XY CLEARANCE to a value that definitely gives clearance from the wall so when the probe goes down it does not hit anything.
- set EDGE WIDTH to a value that describes the distance measured from the corner, along the wall to where you wish to probe. this edge distance will be used along the X wall and then the Y wall.

Sequence after pressing the probe button:

1. Rapid EDGE WIDTH distance away from corner at the same time moving XY CLEARANCE away from edge. So this is a slightly diagonal move.
2. Move probe down by Z CLEARANCE + EXTRA DEPTH,
3. probe wall twice (rough and fine),
4. move diagonally to the other wall as set by EDGE WIDTH and XY CLEARANCE,
5. probe wall twice,
6. raise probe up by Z CLEARANCE + EXTRA DEPTH (returns to starting height),
7. rapid back to starting corner (now calculated using the probed walls),
8. if auto zero button is enabled, set X and Y of the current user system to zero.

### 10.5.13 Touch-Platte

Abbildung 10.31: QtDragon Touch-Fläche (engl. touch plate)

You can use a conductive touch plate or equivalent to auto touch off (zero the user coordinate) for the Z position of a tool. There must be a tool loaded prior to probing. In the tool tab or settings tab, set the touch plate height, search and probe velocity and max. probing distance.

---

**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.

---

Place the plate on top of the surface you wish to zero Z on. Connect the probe input wire to the tool (if using a conductive plate). There is a LED to confirm the probe connection is reliable prior to probing. Move the tool manually within the max probe distance. Press the *Touch Plate* button. The machine will probe down twice and the current user offset (G5X) will be zeroed at the bottom of the plate by calculation from the touchplate height setting.

### 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:

- Touch off your workpiece in X and Y.
- Measure the height of your block from the base, where your tool switch is located, to the upper face of the block (including chuck etc.).
- 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**
When fist setting up auto tool measurement, please use caution until you confirm tool change and probe locations - it is easy to break a tool/probe. Abort will be honoured while the probe is in motion.

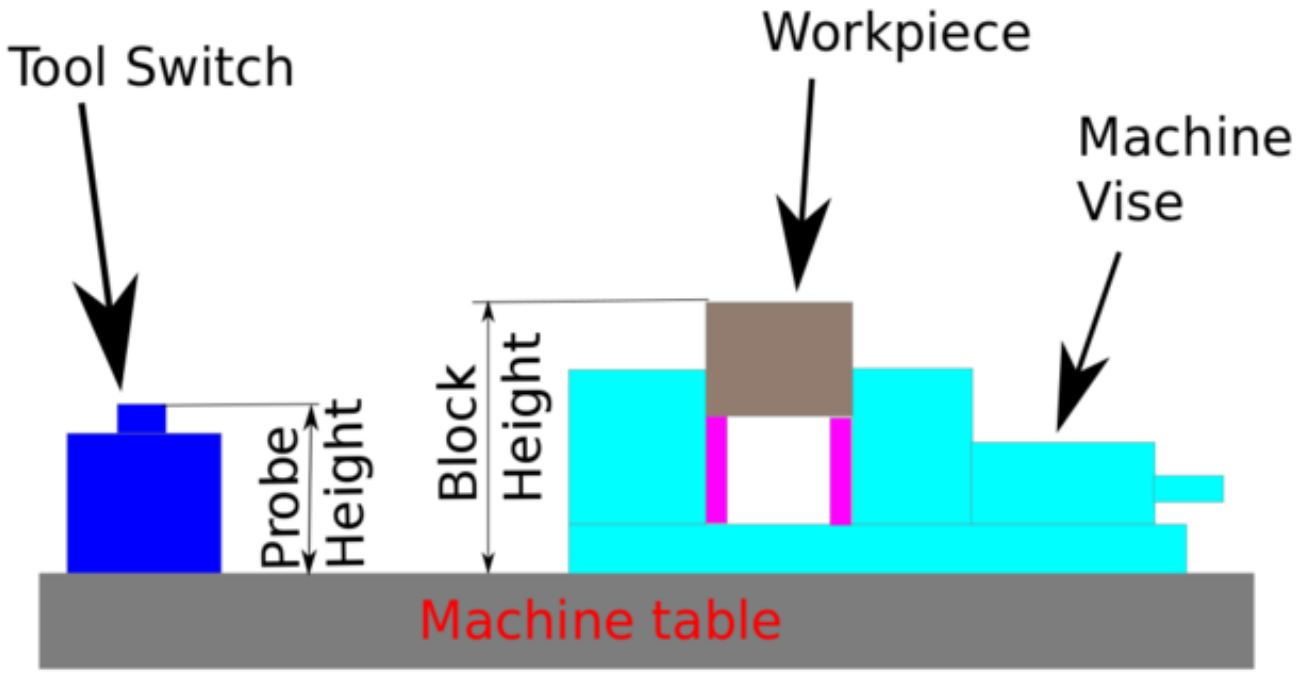



Abbildung 10.32: Automatische Werkzeugvermessung

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.

The sequence of events (using the default files):

- Rapid move in Z to position defined in the INI's [TOOL_CHANGE] Z
- Rapid move in X and Y to number defined in INI's [TOOL_CHANGE] X and Y
- Request tool change
- Rapid move in X and Y to position defined in the INI's [VERSA_TOOLSETTER] X and Y
- Rapid move down in Z to position defined in the INI's [VERSA_TOOLSETTER] Z
- Probe down in Z to maximum defined in the INI's [VERSA_TOOLSETTER] MAXPROBE
- Return Z to position defined in the INI's [TOOL_CHANGE] Z

**Anmerkung**
The [TOOL_CHANGE] Z position should be high enough so the tool will not hit the tool probe when moving to the [VERSA_TOOLSETTER] X and Y position. MAXPROBE distance needs to be high enough for the tool to touch the probe.

### 10.5.14.1 Werkstückhöhe Antasten

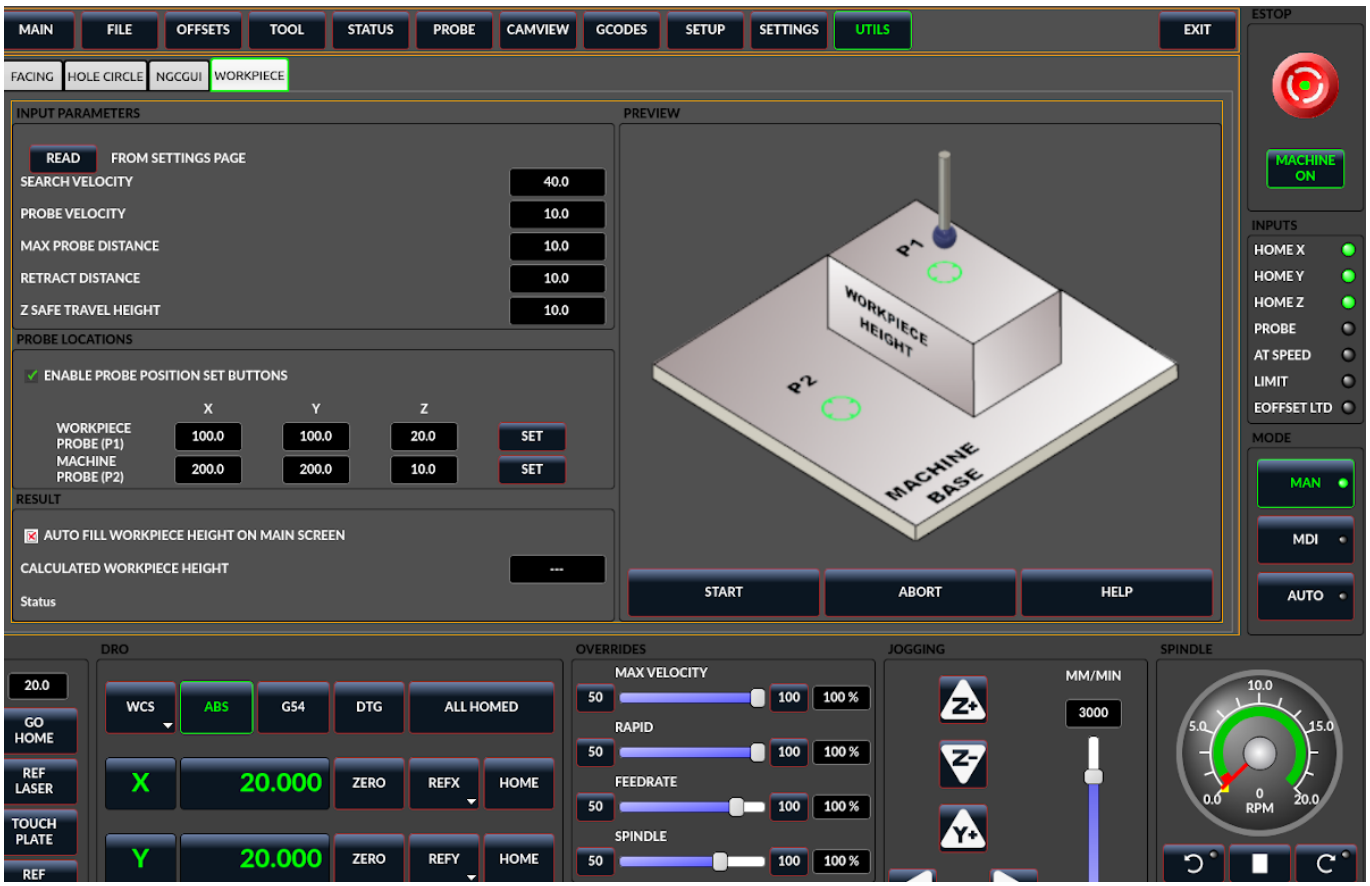



Abbildung 10.33: QtDragon_hd - Werkstück Höhenabtastung

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

- When checked, the SET buttons are enabled.
- 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

- causes all jog and probe routines currently executing to stop.

HELP-Button

- displays this help file.
- 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**
Setting incorrect values can lead to crashes into fixtures on the machine work surface. Initial testing with no tool and safe heights is recommended.

#### 10.5.14.2 Werkzeugmess-Pins

Versaprobe offers 5 output pins for tool measurement purpose. The pins are used to be read from a remap G-code subroutine, so the code can react to different values.

- `qtversaprobe.enable` (HAL_BIT) measurement enabled or not tool. Reflects screen button state.
- `qtversaprobe.blockheight` (HAL_FLOAT) the measured height of the top face of the workpiece. Reflects screen entry.
- `qtversaprobe.probeheight` (HAL_FLOAT) the toolsetter probe switch height. Reflects screen entry.
- `qtversaprobe.searchvel` (HAL_FLOAT) die Geschwindigkeit, mit der nach dem Schalter für den Werkzeugmesstaster gesucht wird
- `qtversaprobe.probevel` (HAL_FLOAT) the velocity to probe tool length. Reflects screen entry.
- *qtversaprobe.backoffdist* (HAL_FLOAT) the distance the probe backs off after triggering. Reflects screen entry.

#### 10.5.14.3 Änderungen an der INI-Datei für Werkzeugmessungen

Modify your INI file to include the following:

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 Section Details
Remap Statement Details
Remap Abort Details

---

**Anmerkung**
These default entries should work fine in most situations. Some systems may need to use *linuxcnc/nc_files/examples/* instead of *linuxcnc/nc_files/*. please check that paths are valid. Custom entries pointing to modified file are possible.

---

```
[RS274NGC]

# Adjust this paths to point to folders with stdglue.py, qt_auto_tool_probe.ngc and  ↩
    on_abort.ngc
# or similarly coded custom remap files.
SUBROUTINE_PATH = ~/linuxcnc/nc_files/remap-subroutines:\
~/linuxcnc/nc_files/remap_lib

# ist die Sub, die aufgerufen wird, wenn ein Fehler beim Werkzeugwechsel auftritt.
ON_ABORT_COMMAND=O <on_abort> Aufruf

# Der Remap-Code für die automatische Werkzeugsonde von Z der Versaprobe von QtVCP
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

The position of the tool sensor and the start position of the probing movement.
All values are absolute (G53) coordinates, except MAXPROBE, which is expressed in relative movement.

All values are in machine native units.
X,Y,Z set the tool setter probe location.
auto probe action sequence (this could be changed with a modified ngc remap file):

- go to TOOLCHANGE Z position
- go to TOOLCHANGE XY position.
- wait for manual tool change acknowledgement
- go to VERSA_TOOLSETTER XY position
- go to VERSA_TOOLSETTER Z position
- fast probe
- slow probe
- go to TOOLCHANGE Z position

Z_MAX_CLEAR is the Z position to go to before moving to the tool setter when using the *Travel to Toolsetter button*.
*Travel to Toolsetter* Action sequence:

- go to VERSA_TOOLSETTER Z_MAX_CLEAR Z position
- go to VERSA_TOOLSETTER XY position
- go to VERSA_TOOLSETTER Z position.

```
[VERSA_TOOLSETTER]
X = 10
Y = 10
Z = -20
Z_MAX_CLEAR = -2
MAXPROBE =  -20
```

This is not named TOOL_CHANGE_POSITION on purpose - **canon uses that name and will interfere otherwise**. The position to move the machine before giving the change tool command. All values are in absolute coordinates. All values are in machine native units.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

The Python section sets up what files LinuxCNC's Python interpreter looks for, e.g., `toplevel.py` file in the `python` folder in the configuration directory: These default entries should work fine in most situations. Some systems may need to use *linuxcnc/nc_files/examples/* instead of *linuxcnc/nc_files/*. Custom entries pointing to modified file are possible.

```
# The path start point for all remap searches, i.e. Python's sys.path.append()
PATH_APPEND = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python
# path to the tremap's 'toplevel file
TOPLEVEL = ~/linuxcnc/nc_files/remap_lib/python-stdglue/python/toplevel.py
```

#### 10.5.14.4 Required HAL Connections

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 run-from-line is not very user friendly. E.g., it does not start the spindle or confirm the proper tool. Also, it does not handle subroutines well. If used it is best to start on a rapid move.

---

### 10.5.16 Laser-Buttons

The LASER ON/OFF button in intended to turn an output on or off which is connected to a small laser crosshair projector. When the crosshair is positioned over a desired reference point on the workpiece, the REF LASER button can be pushed, which then sets the X and Y offsets to the values indicated by the LASER OFFSET fields in the Settings page.

### 10.5.17 Beschreibung der Registerkarten

Tabs allow the user to select the most appropriate info/control on the top three panels. If the on screen keyboard is showing and the user wishes to hide it but keep the current tab, they can do that by pressing the current show tab. In QtDragon, there is a splitter handle between the G-code text display and the G-code graphical display. One can use this to split the size between the two areas. This can be set differently in each tab and in each mode.

#### 10.5.17.1 Hauptregisterkarte

This tab displays the graphical representation of the current program. The side buttons will control the display.

- *User View*: Select/restore a user set view of the current program.
- *P,X,Y,Z*: Set standard views.
- *D*: Toggle display of dimensions.
- +, -: Zoom controls.
- *C*: Clear graphics of tool movement lines.

In `qtdragon_hd` there are also macro buttons available on the right side. Up to tens buttons can be defined in the INI.

#### 10.5.17.2 Registerkarte ”Datei”

You can use this tab to load or transfer programs. Editing of G-code programs can be selected from this tab. With `qtdragon_hd`, this is where you can load the *G-code Ripper*.

#### 10.5.17.3 Registerkarte ”Offsets”

You can monitor/modify system offsets from this tab. There are convenience buttons for zeroing the rotation.G92 and current G5x user offset.

#### 10.5.17.4 Registerkarte ”Werkzeug”

You can monitor/modify tool offsets from this tab. Adding and deleting tools from the tool file can also be done from this tab. When this tab is selected the individual home buttons in the DRO area will change to tool offset setting buttons. They will return to home buttons when you select another tab. Pressing this tool button will drop down a when menu of options:

- Set Current Tool Position
- Adjust Current Tool Position
- Zero Current Tool Position
- Set Tool Offset Directly
- Reset To Last

#### 10.5.17.5 Registerkarte ”Status”

A time-stamped log of important machine or system events will be shown here. Machine events would be more suited to an operator, where the system events may help in debugging problems.

#### 10.5.17.6 Registerkarte ”Sonde

Probing routines options are displayed on this tab. Depending on INI options, this could be VersaProbe or BasicProbe style. They are functionally similar. QtDragon_hd will also show a smaller graphics display window.

#### 10.5.17.7 Camview-Registerkarte

If the recognized webcam is connected, this tab will display the video image overlaid with a cross-hair, circle and degree readout. This can be adjusted to suit a part feature for such things as touchoff. The underlying library uses openCV Python module to connect to the webcam.
To adjust the X or Y size aspect ratio, or camera number, look in the preference file for:

```
[CUSTOM_FORM_ENTRIES]
Camview xscale = 100
Camview yscale = 100
Camview cam number = 0
```

These are in percent, usually the range will be 100 - 200 in one axis.
The preference file can only be edited when QtDragon is not running.

#### 10.5.17.8 G-Codes Registrierkarte

This tab will display a list of LinuxCNC's G-code. if you click on a line, a description of the code will be displayed.

#### 10.5.17.9 Registerkarte "Einstellungen"

It's possible to load HTML or PDF file (.html / .pdf ending) with setup notes, and will be displayed in the setup tab.
If you load a G-code program and there is an HTML/PDF file of the same name, it will load automatically.
Some program, such as Fusion 360 and Aspire will create these files for you. You can also write your own HTML docs with the included SetUp Writer button.
There are three sub tabs:

- *HTML* - any loaded HTML pages are displayed here. The navigation buttons work on this page.
- *PDF* - any loaded PDF setup pages are displayed here.
- *PROPERTIES* - when a program is loaded its G-code properties are displayed here.

There are navigation buttons for HTML page:

- The up arrow returns you to the default HTML page.
- The left arrow moves backward one HTML page.
- The right arrow moves forward one HTML page.

If you wish to include a custom default HTML page, name it *default_setup.html* and place it in your configuration folder.
Custom QtVCP panels can be displayed in this tab by setting the EMBED_TAB_LOCATION option to *tabWidget_setup*.

Abbildung 10.34: 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”

This tabs will display another stab election of G-code utility programs:

- *Facing*: allows quick face milling of a definable area at angles of 0,45 and 90 degrees.
- *Lochkreis*: ermöglicht die schnelle Einstellung eines Programms zum Bohren eines Lochkreises mit definierbarem Durchmesser und Anzahl der Löcher.
- *NGCGUI*: is a QtVCP version of the popular G-code subroutine builder/selector, see Widgets-NGCGUI.

Custom QtVCP panels can be displayed here by setting the EMBED_TAB_LOCATION option to `tabWidget_ut`

#### 10.5.17.12 User Tab

This tab will only be displayed if an embedded panel has been designated for the location `stackedWidget_mai`
If more then one embedded tab has been designated, then pressing the user tab will cycle through them.

### 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.

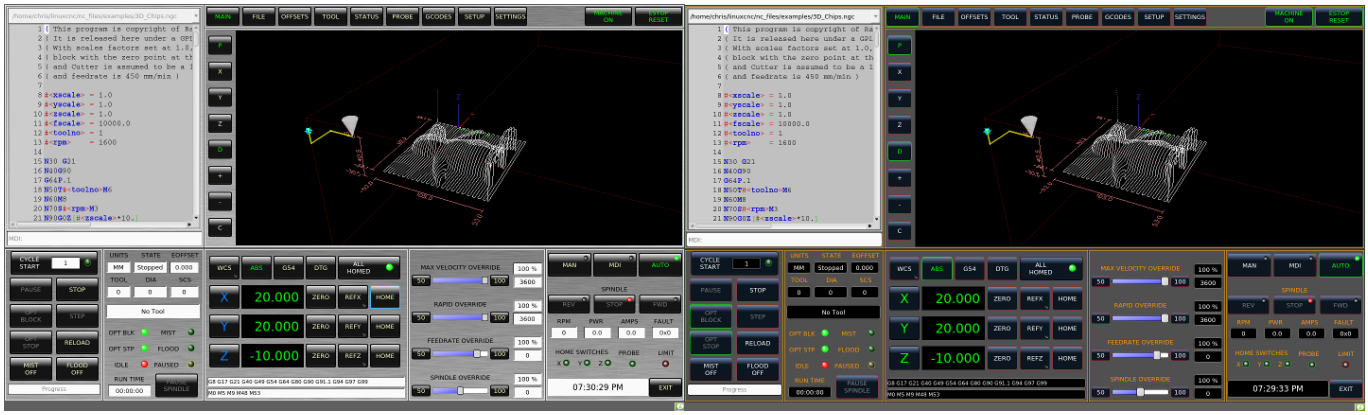


Abbildung 10.35: QtDragon - Zwei Stil-Beispiele

### 10.5.19 Anpassung

A general overview of Customizing Stock Screens.

#### 10.5.19.1 Stylesheets

Stylesheets can be leveraged to do a fair amount of customization, but you usually need to know a bit about the widget names. Pressing F12 will display a stylesheet editor dialog to load/test/save modification. Sometimes these lines will be present and you can change them, otherwise you will need to add them.

For instance, to change the DRO font (look for this entry and change the font name):

```
DROLabel,
StatusLabel#status_rpm {
    border: 1px solid black;
    border-radius: 4px;
    font: 20pt ”Noto Mono”;
}
```

To change the DRO display font and display format:

```
DROLabel {
    font: 25pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.5f’;
    qproperty-metric_template: ’%10.4f’;
    qproperty-angular_template: ’%11.2f’;
}
```

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”;
}
```

To change the Offsets display font and format:

```
ToolOffsetView {
    font: 20pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.1f’;
    qproperty-metric_template: ’%10.1f’;

}

OriginOffsetView {
    font: 12pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.1f’;
    qproperty-metric_template: ’%10.1f’;
}
```

To stop the blur effect with dialogs:

```
#screen_options {
    qproperty-focusBlur_option: false;
}
```

Change the G-code text display colors/fonts:

```
}
EditorBase{
 background:black;
qproperty-styleColorBackground:grey;
qproperty-styleColor0: black;
qproperty-styleColor1: darkblue;
qproperty-styleColor2: blue;
qproperty-styleColor3: red;
qproperty-styleColor4: lightblue;
qproperty-styleColor5: white;
qproperty-styleColor6: lightGreen;
qproperty-styleColor7: yellow ;
qproperty-styleColorSelectionText: white;
qproperty-styleColorSelectionBackground: blue;
qproperty-styleFont0: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont1: ”Times,15,-1,5,90,1,0,1,0,0”;
qproperty-styleFont2: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont3: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont4: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont5: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont6: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont7: ”Times,15,-1,5,90,0,0,1,1,0”;
}
```

To have the manual spindle buttons also incrementally increase/decrease speed:

```
#action_spindle_fwd{
    qproperty-spindle_up_action: true;
}
#action_spindle_rev{
    qproperty-spindle_down_action: true;
}
```

#### 10.5.19.2 Qt Designer und Python-Code

All aspects of the GUI are fully customization through Qt Designer and/or Python code. This capability is included with the QtVCP development environment. The extensive use of QtVCP widgets keeps the

amount of required Python code to a minimum, allowing relatively easy modifications. The LinuxCNC website has extensive documentation on the installation and use of QtVCP libraries. See QtVCP for more information about QtVCP in general.
QtDragon can also utilize QtVCP's rc file to do minor python code modifications without using a custom handler file.

```
[DISPLAY]
USER_COMMAND_FILE = CONFIGFOLDER/qtdragonrc.py
```

See Modifying Screens for more information about customization.

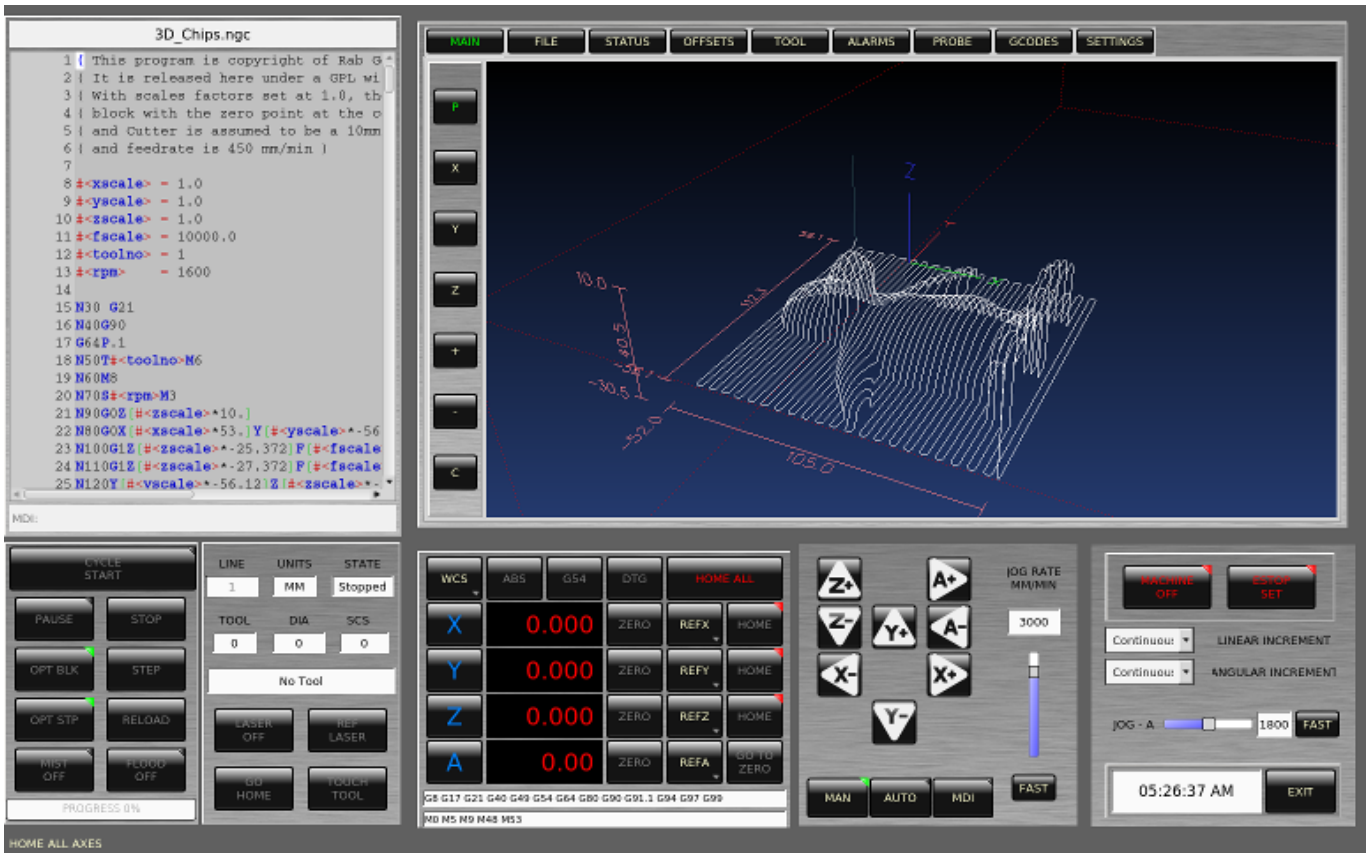



Abbildung 10.36: QtDragon - Angepasster QtDragon

## 10.6 NGCGUI

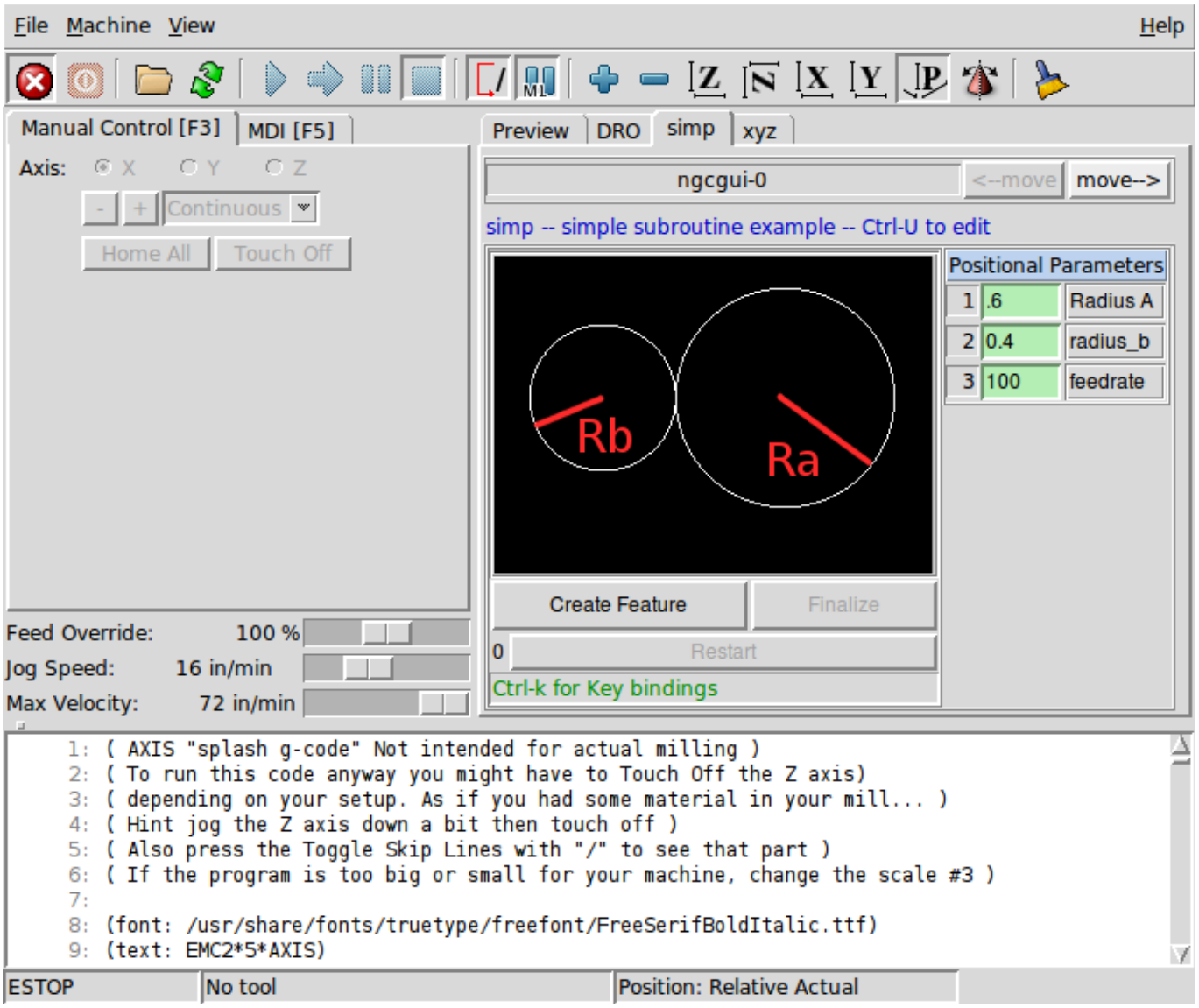



Abbildung 10.37: NGCGUI eingebettet in AXIS

### 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.
- Any gcmc (G-code-meta-compiler) program that conforms to NGCGUI conventions for tagging variables can be used. (The gcmc executable must be installed separately, see: https://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* - Maschinentrainer, der auf eine Kastenform beschränkt ist
- *Custom* - Erzeugt Benutzer-angepasste Registrierkarten (engl. tabs)
- *ttt* - True Type Tracer, um Texte zu erstellen, die graviert werden sollen

- *nc_files/ngcgui_lib/lathe*
  - ngcgui-lathe' - Beispiel-Unterprogramm für Drehmaschinen
  - *g76base.ngc* - GUI für G76 Gewindebohren
  - g76diam.ngc' - Gewinde nach Haupt- und Nebendurchmesser spezifiziert
  - *id.ngc* - bohrt den Innendurchmesser
  - *od.ngc* - dreht den Außendurchmesser
  - taper-od.ngc' - dreht einen Kegel auf dem Außendurchmesser
  - *Custom* - Erzeugt Benutzer-angepasste Registrierkarten (engl. tabs)
- *nc_files/gcmc_lib*
  - *drill.gcmc* - Löcher im Rechteckmuster bohren
  - square.gcmc' - einfache Demo von variablen Tags für gcmc-Dateien
  - *star.gcmc* - GCMC-Demo zur Veranschaulichung von Funktionen und Arrays
  - *wheels.gcmc* - GCMC Demo komplexer Muster

Um eine Demonstration zu versuchen, wählen Sie eine Sim-Konfiguration und starten Sie das LinuxCNC-Programm.

Wenn Sie die AXIS GUI verwenden, drücken Sie auf "Notaus" (engl. E-Stop) und dann auf "Machine Power" und dann auf "Refefernzierfahrt aller Achsen" (engl. Home All). Wählen Sie eine NGCGUI-Registerkarte, füllen Sie alle leeren Felder mit sinnvollen Werten aus und drücken Sie auf "Feature anlegen" (engl. create feature) und dann auf "Finalize". Drücken Sie abschließend auf die Schaltfläche "Ausführen" , um die Ausführung zu beobachten. Experimentieren Sie, indem Sie mehrere Merkmale und Merkmale aus verschiedenen Registerkarten erstellen.

Um mehrere Unterprogramme zu erstellen, die in einer einzigen Datei zusammengefasst sind, gehen Sie zu jeder Registerkarte, füllen Sie die Leerstellen aus und drücken Sie "Feature erstellen". Drücken Sie nun auf "Abschließen" und beantworten Sie die Aufforderung zum Erstellen von

Andere GUIs haben eine ähnliche Funktionalität, aber die Schaltflächen und Namen können unterschiedlich sein.

**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:

```
Item:    [DISPLAY]TKPKG = Ngcgui_ttt version_number
Example: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
Note:    Mandatory, specifies loading of ngcgui_ttt in an AXIS tab page named ttt.
         Must follow the TKPKG = Ngcgui item.

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. The search path begins with the standard directory specified by:

```
[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
```

New users may inadvertently try to use files that are not structured to be compatible with NGCGUI requirements. NGCGUI will likely report numerous errors if the files are not coded per its conventions. Good practice suggests that NGCGUI-compatible subfiles should be placed in a directory dedicated to that,purpose and that preamble, postamble, and helper files should be in separate directory(ies) to discourage attempts to use them as subfiles. Files not intended for use as subfiles can include a special comment: ”(not_a_subfile)” so that NGCGUI will reject them automatically with a relevant message.

#### 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**
*Example*: [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
*Note*: Optional, font_descriptor is a tcl-compatible font specifier with items for fonttype -fontsize fontweight.
Default is: Helvetica -10 normal.
Smaller font sizes may be useful for small screens.
Larger font sizes may be helpful for touch screen applications .

**[ANZEIGE] NGCGUI_SUBFILE = subfile_filename**
*Example*: `[DISPLAY]NGCGUI_SUBFILE = simp.ngc`
*Example*: `[DISPLAY]NGCGUI_SUBFILE = square.gcmc`
*Example*: `[DISPLAY]NGCGUI_SUBFILE = ""`
*Note*: Use one or more items to specify NGCGUI-compatible subfiles or gcmc programs that require a tab page on startup.
A "Custom" tab will be created when the filename is "".
A user can use a "Custom" tab to browse the file system and identify preamble, subfile, and postamble files.

**[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) an image frame (iframe) is made available to display an image for the subfile (if an image is provided);
5) the NGCGUI result file sent to LinuxCNC is terminated with an M2 (and incurs M2 side-effects).

Die Optionen `nonew`, `noremove`, `noauto`, `noiframe`, `nom2` schalten diese Standardverhaltensweisen jeweils aus.

By default, if an image (.png,.gif,jpg,pgm) file is found in the same directory as the subfile, the image is displayed in the iframe. Specifying the `noiframe` option makes available additional buttons for selecting a preamble, subfile, and postamble and additional checkboxes. Selections of the checkboxes are always available with special keys:
`Ctrl-R` Toggle ”Retain values on Subfile read”,
`Ctrl-E` Toggle ”Expand subroutine”,
`Ctrl-a` Toggle ”Autosend”,
`Ctrl-k` lists all keys and functions.

If `noiframe` is specified and an image file is found, the image is displayed in a separate window and all functions are available on the tab page. The `NGCGUI_OPTIONS` apply to all NGCGUI tabs except that the `nonew`, `noremove`, and `noiframe` options are not applicable for ”Custom” tabs. Do not use ”Custom” tabs if you want to limit the user’s ability to select subfiles or create additional tab pages.

**[DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...**
*Example*: `[DISPLAY]GCMC_INCLUDE_PATH = /home/myname/gcmc_includes:/home/myname/gcmc_incl`
*Note*: Optional, each directory will be included when gcmc is invoked using the option: `--include dirname`.

## 10.6.6 Dateianforderungen für NGCGUI-Kompatibilität

### 10.6.6.1 Anforderungen an eine G-code-Unterroutine (.ngc) in einer Datei

An NGCGUI-compatible subfile contains a single subroutine definition. The name of the subroutine must be the same as the filename (not including the .ngc suffix). LinuxCNC supports named or numbered subroutines, but only named subroutines are compatible with NGCGUI. For more information see the O-Codes chapter.

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)
```

The body of the subroutine should begin with a set of statements that define local named parameters for each positional parameter expected for the subroutine call. These definitions must be consecutive beginning with #1 and ending with the last used parameter number. Definitions must be provided for each of these parameters (no omissions).

**Nummerierung der Parameter**

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC considers all numbered parameters in the range #1 thru #30 to be calling parameters, so NGCGUI provides entry boxes for any occurrence of parameters in this range. It is good practice to avoid use of numbered parameters #1 through #30 anywhere else in the subroutine. Using local, named parameters is recommended for all internal variables.

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)
```

If a default_value is provided, it will be entered in the entry box for the parameter on startup. If comment_text is included, it will be used to identify the input instead of the parameter name.

**Globale benannte Parameter** Hinweise zu globalen benannten Parametern und NGCGUI:

(globale benannte Parameter haben einen führenden Unterstrich im Namen, wie #<_irgendeinglobalername>)

As in many programming languages, use of globals is powerful but can often lead to unexpected consequences. In LinuxCNC, existing global named parameters will be valid at subroutine execution and subroutines can modify or create global named parameters.

Passing information to subroutines using global named parameters is discouraged since such usage requires the establishment and maintenance of a well-defined global context that is difficult to maintain. Using numbered parameters #1 thru #30 as subroutine inputs should be sufficient to satisfy a wide range of design requirements.  NGCGUI unterstützt einige globale benannte Eingabeparameter, aber deren Verwendung ist veraltet und hier nicht dokumentiert.

While input global named parameters are discouraged, LinuxCNC subroutines must use global named parameters for returning results. Since NGCGUI-compatible subfiles are aimed at GUI usage, return values are not a common requirement. However, NGCGUI is useful as a testing tool for subroutines which do return global named parameters and it is common for NGCGUI-compatible subfiles to call utility subroutine files that return results with global named parameters.

To support these usages, NGCGUI ignores global named parameters that include a colon (:) character in their name. Use of the colon (:) in the name prevents NGCGUI from making entryboxes for these parameters.

**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
```

In the above example, the utility subroutine will be found in a separate file named helper.ngc. The helper routine returns a result in a global named parameter named #<_helper:answer.

For good practice, the calling subfile immediately localizes the result for use elsewhere in the subfile and the global named parameter used for returning the result is nullified in an attempt to mitigate its inadvertent use elsewhere in the global context. A nullification value of 0.0 may not always be a good choice.

NGCGUI supports the creation and concatenation of multiple features for a subfile and for multiple subfiles. It is sometimes useful for subfiles to determine their order at runtime, so NGCGUI inserts a special global parameter that can be tested within subroutines. The parameter is named #<_feature:>. Its value begins with a value of 0 and is incremented for each added feature.

**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)
```

An optional image file (.png,.gif,.jpg,.pgm) can accompany a subfile. The image file can help clarify the parameters used by the subfile. The image file should be in the same directory as the subfile and have the same name with an appropriate image suffix, e.g. the subfile example.ngc could be accompanied by an image file examp.png. NGCGUI attempts to resize large images by subsampling to a size with maximum width of 320 and maximum height of 240 pixels.

Keine der Konventionen, die für die Herstellung einer NGCGUI-kompatiblen Subdatei erforderlich sind, schließen ihre Verwendung als allgemeine Subroutinendatei für LinuxCNC aus.

The LinuxCNC distribution includes a library (ngcgui_lib directory) that includes both example NGCGUI-compatible subfiles and utility files to illustrate the features of LinuxCNC subroutines and NGCGUI usage. Another library (gcmc_lib) provides examples for subroutine files for the 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.

To make it easier to modify valid lines in a gcmc file, alternate tag line formats accepted. The alternate formats ignore trailing semicolons (;) and trailing comment markers (//). With this provision, it is often makes it possible to just add the //ngcgui: tag to existing lines in a .gcmc file.

**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:

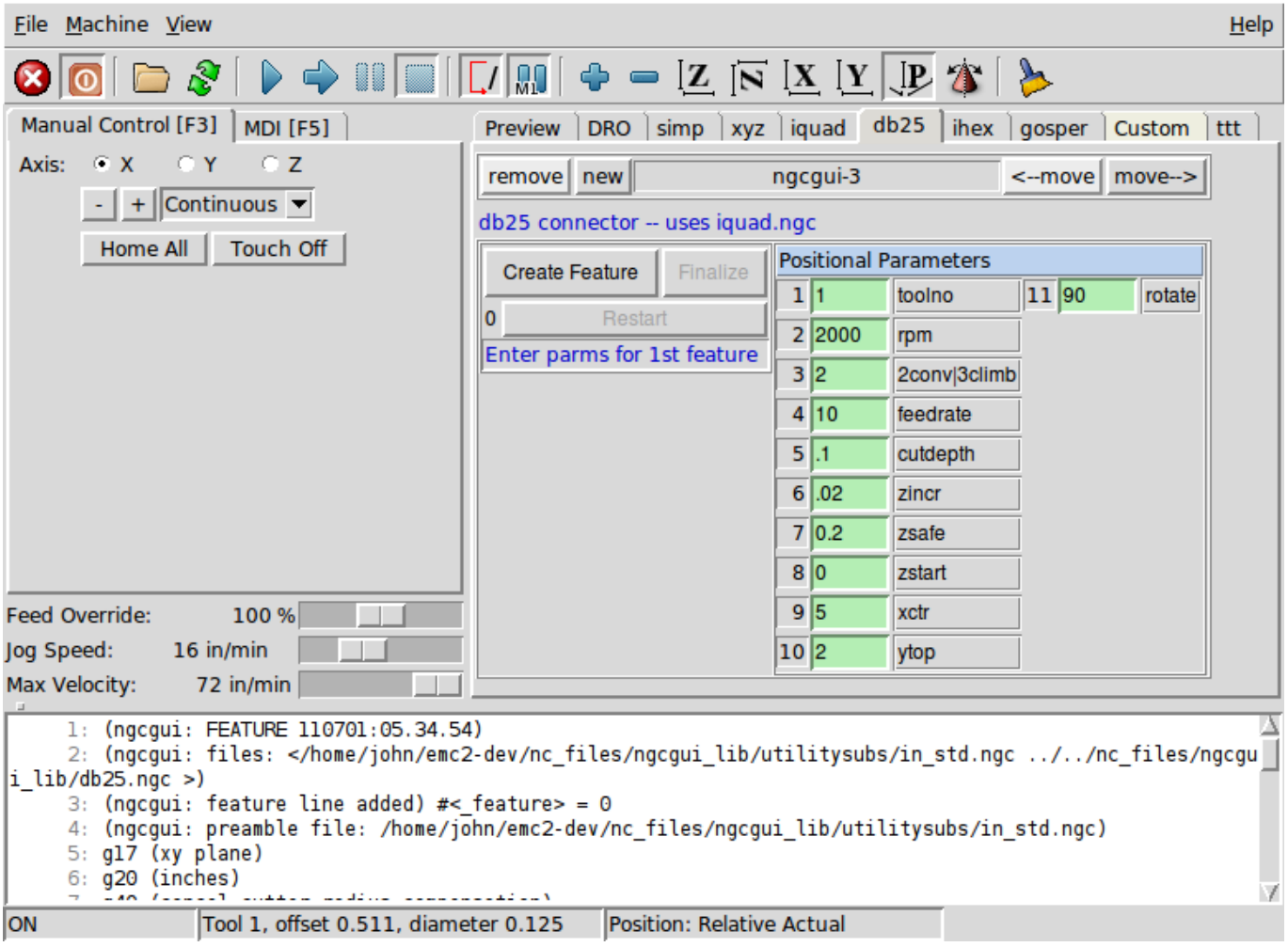
```
include(”ensure_mode.gcmc”)
```

und geben Sie den richtigen Pfad für gcmc include_files in der INI-Datei an, zum Beispiel:

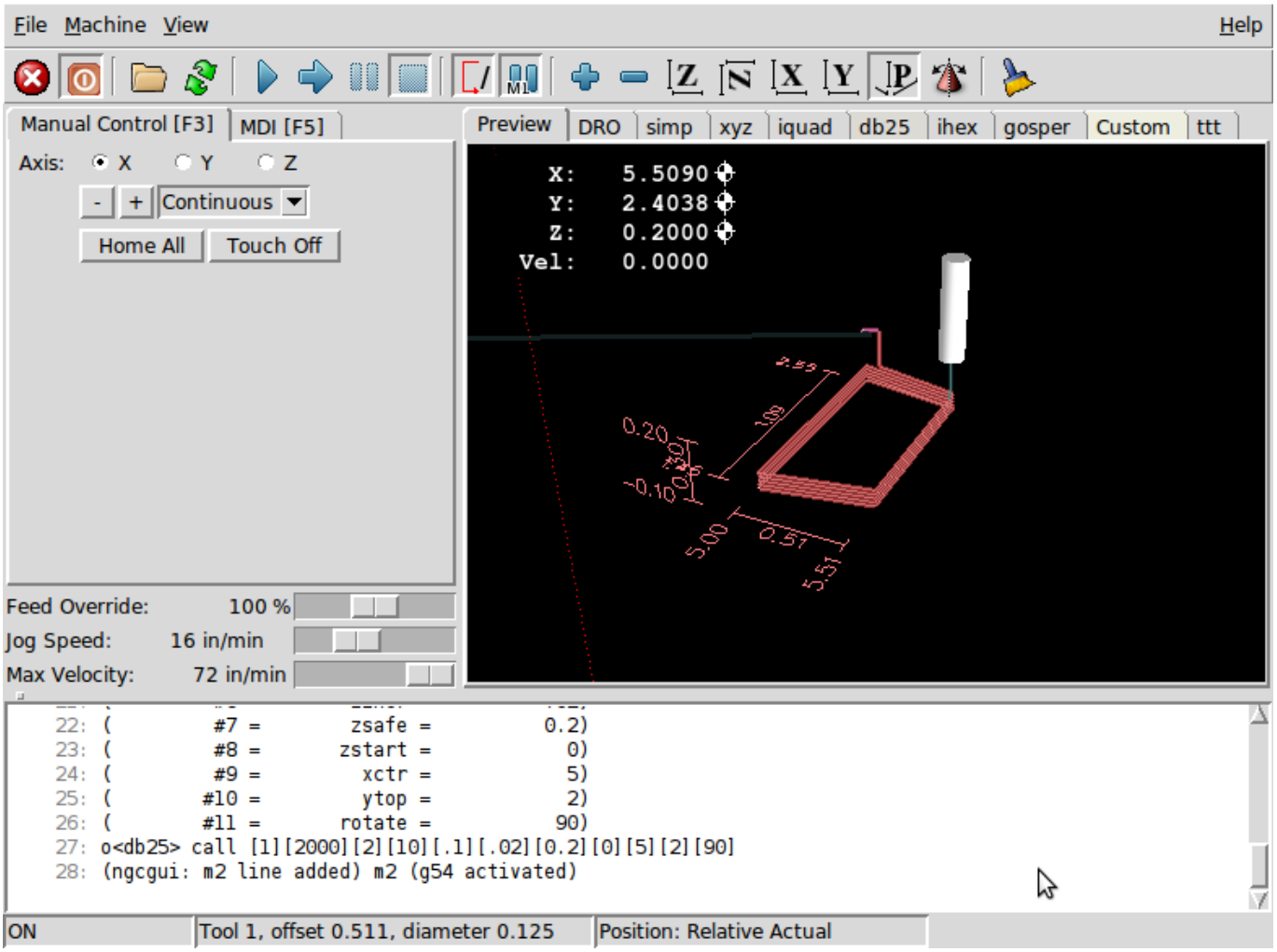
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

### 10.6.7 DB25 Beispiel

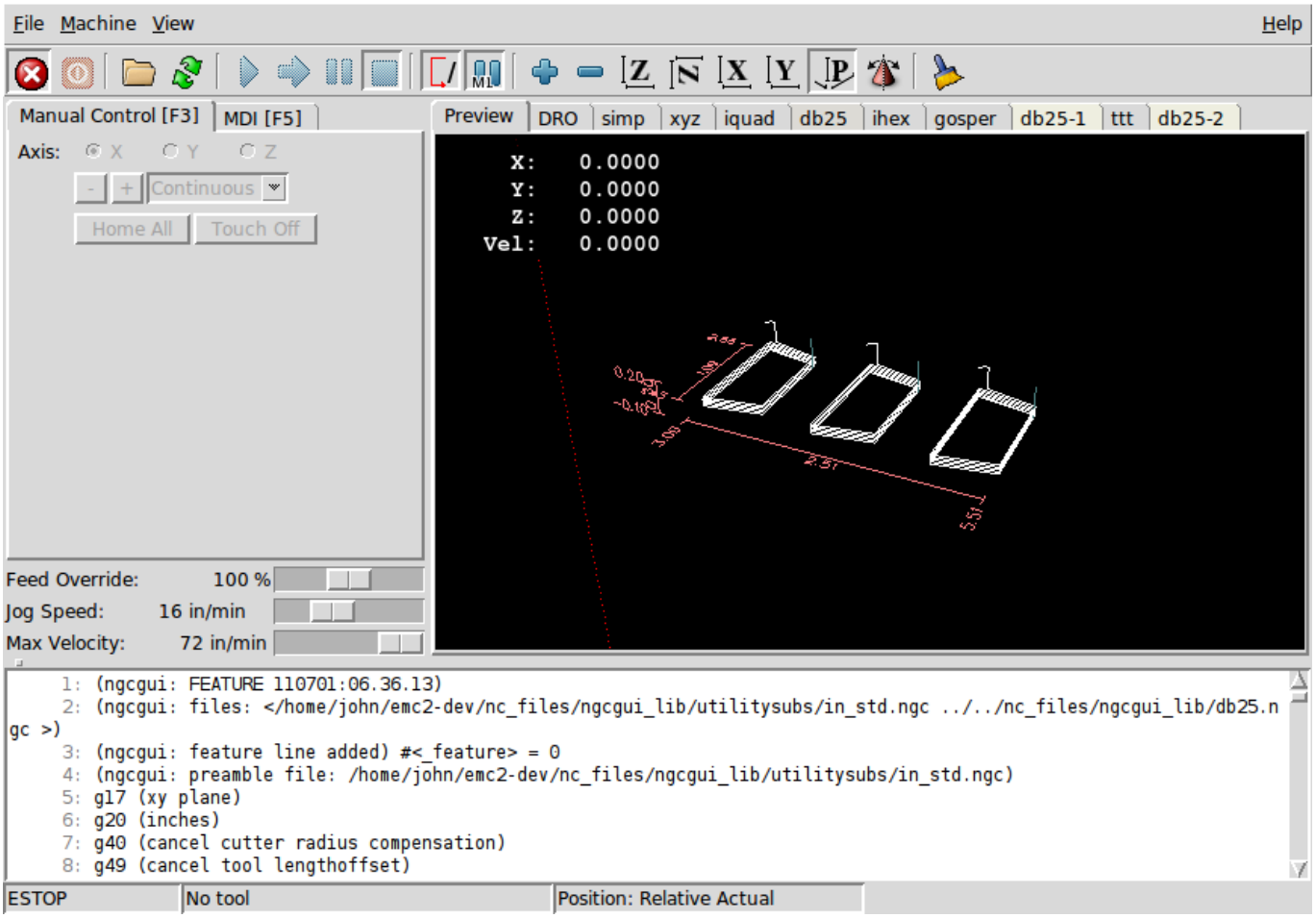
The following shows the DB25 subroutine. In the first photo you see where you fill in the blanks for each variable.




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.

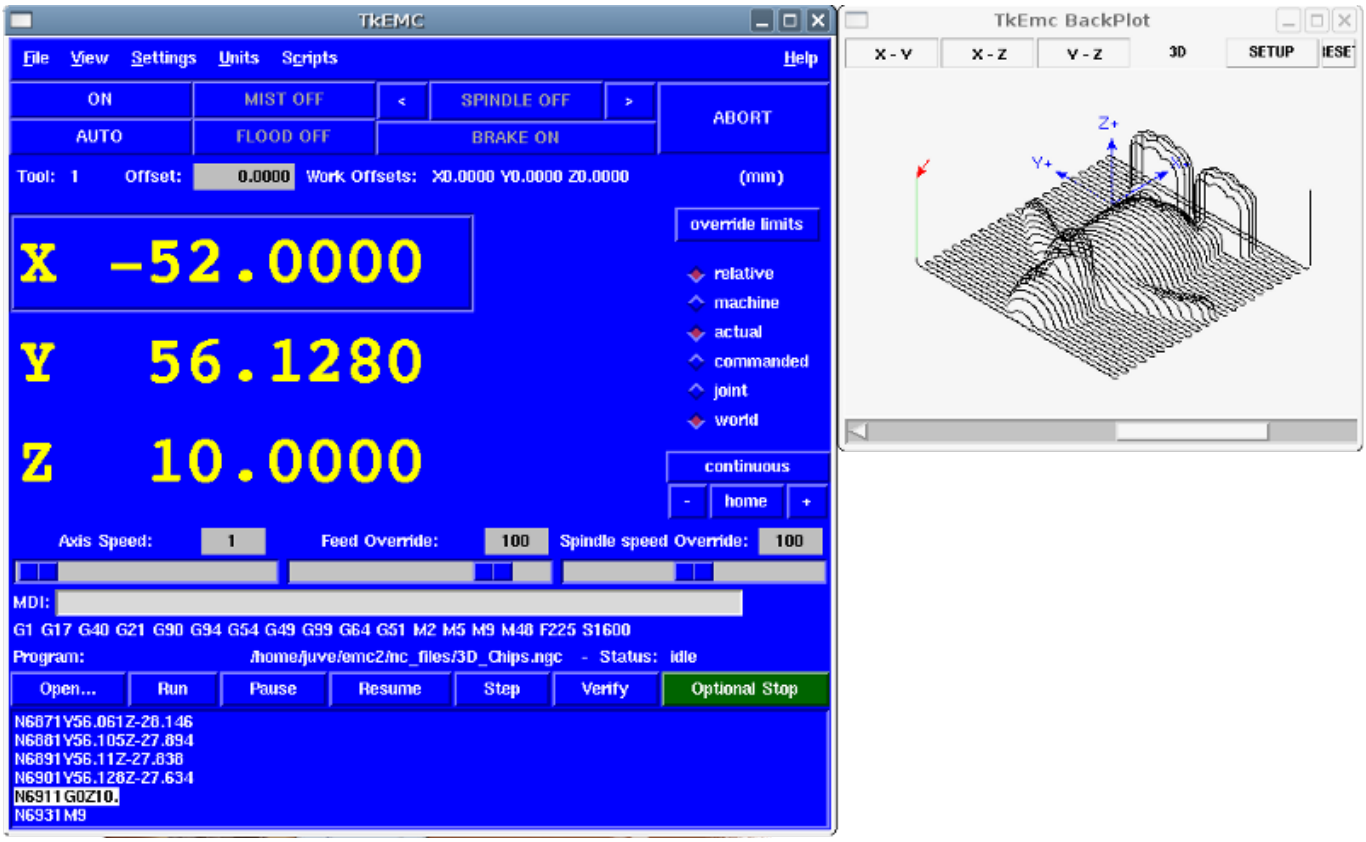



Abbildung 10.38: TkLinuxCNC-Fenster

### 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
- Abbrechen

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

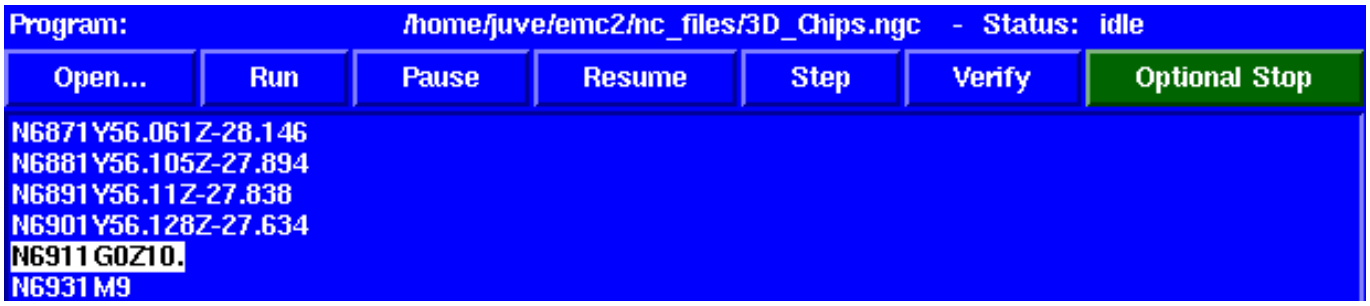
Der Hauptteil der Anzeige zeigt die aktuelle Position des Werkzeugs an. Die Farbe der Positionsanzeige hängt vom Zustand der Achse ab. Wenn die Achse nicht referenziert ist, wird die Achse in gelber Schrift angezeigt. Sobald sie referenziert ist, wird sie in grüner Schrift angezeigt. Wenn es einen Fehler 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 / Automatic Program Control




**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).

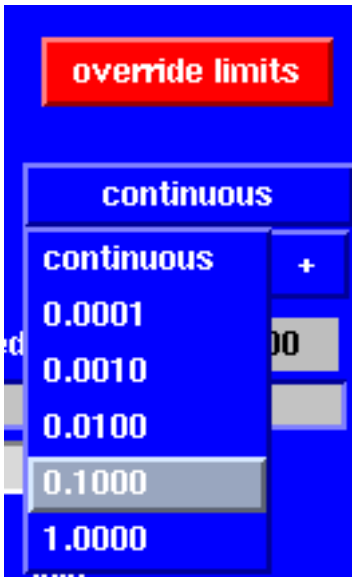


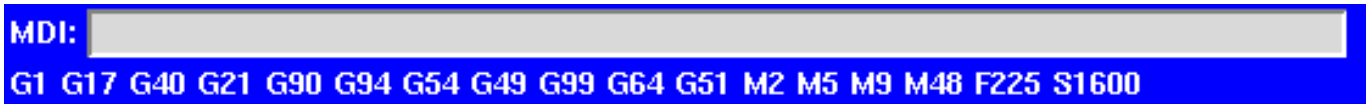

Abbildung 10.39: Beispiel für TkLinuxCNC Override Limits & Jogging Increments

**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 v2.9 or later 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:

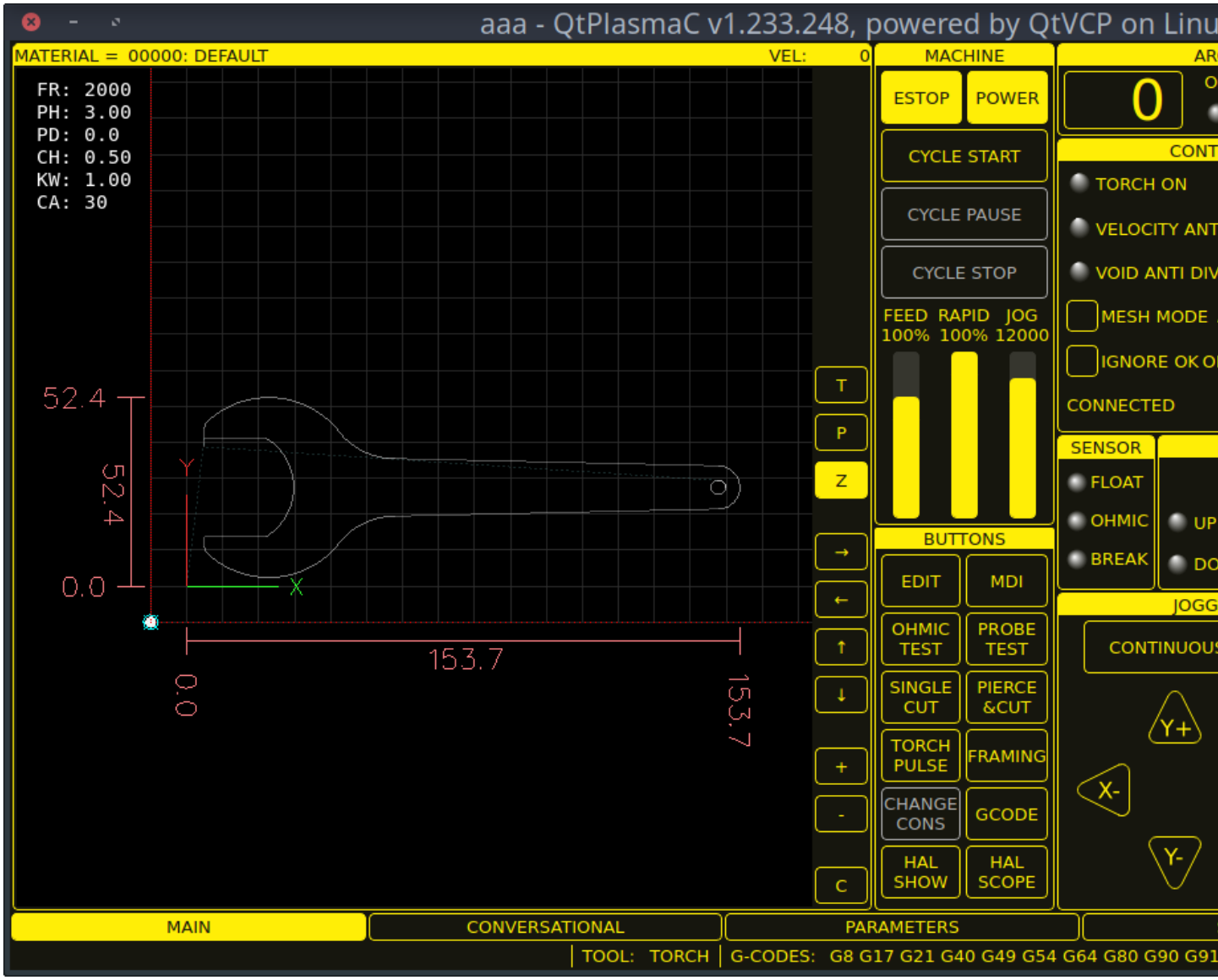
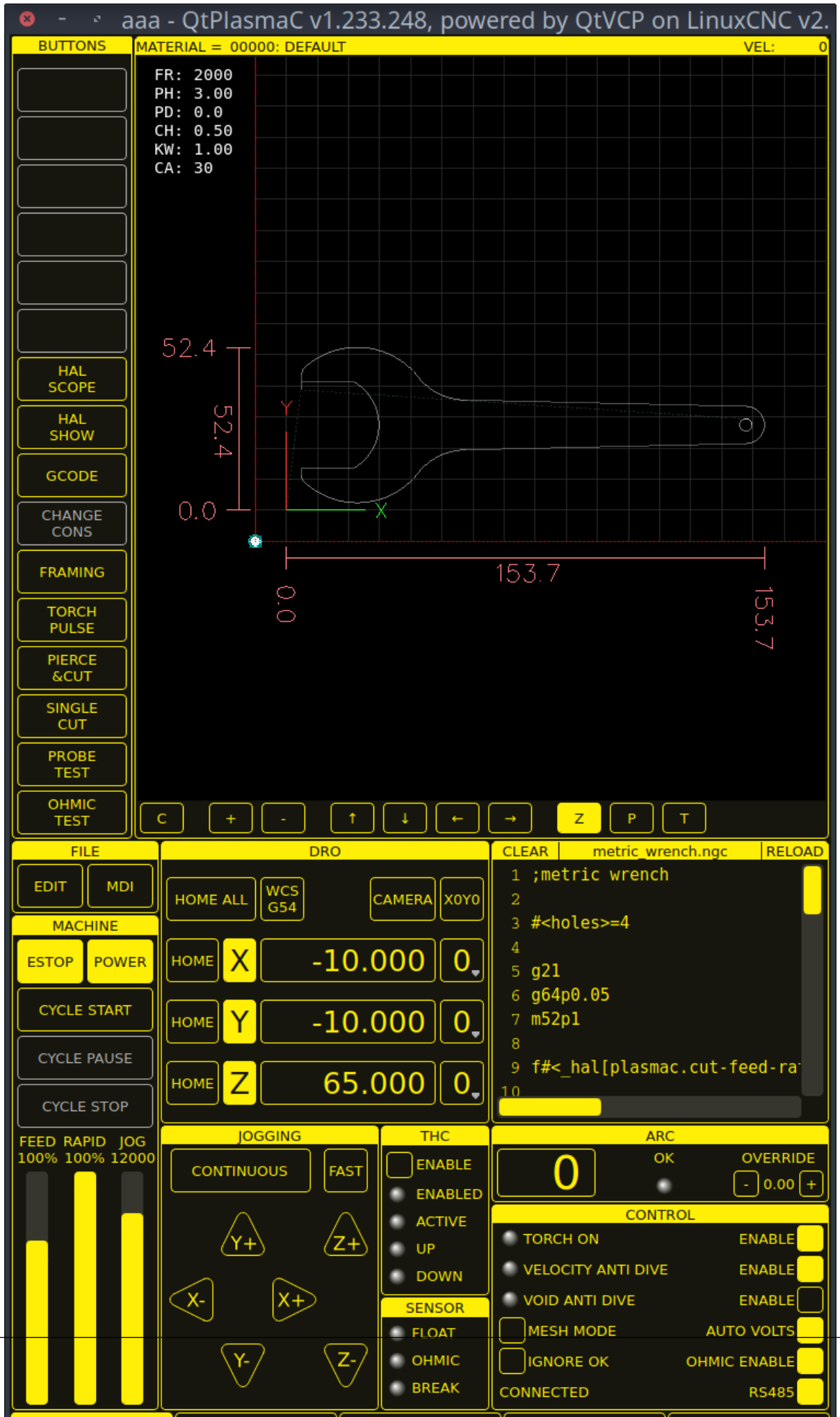



Abbildung 10.40: **16:9**


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
FEED 100%
RAPID 100%
JOG 12000
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
3 #<holes>=4
5 g21
6 g64p0.05
7 m52p1
9 f#<_hal[plasmac.cut-feed-ra
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

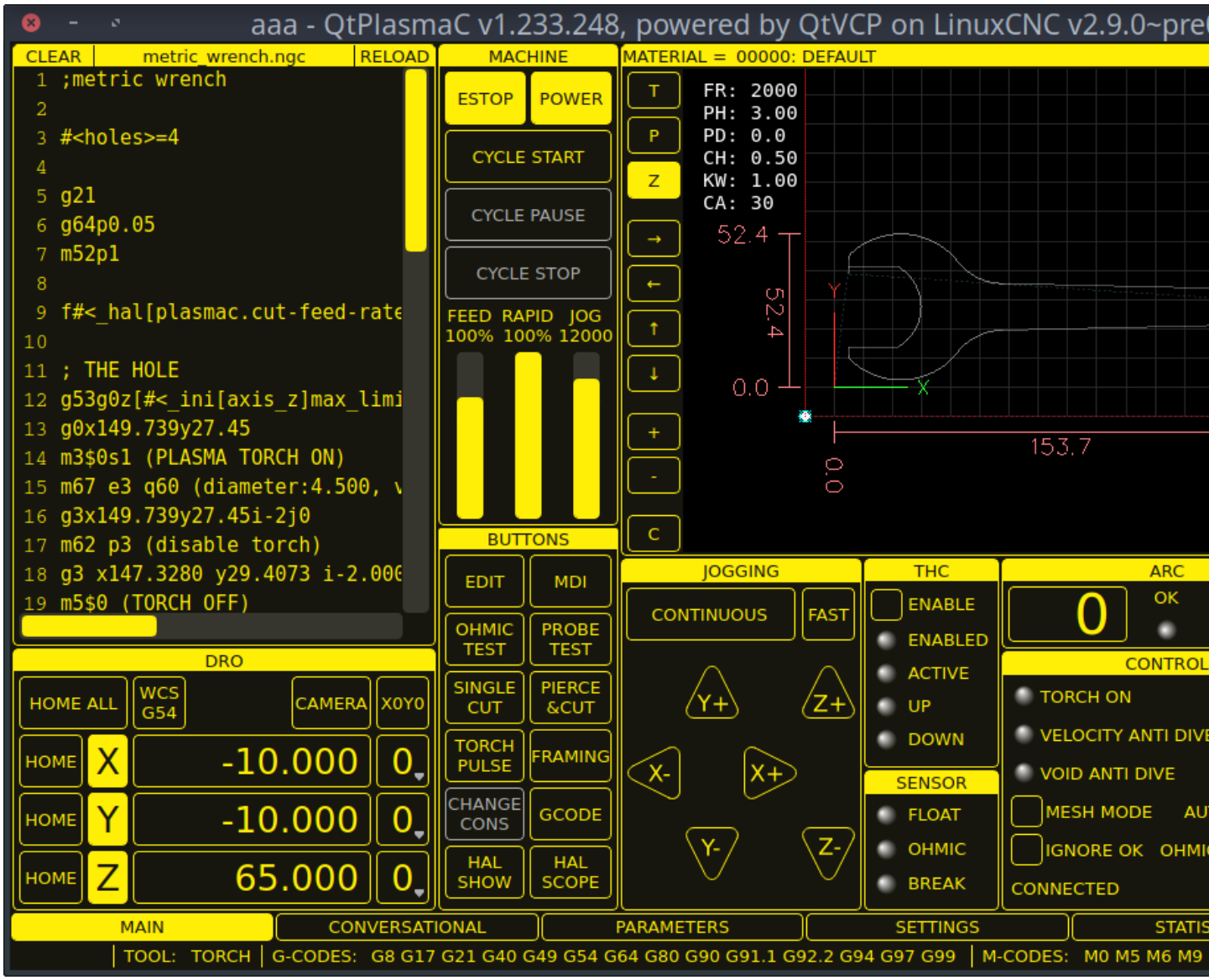



Abbildung 10.42: **4:3**

### 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

Installation instructions are available from here.

Following these instructions will yield a machine with the current stable branch of LinuxCNC (v2.9) on Debian 12 (Bookworm).

#### 10.8.4.2 Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm)

Follow the instructions from the Updating LinuxCNC on Debian Bookworm section from here.

#### 10.8.4.3 Package Installation (Buildbot) If The User Has Linux on Debian 11 (Bullseye) or Debian 10 (Buster)

A package installation (Buildbot) uses prebuilt packages from the LinuxCNC Buildbot.

Add the GPG keys and add the repository to the sources list to suit the Debian version.

The below stanza would add the 2.9 Bullseye repository.

```
deb     http://buildbot2.highlab.com/ bookworm 2.9-uspace
```

#### 10.8.4.4 Run In Place Installation If The User Has Linux Installed

A run in place installation runs LinuxCNC from a locally compiled version usually located at ~/linuxcnc-dev, instructions for building a run in place installation are available from here.

### 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:

| Modus | Beschreibung |
|---|---|
| 0 | Verwendet einen externen Lichtbogenspannungseingang, um sowohl die Lichtbogenspannung (für die Brennerhöhensteuerung) als auch den Lichtbogen-OK zu berechnen. |
| 1 | Uses an external arc voltage input to calculate Arc Voltage (for Torch Height Control).<br>Uses an external Arc OK input for Arc OK. |
| 2 | Uses an external Arc OK input for Arc OK.<br>Use external up/down signals for Torch Height Control. |

**Wichtig**
If the plasma power source has an Arc OK (Transfer) output then it is recommended to use that for Arc OK rather than the soft (calculated) Arc OK provided by mode 0. It may also be possible to use a reed relay as an alternative method to establish an Arc OK signal when the power source does not provide one.

**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.**<br>HAL pin name `plasmac.arc-voltage-in`<br>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.**<br>HAL pin name `plasmac.arc-ok-in`<br>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:**<br>HAL pin name `plasmac.float-switch`<br>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.<br>If used and no ohmic probe is configured, this is the probing method.<br>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:**<br>HAL pin name `plasmac.ohmic-probe`<br>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.<br>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. |
| Ohmic Probe Enable | 0, 1, 2 | Digital output; **optional, see info below table:**<br>HAL pin name `plasmac.ohmic-enable`<br>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:**<br>HAL pin name `plasmac.breakaway`<br>Connected from a breakout board input to a torch breakaway detection switch.<br>This signal senses if the torch has broken away from its cradle. |

| Name | Modi | Beschreibung |
|---|---|---|
| Brenner ein (engl. torch on) | 0, 1, 2 | Digital output; **required.**<br>HAL pin name `plasmac.torch-on`<br>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.**<br>HAL pin name `plasmac.move-up`<br>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.**<br>HAL pin name `plasmac.move-down`<br>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.**<br>HAL pin name `plasmac.scribe-arm`<br>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.**<br>HAL pin name `plasmac.scribe-on`<br>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.**<br>HAL pin name `qtplasmac.laser_on`<br>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 4 mm (0.157") to activate then set the Z minimum to 5 mm (0.2") plus an allowance for overrun (either calculated using the equation below or allow 5 mm (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 5 mm-10 mm (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/s$^2$ and v = velocity in units/s.

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 by running the `stepconf` command in a terminal window or launching it using the **Application -> CNC -> StepConf Wizard** desktop menu entry.

If using a Mesa Electronics board, use the PnCconf wizard by running the `pncconf` command in a terminal window or launching it using the **Application -> CNC -> PnCConf Wizard** desktop menu entry.

If using a Pico Systems board, this LinuxCNC forum thread may be helpful.

Die gerätespezifischen Einstellungen werden hier nicht beschrieben, sondern sind in der Dokumentation des jeweiligen Konfigurationsassistenten nachzulesen, der verwendet wird.

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 adds two pages to the LinuxCNC configuration wizards for QtPlasmaC specific parameters, the two pages are QtPlasmaC options and User Buttons. Complete each of the wizards QtPlasmaC page to suit the machine that is being configured and the user button requirements.

Note that PnCconf options allow user selection of Feed Override, Linear Velocity, and Jog Increments, whereas in StepConf these are automatically calculated and set.

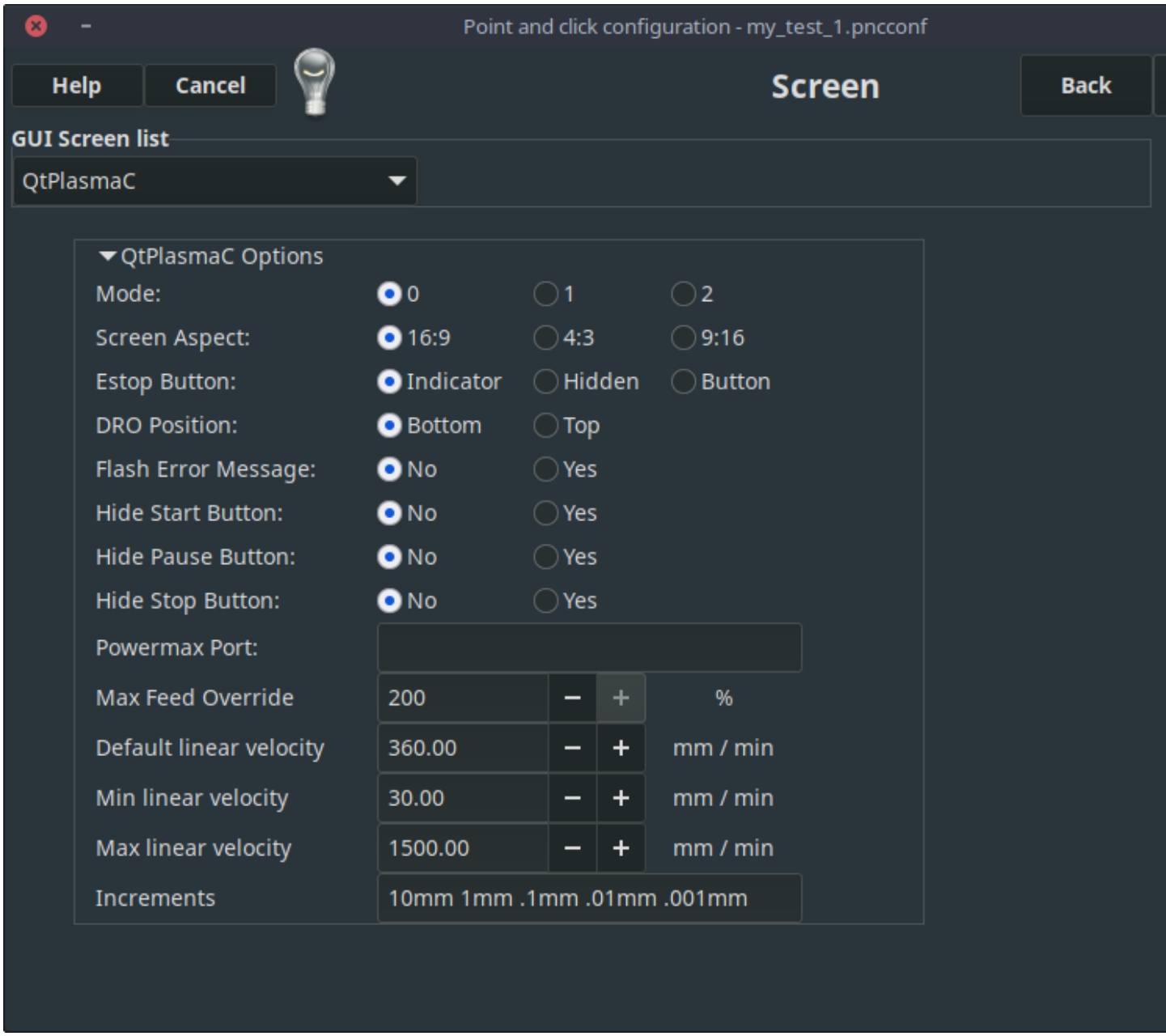



Abbildung 10.43: PnCConf QtPlasmaC Options

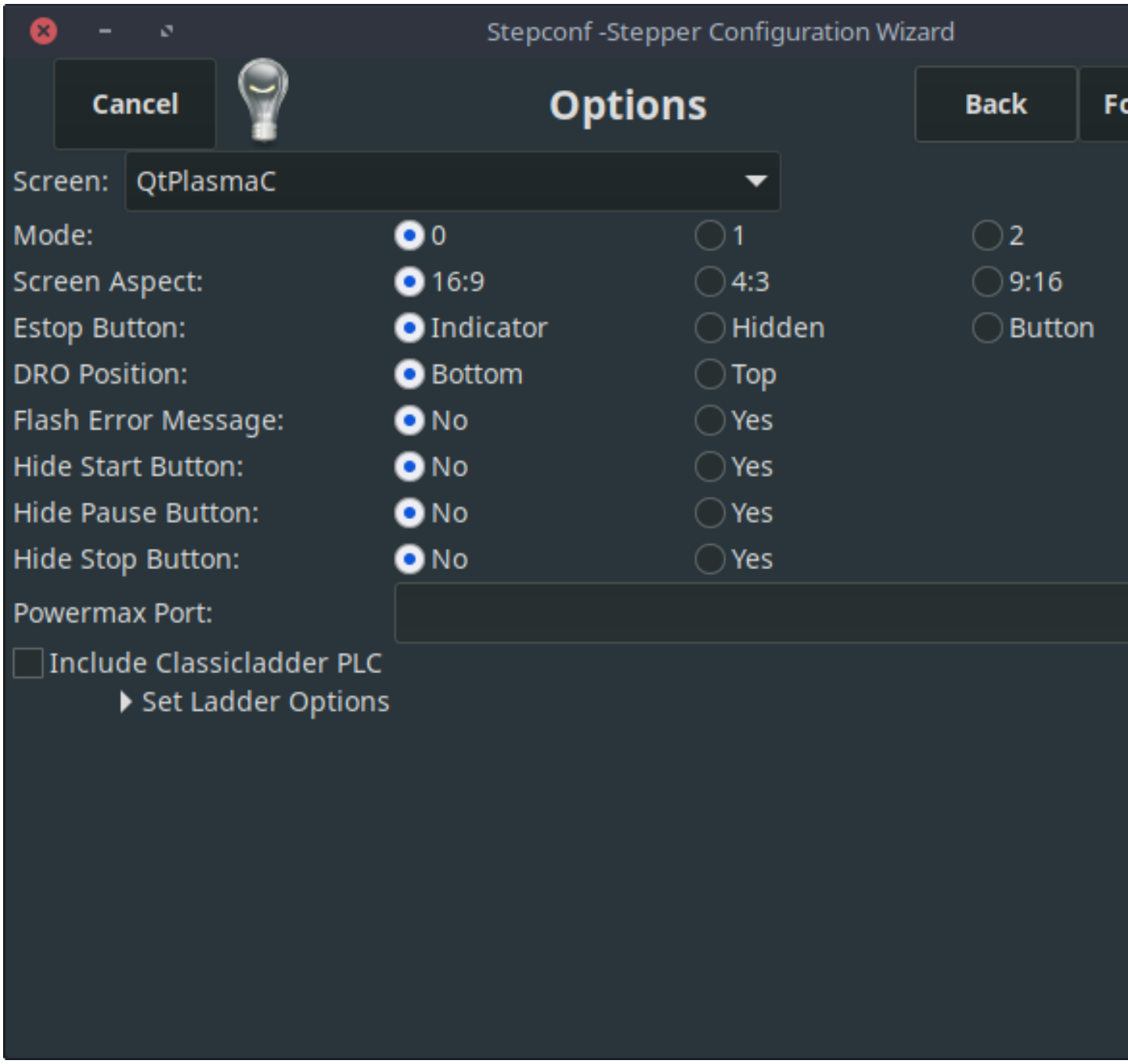



Abbildung 10.44: StepConf QtPlasmaC Options

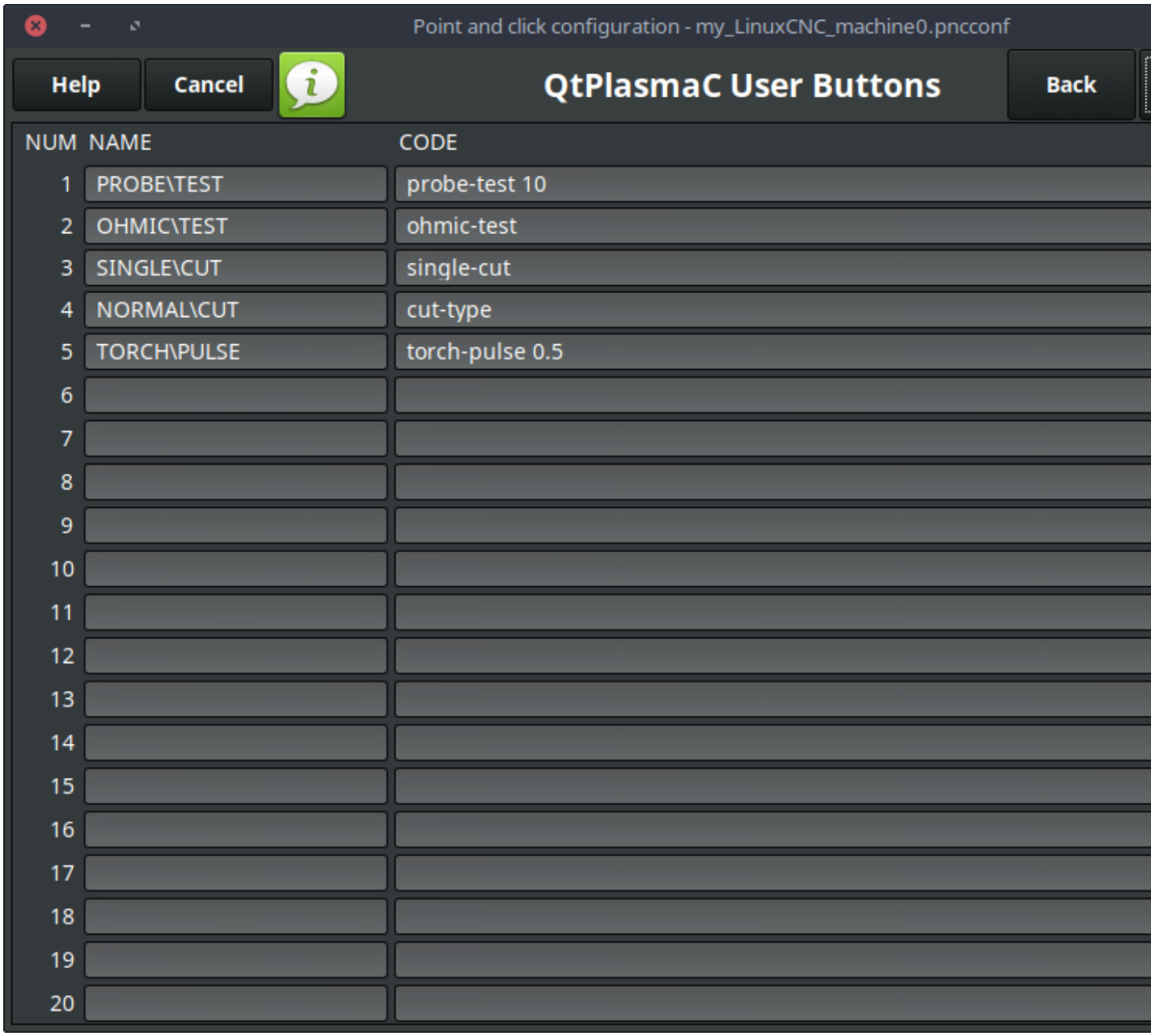



Abbildung 10.45: QtPlasmaC User Buttons

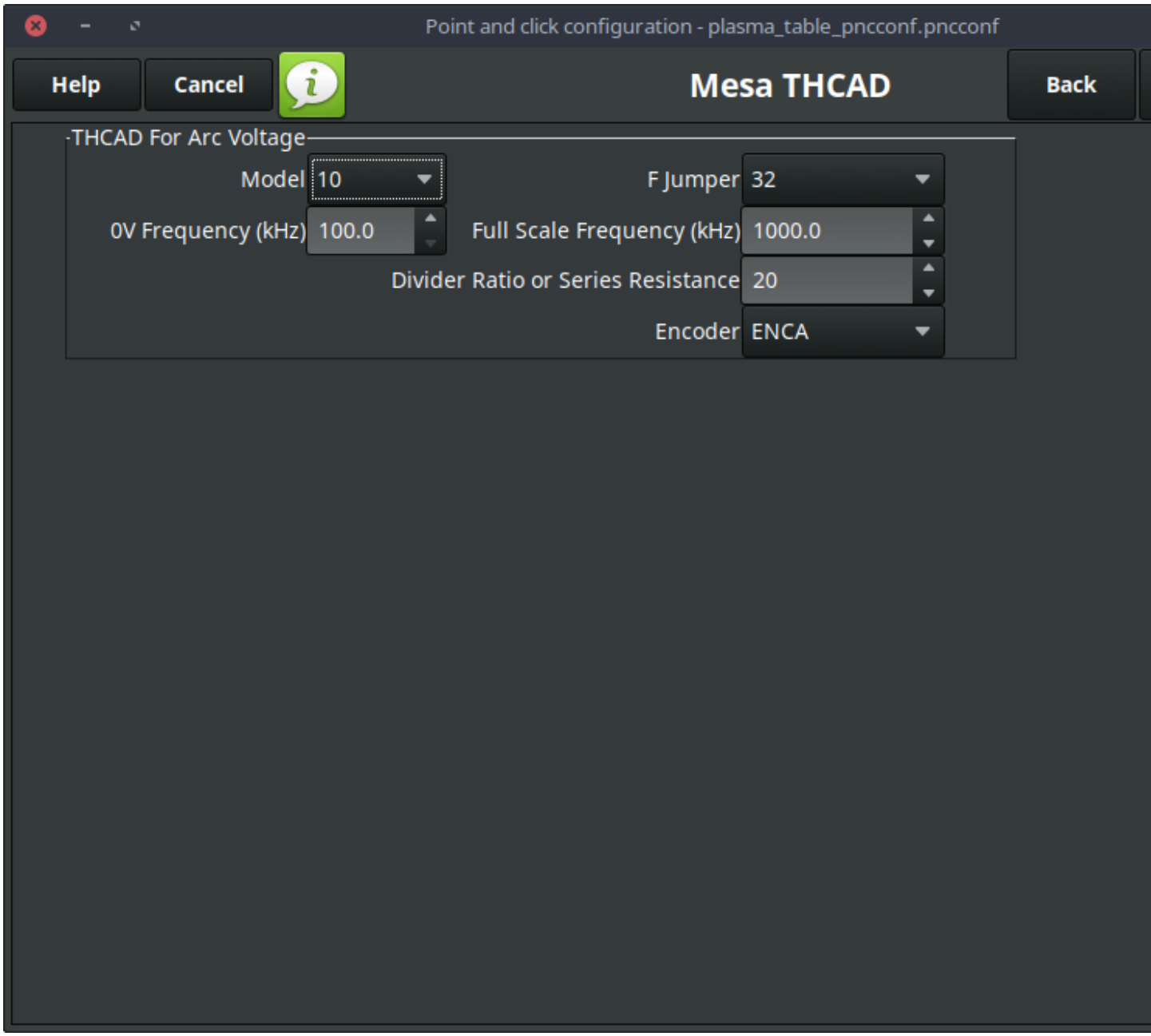



Abbildung 10.46: QtPlasmaC THCAD

The THCAD screen will only appear if a Plasma Encoder is selected in the card screen. The the dedicated section on Mesa THCAD for more information.

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*>.

The way the newly created QtPlasmaC configuration can be run from the terminal command line slightly differs depending the way LinuxCNC was installed:

Für eine Paketinstallation (Buildbot):

```
linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

For a run in place installation:

```
~/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:

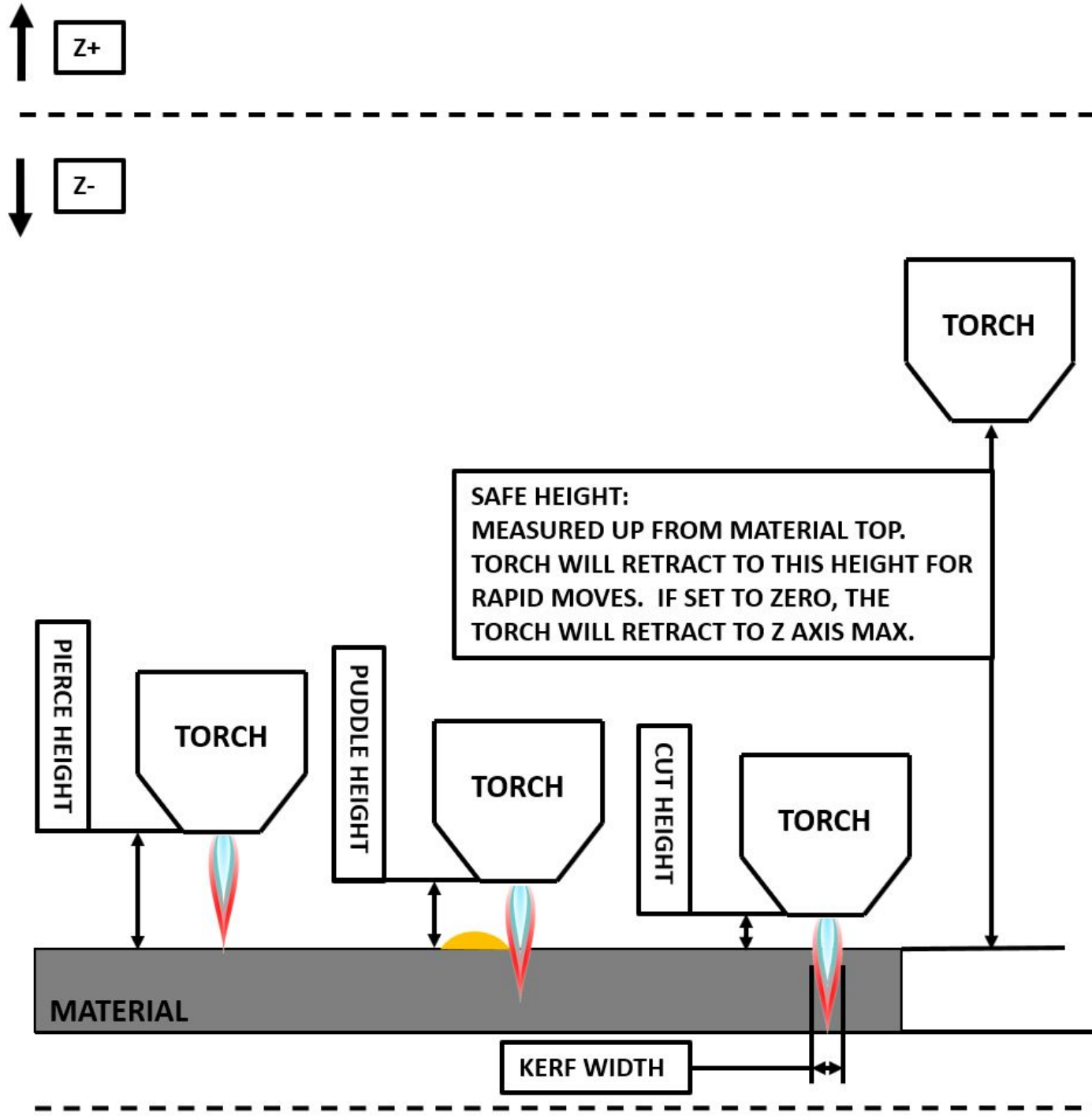
```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

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:




Click on the Parameters Tab to view the **CONFIGURATION** section which shows the user settable parameters. It is necessary to ensure every one of these settings is tailored to the machine.

To set the Z axis DRO relative to the Z axis MINIMUM_LIMIT, the user should perform the following steps. It is important to understand that in QtPlasmaC, touching off the Z axis DRO has no effect on the Z axis position while running a G-code program. These steps simply allow the user to more easily

set the probe height as after performing the steps, the displayed Z axis DRO value will be relative to Z axis MINIMUM_LIMIT.

---

**Anmerkung**
The user should be familiar with the recommended Z Axis Settings.

---

1. Referenzierung der Z-Achse.
2. Vergewissern Sie sich, dass sich nichts unter dem Brenner befindet, dann bewegen Sie die Z-Achse nach unten, bis sie am MINIMUM_LIMIT der Z-Achse anhält, und klicken Sie dann auf die 0 neben der Z-Achsen-Anzeige, um die Z-Achse mit der ausgewählten Z-Achse auf Nullpunktverschiebung zu setzen. Dieser Schritt dient nur dazu, dem Benutzer eine einfachere Visualisierung und Einstellung der **Sondenhöhe** zu ermöglichen - dieser Wert wird vom MINIMUM_LIMIT der Z-Achse aufwärts gemessen.
3. Erneute Referenzierfahrt der Z-Achse.

**Sonden-Test**

Wenn das Gerät mit einem Schwimmerschalter ausgestattet ist, muss der Benutzer den Offset im Abschnitt **KONFIGURATION** auf der Registerkarte **PARAMETER** einstellen. Dies geschieht durch Ausführen eines ”Probe Test”-Zyklus.

1. Überprüfen Sie, ob die Sondengeschwindigkeit und die Sondenhöhe im Abschnitt **CONFIGURATION** auf der Registerkarte **PARAMETERS** korrekt sind. QtPlasmaC kann mit der vollen Geschwindigkeit der Z-Achse tasten, solange die Maschine genügend Bewegung im Schwimmerschalter hat, um einen eventuellen Überlauf aufzufangen. Wenn die Maschine dafür geeignet ist, kann der Benutzer die Sondenhöhe auf einen Wert in der Nähe des Z-Achsen-Minimums einstellen und die gesamte Abtastung mit voller Geschwindigkeit durchführen.
2. Wenn die Maschine noch keine Referenzierfahrt durchführt und nicht in der Ausgangsposition ist, dann führen Sie die Referenzierfahrt durch.
3. Legen Sie etwas Material auf die Latten unter den Brenner.
4. Drücken Sie die Taste **PROBE TEST**.
5. 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**.

**Warnung**
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:

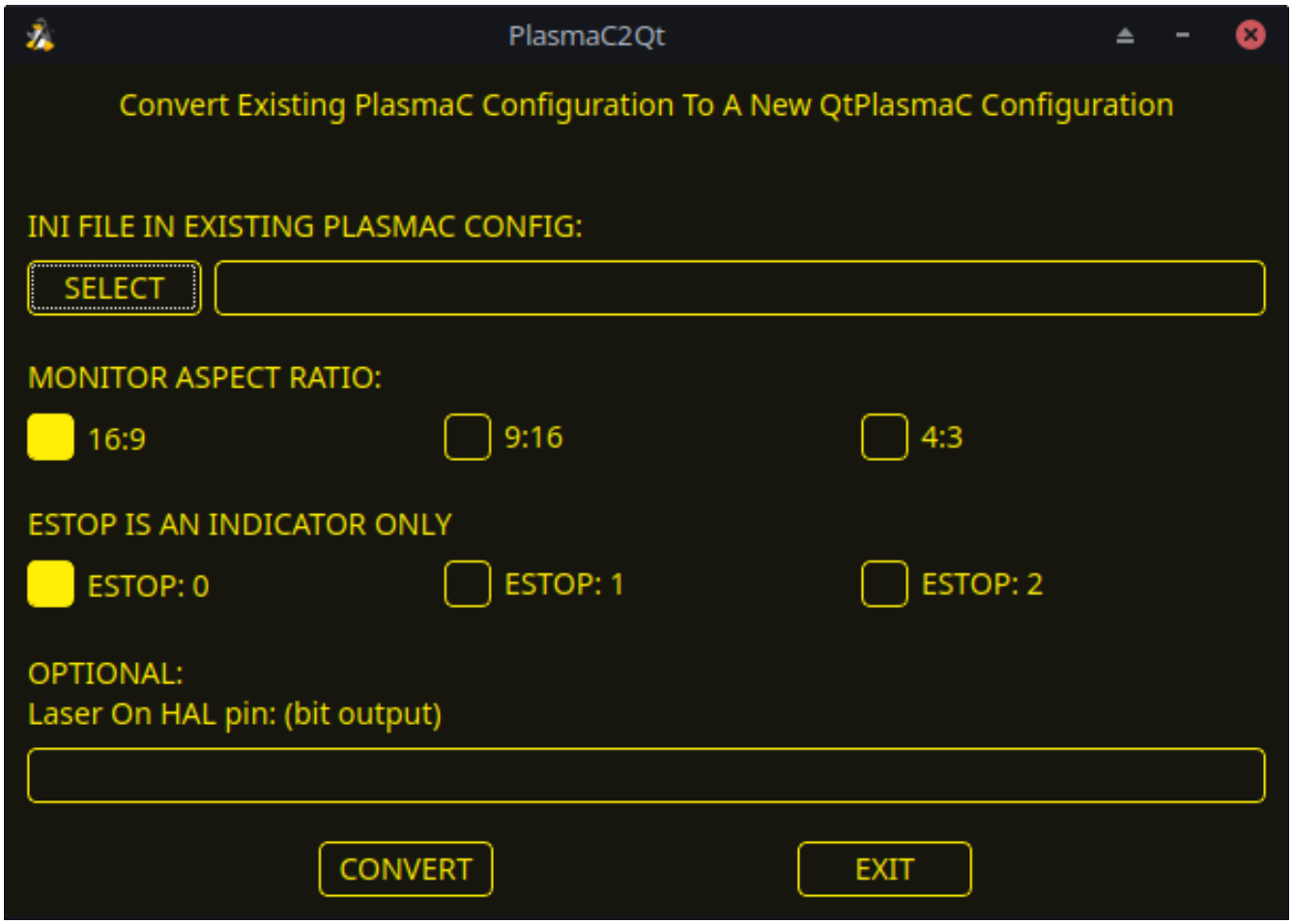



Tabelle 10.6: **Mandatory Settings**

| **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 | This is the aspect ratio format for the GUI. | 16:9 |
| ESTOP (engl. für Notaus) | Selects the required E-stop type based on the following criteria:<br>0 - Estop is an indicator only.<br>1 - Estop indicator is hidden.<br>2 - Estop is a button. | ESTOP:1 |

**Optional Setting - This setting is not required unless the machine has a laser for sheet alignment. Leave this blank if it is not used/required.** 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 Example:** parport.0.pin-16-out **Mesa 7i96 Example:** hm2_7i96.0.ssr.00.out-00 |

Nachdem Sie die entsprechenden Eingaben gemacht haben, drücken Sie **CONVERT**.

**Anmerkung**
This method will not change any existing debounce components to the new dbounce component. If the user wishes to change to the new dbounce component then the New Base Config method should be used for migration.

#### 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.

After using the wizard, the user can then run a conversion program (cfg2prefs) to convert the parameters, settings, and materials from the previous PlasmaC configuration to the new QtPlasmaC configuration. This tool should be used immediately after the user has created a new QtPlasmaC configuration.

Prior to running this conversion program, it is mandatory that the user have both an existing PlasmaC configuration and a new QtPlasmaC configuration. This program **will overwrite** the existing QtPlasmaC preferences and materials files, and should be used with caution if it is not being run on a new QtPlasmaC configuration.

The program will create a time-stamped backup of the original preferences file and the existing materials file (if it exists).

It will read the existing <*machine_name*>*config.cfg*, <machine_name>*_run.cfg*, <machine_name>*_wizards 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
```

Abbildung 10.47: `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 5000000 ns, or 5 ms.

For the Float and Ohmic switches this equates to a 0.001 mm (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 0.1 µF film capacitor placed across the contacts.
2. A 1200 Ω 1 W 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 VI of the finder General Technical Information document.

**Calculations:**

If using a Mesa card, the input resistance of a 7I96 is 4700 Ω (symbol R)(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 (symbol I) assuming a supply voltage (symbol U) of 24 V (I = U/R)[1].

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

If a link to the launch the configuration was not created when creating the config, the user could create a desktop launcher to the config by right clicking on the desktop and selecting Create Launcher or similar. This will bring up a dialog box to create a launcher. Give the icon a nice short name, enter anything for the command and click 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
```

[1] In the US, the letter V is commonly used as a symbol (Voltage) and as a unit (Volt).

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** | **Funktion** |
|---|---|
| <*machine_name*>.ini | Configuration file for the machine. |
| <*machine_name*>.hal | HAL for the machine. |
| <*machine_name*>.prefs | Configuration file for QtPlasmaC specific parameters and preferences. |
| custom.hal | HAL file for user customization. |
| custom_postgui.hal | HAL file for user customization which is run after the GUI has initialized. |
| shutdown.hal | HAL file which is run during the shutdown sequence. |
| tool.tbl | Tool table used to store offset information for additional tools (scribe, etc.) used by the QtPlasmaC configuration. |
| qtplasmac | Link to the directory containing common qtplasmac support files. |
| backup | 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 | File for storing the material settings from the MATERIAL section of the PARAMETERS Tab. |
| update_log.txt | File for storing log of major updates.<br>Major updates are those that make any modification to a user’s configuration. |
| qtvcp.prefs | File containing the QtVCP preferences. |
| qtplasmac.qss | File storing 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]** Abschnitt

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]** Abschnitt

These variables are mandatory.

```
MIN_LIMIT = knapp unterhalb der Oberkante der Tischlatten
MAX_VELOCITY = das Doppelte des Wertes des entsprechenden Gelenks
MAX_ACCELERATION = das Doppelte des Wertes des entsprechenden Gelenks
OFFSET_AV_RATIO = 0,5
```

---

**Anmerkung**
QtPlasmaC verwendet das LinuxCNC-Feature ”Externe Offsets” für alle Z-Achsen-Bewegungen und für das Bewegen der X- und/oder Y-Achse für einen Verschleißteilwechsel im Pausenzustand. Für weitere Informationen über diese Funktion lesen Sie bitte External Axis Offsets in der LinuxCNC Dokumentation.

---

## 10.8.8 QtPlasmaC GUI Überblick

Die folgenden Abschnitte geben einen allgemeinen Überblick über das Layout von QtPlasmaC.

### 10.8.8.1 Beenden von QtPlasmaC

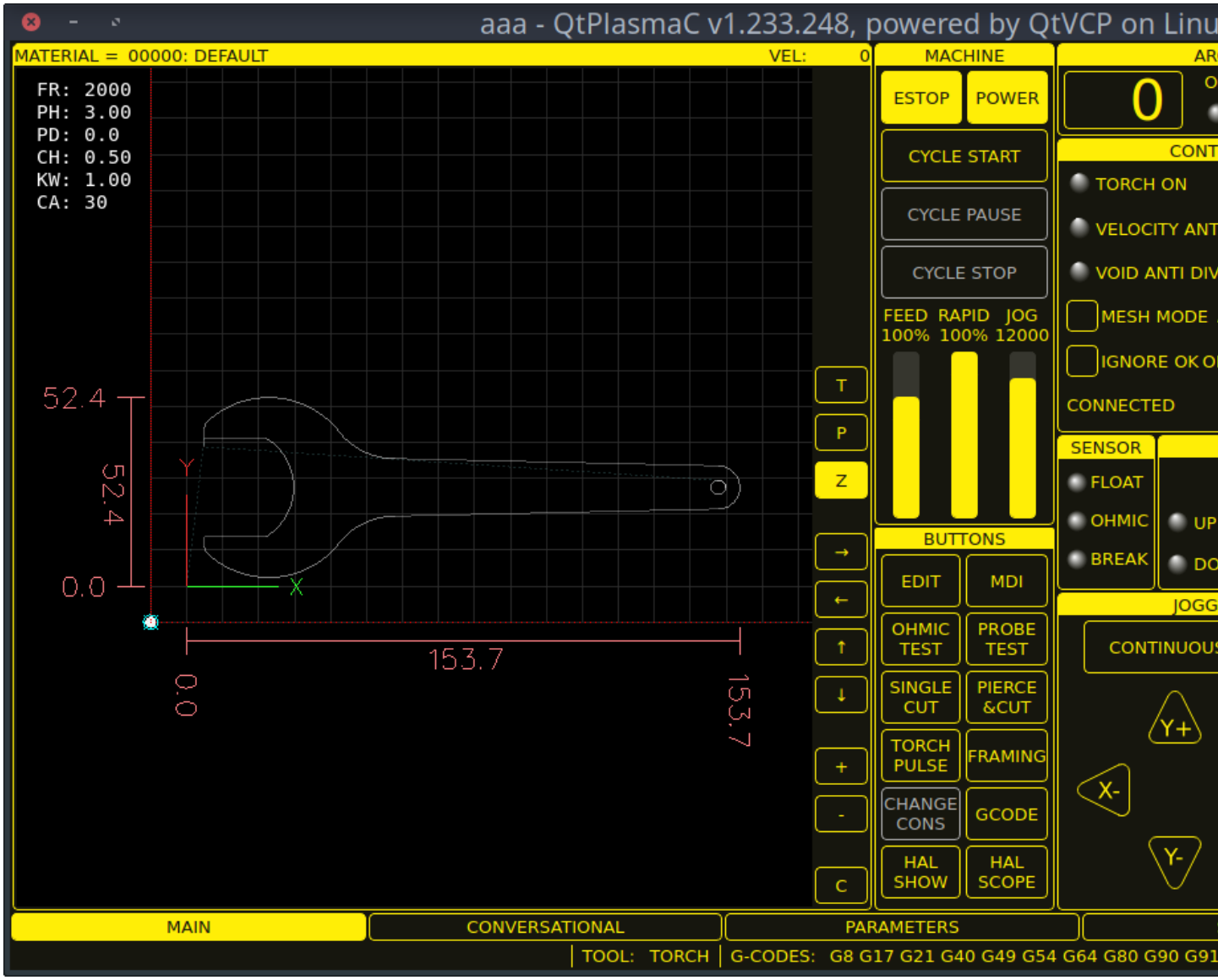
Das Beenden oder Herunterfahren von QtPlasmaC erfolgt entweder durch:

1. Klicken Sie auf die Schaltfläche zum Herunterfahren des Fensters in der Titelleiste des Fensters
2. Drücken Sie lange auf die Taste **POWER** auf der Haupt-Registerkarte (engl. main).

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 example of the QtPlasmaC MAIN 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.

Tabelle 10.7: Features of the **PREVIEW WINDOW**

| **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. |

Tabelle 10.7: (continued)

| 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. |

Tabelle 10.8: **MACHINE** representation

| Name | Beschreibung |
|---|---|
| ESTOP (engl. für Notaus) | Setting Estop type = 0 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will change this button to an indicator of the hardware E-stop’s status only. This is the default behavior.<br>Setting the option Estop type = 1 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will hide this button.<br>Setting the option Estop type = 2 in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file, will enable this button to act as a GUI E-stop. |
| POWER (engl. für Leistung oder Strom) | Diese Schaltfläche schaltet die GUI ein und erlaubt QtPlasmaC/LinuxCNC die Steuerung der Hardware.<br>Wenn Sie die **POWER**-Taste länger als zwei Sekunden gedrückt halten, wird ein Dialog zum Beenden der QtPlasmaC-Anwendung angezeigt. |
| ZYKLUSSTART | Mit dieser Button startet den Zyklus für jede geladene G-Code-Datei. |
| ZYKLUSPAUSE | Mit diesem Button wird der Zyklus für jede geladene G-Code-Datei angehalten.<br>Wenn ein Zyklus pausiert wird, zeigt dieser Button ”ZYKLUS FORTSETZEN (engl. CYCLE RESUME) an und blinkt.<br>Durch Drücken von **ZYKLUS FORTSETZEN** wird der Zyklus fortgesetzt. |
| ZYKLUS STOP (engl. cycle stop) | This button stops any actively running or paused cycle.<br>This includes:<br>- G-code Programs<br>- Torch pulse if the pulse was started during **CYCLE PAUSE** (this will cancel the paused G-code program execution as well)<br>- Probe Test<br>- Framing<br>- Manual Cut |

Tabelle 10.8: (continued)

| Name | Beschreibung |
|---|---|
| FEED | This slider overrides the feed rate for all feed moves.<br>Any value other than 100% will cause the label to flash.<br>Clicking the label will return the slider to 100%. |
| RAPID | This slider overrides the rapid rate for all rapid moves.<br>Any value other than 100% will cause the label to flash.<br>Clicking the label will return the slider to 100%. |
| JOG | This slider sets the jog rate.<br>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.<br>Once pressed, this button will display ”MDI CLOSE”.<br>Pressing **MDI CLOSE** will close the MDI.<br>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.<br>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. |

Tabelle 10.9: **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. |

Tabelle 10.10: **CONTROL**

| Name | Modi | Beschreibung |
|---|---|---|
| TORCH ON | 0, 1, 2 | Indicates the status of the Torch On output signal. |
| TORCH ON ENABLE | 0, 1, 2 | This box toggles between Enabling and Disabling the torch.<br>This box defaults to unfilled (disabled) when QtPlasmaC is first run.<br>This box must be filled to change it to ”Torch Enabled” before material cutting can commence.<br>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.<br>Mesh mode:<br>- Will require an Arc OK signal to start machine motion.<br>- Will disable the THC.<br>- Will not stop machine motion if the Arc OK signal is lost.<br>- Will automatically select CPA mode if PowerMax communications are being used.<br>For more information see Mesh Mode (expanded metal). |
| AUTO VOLTS | 0, 1 | This box will enable or disable Auto Volts. |
| 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.<br>Ignore Arc OK mode:<br>- Will not require an Arc OK signal be received before starting machine motion after the ”Torch On” signal is given.<br>- Will disable the THC.<br>- Will not stop machine motion if the Arc OK signal is lost.<br>For more information see Ignore Arc Ok. |
| OHMIC PROBE | 0, 1, 2 | Dieses Feld aktiviert oder deaktiviert den Eingang der ohmschen Sonde.<br>Wenn der Eingang der ohmschen Sonde deaktiviert ist, zeigt die LED der ohmschen Sonde weiterhin den Status des Sondeneingangs an, aber die Ergebnisse der ohmschen Sonde werden ignoriert. |
| RS485 | 0, 1, 2 | This box will enable or disable the communications to a PowerMax. This button is only visible if a PM_PORT option is configured in the `[POWERMAX]` section of the <*machine_name*>.prefs file. |

Tabelle 10.10: (continued)

| Name | Modi | Beschreibung |
|---|---|---|
| Status | 0, 1, 2 | When PowerMax communications are enabled, this will display one of the following:<br>**CONNECTING**, **CONNECTED**, **COMMS ERROR**, or a **Fault Code**.<br>For more information, see the PowerMax Communications section. |

Tabelle 10.11: 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. |

Tabelle 10.12: 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. |

**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. |

**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.<br>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.<br>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.<br>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. |

Tabelle 10.13: **G-CODE WINDOW**

| Name | Beschreibung |
|---|---|
| CLEAR | This button will clear the currently opened program.<br>If a file is open, the default material will be selected.<br>If no file is open, the preview will be reset to a top down full table view.<br>The torch (T0) will be selected if it was not the active tool. |
| OPEN | Diese Schaltfläche öffnet ein DATEI-ÖFFNEN-Panel über dem VORSCHAU-FENSTER. |
| NEU LADEN | Diese Schaltfläche lädt die aktuell geladene G-Code-Datei neu. |

Tabelle 10.14: **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.<br>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. |

Tabelle 10.14: (continued)

| Name | Beschreibung |
|---|---|
| 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:<br>**Zero** - zeros the axis.<br>**Set** - launches a dialog box to manually input the axis' coordinate.<br>**Divide By 2** - divides the currently displayed coordinate in the DRO by two.<br>**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.

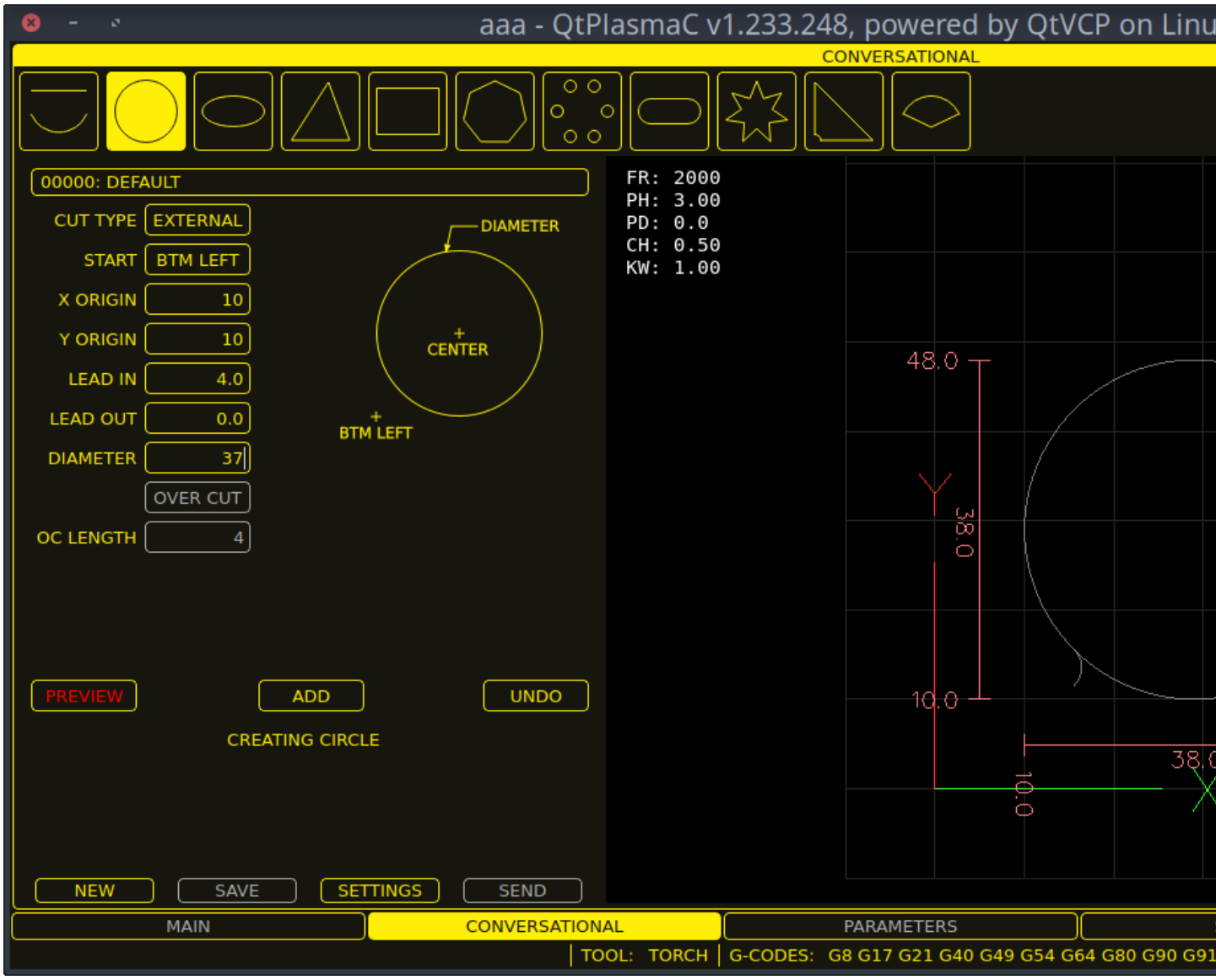
When a G-code file is loaded, the display will change to the selected view.

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 hide 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 configuration parameters that are modified infrequently.

It is possible to hide 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
```

Tabelle 10.15: **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. |

Tabelle 10.15: (continued)

| Name | Modi | Beschreibung |
|---|---|---|
| Retry Delay | 0, 1, 2 | This sets the time (in seconds) between an arc failure and another arc start attempt. |
| Voltage Scale | 0, 1 | This sets the arc voltage input scale and is used to display the correct arc voltage.<br>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.<br>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.<br>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.

Tabelle 10.16: **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.

Tabelle 10.17: **CONFIGURATION - SAFETY**

| **Name** | **Beschreibung** |
|---|---|
| Safe Height | This sets the height above the material that the torch will retract to before executing rapid moves.<br>If set to Zero then Z axis maximum height will be used for the safe height.<br>Refer to the Heights Diagram diagram for a visual representation. |

Tabelle 10.18: **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. |

Tabelle 10.19: **CONFIGURATION - SPOTTING**

| **Name** | **Beschreibung** |
|---|---|
| Threshold | This sets the arc voltage at which the delay timer will begin.<br>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. |

Tabelle 10.20: **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. |
| 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.

Tabelle 10.21: **MATERIAL** - 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. |

Tabelle 10.21: (continued)

| Name | Beschreibung |
|---|---|
| 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.<br>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.<br>Typically used for thicker materials, Puddle Jump allows the torch to have an intermediate step between Pierce Height and Cut Height.<br>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. |
| 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.<br>This setting is only valid if PowerMax communications are being used.<br>0 = Use the PowerMax’s automatic pressure mode. |
| Cut Mode | This sets the cut mode for the currently selected material.<br>This setting is only valid if PowerMax communications are being used.<br>1 = Normal<br>2 = CPA (Constant Pilot Arc)<br>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.

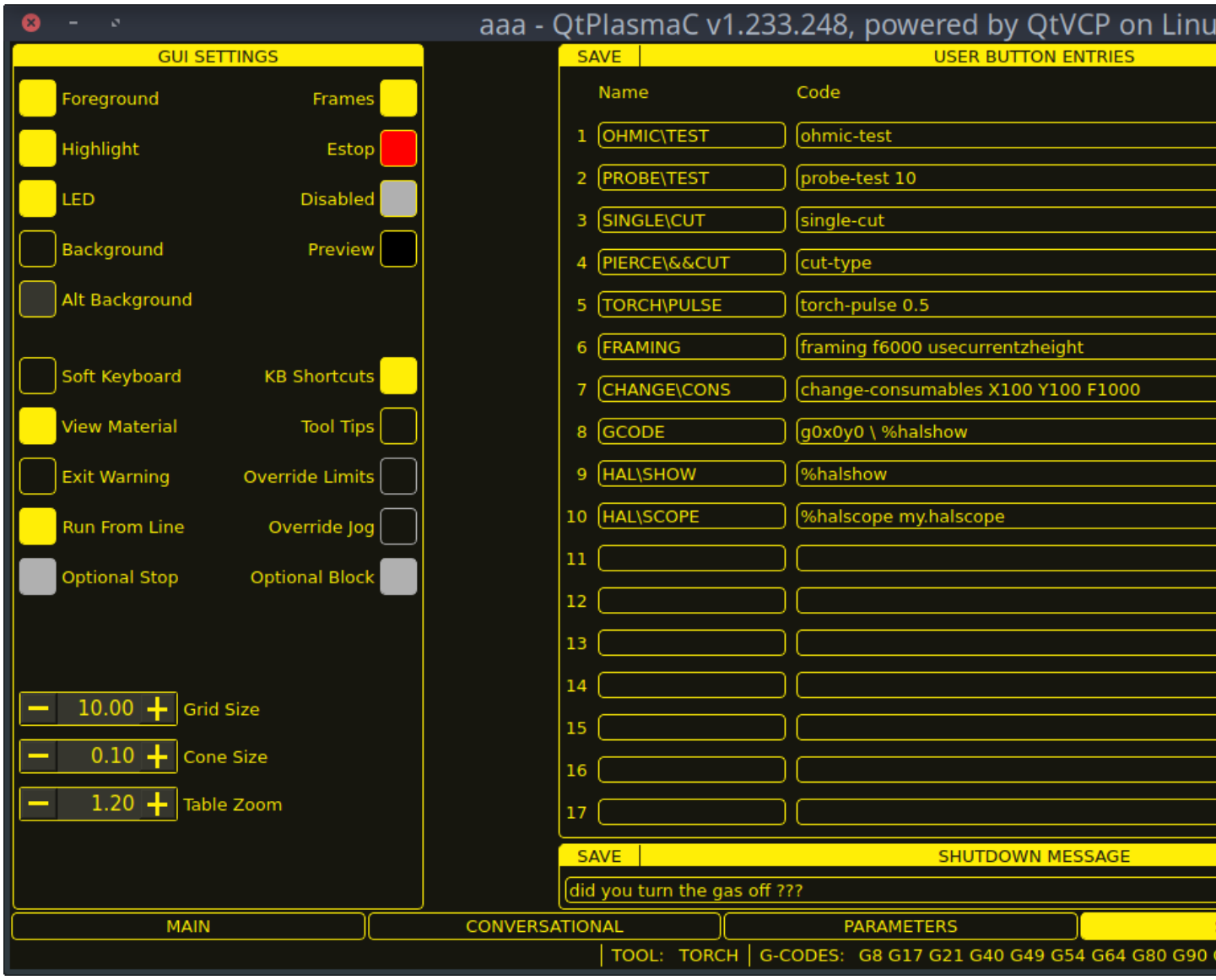
The **RELOAD** button will reload the material set from the <*machine_name*>_material.cfg file.

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:




Diese Registerkarte wird verwendet, um GUI-Konfigurationsparameter, Schaltflächentext und Herunterfahrtext anzuzeigen, die selten geändert werden, sowie einige Dienstprogrammschaltflächen.

It is possible to hide 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.

Tabelle 10.22: **GUI SETTINGS** Parameters that effect the GUI appearance and GUI behaviors.

| 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. |

Tabelle 10.22: (continued)

| Name | Beschreibung |
|---|---|
| Background | This button allows the user to change the color of the GUI Background. |
| 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.<br>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).<br>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.<br>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.<br>It is possible to add a custom message to the warning by editing the EXIT WARNING MESSAGE option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file.<br>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**

Dieser Abschnitt zeigt den Text, der im Dialogfeld zum Herunterfahren angezeigt wird, wenn die Option **Exit-Warnung** aktiviert ist.

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 formatted 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.

**Version Information** - QtPlasmaC will display versioning information in the title of the main window. The information will be displayed as followed ”QtPlasmaC v*N.XXX.YYY* - powered by QtVCP on LinuxCNC vZ.Z.Z” where *N* is the version of QtPlasmaC, *XXX* is the version of the HAL component (PlasmaC.comp), *YYY* is the GUI version, and *Z.Z.Z* is the version of LinuxCNC.

#### 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.

If the user is attempting to run a G-code file that is in the ”other” units system then all parameters including the material file parameters are still required to be in the native machines units. Any further conversions necessary to run the G-code file will be handled automatically by the G-code filter program.

For example: If a user had a metric machine and wished to run a G-code file that was set up to cut 1/4” thick material using imperial units (inch - G20) then the user with the metric machine would need to ensure that either the material number in the G-code file was set to the corresponding metric material to be cut, or that a new material is created with the correct metric parameters for the metric material to be cut. If the metric user wanted to cut the G-code file using imperial material, then the new material parameters would need to be converted from imperial units to metric when they are entered.

#### 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

Aside from the preamble code, postamble code, and X/Y motion code, the only mandatory G-code syntax for QtPlasmaC to run a G-code program using a torch for cutting is `M3 $0 S1` to begin a cut and `M5 $0` to end a cut.

For backwards compatibility it is permissible to use `M3 S1` in lieu of `M3 $0 S1` to begin a cutting job and `M5` in lieu of `M5 $0` to end a cutting job. Note, that this applies to cutting jobs only, for scribe and spotting jobs the `$n` tool identifier is mandatory.

#### 10.8.9.4 Koordinaten

See recommended Z axis settings.

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.

Depending on the machine's configuration there will either be a **Home All** button or each axis will need to be homed individually. Use the appropriate button/buttons to home the machine.

As mentioned in the Initial Setup section, it is recommended that the first time QtPlasmaC is used that the user ensure there is nothing below the torch then jog the Z axis down until it stops at the Z axis MINIMUM_LIMIT then click the 0 next to the Z axis DRO to **Touch Off** with the Z axis selected to set the Z axis at zero offset. This should not need to be done again.

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 is able to read a material file to load all the required cut parameters. To enable to G-code file to use the cut feed rate setting from the cut parameters use the following code in the G-code file:

```
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

Material handling uses a material file that was created for the machine configuration when the configuration wizard was ran and allows the user to conveniently store known material settings for easy recall either manually or automatically via G-code. The resulting material file is named ***<machine_name>*_material.cfg**.

QtPlasmaC does not require the use of a material file. Instead, the user could change the cut parameters manually from the MATERIAL section of the PARAMETERS Tab. It is also not required to use the automatic material changes. If the user does not wish to use this feature they can simply omit the material change codes from the G-code file.

It is also possible to not use the material file and automatically load materials from within the G-code file.

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)
```

It is possible to add new material, delete material, or edit existing material from the PARAMETERS tab.. It is also possible to achieve this by using magic comments in a G-code file.

The material file may be edited with a text editor while LinuxCNC is running. After any changes have been saved, press **Reload** in the MATERIAL section of the PARAMETERS Tab to reload the material file.

#### 10.8.9.7 Manuelles Materialhandling

For manual material handling, the user would manually select the material from the materials list in the MATERIAL section of the PARAMETERS Tab before starting the G-code program. In addition to selecting materials with materials list in the MATERIAL section of the PARAMETERS Tab, the user could use the MDI to change materials with the following command:

```
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]>** - Sets the cut feed rate to the feed rate shown in the MATERIAL section of the PARAMETERS Tab.

For automatic material handling, the codes MUST be applied in the order shown. If a G-code program is loaded which contains one or more material change commands then the first material will be displayed in the top header of the PREVIEW WINDOW on the MAIN Tab as the program is loading.

**Minimum code necessary to have a successful cut using the automatic material selection method:**

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

**Anmerkung**
Returning to the default material prior to the end of the program is possible with the code **M190 P-1**.

#### 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 | Erzeugt ein temporäres Standardmaterial.<br>Die mit dieser Option hinzugefügten Materialinformationen werden durch einen LinuxCNC-Neustart oder ein Neuladen der Materialien verworfen. Sie können auch durch eine neue G-Code-Datei überschrieben werden, die temporäre Materialien enthält. |
| 1 | Fügt ein neues Material hinzu, wenn die angegebene Nummer nicht vorhanden ist. |
| 2 | Überschreibt ein vorhandenes Material, wenn die angegebene Nummer existiert.<br>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
```

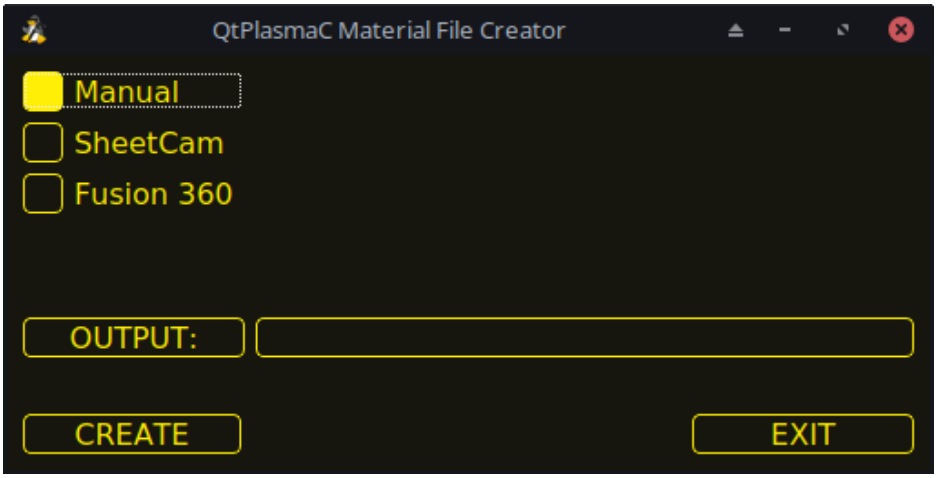
Geben Sie für eine ”run in place”-Installation die folgenden beiden Befehle in ein Terminalfenster ein:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

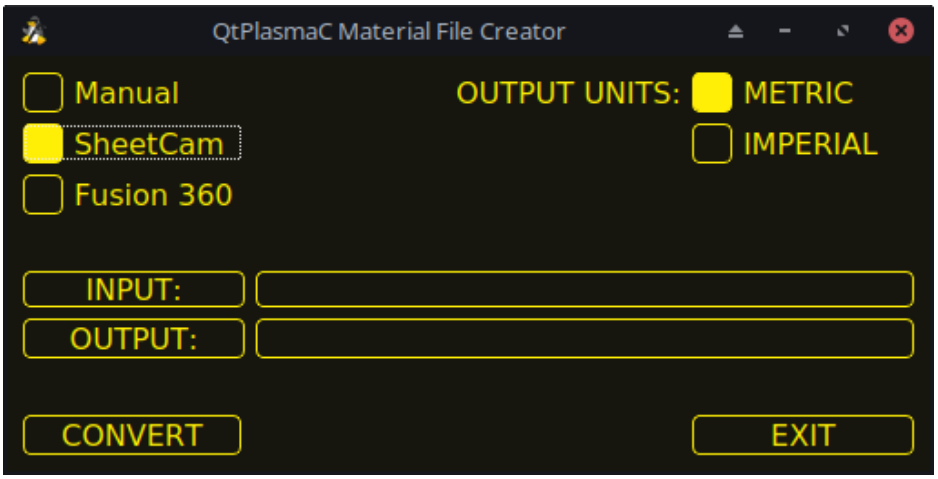
Daraufhin wird das Hauptdialogfeld Materialkonverter mit der Standardeinstellung Manuell angezeigt.

Wählen Sie eines aus:

- **Manuell** - um manuell eine neue Materialdatei zu erstellen.

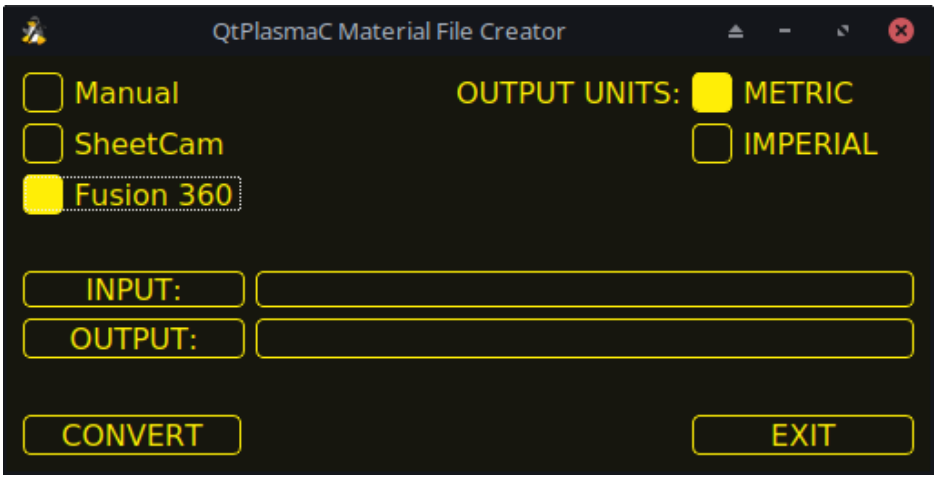



- **SheetCam** - um eine SheetCam-Werkzeugdatei zu konvertieren.




Wählen Sie nur für SheetCam, ob der Benutzer eine metrische oder imperiale Ausgabedatei benötigt.

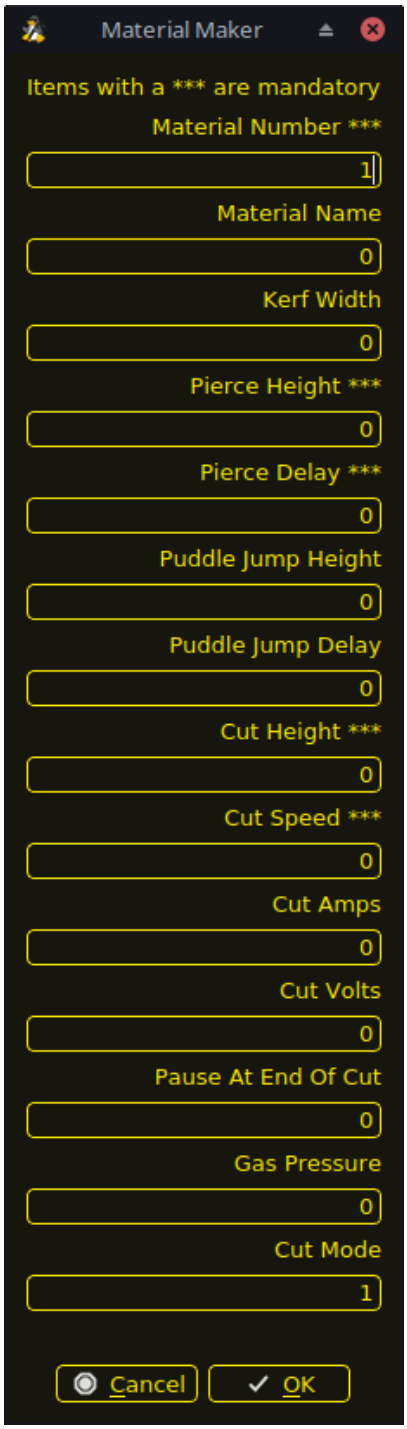
- **Fusion 360** - zum Konvertieren einer Fusion 360 Werkzeugdatei.




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. Select the Output File to write to, press **OUTPUT** to bring up a file selector or directly enter the file in the entry box. This would normally be ~/linuxcnc/configs/<*machine_name*>_material.cfg. If necessary, the user could select a different file and hand edit the <*machine_name*>_material.cfg file.
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**
If the user selects ~/linuxcnc/configs/<*machine_name*>_material.cfg and the file already exists, it will be overwritten.

#### 10.8.9.11 LASER

QtPlasmaC has the ability to use a laser to set the origin with or without rotation compensation. Rotation compensation can be used to align the work offset to a sheet of material with edge(s) that

are not parallel to the machine's X/Y axes. The LASER button will be enabled after the machine is homed.

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 options 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:**

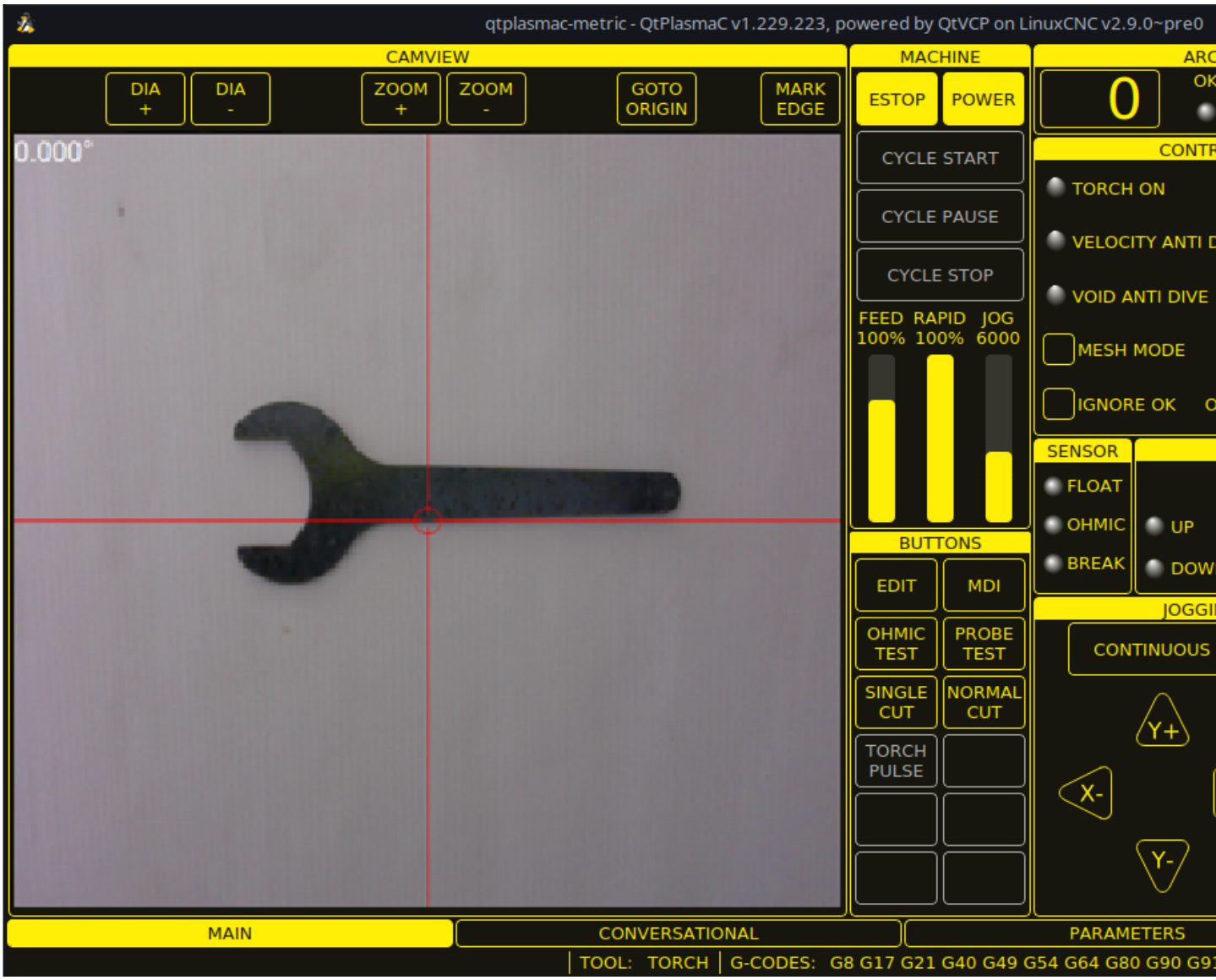
1. Klicken Sie auf die Schaltfläche **LASER**.
2. Die Beschriftung des Buttons **LASER** ändert sich zu **MARK EDGE** und der HAL-Pin namens qtplasmac.laser_on wird eingeschaltet.
3. Rütteln Sie so lange, bis sich das Fadenkreuz des Lasers an der Kante des Materials in einem angemessenen Abstand zum 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.

If an alignment laser has been set up then it is possible to use the laser during CUT RECOVERY for accurate positioning of the new start coordinates.

#### 10.8.9.12 CAMERA




QtPlasmaC has the ability to use a USB camera to set the origin with or without rotation compensation. Rotation compensation can be used to align the work offset to a sheet of material with edge(s) that are not parallel to the machine's X/Y axes. The CAMERA button will be enabled after the machine is homed.

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 axes options in the **[CAMERA_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
Camera port = 0
```

wobei *n.n* der Abstand von der Mittellinie des Brenners zum Fadenkreuz der Kamera ist.

**Ursprung mit Null-Drehung festlegen:**

1. Bewegen Sie die Maus, bis sich das Fadenkreuz über dem gewünschten Ursprungspunkt befindet.
2. Drücken Sie **MARK EDGE**. Die Beschriftung der Schaltfläche **MARK EDGE** ändert sich in **SET ORIGIN** und die Schaltfläche **GOTO ORIGIN** wird deaktiviert.
3. Drücken Sie **Ursprung festlegen**. Die Beschriftung der Schaltfläche **Herkunft festlegen** ändert sich in **KANTE MARKIEREN** und die Schaltfläche **Herkunft gehen** wird aktiviert.
4. Der Brenner fährt nun in die Position X0 Y0.
5. Der Offset ist nun erfolgreich gesetzt.

**Ursprung mit Drehung festlegen:**

1. Bewegen Sie das Fadenkreuz, bis es sich am Rand des Materials in einem angemessenen Abstand zum gewünschten Ursprungspunkt befindet.
2. Drücken Sie **MARK EDGE**. Die Beschriftung der Schaltfläche **MARK EDGE** ändert sich in **SET ORIGIN** und die Schaltfläche **GOTO ORIGIN** wird deaktiviert.
3. Bewegen Sie die Maus, bis sich das Fadenkreuz am Ausgangspunkt des Materials befindet.
4. Drücken Sie **Ursprung festlegen**. Die Beschriftung der Schaltfläche **Herkunft festlegen** ändert sich in **KANTE MARKIEREN** und die Schaltfläche **Herkunft gehen** wird aktiviert.
5. Der Brenner fährt nun in die Position X0 Y0.
6. Der Offset ist nun erfolgreich eingestellt.

Im CAMVIEW-Bedienfeld kann die Maus das Fadenkreuz und die Zoomstufe wie folgt beeinflussen:

- Mausrad scrollen - Durchmesser des Fadenkreuzes ändern.
- Doppelklick mit der Mausradtaste - Stellt den Fadenkreuzdurchmesser auf den Standardwert zurück.
- Linke Maustaste gedrückt + Scrollrad - Ändert die Zoomstufe der Kamera.
- Klicken mit der linken Maustaste + Doppelklick mit der Radtaste - Stellt die Standard-Zoomstufe der Kamera wieder her.

#### 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.

Using this feature will cause all motion to pause at the end of the cut while the torch is still on. After the dwell time (in seconds) set by the **Pause At End** parameter in the MATERIAL section of the PARAMETERS Tab has expired, QtPlasmaC will proceed with the M5 command to turn off and raise the torch.

#### 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:

- `M3 $0 S1` will select and start the plasma cutting tool.
- `M3 $1 S1` will select and start the scribe.
- `M3 $2 S1` will select and start the plasma spotting tool.
- `M5 $0` will stop the plasma cutting tool.
- `M5 $1` will stop the scribe.
- `M5 $2` will stop the plasma spotting tool.

Es ist zulässig, **M5 $-1** anstelle der obigen M5 $*n*-Codes zu verwenden, um alle Werkzeuge anzuhalten.

In order to use a scribe, it is necessary for the user to add the X and Y axis offsets to the LinuxCNC tool table. Tool 0 is assigned to the Plasma Torch and Tool 1 is assigned to the scribe. Tools are selected with a **T*n* M6** command, and then a **G43 H0** command is required to apply the offsets for the selected tool. It is important to note that the LinuxCNC tool table and tool commands only come into play if the user is using a scribe in addition to a plasma torch. For more information, see 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` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M67.
- `M68` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

Beispiele:

- `M67 E3 Q0` would set the velocity to 100% of **CutFeedRate**.
- `M67 E3 Q40` would set the velocity to 40% of **CutFeedRate**.
- `M67 E3 Q60` would set the velocity to 60% of **CutFeedRate**.

- `M67 E3 Q100` would set the velocity to 100% of **CutFeedRate**.

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.

If the user intends to use this feature it would be prudent to add `M68 E3 Q0` to both the preamble and postamble of the G-code program so the machine starts and ends in a known state.

**Wichtig**
**G-CODE THC** UND **VELOCITY BASED THC** KÖNNEN NICHT VERWENDET WERDEN, WENN **CUTTER COMPENSATION** IN KRAFT IST; ES WIRD EINE FEHLERMELDUNG ANGEZEIGT.

**Warnung**
If Cut Feed Rate in the MATERIAL section of the PARAMETERS Tab is set to Zero then QtPlasmaC will use **motion.requested-velocity** (as set by a standard Feedrate call in the G-code) for the THC calculations. This is not recommended as it is not a reliable way of implementing velocity based THC.

**Anmerkung**
All references to CutFeedRate refer to the **Cut Feed Rate** value displayed in the MATERIAL section of the PARAMETERS Tab.

#### 10.8.9.18 THC (Brennerhöhensteuerung, engl. torch height controller)

The THC can be enabled or disabled from the THC frame of the MAIN Tab.

Die THC kann auch direkt über das G-Code-Programm aktiviert oder deaktiviert werden.

The THC does not become active until the velocity reaches 99.9% of the **CutFeedRate** and then the THC **Delay** time if any in the THC section of the PARAMETERS Tab has timed out. This is to allow the arc voltage to stabilize.

QtPlasmaC uses a control voltage which is dependent on the state of the **AUTO VOLTS** checkbox on the MAIN Tab:

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. If **Use Auto Volts** is not checked then the voltage displayed as Cut Volts in the MATERIAL section of the PARAMETERS Tab is used as the target voltage to adjust the height of the torch.

**G-code THC** THC may be disabled and enabled directly from G-code, provided the THC is not disabled in the THC Section of the MAIN Tab, by setting or resetting the **motion.digital-out-02** pin with the M-Codes M62-M65:

- `M62 P2` will disable THC (Synchronized with Motion)
- `M63 P2` will enable THC (Synchronized with Motion)
- `M64 P2` will disable THC (Immediately)
- `M65 P2` will enable THC (Immediately)

Es ist wichtig, den Unterschied zwischen **Synchronisiert mit Bewegung** und **Sofort** gründlich zu verstehen:

- `M62` and `M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64` and `M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

**Velocity Based THC**

If the cut velocity falls below a percentage of **CutFeedRate** (as defined by the VAD Threshold % value in the THC frame of the CONFIGURATION section of the PARAMETERS Tab) the THC will be locked until the cut velocity returns to at least 99.9% of **CutFeedRate**. This will be made apparent by the **VELOCITY ANTI DIVE** indicator illuminating in the CONTROL Panel on the MAIN Tab.

Die geschwindigkeitsabhängige THC verhindert, dass die Brennerhöhe verändert wird, wenn die Geschwindigkeit für eine scharfe Ecke oder ein kleines Loch reduziert wird.

It is important to note that Velocity Reduction affects the Velocity Based THC in the following ways:

1. Wenn die Geschwindigkeitsreduzierung in der Mitte des Schnitts aufgerufen wird, dann wird die THC gesperrt.
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 options 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)** - Keeping the torch on for a short time at the end of the hole while motion is stopped to allow the arc to catch up.
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 for hole cutting with reduced velocity.**

```
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 4 mm (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 and M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64 and M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

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` - Causes QtPlasmaC to disable hole sensing if it was previously enabled.
- `#<holes> = 1` - Causes QtPlasmaC to reduce the speed of holes less than 32 mm (1.26”) to 60% of **CutFeedRate**.
- `#<holes> = 2` - Causes QtPlasmaC to Over cut the hole in addition to the velocity changes in setting 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` - Causes QtPlasmaC to Over cut the hole in addition to the velocity change in setting 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* - To set a diameter (*nn*) in the same units system as the rest of the G-code file.

The default velocity for QtPlasmaC small holes is 60% of the current feed rate. It is possible to change this value with the following command in a G-code file:

- `#<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 4 mm (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

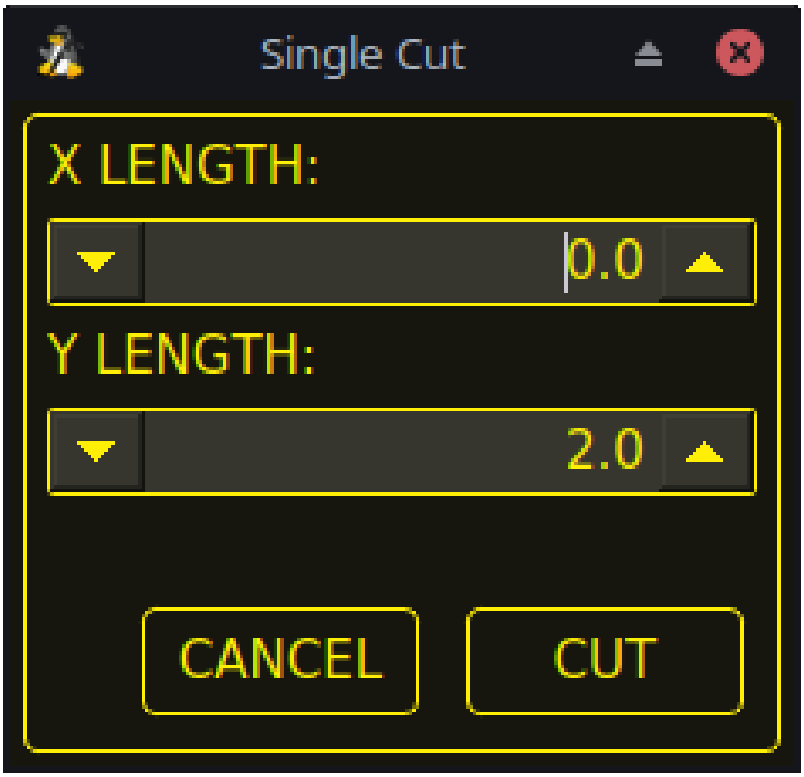
A single cut is a single unidirectional cutting move often used to cut a sheet into smaller pieces prior to running a G-code program.

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. Wenn der Schnitt beendet ist, stoppen Sie die Spindel.
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 that 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)

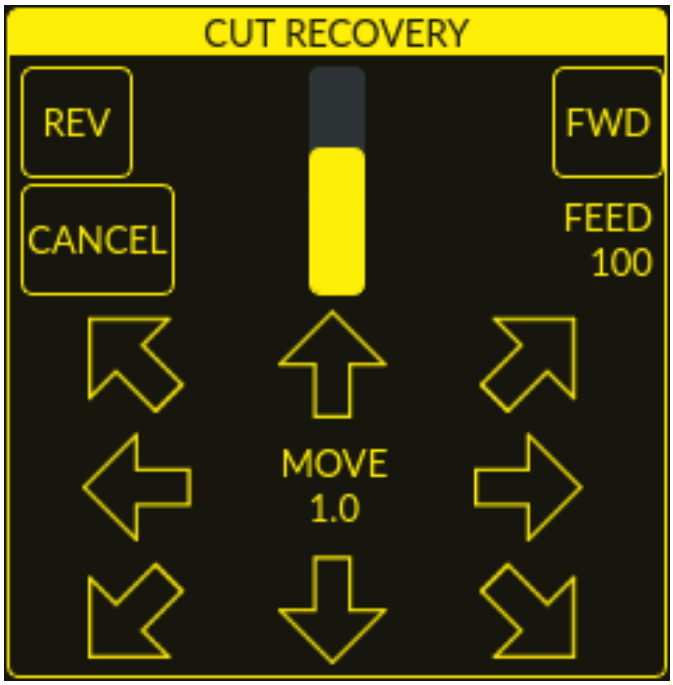
It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- `M62` and `M63` (Synchronized with Motion) - The actual change of the specified output (P2 (THC) for example) will happen at the beginning of the next motion command. If there is no subsequent motion command, the output changes will not occur. It is best practice to program a motion code (G0 or G1 for example) right after a M62 or M63.
- `M64` and `M65` (Immediate) - These commands happen immediately as they are received by the motion controller. Since these are not synchronized with motion, they will break blending. This means if these codes are used in the middle of active motion codes, the motion will pause to activate these commands.

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.

Pressing **CANCEL MOVE** will cause the torch to move back to where it was positioned before the direction keys were used to offset the torch. It will not reset any **REV** or **FWD** motion.

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 10 mm (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 then 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.

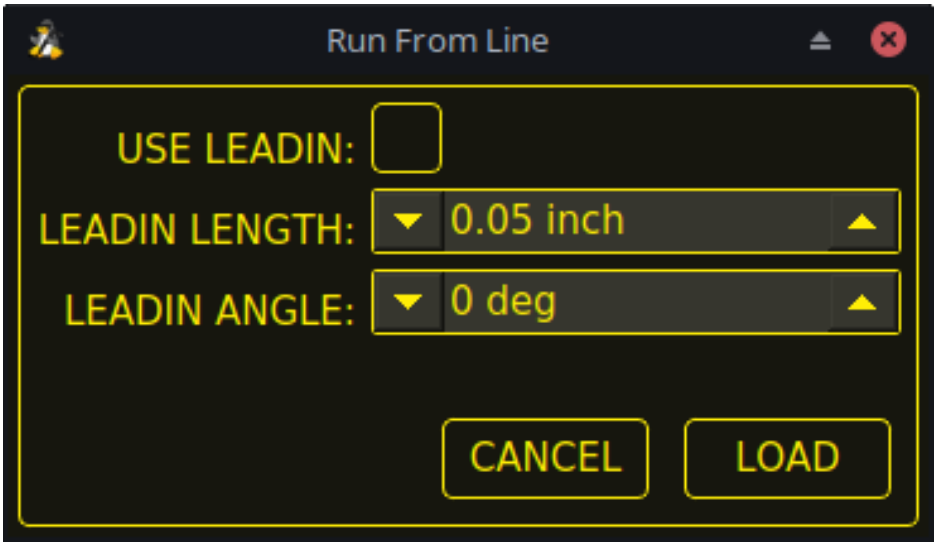
It is important to note that G-code programs can be run from any selected line using this method, however a leadin may not be possible depending on the line selected. In this case, an error message will be displayed to let the user know the leadin calculation was not possible.

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.<br>The angle is measured such that positive increases in value move the leadin counter-clockwise:<br>0 Degrees = 3 o’clock position<br>90 Degrees = 12 o’clock position<br>180 Degrees = 9 o’clock position<br>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.<br>If the leadin cannot be calculated for the selected line, the following error message will be displayed:<br>”Unable to calculate a leadin for this cut<br>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.

Using a scribe requires the use of the LinuxCNC tool table. Tool 0 is assigned to the plasma torch and Tool 1 is assigned to the scribe. The scribe X and Y axes offsets from the plasma torch need to be entered into the LinuxCNC tool table. This is done by editing the tool table via the main GUI, or by editing the **tool.tbl** file in the ***<machine_name>*** configuration directory. This will be done after the scribe can move to the work piece to help determine the appropriate offset.

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**.

For more information on multiple tools, see multiple tools.

LinuxCNC (QtPlasmaC) requires some motion between any **M3** and **M5** commands. For this reason, a minimal movement at a high speed is required to be programmed.

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 10.48: Number keypad - used for the CONVERSATIONAL Tab and the PARAMETERS Tab

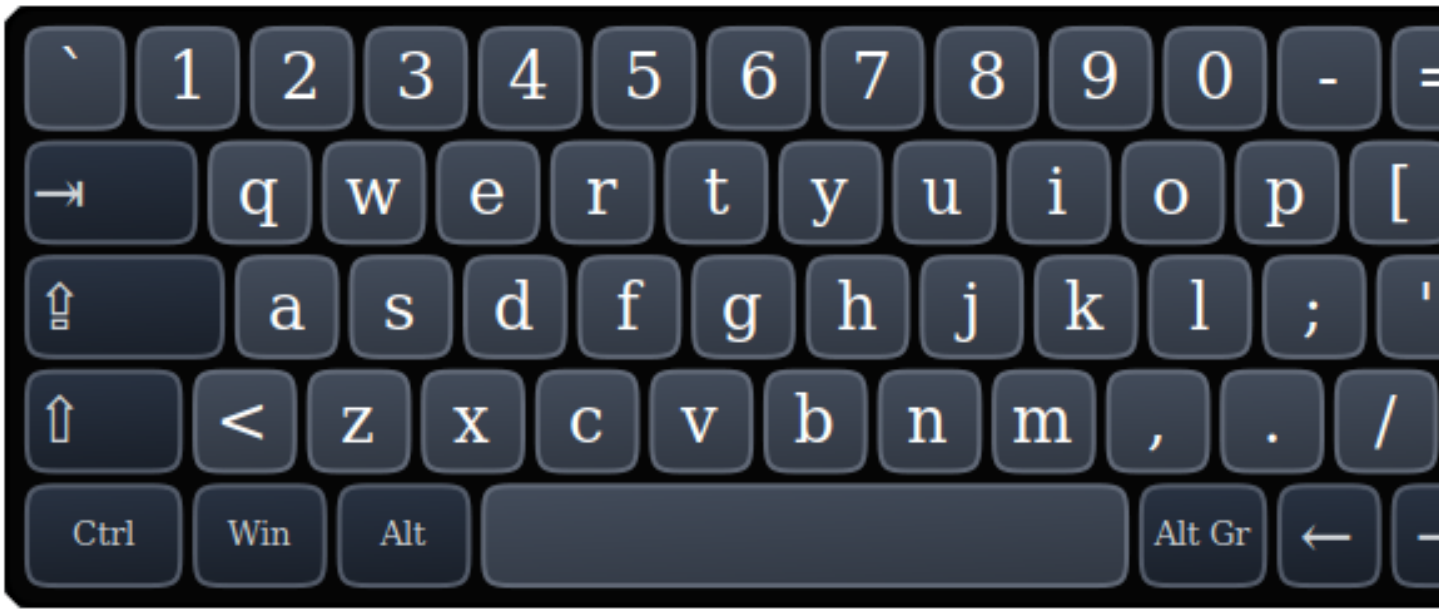



Abbildung 10.49: Alpha-numeric keypad - used for G-code editing and file management.

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.<br>Note that ALT + ENTER will achieve the same result.<br>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 | Joggt die Z-Achse negativ. |
| [ | Joggt die A-Achse positiv. |
| ] | Joggt die A-Achse negativ. |
| . | Joggt die B-Achse positiv. |
| , | Joggt die B-Achse negativ. |
| SHIFT (+ Jog-Taste) | Die ”Umschalttaste” (auch ”Hochtaste”, engl. shift key) wird zusammen mit einer beliebigen Jog-Taste verwendet, um ein schnelles Joggen auszulösen. |
| + (+Jog Taste) | Die Plus-Taste kann mit jeder Jog-Taste verwendet werden, um einen schnellen Jog aufzurufen (verhält sich wie SHIFT). |

| Keyboard Shortcut | Action |
|---|---|
| - (+Jog Taste) | Die Minustaste kann mit einer beliebigen Tipptaste verwendet werden, um einen langsamen Tippbetrieb (10% der angezeigten Tippgeschwindigkeit) aufzurufen. Wenn der langsame Tippbetrieb bereits aktiv ist, wird die Achse mit der angezeigten Jog-Geschwindigkeit verfahren. |

#### 10.8.9.37 MDI

Zusätzlich zu den typischen G- und M-Codes, die von LinuxCNC im MDI-Modus erlaubt sind, kann die MDI in QtPlasmaC verwendet werden, um auf mehrere andere praktische Funktionen zuzugreifen. Der folgende Link umreißt die Funktionen und ihre Verwendung: Abschnitt 12.7.2.21[MDILine Widget]

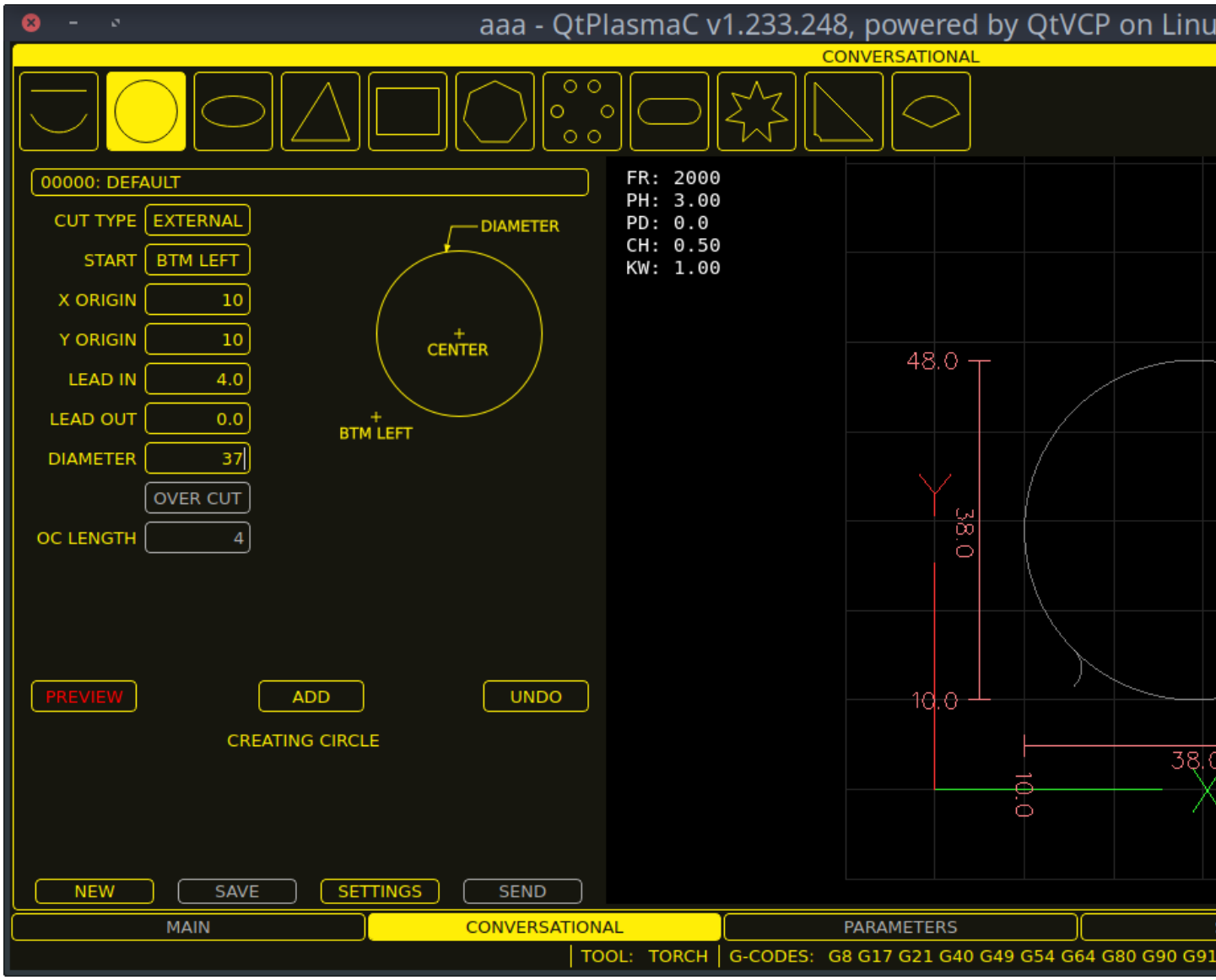
---

**Anmerkung**
M3, M4 und M5 sind in der QtPlasmaC MDI nicht erlaubt.

---

*Außerdem wird das MDI-Fenster geschlossen, wenn Sie RETURN (oder ENTER) drücken, ohne dass ein Eintrag in der MDI erfolgt.

### 10.8.10 Conversational Shape 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.

---

Bei leeren Einträgen in den Form-Eingabefeldern wird die aktuelle Einstellung zum Zeitpunkt der Erstellung des G-Codes verwendet. Wenn z.B. **X Start** leer gelassen wurde, wird die aktuelle Position der X-Achse verwendet.

Alle An- und Ableitungen (engl. leadins und leadouts) sind Bögen, mit Ausnahme von **Kreisen** und **Sternen**:

**Kreise** (engl. circles):

- Wenn der Kreis extern ist, dann ist jede Hin- oder Rückführung (leadin or leadout) ein Bogen.

- Wenn der Kreis innenliegend ist und ein **kleines Loch** hat, dann ist jeder leadin senkrecht und es gibt keinen Auslauf.
- Wenn der Kreis intern und kein **kleines Loch** ist, dann ist jeder Leadin und Leadout ein Bogen. Wenn der Leadin eine Länge von mehr als der Hälfte des Radius hat, wird der Leadin senkrecht zurückgesetzt und es gibt keinen Leadout. Wenn der Leadout eine Länge von mehr als der Hälfte des Radius hat, gibt es keinen Leadout.

**Sterne** (engl. stars):

- Der Leadin befindet sich im gleichen Winkel wie der erste Schnitt und der Leadout im gleichen Winkel wie der letzte Schnitt.

---

**Anmerkung**
Ein **kleines Loch** ist ein Kreis, der kleiner ist als der auf der Seite KONVERSATIONSEINSTELLUNGEN (engl. conversational settings) angegebene KLEINE LOCHDURCHMESSER.

---

---

**Anmerkung**
The holes in a BOLT CIRCLE shape will also abide by the above rules.

---

*Die Schnittreihenfolge entspricht der Reihenfolge, in der die Form gebaut wurde.

Wenn Sie während der Bearbeitung der Parameter **Return** auf der Tastatur drücken, wird automatisch die Vorschau der Form angezeigt, wenn genügend Parameter eingegeben wurden, um die Form zu erstellen. Ein Klick auf eines der verfügbaren Kontrollkästchen bewirkt dasselbe.

Die allgemeinen Funktionen sind wie folgt:

| **Name** | **Beschreibung** |
|---|---|
| Material-Dropdown | Allows the user to select the desired material for cutting.<br>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.<br>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. |
| NEU LADEN | 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

Global settings for the shape library can be set by pressing the **SETTINGS** button in the CONVERSATIONAL Tab. This will display all of the available settings parameters that are used for G-code program creation. These include:

- **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:

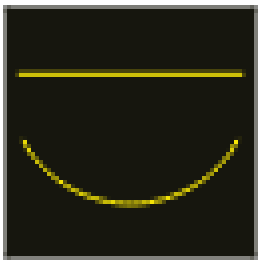
```
G21\nG40\nG49\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

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.

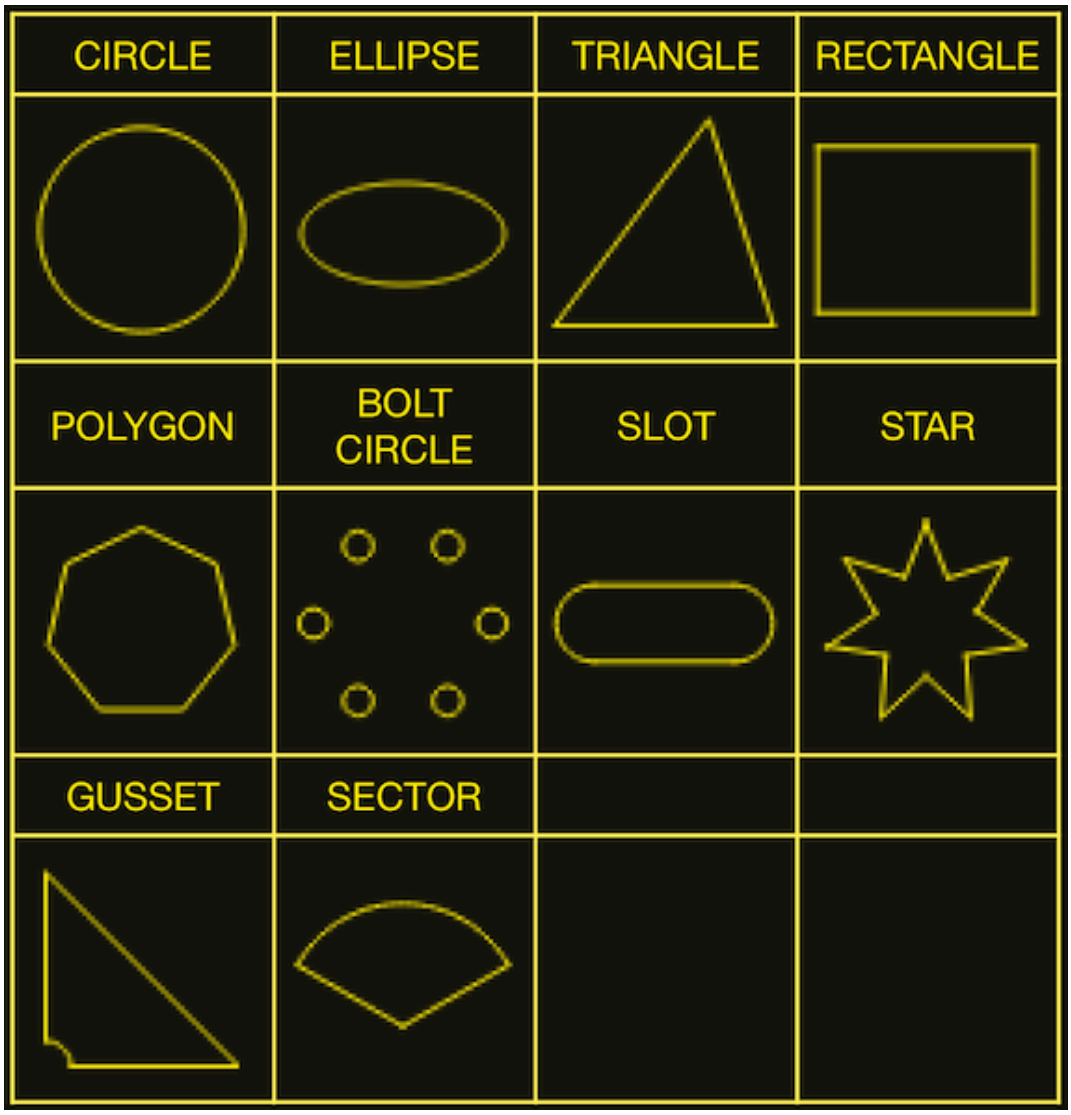
---

**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.

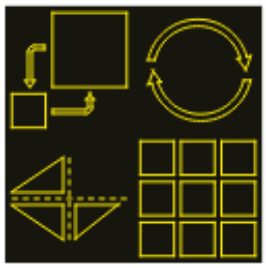
Once a shape is added to the group it cannot be edited or removed.

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



The Conversational Block feature allows block operations to be performed on the current shape or group of shapes displayed in the CONVERSATIONAL Tab. This can include a G-code file not created using the Conversational Shape Library that has been previously loaded from the MAIN Tab.

A previously saved Block G-code file may also be loaded from the MAIN Tab and then have any of its operations edited using the Conversational Block feature.

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

The current job displayed in the Preview Panel may be saved at any time by using the bottom **SAVE** button. If the G-code has been sent to LinuxCNC (QtPlasmaC) and the user has left the CONVERSATIONAL Tab, the user may still save the G-code file from the GUI. Alternatively, the user could click the CONVERSATIONAL Tab which will reload the job, at which time they can press the **SAVE** button.

### 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 option 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 option 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**

Diese Fehler deuten darauf hin, dass der entsprechende Sensor aktiviert wurde, bevor die Sondierung begann:

- **ohmische Sonde erkannt, bevor das Sondenprogramm pausiert wird**
- **float switch detected before probing program is paused**
- **Ausbruchsschalter erkannt, bevor das Antastprogramm unterbrochen wird**.

Das Lichtbogen-OK-Signal ging während der Schneidbewegung verloren, bevor der Befehl **M5** erreicht wurde:

- **gültiger Lichtbogen verloren Programm wird angehalten**

Die Z-Achse erreichte die untere Grenze, bevor das Werkstück erkannt wurde:

- **Untere Grenze erreicht, während das Sondierungsprogramm pausiert wird**.

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:

- **kein Lichtbogen erkannt nach <*n*>d Startversuchen Programm ist pausiert**
- **kein Lichtbogen erkannt nach <*n*>d Startversuchen Manueller Schnitt wird gestoppt**

THC hat dazu geführt, dass die Untergrenze beim Schneiden erreicht wurde:

- **Untere Grenze erreicht, während das THC-Abwärtsprogramm pausiert**.

THC hat dazu geführt, dass die Obergrenze beim Schneiden erreicht wird:

- **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 Messages

Warning messages will not pause a running program and are informational only.

These messages 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 enabled by setting the following option in the **[GUI_OPTIONS]** section of the *<machine_name>*.prefs file. This option must be set to the filename of the stylesheet as shown below.

```
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<br>foreground of latching user buttons<br>foreground of camera/laser buttons<br>foreground of conversational shape buttons<br>background of active conversational shape buttons |
| color2 (#16160e) | Background | background of latching user buttons<br>background of camera/laser buttons<br>background of G-code editor active line<br>background of conversational shape buttons |
| color3 (#ffee06) | Highlight | background of active latching user buttons<br>background of active camera/laser buttons<br>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 option 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 and options 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 ways: 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 only once during startup.

```
USER_COMMAND_FILE = my_custom_code.py
```

A user periodic file must be named user_periodic.py and must be loaded in the machines config directory. This file is processed every cycle (usually 100 ms) and is used for functions that require regular updating.

#### 10.8.14.5 Custom G-code Filter

All incoming G-code is parsed by a G-code filter to ensure it is suitable for QtPlasmaC. It is possible to extend this filter with custom python code executed from a file in the configuration directory to aid in converting different flavours of G-code to a format suitable for QtPlasmaC.

The name of this file is custom_filter.py and it will be automatically used if it exists.

There are three preset methods available for use:

| Name | Funktion |
|---|---|
| custom_pre_process | This does basic processing of each line before any processing is done in the filter. |
| custom_pre_parse | This parses any G-code from a line **before** any parsing done in the filter. |
| custom_post_parse | This parses any G-code from a line **after** any parsing done in the filter. |

These methods are applied by the following procedure:

- Define the method with an argument for the incoming data.
- Add any required code to manipulate the data.
- Return the resultant data.
- Attach the new method.

For example to remove any code beginning with *G71* and change *M2* to *M5 $0* and *M2*:

```
def custom_pre_parse(data):
    if data[:3] == 'G71':
        return(None)
    if data == 'M2':
        return(f'M5 $0\n\n{data}')
    return(data)
self.custom_pre_parse = custom_pre_parse
```

In addition to these it is also possible to override any existing method in the filter the same way. This requires defining the same number of arguments as the existing method, noting that *self* in the original does not constitute an argument.

```
def new_method_name(data):
    if data[:3] == 'G71':
        return(None)
    return(data)
self.old_method_name = new_method_name
```

---

**Anmerkung**
The existing filter code may be observed in the file /bin/qtplasmac_gcode.
The file sim/qtplasmac/custom_filter.py has example skeleton code for custom filtering.

---

## 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. External commands
2. External python scripts
3. G-code commands
4. Toggle a HAL pin
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. Load a G-code program
12. Pulse the torch on
13. Single unidirectional cut
14. Framing a job
15. Begin/End a manual cut

16. Display/Hide an offsets viewer
17. Load the latest modified NGC file found in a directory
18. Display/Hide the online HTML user manual

#### 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 = %halshow
```

#### 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 the standard LinuxCNC G-code format. If expressions are included then they need to be surrounded by brackets.

```
n Code = G0 X#<_ini[joint_0]home> Y1
n Code = G53 G0 Z[#<_ini[axis_z]max_limit> - 1.001]
```

<*machine_name*>.prefs file variables and also <*machine_name*>.ini variables can be entered by enclosing each option in **{}** . You must put a space after each **}** if there are any following characters. If expressions are included then they need to be surrounded by brackets.

```
BUTTON_n_CODE = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis}
BUTTON_n_CODE = G0 X{JOINT_0 HOME} Y1
BUTTON_n_CODE = G53 G0 Z[{AXIS_Z MAX_LIMIT} - 1.001]
```

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.

For toggle-halpin buttons, it is possible for the user to mark the associated HAL pin as being required to be turned ”ON” before starting a cut cycle by adding ”runcritical” after the HAL pin in the button code. If **TORCH ENABLE** is checked and **CYCLE START** is pressed while the ”runcritical” button is not ”ON” then the user will receive a dialog warning them as such and asking to CONTINUE or CANCEL.

```
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 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis} \ 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. At least one of the X or Y coordinates are required, Feed Rate (F) is optional.

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.

Feed Rate (F) is optional, if it is missing or invalid then the feed rate of the current material 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 will return to the original coordinates and the program will resume.
3. Press **CYCLE STOP** - the torch will 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 - 5 mm (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
```

**User Manual**

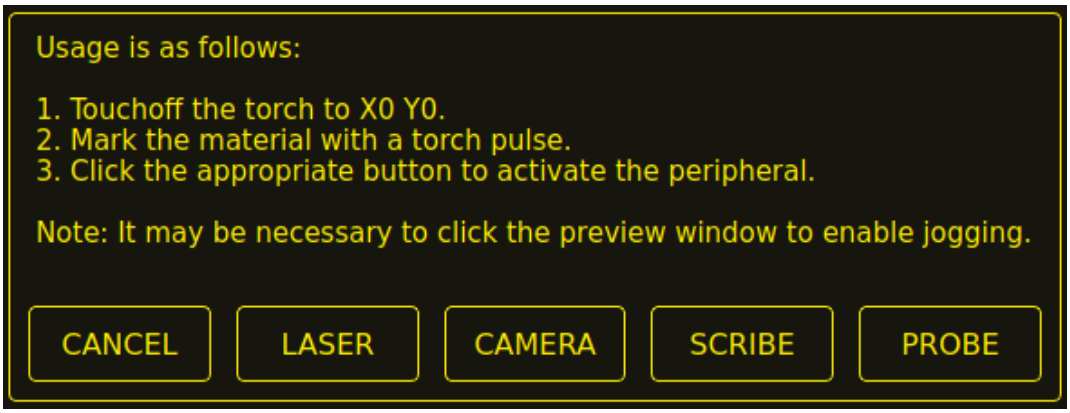
This allows the showing/hiding of the online HTML user manual specific to the version of LinuxCNC currently running. Note that internet access is required for this functionality.

```
n Code = user-manual
```

#### 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.



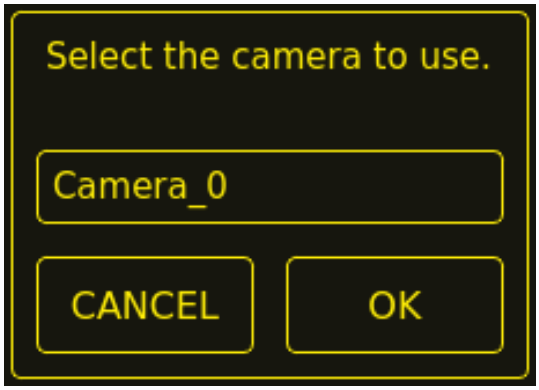

5. Click the X0Y0 button to set the torch position to zero.
6. Make a mark on the material by one of:
   a. Jog the torch down to pierce height then pulse the torch on to make a dimple in the material.
   b. Place marking dye on the torch shield then jog the torch down to mark the material.
7. Click the appropriate button to activate the peripheral.
8. The Get Peripheral Offsets dialog will now be showing.

9. Raise the Z axis so the torch and peripheral are clear of the material.
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.

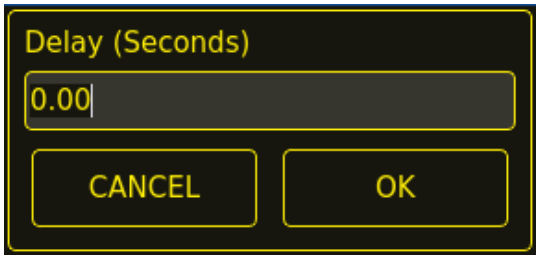
12. Click SET OFFSETS and the offsets will now be saved.

Canceling may be done at any stage by pressing the CANCEL button which will close the dialog and no changes will be saved.

If CAMERA was selected at item 7 above and more than one camera exists then a camera selection dialog will show. The appropriate camera needs to be selected before the Get Peripheral Offsets dialog will appear.



If PROBE was selected at item 7 above then a delay dialog will show prior to the confirmation dialog at item 11. This is for the delay required for the probe to deploy to its working position.




**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` | X0Y0 |
| Use a laser to set an origin with or without rotation | `qtplasmac.ext_laser_touch` | LASER |
| Toggle qtplasmac.laser_on pin | `qtplasmac.ext_laser_toggle` | LASER |
| Run/Pause/Resume the loaded G-code program | `qtplasmac.ext_run_pause` | CYCLE START, CYCLE PAUSE, CYCLE RESUME in sequence |

| User Button Function | HAL Pin | GUI Function |
|---|---|---|
| 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 | CORNERLOCK ANTI DIVE ENABLE |
| Voidlock-Freigabe umschalten | qtplasmac.ext_voidlock_enable | VOIDLOCK 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 |
| 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 |

The following HAL bit pins are only created if the function is specified in a custom user button. The HAL pin has the identical behaviour of the related custom user 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 |

The following HAL bit output pins are always created and can be used by either the Toggle HAL Pin

or Pulse HAL Pin custom user buttons to change the state of an output.

| 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 options 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 5 mm 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 E-stop. 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 option to **True** in the **[GUI_OPTIONS]** of the <*machine_name*>.prefs file in the <*machine_name*> folder:
`Override jog inhibit via 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.

Tabelle 10.23: Different states QtPlasmaC could encounter

| 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 |

Tabelle 10.23: (continued)

| Zustand | Name | Beschreibung |
|---|---|---|
| 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.

Die PowerMax-Maschine wechselt nach dem Aufbau der Kommunikation in den Remote-Modus und kann zu diesem Zeitpunkt nur ferngesteuert werden (über die QtPlasmaC-GUI). Die Verbindung kann durch Beobachtung des PowerMax-Displays validiert werden.

Um den PowerMax wieder in den lokalen Modus zu schalten, kann der Benutzer entweder:

1. Disable PowerMax Comms from the MAIN Tab
2. LinuxCNC Schließen, wodurch der PowerMax während des Herunterfahrens in den lokalen Modus versetzt wird.
3. Den PowerMax für 30 Sekunden aus- und dann wieder einschalten.

---

**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.

---

Um pyserial zu installieren, geben Sie den folgenden Befehl in ein Terminalfenster ein:

```
sudo apt install python3-serial
```

A typical connection diagram is shown in the appendix of this document as well as confirmed working interfaces.

### 10.8.16 Internationalisierung

Es ist möglich, Übersetzungsdateien für QtPlasmaC zu erstellen, um sie in der Sprache des aktuellen Gebietsschemas anzuzeigen.

Um eine Übersetzungsdatei zu erstellen und oder zu bearbeiten, muss LinuxCNC installiert sein und vor Ort ausgeführt werden.

Im Folgenden wird davon ausgegangen, dass das LinuxCNC git-Verzeichnis ~/linuxcnc-dev ist.

Alle Sprachdateien werden in ~/linuxcnc-dev/share/screens/qtplasmac/languages gespeichert.

The qtplasmac.py file is a Python version of the GUI file used for translation purposes.

Die .ts-Dateien sind die Quelldateien für die Übersetzungen. Dies sind die Dateien, die für jede Sprache erstellt/bearbeitet werden müssen.

Die .qm-Dateien sind die kompilierten Übersetzungsdateien, die von pyqt verwendet werden.

The language is determined by an underscore plus the first two letters of the locale, for example if an Italian translation was being done then it would be `_it`. It will be referred to as `_xx` in this document, so `qtplasmac_xx.ts` in this document would actually be `qtplasmac_it.ts` for an Italian translation.

The default locale for QtPlasmaC is `_en` which means that any translation files created as `qtplasmac_en.*` will not be used for translations.

If any of the required utilities (pyuic5, pylupdate5, linguist) are not installed then the user will need to install the required development tools:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Wechseln Sie in das Sprachenverzeichnis:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtplasmac/languages
```

If any text changes have been made to the GUI then run the following to update the GUI Python file:

```
pyuic5 ../qtplasmac.ui > qtplasmac.py
```

Der Benutzer kann entweder eine neue Übersetzungsquelldatei für eine nicht existierende Sprachübersetzung erstellen oder eine existierende Übersetzungsquelldatei modifizieren, wenn ein Text in einer QtPlasmaC-Quelldatei geändert wurde. Wenn Sie eine bestehende Übersetzung modifizieren, die keine Änderungen in der Quelldatei erfahren hat, ist dieser Schritt nicht erforderlich.

Erstellen oder bearbeiten Sie eine .ts-Datei:

```
./langfile xx
```

---

**Anmerkung**
Dieser Befehl ist ein Skript, das Folgendes ausführt: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

---

Die Bearbeitung der Übersetzung erfolgt mit der Anwendung Linguist:

```
linguist
```

1. Open the TS file and translate the strings

Es ist nicht notwendig, für jede Textzeichenfolge eine Übersetzung bereitzustellen. Wenn für eine Zeichenfolge keine Übersetzung angegeben ist, wird die ursprüngliche Zeichenfolge in der Anwendung verwendet. Der Benutzer muss auf die Länge der Zeichenketten achten, die in den Widgets erscheinen, da der Platz begrenzt ist. Wenn möglich, sollte die Übersetzung nicht länger als das Original sein.

When editing is complete save the file:
`File -> Save`

Then create the .qm file:
`File -> Release`

Dann erstellen Sie Links zu der kompilierten .qm Datei für die anderen QtPlasmaC GUIs.

```
$ ln -s qtplasmac_en.qm ../../qtplasmac_4x3/languages/
$ ln -s qtplasmac_en.qm ../../qtplasmac_9x16/languages/
```

QtPlasmaC wird beim nächsten Start in die Sprache des aktuellen Gebietsschemas übersetzt, solange eine .qm Datei in dieser Sprache existiert.

Users are welcome to submit translation files for inclusion into QtPlasmac. The preferred method is to submit a pull request from the users GitHub account as described in the contributing to LinuxCNC documentation. The only files required to be committed are qtplasmac_xx.ts and qtplasmac_xx.qm.

## 10.8.17 Anhang

### 10.8.17.1 Beispielkonfigurationen

Es gibt Beispielkonfigurationsdateien, um mit der QtPlasmaC-GUI Plasmaschneidmaschinen zu simulieren.

They can be found in the LinuxCNC chooser under: Sample Configurations -> sim -> qtplasmac

Drei Versionen sind sowohl in metrischen als auch in imperialen Einheiten erhältlich:

1. qtplasmac_l - 16:9-Format, Mindestauflösung 1366x768
2. qtplasmac_p - 9:16-Format, Mindestauflösung 786x1366
3. qtplasmac_s - 4:3-Format, Mindestauflösung 1024x768

Jede Beispielkonfiguration enthält ein Popup-Bedienfeld, mit dem verschiedene Eingaben in die grafische Benutzeroberfläche simuliert werden können, z. B:

1. LICHTBOGENSPANNUNG (engl. arc voltage)
2. OHMIC SENSE
3. FLOAT SWITCH
4. BREAKAWAY SWITCH
5. ESTOP (engl. für Notaus)

### 10.8.17.2 NGC Beispiele

Im Verzeichnis ~/linuxcnc/nc_files/examples/plasmac befinden sich einige Beispiel-G-Code-Dateien.

### 10.8.17.3 QtPlasmaC-spezifische 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`*<br>*`n`* `denotes the material number.` |

| Beschreibung | Code |
|---|---|
| Wait for material change confirmation. | `M66 P3 L3 Q`*n* + *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)<br>`M64 P1` (immediate) |
| Disable Ignore Arc OK | `M63 P1` (synchronized with motion)<br>`M65 P1` (immediate) |
| Disable THC | `M62 P2` (synchronized with motion)<br>`M64 P2` (immediate) |
| Enable THC | `M63 P2` (synchronized with motion)<br>`M65 P2` (immediate) |
| Disable Torch | `M62 P3` (synchronized with motion)<br>`M64 P3` (immediate) |
| Enable Torch | `M63 P3` (synchronized with motion)<br>`M65 P3` (immediate) |
| Set velocity to a percentage of feed rate. | `M67 E3 Q`*n* (synchronized with motion)<br>`M68 E3 Q`*n* (immediate)<br>*n* is the percentage to set<br>10 is the minimum, below this will be set to 100%<br>100 is the maximum, above this will be set to 100%<br>**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]>` |
| Cutter compensation - right of path | `G42.1 D#<_hal[plasmac.kerf-width]>` |
| Cutter compensation off | `G40`<br>**Note that M62 through M68 are invalid while cutter compensation is on.** |
| Cut holes at 60% feed rate | `#<holes> = 1`<br>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`<br>for holes less than 32 mm (1.26”) diameter<br>over cut length = 4 mm (0.157”) |
| Cut holes and arcs at 60% feed rate. | `#<holes> = 3`<br>for holes less than 32 mm (1.26”) diameter<br>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`<br>for holes less than 32 mm (1.26”) diameter<br>for arcs less than 16 mm (0.63”) radius<br>over cut length = 4 mm (0.157”) |
| Specify hole diameter for #<holes> = 1-4. | `#<h_diameter> =` *n*<br>(*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*<br>(*n* is the percentage, set the percentage of the current feed rate) |
| Specify over cut length. | `#<oclength> =` *n*<br>(*n* is the length, use the same units system as the rest of the G-code file) |
| Specify pierce-only mode. | `#<pierce-only> =` *n*<br>(*n* is the mode, 0=normal cut mode, 1=pierce only mode) |

| Beschreibung | Code |
|---|---|
| Create or edit materials.<br>Options:<br>0 - Create temporary default<br>1 - Add if not existing<br>2 - Overwrite if existing else add new | mandatory parameters:<br>(o=<*option*>, nu=<*nn*>, na=<*ll*>, ph=<*nn*>, pd=<*nn*>, ch=<*nn*>, fr=<*nn*>)<br>optional parameters:<br>(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`<br>`M66 P3 L3 Q1`<br>`F#<_hal[plasmac.cut-feed-rate]>`<br>`M3 $0 S1`<br>`.`<br>`.`<br>`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` |
| Schneiden Sie ein Loch mit 60% reduzierter Geschwindigkeit mit der Geschwindigkeitseinstellung | `G21 (metric)`<br>`G64 P0.05`<br>`M52 P1 (allow paused motion)`<br>`F#<_hal[plasmac.cut-feed-rate]>`<br>`G0 X10 Y10`<br>`M3 $0 S1 (start cut)`<br>`G1 X0`<br>`M67 E3 Q60 (reduce feed rate to 60%)`<br>`G3 I10 (the hole)`<br>`M67 E3 Q100 (restore feed rate to 100%)`<br>`M5 $0 (end cut)`<br>`G0 X0 Y0`<br>`M2 (end job)` |
| Schneiden Sie ein Loch mit 60% reduzierter Geschwindigkeit mit dem Befehl #<holes> | `G21 (metric)`<br>`G64 P0.05`<br>`M52 P1 (allow paused motion)`<br>`#<holes> = 1 (velocity reduction for holes)`<br>`F#<_hal[plasmac.cut-feed-rate]>`<br>`G0 X10 Y10`<br>`M3 $0 S1 (start cut)`<br>`G1 X0`<br>`G3 I10 (the hole)`<br>`M5 $0 (end cut)`<br>`G0 X0 Y0`<br>`M2 (end job)` |

| Beschreibung | Beispiel |
|---|---|
| Schneiden Sie ein Loch mit Überschnitt mit Brenner deaktivieren | G21 (metric)<br>G64 P0.05<br>M52 P1 (allow paused motion)<br>F#<_hal[plasmac.cut-feed-rate]><br>G0 X10 Y10<br>M3 $0 S1 (start cut)<br>G1 X0<br>M67 E3 Q60 (reduce feed rate to 60%)<br>G3 I10 (the hole)<br>M62 P3 (turn torch off)<br>G3 X0.8 Y6.081 I10 (continue motion for 4 mm)<br>M63 P3 (allow torch to be turned on)<br>M67 E3 Q0 (restore feed rate to 100%)<br>M5 $0 (end cut)<br>G0 X0 Y0<br>M2 (end job) |
| Schneiden Sie ein Loch mit Überschnitt mit dem Befehl #<holes> | G21 (metric)<br>G64 P0.05<br>M52 P1 (allow paused motion)<br>#<holes> = 2 (over cut for holes)<br>F#<_hal[plasmac.cut-feed-rate]><br>G0 X10 Y10<br>M3 $0 S1 (start cut)<br>G1 X0<br>G3 I10 (the hole)<br>M5 $0 (end cut)<br>G0 X0 Y0<br>M2 (end job) |
| Cut a hole with 6.5 mm over cut using the #<holes> command | G21 (metric)<br>G64 P0.05<br>M52 P1 (allow paused motion)<br>#<holes> = 2 (over cut for holes)<br><oclength> = 6.5 (6.5 mm over cut length)<br>F<_hal[plasmac.cut-feed-rate]><br>G0 X10 Y10<br>M3 $0 S1 (start cut)<br>G1 X0<br>G3 I10 (the hole)<br>M5 $0 (end cut)<br>G0 X0 Y0<br>M2 (end job) |
| Wählen Sie Ritzer/Schreiber (engl. scribe) und wählen Sie den Brenner am Ende des Ritzens | .<br>.<br>M52 P1 (paused motion on)<br>F#<_hal[plasmac.cut-feed-rate]><br>T1 M6 (select scribe)<br>G43 H0 (apply offsets)<br>M3 $1 S1 (start plasmac with scribe)<br>.<br>.<br>T0 M6 (select torch)<br>G43 H0 (apply offsets)<br>G0 X0 Y0 (parking position)<br>M5 $1 (end) |

| Beschreibung | Beispiel |
|---|---|
| Hole center spotting. | (Erfordert einen kleinen Bewegungsbefehl, sonst passiert nichts)<br>G21 (metrisch)<br>F99999 (hohe Vorschubgeschwindigkeit)<br>G0 X10 Y10<br>M3 $2 S1 (spotting ein)<br>G91 (relativer Abstandsmodus)<br>G1 X0.000001<br>G90 (absoluter Abstandsmodus)<br>M5 $2 (spotting aus)<br>G0 X0 Y0<br>G90<br>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) |
| Material bearbeiten, falls nicht vorhanden, ein neues Material anlegen | (o=2, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=1.0) |

#### 10.8.17.5 Mesa THCAD

Das Mesa THCAD ist eine gängige Methode, um die Lichtbogenspannung von einem Plasmaschneider zu erhalten, und ist auch für die ohmsche Messung des Materials während des Abtastens nützlich. Das THCAD kann sowohl für Konfigurationen mit parallelen Anschlüssen als auch für Konfigurationen mit Mesa Electronics-Hardware verwendet werden. Das THCAD ist in drei verschiedenen Modellen erhältlich: THCAD-5, THCAD-10 und THCAD-300.

Auf jeder THCAD-Karte befindet sich ein Mode-Jumper, der auf **UNIPOLAR** gesetzt werden sollte.

Auf jeder THCAD-Karte befindet sich ein Frequenzteiler-Jumper, der je nach Hardware-Typ gesetzt werden sollte:

| Input Device | Recommended Setting |
|---|---|
| Parallele Schnittstelle mit sehr geringer Latenzzeit | F/32 |
| Parallele Schnittstelle empfohlener Startpunkt | F/64 |
| Paralleler Anschluss mit höherer Latenzzeit oder beim Schneiden von dickem Material | F/128 |
| Mesa 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.

Auf der Rückseite des THCADs befindet sich ein Kalibrierungsaufkleber:

```
   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
```

$r$ = Teilerverhältnis (siehe unten).

$f$ = Skalenendwert vom Kalibrierungsaufkleber.

$z$ = 0 V value from calibration sticker.

$d$ = Wert von Jumper oben.

$v$ = full scale voltage of THCAD

**Teiler-Verhältnis** *THCAD-5 oder 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**
WENN DER BENUTZER EINE HF-STARTPLASMA-STROMVERSORGUNG VERWENDET, SOLLTE JEDER DIESER WIDERSTÄNDE AUS MEHREREN HOCHSPANNUNGSWIDERSTÄNDEN BESTEHEN.

**Achtung**
WENN DER BENUTZER EINE HF-STARTPLASMA-STROMVERSORGUNG VERWENDET, WIRD EINE OHMSCHE ABTASTUNG NICHT EMPFOHLEN.

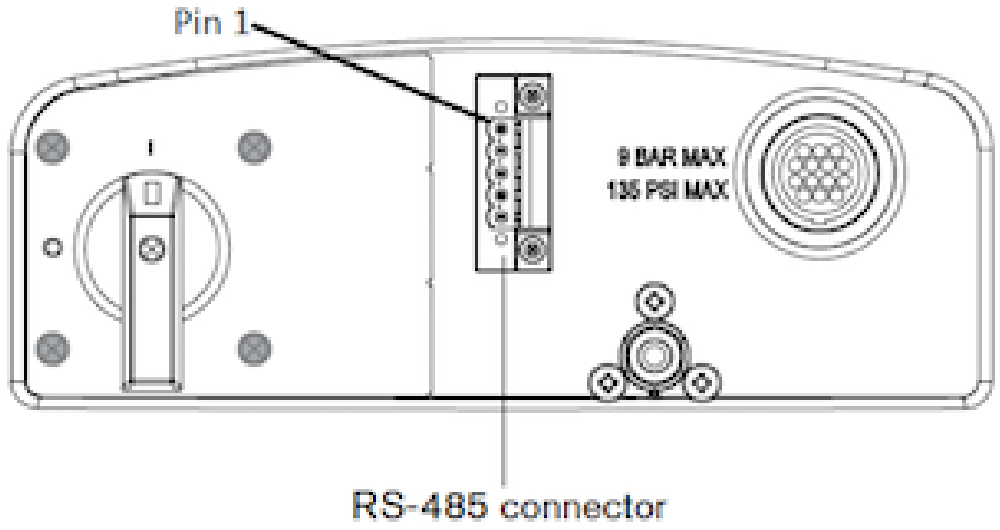
**Anmerkung**
Diese Werte können mit diesem Online-Rechner berechnet werden.

**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-Verbindungen

Hypertherm RS485 Verdrahtungsplan (Drahtfarben innerhalb des Hypertherm in Klammern):

| Connection at Machine Pin # | Connection at Breakout Board |
|---|---|
| **1** - Tx+ (Rot) | ->RXD+ |
| **2** - Tx- (schwarz) | ->RXD- |
| **3** - Rx+ (braun) | ->T/R+ |
| **4** - Rx- (weiß) | ->T/R- |
| **5** - GND (grün) | ->GND |



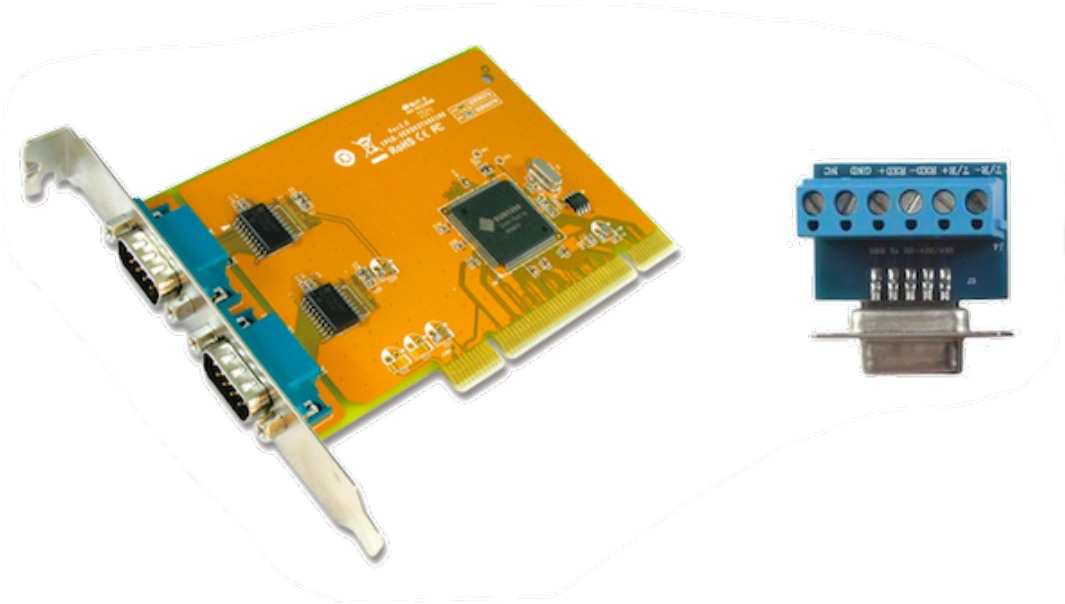

RS485-Schnittstellen, von denen bekannt ist, dass sie funktionieren:

DTECH DT-5019 USB zu RS-485 Konverter Adapter:

Um eine serielle Verbindung der Hauptplatine oder eine serielle Karte (RS232) in RS485 umzuwandeln, sind folgende Schritte erforderlich:

DTECH RS-232 zu RS-485 Konverter:

Beispiel einer seriellen Karte (Sunnix SER5037A PCI-Karte mit Breakout Board):

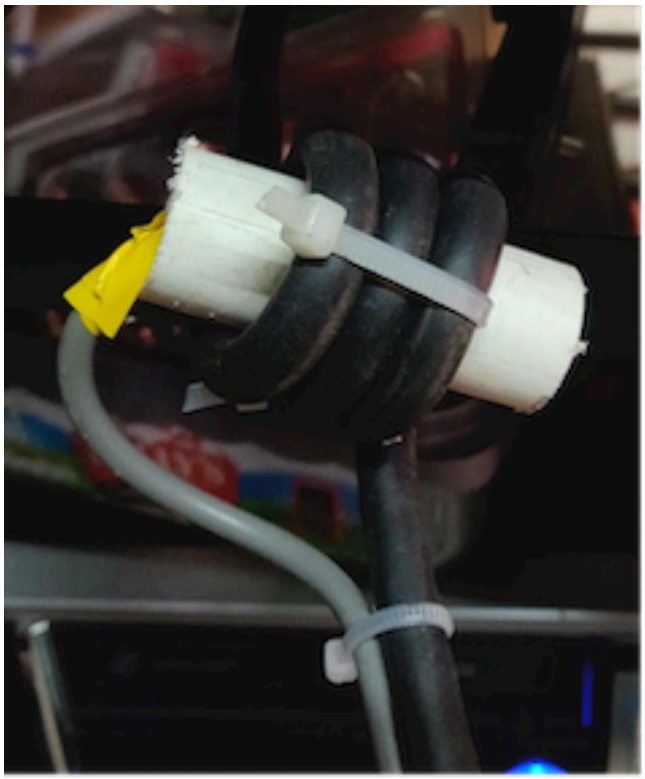


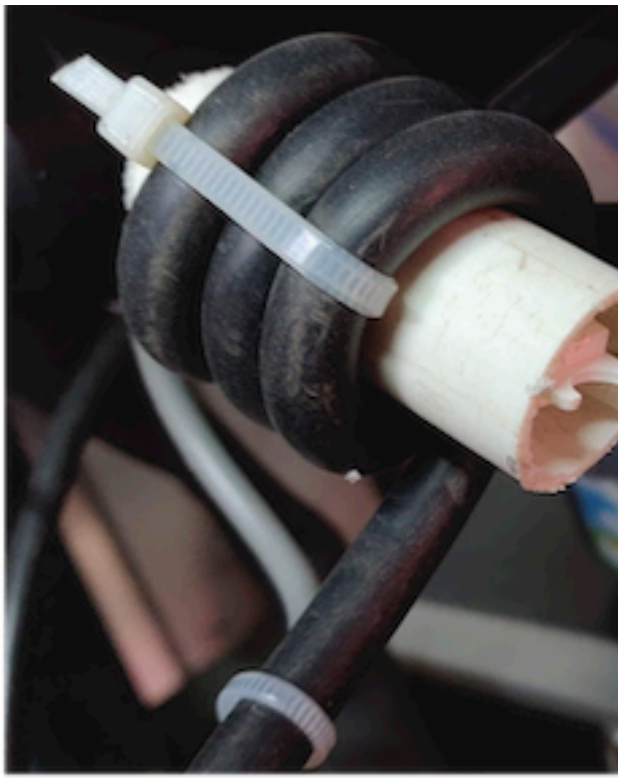
### 10.8.17.7 Lichtbogen OK mit einem Reed-Relais

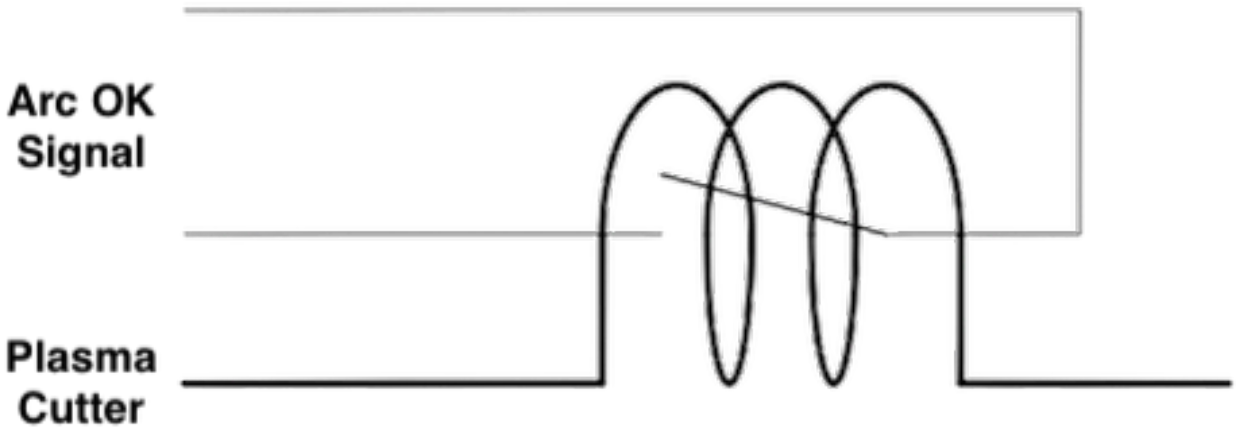
Eine effektive und sehr zuverlässige Methode, um ein Lichtbogen-OK-Signal von einer Plasmaversorgung ohne CNC-Anschluss zu erhalten, besteht darin, ein Reed-Relais in einer nicht leitenden Röhre zu montieren und drei Windungen des Arbeitskabels um die Röhre zu wickeln und zu sichern.

Diese Baugruppe fungiert nun als Relais, das sich einschaltet, wenn Strom durch die Arbeitsleitung fließt, was nur dann der Fall ist, wenn sich ein Lichtbogen gebildet hat.

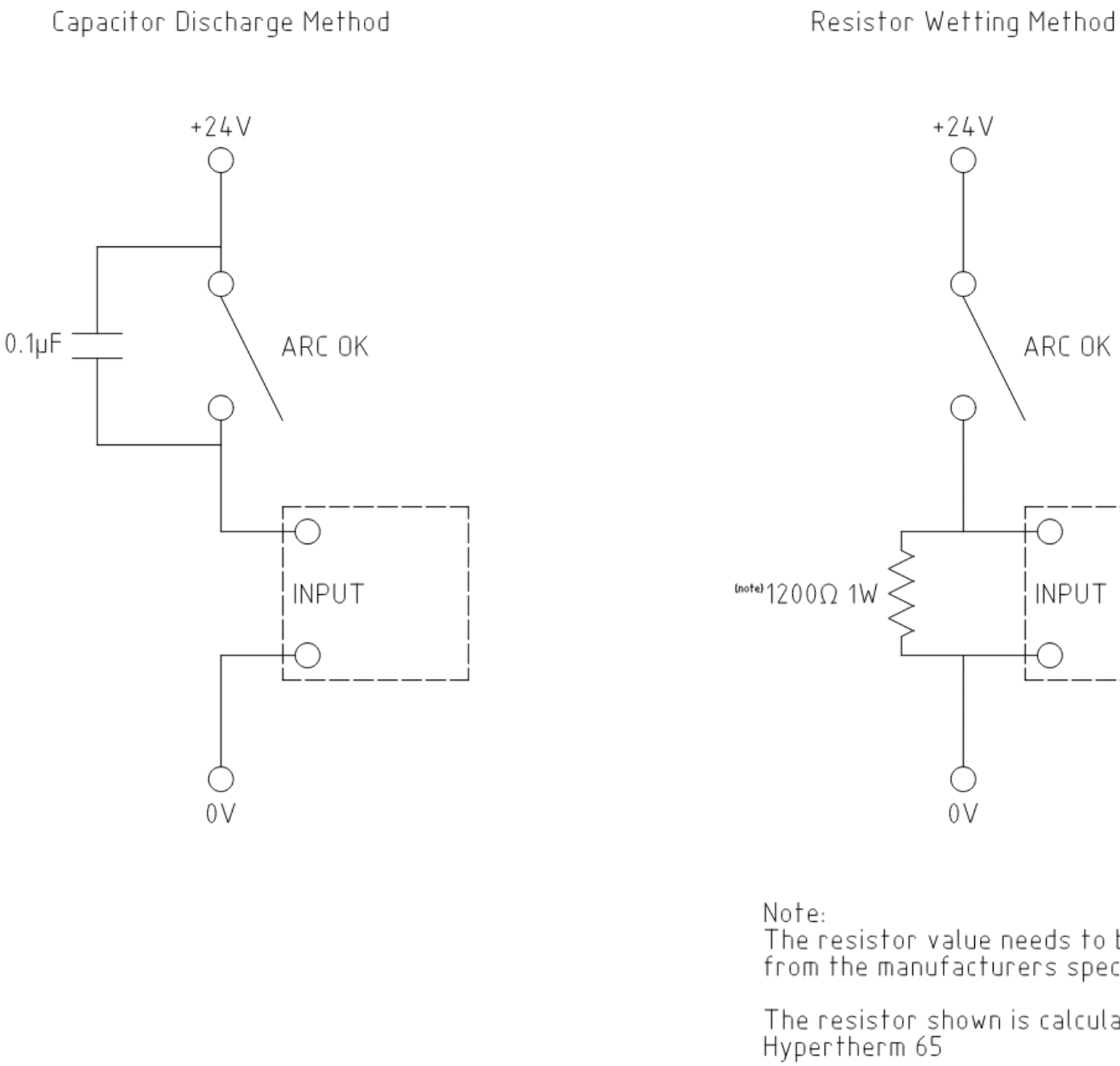
This will require that QtPlasmaC be operated in Mode 1 rather than Mode 0. See the QtPlasmaC Modes sections for more information.






Arc OK
Signal
Plasma
Cutter

#### 10.8.17.8 Schematische Darstellung der Kontaktbelastung




A full description is at Contact Load.

### 10.8.18 Bekannte Probleme

#### 10.8.18.1 Tastatur-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 option 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 is a simple graphical front-end for LinuxCNC providing a display of data from Digital Read Out (DRO) scales. It provides functionality similar to a normal machinist's DRO display, allowing the user to use the DRO scales on the machine when operating in a manual-only (hand-cranked) mode. It is most useful for manual machines such as DRO equipped Bridgeport style mills that have been converted to CNC but still have the manual controls.

MDRO ist Maus- und Touchscreen-freundlich.

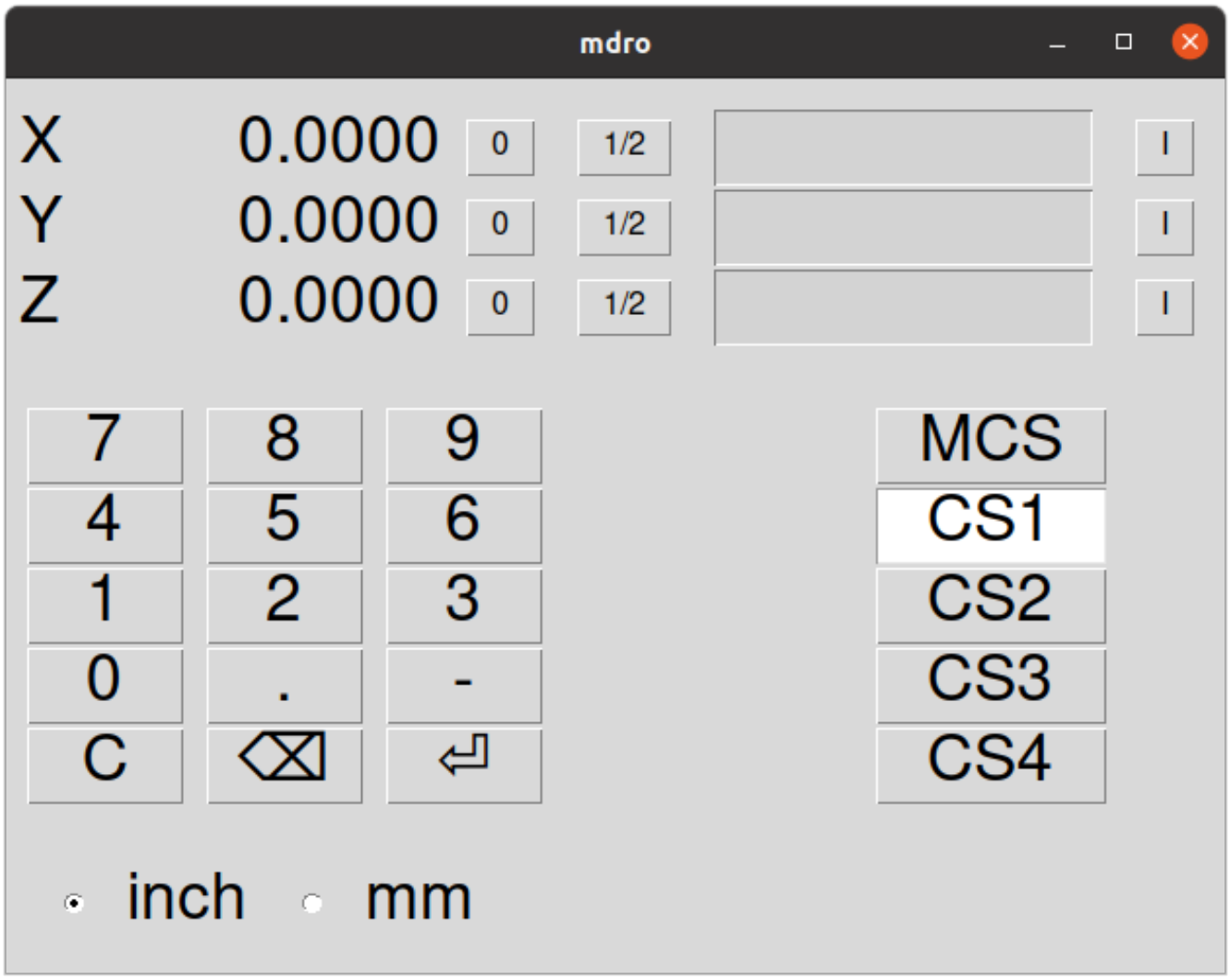



Abbildung 10.50: MDRO Fenster

### 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 10.50 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 this chapter, we will try to demystify coordinate systems. It is a very important concept to understand the operation of a CNC machine, its configuration and its use.

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

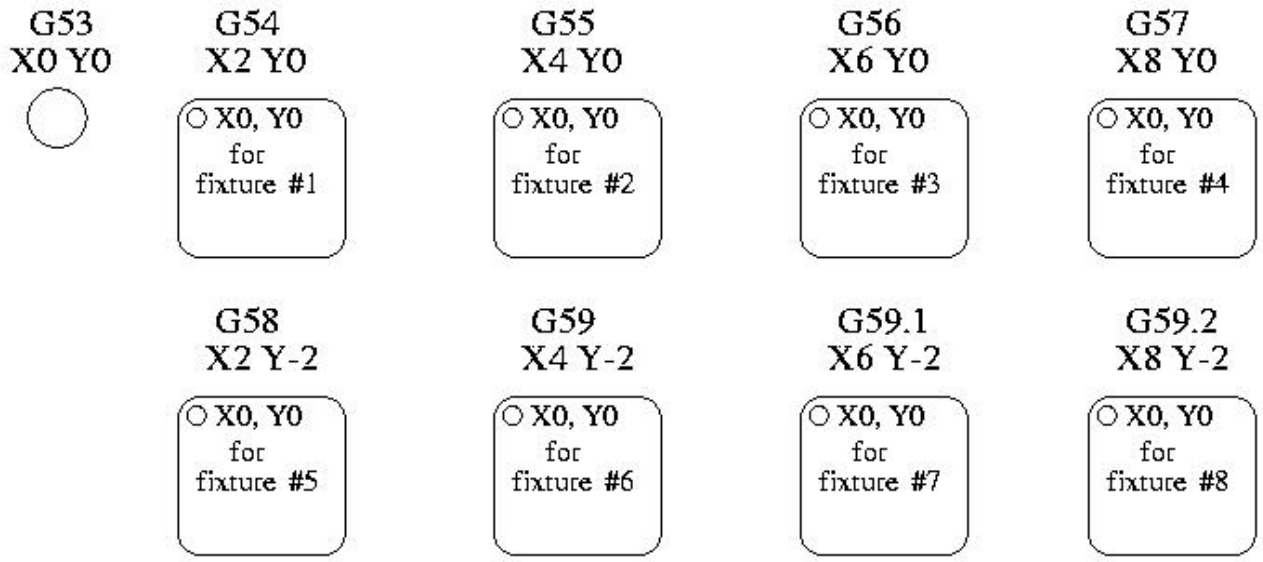



Abbildung 11.1: Beispiel für Koordinatensysteme

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 Tool Compensation

### 11.2.1 Touch-Off

Using the Touch Off Screen in the AXIS interface you can update the tool table automatically.

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.

The Tool Table will be changed with the correct Z length to make the DRO display the correct Z position and a G43 command will be issued so the new tool Z length will be in effect. Tool table touch off is only available when a tool is loaded with *T*n *M6*.

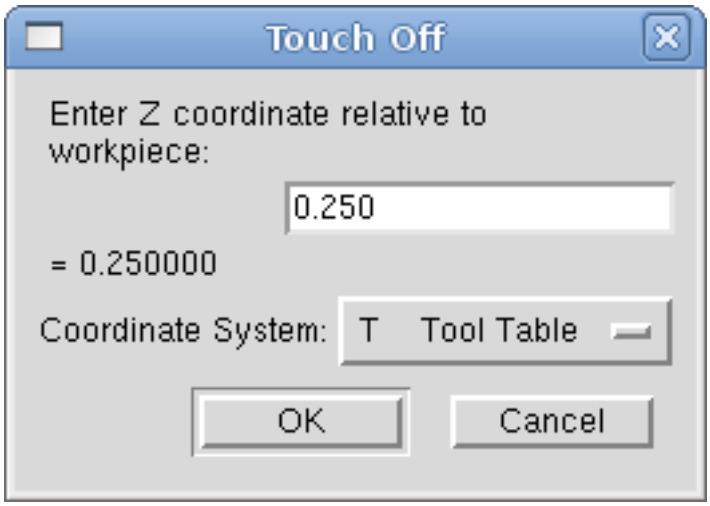



Abbildung 11.2: Touch-Off-Werkzeugtabelle

#### 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__` - Set offset(s) to a value. Current position irrelevant (see G10 L1 for details).
- `G10 L10 P__n__` - Set offset(s) so current position w/ fixture 1-8 becomes a value (see G10 L10 for details).
- `G10 L11 P__n__` - Set offset(s) so current position w/ fixture 9 becomes a value (see G10 L11 for details).

---

**Anmerkung**
Dies ist nur eine kurze Darstellung, genauere Erläuterungen finden Sie im Referenzhandbuch des G-Codes.

---

### 11.2.2 Werkzeugtabelle

The *Tool Table* is a text file that contains information about each tool. The file is located in the same directory as your configuration and is called *tool.tbl* by default. A file name may be specified with the INI file [EMCIO]TOOL_TABLE setting. The tools might be in a tool changer or just changed manually. The file can be edited with a text editor or be updated using G10 L1. See the Lathe Tool Table section for an example of the lathe tool table format. The maximum pocket number is 1000.

The Tool Editor or a text editor can be used to edit the tool table. If you use a text editor make sure you reload the tool table in the GUI.

#### 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.

One line may contain as many as 16 entries, but will likely contain much fewer. The entries for T (tool number) and P (pocket number) are required. The last entry (a remark or comment, preceded by a semicolon) is optional. It makes reading easier if the entries are arranged in columns, as shown in the table, but the only format requirement is that there be at least one space or tab after each of the entries on a line and a newline character at the end of each entry.

Die Bedeutung der Einträge und die Art der Daten, die sie enthalten, sind wie folgt.

**Werkzeugnummer (erforderlich)**
: The *T* column contains the number (unsigned integer) which represents a code number for the tool. The user may use any code for any tool, as long as the codes are unsigned integers.

**Taschen-Nummer (erforderlich)**
: The *P* column contains the number (unsigned integer) which represents the pocket number (slot number) of the tool changer slot where the tool can be found. The entries in this column must all be different.

The pocket numbers will typically start at 1 and go up to the highest available pocket on your tool changer. But not all tool changers follow this pattern. Your pocket numbers will be determined by the numbers that your tool changer uses to refer to the pockets. So all this is to say that the pocket numbers you use will be determined by the numbering scheme used in your tool changer, and the pocket numbers you use must make sense on your machine.

**Daten-Offset-Nummern (optional)**
: The *Data Offset* columns (XYZABCUVW) contain real numbers which represent tool offsets in each axis. This number will be used if tool length offsets are being used and this tool is selected. These numbers can be positive, zero, or negative, and are in fact completely optional. Although you will probably want to make at least one entry here, otherwise there would be little point in making an entry in the tool table to begin with.

In a typical mill, you probably want an entry for Z (tool length offset). In a typical lathe, you probably want an entry for X (X tool offset) and Z (Z tool offset). In a typical mill using cutter diameter compensation (cutter comp), you probably also want to add an entry for D (cutter diameter). In a typical lathe using tool nose diameter compensation (tool comp), you probably also want to add an entry for D (tool nose diameter).

A lathe also requires some additional information to describe the shape and orientation of the tool. So you probably want to have entries for I (tool front angle) and J (tool back angle). You probably also want an entry for Q (tool orientation).

Siehe das Kapitel Informationen für Nutzer von Drehmaschinen für weitere Details.

The *Diameter* column contains a real number. This number is used only if cutter compensation is turned on using this tool. If the programmed path during compensation is the edge of the material being cut, this should be a positive real number representing the measured diameter of the tool. If the programmed path during compensation is the path of a tool whose diameter is nominal, this should be a small number (positive or negative, but near zero) representing only the difference between the measured diameter of the tool and the nominal diameter. If cutter compensation is not used with a tool, it does not matter what number is in this column.

The *Comment* column may optionally be used to describe the tool. Any type of description is OK. This column is for the benefit of human readers only. The comment must be preceded by a semicolon.

---

**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 Tool IO

The non-realtime program specified by `[EMCIO]EMCIO = io` is conventionally used for tool changer management (and other io functions for enabling LinuxCNC and the control of coolant/lube hardware). The HAL pins used for tool management are prefixed with `iocontrol.0.`.

A G-code **T** command asserts the HAL output pin `iocontrol.0.tool-prepare`. The HAL input pin, `iocontrol.0.tool-prepared`, must be set by external HAL logic to complete tool preparation leading to a subsequent reset of the tool-prepare pin.

A G-code **M6** command asserts the HAL output pin `iocontrol.0.tool-change`. The related HAL input pin, `iocontrol.0.tool-prepared`, must be set by external HAL logic to indicate completion of the tool change leading to a subsequent reset of the tool-change pin.

Tooldata is accessed by an ordered index (idx) that depends on the type of toolchanger specified by `[EMCIO]RANDOM_TOOLCHANGER=`*type*.

1. For `RANDOM_TOOLCHANGER = 0`, (0 is default and specifies a non-random toolchanger) idx is a number indicating the sequence in which tooldata was loaded.
2. For `RANDOM_TOOLCHANGER = 1`, idx is the **current** pocket number for the tool number specified by the G-code select tool command **T*n***.

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 Tool Changers

LinuxCNC supports three types of tool changers: *manual*, *random location* and *non-random or fixed location*. Information about configuring a LinuxCNC tool changer is in the EMCIO Section of the INI chapter.

**Manueller Werkzeugwechsler** Manual tool changer (you change the tool by hand) is treated like a fixed location tool changer. Manual toolchanges can be aided by a HAL configuration that employs the non-realtime program **hal_manualtoolchange** and is typically specified in an INI file with INI statements:

```
[HAL]
HALFILE = axis_manualtoolchange.hal
```

**Fixed Location Tool Changers** Fixed location tool changers always return the tools to a fixed position in the tool changer. This would also include designs like lathe turrets. When LinuxCNC is configured for a fixed location tool changer the *P* number is not used internally (but read, preserved and rewritten) by LinuxCNC, so you can use P for any bookkeeping number you want.

---

**Anmerkung**
When using `[EMCIO]RANDOM_TOOLCHANGER = 0` (the default), the *P* pocket number is a parameter of the tooldata that is retrieved from the tooldata source (`[EMCIO]TOOL_TABLE` or `[EMCIO]DB_PROGRAM`). In many applications it is fixed but it may be changed by edits to the `[EMCIO]TOOL_TABLE` or programmatically when the `[EMCIO]DB_PROGRAM` is used. LinuxCNC pushes updates to the data source (`[EMCIO]TOOL_TABLE` or `[EMCIO]DB_PROGRAM`) for G-codes G10L1, G10L10, G10L11, M61. LinuxCNC can pull tooldata updates from the data source by UI (user-interface) commands (Python example: `linuxcnc.command().load_tool_table()`) or by the G-code: `G10L0`.

---

**Werkzeugwechsler mit zufälliger Position** Random location tool changers (`[EMCIO]RANDOM_TOOLCHANGER = 1`) swap the tool in the spindle with the one in the changer. With this type of tool changer the tool will always be in a different pocket after a tool change. When a tool is changed LinuxCNC rewrites the pocket number to keep track of where the tools are. T can be any number but P must be a number that makes sense for the machine.

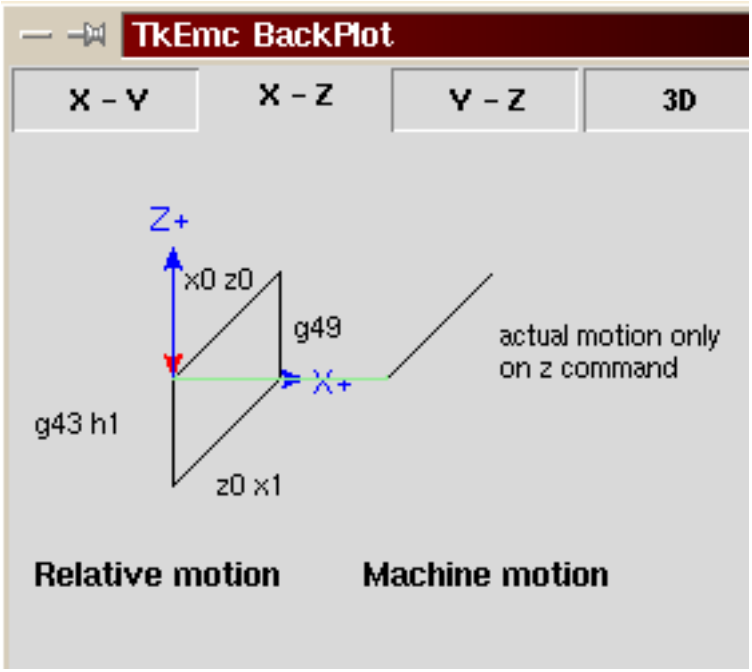
### 11.2.3 Tool Length Compensation

The tool length compensations are given as positive numbers in the tool table. A tool compensation is programmed using G43 H_n_, where *n* is the index number of the desired tool in the tool table. It is intended that all entries in the tool table are positive. The value of H is checked, it must be a non-negative integer when read. The interpreter behaves as follows:

1. If G43 H_n_ is programmed, a call to the function `USE_TOOL_LENGTH_OFFSET( `__length__)` is made (where *length* is the length difference, read from the tool table, of the indexed tool *n*), tool_length_offset is repositioned in the machine settings model and the value of current_z in the model is adjusted. Note that *n* does not have to be the same as the slot number of the tool currently in the spindle.
2. If G49 is programmed, `USE_TOOL_LENGTH_OFFSET(0.0)` is called, tool_length_offset is reset to 0.0 in the machine settings template and the current value of current_z in the model is adjusted. The effect of the tool length compensation is illustrated in the capture below. Note that the tool length is subtracted from Z so that the programmed control point corresponds to the tip of the tool. Note also that the effect of the length compensation is immediate when you see the compensation is immediate when the position of Z is seen as a relative coordinate, but it has no effect on the actual machine position until a Z movement is programmed.

**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 Cutter Radius Compensation

Cutter Compensation allows the programmer to program the tool path without knowing the exact tool diameter. The only caveat is the programmer must program the lead in move to be at least as long as the largest tool radius that might be used.

There are two possible paths the cutter can take since the cutter compensation can be on to the left or right side of a line when facing the direction of cutter motion from behind the cutter. To visualize this imagine you were standing on the part walking behind the tool as it progresses across the part. G41 is your left side of the line and G42 is the right side of the line.

The end point of each move depends on the next move. If the next move creates an outside corner the move will be to the end point of the compensated cut line. If the next move creates in an inside corner the move will stop short so to not gouge the part. The following figure shows how the compensated move will stop at different points depending on the next move.

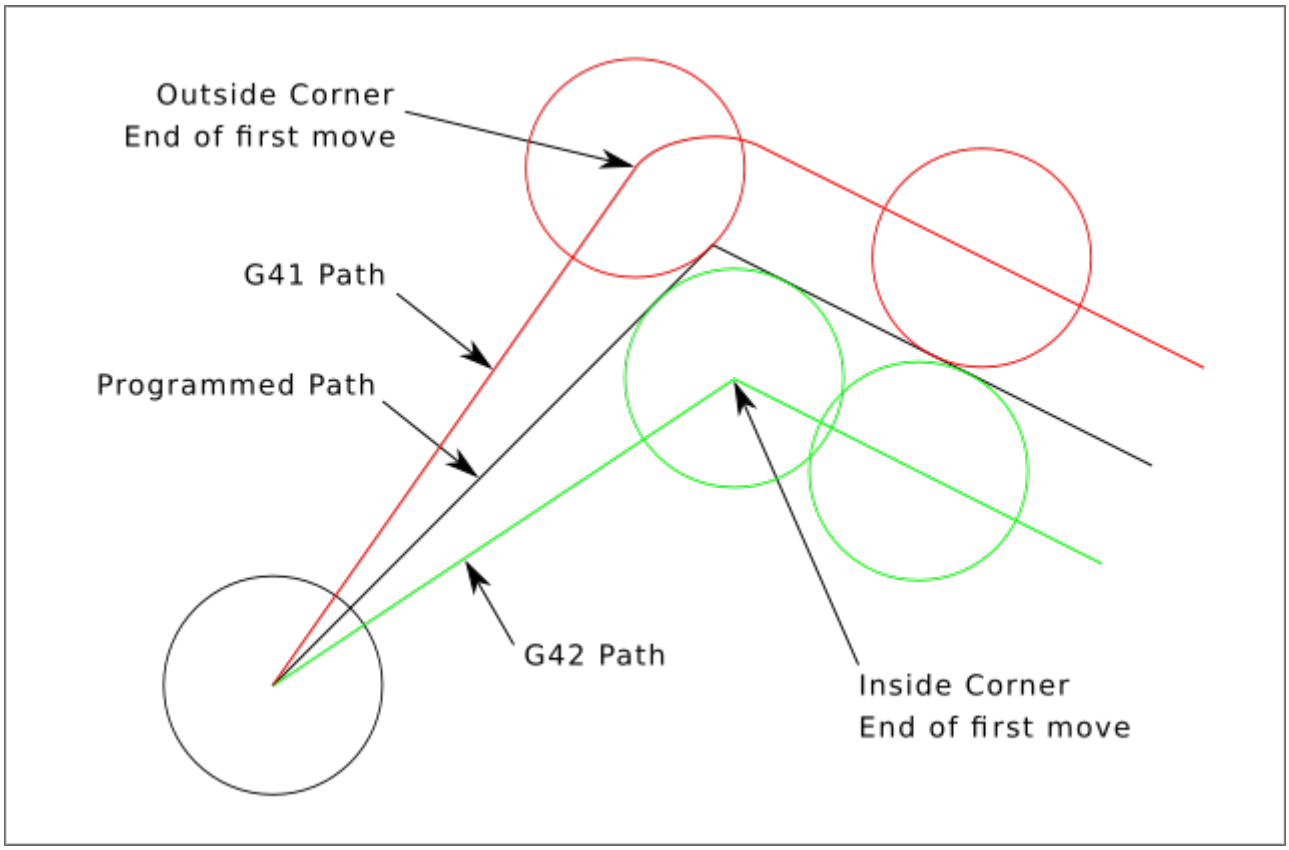



Abbildung 11.3: Ausgleich am Endpunkt

#### 11.2.4.1 Übersicht

Cutter compensation uses the data from the tool table to determine the offset needed. The data can be set at run time with G10 L1.

Any move that is long enough to perform the compensation will work as the entry move. The minimum length is the cutter radius. This can be a rapid move above the work piece. If several rapid moves are issued after a G41/42 only the last one will move the tool to the compensated position.

In the following figure you can see that the entry move is compensated to the right of the line. This puts the center of the tool to the right of X0 in this case. If you were to program a profile and the end is at X0 the resulting profile would leave a bump due to the offset of the entry move.

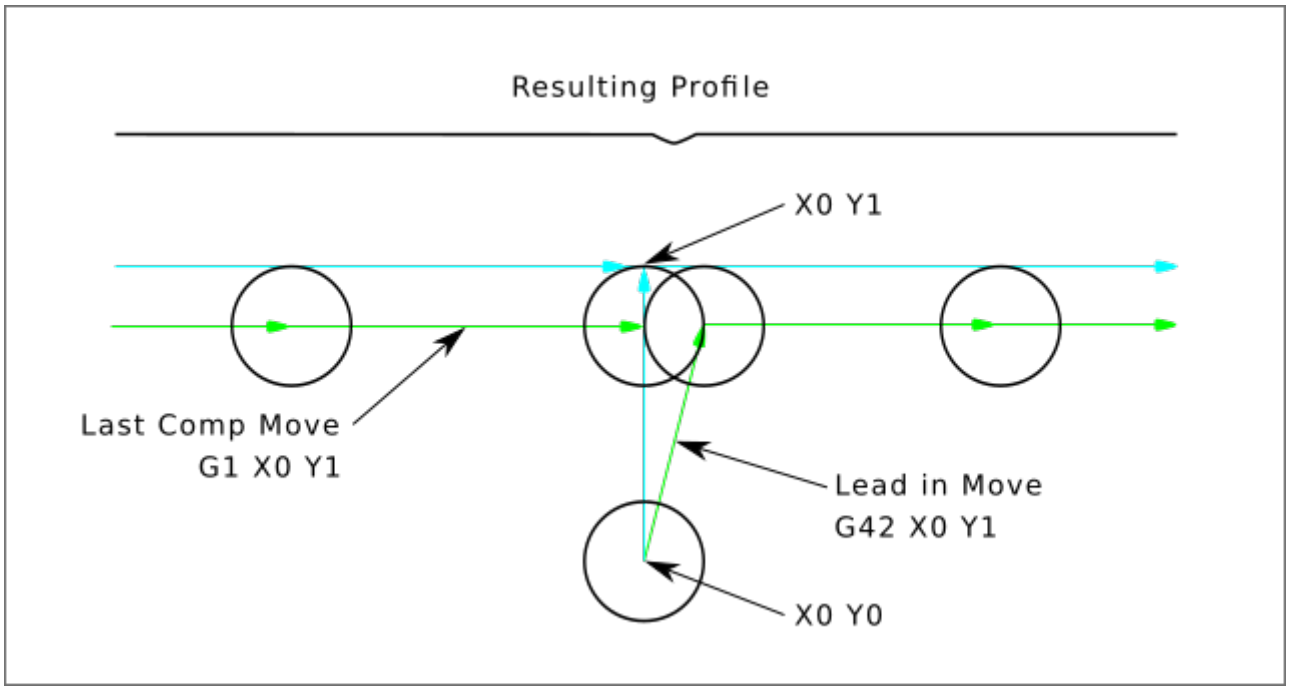



Abbildung 11.4: Eingangs-Bewegung (engl. entry move)

Z axis motion may take place while the contour is being followed in the XY plane. Portions of the contour may be skipped by retracting the Z axis above the part and by extending the Z-axis at the next start point.

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 )
```

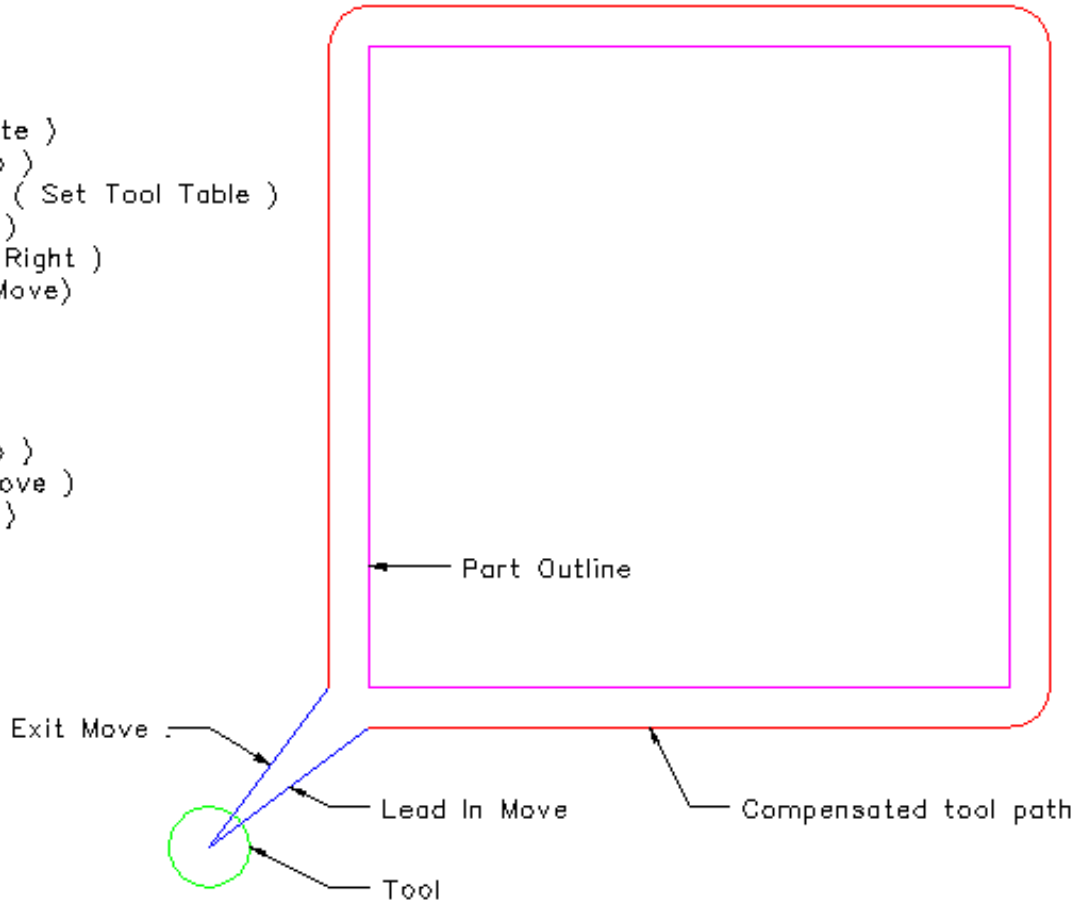



Abbildung 11.5: Äußeres Profil

```
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 )
```

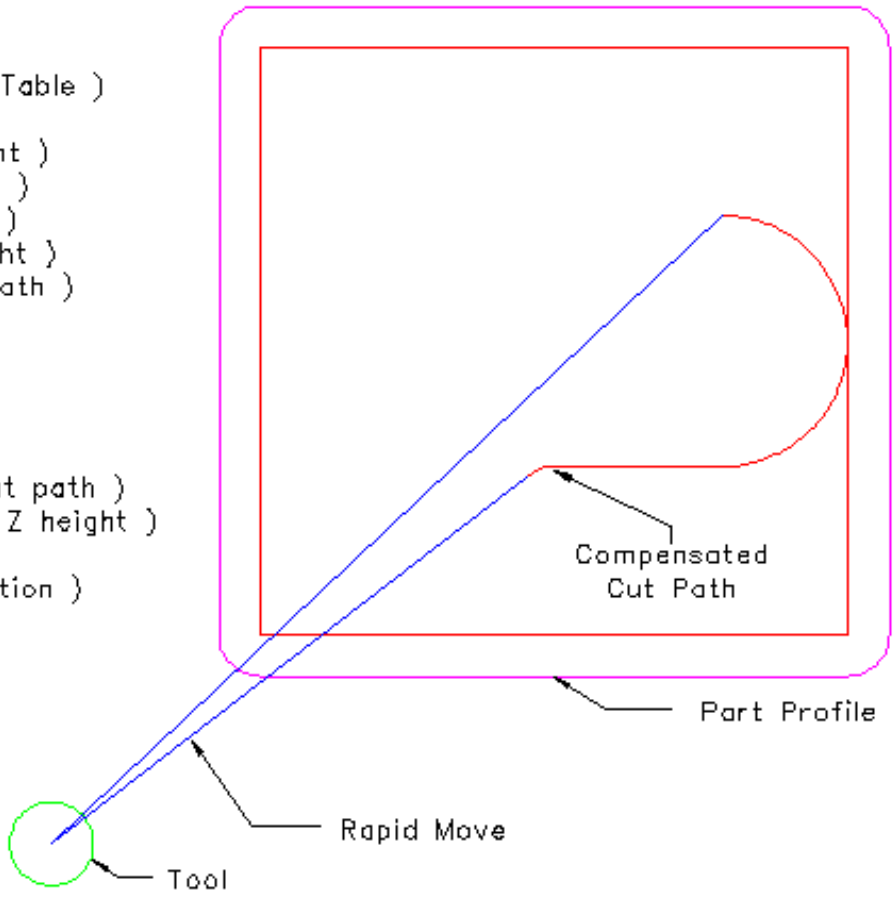



Abbildung 11.6: Innenprofil

## 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.

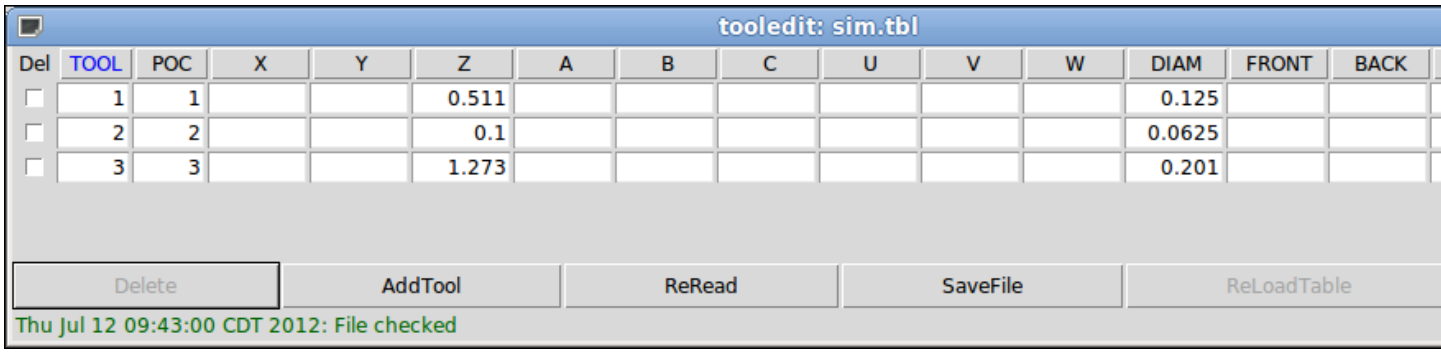



Abbildung 11.7: Tool Edit GUI - Überblick

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.

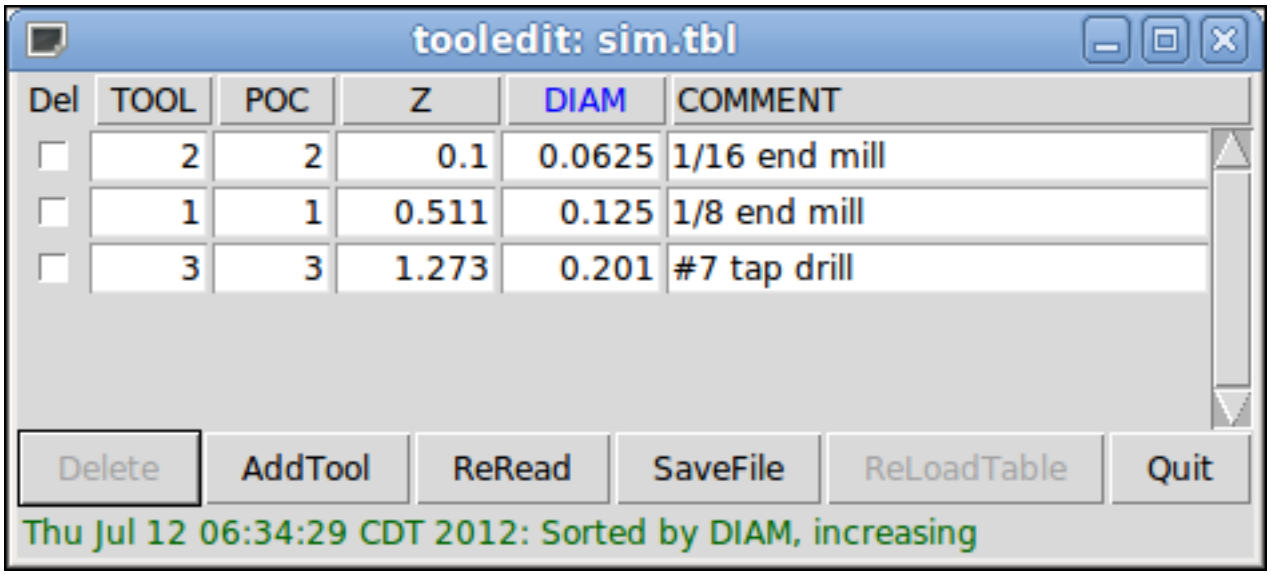



Abbildung 11.8: Tool Edit GUI - Spaltensortierung

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
```

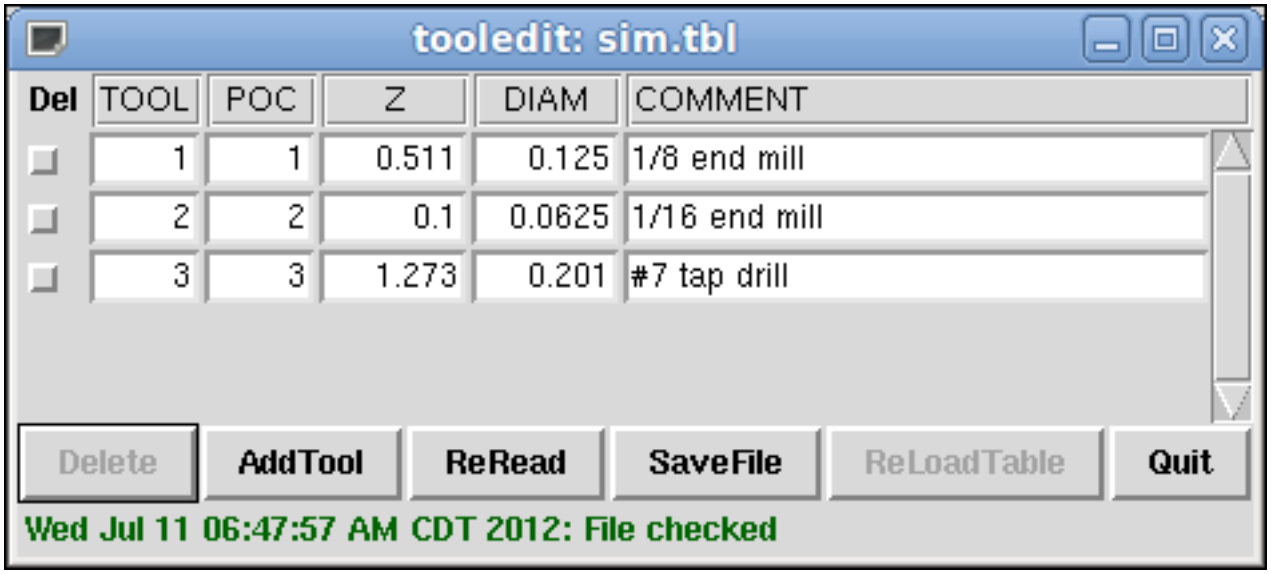



Abbildung 11.9: Tool Edit GUI - Spaltenauswahl Beispiel

### 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

The LinuxCNC G-code language is based on the RS274/NGC language. The G-code language is based on lines of code. Each line (also called a *block*) may include commands to do several different things. Lines of code may be collected in a file to make a program.

A typical line of code consists of an optional line number at the beginning followed by one or more *words*. A word consists of a letter followed by a number (or something that evaluates to a number). A word may either give a command or provide an argument to a command. For example, *G1 X3* is a valid line of code with two words. *G1* is a command meaning *move in a straight line at the programmed feed rate to the programmed end point*, and *X3* provides an argument value (the value of X should be 3 at the end of the move). Most LinuxCNC G-code commands start with either G or M (for General and Miscellaneous). The words for these commands are called *G-codes* and *M-codes*. Also common are subroutine codes that begin with *o-* which are called *o-codes*.

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 Delete

The optional block delete character the slash / when placed first on a line can be used by some user interfaces to skip lines of code when needed. In Axis the key combination Alt-m-/ toggles block delete on and off. When block delete is on any lines starting with the slash / are skipped.

In AXIS ist es auch möglich, das Löschen von Blöcken mit dem folgenden Symbol zu aktivieren:

**AXIS-Block Löschsymbol**

### 11.4.2.2 Line Number

A line number is the letter N followed by an unsigned integer, optionally followed by a period and another unsigned integer. For example, *N1234* and *N56.78* are valid line numbers. They may be repeated or used out of order, although normal practice is to avoid such usage. Line numbers may also be skipped, and that is normal practice. A line number is not required to be used, but must be in the proper place if used.

### 11.4.2.3 Word

A word is a letter other than N followed by a real value.

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 | General function (See table G-code Modal Groups) |
| 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 | Miscellaneous function (See table M-code 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 |

Parameters are identified with a "#" symbol in front of them. See Parameters Section below.

Also called *o-codes* these provide program flow control (such as if-else logic and callable subroutines) and are covered fully at the page on o-Codes and also below in Subroutine Codes and Parameters.

**Anmerkung**
o-codes are sometimes also called o-words.

Comments can be embedded in a line using parentheses () or for the remainder of a line using a semi-colon. There are also *active* comments like MSG, DEBUG, etc. See the section on comments.

#### 11.4.2.4 End of Line Marker

This is any combination of carriage return or line feed.

#### 11.4.2.5 Numbers

The following rules are used for (explicit) numbers. In these rules a digit is a single character between 0 and 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

The RS274/NGC language supports *parameters* - what in other programming languages would be called *variables*. There are several types of parameter of different purpose and appearance, each described in the following sections. The only value type supported by parameters is floating-point; there are no string, boolean or integer types in G-code like in other programming languages. However, logic expressions can be formulated with boolean operators ( *AND*, *OR*, *XOR*, and the comparison operators *EQ*,*NE*,*GT*,*GE*,*LT*,*LE*), and the *MOD*, *ROUND*, *FUP* and *FIX* operators support integer arithmetic.

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 Numbered Parameters

A numbered parameter is the pound character # followed by an integer between 1 and (currently) 5602 [2]. The parameter is referred to by this integer, and its value is whatever number is stored in the parameter.

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.

[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] The RS274/NGC interpreter maintains an array of numbered parameters. Its size is defined by the symbol *RS274NGC_MAX_PARAMETERS* in the file src/emc/rs274ngc/interp_internal.hh). This number of numerical parameters may also increase as development adds support for new parameters.

- *31-5000* - G-Code Benutzerparameter. Diese Parameter sind global in der G-Codedatei und für die allgemeine Verwendung verfügbar. Flüchtig.
- *5021-5029* - Current absolute machine position (G53) for X, Y, Z, A, B, C, U, V & W, in the current program units. Includes no coordinate system, G92/G52 or tool length offset. Same as `#<_abs_x>` through `#<_abs_w>`. Read-only, volatile.
- *5061-5069* - Koordinaten eines G38 Sondenergebnisses (X, Y, Z, A, B, C, U, V & W). Koordinaten befinden sich in dem Koordinatensystem, in dem die G38 stattfand. Flüchtig.
- *5070* - G38 Prüfpunktergebnis: 1 bei Erfolg, 0, wenn Prüfpunkt nicht geschlossen werden konnte. Verwendet mit G38.3 und G38.5. Flüchtig.
- *5081-5089* - Tool length offset currently applied to motion for X, Y, Z, A, B, C, U, V & W, in the current program units. Set by `G43`/`G43.1`/`G43.2`, and 0 when `G49` is in effect. These report the offset actually in effect, which differs from the loaded tool's stored offset in *5401-5409*. Volatile.
- *5161-5169* - "G28" Home für X, Y, Z, A, B, C, U, V & W. Persistent.
- *5181-5189* - "G30" Home für X, Y, Z, A, B, C, U, V & W. Persistent.
- *5210* - 1 if "G52" or "G92" offset is currently applied, 0 otherwise. Persistent by default; volatile if *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.
- *5211-5219* - Gemeinsamer "G52" und "G92" Offset für X, Y, Z, A, B, C, U, V & W. Standardmäßig flüchtig; persistent, wenn *DISABLE_G92_PERSISTENCE = 1* im Abschnitt *[RS274NGC]* der INI-Datei.
- *5220* - Koordinatensystem Nummer 1 - 9 für G54 - G59.3. Persistent.
- *5221-5230* - Koordinatensystem 1, G54 für X, Y, Z, A, B, C, U, V, W & R. R bezeichnet den XY-Drehwinkel um die Z-Achse. Persistent.
- *5241-5250* - Koordinatensystem 2, G55 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5261-5270* - Koordinatensystem 3, G56 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5281-5290* - Koordinatensystem 4, G57 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5301-5310* - Koordinatensystem 5, G58 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5321-5330* - Koordinatensystem 6, G59 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5341-5350* - Koordinatensystem 7, G59.1 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5361-5370* - Koordinatensystem 8, G59.2 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5381-5390* - Koordinatensystem 9, G59.3 für X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5399* - Ergebnis von M66 - Prüfen oder auf Eingabe warten. Flüchtig.
- *5400* - Werkzeugnummer. Flüchtig.
- *5401-5409* - The loaded tool's stored offset for X, Y, Z, A, B, C, U, V & W. Refreshed from the tool table on tool change (`M6`), at startup, and by `G10 L1`/`L10`/`L11`. These are not affected by `G43`/`G43.1`/`G43.2` or `G49`; for the offset actually applied to motion see *5081-5089*. Volatile.
- *5410* - Werkzeugdurchmesser. Flüchtig.
- *5411* - Werkzeug-Frontwinkel. Flüchtig.
- *5412* - Werkzeug-Rückenwinkel (engl. back angle). Flüchtig.
- *5413* - Werkzeugausrichtung. Flüchtig.

- *5420-5428* - Aktuelle relative Position im aktiven Koordinatensystem inklusive aller Offsets und in den aktuellen Programmeinheiten für X, Y, Z, A, B, C, U, V & W, flüchtig.
- *5599* - Flag zur Steuerung der Ausgabe von (DEBUG,)-Anweisungen. 1=Ausgabe, 0=keine Ausgabe; default=1. Flüchtig.
- *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 Subroutine Parameters

* *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 Named Parameters

Named parameters work like numbered parameters but are easier to read. All parameter names are converted to lower case and have spaces and tabs removed, so <*param*> and <*P a R am* > refer to the same parameter. Named parameters must be enclosed with < > marks.

#<*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 Predefined Named Parameters

The following global read only named parameters are available to access internal state of the interpreter and machine state. They can be used in arbitrary expressions, for instance to control flow of the program with if-then-else statements. Note that new predefined named parameters can be added easily without changes to the source code.

- *#<_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 |

| Bewegungsmodus | Rückgabewert |
|---|---|
| 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 System Parameters

* <_spindle_rpm_mode> - Gibt 1 zurück, wenn der Spindeldrehzahlmodus (G97) eingeschaltet ist, sonst 0. * <_spindle_css_mode> - Gibt 1 zurück, wenn der Modus für konstante Schnittgeschwindigkeit (G96) eingeschaltet ist, sonst 0. * <_ijk_absolute_mode> - Gibt 1 zurück, wenn der Modus für den absoluten Bogenabstand (G90.1) eingeschaltet ist, sonst 0. * <_lathe_diameter_mode> - Gibt 1 zurück, wenn es sich um eine Drehbankkonfiguration handelt und der Durchmessermodus (G7) aktiviert ist, sonst 0. * <_lathe_radius_mode> - Gibt 1 zurück, wenn es sich um eine Drehbankkonfiguration handelt und der Radiusmodus (G8) aktiviert ist, sonst 0. * <_spindle_on> - Gibt 1 zurück, wenn die Spindel gerade läuft (M3 oder M4), sonst 0. * <_spindle_cw> - Gibt 1 zurück, wenn die Spindeldrehrichtung im Uhrzeigersinn ist (M3), sonst 0. * <_mist> - Gibt 1 zurück, wenn Nebel (M7) eingeschaltet ist. * <_flut> - Rückgabe 1, wenn Flut (M8) eingeschaltet ist. * <_speed_override> - Rückgabe 1, wenn Vorschubneufestsetzung (M48 oder M50 P1) eingeschaltet ist, sonst 0. * <_feed_override> - Gibt 1 zurück, wenn die Vorschubüberbrückung (M48 oder M51 P1) eingeschaltet ist, sonst 0. * <_adaptive_feed> - Gibt 1 zurück, wenn der adaptive Feed (M52 oder M52 P1) eingeschaltet ist, sonst 0. * <_feed_hold> - Rückgabe 1, wenn der Schalter für die Vorschubfreigabe aktiviert ist (M53 P1), sonst 0. * <_feed> - Gibt den aktuellen Wert von F zurück, nicht den tatsächlichen Vorschub. * <_rpm> - Gibt den aktuellen Wert von S zurück, nicht die tatsächliche Spindeldrehzahl. * <_x> - Gibt die aktuelle relative X-Koordinate einschließlich aller Offsets zurück. Dasselbe wie 5420. In einer Drehbank-Konfiguration wird immer der Radius zurückgegeben. * <_y> - Liefert die aktuelle relative Y-Koordinate einschließlich aller Offsets. Dasselbe wie 5421. * <_z> - Liefert die aktuelle relative Z-Koordinate einschließlich aller Offsets. Dasselbe wie 5422. * <_a> - Liefert die aktuelle relative A-Koordinate einschließlich aller Offsets. Dasselbe wie 5423. * <_b> - Gibt die aktuelle relative B-Koordinate einschließlich aller Offsets zurück. Dasselbe wie 5424. * <_c> - Gibt die aktuelle relative C-Koordinate einschließlich aller Offsets zurück. Dasselbe wie 5425. * <_u> - Liefert die aktuelle relative U-Koordinate einschließlich aller Offsets. Dasselbe wie 5426. * <_v> - Liefert die aktuelle relative V-Koordinate einschließlich aller Offsets. Dasselbe wie 5427. * <_w> - Gibt die aktuelle relative W-Koordinate einschließlich aller Offsets zurück. Dasselbe wie 5428. * <_abs_x> - Rückgabe der aktuellen absoluten X-Koordinate (G53) ohne Offsets. * <_abs_y> - Rückgabe der aktuellen absoluten Y-Koordinate (G53) ohne Offsets. * <_abs_z> - Rückgabe der aktuellen absoluten Z-Koordinate (G53) ohne Offsets. * <_abs_a> - Rückgabe der aktuellen absoluten A-Koordinate (G53) ohne Offsets. * <_abs_b> - Rückgabe der absoluten B-Koordinate

(G53) ohne Offsets. * <*_abs_c*> - Rückgabe der aktuellen absoluten C-Koordinate (G53) ohne Offsets. * <*_abs_u*> - Return current absolute U coordinate (G53) including no offsets. * <*_abs_v*> - Return current absolute V coordinate (G53) including no offsets. * <*_abs_w*> - Return current absolute W coordinate (G53) including no offsets. * <*_aktuelles_Werkzeug*> - Rückgabe der Nummer des aktuellen Werkzeugs in der Spindel. Dasselbe wie #5400. * <*_current_pocket*> - Liefert den Tooldatenindex für das aktuelle Werkzeug. * <*_selected_tool*> - Rückgabe der Nummer des ausgewählten Werkzeugs nach einem T-Code. Voreinstellung -1. * #<_selected_pocket>' - Gibt den tooldata-Index der ausgewählten Tasche nach einem T-Code zurück. Voreinstellung -1 (keine Tasche ausgewählt). * <*_value*> - Rückgabewert des letzten O-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 and INI values

If enabled in the INI file G-code has access to the values of INI file entries and 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
```

The value is read from the INI file once, and cached in the interpreter. These parameters are read-only - assigning a value will cause a runtime error. In the G-code the names are not case sensitive - they are converted to uppercase before consulting the INI file. Hence INI entries that contain lowercase characters can not be accessed from G-code.

- #<*_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 Expressions

An expression is a set of characters starting with a left bracket *[* and ending with a balancing right bracket *]* . In between the brackets are numbers, parameter values, mathematical operations, and other expressions. An expression is evaluated to produce a number. The expressions on a line are evaluated when the line is read, before anything on the line is executed. An example of an expression is *[1 + acos[0] - [#3 ** [4.0/2]]]*.

### 11.4.6 Binary Operators

Binary operators only appear inside expressions. There are four basic mathematical operations: addition (+), subtraction (-), multiplication (*), and division (*/*). There are three logical operations: non-exclusive or (*OR*), exclusive or (*XOR*), and logical and (*AND*). The eighth operation is the modulus operation (*MOD*). The ninth operation is the *power* operation (**) of raising the number on the left of the operation to the power on the right. The relational operators are equality (*EQ*), inequality (*NE*), strictly greater than (*GT*), greater than or equal to (*GE*), strictly less than (*LT*), and less than or equal to (*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: Operators Precedence

| **Operatoren** | **Vorrang** |
|---|---|
| ** | *höchste* |
| * / MOD | |
| + - | |
| EQ NE GT GE LT LE | |
| AND OR XOR | *niedrigste* |

### 11.4.7 Gleichheit und Gleitkommawerte

Testing for equality or inequality of two double-precision floating-point values is inherently problematic. The interpreter solves this problem by considering values equal if their absolute difference is less than 1e-6 (this value is defined as *TOLERANCE_EQUAL* in src/emc/rs274ngc/interp_internal.hh).

### 11.4.8 Funktionen

The available functions are shown in following table. Arguments to unary operations which take angle measures (*COS*, *SIN*, and *TAN* ) are in degrees. Values returned by unary operations which return angle measures (*ACOS*, *ASIN*, and *ATAN*) are also in degrees.

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 Polar Coordinates

Polar Coordinates can be used to specify the XY coordinate of a move. The @n is the distance and ^n is the angle. The advantage of this is for things like bolt hole circles which can be done very simply by moving to a point in the center of the circle, setting the offset and then moving out to the first hole then run the drill cycle. Polar Coordinates always are from the current XY zero position. To shift the Polar Coordinates from machine zero use an offset or select a coordinate system.

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.

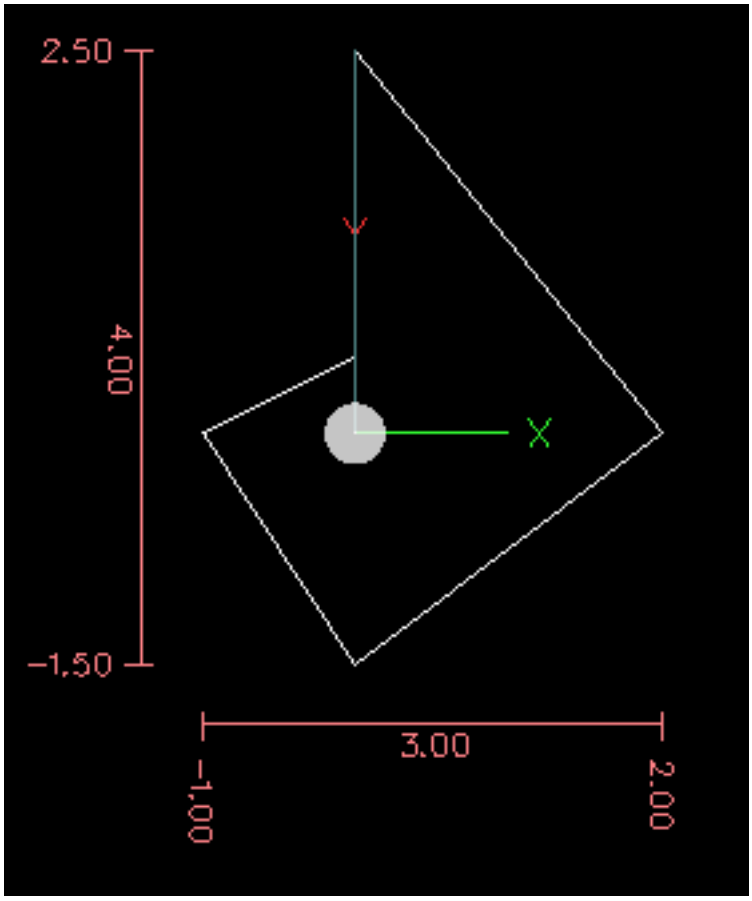



Abbildung 11.10: Polare Spirale

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.

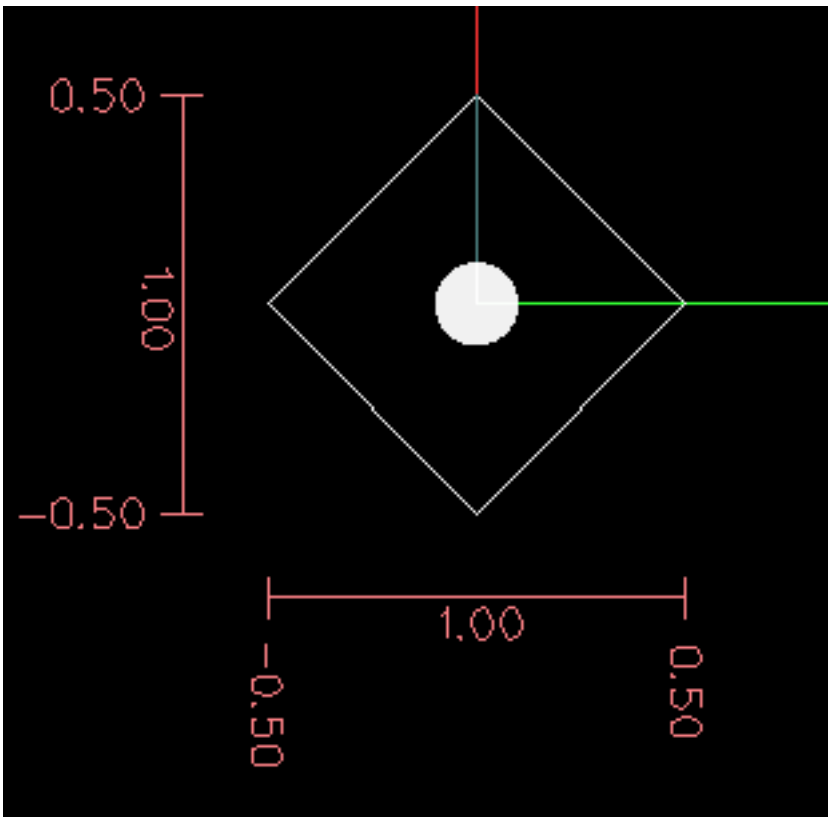



Abbildung 11.11: Polares Quadrat

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 Modal Groups

Modal commands are arranged in sets called *modal groups*, and only one member of a modal group may be in force at any given time. In general, a modal group contains commands for which it is logically impossible for two members to be in effect at the same time - like measure in inches vs. measure in millimeters. A machining center may be in many modes at the same time, with one mode from each modal group being in effect. The modal groups are shown in the following Table.

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 |
| 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 |
| I/O Pins (Group 5) | (M62-M65 digital output), (M66 digital or analog input), (M67, M68 analog output) |
| Toolchange (Group 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

Comments are purely informative and have no influence on machine behaviour.

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 Nachrichten

* *(MSG,)* - zeigt eine Meldung an, wenn *MSG* nach der linken Klammer und vor einem anderen Druckzeichen erscheint. Varianten von *MSG*, die Leerzeichen und Kleinbuchstaben enthalten, sind zulässig. Der Rest der Zeichen vor der rechten Klammer wird als Nachricht betrachtet. Meldungen sollten auf dem Meldungsanzeigegerät der Benutzeroberfläche angezeigt werden, falls vorhanden.

**Beispiel für eine Nachricht**

```
(MSG, Dies ist eine Nachricht)
```

### 11.4.16 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 Messages

* *(DEBUG,)* - zeigt eine Meldung wie *(MSG,)* an, mit dem Zusatz einer besonderen Behandlung von Kommentarparametern, wie unten beschrieben.

### 11.4.19 Print Messages

* *(PRINT,)* - Meldungen werden auf *stderr* ausgegeben, wobei Kommentarparameter wie unten beschrieben besonders behandelt werden.

### 11.4.20 Comment Parameters

In the DEBUG, PRINT and LOG comments, the values of parameters in the message are expanded.

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 File Requirements

A G-code file must contain one or more lines of G-code and be terminated with a Program End. Any G-code past the program end is not evaluated.

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 File Size

The interpreter and task are carefully written so that the only limit on part program size is disk capacity. The TkLinuxCNC and Axis interface both load the program text to display it to the user, though, so RAM becomes a limiting factor. In Axis, because the preview plot is drawn by default, the redraw time also becomes a practical limit on program size. The preview can be turned off in Axis to speed up loading large part programs. In Axis sections of the preview can be turned off using preview control comments.

### 11.4.23 G-code Order of Execution

The order of execution of items on a line is defined not by the position of each item on the line, but by the following list:

- 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 G-code Best Practices

.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 Quick Reference Table

| 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 Rapid Move

```
G0 <Achsen>
```

For rapid motion, program *G0 axes*, where all the axis words are optional. The *G0* is optional if the current motion mode is *G0*. This will produce coordinated motion to the destination point at the maximum rapid rate (or slower). *G0* is typically used as a positioning move.

#### 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 Linear Move

```
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 Arc Move

```
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)
```

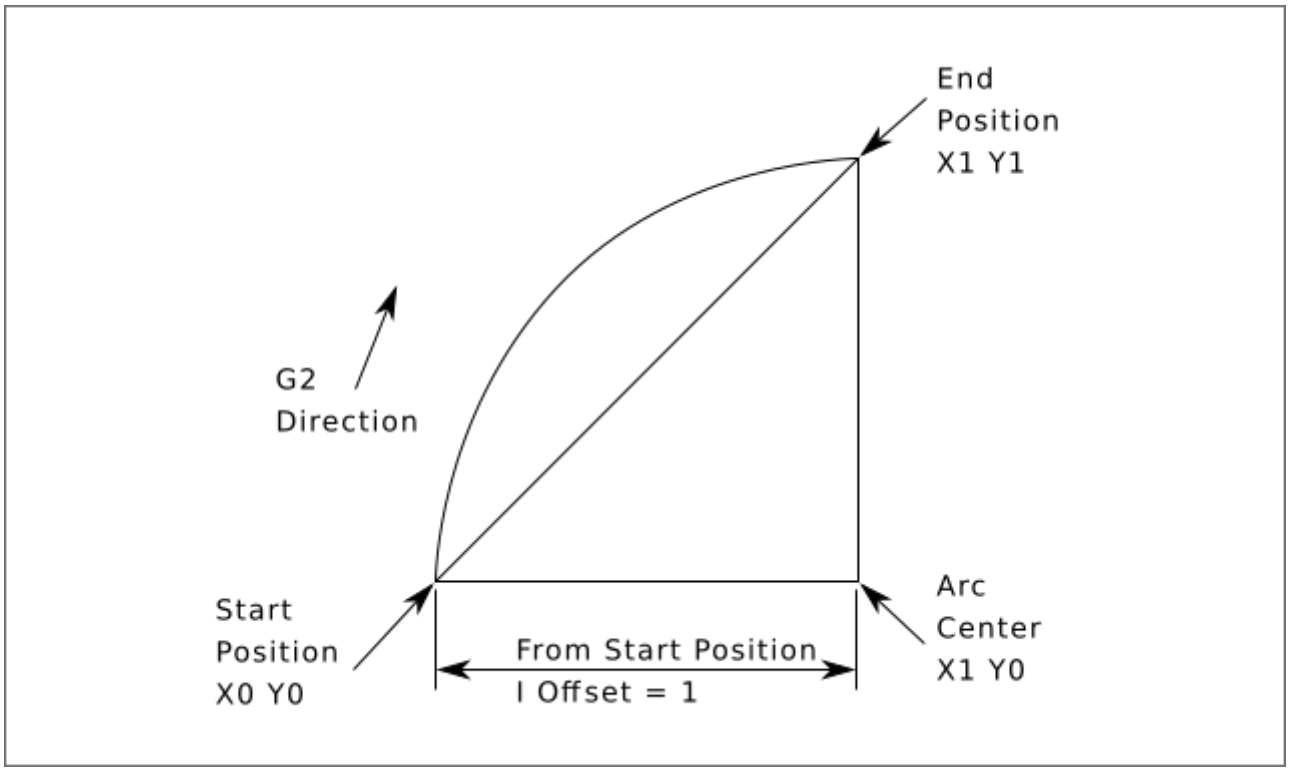



Abbildung 11.12: G2-Beispiel

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)
```

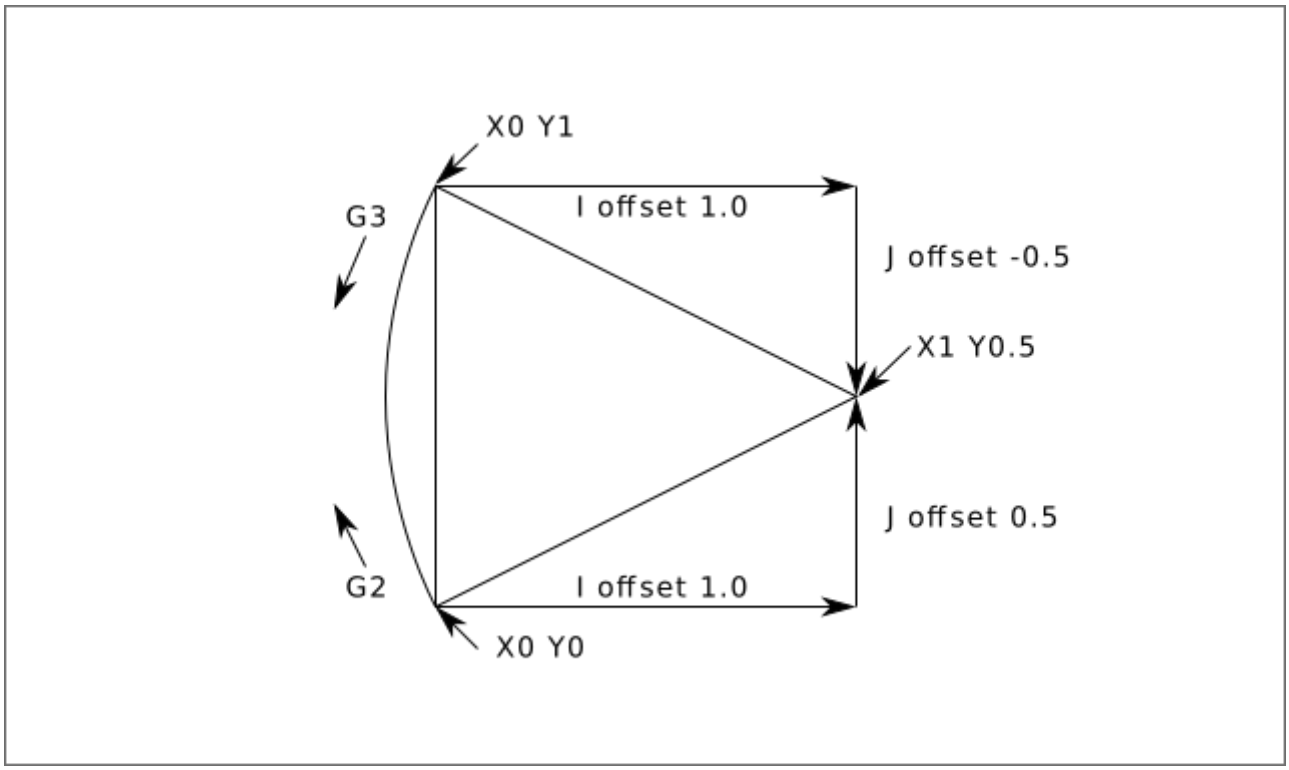



Abbildung 11.13: G2-G3 Beispiel

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 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 Cubic 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 Quadratic 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.

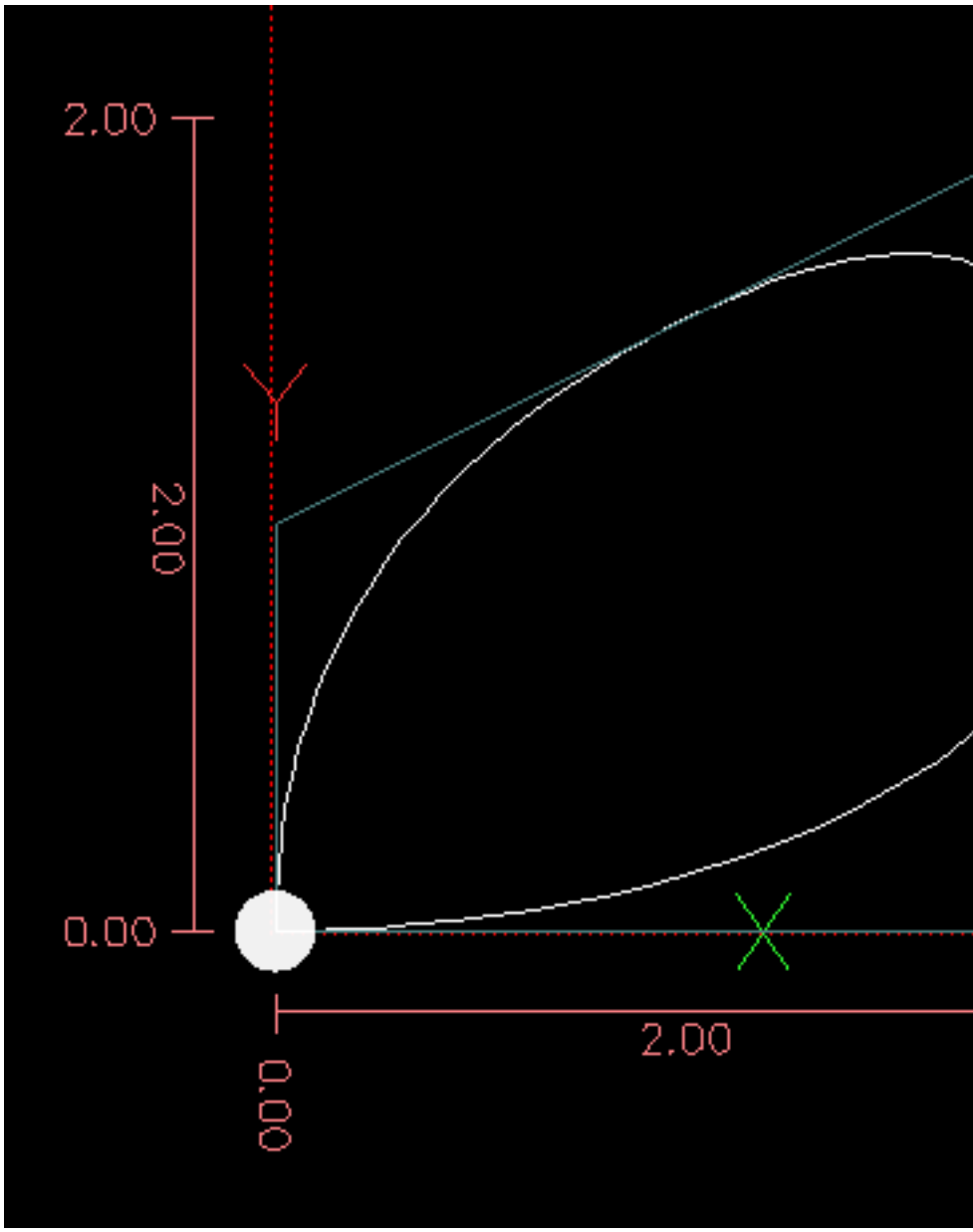
G5.2 dient zum Öffnen des Datenblocks, der ein NURBS definiert, und G5.3 zum Schließen des Datenblocks. In den Zeilen zwischen diesen beiden Codes werden die Kurvenkontrollpunkte mit ihren zugehörigen ”Gewichten” (P) und dem Parameter (L), der die Reihenfolge der Kurve bestimmt, definiert.

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:

https://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS

### 11.5.10 G7 Lathe Diameter Mode

```
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 Lathe Radius Mode

```
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 Reload Tool Table Data

```
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 Set Tool Table

```
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 Set Coordinate System

```
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` does not change from the current coordinate system to the one specified by P, you have to use G54-59.3 to select a coordinate system.
- 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 Set Tool Table

```
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 Set Tool Table

```
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 Set Coordinate System

```
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 Plane Select

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 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 Go/Set Predefined 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 Go/Set Predefined 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 (rapid to Z2.5 then to the Z location specified in #5183)
```

Es ist ein Fehler, wenn :

- Fräserausgleich (engl. cutter compensation) aktiviert ist

### 11.5.22 G33 Spindle Synchronized Motion

```
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 Rigid Tapping

```
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 Straight Probe

```
G38.n Achsen
```

- G38.2” - Messtaster zum Werkstück, Stopp bei Berührung, Signalfehler bei Ausfall
- *G38.3* - Sonde in Richtung Werkstück, Stopp bei Kontakt
- *G38.4* - Messtaster vom Werkstück weg, Stopp bei Kontaktverlust, Fehlersignal bei Ausfall
- *G38.5* - Sonde weg vom Werkstück, Stopp bei Kontaktverlust

**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.

Here is an example formula to probe tool height with conversion from a local coordinate system Z offset to machine coordinates which is stored in the tool table. The existing tool height compensation is first cancelled with G49 to avoid including it in the calculation of height, and the new height is loaded from the tool table. The start position must be high enough above the tool height probe to compensate for the use of G49.

**G38.2 Example**

```
G49
G38.2 Z-100 F100
#<zworkoffset> = [#[5203 + #5220 * 20] + #5213 * #5210]
G10 L1 P#5400 Z#<zworkoffset> (set new tool offset)
G43
```

Probe results in #5061 to #5069 are expressed in the current work coordinate system, not in machine coordinates. To convert one, add the offset between the two frames. That offset is the difference between the current position reported in both frames at once: #5021 to #5029 hold the absolute machine position and #5420 to #5428 hold the work position. Because the offset is the same everywhere, it can be read at any later position and still applies to the earlier probe result.

**Reading the machine coordinates of a probe**

```
G38.2 Z-100 F100
#<zmachine> = [#5063 + #5023 - #5422]
```

#5063 is the probed Z in the work frame, and #5023 - #5422 is the machine-to-work Z offset (current machine Z minus current work Z). The same pattern applies to the other axes using their respective parameters. This holds whichever coordinate system, G92/G52 or tool length offset is active, as long as no coordinate-system rotation is in effect.

Ein Kommentar der Form *(PROBEOPEN dateiname.txt)* öffnet *dateiname.txt* und speichert darin die 9-stellige Koordinate, bestehend aus XYZABCUVW, jeder erfolgreichen geraden Sonde. Die Datei muss mit *(PROBECLOSE)* geschlossen werden. Weitere Informationen finden Sie im Abschnitt Kommentare.

Eine Beispieldatei *smartprobe.ngc* ist enthalten (im Verzeichnis *examples*), um zu demonstrieren, wie die Koordinaten eines Werkstücks mit Hilfe von Tasterbewegungen in eine Datei geschrieben werden. Das Programm *smartprobe.ngc* kann mit minimalen Änderungen mit *ngcgui* verwendet werden.

Es ist ein Fehler, wenn:

- der aktuelle Punkt ist derselbe wie der programmierte Punkt.
- es wird kein Achswort verwendet
- Fräserkompensation ist aktiviert
- der Vorschub ist null
- die Sonde befindet sich bereits im Zielzustand

### 11.5.25 G40 Compensation Off

- *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 Cutter Compensation

```
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 Dynamic Cutter Compensation

```
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 Tool Length 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.

The tool length compensation values actually applied to motion are stored in the numbered parameters *5081-5089*. The numbered parameters *5401-5409* instead hold the loaded tool’s stored offset, refreshed on tool change (*M6*) and *G10 L1/L10/L11*.

---

**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 Dynamic Tool Length 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 (set absolute mode)
T1 M6 G43 (load tool 1 and tool length offsets, Z is at machine 0 and DRO shows Z1.500)
G43.1 Z0.250 (replace current tool offset with 0.250, DRO now shows Z0.250)
M2 (end program)
```

- 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 Apply additional Tool Length Offset

```
G43.2 H- or axes-
```

- *H* - Werkzeugnummer
- *G43.2 Hn* - applies an additional simultaneous tool offset to subsequent motions by adding the offset(s) of tool n.
- *G43.2 axes* - applies an additional simultaneous tool offset to subsequent motions by adding the value(s) of any axis words.

#### G43.2 Hn Example

```
G90 (Absolutmodus einstellen)
T1 M6 (Werkzeug 1 laden)
G43 (oder G43 H1 - alle Werkzeugkorrekturen durch die Offsets/Korrekturen von T1 ersetzen)
G43.2 H10 (fügen Sie auch die Werkzeugkorrektur von T10 ein)
M2 (Programm beenden)
```

**G43.2 axes Example**

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1's offset)
G43.2 X0.01 Z0.02 (also add 0.01 to the x tool offset and 0.02 to the z tool offset)
M2 (end program)
```

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 Cancel Tool Length Compensation

- *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 Local Coordinate System Offset

```
G52-Achsen
```

G52 is used in a part program as a temporary "local coordinate system offset" within the workpiece coordinate system. For more information about `G92` and `G52` and how they interact see Local and Global Offsets.

### 11.5.33 G53 Move in Machine Coordinates

```
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 Select Coordinate System

- *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 Exact Path Mode

- 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 Exact Stop Mode

- G61.1’ - Exakter Stoppmodus, die Bewegung wird am Ende jedes programmierten Segments angehalten.

### 11.5.37 G64 Path Blending

```
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- >* Überlagerung mit Toleranz. Es ist eine Möglichkeit, Ihr System fein abzustimmen, um den besten Kompromiss zwischen Geschwindigkeit und Genauigkeit zu erzielen. Die P-Toleranz bedeutet, dass der tatsächliche Pfad nicht mehr als P- vom programmierten Endpunkt entfernt ist. Die Geschwindigkeit wird bei Bedarf reduziert, um den Pfad beizubehalten. Wenn Sie Q auf einen Wert ungleich Null setzen, wird der *Naive CAM Detector* aktiviert: Wenn es eine Reihe von linearen XYZ-Vorschubbewegungen mit demselben feed rate gibt, die weniger als Q- von der Kollinearität entfernt sind, werden sie zu einer einzigen linearen Bewegung zusammengebrochen. Auf G2/G3 bewegt sich in der G17 (XY) Ebene, wenn die maximale Abweichung eines Bogens von einer geraden Linie kleiner als die G64 P-Toleranz ist, wird der Bogen in zwei Linien aufgeteilt (vom Bogenanfang zum Mittelpunkt und vom Mittelpunkt zum Ende). Diese Zeilen unterliegen dann dem naiven CAM-Algorithmus für Linien. So profitieren Line-Arc-, Arc-Arc- und Arc-Line-Gehäuse sowie Line-Line vom ”Naive CAM Detector”. Dies verbessert die Konturierungsleistung, indem der Pfad vereinfacht wird. Es ist in Ordnung, für den Modus zu programmieren, der bereits aktiv ist. Weitere Informationen zu diesen Modi finden Sie auch im Abschnitt Trajectory Control. Wenn Q nicht angegeben ist, hat es das gleiche Verhalten wie zuvor und verwendet den Wert von P-. Setzen Sie Q auf Null, um den *Naive CAM Detector* zu deaktivieren.

#### G64 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 Lathe finishing cycle

```
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 Lathe roughing cycles

**Anmerkung**
The G71 and G72 cycles are currently somewhat fragile. See issues #707 and #1146.

```
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 Drilling Cycle with Chip Breaking

```
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 Left-hand Tapping Cycle with Dwell

```
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 Threading Cycle

```
G76 P- Z- I- J- R- K- Q- H- E- L- $-
```

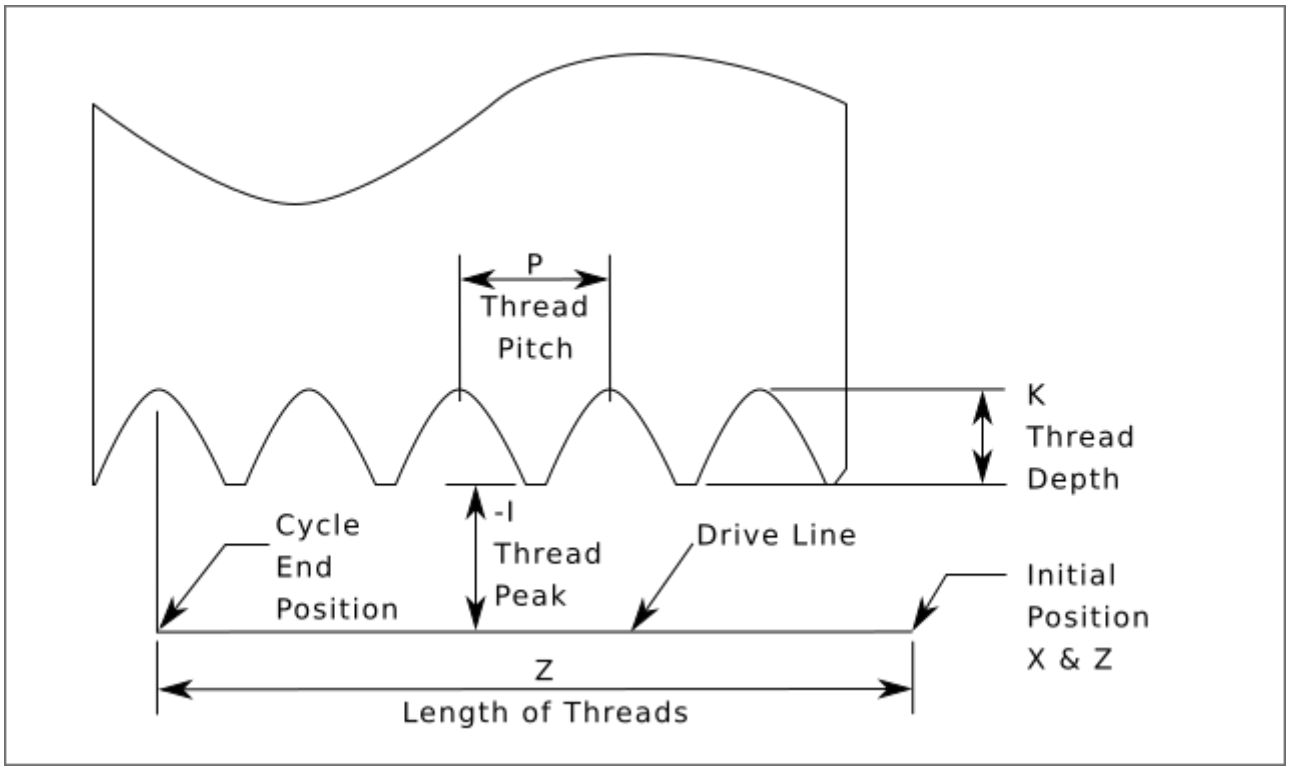



Abbildung 11.14: G76 Gewindeschneiden

- *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**
When G7 *Lathe Diameter Mode* is in force the values for *I*, *J* and *K* are diameter measurements. When G8 *Lathe Radius Mode* is in force the values for *I*, *J* and *K* are radius measurements.

- *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

- *$*- - The spindle number to which the motion will be synchronised (default 0). For example if $1 is programmed then the motion will begin on the reset of `spindle.1.index-enable` and proceed in synchrony with the value of `spindle.1.revs`.
- *R*- - The *depth degression*. *R1.0* selects constant depth on successive threading passes. *R2.0* selects constant area. Values between 1.0 and 2.0 select decreasing depth but increasing area. Values above 2.0 select decreasing area. Beware that unnecessarily high degression values will cause a large number of passes to be used. (degression = a descent by stages or steps.)

**Warnung**
Unnötig hohe Degressionswerte führen zu einer unnötig hohen Anzahl von Durchgängen. (Degression = Tauchen in Stufen)

- *Q*- - The *compound slide angle* is the angle (in degrees) describing to what extent successive passes should be offset along the drive line. This is used to cause one side of the tool to remove more material than the other. A positive *Q* value causes the leading edge of the tool to cut more heavily. Typical values are 29, 29.5 or 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.

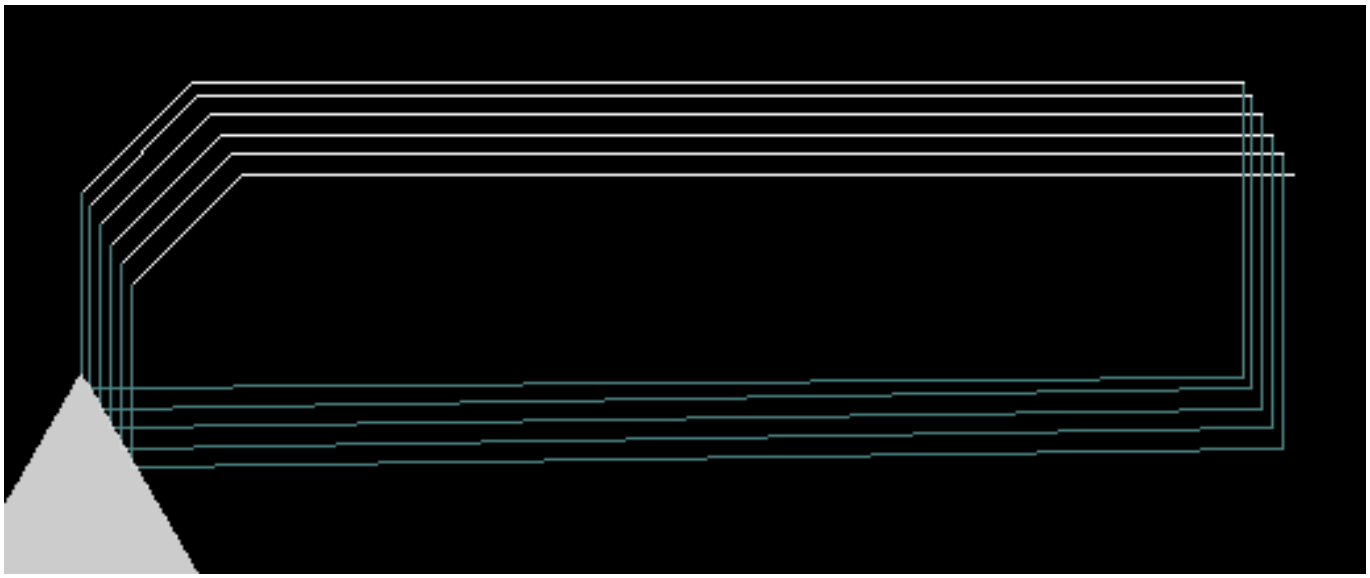


Abbildung 11.15: G76 Beispiel

## 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 Canned Cycle Errors

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.

---

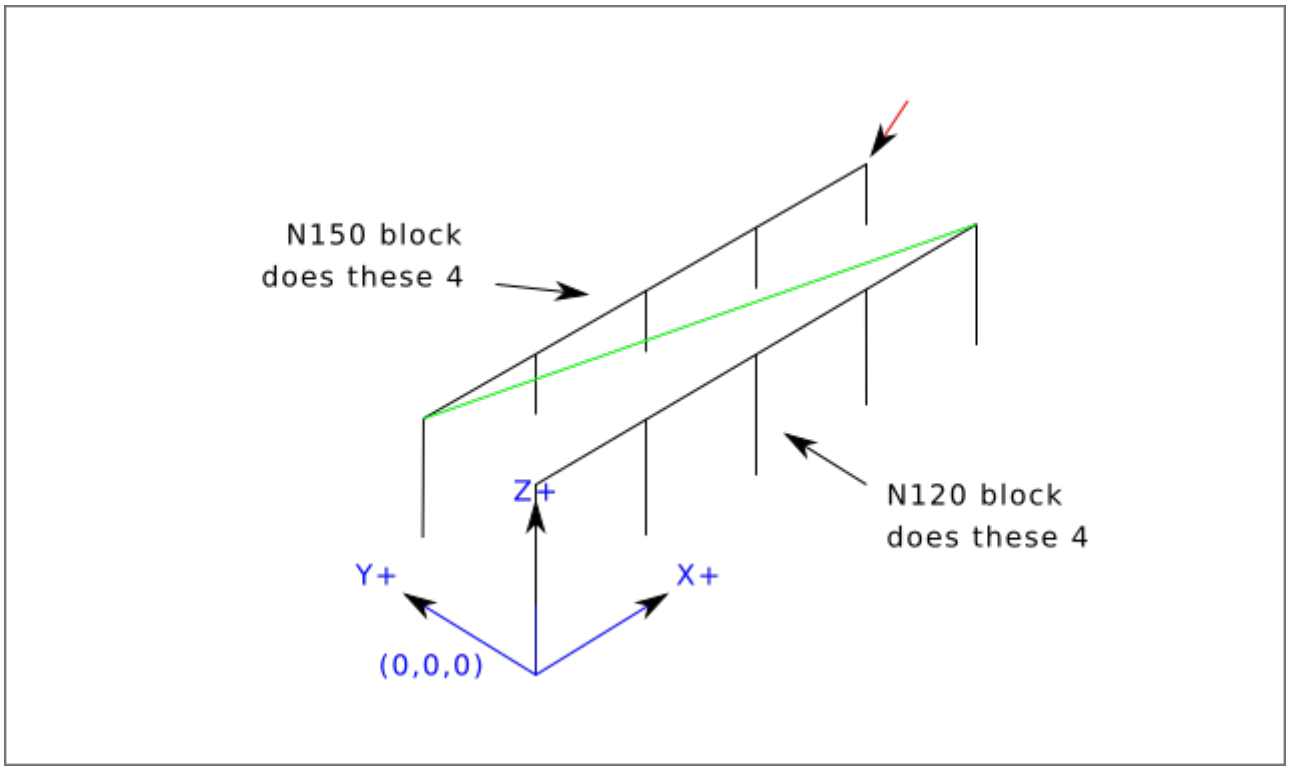
**Anmerkung**
Zeilennummern sind nicht erforderlich, dienen aber der Verdeutlichung dieser Beispiele.

---

**Acht Löcher**

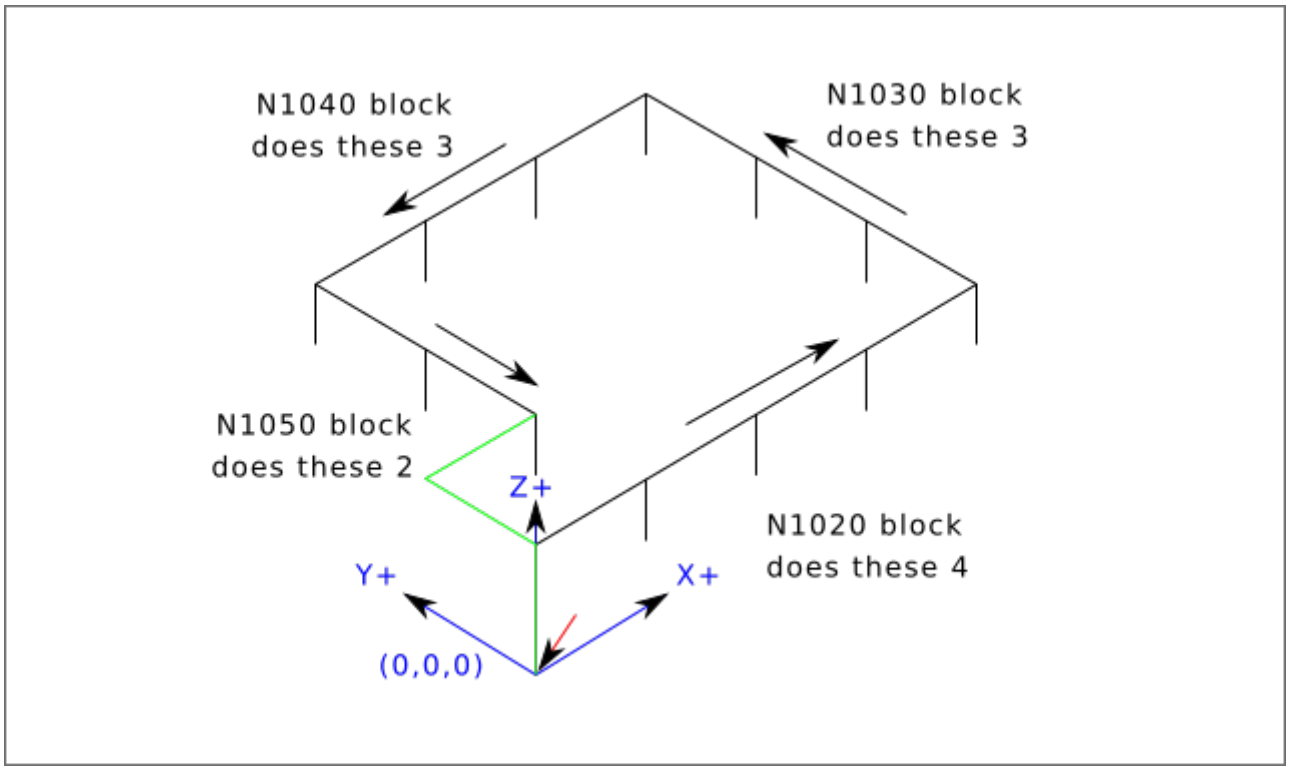
```
N100 G90 G0 X0 Y0 Z0 (Koordinate zum Referenzpunkt setzen)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4 (Festbohrzyklus)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0,5 R1 L4(Festbohrzyklus)
N160 G80 (Ausschalten des Festzyklus)
N170 M2 (Programmende)
```

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 Cancel Canned Cycle

- *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 Drilling Cycle

```
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.

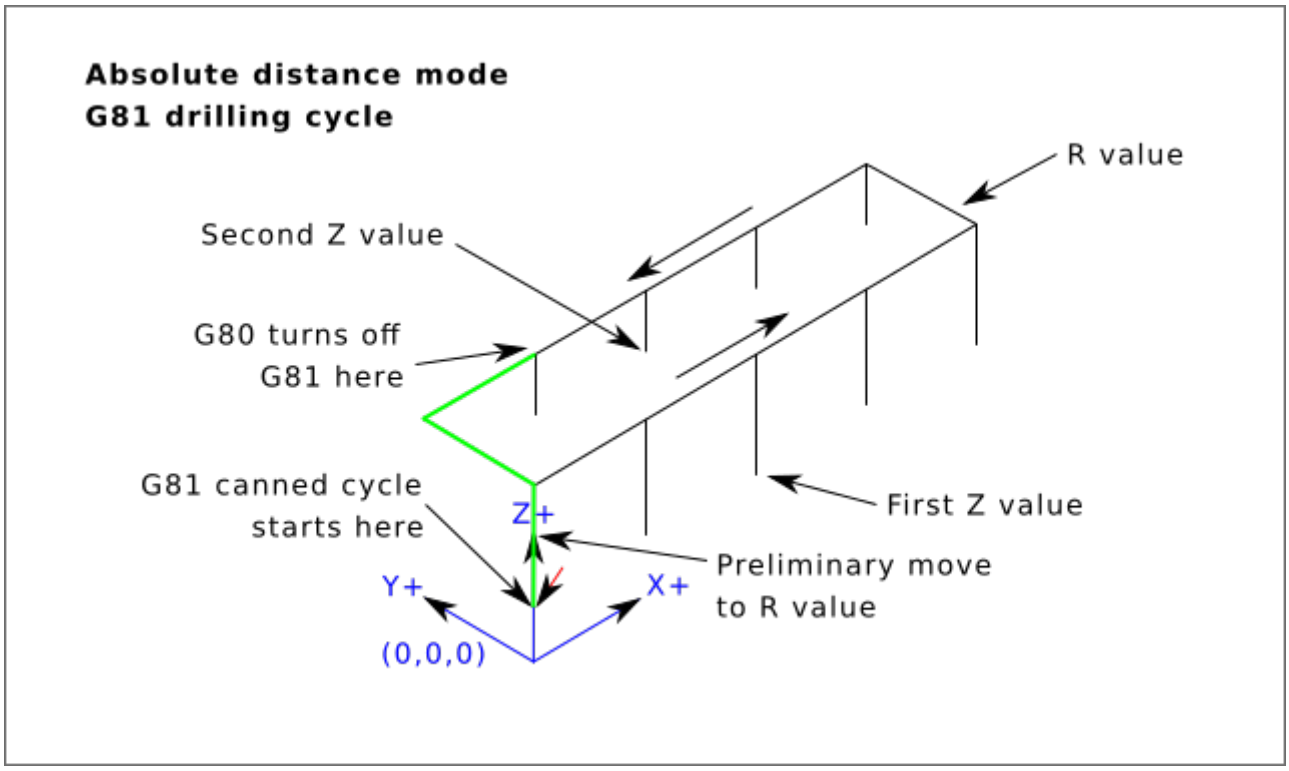



Abbildung 11.16: G81 Zyklus

**Beispiel 1 – Absolute Position G81**

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

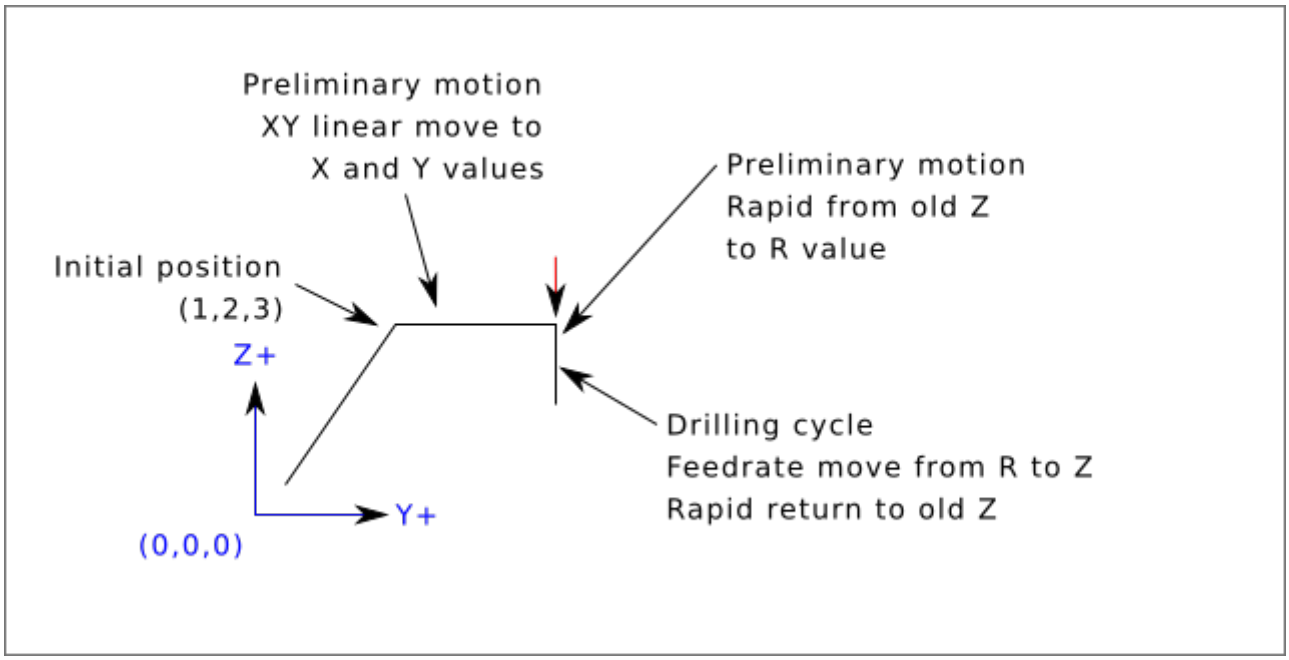
Angenommen, die aktuelle Position ist (X1, Y2, Z3) und die vorangehende Zeile des NC-Codes wird interpretiert.

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)

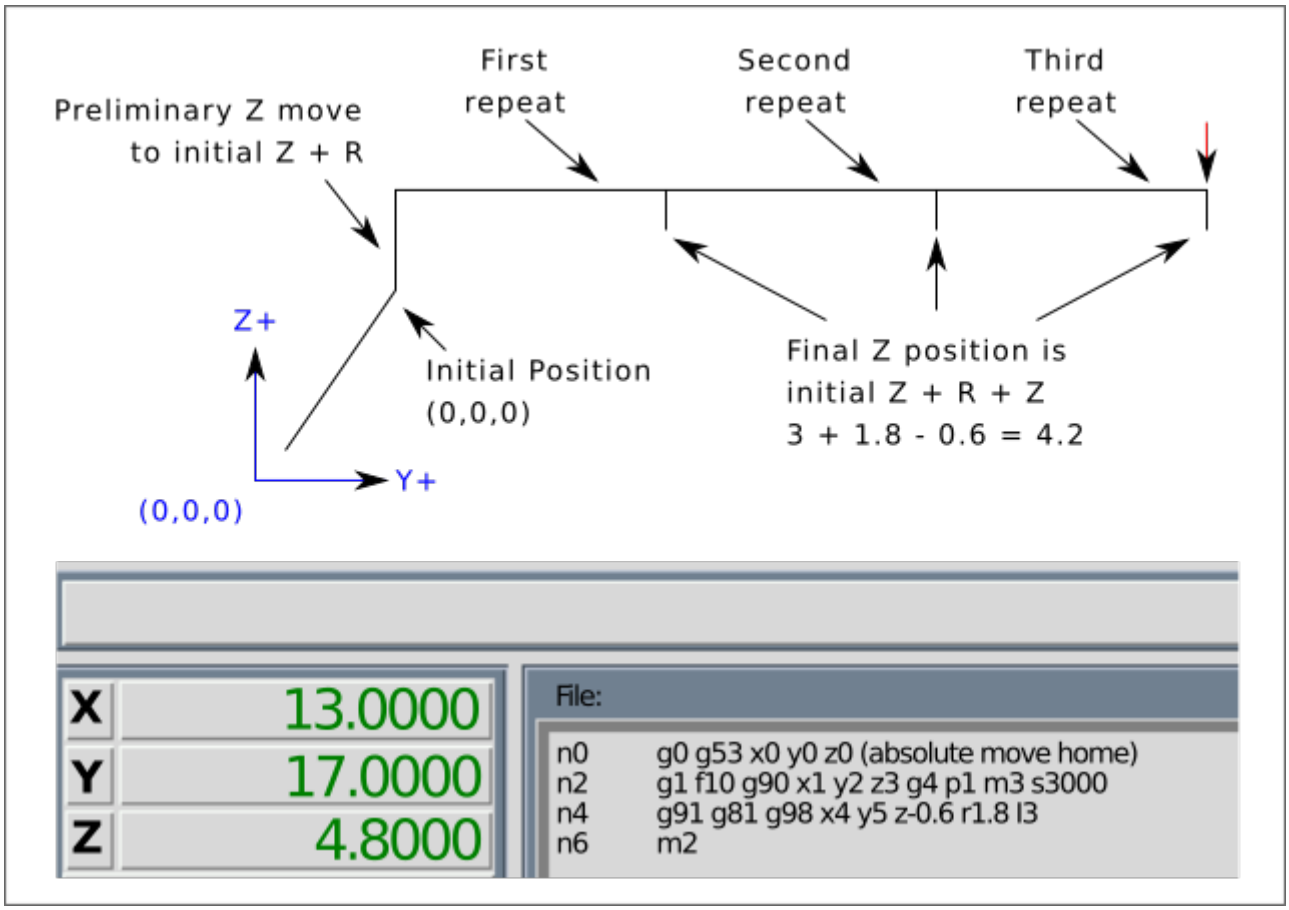
Die zweite Wiederholung besteht aus 3 Zügen. Die X-Position wird auf 9 (=5+4) und die Y-Position auf 12 (=7+5) zurückgesetzt.

- 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)

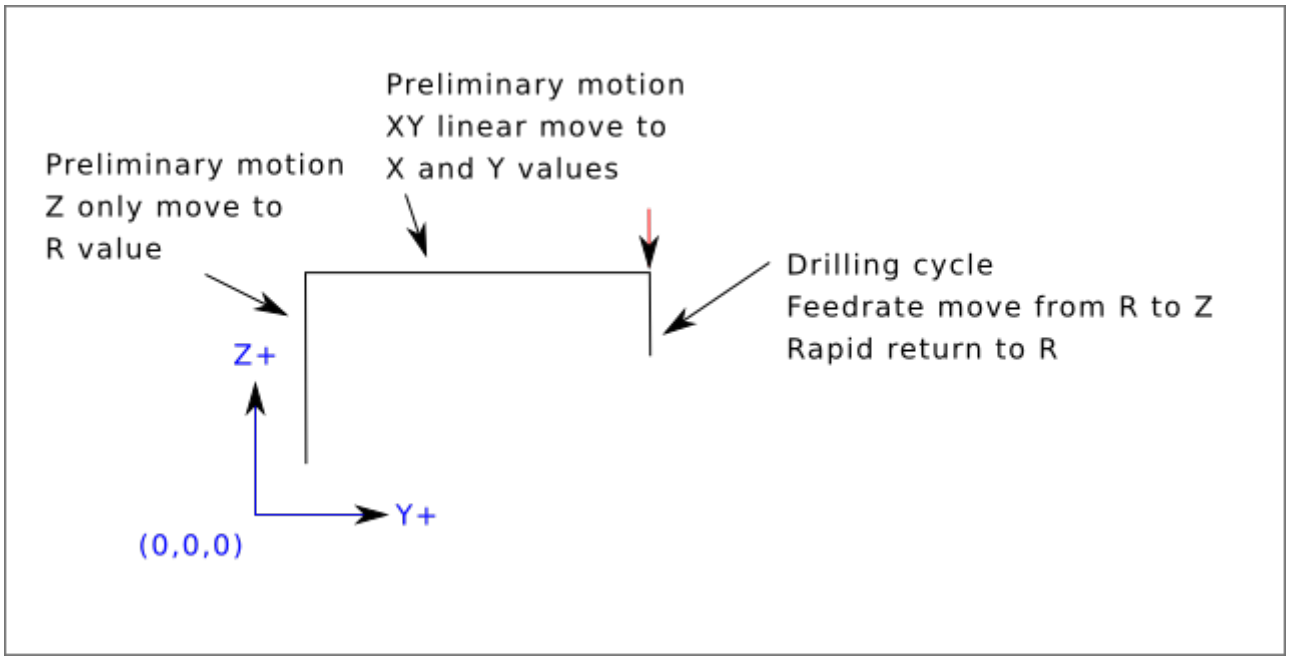



**Beispiel 3 - Relative Position G81**

```
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.

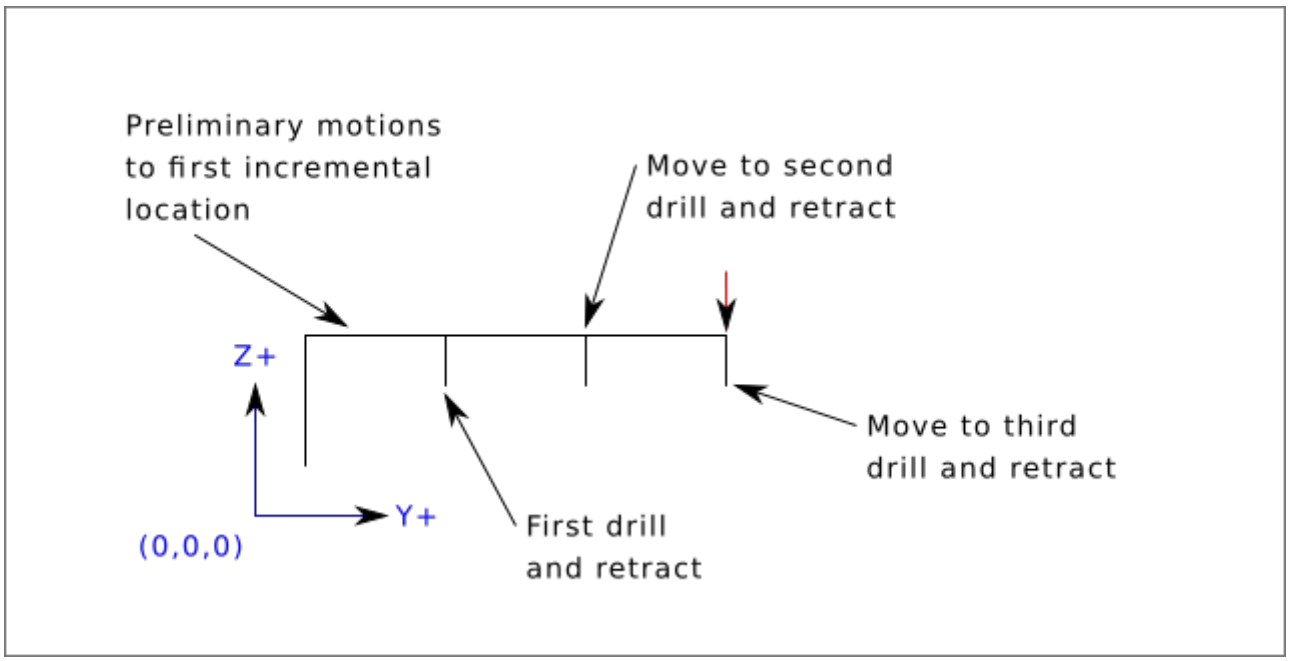
Da OLD_Z unter dem R-Wert liegt, fügt es der Bewegung nichts hinzu, aber da der Anfangswert von Z kleiner als der in R angegebene Wert ist, wird es während der vorbereitenden Bewegungen eine anfängliche Z-Bewegung geben.




**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 Drilling Cycle, Dwell

```
G82 (X- Y- Z-) or (U- V- W-) R- L- P-
```

Der Zyklus *G82* ist für das Bohren mit einer Verweilzeit am Bohrungsgrund vorgesehen.

- Vorläufige Bewegung, wie im Abschnitt Preliminary and In-Between Motion beschrieben.
- Fahren Sie die Z-Achse mit dem aktuellen Vorschub auf die Z-Position.
- Verweilen für die Anzahl von P Sekunden.
- Die Z-Achse führt einen Eilgang aus, um über Z den Weg freizumachen.

Die Bewegung eines G82-Bohrzyklus (engl. canned cycle) sieht genauso aus wie G81 mit dem Zusatz einer Verweilzeit am Ende der Z-Bewegung. Die Länge der Verweilzeit wird durch ein ’P-’Wort im G82-Satz festgelegt.

```
G90 G82 G98 X4 Y5 Z1.5 R2.8 P2
```

Dies entspricht dem obigen Beispiel 3, nur mit einer zusätzlichen Verweilzeit von 2 Sekunden am Boden des Lochs.

### 11.5.47 G83 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 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 Boring Cycle, Feed Out

```
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 Boring Cycle, Spindle Stop, Rapid Move Out

```
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 Back Boring Cycle

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

### 11.5.52 G88 Boring Cycle, Spindle Stop, Manual Out

Dieser Code ist derzeit nicht in LinuxCNC implementiert. Es wird akzeptiert, aber das Verhalten ist undefiniert.

### 11.5.53 G89 Boring Cycle, Dwell, Feed Out

```
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 Distance Mode

- *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 Arc Distance Mode

- *G90.1* - absoluter Abstandsmodus für I, J und K Versatz. Wenn G90.1 in Kraft ist, müssen I und J beide mit G2/3 für die XY-Ebene oder J und K für die XZ-Ebene angegeben werden, sonst ist es ein Fehler.
- *G91.1* - incremental distance mode for I, J & K offsets. `G91.1` Returns I, J & K to their default behavior.

### 11.5.56 G92 Coordinate System 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.

For example, suppose the current point is at X=4 and there is currently no *G92* offset active. Then *G92 X7* is programmed. This moves all origins -3 in X, which causes the current point to become X=7. This -3 is saved in parameter 5211.

Being in incremental distance mode (G91 instead of G90) has no effect on the action of G92.

*G92* offsets may be already be in effect when the G92 is called. If this is the case, the offset is replaced with a new offset that makes the current point become the specified value.

Es ist ein Fehler, wenn alle Achsworte weggelassen werden.

LinuxCNC stores the G92 offsets and reuses them on the next run of a program. To prevent this, one can program a G92.1 (to erase them), or program a G92.2 (to remove them - they are still stored).

**Anmerkung**
The *G52* command can also be used to change this offset; see the Offsets section for more details about G92 and G52 and how they interact.

Einen Überblick über Koordinatensysteme finden Sie im Abschnitt Koordinatensystem.

Siehe den Abschnitt zu Parametern für weitere Informationen.

### 11.5.57 G92.1, G92.2 Reset G92 Offsets

- *G92.1* - Schalten Sie G92-Offsets aus und setzt Parameter 5211 - 5219 auf Null zurück.
- *G92.2* - G92-Offsets ausschalten, aber Parameter 5211 - 5219 verfügbar lassen.

**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 Restore G92 Offsets

- *G92.3* - set the G92 offset to the values saved in parameters 5211 to 5219

You can set axis offsets in one program and use the same offsets in another program. Program G92 in the first program. This will set parameters 5211 to 5219. Do not use G92.1 in the remainder of the first program. The parameter values will be saved when the first program exits and restored when the second one starts up. Use G92.3 near the beginning of the second program. That will restore the offsets saved in the first program.

### 11.5.59 G93, G94, G95 Feed Rate Mode

- *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.

  When the inverse time feed rate mode is active, an F word must appear on every line which has a G1, G2, or G3 motion, and an F-word on a line that does not have G1, G2, or G3 is ignored. Being in inverse time feed rate mode does not affect G0 (rapid move) motions.

- ”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* - is Units per Revolution Mode. In units per revolution mode, an F-word is interpreted to mean the controlled point should move a certain number of inches per revolution of the spindle, depending on what length units are being used and which axis or axes are moving. `G95` is not suitable for threading, for threading use `G33` or `G76`. `G95` requires that `spindle.N.speed-in` to be connected. The actual spindle to which the feed is synchronised is chosen by the `$` parameter.

Es ist ein Fehler, wenn:

- Inverse time feed mode is active and a line with `G1`, `G2`, or `G3` (explicitly or implicitly) does not have an F-word.
- A new feed rate is not specified after switching to `G94` or `G95`

## 11.5.60 G96, G97 Spindle Control Mode

```
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 feet per minute if `G20` is in effect,
     - or meters per minute if `G21` is in effect.

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 Canned Cycle Return Level

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 Program Pause

* M0’ - pausiert ein laufendes Programm vorübergehend. LinuxCNC bleibt im Auto-Modus, somit sind MDI und andere manuelle Aktionen sind nicht aktiviert. Durch Drücken der Taste *Fortführen* (engl. *Resume*) wird das Programm in der folgenden Zeile weiterlaufen. * M1’ - pausiert ein laufendes Programm vorübergehend, wenn der optionale Stop-Schalter eingeschaltet ist. LinuxCNC bleibt im Auto-Modus so MDI und andere manuelle Aktionen sind nicht aktiviert. Durch Drücken der Resume-Taste wird das Programm in der folgenden Zeile neu gestartet.

---

**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 Program End

* *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 Pallet Change 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 Spindle Control

* *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 Tool Change

#### 11.6.6.1 Manueller Werkzeugwechsel

If the HAL component *hal_manualtoolchange* is loaded, M6 will stop the spindle and prompt the user to change the tool based on the last *T*- number programmed. For more information on hal_manualtoolchange see the Manual Tool Change section.

#### 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 Coolant Control

* *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 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 Speed and Feed Override Control

* *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 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 Spindle Speed Override Control

* 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 Feed Control

* *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 Feed Stop Control

* *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 Set Current Tool

* *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 Digital Output Control

* *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 Wait on Input

```
M66 P- | E- <L->
```

- *P-* - gibt die digitale Eingangsnummer von 0 bis 3 an. (Einstellbar mit motmod Argument num_dio)
- *E-* - gibt die Nummer des Analogeingangs von 0 bis 3 an (einstellbar über das motmod-Argument num_aio)
- *L-* - legt den Wartemodus fest.
  - Mode 0: IMMEDIATE' - kein Warten, kehrt sofort zurück. Der aktuelle Wert des Eingangs wird in Parameter #5399 gespeichert
  - *Mode 1: RISE* - wartet darauf, dass die ausgewählte Eingabe ein RISE-Ereignis ausführt.
  - *Mode 2: FALL* - wartet darauf, dass die ausgewählte Eingabe ein Fallereignis ausführt.
  - *Mode 3: HIGH* - wartet darauf, dass die ausgewählte Eingabe in den HIGH-Zustand wechselt.

- *Mode 4: LOW* - wartet, bis die ausgewählte Eingabe in den LOW-Zustand wechselt.

- x'Q-' - gibt das Timeout in Sekunden für das Warten an. Wird das Timeout überschritten, so wird die Wartezeit unterbrochen, und die Variable #5399 enthält den Wert -1. Der Q-Wert wird ignoriert, wenn das L-Wort Null ist (SOFORT). Ein Q-Wert von Null ist ein Fehler, wenn das L-Wort ungleich Null ist.
- Für einen Analogeingang ist nur der Modus 0 zulässig.

**M66 Beispielzeilen**

```
M66 P0 L3 Q5 (bis zu 5 Sekunden warten, bis der digitale Eingang 0 eingeschaltet wird)
```

M66 wartet auf eine Eingabe und stoppt die weitere Ausführung des Programms, bis das gewählte Ereignis (oder der programmierte Timeout) eintritt.

Es ist ein Fehler, M66 sowohl mit einem P-Wort als auch mit einem E-Wort zu programmieren (also sowohl einen analogen als auch einen digitalen Eingang zu wählen). In LinuxCNC sind diese Eingänge nicht in Echtzeit überwacht und sollte daher nicht für zeitkritische Anwendungen verwendet werden.

Die Anzahl der E/A kann durch Verwendung des Parameters num_dio oder num_aio beim Laden des Motion Controllers erhöht werden. Siehe den Abschnitt zu Bewegungen (engl. motion) für weitere Informationen.

---

**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 Analog Output, Synchronized

```
M67 E- Q-
```

- *M67* - stellt einen mit der Bewegung synchronisierten analogen Ausgang ein.
- *E-* - Ausgangsnummer im Bereich von 0 bis 3 (einstellbar über das motmod-Argument num_aio)
- *Q-* - ist der einzustellende Wert (zum Ausschalten auf 0 setzen).

Die tatsächliche Änderung der angegebenen Ausgänge erfolgt zu Beginn des nächsten Fahrbefehls. Wenn es keinen nachfolgenden Bewegungsbefehl gibt, werden die in der Warteschlange stehenden Ausgangsänderungen nicht durchgeführt. Am besten programmieren Sie immer einen Bewegungs-G-Code (G0, G1, etc.) direkt nach M67. M67 funktioniert genauso wie M62-63.

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 Analog Output, Immediate

```
M68 E-Q-
```

- *M68* - sofort einen analogen Ausgang einstellen.
- *E-* - Ausgangsnummer im Bereich von 0 bis 3 (einstellbar über das motmod-Argument num_aio)
- *Q-* - ist der einzustellende Wert (zum Ausschalten auf 0 setzen).

M68-Ausgaben erfolgen sofort, wenn sie vom Motion Controller empfangen werden. Sie sind nicht mit der Bewegung synchronisiert und unterbrechen das Blending. M68 funktioniert genauso wie M64-65.

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 Save Modal State

To explicitly save the modal state at the current call level, program *M70*. Once modal state has been saved with *M70*, it can be restored to exactly that state by executing an *M72*.

Ein Paar von *M70*- und *M72*-Befehlen wird normalerweise verwendet, um ein Programm gegen unbeabsichtigte Modaländerungen innerhalb von Unterprogrammen zu schützen.

**M70 Saved state**

The state saved consists of:

- 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 Invalidate Stored Modal State

Modal state saved with an *M70* or by an *M73* at the current call level is invalidated (cannot be restored from anymore).

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 Restore Modal State

Modal state saved with an M70 code can be restored by executing an *M72*.

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 Save and Autorestore Modal State

To save modal state within a subroutine, and restore state on subroutine *endsub* or any *return* path, program *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 User Defined Commands

```
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.

Execution of the G-code file pauses until the external program exits. If the external program exits with exit code other than *0* gcode program execution is stopped. Any valid executable file can be used. The file must be located in the search path specified in the INI file configuration. See the Display section for more information on search paths.

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

Comments on the same line as the O word should not be used as the behavior can change in the future.

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 Subroutines

Subroutines starts at *Onnn sub* and ends at *Onnn endsub*. The lines between *Onnn sub* and *Onnn endsub* are not executed until the subroutine is called with *Onnn call*. Each subroutine must use a unique number.

**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.

Subroutines can change the value of parameters above #30 and those changes will be visible to the calling code. Subroutines may also change the value of global named parameters (i.e. parameters whose names begin with the underscore character "_").

#### 11.7.4.1 Fanuc-Style Numbered Programs

Numbered programs (both main and subprograms), the *M98* call and *M99* return M-codes, and their respective semantic differences are an alternative to the rs274ngc subroutines described above, provided for compatibility with Fanuc and other machine controllers.

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 Looping

The *while loop* has two structures: *while/endwhile*, and *do/while*. In each case, the loop is exited when the *while* condition evaluates to false. The difference is when the test condition is done. The *do/while* loop runs the code in the loop then checks the test condition. The *while/endwhile* loop does the test first.

**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 Conditional

The *if* conditional consists of a group of statements with the same *o* number that start with *if* and end with *endif*. Optional *elseif* and *else* conditions may be between the starting *if* and the ending *endif*.

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 Repeat

The *repeat* will execute the statements inside of the repeat/endrepeat the specified number of times. The example shows how you might mill a diagonal series of shapes starting at the present 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 Indirection

The O-number may be given by a parameter and/or calculation.

**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 Calling Files

To call a separate file with a subroutine name the file the same as your call and include a sub and endsub in the file. The file must be in the directory pointed to by *PROGRAM_PREFIX* or *SUBROUTINE_PATH* in the INI file. The file name can include **lowercase** letters, numbers, dash, and underscore only. A named subroutine file can contain only a single subroutine definition.

**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 Subroutine return values

Subroutines may optionally return a value by an optional expression at an *endsub* or *return* statement.

**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 Errors

The following statements cause an error message and abort the interpreter:

- 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 Other Codes

### 11.8.1 F: Set Feed Rate

*Fx* - set the feed rate to *x*. *x* is usually in machine units (inches or millimeters) per 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: Set Spindle Speed

*Sx [$n]* - set the speed of the spindle to *x* revolutions per minute (RPM) with the optional $ set the spindle speed for a specific spindle. Without the $ the command will default to spindle.0.

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: Select Tool

*Tx* - prepare to change to tool *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.

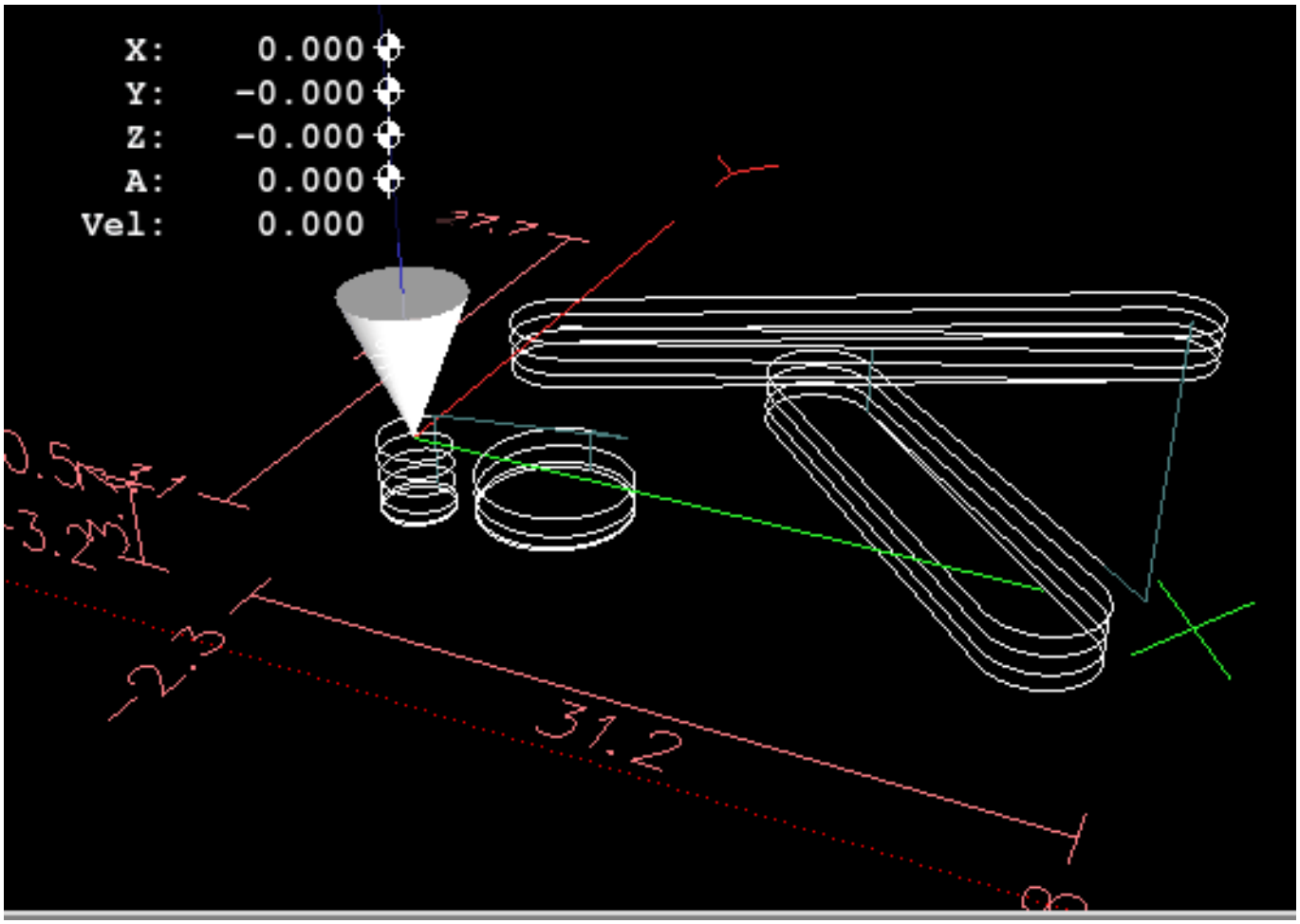
### 11.9.1 Beispiele für eine Fräsmaschine

#### 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)

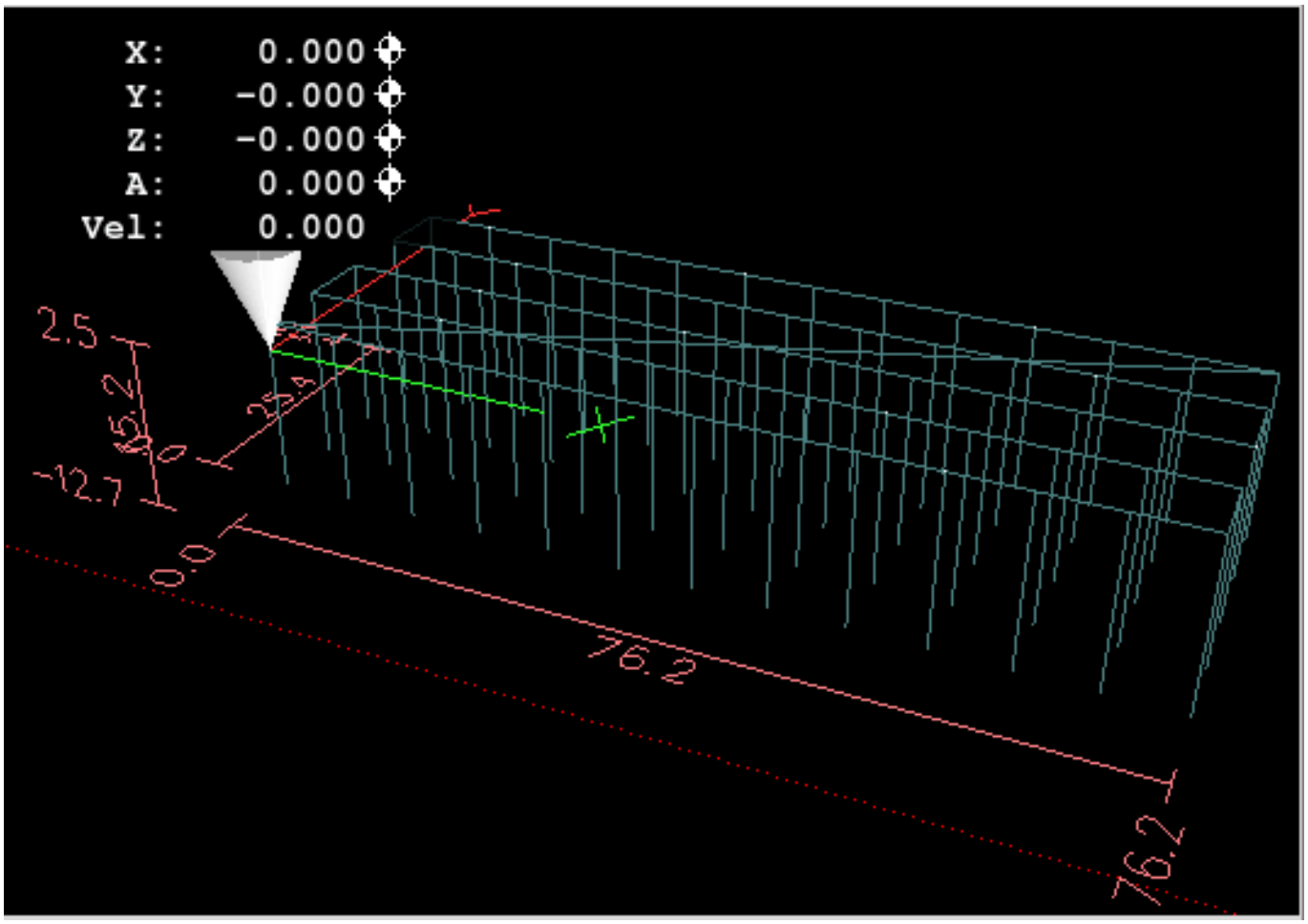
- Dateiname: useful-subroutines.ngc
- Beschreibung: Unterprogramm zum Fräsen einer Nut mit Parametern.




#### 11.9.1.3 Rastersonde (engl. grid probe)

- Dateiname: gridprobe.ngc
- Beschreibung: Rechteckige Sondierung

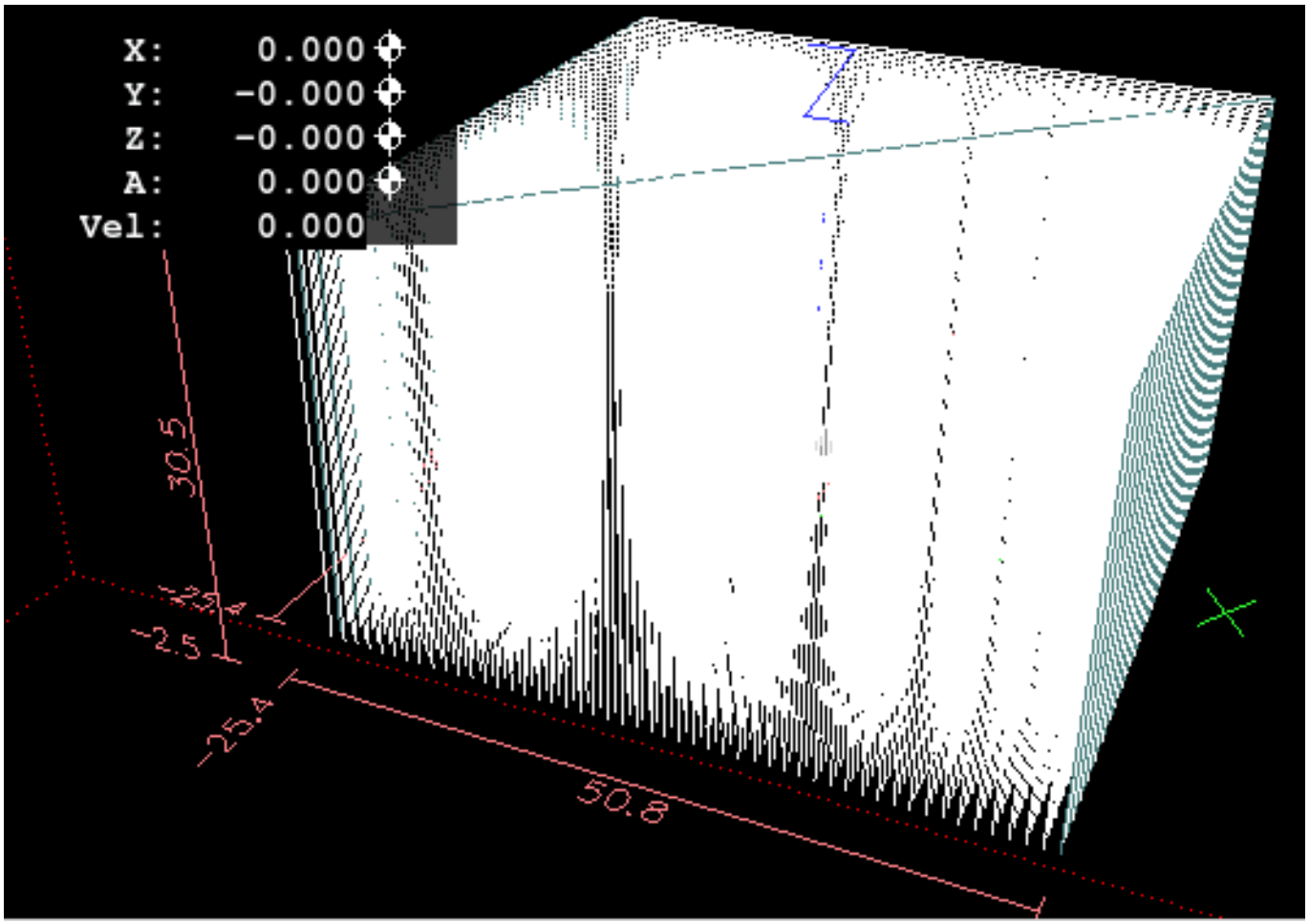
Dieses Programm testet wiederholt in einem regelmäßigen XY-Raster und schreibt die getestete Position in die Datei *probe-results.txt* im selben Verzeichnis wie die .ini-Datei.




#### 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.

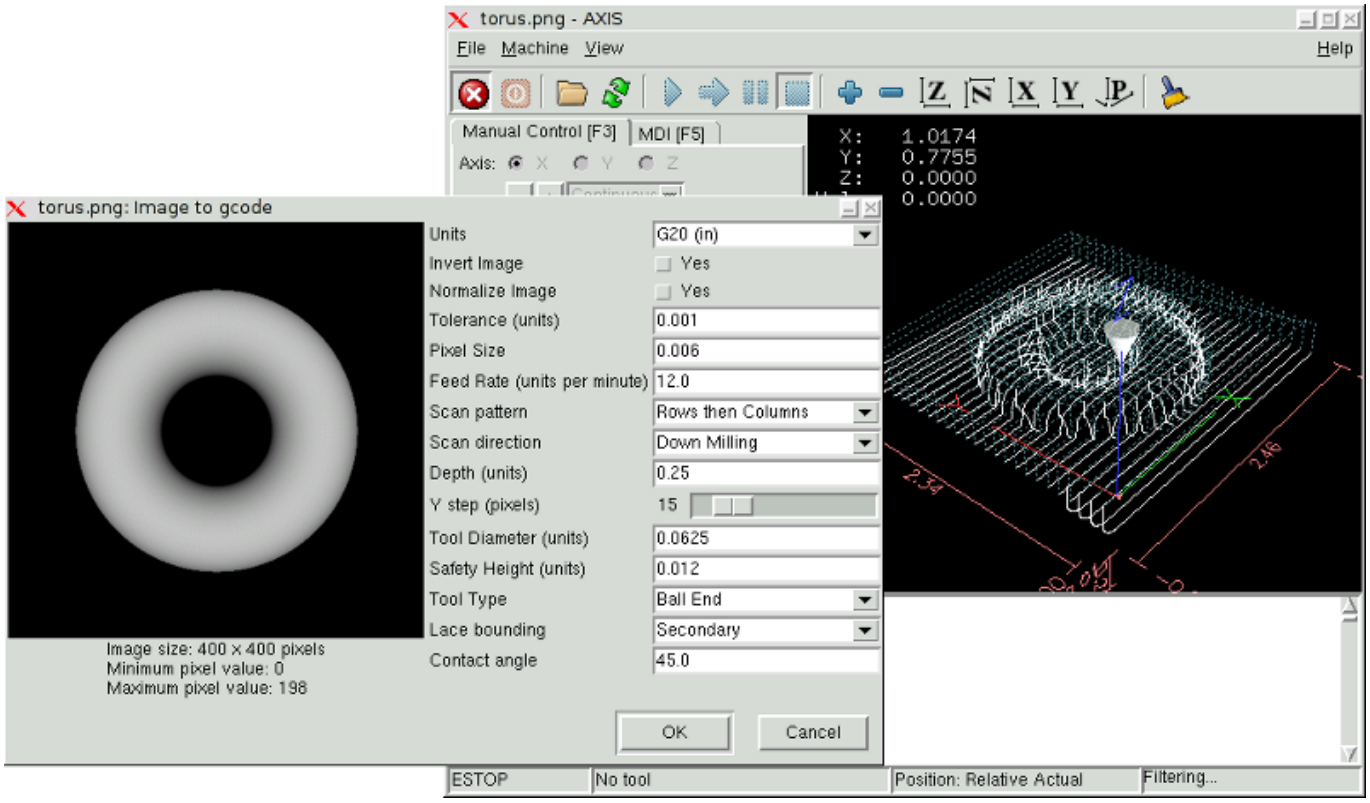
### 11.9.2 Beispiele für Drehmaschinen

#### 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)

The top of material is always at *Z=0*. The deepest cut into the material is at *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.

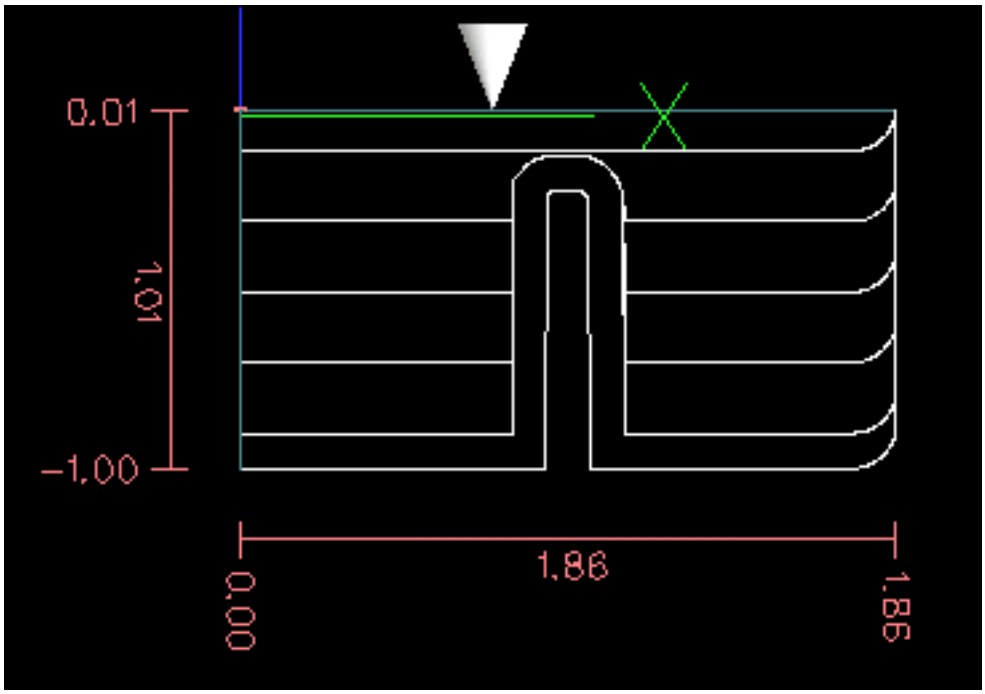



Abbildung 11.17: Schruppen und Schlichten

## 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.

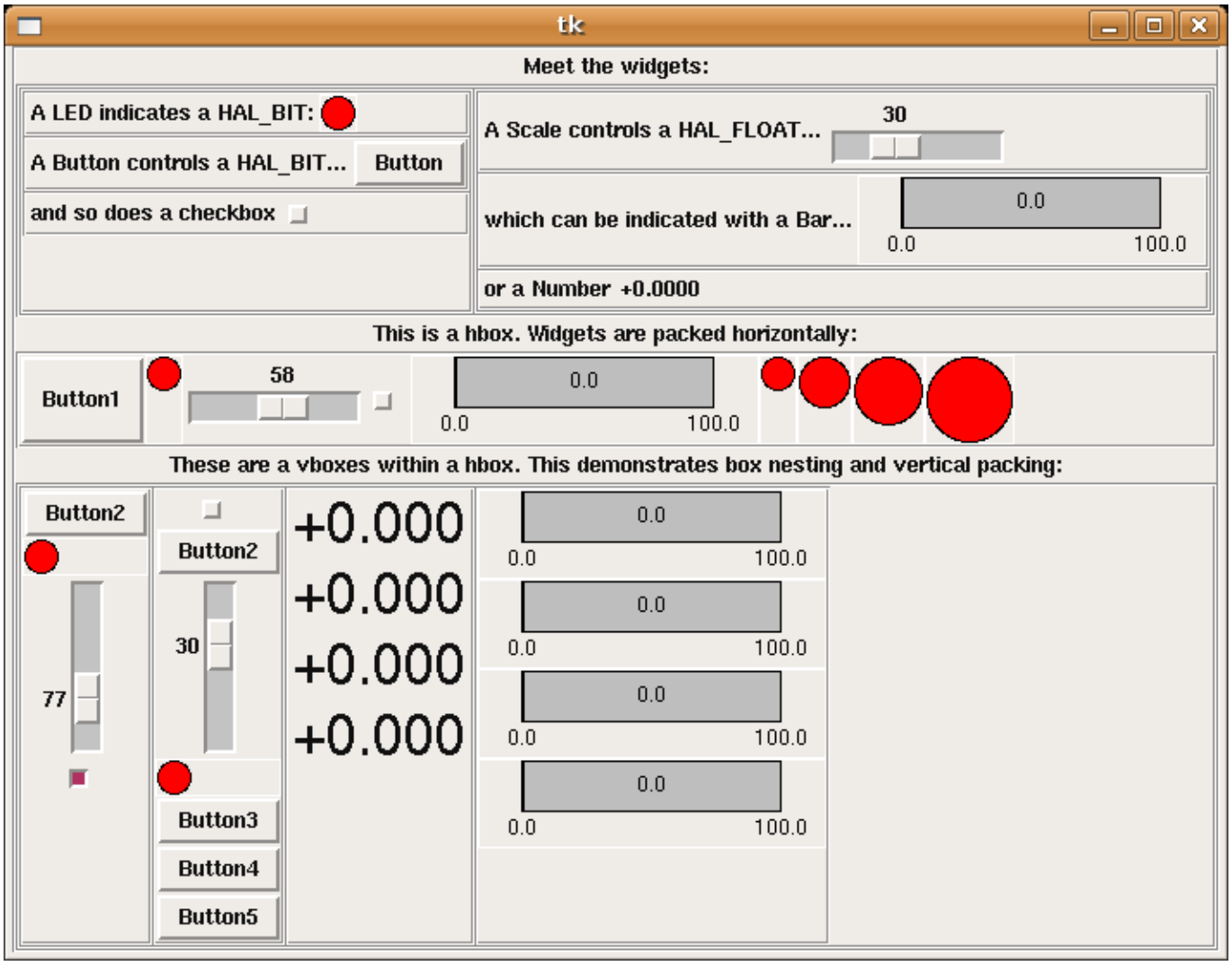



Abbildung 12.1: PyVCP Widgets Showcase

### 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

Parts of PyVCP files are evaluated as Python code, and can take any action available to Python programs. Only use PyVCP XML files from a source that you trust.

### 12.1.4 ACHSE

Since AXIS uses the same GUI toolkit (Tkinter) as PyVCP, it is possible to include a PyVCP panel at either the right side or the bottom of the AXIS user interface. It is not possible to display a panel in both of these positions simultaneously. A typical example is explained below.

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

Place your PyVCP XML file describing the panel in the same directory where your INI file is. Say we we want to display the current spindle speed using a Bar widget. Place the following in a file called spindle.xml:

```
<pyvcp>
    <label>
       <text>”Spindeldrehzahl:”</text>
    </label>
   <bar>
        <halpin>”spindle-speed”</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Here we've made a panel with a Label and a Bar widget, specified that the HAL pin connected to the Bar should be named *spindle-speed*, and set the maximum value of the bar to 5000 (see widget reference below for all options). To make AXIS aware of this file, and call it at start up, we need to specify the following in the `[DISPLAY]` section of the INI file:

```
PYVCP = spindle.xml
```

If the panel should appear at the bottom of the AXIS user interface then we need to specify the following in the `[DISPLAY]` section of the INI file:

```
PYVCP_POSITION = BOTTOM
```

Alles andere als BOTTOM oder das Weglassen dieser Variable platziert das PyVCP-Panel auf der rechten Seite.

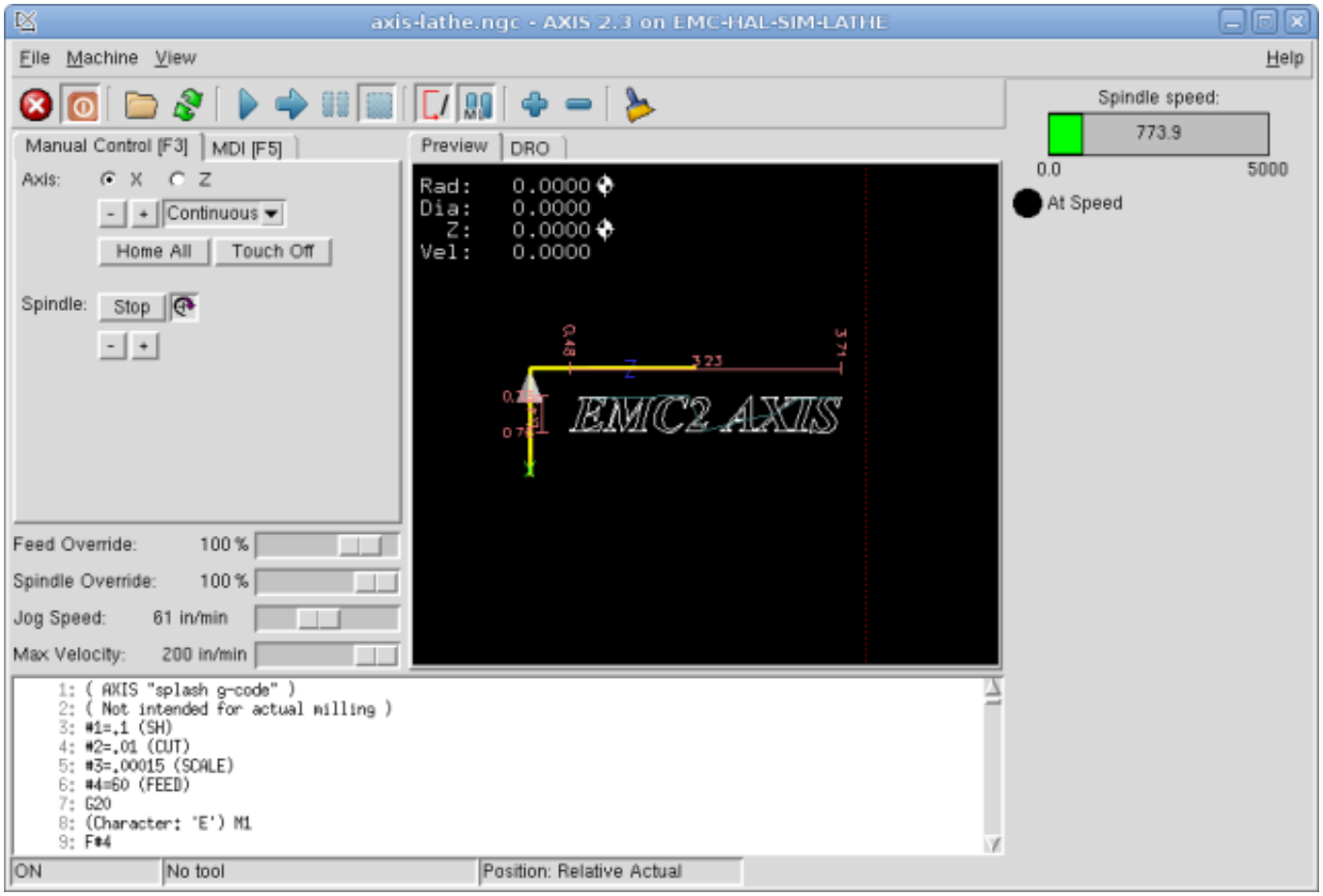
To make our widget actually display the spindle-speed it needs to be hooked up to the appropriate HAL signal. A HAL file that will be run once AXIS and PyVCP have started can be specified in the [HAL] section of the INI file:

```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

This change will run the HAL commands specified in *spindle_to_pyvcp.hal*. In our example the contents could look like this:

```
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* - makes HAL wait for the component *panelname* to finish loading (*become ready* in HAL speak) before processing more HAL commands. This is important because PyVCP panels export HAL pins, and other HAL components will need them present to connect to them. Note the capital W and lowercase n. If you use the -Wn option you must use the -c option to name the panel.
- *pyvcp < -g> < -c> panel.xml* - builds the panel with the optional geometry and/or panelname from the XML panel file. The panel.xml can be any name that ends in .xml. The .xml file is the file that describes how to build the panel. You must add the path name if the panel is not in the directory that the HAL script is in.
- *-g <WxH><+X+Y>* - specifies the geometry to be used when constructing the panel. The syntax is *Width x Height + X Anchor + Y Anchor*. You can set the size or position or both. The anchor point is the upper left corner of the panel. An example is -g 250x500+800+0 This sets the panel at 250 pixels wide, 500 pixels tall, and anchors it at X800 Y0.
- *-c panelname* - tells PyVCP what to call the component and also the title of the window. The panelname can be any name without spaces.

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.

The loadusr command is used when you also load a component that will stop HAL from closing until it’s done. If you loaded a panel and then loaded Classic Ladder using *loadusr -w classicladder*, CL would hold HAL open (and the panel) until you closed CL. The *-Wn* above means wait for the component *-Wn ”name”* to become ready. (*name* can be any name. Note the capital W and lowercase n.) The -c tells PyVCP to build a panel with the name *panelname* using the info in *panel_file_name.xml*. The name *panel_file_name.xml* can be any name but must end in .xml - it is the file that describes how to build the panel. You must add the path name if the panel is not in the directory that the HAL script is in.

An optional command to use if you want the panel to stop HAL from continuing commands / shutting down. After loading any other components you want the last HAL command to be:

```
waitusr panelname
```

This tells HAL to wait for component *panelname* to close before continuing HAL commands. This is usually set as the last command so that HAL shuts down when the panel is closed.

### 12.1.6 Widgets

HAL signals come in two variants, bits and numbers. Bits are off/on signals. Numbers can be *float*, *s32* or *u32*. For more information on HAL data types see the HAL Data section. The PyVCP widget can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of PyVCP widgets that you can connect to a HAL signal. A fifth class of helper widgets allow you to organize and label your panel.

- Widgets 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

Each widget is described briefly, followed by the markup used, and a screen shot. All tags inside the main widget tag are optional.

#### 12.1.6.2 Allgemeine Anmerkungen

At the present time, both a tag-based and an attribute-based syntax are supported. For instance, the following XML fragments are treated identically:

```
<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** Edit the XML file with a text editor. In most cases you can right click on the file and select *open with text editor* or similar.

**Farben**

Colors can be specified using the X11 rgb colors by name *gray75* or hex *#0000ff*. A complete list is located here https://sedition.com/perl/rgb.html.

Gebräuchliche Farben (Nummern bezeichnen Schattierungen der jeweiligen Farbe)

- white (engl. für weiß)
- black (engl. für schwarz)
- blue (engl. für blau) und blue1 - 4
- cyan und cyan1 - 4
- green (engl. für grün) und green1 - 4
- yellow (engl. für gelb) und yellow1 - 4
- red (engl. für rot) und red1 - 4
- purple (engl. für violett) und purple1 - 4
- gray (amerikanisch für grau) und gray0 - 100

**HAL-Pins** HAL-Pins bieten die Möglichkeit, das Widget mit etwas zu ”verbinden”. Sobald Sie einen HAL-Pin für Ihr Widget erstellt haben, können Sie ihn mit einem *net*-Befehl in einer .hal-Datei mit einem anderen HAL-Pin *verbinden*. Für weitere Informationen über den *net*-Befehl siehe den Abschnitt HAL Befehle.

#### 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.
- *<font>(”Helvetica”,20)</font>* - Schriftart und -größe des Textes angeben.
- *<relief>FLAT</relief>* - Angabe des Rahmens um das Etikett (*FLAT*, *RAISED*, *SUNKEN*) Standard ist *FLAT*.
- *<bd>*n*</bd>* - wobei *n* die Breite des Rahmens ist, wenn *RAISED* oder *SUNKEN* verwendet wird.
- *<padx>*n*</padx>* - wobei *n* die Menge des zusätzlichen horizontalen Raums ist.
- *<pady>*n*</pady>* - wobei *n* für die Anzahl der zusätzlichen vertikalen Leerzeichen steht.

Das Etikett hat einen optionalen Deaktivierungsstift, der erstellt wird, wenn Sie *<disable_pin>True</disable* 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

A LED is used to indicate the status of a *bit* halpin. The LED color will be on_color when the halpin is true, and off_color otherwise.

- *<led></led>* - erzeugt eine runde LED
- *<rectled></rectled>* - erzeugt eine rechteckige LED
- *<halpin>*name*</halpin>* - Name des Pins, Standard ist *led*.n, wobei *n* eine ganze Zahl ist, die für jede LED inkrementiert wird.
- *<size>*n*</size>* - *n* ist die Größe der LED in Pixeln, Standardwert ist 20.

- *<on_color>*color*</on_color>* - sets the color of the LED to *color* when the pin is true. Default is *green*. See section on colors for more info.
- *<off_color>*color*</off_color>* - sets the color of the LED to *color* when the pin is false. Default is *red*.
- *<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:

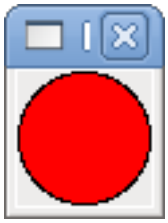


Abbildung 12.4: Beispiel für eine runde LED

**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

A button is used to control a BIT pin. The pin will be set True when the button is pressed and held down, and will be set False when the button is released. Buttons can use the following optional options.

- <*padx*>n</*padx*> - wobei *n* die Menge des zusätzlichen horizontalen Raums ist.
- <*pady*>n</*pady*> - wobei *n* für die Anzahl der zusätzlichen vertikalen Leerzeichen steht.
- <*activebackground*>”color”</*activebackground*> - der Cursor über Farbe auf *color* gesetzt.
- <*fg*>”color”</*fg*> - die Vordergrundfarbe wird auf *color* gesetzt.
- <*bg*>”color”</*bg*> - die auf *color* gesetzte Hintergrundfarbe.
- <*disable_pin*>*True*</*disable_pin*> - Pin deaktivieren.

**Text Button** A text button controls a *bit* halpin. The halpin is false until the button is pressed then it is true. The button is a momentary button.

Die Text Button hat einen optionalen Deaktivierungsstift, der beim Hinzufügen von <disable_pin>True</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** A checkbutton controls a bit halpin. The halpin will be set True when the button is checked, and false when the button is unchecked. The checkbutton is a toggle type button. The checkbuttons may be set initially as TRUE or FALSE the initval field A pin called changepin is also created automatically, which can toggle the Checkbutton via HAL, if the value linked is changed, to update the display remotely.

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:

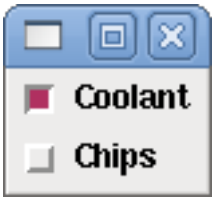



Abbildung 12.9: Einfaches Checkbutton-Beispiel

Das Kontrollkästchen für das Kühlmittel ist aktiviert.
Beachten Sie die zusätzlichen Leerzeichen im Text ”Chips”, um die Kontrollkästchen auszurichten.

**Radiobutton** A radiobutton will set one of the halpins true. The other pins are set false. The initval field may be set to choose the default selection when the panel displays. Only one radio button may be set to TRUE (1) or only the highest number pin set TRUE will have that value.

```
<radiobutton>
    <choices>[”one”,”two”,”three”]</choices>
    <halpin>”my-radio”</halpin>
    <initval>0</initval>
</radiobutton>
```

Der obige Code ergab dieses Beispiel:

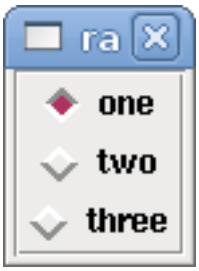



Abbildung 12.10: Einfaches Radiobutton-Beispiel

Note that the HAL pins in the example above will be named my-radio.one, my-radio.two, and my-radio.three. In the image above, *one* is the selected value. Use the tag `<orient>HORIZONTAL</orient>` to display horizontally.

#### 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>* - is a Tkinter font type and size specification. One font that will show up to at least size 200 is *courier 10 pitch*, so for a really big Number widget you could specify:

```
<font>("courier 10 pitch",100)</font>
```

- *<format>* - ist ein angegebenes Format im C-Stil, das bestimmt, wie die Zahl angezeigt wird.

**s32-Nummer** The s32 number widget displays the value of a s32 number. The syntax is the same as *number* except the name which is <s32>. Make sure the width is wide enough to cover the largest number you expect to use.

```
<s32>
    <halpin>”my-number”</halpin>
    <font>(”Helvetica”,24)</font>
    <format>”6d”</format>
    <width>6</width>
</s32>
```

Der obige Code ergab dieses Beispiel:

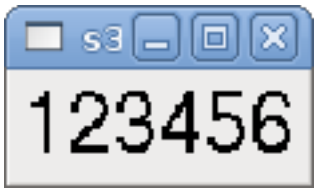



Abbildung 12.12: Einfaches s32 Zahlenbeispiel

**u32 Zahl** The u32 number widget displays the value of a u32 number. The syntax is the same as *number* except the name which is <u32>.

**Bar (engl. für Balken)** A bar widget displays the value of a FLOAT signal both graphically using a bar display and numerically. The color of the bar can be set as one color throughout its range (default using fillcolor) or set to change color, dependent upon the value of the halpin (range1, range2 range3 must all be set, if you only want 2 ranges, set 2 of them to the same color).

- `<halpin>”my-bar”</halpin>` text, setzt den Pin-Namen: `pyvcp.my-bar`.
- `<min_>0</min_>` Zahl, legt die minimale Skalierung fest.
- `<max_>140</max_>` (Zahl), legt die maximale Skalierung fest.
- `<format>”3.1f”</format>` (Text), setzt das Zahlenformat mit Python-Zahlenformatierung.
- `<bgcolor>”grau”</bgcolor>` (Text), legt die Hintergrundfarbe fest.
- `<fillcolor>”rot”</fillcolor>` (Text), setzt die Füllfarbe.
- `<range1>0,100, ”grün”</range1>` (Zahl, Zahl, Text), legt den ersten Bereich und die Farbe fest.
- `<range2>101,135, ”orange”</range2>` (number, number, text), legt den ersten Bereich und die Farbe fest.
- <range3>136, 150, ”rot”</range3> Zahl, Zahl, Text, legt den ersten Bereich und die Farbe fest.
- `<canvas_width>200</canvas_width>`, (Zahl), legt die Gesamtbreite fest.
- `<canvas_height>50</canvas_height>` (Zahl), legt die Gesamthöhe fest.
- `<bar_height>30</bar_height>` (Zahl), legt die Balkenhöhe fest, muss kleiner sein als canvas_height.
- `<bar_width>150</bar_width>` (Zahl), setzt die Balkenbreite, muss kleiner als canvas_width sein.

```
<bar>
    <halpin>”my-bar”</halpin>
    <min_>0</min_>
    <max_>123</max_>
    <format>”3.1f”</format>
    <bgcolor>”grey”</bgcolor>
    <fillcolor>”red”</fillcolor>
    <range1>0,100,”green”</range1>
    <range2>101,135,”orange”</range2>
    <range3>136, 150,”red”</range3>
    <canvas_width>200</canvas_width>
    <canvas_height>50</canvas_height>
    <bar_height>30</bar_height>
    <bar_width>150</bar_width>
</bar>
```

Der obige Code ergab dieses Beispiel:

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:

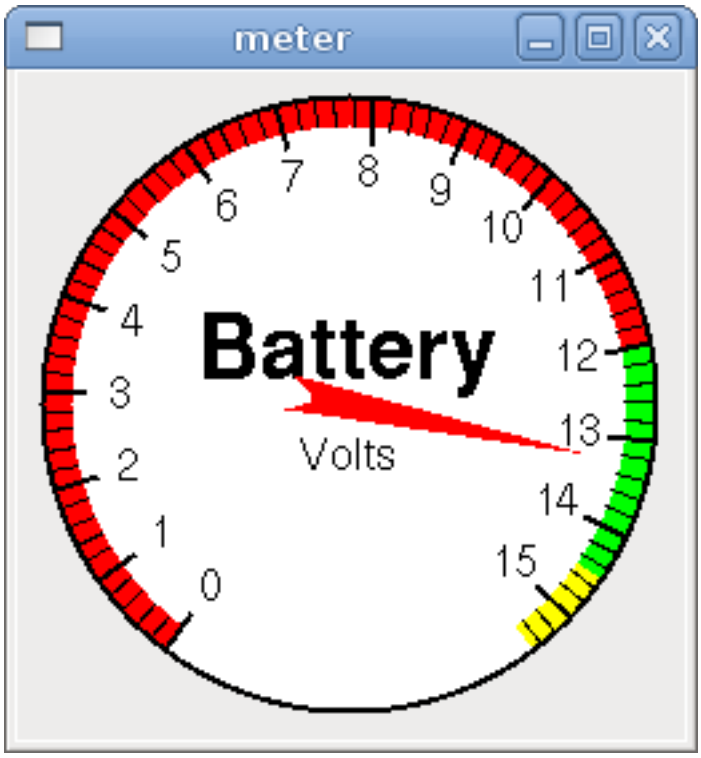



Abbildung 12.14: Einfaches Beispiel für ein Messgerät

#### 12.1.6.8 Numerische Eingaben

**Spinbox (Einstellrad)** A spinbox controls a FLOAT pin. You increase or decrease the value of the pin by either pressing on the arrows, or pointing at the spinbox and rolling your mouse-wheel. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<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** A scale controls a float or a s32 pin. You increase or decrease the value of the pin be either dragging the slider, or pointing at the scale and rolling your mouse-wheel. The *halpin* will have both *-f* and *-i* added to it to form the float and s32 pins. Width is the width of the slider in vertical and the height of the slider in horizontal orientation. If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<scale>
    <font>("Helvetica",16)</font>
    <width>"25"</width>
    <halpin>"my-hscale"</halpin>
    <resolution>0.1</resolution>
    <orient>HORIZONTAL</orient>
    <initval>-15</initval>
    <min_>-33</min_>
    <max_>26</max_>
    <param_pin>1</param_pin>
</scale>
<scale>
    <font>("Helvetica",16)</font>
    <width>"50"</width>
    <halpin>"my-vscale"</halpin>
    <resolution>1</resolution>
    <orient>VERTICAL</orient>
    <min_>100</min_>
    <max_>0</max_>
    <param_pin>1</param_pin>
</scale>
```

Der obige Code ergab dieses Beispiel:

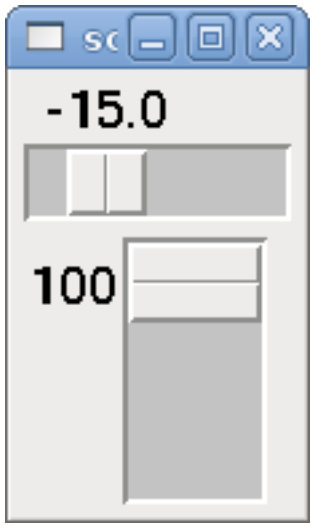



Abbildung 12.16: Beispiel für eine einfache Skalierung

**Anmerkung**
Beachten Sie, dass standardmäßig "min" angezeigt wird, auch wenn es größer als "max" ist, es sei denn, "min" ist negativ.

**Wählscheibe (engl. dial)** The Dial outputs a HAL float and reacts to both mouse wheel and dragging. Double left click to increase the resolution and double right click to reduce the resolution by one digit.

The output is capped by the min and max values. The <cpr> is how many tick marks are on the outside of the ring (beware of high numbers). If the param_pin field is set TRUE(1), a pin will be created that can be used to set the spinbox to an initial value and to remotely alter its value without HID input.

```
<dial>
    <size>200</size>
    <cpr>100</cpr>
    <min_>-15</min_>
    <max_>15</max_>
    <text>”Dial”</text>
    <initval>0</initval>
    <resolution>0.001</resolution>
    <halpin>”anaout”</halpin>
    <dialcolor>”yellow”</dialcolor>
    <edgecolor>”green”</edgecolor>
    <dotcolor>”black”</dotcolor>
    <param_pin>1</param_pin>
</dial>
```

Der obige Code ergab dieses Beispiel:

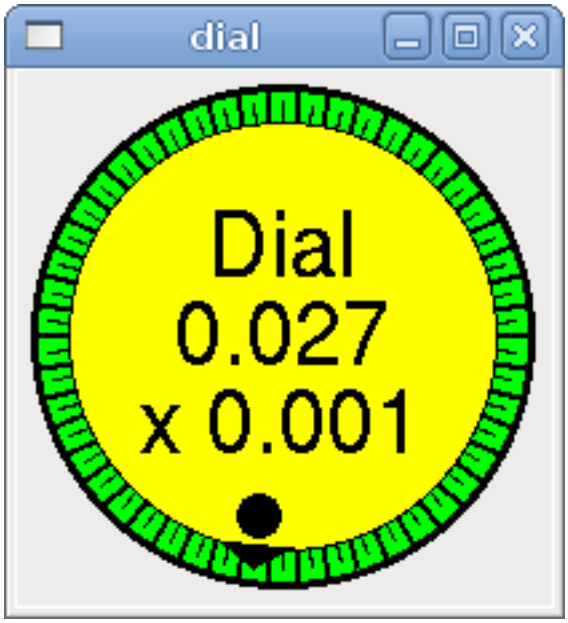



Abbildung 12.17: Einfaches Wählbeispiel

**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>* creates scale text and a `FLOAT.scale` pin to display jog scale * *<clear_pin>1</clear_pin>* creates `DRO` and a `BIT.reset` pin to reset DRO. Needs scale_pin for scaled DRO. Shift+click resets DRO also

```
<jogwheel>
    <halpin>”my-wheel”</halpin>
    <cpr>45</cpr>
    <size>250</size>
</jogwheel>
```

Der obige Code ergab dieses Beispiel:

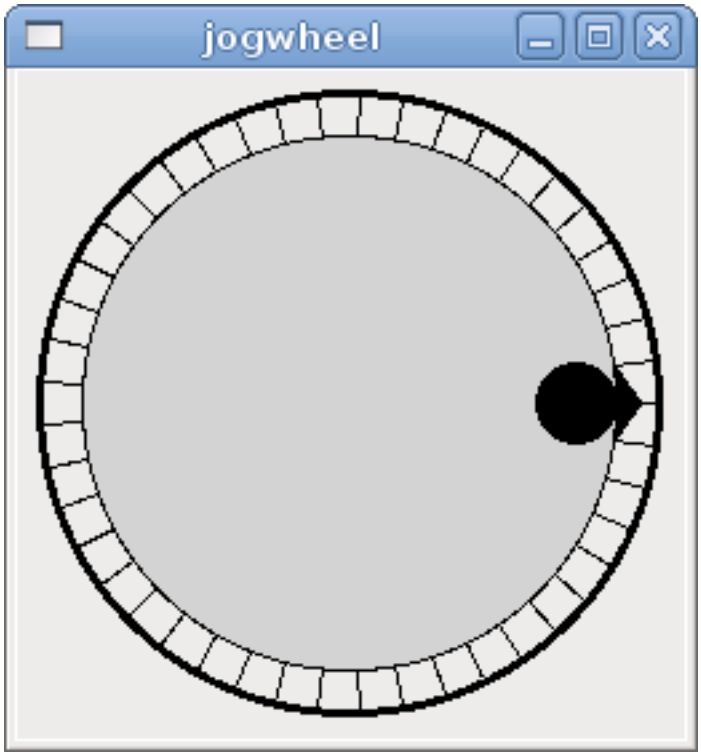



Abbildung 12.18: Einfaches Jogwheel-Beispiel

#### 12.1.6.9 Bilder

Image displays use only .gif image format. All of the images must be the same size. The images must be in the same directory as your INI file (or in the current directory if running from the command line with halrun/halcmd).

**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** The *image_u32* is the same as *image_bit*, except you have essentially an unlimited number of images and you *select* the image by setting the halpin to a integer value with 0 for the first image in the images list and 1 for the second image, etc.

```
<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)

Containers are widgets that contain other widgets. Containers are used to group other widgets.

**Begrenzungen (engl. borders)** Container borders are specified with two tags used together. The <relief> tag specifies the type of border and the <bd> specifies the width of the border.

***<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** Container fill are specified with the <*boxfill fill=””*/> tag. Valid entries are none, x, y and both. The x fill is a horizontal fill and the y fill is a vertical fill

**<boxfill fill =”style”/>**
Dabei ist *style* none, x, y oder beides. Standardwert ist x für Vbox und y für Hbox.

**Anker** Container anchors are specified with the <boxanchor anchor=””/> tag. The anchor specifies where to position each slave in its parcel. Valid entries are center, n, s, e, w, for center, north, south, east and west. Combinations like sw, se, nw and ne are also valid.

**<boxanchor anchor=”position”/>**
Dabei ist *Position* center, n, s, e, w, ne, nw, se oder sw. Standardwert ist Mitte.

**Erweitern** Container expand is specified with the boolean <boxexpand expand=””/> tag. Valid entries are ”yes”, ”no”.

**<boxexpand expand=”boolean”/>**
Where *boolean* is either ”yes” or ”no”. Default is yes.

**Hbox** Verwenden Sie eine Hbox, wenn Sie Widgets horizontal nebeneinander stapeln möchten.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>”a hbox:”</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

Der obige Code ergab dieses Beispiel:

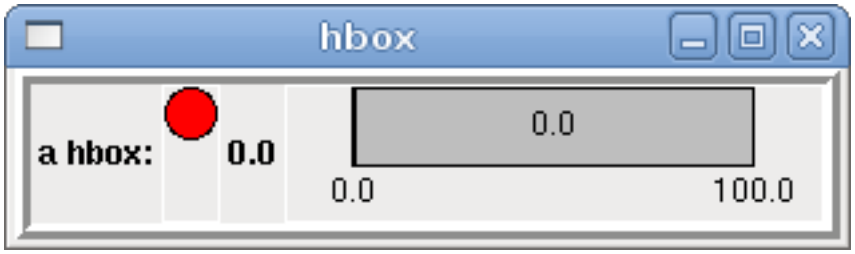



Abbildung 12.25: Einfaches hbox-Beispiel

Inside an Hbox, you can use the *<boxfill fill=””/>*, *<boxanchor anchor=””/>*, and *<boxexpand expand=””/>* tags to choose how items in the box behave when the window is re-sized. The default is *fill=”y”*, *anchor=”center”*, *expand=”yes”* for an 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:

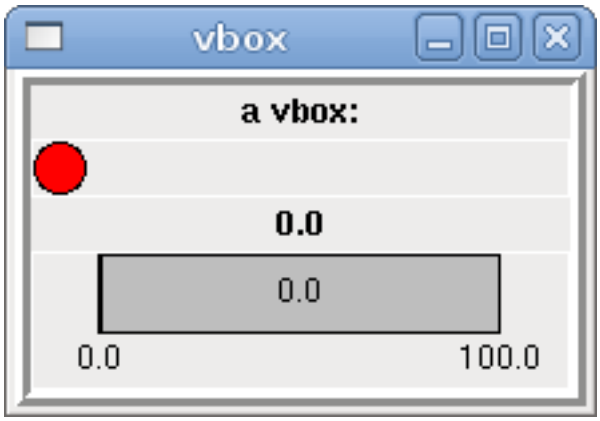



Abbildung 12.26: Einfaches vbox-Beispiel

Inside a Vbox, you can use the *<boxfill fill=””/>*, *<boxanchor anchor=””/>*, and *<boxexpand expand=””/>* tags to choose how items in the box behave when the window is re-sized. The default is *fill=”x”*, *anchor=”center”*, *expand=”yes”* for a Vbox.

**Etikettrahmen (engl. labelframe)** Ein Etikettenrahmen (engl. labelframe) ist ein Rahmen mit einer Rille und einem Etikett in der oberen linken Ecke.

```
<labelframe text=”Label: Leds groupées”>
```

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

Der obige Code ergab dieses Beispiel:

Abbildung 12.27: Einfaches Labelframe-Beispiel

**Tabelle (engl. table)** A table is a container that allows layout in a grid of rows and columns. Each row is started by a *<tablerow/>* tag. A contained widget may span rows or columns through the use of the *<tablespan rows= cols=/>* tag. The sides of the cells to which the contained widgets "stick" may be set through the use of the *<tablesticky sticky=/>* tag. A table expands on its flexible rows and columns.

**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:

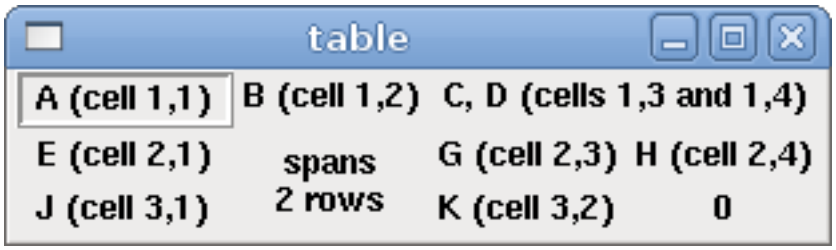
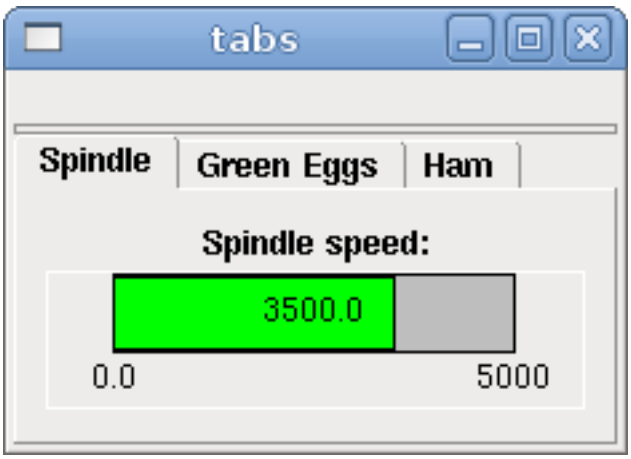



Abbildung 12.28: Beispiel für eine Tabelle

**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.



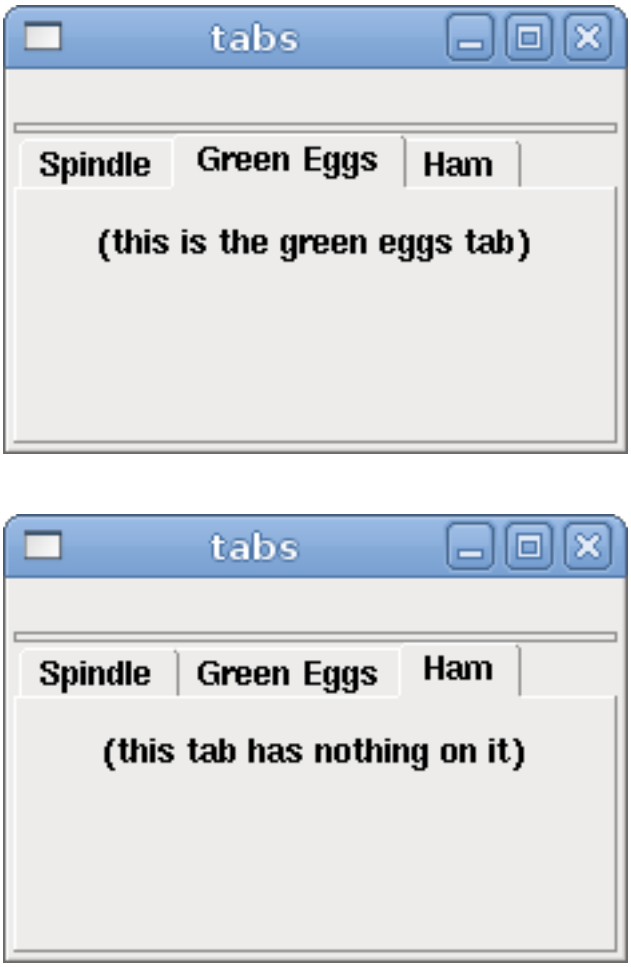



Abbildung 12.29: Einfache Tabs-Beispiel

## 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 this example we will create a PyVCP panel with jog buttons for X, Y, and Z. This configuration will be built upon a StepConf Wizard generated configuration. First we run the StepConf Wizard and configure our machine, on the Advanced Configuration Options page we then make a couple of selections to add a blank PyVCP panel as shown in the following figure. For this example we named the configuration *pyvcp_xyz* on the Basic Machine Information page of the StepConf Wizard.

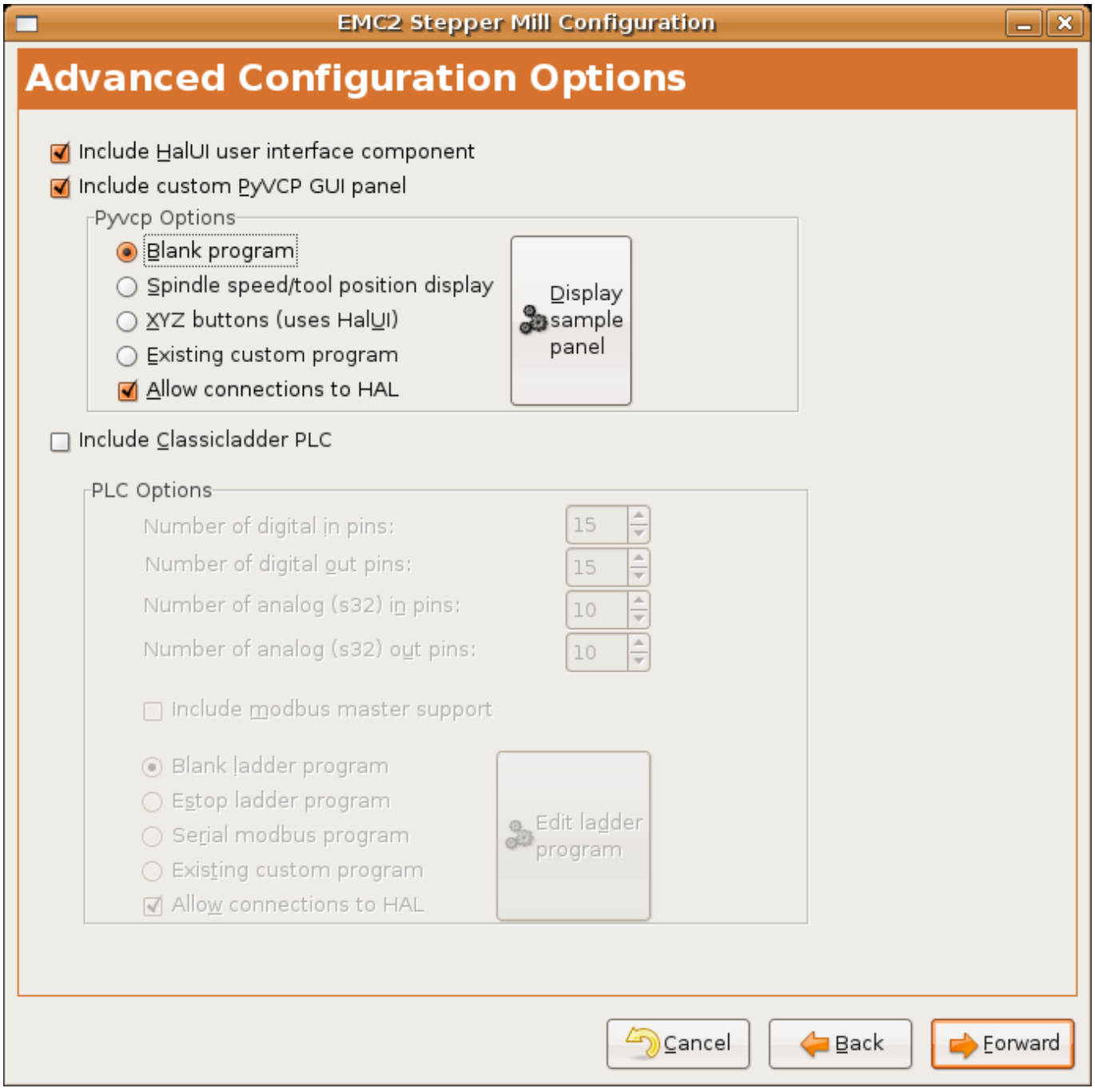



Abbildung 12.30: XYZ-Assistent Konfiguration

The StepConf Wizard will create several files and place them in the linuxcnc/configs/pyvcp_xyz directory. If you left the create link checked you will have a link to those files on your desktop.

#### 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.

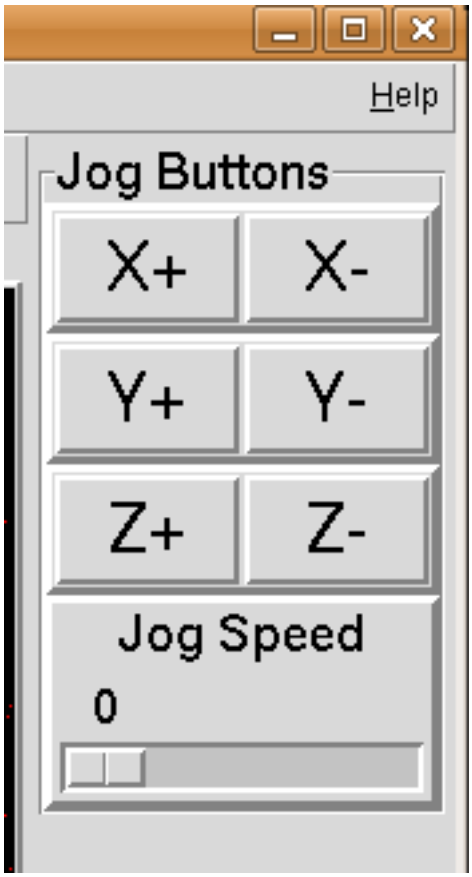
Schauen Sie in den Abschnitt PyVCP Widgets Referenz des Handbuchs für detailliertere Informationen über jedes Widget documentation des widgets.

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

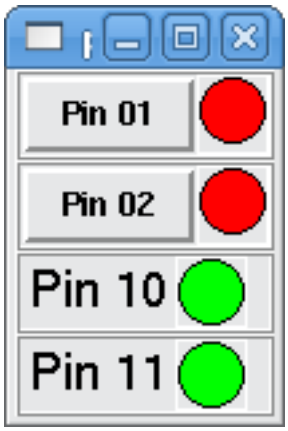
Dieses Beispiel zeigt Ihnen, wie Sie mit PyVCP und HAL einen einfachen Parallelport-Tester erstellen können.

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.



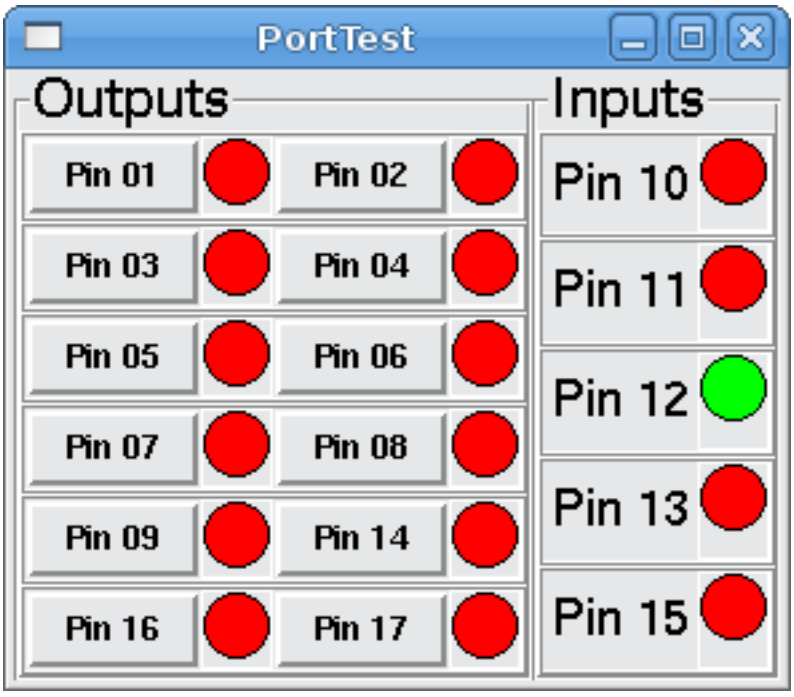

Um die HAL-Befehle auszuführen, die wir benötigen, um alles zum Laufen zu bringen, fügen wir Folgendes in unsere Datei ptest.hal ein.

```
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.

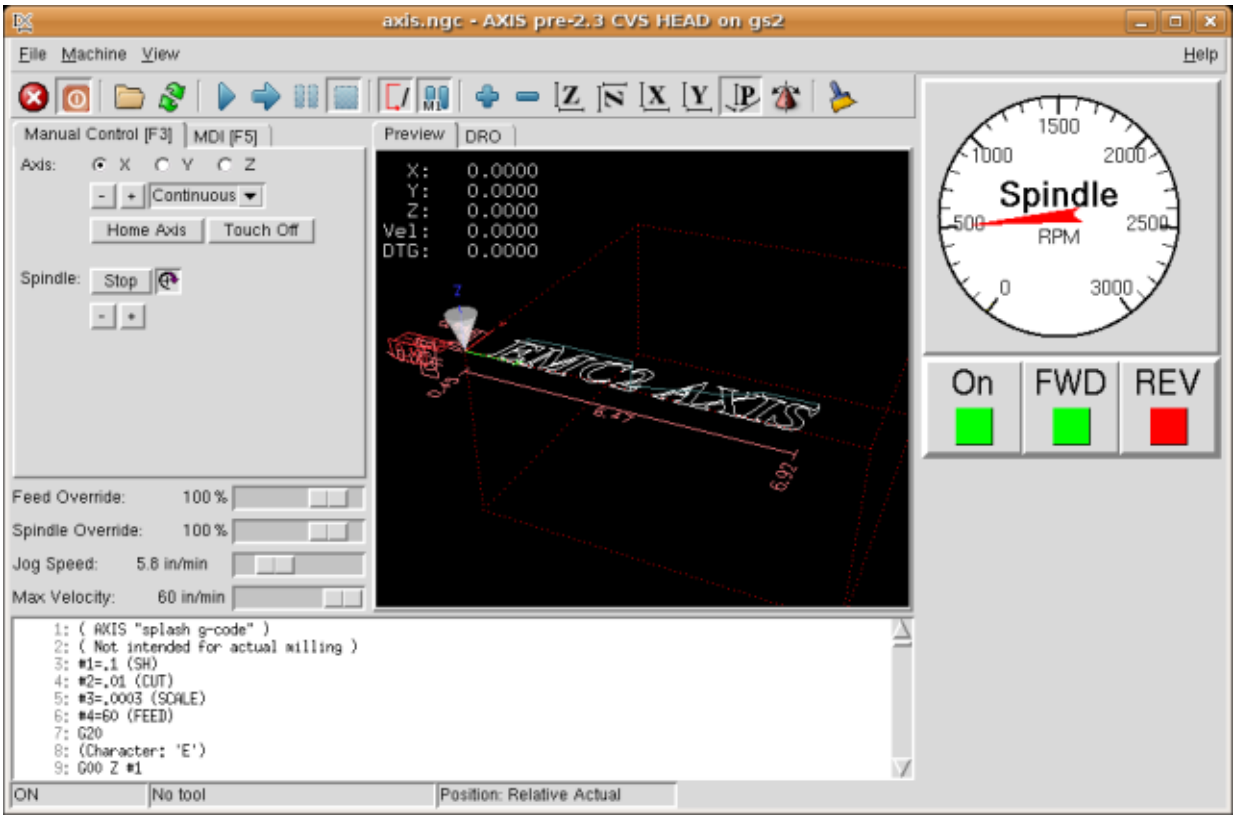
### 12.2.5.1 Das Panel

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

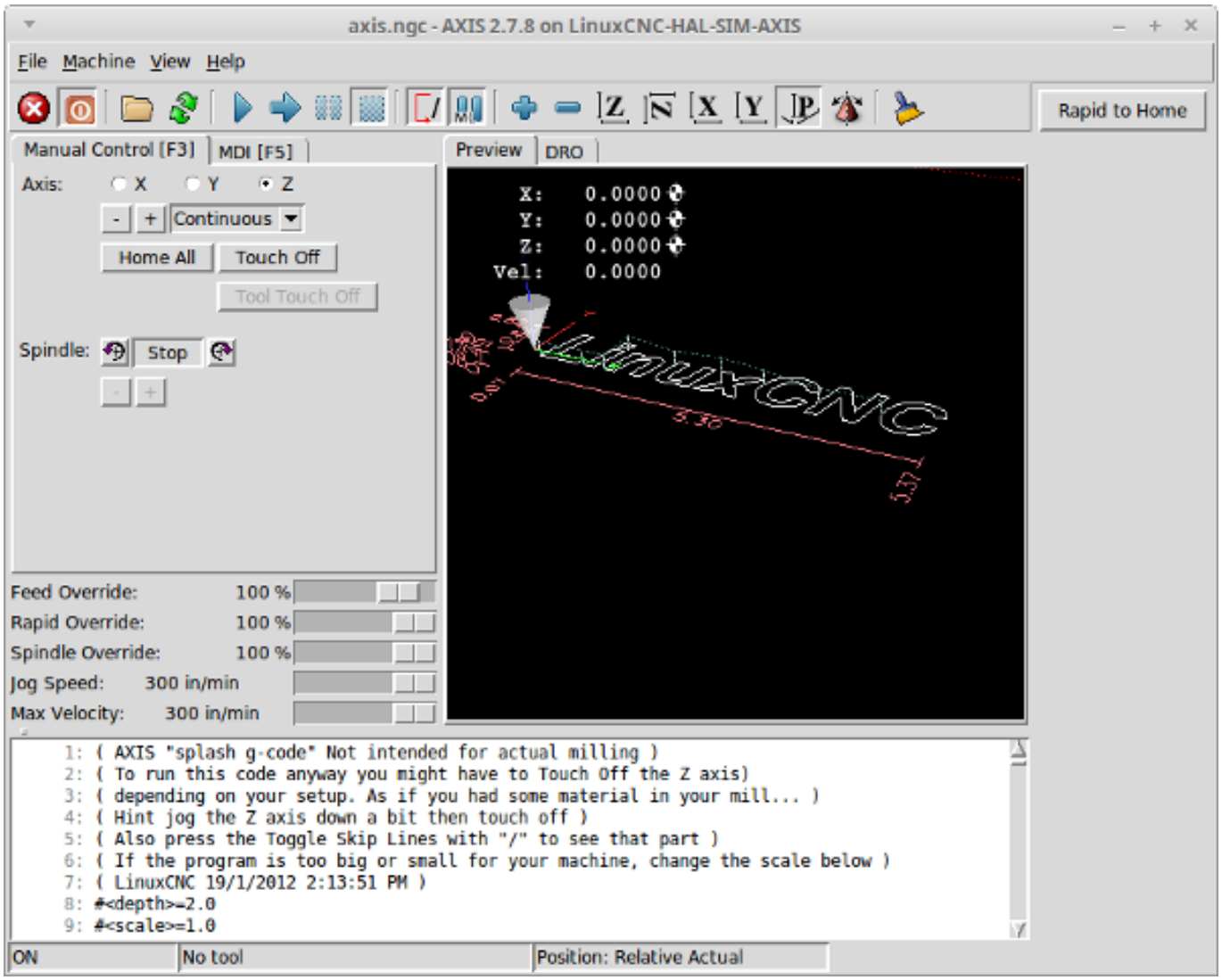
Damit das funktioniert, fügen wir den folgenden Code in die Datei custom_postgui.hal ein.

```
# 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 Virtual Control Panel

### 12.3.1 Was ist GladeVCP?

GladeVCP is a LinuxCNC component which adds the ability to add a new user interface panel to LinuxCNC user interfaces like:

- 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 uses the Glade WYSIWYG user interface editor, which makes it easy to create visually pleasing panels. It relies on the PyGObject bindings to the rich GTK3 widget set, and in fact all of these widgets may be used in a GladeVCP application - not just the specialized widgets for interacting with HAL and LinuxCNC, which are documented here.

#### 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: relies on the GTK3 widget set.
- User interface creation: uses the Glade WYSIWYG user interface editor.
- 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** The following instructions only apply if you're using a git checkout. Open a terminal and change to the directory created by git then issue the commands as shown.

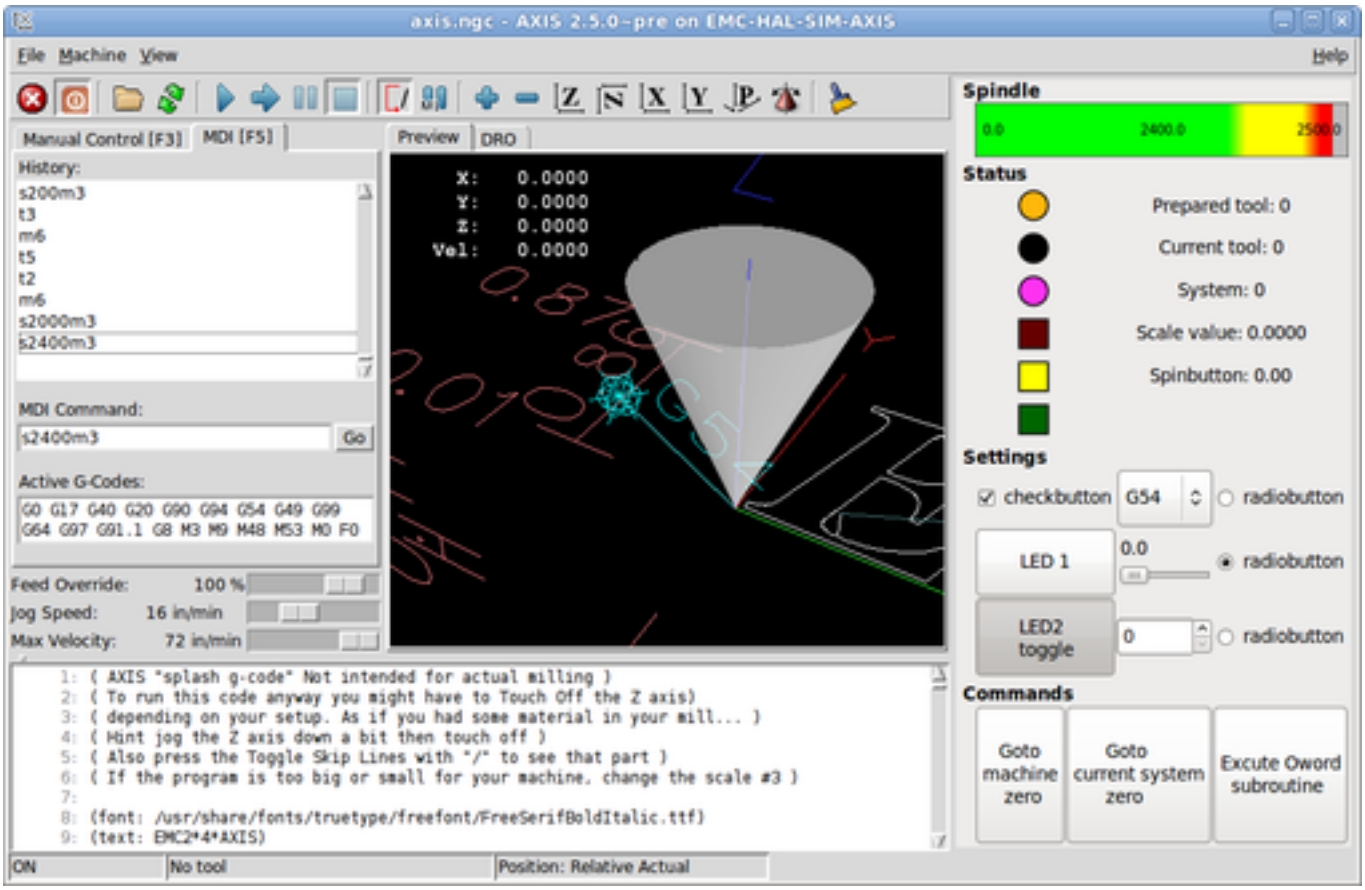
---

**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.

---

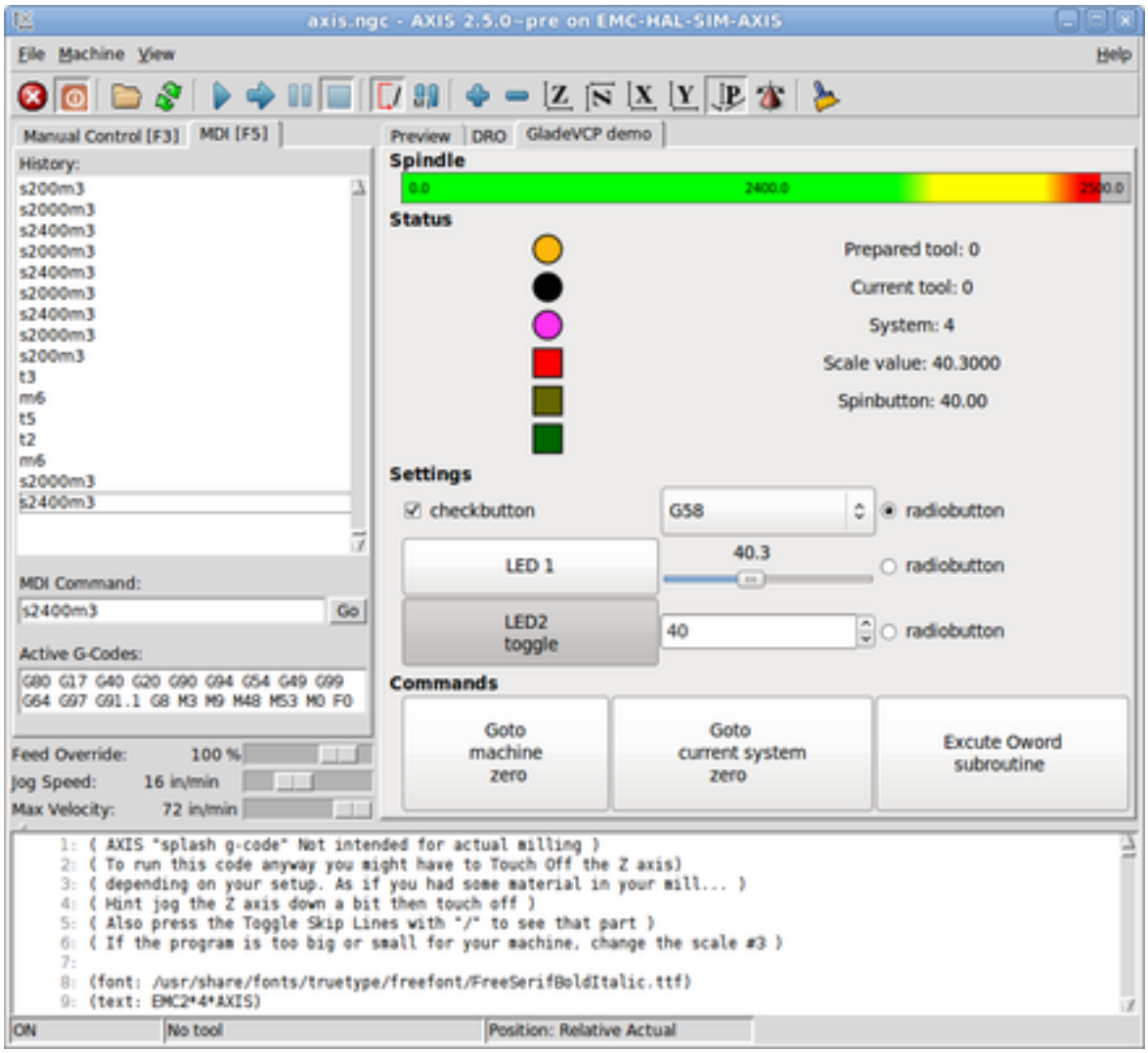
Führen Sie das in AXIS integrierte GladeVCP-Beispielpanel wie PyVCP wie folgt aus:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



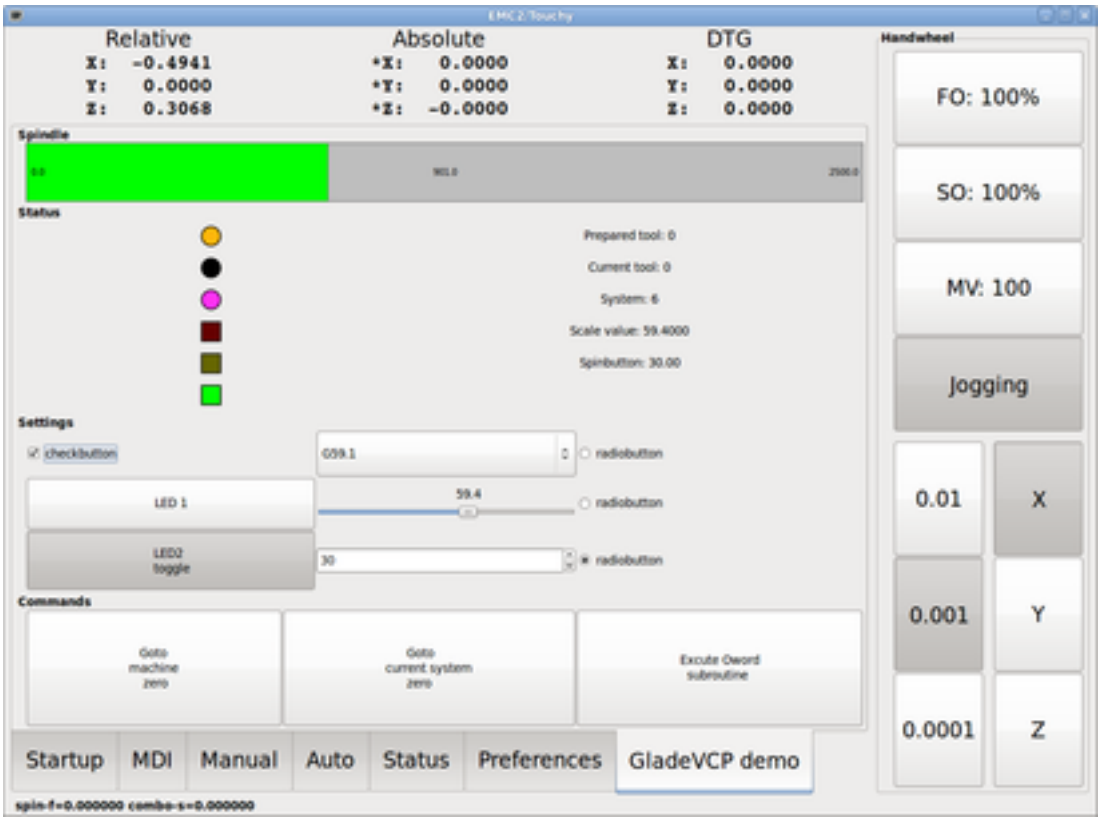

Führen Sie das gleiche Panel aus, aber als Registerkarte innerhalb von AXIS:

```
$ 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
```




Functionally these setups are identical - they only differ in screen real estate requirements and visibility. Since it is possible to run several GladeVCP components in parallel (with different HAL component names), mixed setups are possible as well - for instance a panel on the right hand side, and one or more tabs for less-frequently used parts of the interface.

#### 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*.

The example panel has two frames at the bottom. The panel is configured so that resetting ESTOP activates the Settings frame and turning the machine on enables the Commands frame at the bottom. The HAL widgets in the Settings frame are linked to LEDs and labels in the *Status* frame, and to the current and prepared tool number - play with them to see the effect. Executing the *T<toolnumber>* and *M6* commands in the MDI window will change the current and prepared tool number fields.

The buttons in the *Commands* frame are *MDI Action widgets* - pressing them will execute an MDI command in the interpreter. The third button *Execute Oword subroutine* is an advanced example - it takes several HAL pin values from the *Settings* frame, and passes them as parameters to the Oword subroutine. The actual parameters received by the routine are displayed by *(DEBUG, )* commands - see *../../nc_files/oword.ngc* for the subroutine body.

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

The user interface is created with the Glade UI editor - to explore it, you need to have Glade installed. To edit the user interface, run the command

```
$ glade configs/axis/gladevcp/manual-example.ui
```

Das erforderliche glade-Programm kann auf neueren Systemen den Namen glade-gtk2 tragen.

The center window shows the appearance of the UI. All user interface objects and support objects are found in the right top window, where you can select a specific widget (or by clicking on it in the center window). The properties of the selected widget are displayed, and can be changed, in the right bottom window.

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

To view or modify Glade UI files, you need Glade 3.38.2 or later installed - it is not needed just to run a GladeVCP panel. If the *glade* command is missing, install it with the command:

```
$ 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.

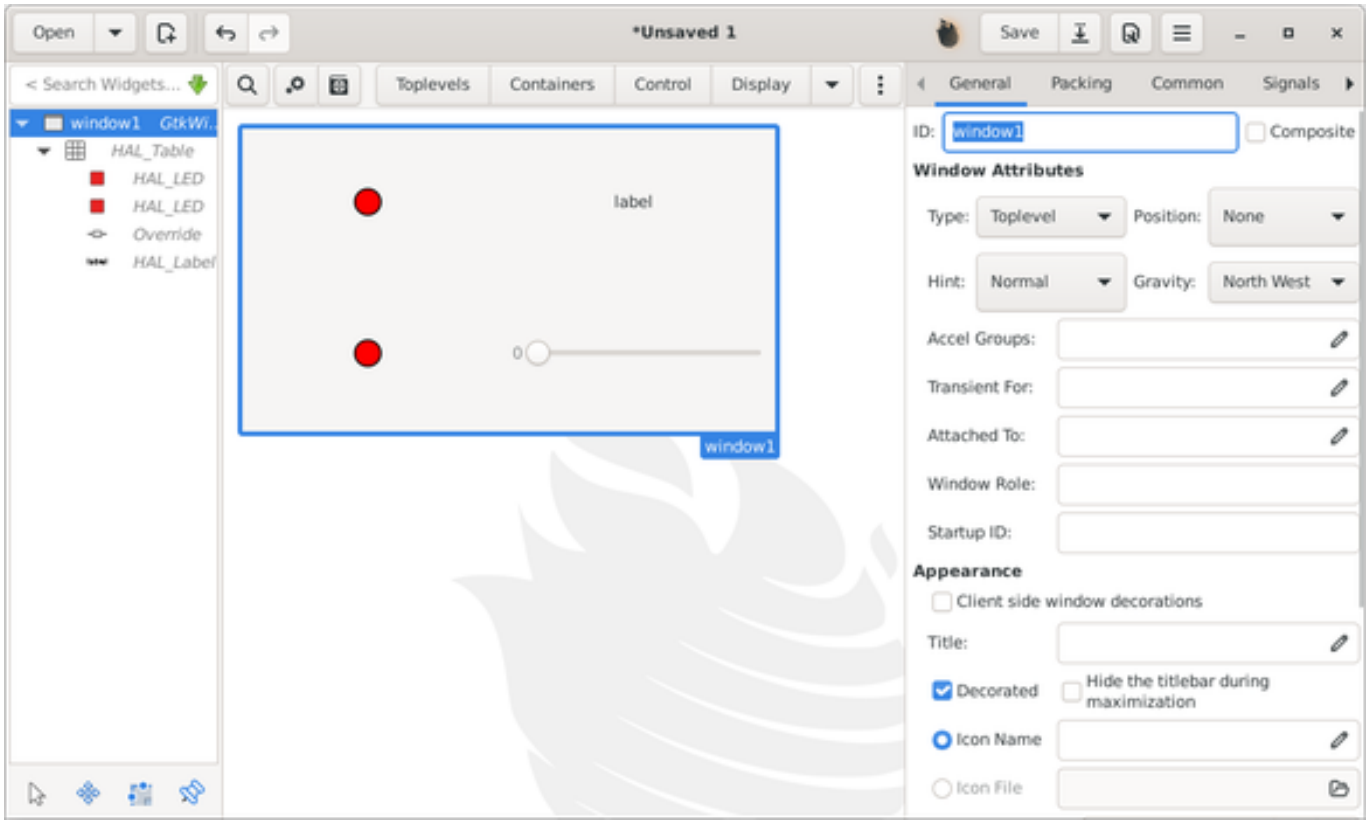
#### 12.3.3.2 Glade ausführen, um eine neue Benutzeroberfläche zu erstellen

This section just outlines the initial LinuxCNC-specific steps. For more information and a tutorial on Glade, see http://glade.gnome.org. Some Glade tips & tricks may also be found on 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.

- If LinuxCNC was not installed from a package, the LinuxCNC shell environment needs to be set up with `source <linuxcncdir>/scripts/rip-environment`, otherwise Glade won't find the LinuxCNC-specific widgets.
- Wenn Sie nach nicht gespeicherten Einstellungen gefragt werden, akzeptieren Sie einfach die Standardeinstellungen und klicken Sie auf „Schließen".
- From *Toplevels* (toolbar), pick *GtkWindow* (first entry) as top level window. Set *window1* as ID in the right pane under the tab *General*. This naming is important because GladeVCP relies on it.
- From the button with the three dots you can find the LinuxCNC specific widgets.
- Add a container like a HAL_Box or a HAL_Table from *HAL Python* to the frame.
- Pick and place some elements like LED, button, etc. within a container.

This will look like this:




Glade tends to write a lot of messages to the shell window, which mostly can be ignored. Select *File→Save as*, give it a name like *myui.ui* and make sure it is saved as *GtkBuilder* file (radio button left bottom corner in Save dialog). GladeVCP will also process the older *libglade* format correctly but there is no point in using it. The convention for GtkBuilder file extension is *.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

The suggested way of linking HAL pins in a GladeVCP panel is to collect them in a separate file with extension *.hal*. This file is passed via the `POSTGUI_HALFILE=` option in the `HAL` section of your INI file.

**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**
The file specifiers like ./hitcounter.py, ./manual-example.ui, etc. indicate that the files are located in the same directory as the INI file. You might have to copy them to you directory (alternatively, specify a correct absolute or relative path to the file(s)).

---

---

**Anmerkung**
The `[RS274NGC]SUBROUTINE_PATH=` option is only set so the example panel will find the Oword subroutine (oword.ngc) for the MDI Command widget. It might not be needed in your setup. The relative path specifier ../../nc_files/gladevcp_lib is constructed to work with directories copied by the configuration picker and when using a run-in-place setup.

---

#### 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.

Try it out by running AXIS - there should be a new tab called *GladeVCP demo* near the DRO tab. Select that tab, you should see the example panel nicely fit within AXIS.

---

**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**
The file specifiers like ./hitcounter.py, ./manual-example.ui, etc. indicate that the files are located in the same directory as the INI file. You might have to copy them to you directory (alternatively, specify a correct absolute or relative path to the file(s)).

---

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.
- There is no *POSTGUI_HALFILE*= INI option, but passing the HAL command file on the *EMBED_TAB_COMM* line is ok.
- The *halcmd loaduser -Wn …* incantation is not needed.

### 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 GEOMETRY::
    Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in pixel units, XOFFSET/YOFFSET  ↩
        is referenced from top left of screen.
    Use -g WIDTHxHEIGHT for just setting size or -g +XOFFSET+YOFFSET for just position

-H FILE::
    Execute HAL statements from FILE with halcmd after the component is set up and ready

-m MAXIMUM::
    Force panel window to maximize.
```

```
    Together with the -g geometry option one can move the panel to a second monitor and  ↩
        force it to use all of the screen

-t THEME::
    Set gtk theme. Default is system theme. Different panels can have different themes.

-x XID::
    Re-parent GladeVCP into an existing window XID instead of creating a new top level  ↩
        window

-u FILE::
    Dateien als zusätzliche benutzerdefinierte Module mit Handlern verwenden

-U USEROPT::
    übergibt USEROPTs an Python-Module
```

### 12.3.5 Den GladeVCP-Startvorgang verstehen

The integration steps outlined above look a bit tricky, and they are. It does therefore help to understand the startup process of LinuxCNC and how this relates to GladeVCP.

Der normale Startvorgang von LinuxCNC macht folgendes:

- Die Echtzeitumgebung wird gestartet.
- Alle HAL-Komponenten werden geladen.
- Die HAL-Komponenten sind durch die .hal cmd-Skripte miteinander verbunden.
- `task`, `iocontrol` und schließlich wird die Benutzeroberfläche gestartet.
- Vor der Einführung von GladeVCP wurde davon ausgegangen, dass zu dem Zeitpunkt, zu dem die Benutzeroberfläche gestartet wird, die gesamte HAL geladen, verkabelt und einsatzbereit ist.

Die Einführung von GladeVCP brachte das folgende Problem mit sich:

- GladeVCP-Panels müssen in ein übergeordnetes GUI-Fenster eingebettet werden.
- GladeVCP panels 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.

This is the purpose of the `POSTGUI_HALFILE`. This INI option is inspected by the GUIs. If a GUI detects this option, it runs the corresponding HAL file after any embedded GladeVCP panel is set up. However, it does not check whether a GladeVCP panel is actually used, in which case the HAL cmd file is just run normally. So if you do NOT start GladeVCP through `GLADEVCP` or `EMBED_TAB` etc, but later in a separate shell window or some other mechanism, a HAL command file in `POSTGUI_HALFILE` will be executed too early. Assuming GladeVCP pins are referenced herein, this will fail with an error message indicating that the GladeVCP HAL component is not available.

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 includes a collection of Gtk widgets with attached HAL pins called HAL Widgets, intended to control, display or otherwise interact with the LinuxCNC HAL layer. They are intended to be used with the Glade user interface editor. With proper installation, the HAL Widgets should show up in Glade’s *HAL Python* widget group. Many HAL specific fields in the Glade *General* section have an associated mouse-over tool tip.

HAL signals come in two variants, bits and numbers. Bits are off/on signals. Numbers can be ”float”, ”s32” or ”u32”. For more information on HAL data types see the HAL manual. The GladeVCP widgets can either display the value of the signal with an indicator widget, or modify the signal value with a control widget. Thus there are four classes of GladeVCP widgets that you can connect to a HAL signal. Another class of helper widgets allow you to organize and label your panel.

- Widgets 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 are instances of GtKWidgets and hence inherit the methods, properties and signals of the applicable GtkWidget class. For instance, to figure out which GtkWidget-related methods, properties and signals a *HAL_Button* has, lookup the description of GtkButton in the PyGObject API Reference.

An easy way to find out the inheritance relationship of a given HAL widget is as follows: Run glade, place the widget in a window, and select it; then choose the *Signals* tab in the *Properties* window. For example, selecting a *HAL_LED* widget, this will show that a *HAL_LED* is derived from a *GtkWidget*, which in turn is derived from a *GtkObject*, and eventually a *GObject*.

Full class hierarchy can be seen by invoking the GtkInspector while in the Glade GUI by selecting a widget then pressing Control-Shift-I. If the Inspector doesn't open then it can be enabled from a terminal by entering:

```
gsettings set org.gtk.Settings.Debug enable-inspector-keybinding true
```

The Inspector is also handy for testing css style changes "on the fly" as well as determining all the properties and signals available for a widget.

HAL-Widgets haben auch einige HAL-spezifische Python-Attribute:

**hal_pin**
: The underlying HAL pin Python object in case the widget has a single pin type

**hal_pin_s, hal_pin_f**
: The s32 and float pins of the *HAL_Spinbutton* and *HAL_ComboBox* widgets - note these widgets do not have a *hal_pin* attribute!

**hal_pin_scale**
: The float input pin of *HAL_ProgressBar* widget representing the maximum absolute value of input.

Es gibt mehrere HAL-spezifische Methoden von HAL Widgets, aber die einzige relevante Methode ist:

**<halpin>.get()**
: Abrufen des Wertes des aktuellen HAL-Pins, wobei *<halpin>* der oben aufgeführte zutreffende HAL-Pinname ist.

#### 12.3.6.3 Einstellung von Pin- und Widget-Werten

As a general rule, if you need to set a HAL output widget's value from Python code, do so by calling the underlying Gtk *setter* (e.g., `set_active()`, `set_value()`). Do not try to set the associated pin's value by `halcomp[pinname] = value` directly because the widget will not take notice of the change!

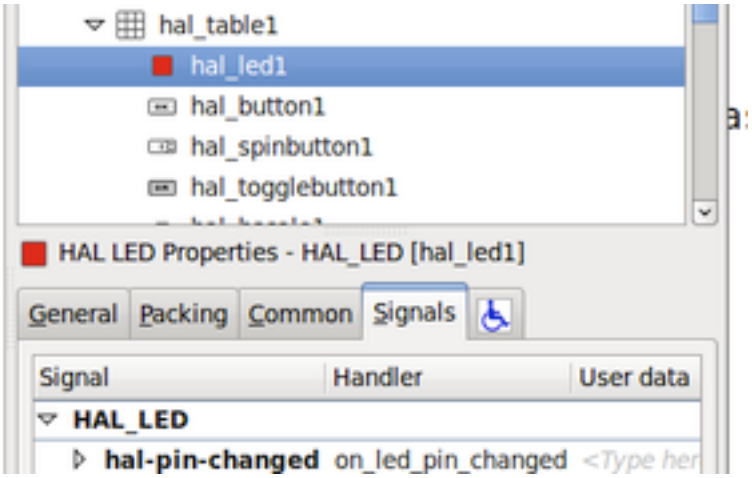
It might be tempting to *set HAL widget input pins* programmatically. Note this defeats the purpose of an input pin in the first place - it should be linked to, and react to signals generated by other HAL components. While there is currently no write protection on writing to input pins in HAL Python, 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

Event-driven programming means that the UI tells your code when ”something happens” - through a callback, like when a button was pressed. The output HAL widgets (those which display a HAL pin’s value) like LED, Bar, VBar, Meter, etc., support the `hal-pin-changed` signal, which may cause a callback into your Python code when - well, a HAL pin changes its value. This means there’s no more need for permanent polling of HAL pin changes in your code, the widgets do that in the background and let you know.

Here is an example how to set a `hal-pin-changed` signal for a HAL_LED in the Glade UI editor:




Das Beispiel in `configs/apps/gladevcp/complex` zeigt, wie dies in Python gehandhabt wird.

#### 12.3.6.5 Buttons

This group of widgets are derived from various Gtk buttons and consists of HAL_Button, HAL_ToggleButton, HAL_RadioButton and CheckButton widgets. All of them have a single output BIT pin named identical to the widget. Buttons have no additional properties compared to their base Gtk classes.

- 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`

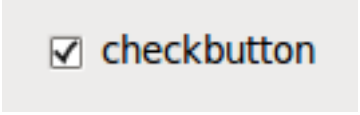



Abbildung 12.31: Checkbutton

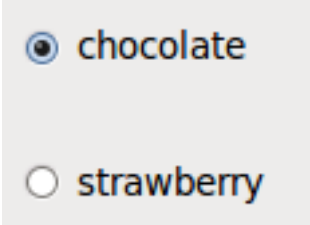



Abbildung 12.32: Radiobuttons

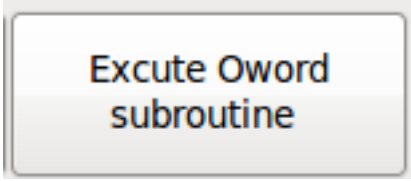



Abbildung 12.33: Toggle-Button

---

**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

To make a scale useful in Glade, add an *Adjustment* (General → Adjustment → New or existing adjustment) and edit the adjustment object. It defines the default/min/max/increment values. Also, set adjustment *Page size* and *Page increment* to zero to avoid warnings.

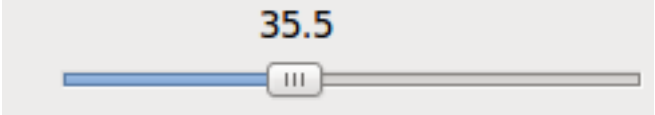



Abbildung 12.34: Beispiel HAL_HScale

#### 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.

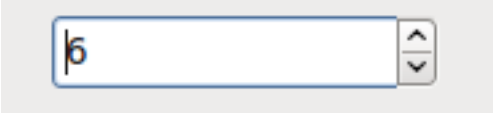



Abbildung 12.35: Beispiel SpinButton

#### 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` exports its count value as 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.

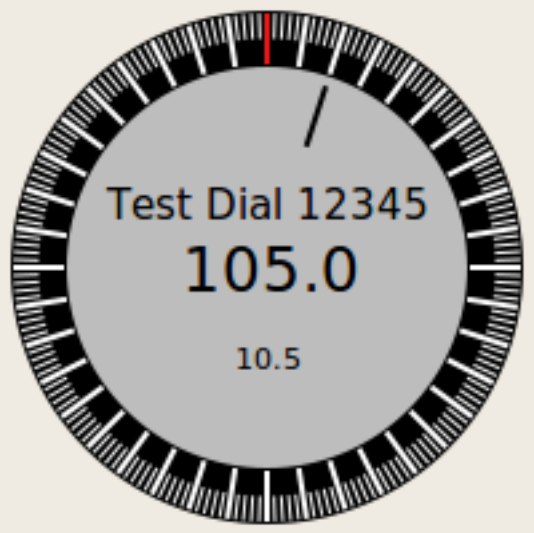



Abbildung 12.36: Beispiel Hal_Dial

#### 12.3.6.9 Jog-Handrad (engl. jog wheel)

The `jogwheel` widget simulates a real jogwheel. It can be operated with the mouse. You can just use the mouse wheel, while the mouse cursor is over the JogWheel widget, or you push the left mouse button and move the cursor in circular direction to increase or degrease the counts.

- 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` exports its count value as 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”

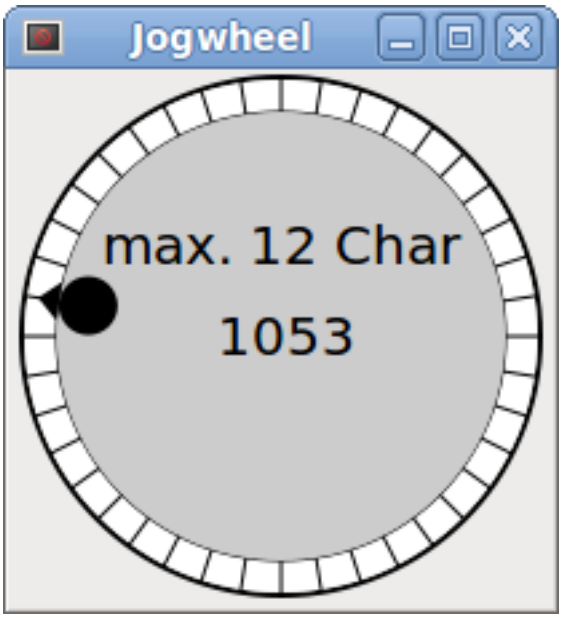



Abbildung 12.37: Beispiel JogWheel

#### 12.3.6.10 Geschwindigkeitsregelung

`speedcontrol` is a widget specially made to control an adjustment with a touch screen. It is a replacement to the normal scale widget which is difficult to slide on a touch screen.

The value is controlled with two button to increase or decrease the value. The increment will change as long a button is pressed. The value of each increment as well as the time between two changes can be set using the widget properties.

*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)
```

You can assign a existing adjustment to the control, that way it is easy to replace existing sliders without many code changes. Be aware, that after changing the adjustment you may need to set a new increment, as it will be reset to its default (100 steps from MIN to 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

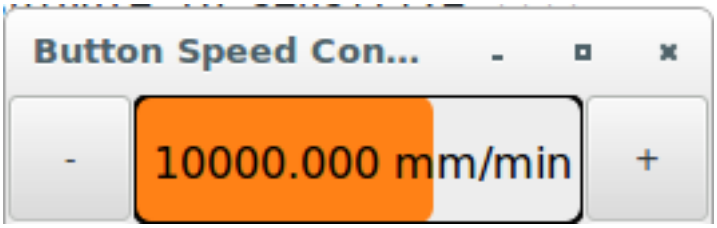



Abbildung 12.38: Beispiel Speedcontrol

#### 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**
The pin's HAL type (0:s32, 1:float, 2:u32), see also the tooltip on *General→HAL pin type* (note this is different from PyVCP which has three label widgets, one for each type).

**text_template**
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 its child widgets are hidden or not.

**Pin: , <Panel_basename>.<widgetname>**
in Bit-Pin
Wenn der Pin niedrig ist, sind die untergeordneten Widgets sichtbar, was der Standardzustand ist.

**HAL_Table und 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
Wenn der Pin niedrig ist, sind die untergeordneten Widgets inaktiv, was der Standardzustand ist.

**State_Sensitive_Table**
Reagiert auf den Zustand der LinuxCNC's Interpreter.
Optional auswählbar, um auf *must-be-all-homed*, *must-be-on* und *must-idle* zu reagieren.
Sie können sie kombinieren. Bei Notaus (engl. E-stop) ist es immer unempfindlich.
(Hat keinen Pin).

**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

The `hal_led` simulates a real indicator LED.
It has a single input BIT pin which controls its state: ON or OFF.

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**
If set and LED is ON then it is blinking.
Blink period is equal to ”led_blink_rate” specified in milliseconds.

**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.

As an input widget, LED also supports the `hal-pin-changed` signal. If you want to get a notification in your code when the LED’s HAL pin was changed, then connect this signal to a handler, for example `on_led_pin_changed` and provide the handler as follows:

```
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.

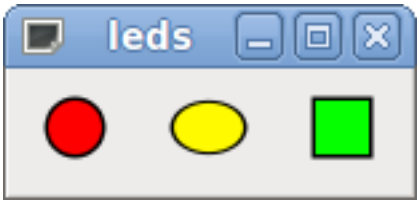



Abbildung 12.39: Beispiel LEDs

#### 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.

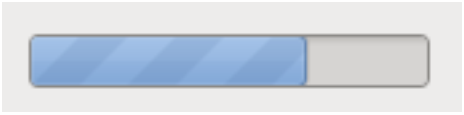


Abbildung 12.40: Beispiel HAL_ProgressBar

#### 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.

In default mode this widgets sets the pins to the index of the chosen list entry. So if your widget has three labels, it may only assume values 0,1 and 2.

In column mode (column > -1), the value reported is chosen from the ListStore array as defined in Glade. So typically your widget definition would have two columns in the ListStore, one with text displayed in the dropdown, and an int or float value to use for that choice.

Es gibt ein Beispiel in `configs/apps/by-widget/combobox. {py,ui}`, das den Spaltenmodus verwendet, um einen Gleitkommawert aus dem ListStore auszuwählen.

If you're confused like me about how to edit ComboBox ListStores and CellRenderer, see https://youtu.be/-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)**
: Zero point of range.
If it is inside of min/max range then the bar will grow from that value and not from the left (or right) side of the widget.
Useful to represent values that may be both positive or negative.

**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.

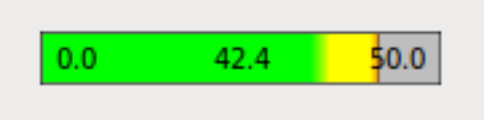



Abbildung 12.41: Horizontaler Balken

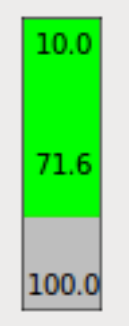



Abbildung 12.42: Vertikaler Balken

#### 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.

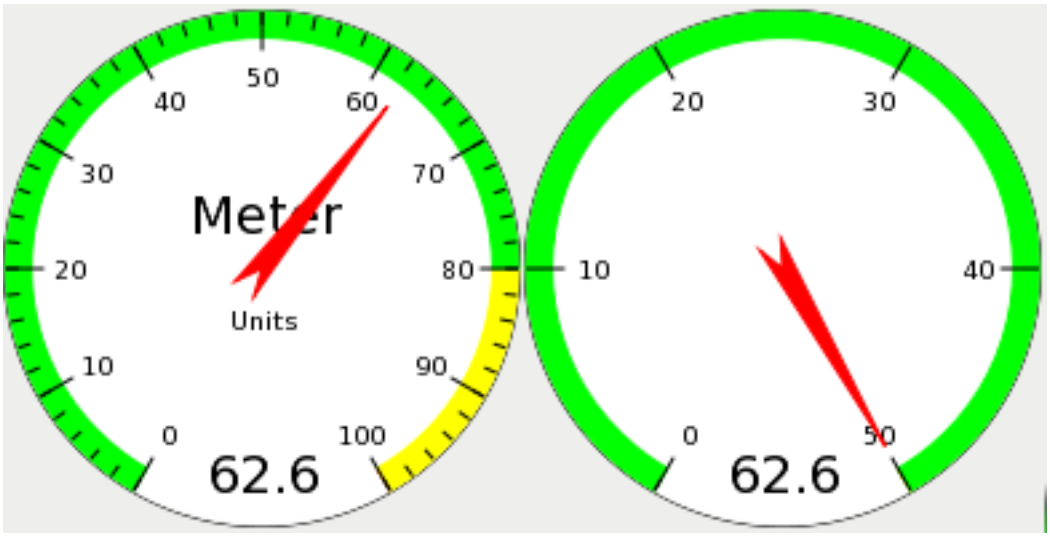



Abbildung 12.43: Beispiel HAL-Messgeräte

#### 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:

**enable_dro**
This displays the dro on the graphics.
Default = true.

**show_velocity**
Dies zeigt die Werkzeuggeschwindigkeit an.
Vorteinstellung = true.

**use_commanded**
Dies wählt die zu verwendende DRO aus: befohlene oder tatsächliche Werte.
Voreinstellung = true.

**metric_units**
Dies legt fest was die DRO nutzt: metrische oder imperiale Einheiten.
Voreinstellung = true.

**show_rapids**
Dies legt fest, dass der Plotter die schnelle Bewegungen zeigt.
Voreinstellung = true.

**show_dtg_**
Hiermit wird die Anzeige des Restweges auf der DRO ausgewählt.
Voreinstellung = true.

**use_relative**
Hiermit wird festgelegt, dass die Anzeige der Werte relativ zu den Koordinaten des Benutzersystems oder der Maschine erfolgen soll.
Voreinstellung = true.

**show_live_plot**
Hiermit wird dem Plotter mitgeteilt, ob er zeichnen soll oder nicht.
Voreinstellung = true.

**show_limits**
Hiermit wird der Plotter angewiesen, die Grenzen der Maschine anzuzeigen.
Voreinstellung = true.

**show_lathe_radius**
Hiermit wird die DRO Anzeige der X-Achse in Radius oder Durchmesser gewählt, wenn der Drehmaschinenmodus aktiviert ist (wählbar in der INI-Datei mit LATHE = 1).
Voreinstellung = true.

**show_extents_option**
This tells the plotter to show the machine's extents.
Default = true.

**show_tool**
Hiermit wird der Plotter angewiesen, das Werkzeug zu zeichnen.
Voreinstellung = true.

**show_program**
Shows the G-code program.
Default = True

**use_joints_mode**
Wird in nicht trivialkins Maschinen (z.B. Robotern) verwendet.
Standard = false.

**grid_size**
Legt die Größe des Gitters fest (nur in den Ansichten X, Y und Z sichtbar). +
Standardwert ist 0

**use_default_controls**
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 GObject, um die oben aufgeführten Eigenschaften einzustellen:

```
[widget name].set_property('view','P')
[widget name].set_property('metric_units',False)
[widget name].set_property('use_default_controls',False)
[widget name].set_property('enable_dro' False)
[widget name].set_property('show_program', False)
[widget name].set_property('show_limits', False)
[widget name].set_property('show_extents_option', False)
[widget name].set_property('show_live_plot', False)
[widget name].set_property('show_tool', False)
[widget name].set_property('show_lathe_radius',True)
[widget name].set_property('show_dtg',True)
[widget name].set_property('show_velocity',False)
[widget name].set_property('mouse_btn_mode', 4)
```

Es gibt Python-Methoden:

```
[widget name].show_offsets = True
[widget name].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.
- If you get the zoom distance before changing the view then reset the zoom distance, it is much more user friendly.
- wenn Sie ein Element in der Vorschau auswählen, wird das ausgewählte Element als Rotationsmittelpunkt verwendet

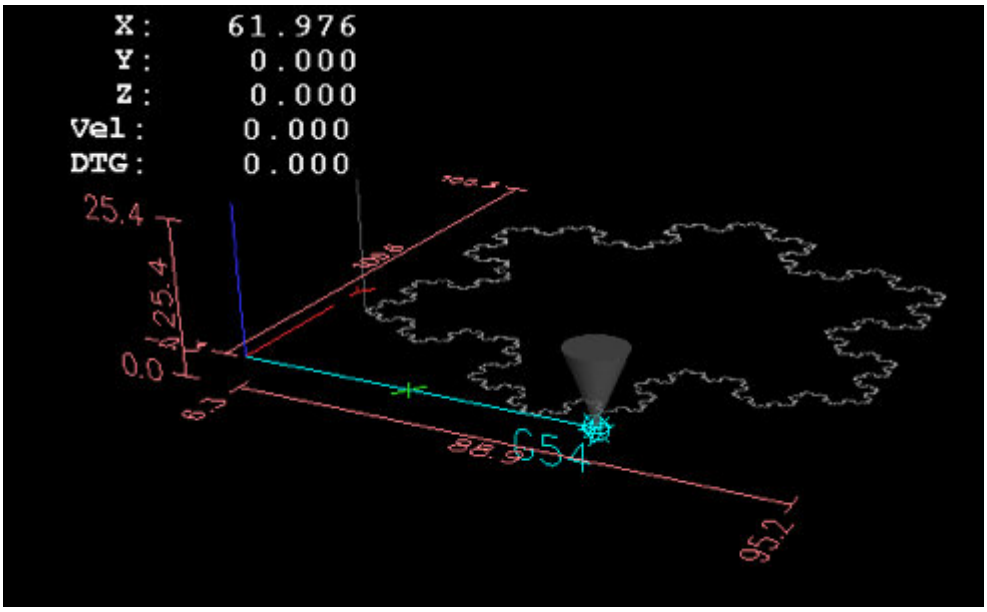



Abbildung 12.44: Gremlin Beispiel

### 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:

**display_units_mm**
Display in metric units.

**joint_number**
Used to select which axis (technically which joint) is displayed.
On a trivialkins machine (mill, lathe, router) axis vs. joint number are:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

**mm_text_template**
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

**imperial_text_template**
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

**reference_type**

```
0:G5x 1:Werkzeug 2:G92 3:Drehung um Z
```

#### 12.3.6.21 DRO-Widget

Das DRO-Widget wird verwendet, um die aktuelle Achsenposition anzuzeigen.

Es hat die folgenden Eigenschaften:

**display_units_mm**
Used to toggle the display units between metric and imperial. Default is False.

**actual (engl. für tatsächlich)**
Select actual (feedback) position or commanded position. Default is True.

**Durchmesser**
Display diameter for a lathe. Default is False.

**mm_text_template**
You can use Python formatting to display the position with different precision. Default is ”%10.3f”.

**imperial_text_template**
You can use Python formatting to display the position with different precision. Default is ”%9.4f”.

**joint_number**
Used to select which axis (technically which joint) is displayed. Default is 0.
On a trivialkins machine (mill, lathe, router) axis vs. joint number are:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W +
```

**reference_type**

- 0 = `absolute` (machine origin).
- 1 = `relative` (to current user coordinate origin - G5x).
- 2 = `distance-to-go` (relative to current user coordinate origin). Default is 0.

**font_family**
Specify the font family e.g. mono. Defaults to sans. If the font does not exist then the current system font will be used. Default is sans.

**font_size**
Specify the size of the font between 8 and 96. Default is 26.

**font_weight**
Specify the weight of the font. Select from lighter, normal, bold, or bolder. Default is bold.

**unhomed_color**
The text color when unhomed specified as a Gdk.RGBA color. Default is red, Gdk.RGBA(red=1.000000, green=0.000000, blue=0.000000, alpha=1.000000)

**homed_color**
The text color when homed specified as a Gdk.RGBA color. Default is green, Gdk.RGBA(red=0.000000, green=0.501961, blue=0.000000, alpha=1.000000)

**Hinweise**

- If you want the display to be right justified, set the Horizontal Alignment to `End`.
- The background of the widget is actually see through, so if you place it over an image, the DRO numbers will show on top of it with no background. There is a special technique to do this. See the animated function diagrams below.
- The DRO widget is a modified gtk label widget. As such, much of what can be done to a gtk label can be done to the DRO widget.
- The font properties may also be set from a css stylesheet which has the highest priority and will override values set by GObject properties.

Es gibt mehrere Möglichkeiten, das Widget direkt mit Python zu steuern.

**Using GObject to set the above listed properties**

```
[widget name].set_property(”display_units_mm”, True)
[widget name].set_property(”actual”, True)
[widget name].set_property(”diameter”, True)
[widget name].set_property(”mm_text_template”, ”%10.3f”)
[widget name].set_property(”imperial_text_template”, ”%9.4f”)
[widget name].set_property(”joint_number”, 3)
[widget name].set_property(”reference_type”, 3)
[widget name].set_property(”font_family”, ”mono”)
[widget name].set_property(”font_size”, 30)
[widget name].set_property(”font_weight”, ”bold”)
```

```
# it is easier to read colors by calling a function:
def str_to_rgba(color):
  c = Gdk.RGBA()
  c.parse(color)
  return c

[widget name].set_property(”unhomed_color”, str_to_rgba(”magenta”))
[widget name].set_property(”homed_color”, str_to_rgba(”cyan”))
```

**Using a CSS stylesheet to set font properties**
Colors may be specified in one of several formats, these would all specify the same color, red, *#ff0000, *rgb(255,0,0), or rgba(255,0,0,255).

Colors may be referenced either collectively:

```
.dro_unhomed {color: magenta}
.dro_homed {color: cyan}
```

or individually by widget name:

```
#[widget name].dro_unhomed {color: magenta}
#[widget name].dro_homed {color: cyan}
```

The other style properties need to be referenced by widget name:

```
#[widget name], #[widget name], #[widget name] {
    font-family: mono;
    font-size: 60px;
    font-weight: lighter;
}
```

**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**
: Used to select which axis (technically which joint) is displayed.
On a trivialkins machine (mill, lathe, router) axis/joint numbers are:

`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**
The time the DRO waits between two polls.
This setting should only be changed if you use more than 5 DRO at the same time, i.e. on a 6 axis config, to avoid that the DRO slows down the main application too much.
The value must be an integer in the range of 100 to 1000. FIXME unit=ms ?
Default is 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` = string
    The actual coordinate system. Will be one of G54 G55 G56 G57 G58 G59 G59.1 G59.2 G59.3 or Rel if none has been selected at all, what will only happen in Glade with no LinuxCNC running.

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:

- `[widget name].show_buttonbox(state)`
  If False the bottom button box will be hidden.
  This is helpful in custom screens, with special buttons layouts to not alter the layout of the GUI. Good example for that is GMOCCAPY.
  `state` = boolean (True or False).
  Default is 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.

If the button box has been hidden, you can reach the functions of this button through its clicked signals like so:

```
[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.
- `sensitive`
  This signal is emitted when the buttons change their state from sensitive to not sensitive or vice versa.
  This signal is useful to maintain surrounding GUI synchronized with the button of the widget. See GMOCCAPY as example.
  It will return the **buttonname** and the new **state**:

  - 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.

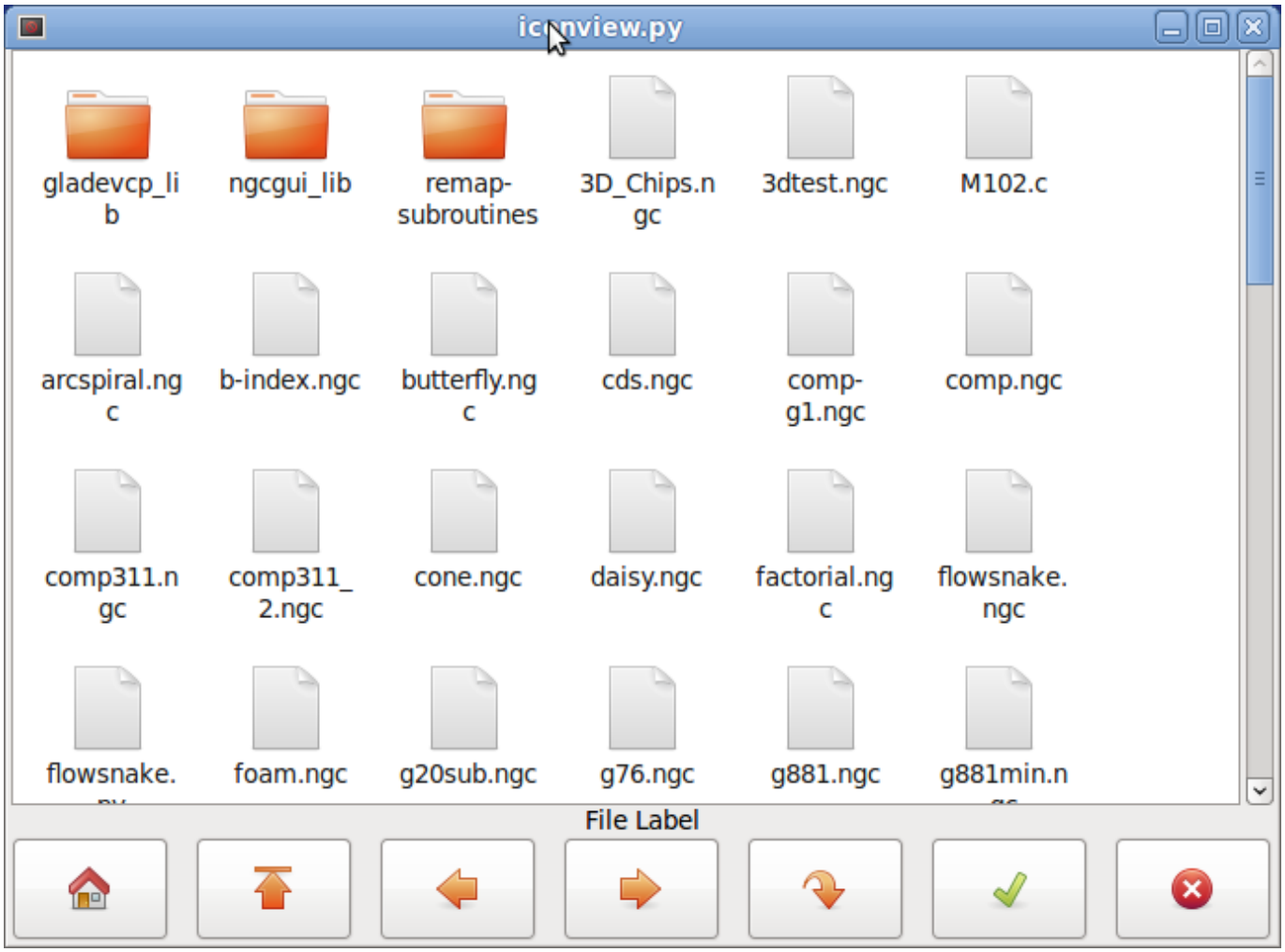



Abbildung 12.46: Iconview Beispiel

#### 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:

**is_editable**
: Damit kann die Eingabeanzeige über eine Tastatur eingegeben werden.

**font**
: 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:

**font**
: Display font to use

**hide_columns**
: Dadurch werden die angegebenen Spalten ausgeblendet. +
Die Spalten werden (in dieser Reihenfolge) wie folgt bezeichnet: `s,t,p,x,y,z,a,b,c,u,v,w,d,i,j,q`. +
Sie können eine beliebige Anzahl von Spalten ausblenden, einschließlich der Auswahl und der Kommentare.

**lathe_display_type**
: Show lathe format

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).

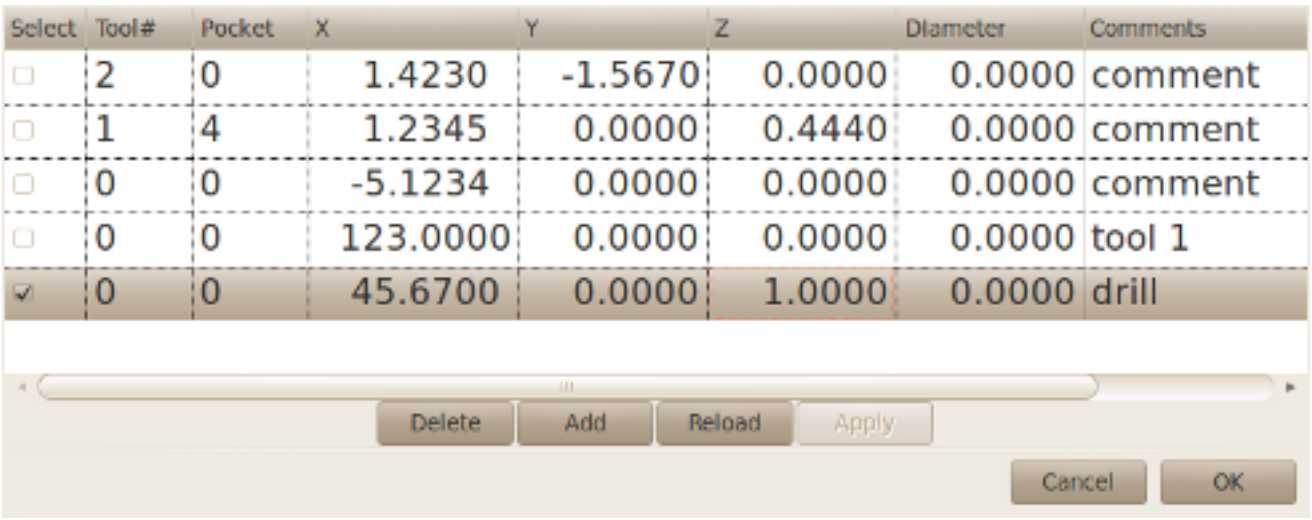



Abbildung 12.47: Werkzeug-Editor (engl. tooleditor) Beispiel

#### 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:

**display_units_mm**
Display in metrice units

**hide_columns**
Eine Liste der auszublendenden Spalten ohne Leerzeichen. Die Spalten werden (in dieser Reihenfolge) wie folgt bezeichnet: `xyzabcuvwt`.
Sie können jede der Spalten ausblenden.

**hide_rows**
Eine Liste der auszublendenden Zeilen ohne Leerzeichen.
Die Zeilen werden (der Reihe nach) wie folgt bezeichnet: `0123456789abc`.
Sie können jede der Zeilen ausblenden.

**font**
Legt Art und Größe der Schriftart fest.

**highlight_color**
Beim Bearbeiten ist dies die Hervorhebungsfarbe.

**foreground_color**
Wenn `OffsetPage` ein aktives Benutzerkoordinatensystem erkennt, wird diese Farbe für den Text verwendet.

**mm_text_template**
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

**imperial_text_template**
Sie können die Python-Formatierung verwenden, um die Position mit unterschiedlicher Genauigkeit anzuzeigen.

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.

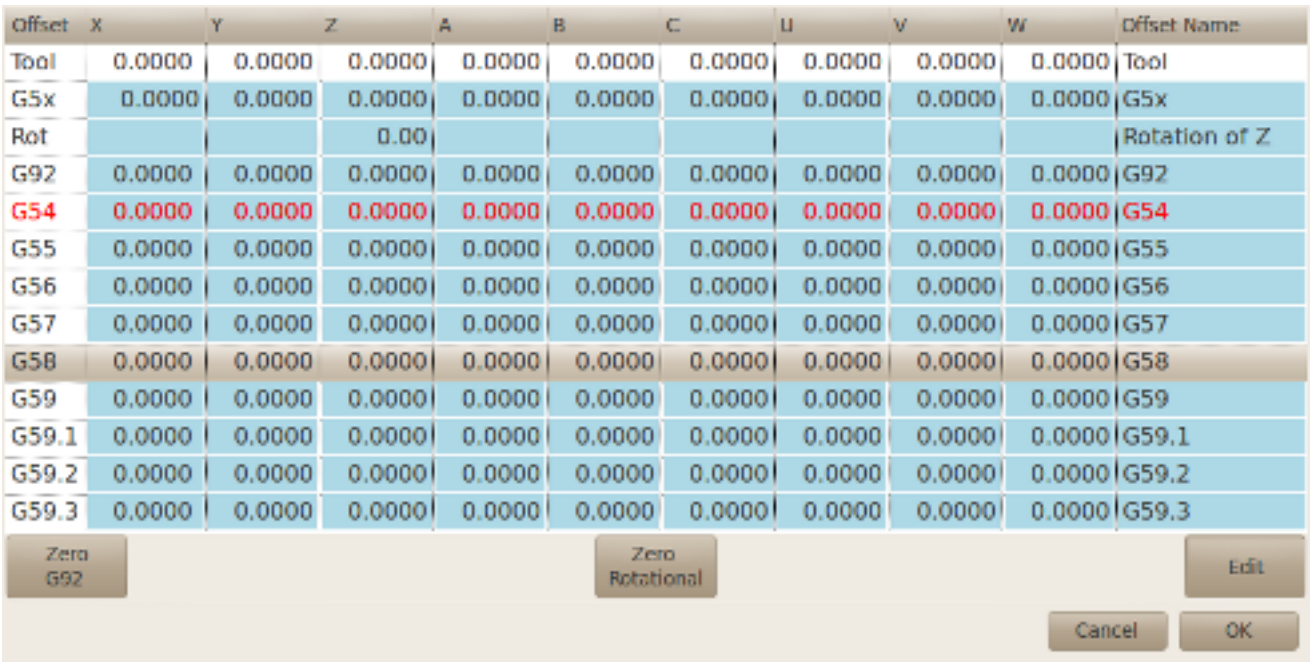



Abbildung 12.48: Beispiel für eine Offset-Seite

#### 12.3.6.27 HAL_sourceview-Widget

This is for displaying and simple editing of G-code. It looks for `.ngc` highlighting specs in `~/share/gtksource`
The current running line will be highlighted.

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

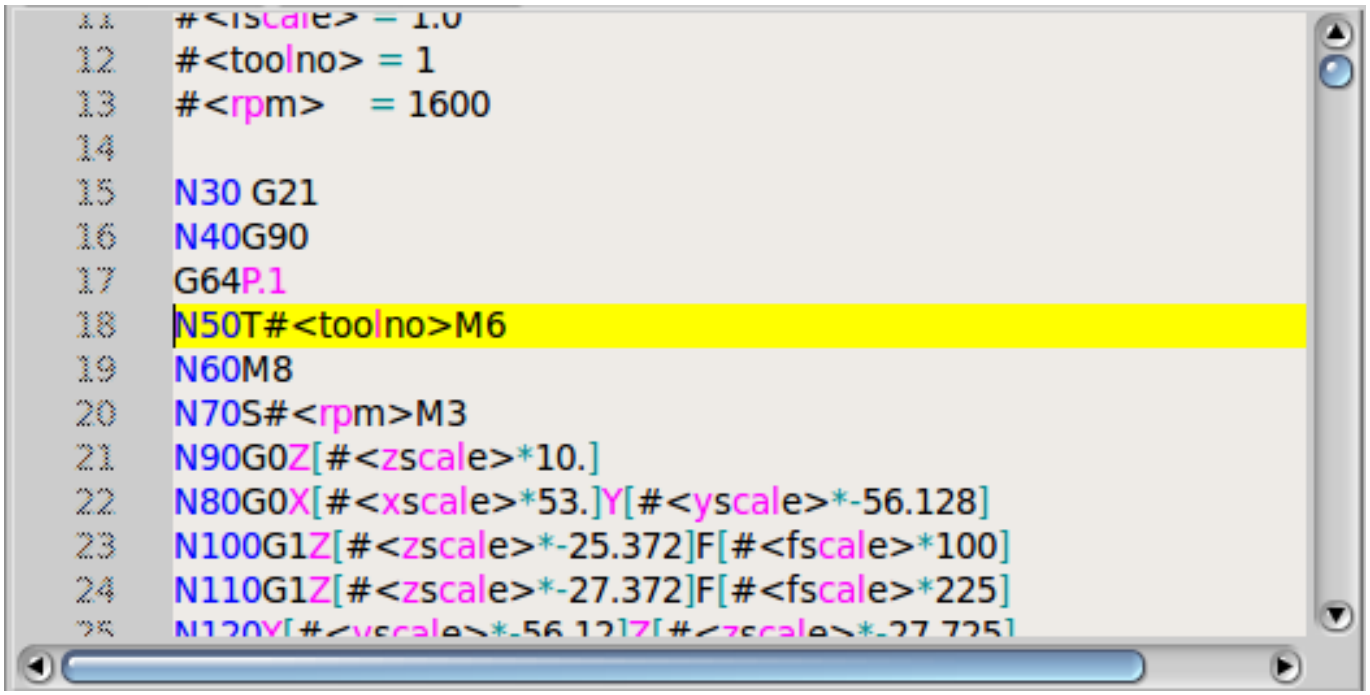



Abbildung 12.49: Quellen-Ansicht (engl. sourceview) Beispiel

#### 12.3.6.28 MDI-Geschichte

This is for displaying and entering MDI codes.
It will be automatically grayed out when MDI is not available, e.g., during E-stop and program running.

**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 enables the mouse double click feature and a double click on an entry will submit that command.
It is not recommended to use this feature on real machines, as a double click on a wrong entry may cause dangerous situations.

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:

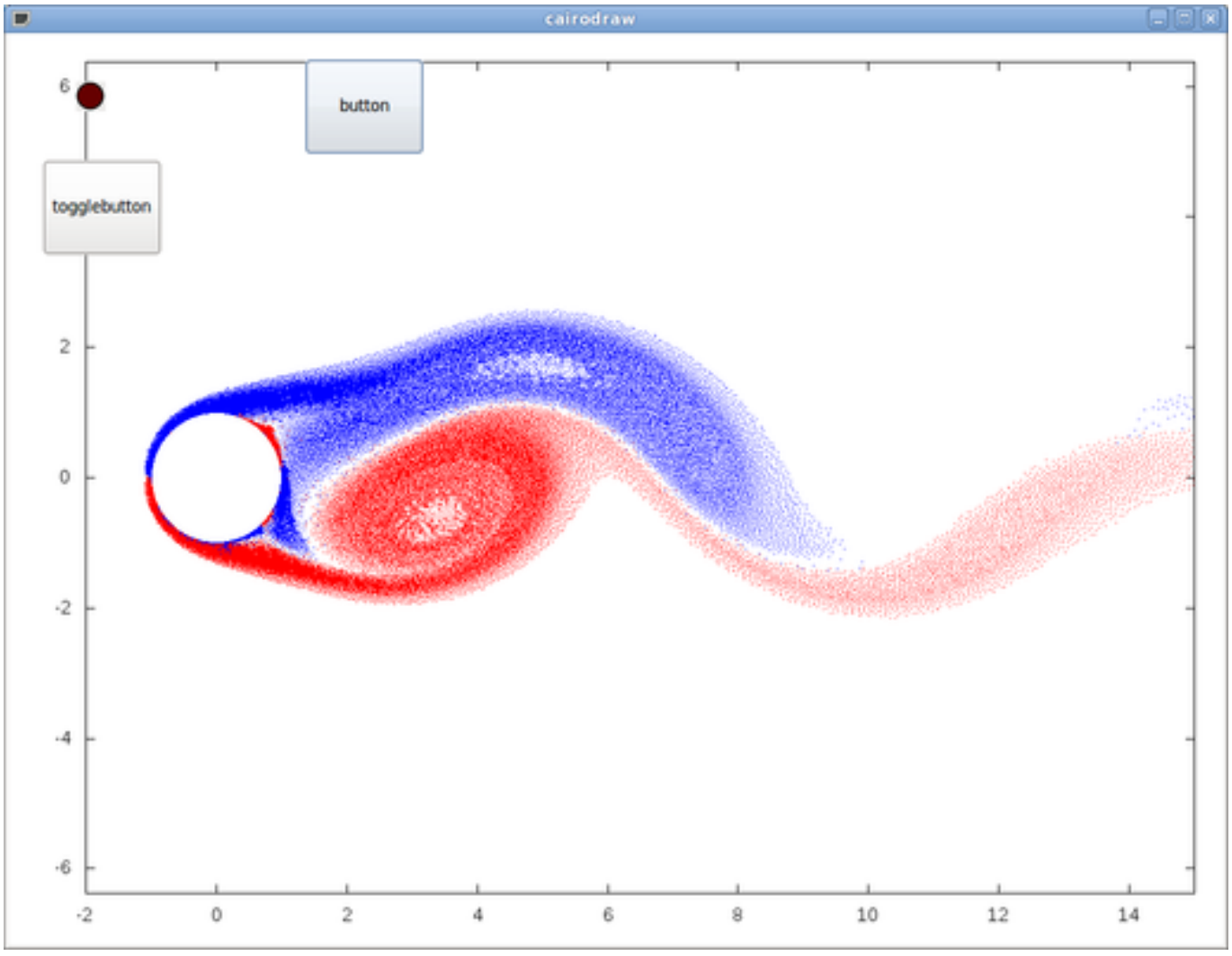



Abbildung 12.50: HAL-Widgets in einem Bitmap Beispiel

### 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:

- It is an object available in Glade
- It has no visual appearance by itself
- Its purpose: Associate a visible, sensitive UI component like menu, toolbutton, button with a command. See these widget's *General→Related→Action* property.
- The "canned action" will be executed when the associated UI component is triggered (button press, menu click..).
- It provides an easy way to execute commands without resorting to Python programming.

Das Erscheinungsbild der VCP-Aktionen in Glade sieht in etwa wie folgt aus:

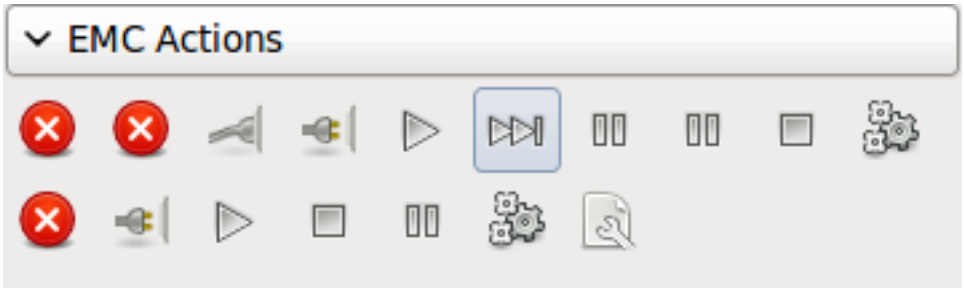



Abbildung 12.51: Action-Widgets

Hover über Werkzeugspitzen (engl. tooltips) liefert eine Beschreibung.

#### 12.3.7.1 VCP Action-Widgets

VCP Action widgets are one-shot type widgets. They implement a single action and are for use in simple buttons, menu entries or radio/check groups.

#### 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` für den Zugriff auf LinuxCNC's rohen Status über das LinuxCNC Python Modul.

- `CMD` für den Zugriff auf LinuxCNC-Befehle über das LinuxCNC-Python-Modul.
- `EXT` für den Zugriff auf die Handler-Dateifunktionen, falls verfügbar.
- `linuxcnc` für den Zugriff auf das LinuxCNC-Python-Modul.
- `self` für den Zugriff auf die Widget-Instanz.
- *dir* für den Zugriff auf die Handler-Attributliste.

Es gibt Optionen für

- wählen Sie aus, wann das Widget aktiv sein soll,
- den Modus einstellen, bevor der Befehl ausgeführt wird.

Beispiel für einen Befehl, um einfach eine Nachricht auf dem Terminal zu auszugeben:

```
print('Aktion aktiviert')
```

Beispiel für einen Befehl, um die Maschine in den Aus-Zustand zu versetzen:

```
CMD.state(linuxcnc.STATE_OFF)
```

Beispiel für einen Befehl für den Aufruf einer Handler-Funktion, die Daten übergibt:

```
EXT.on_button_press(self, 100)
```

You can use a semicolon to separate multiple commands;

```
print('Set Machine Off');CMD.state(linuxcnc.STATE_OFF)
```

More information on INFO and ACTION can be found here: GladeVCP Libraries modules.

More information on GStat can be found here: GStat.

#### 12.3.7.3 VCP ToggleAction-Widgets

These are **bi-modal** widgets. They implement two actions or use a second (usually `pressed`) state to indicate that currently an action is running. Toggle actions are aimed for use in `ToggleButtons`, `ToggleToolButtons` or toggling menu items. A simplex example is the `ESTOP` toggle button.

Derzeit sind die folgenden Widgets verfügbar:

- The `ESTOP` toggle sends `ESTOP` or `ESTOP_RESET` commands to LinuxCNC depending on its state.
- 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. Open it in Glade and study how it is done.
2. Starten Sie AXIS, und starten Sie es dann von einem Terminalfenster aus mit `gladevcp whoareyou.ui`.
3. See the `hal_action_mdi1` action and its `MDI command` property - this just executes `(MSG, "Hi, I'm an VCP_Action_MDI")` so there should be a message popup in AXIS like so:

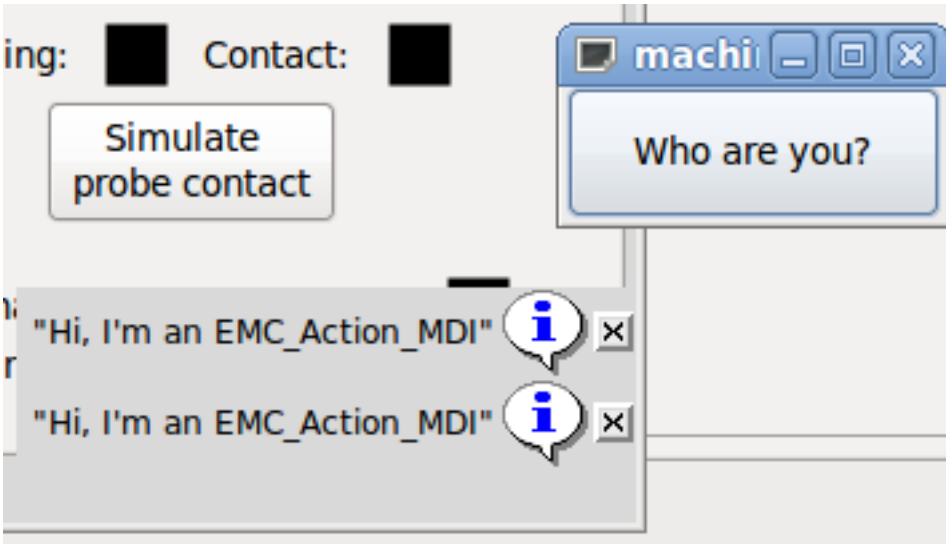



Abbildung 12.52: Action_MDI Einfaches Beispiel

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:

- assume you have a *HAL SpinBox* named `speed`, and you want to pass its current value as a parameter in an MDI command.
- 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:

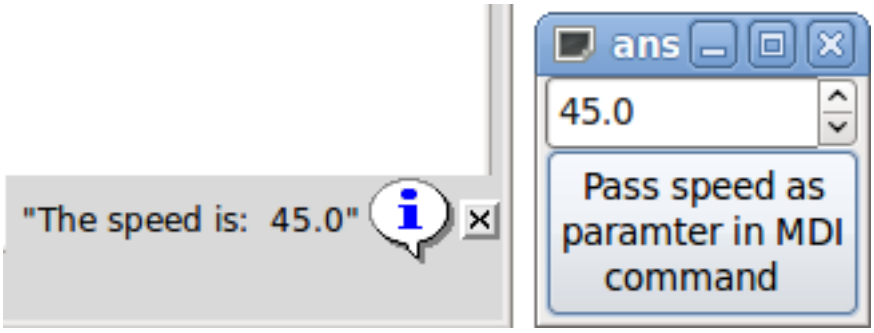



Abbildung 12.53: Action_MDI Parameterübergabe Beispiel

#### 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:

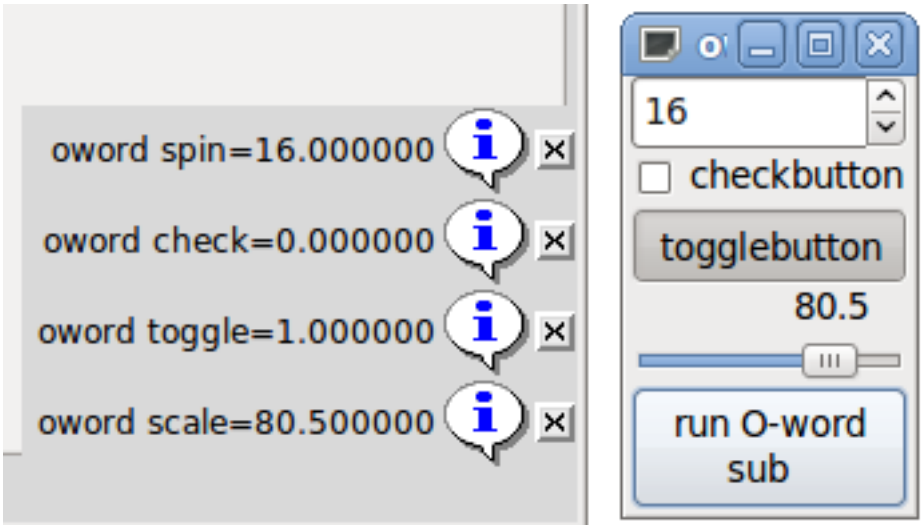



Abbildung 12.54: Action_MDI Erweitertes Beispiel

#### 12.3.7.8 Vorbereitung einer MDI-Aktion und anschließendes Aufräumen

The LinuxCNC G-code interpreter has a single global set of variables, like feed, spindle speed, relative/absolute mode and others. If you use G-code commands or O-word subs, some of these variables might get changed by the command or subroutine - for example, a probing subroutine will very likely

set the feed value quite low. With no further precautions, your previous feed setting will be overwritten by the probing subroutine's value.

To deal with this surprising and undesirable side effect of a given O-word subroutine or G-code statement executed with an LinuxCNC ToggleAction_MDI, you might associate pre-MDI and post-MDI handlers with a given LinuxCNC ToggleAction_MDI. These handlers are optional and provide a way to save any state before executing the MDI Action, and to restore it to previous values afterwards. The signal names are `mdi-command-start` and `mdi-command-stop`; the handler names can be set in Glade like any other handler.

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

Many actions depend on LinuxCNC status - is it in manual, MDI or auto mode? Is a program running, paused or idle? You cannot start an MDI command while a G-code program is running, so this needs to be taken care of. Many LinuxCNC actions take care of this themselves, and related buttons and menu entries are deactivated when the operation is currently impossible.

When using Python event handlers - which are at a lower level than Actions - one needs to take care of dealing with status dependencies oneself. For this purpose, there's the LinuxCNC Stat widget: to associate LinuxCNC status changes with event handlers.

LinuxCNC Stat has no visible component - you just add it to your UI with Glade. Once added, you can associate handlers with its following signals:

- 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

Most widget sets, and their associated user interface editors, support the concept of callbacks, i.e. functions in user-written code which are executed when *something happens* in the UI - events like mouse clicks, characters typed, mouse movement, timer events, window hiding and exposure and so forth.

HAL output widgets typically map input-type events like a button press to a value change of the associated HAL pin by means of such a - predefined - callback. Within PyVCP, this is really the only type of event handling supported - doing something more complex, like executing MDI commands to call a G-code subroutine, is not supported.

Within GladeVCP, HAL pin changes are just one type of the general class of events (called signals) in GTK+. Most widgets may originate such signals, and the Glade editor supports associating such a signal with a Python method or function name.

If you decide to use user-defined actions, your job is to write a Python module whose class methods - or in the simple case, just functions - can be referred to in Glade as event handlers. GladeVCP provides a way to import your module(s) at startup and will automatically link your event handlers with the widget signals as set in the Glade UI description.

### 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()
```

Using the library functions:

```
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 can not only manipulate or display HAL pins, you can also write regular event handlers in Python. This could be used, among others, to execute MDI commands. Here's how you do it:

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)
```

In Glade, define a button or HAL button, select the *Signals* tab, and in the GtkButton properties select the *pressed* line. Enter *on_button_press* there, and save the Glade file.

Then add the option *-u handlers.py* to the GladeVCP command line. If your event handlers are spread over several files, just add multiple *-u <pyfilename>* options.

Now, pressing the button should change its label since it is set in the callback function.

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).

Beyond this built-in functionality, one may associate a change callback with any HAL pin, including those of predefined HAL widgets. This fits nicely with the event-driven structure of a typical widget application: Every activity, be it mouse click, key, timer expired, or the change of a HAL pin's value, generates a callback and is handled by the same orthogonal mechanism.

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.

For simple tasks it is sufficient to define functions named after the Glade signal handlers. These will be called when the corresponding event happens in the widget tree. Here’s a trivial example - it assumes that the *pressed* signal of a Gtk Button or HAL Button is linked to a callback called *on_button_press*:

```
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.

The idea here is: Handlers are linked to class methods. The underlying class(es) are instantiated and inspected during GladeVCP startup and linked to the widget tree as signal handlers. So the task now is to write:

- One or more several class definition(s) with one or several methods, in one module or split over several modules,
- a function *get_handlers* in each module which will return a list of class instances to GladeVCP - their method names will be linked to signal handlers.

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` - eine Liste von Zeichenfolgen, die von der GladeVCP-Befehlszeile mit der Option *-U <useropts>* gesammelt werden.

GladeVCP then inspects the list of class instances and retrieves their method names. Qualifying method names are connected to the widget tree as signal handlers. Only method names which do not begin with an _ (underscore) are considered.

Note that regardless whether you're using the libglade or the new GtkBuilder format for your Glade UI, widgets can always be referred to as `builder.get_object(<widgetname>)`. Also, the complete list of widgets is available as `builder.get_objects()` regardless of 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).
- The halcomp HAL component is set up and all HAL widgets' pins have already been added to it.
- 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.

So when your handler class is initialized, all widgets are existent but not yet realized (displayed on screen). And the HAL component isn't ready as well, so its unsafe to access pins values in your `__init__()` method.

If you want to have a callback to execute at program start after it is safe to access HAL pins, then a connect a handler to the realize signal of the top level window1 (which might be its only real purpose). At this point GladeVCP is done with all setup tasks, the HAL file has been run, and GladeVCP is about to enter the Gtk main loop.

#### 12.3.8.7 Mehrere Callbacks mit demselben Namen

Within a class, method names must be unique. However, it is OK to have multiple class instances passed to GladeVCP by get_handlers() with identically named methods. When the corresponding signal occurs, these methods will be called in definition order - module by module, and within a module, in the order class instances are returned by `get_handlers()`.

#### 12.3.8.8 Die GladeVCP `-U <useropts>` Flag

Instead of extending GladeVCP for any conceivable option which could potentially be useful for a handler class, you may use the -U <useroption> flag (repeatedly if you wish). This flag collects a list of <useroption> strings. This list is passed to the get_handlers() function (useropts argument). Your code is free to interpret these strings as you see fit. An possible usage would be to pass them to the Python exec function in your `get_handlers()` as follows:

```
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

An annoying aspect of GladeVCP in its earlier form and PyVCP is the fact that you may change values and HAL pins through text entry, sliders, spin boxes, toggle buttons, etc., but their settings are not

saved and restored at the next run of LinuxCNC - they start at the default value as set in the panel or widget definition.

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** Imagine renaming, adding or deleting widgets in Glade: An .INI file lying around from a previous program version, or an entirely different user interface, would be not be able to restore the state properly since variables and types might have changed.

GladeVCP detects this situation by a signature which depends on all object names and types which are saved and to be restored. In the case of signature mismatch, a new INI file with default settings is generated.

#### 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”
```

Note that types are saved and preserved on restore. This example assumes that the INI file didn’t exist or had the default values from self.defaults.

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.
- Add the handler name, e.g. *on_destroy*, to the second column.
- 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)

You can do that, but note that the values in self.defaults override your edits if there is a syntax or type error in your edit. The error is detected, a console message will hint about that happened, and the bad inifile will be renamed to have the .BAD suffix. Subsequent bad INI files overwrite earlier .BAD files.

#### 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

Since GladeVCP uses Gtk widgets which rely on the PyGObject base class, the full GLib functionally is available. Here is an example for a timer callback:

```
def _on_timer_tick(self,userdata=None):
    ...
    return True # to restart the timer; return False for on-shot
...
# demonstrate a slow background timer - granularity is one second
# for a faster timer (granularity 100 ms), use this:
# 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

With Glade, widget properties are typically set fixed while editing. You can, however, set widget properties at runtime, for instance from INI file values, which would typically be done in the handler initialization code. Setting properties from HAL pin values is possible, too.

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 mit Änderungs-Callback.
        # Wenn sich der Wert des Pins ändert, wird der Callback ausgeführt.
        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 Value-changed callback with hal_glib

GladeVCP nutzt die hal_glib-Bibliothek, die dazu verwendet werden kann, ein "watcher" Signal an einen HAL-Eingangspin anzuschließen.
Dieses Signal kann verwendet werden, um eine Funktion zu registrieren, die aufgerufen wird, wenn sich der Zustand des HAL-Pins ändert.

One must import the `hal_glib` and the `hal` modules:

```
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: {}".format(pin.get()))
        print("pin name= {}".format(pin.get_name()))
        print("pin type= {}".format(pin.get_type()))

        # dies kann außerhalb der Funktion aufgerufen werden
        self.example_trigger.get()
```

#### 12.3.8.18 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?*

   This is a consequence of your Glade UI file having the window1 Visible property set to True, together with re-parenting the GladeVCP window into AXIS or touchy. The GladeVCP widget tree is created, including a top level window, and then *reparented into AXIS*, leaving that toplevel window laying around orphaned. To avoid having this useless empty window hanging around, it is unmapped (made invisible), which is the cause of the unmap signal you get. Suggested fix: set window1.visible to False, and ignore an initial unmap event.

2. *Mein GladeVCP-Programm startet, aber es erscheint kein Fenster dort, wo ich es erwarte?*

   The window AXIS allocates for GladeVCP will obtain the *natural size* of all its child widgets combined. It is the child widget's job to request a size (width and/or height). However, not all widgets do request a width greater than 0, for instance the Graph widget in its current form. If there's such a widget in your Glade file and it is the one which defines the layout you might want to set its width explicitly. Note that setting the window1 width and height properties in Glade does not make sense because this window will be orphaned during re-parenting and hence its geometry will have no impact on layout (see above). The general rule is: if you manually run a UI file with *gladevcp <uifile>* and its window has reasonable geometry, it should come up in AXIS properly as well.

3. *Ich möchte eine blinkende LED, aber sie blinkt nicht*

   I ticked the checkbutton to let it blink with 100 msec interval. It wont blink, and I get a startup warning: Warning: value "0" of type ‚gint' is invalid or out of range for property ‚led-blink-rate' of type ‚gint'? This seems to be a Glade bug. Just type over the blink rate field, and save again - this works for me.

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*

   Very likely this handler is linked to window1, which due to reparenting isn't usable for this purpose. Please link the on_destroy handler to the destroy signal of an interior window. For instance, I have a notebook inside window1, and linked on_destroy to the notebooks destroy signal, and that works fine. It doesn't work for window1.

5. *Ich möchte die Hintergrundfarbe oder den Text eines HAL_Label-Widgets abhängig von seinem HAL-Pin-Wert einstellen*

   See the example in configs/apps/gladevcp/colored-label. Setting the background color of a GtkLabel widget (and HAL_Label is derived from GtkLabel) is a bit tricky. The GtkLabel widget has no window object of its own for performance reasons, and only window objects can have a

background color. The solution is to enclose the Label in an EventBox container, which has a window but is otherwise invisible - see the coloredlabel.ui file.

**Ich habe ein "hal_spinbutton"-Widget in Glade definiert und eine Standard-Eigenschaft `value` in**

This is due to a bug in the old Gtk version distributed with Ubuntu 8.04 and 10.04, and is likely to be the case for all widgets using adjustment. The workaround mentioned for instance in http://osdir.com/ml/gtk-app-devel-list/2010-04/msg00129.html does not reliably set the HAL pin value, it is better to set it explicitly in an `on_realize` signal handler during widget creation. See the example in `configs/apps/gladevcp/by-widget/spinbutton.{ui,py}`.

### 12.3.10 Fehlersuche

- Make sure you have the development version of LinuxCNC installed. You don't need the axisrc file any more, this was mentioned in the old GladeVCP wiki page.
- 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*.
- If you're using *HAL_table* or *HAL_HBox* widgets, be aware they have an HAL pin associated with it which is off by default. This pin controls whether these container's children are active or not.

### 12.3.11 Implementierungshinweis: Schlüsselbehandlung in AXIS

We believe key handling works OK, but since it is new code, we're telling about it you so you can watch out for problems; please let us know of errors or odd behavior. This is the story:

AXIS uses the TkInter widget set. GladeVCP applications use Gtk widgets and run in a separate process context. They are hooked into AXIS with the Xembed protocol. This allows a child application like GladeVCP to properly fit in a parent's window, and - in theory - have integrated event handling.

However, this assumes that both parent and child application properly support the Xembed protocol, which Gtk does, but TkInter does not. A consequence of this is that certain keys would not be forwarded from a GladeVCP panel to AXIS properly under all circumstances. One of these situations was the case when an Entry, or SpinButton widget had focus: In this case, for instance an Escape key would not have been forwarded to AXIS and cause an abort as it should, with potentially disastrous consequences.

Therefore, key events in GladeVCP are explicitly handled, and selectively forwarded to AXIS, to assure that such situations cannot arise. For details, see the `keyboard_forward()` function in `lib/python/gladevcp`

### 12.3.12 Hinzufügen von benutzerdefinierten Widgets

The LinuxCNC Wiki has information on adding custom widgets to 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*.

For testing, a runtime specification of auxiliary applications may be specified using the exported environmental variable: GLADEVCP_EXTRAS. This variable should be a path list of one or more configuration directories separated by a (:). Typically, this variable would be set in a shell starting `linuxcnc` or in a user's `~/.profile` startup script. Example:

```
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

Bibliotheken sind vorgefertigte Python-Module, die GladeVCP zusätzliche Funktionen verleihen. Auf diese Weise können Sie auswählen, welche Funktionen Sie wünschen - und müssen die gängigen Funktionen nicht selbst erstellen.

### 12.4.1 Info

Info ist eine Bibliothek zum Sammeln und Filtern von Daten aus der INI-Datei.

Die verfügbaren Daten und Voreinstellungen:

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
```

```
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = 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

Diese Bibliothek wird verwendet, um die Bewegungssteuerung von LinuxCNC zu steuern. Es versucht, zufällige Details zu verbergen und praktische Methoden für Entwickler hinzuzufügen.

Um diese Module zu importieren, fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Action
```

Um das Modul zu instanziieren, damit Sie es verwenden können, fügen Sie den folgenden Python-Code in Ihren instanziierten Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

ACTION = Action()
```

Für den Zugriff auf Aktionsbefehle verwenden Sie eine allgemeine Syntax wie die folgende:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()

ACTION.OPEN_PROGRAM(filename)
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
ACTION.ABORT()
ACTION.PAUSE()

ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)
```

```
ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()

ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)

ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)
```

Es gibt einige *Hilfsfunktionen*, die hauptsächlich für die Unterstützung dieser Bibliothek verwendet werden

```
get_jog_info (num)
jnum_check(num)
ensure_mode(modes)
open_filter_program(filename, filter)
```

## 12.5 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 is completely *customizable*: you can add different buttons and status LEDs etc. or add Python code for even finer grain customization.

### 12.5.1 Schaukasten

Einige Beispiele für mit QtVCP erstellte Bildschirme und virtuelle Bedienfelder:

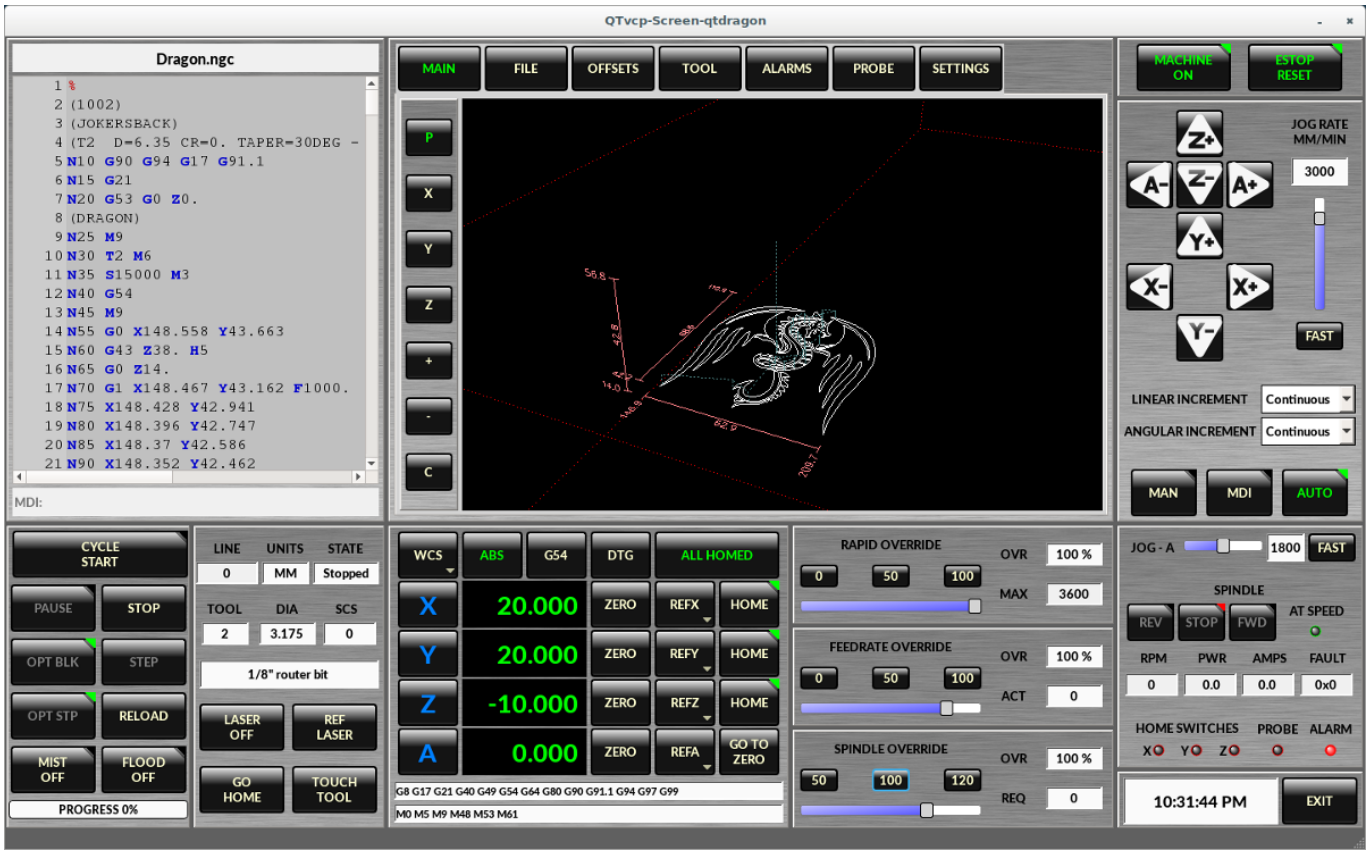



Abbildung 12.55: QtDragon - 3/4-Achsen-Beispiel

Abbildung 12.56: QtDefault - 3-Achsen-Beispiel

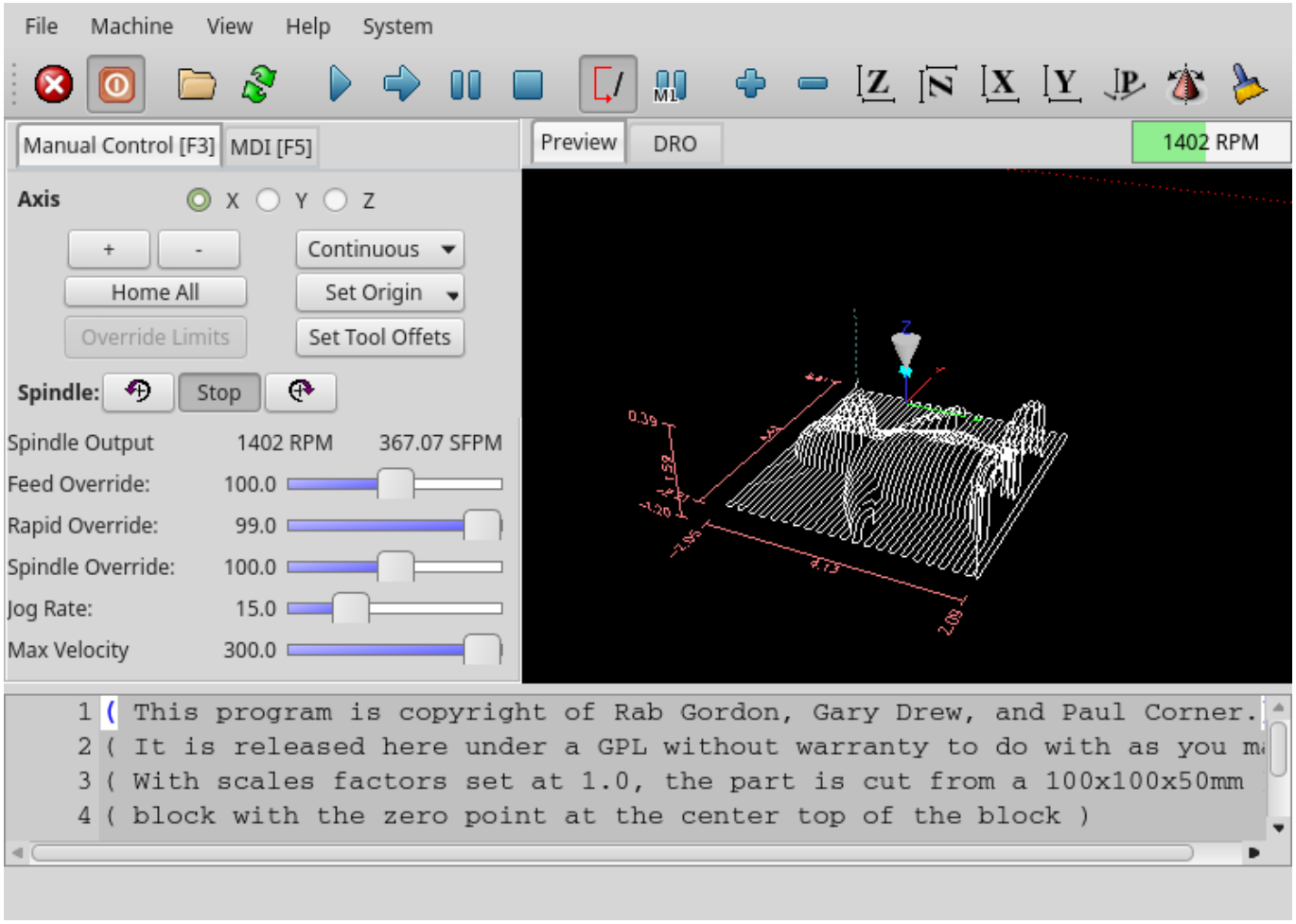



Abbildung 12.57: QtAxis - Beispiel für selbsteinstellende Achsen

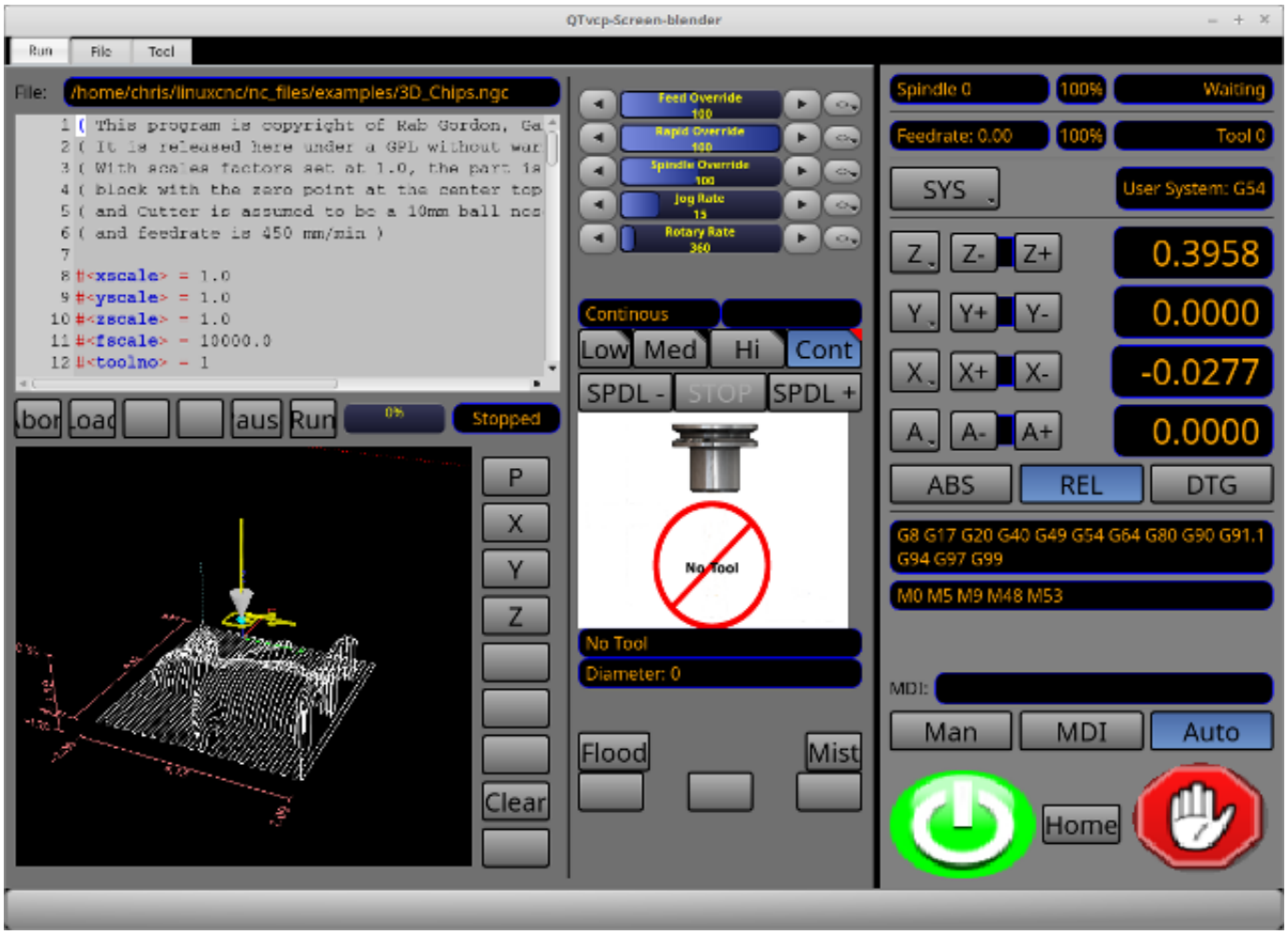



Abbildung 12.58: Blender - 4-Achsen-Beispiel

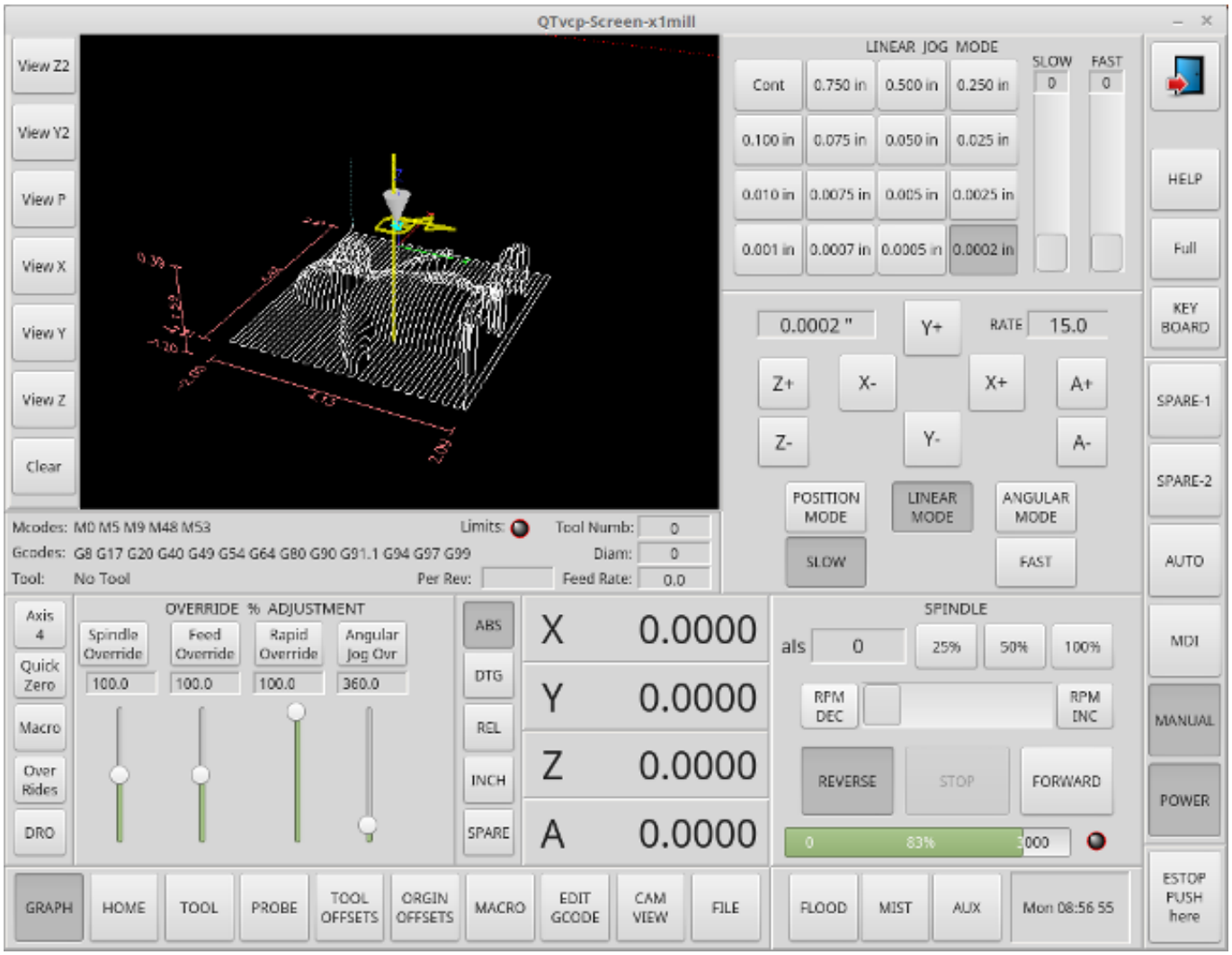



Abbildung 12.59: X1mill - 4-Achsen-Beispiel

Abbildung 12.60: cam_align – Kameraausrichtung VCP

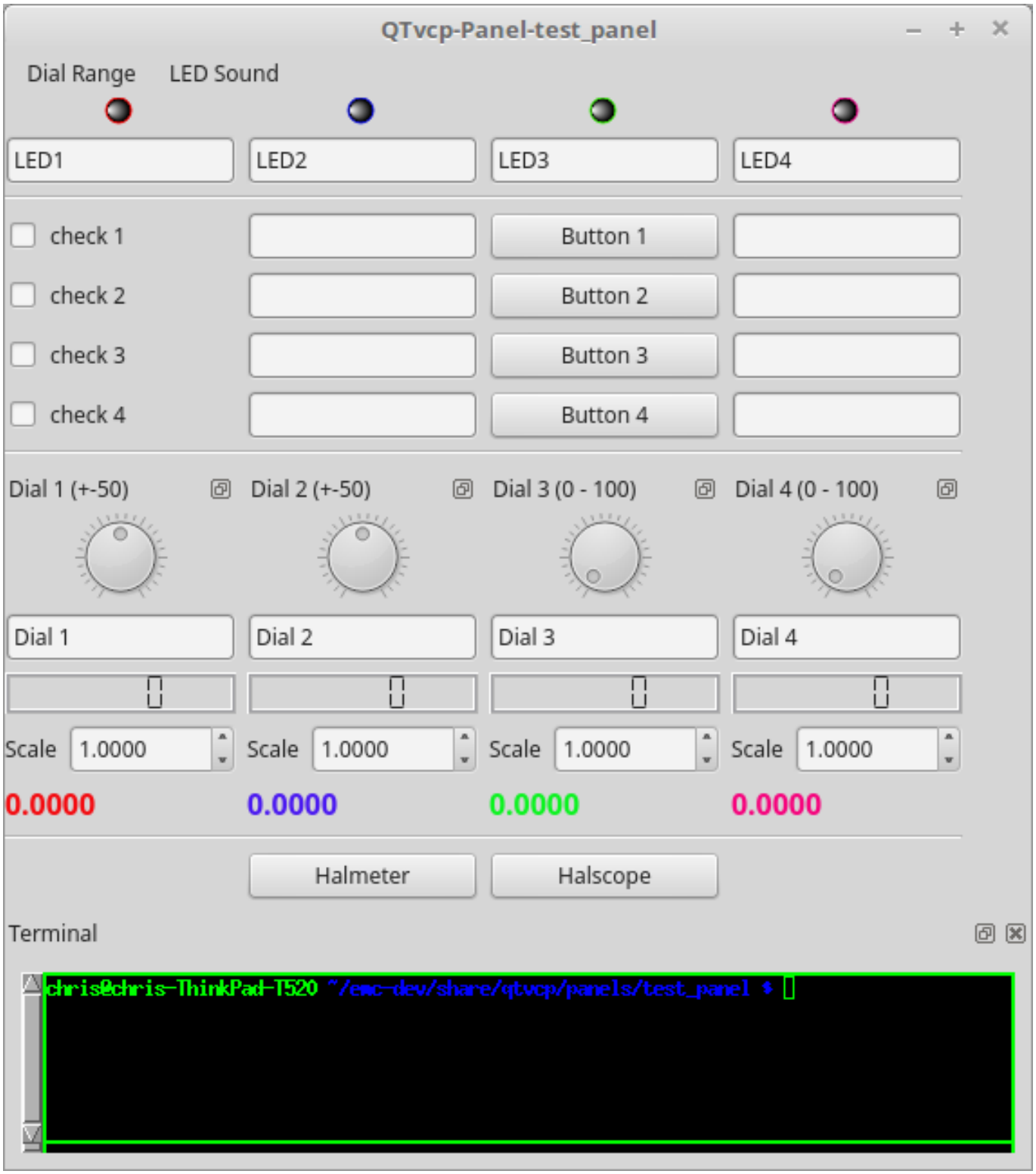



Abbildung 12.61: test_panel - Test Panel VCP

### 12.5.2 Übersicht

*Two files are used, individually or in combination*, to add customizations:

- 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 motion control screen (rather then a HAL based panel), 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.
- `-i` Enable info output.

- `-v` Enable verbose debug output.
- `-q` Enable only error debug output.
- `-a` Set window always on top.
- `-c NAME` HAL component name. Default is to use the UI file name.
- `-g GEOMETRY` Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in pixel units, XOFFSET/YOFFSET is referenced from top left of screen. Use -g WIDTHxHEIGHT for just setting size or -g +XOFFSET+YOFFSET for just position. Example: `-g 200x400+0+100`
- `-H FILE` Execute hal statements from FILE with halcmd after the component is set up and ready.
- `-m` Maximize window.
- `-f` Fullscreen the window.
- `-t THEME` Default is system theme
- `-x XID` Embed into a X11 window that doesn't support embedding.
- `--push_xid` Sendet die X11-Fenster-Identifikationsnummer von QtVCP an die Standardausgabe; zum Einbetten.
- `-u USERMOD` File path of a substitute handler file.
- `-o USEROPTS` Pass a string to QtVCP's handler file under `self.w.USEROPTIONS_` list variable. Can be multiple -o.

**<Bildschirm_name>** `<screen_name>` is the *base name of the .ui and _handler.py files*. If `<screen_name>` is missing, the default screen will be loaded.

QtVCP assumes the UI file and the handler file use the **same base name**. QtVCP will first search the LinuxCNC configuration directory that was launched for the files, then in the system skin folder holding standard screens.

**Zykluszeiten**

```
[DISPLAY]
CYCLE_TIME = 100
GRAPHICS_CYCLE_TIME = 100
HALPIN_CYCLE = 100
```

Adjusts the response rate of the GUI updates in milliseconds. Defaults to 100, useable range 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**.

A handler file allows one to *modify defaults*, or *add logic* to a QtVCP screen without having to modify QtVCP's core code. In this way you can have **custom behaviors**.

If present a handler file will be loaded. **Only one file** is allowed.

#### 12.5.2.5 Bibliotheken Module

QtVCP, as built, does little more than display the screen and react to widgets. For more **prebuilt behaviors** there are available libraries (found in `lib/python/qtvcp/lib` in RIP LinuxCNC install).

**Libraries** are prebuilt *Python modules* that **add features** to QtVCP. In this way you can select what features you want - yet don't have to build common ones yourself.

Such libraries include:

- `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/QSS/Python files in the configuration folder will be loaded instead of the stock UI files.

Local UI/QSS/Python files allow you to use your customized designs rather than the default screens.

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.

Stylesheets can be used to **set Qt properties**. If a widget uses properties then they usually can be modified by stylesheets.

**Example of a widget with accompanying style sheet settings.**

```
State_LED #name_of_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
  }
```

We can have QtVCP load a subclassed version of the standard handler file. in that file we can manipulate the original functions or add new ones.
Subclassing just means our handler file first loads the original handler file and adds our new code on top of it - so a patch of changes.
This is useful for changing/adding behaviour while still retaining standard handler updates from LinuxCNC repositories.

You may still need to use the handler copy dialog to copy the original handler file to decide how to patch it. See *custom handler file*

There should be a folder in the config folder; for screens: named *<CONFIG FOLDER>/qtvcp/screens/<SCRE NAME>/*
add the handle patch file there, named like so <ORIGINAL SCREEN NAME>_handler.py
ie for Qtdragon the file would be called *qtdragon_handler.py*

Here is a sample to add X axis jog pins to a screen like Qtdragon:

```
import sys
import importlib
from qtvcp.core import Path, Qhal, Action
PATH = Path()
QHAL = Qhal()
ACTION = Action()

# get reference to original handler file so we can subclass it
sys.path.insert(0, PATH.SCREENDIR)
module = "{}.{}_handler".format(PATH.BASEPATH,PATH.BASEPATH)
mod = importlib.import_module(module, PATH.SCREENDIR)
sys.path.remove(PATH.SCREENDIR)
HandlerClass = mod.HandlerClass

# return our subclassed handler object to QtVCP
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# sub class HandlerClass which was imported above
class UserHandlerClass(HandlerClass):
    # add a terminal message so we know this got loaded
```

```
    print('\nCustom subclassed handler patch loaded.\n')

    def init_pins(self):
        # call original handler init_pins function
        super().init_pins()

        # add jog pins X axis
        pin = QHAL.newpin("jog.axis.jog-x-plus", QHAL.HAL_BIT, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, 1, fast = False, linear =  ↩
            True))

        pin = QHAL.newpin("jog.axis.jog-x-minus", QHAL.HAL_BIT, QHAL.HAL_IN)
        pin.value_changed.connect(lambda s: self.kb_jog(s, 0, -1, fast = False, linear =  ↩
            True))
```

Another Python file can be used to **add commands** to the screen, after the handler file is parsed. This can be useful for minor changes while still honouring standard handler updates from LinuxCNC repositories.

---

**Anmerkung**
Handler patching is a better way to add changes - instance patching is black magic voodoo - this is here for legacy reasons.

---

In the *INI file* under the `[DISPLAY]` heading add **`USER_COMMAND_FILE = _PATH_`**
*PATH* can be any valid path. It can use ~ for home directory or `WORKINGFOLDER` or `CONFIGFOLDER` to represent QtVCP's idea of those directories:

```
[DISPLAY]
USER_COMMAND_FILE = CONFIGFOLDER/<screen_name_added_commands>
```

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, e.g., **`CONFIGURATION DIRECTORY/.<screen_`**

Diese Datei wird als Python-Code im **handler-Dateikontext** gelesen und ausgeführt.

**Only local functions and local attributes** can be referenced.
Global libraries defined in the screen's handler file can be referenced by importing the handler file. These are usually seen as all capital words with no preceding self.
*self* references the window class functions
*self.w* typically references the widgets

Was verwendet werden kann, mag je nach Bildschirm und Entwicklungszyklus variieren.

**A simple example** Reference the main window to change the title (won't show if using INI entries for title change).

```
self.w.setWindowTitle('Mein Titel-Test')
```

**An advanced instance patching example** This could work with the Qtdragon screen's handler file. Here we show how to add new functions and override existing ones.

```
# Needed to instance patch.
# reference: https://ruivieira.dev/python-monkey-patching-for-readability.html
import types

# import the handlerfile to get reference to its libraries.
# use <screenname>_handler
import qtdragon_handler as hdlr

# This is actually an unbounded function with 'obj' as a parameter.
```

```
# You call this function without the usual preceding 'self.'.
# This is because will will not be patching it into the original handler class instance
# It will only be called from code in this file.
def test_function(obj):
    print(dir(obj))

# This is a new function we will added to the existing handler class instance.
# Notice it calls the unbounded function with 'self' as an parameter 'self' is the only global reference available.
# It references the window instance.
def on_keycall_F10(self,event,state,shift,cntrl):
    if state:
        print ('F10')
        test_function(self)

# This will be used to override an existing function in the existing handler class instance.
# Note, we also call a copy of the original function too.
# This shows how to extend an existing function to do extra functions.
def on_keycall_F11(self,event,state,shift,cntrl):
    if state:
        self.on_keycall_F11_super(event,state,shift,cntrl)
        print ('Hello')

# We are referencing the KEYBIND library that was instantiated in the original handler class instance
# by adding 'hdlr.' to it (from the imp).
# This function tells KEYBIND to call 'on_keycall_F10' when F10 is pressed.
hdlr.KEYBIND.add_call('Key_F10','on_keycall_F10')

# Here we are instance patching the original handler file to add a new function
# that calls our new function (of the same name) defined in this file.
self.on_keycall_F10 = types.MethodType(on_keycall_F10, self)

# Here we are defining a copy of the original 'on_keycall_F11' function,
# so we can call it later. We can use any valid, unused function name.
# We need to do this before overriding the original function.
self.on_keycall_F11_super = self.on_keycall_F11

# Here we are instance patching the original handler file to override an existing function
# to point to our new function (of the same name) defined in this file.
self.on_keycall_F11 = types.MethodType(on_keycall_F11, self)


# add a new pin to the screen:

# pin callback to print the state
def new_pin_changed(data):
    print(data)

# Special function that gets called before the HAL component is set ready.
# Here we used the function to add a bit input pin with a callback.
def after_override__(self):
    try:
        pin = hdlr.QHAL.newpin("new_pin", hdlr.QHAL.HAL_BIT, hdlr.QHAL.HAL_IN)
        pin.value_changed.connect(new_pin_changed)
    except Exception as e:
        print(e)

# Here we are instance patching the original handler file to add a new function
# that calls our new function (of the same name) defined in this file.
```

```
self.after_override__ = types.MethodType(after_override__, self)
```

If you wish to **modify a stock screen** with full control, *copy its UI and handler file to your configuration folder*.

Es gibt ein QtVCP-Panel, das dabei hilft:

- Öffnen Sie ein Terminal und führen den folgenden Befehl aus:

  ```
  qtvcp copy
  ```

- Wählen Sie den Bildschirm und den Zielordner im Dialog
- Wenn Sie Ihren Bildschirm anders **benennen** möchten als den Standardnamen des eingebauten Bildschirms, ändern Sie den *Basisnamen* im Bearbeitungsfeld.
- There should be a folder in the config folder; for screens: named *<CONFIG FOLDER>/qtvcp/screens/* for panels: named *<CONFIG FOLDER>/qtvcp/panels/* add the folders if they are missing and copy your folder/files in it.
- Bestätigen, um alle Dateien zu kopieren
- Delete the files you don’t wish to modify so that the original files will be used.

## 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.

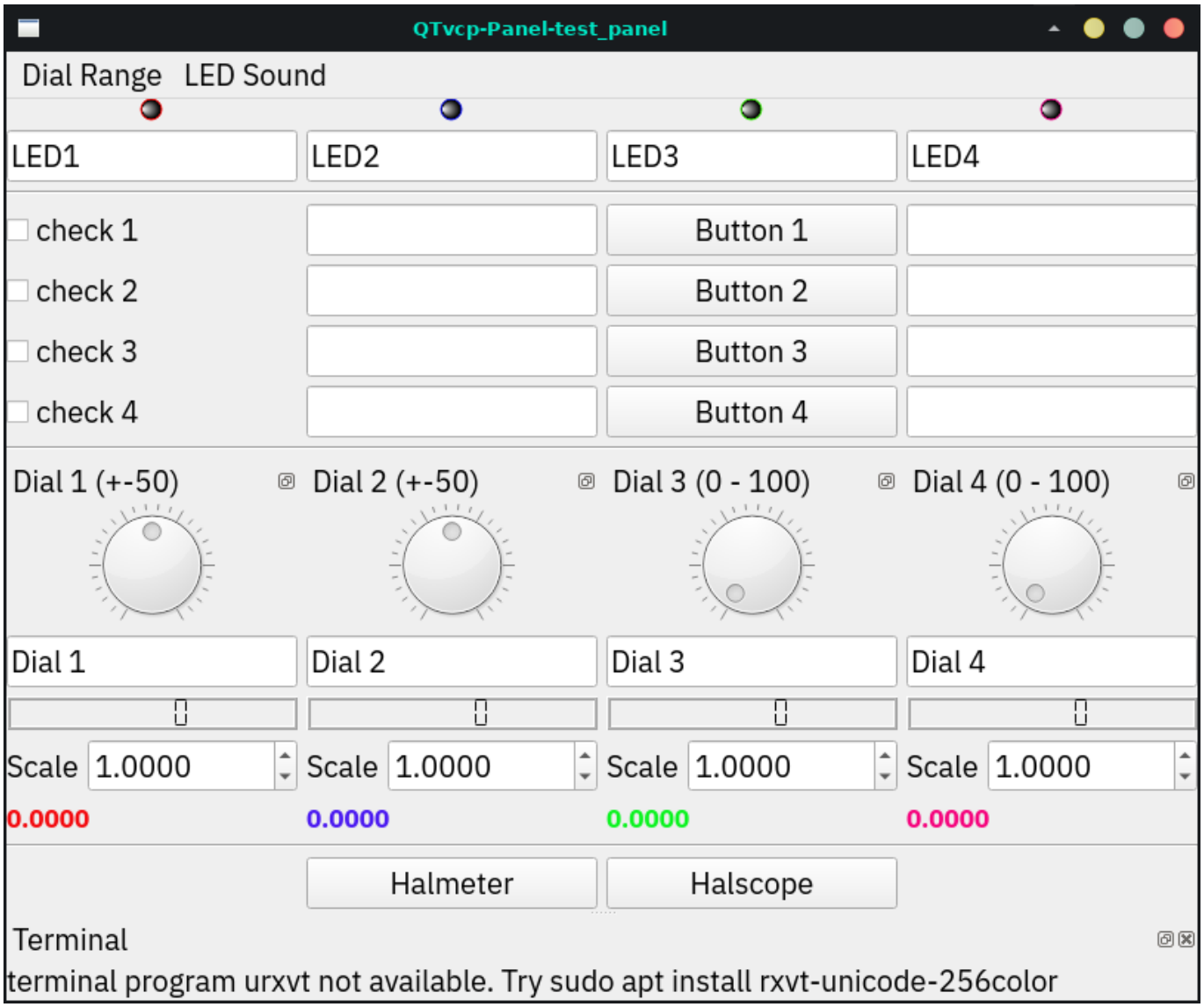



Abbildung 12.62: QtVCP HAL Test Integriertes Panel

**cam_align**
Ein Kameraanzeige-Widget für die Rotationsausrichtung.

Abbildung 12.63: cam_align – Kameraausrichtung VCP

**sim_panel**
: A small control panel to simulate MPG jogging controls etc.
For simulated configurations.

Abbildung 12.64: QtVCP Sim Builtin Panel

**vismach_mill_xyz**
: 3D OpenGL view of a 3-axis milling machine.

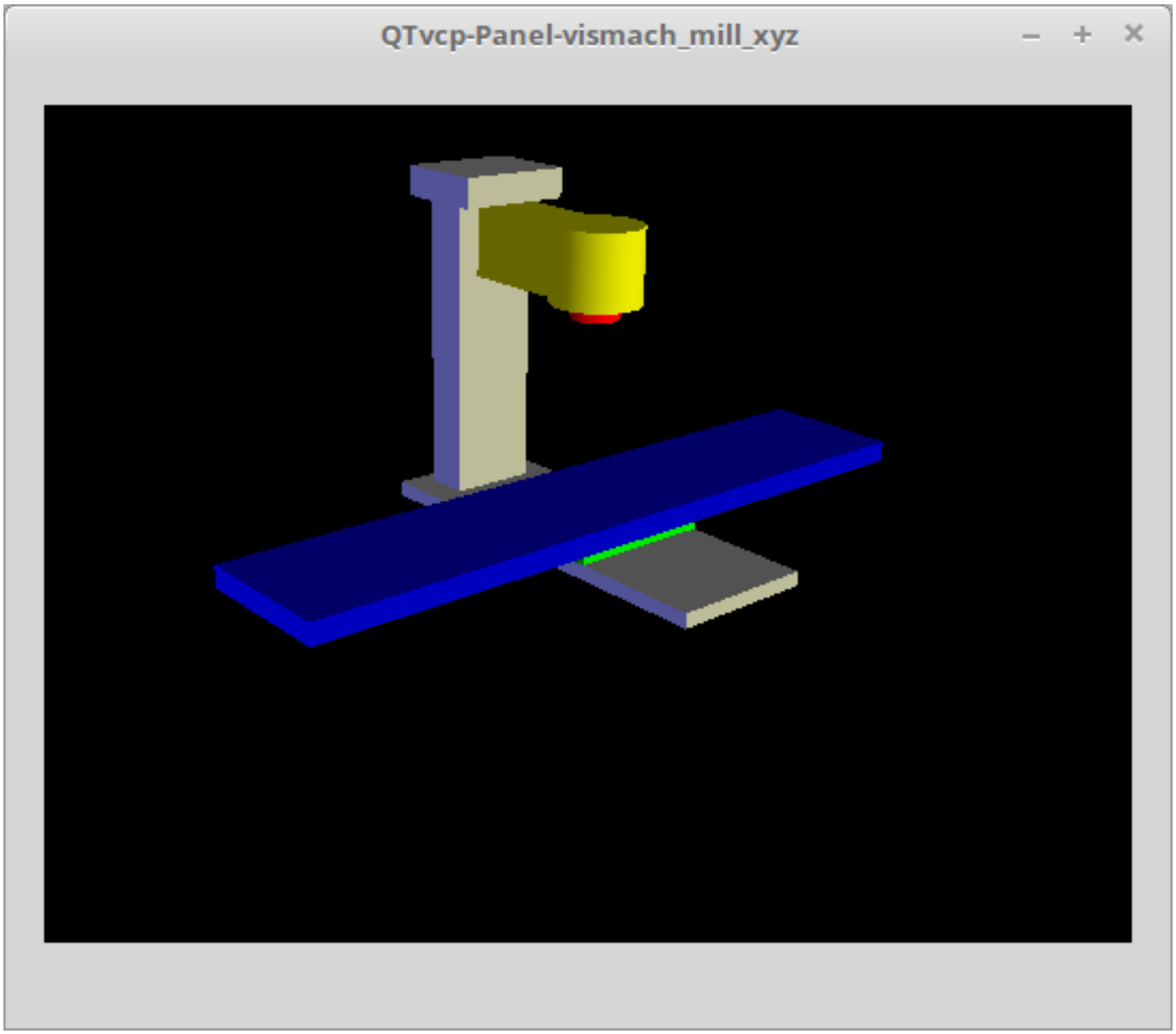



Abbildung 12.65: QtVismach - 3-Axis Mill Builtin Panel

You can load these from the terminal or from a HAL file with this basic command:

```
loadusr qtvcp test_panel
```

But more typically like this:

```
loadusr -Wn test_panel qtvcp test_panel
```

In this way HAL will wait till the HAL pins are made before continuing on.

#### 12.5.3.2 Benutzerdefinierte Bedienfelder

Sie können natürlich **Ihr eigenes Panel erstellen und laden**.

Wenn Sie eine UI-Datei mit dem Namen `my_panel.ui` und eine HAL-Datei mit dem Namen `my_panel.hal` erstellt haben, würden Sie diese dann von einem Terminal aus laden mit:

```
halrun -I -f my_panel.hal
```

### Beispiel einer HAL-Datei, die ein QtVCP-Panel lädt

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime 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 diesem Fall laden wir `qtvcp` mit **`-Wn`**, das wartet, bis das Panel das Laden beendet hat, bevor es mit der Ausführung des nächsten HAL-Befehls fortfährt.
Damit soll *gewährleistet werden, dass die vom Panel erstellten HAL-Pins tatsächlich fertig sind*, falls sie im Rest der Datei verwendet werden.

### 12.5.4 Build A Simple Clean-sheet Custom Screen

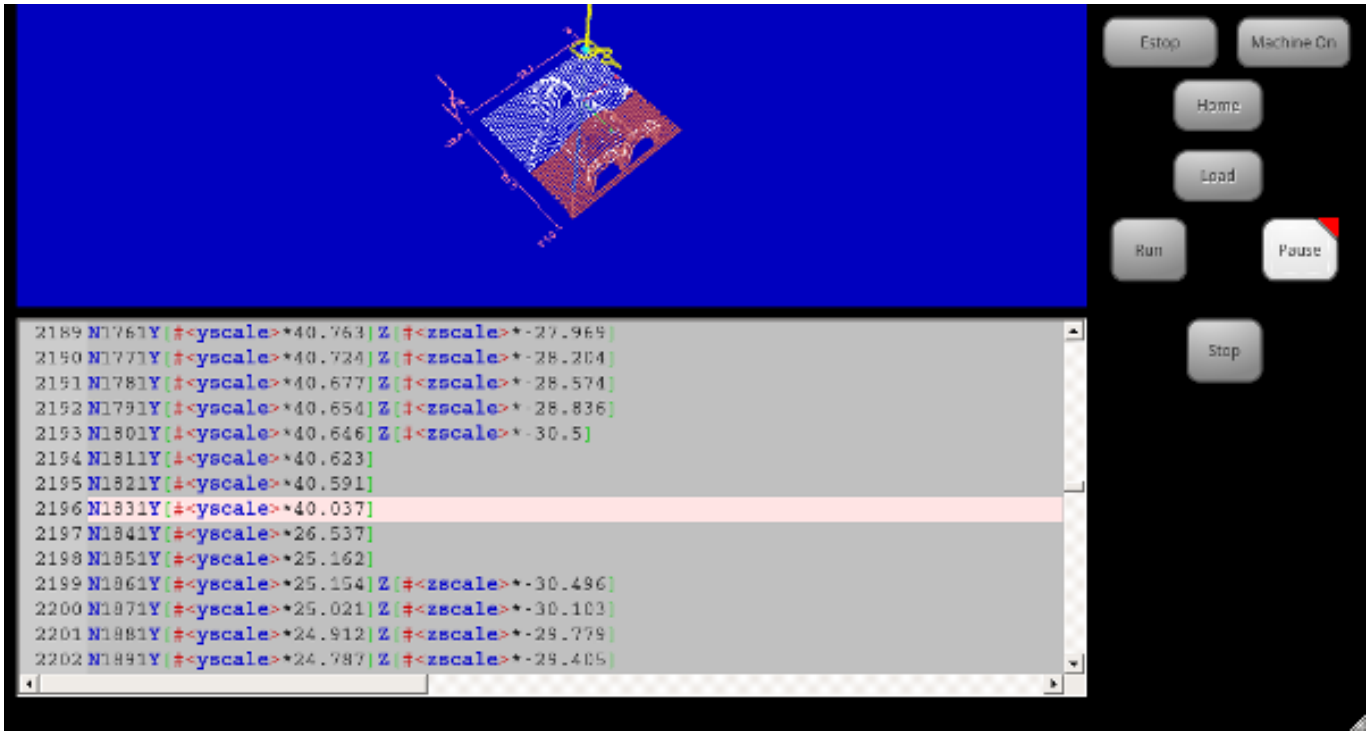



Abbildung 12.66: QtVCP Hässlicher benutzerdefinierter Bildschirm

#### 12.5.4.1 Übersicht

So erstellen Sie ein Bedienfeld oder einen Bildschirm:

- Verwenden Sie Qt Designer, um ein Design zu erstellen, das Ihnen gefällt, und speichern Sie es in Ihrem Konfigurationsordner unter einem Namen Ihrer Wahl, der mit `.ui` endet
- Ändern Sie die Konfigurations-INI-Datei, um QtVCP mit Ihrer neuen `.ui`-Datei zu laden.
- Dann verbinden Sie alle erforderlichen HAL-Kenntnisse in einer HAL-Datei.

#### 12.5.4.2 Holen Sie sich Qt Designer, um LinuxCNC-Widgets einzubinden

**Install Qt Designer** Zuerst müssen Sie den **Qt Designer installieren**.
Die folgenden Befehle sollten ihn zu Ihrem System hinzufügen, oder verwenden Sie Ihren Paketmanager, um dasselbe zu tun:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

**Hinzufügen des Links qtvcp_plugin.py zum Qt Designer Suchpfad** Dann müssen Sie einen Link zu `qtvcp_plugin.py` in einem der Ordner hinzufügen, in denen Qt Designer suchen wird.

In einer *RIP* (engl. Abkürzung von ”run in place”, d.h. das Programm started dort wo es durch den Quellcode auch kompiliert wurde) Version von LinuxCNC wird `qtvcp_plugin.py` sein:

`'~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py'`

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'
```

Legen Sie einen symbolischen Link auf die obige Datei an und verschieben Sie sie an einen der Orte, an denen Qt Designer sucht.

Qt Designer sucht an diesen beiden Stellen nach Links (wählen Sie einen aus):

```
'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' or
'~/.designer/plugins/python'
```

Möglicherweise müssen Sie den Ordner `plugins/python` erstellen.

Starten Sie Qt Designer:

- Für eine *RIP-Installation*:
  Öffnen Sie ein Terminal, setzen Sie die Umgebungsvariablen für LinuxCNC <1>, dann laden Sie Qt Designer <2> mit :

```
. scripts/rip-environment    (1)
designer -qt=5               (2)
```

- Für eine *Paketinstallation*:
  Öffnen Sie ein Terminal und geben Sie ein:

```
designer -qt=5
```

Wenn alles gut geht, wird Qt Designer gestartet und Sie werden die auswählbaren LinuxCNC Widgets auf der linken Seite sehen.

#### 12.5.4.3 Build The Screen `.ui` File

**Create MainWindow Widget** Wenn Qt Designer zum ersten Mal gestartet wird, erscheint ein *'New Form' Dialog*.
Wählen Sie *'Main Window'* und drücken Sie die Schaltfläche *'Create'*.
Ein *`MainWindow`-Widget* wird angezeigt.

We are going to make this window a specific non resizeable size:

Minimale und maximale Größe des Hauptfensters festlegen

- Fassen Sie die Ecke des Fensters an und ändern Sie die Größe auf eine geeignete Größe, z. B. 1000x600.

- Klicken Sie mit der rechten Maustaste auf das Fenster und klicken Sie auf *Mindestgröße* einstellen.
- Wiederholen Sie dies und stellen Sie *maximale Größe* ein.

Unser Beispiel-Widget ist nun nicht mehr größenveränderbar.

**Hinzufügen des Widgets ScreenOptions** Ziehen Sie das `ScreenOptions`-Widget per Drag-and-Drop an eine beliebige Stelle im Hauptfenster.

Dieses Widget fügt visuell nichts hinzu, richtet aber einige **allgemeine Optionen** ein.

Es wird empfohlen, dieses Widget immer *vor allen anderen* hinzuzufügen.

Klicken Sie mit der rechten Maustaste auf das Hauptfenster, nicht auf das ”ScreenOptions”-Widget, und stellen Sie ”Layout” auf ”Vertikal”, um ”ScreenOptions” in voller Größe anzuzeigen.

**Add Panel Content** Auf der rechten Seite befindet sich ein Panel mit Registerkarten für einen *Eigenschaftseditor* und einen *Objektinspektor*.

Klicken Sie im Objektinspektor auf *ScreenOptions*.
Wechseln Sie dann zum Eigenschaftseditor (engl. property editor) und schalten Sie unter der Überschrift *ScreenOptions* die **`filedialog_option`** um.

Ziehen Sie ein **`GCodeGraphics`** *widget* und ein **`GcodeEditor`** *widget* per Drag and Drop.
Platzieren Sie sie und ändern Sie die Größe, wie Sie es für richtig halten, und lassen Sie etwas Platz für Schaltflächen.

**Add Action Buttons** Fügen Sie dem Hauptfenster 7 Aktionsschaltflächen hinzu.

Wenn Sie auf die Schaltfläche (engl. button) doppelklicken, können Sie Text hinzufügen.
Bearbeiten Sie die Schaltflächenbeschriftungen für ”Notaus” (engl. ”E-stop”), ”Maschine ein”, Referenzpunkt (engl. ”Home”), ”Laden” (engl. ”Load”), ”Ausführen” (engl. ”Run”), ”Pause” und ”Stopp”.

Aktionsschaltflächen sind standardmäßig auf keine Aktion eingestellt, daher müssen wir die Eigenschaften für definierte Funktionen ändern. Sie können die Eigenschaften bearbeiten:

- direkt im *Eigenschaften-Editor* auf der rechten Seite des Qt-Designers, oder
- praktischerweise lassen sich durch einen Doppelklick mit der linken Maustaste auf die Schaltfläche ein Dialogfeld ”Eigenschaften” aufrufen, was die Auswahl von Aktionen ermöglicht, wobei nur die für die Aktion relevanten Daten angezeigt werden.

Wir werden zunächst den bequemen Weg beschreiben:

- Klicken Sie mit der rechten Maustaste auf die Schaltfläche *Maschine ein* (engl.machine on) und wählen Sie *Aktionen festlegen* (engl. set actions).
- Wenn das Dialogfeld angezeigt wird, verwenden Sie die Combobox, um zu `MASCHINENSTEUERUNGEN - Maschine ein`(engl. `MACHINE CONTROLS - Machine On`) zu navigieren.
- In diesem Fall gibt es keine Option für diese Aktion, also wählen Sie ”OK”.

Now the button will turn the machine on when pressed.

Und nun der direkte Weg mit dem Eigenschaftseditor von Qt Designer:

- Wählen Sie die Schaltfläche ”Maschine ein”.
- Gehen Sie zum Eigenschaftseditor auf der rechten Seite von Qt Designer.
- Blättern Sie nach unten, bis Sie die Überschrift *ActionButton* finden.
- Klicken Sie auf das Kontrollkästchen für die Aktion ”Machine_on”, das Sie in der Liste der Eigenschaften und Werte sehen.

Die Taste steuert nun das Ein- und Ausschalten der Maschine.

Machen Sie das Gleiche für alle anderen Schaltflächen und fügen Sie noch einen hinzu:

- Bei der Schaltfläche ”Home” müssen wir auch die Eigenschaft `joint_number` auf `1` ändern.
  Dadurch wird der Controller angewiesen, *alle Achsen* und nicht nur eine bestimmte Achse zu *referenzieren*.
- Mit dem Button ”Pause”:

- Unter der Überschrift `Indicated_PushButton` überprüfen Sie die `Indicator_option`.
- Unter der Überschrift `QAbstactButton` markieren Sie `checkable`.

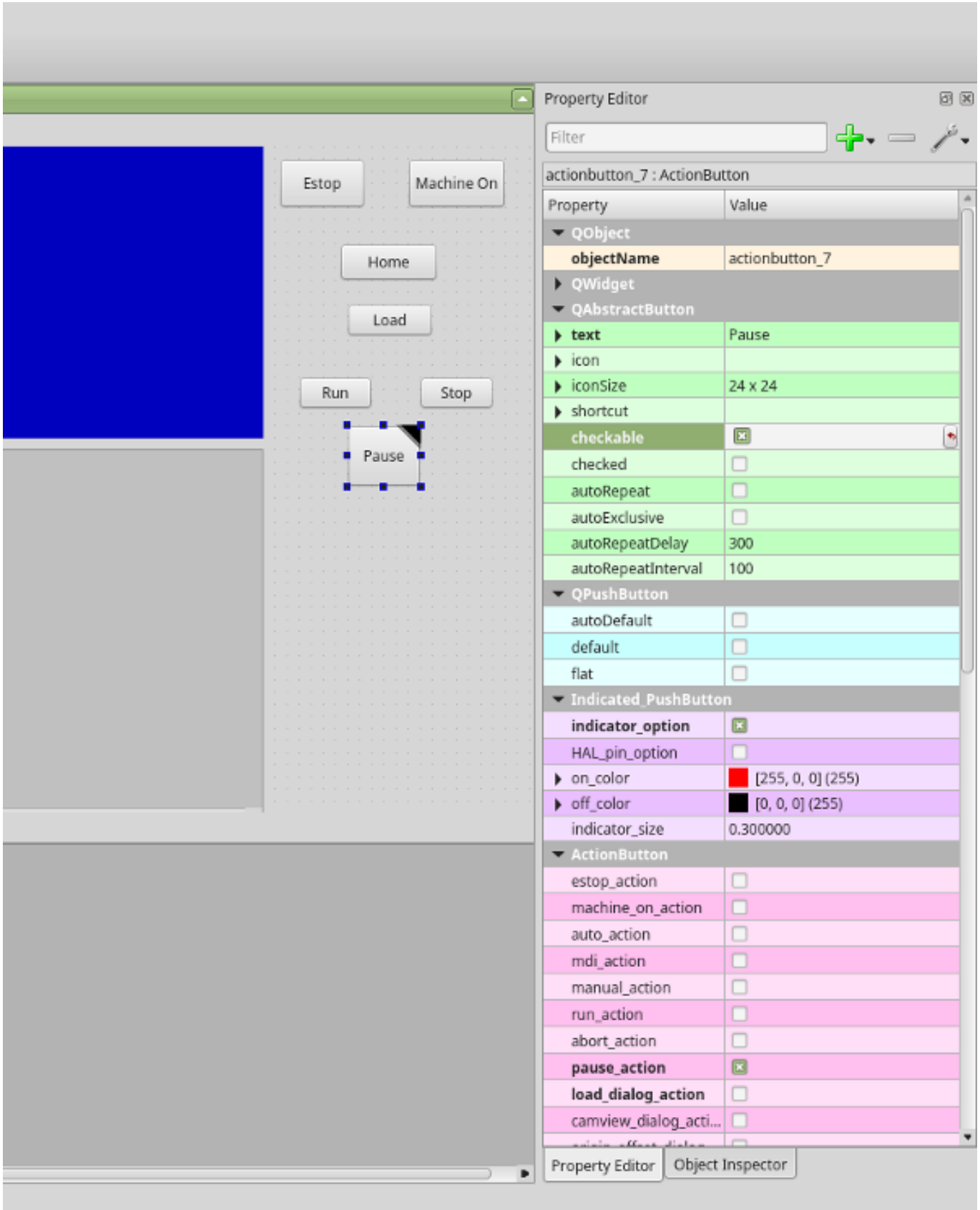



Abbildung 12.67: Qt Designer: Auswahl der Eigenschaften des Pause-Buttons (Schaltfläche)

**.ui-Datei speichern** Diesen Entwurf müssen wir dann als `tester.ui` im Ordner `sim/qtvcp` speichern.

We are saving it as *tester* as that is a file name that QtVCP recognizes and will use a built in handler file to display it.

#### 12.5.4.4 Handler-Datei

A handler file is **required**.

Er ermöglicht das Schreiben von Anpassungen in Python.

Zum Beispiel werden *Tastatursteuerungen* normalerweise in die Handler-Datei geschrieben.

In this example, the built in file `tester_handler.py` is automatically used: It does the minimum required to display the `tester.ui` defined screen and do basic keyboard jogging.

#### 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=<Dateiname>`**
: Der Konvention nach wäre `<Dateiname>` als +`<Bildschirm_name>_postgui.hal`+ genannt, aber es kann jeder legale Dateiname sein.
Sie können *mehrere `POSTGUI_HALFILE`-Zeilen* in der INI haben: jede wird nacheinander in der Reihenfolge ausgeführt, in der sie erscheint.
Diese Befehle werden *nach der Erstellung des Bildschirms ausgeführt*, um sicherzustellen, dass die HAL-Pins des Widgets verfügbar sind.

**`POSTGUI_HALCMD=<Befehl>`**
: `<Befehl>` wäre *jeder gültige HAL-Befehl*.
Sie können *mehrere `POSTGUI_HALCMD`-Zeilen* in der INI haben: jede wird nacheinander in der Reihenfolge ausgeführt, in der sie erscheint.
Um zu garantieren, dass die HAL-Pins des Widgets verfügbar sind, werden diese Befehle ausgeführt:

- *after the screen is built*,
- *nachdem alle POSTGUI_HALFILEs ausgeführt wurden*.

In unserem Beispiel gibt es keine HAL-Pins zu verbinden.

### 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 is one we want the keypress events to go to
            flag = False
            receiver2 = receiver
            while receiver2 is not None and not flag:
                if isinstance(receiver2, QtWidgets.QDialog):
                    flag = True
                    break
                if isinstance(receiver2, MDI_WIDGET):
                    flag = True
                    break
                if isinstance(receiver2, GCODE):
                    flag = True
                    break
                receiver2 = receiver2.parent()

            if flag:
                if isinstance(receiver2, GCODE):
                    # if in manual do our keybindings - otherwise
                    # send events to G-code widget
                    if STATUS.is_man_mode() == False:
                        if is_pressed:
                            receiver.keyPressEvent(event)
                            event.accept()
                        return True
                elif is_pressed:
                    receiver.keyPressEvent(event)
                    event.accept()
                    return True
                else:
                    event.accept()
                    return True

        if event.isAutoRepeat():return True

        # ok if we got here then try keybindings
        try:
            return KEYBIND.call(self,event,is_pressed,shift,cntrl)
        except NameError as e:
            LOG.debug('Exception in KEYBINDING: {}'.format (e))
        except Exception as e:
            LOG.debug('Exception in KEYBINDING:', exc_info=e)
            print('Error in, or no function for: %s in handler file for-%s'%(KEYBIND. ↩
                convert(event),key))
            return False

    ########################
    # CALLBACKS FROM STATUS #
    ########################

    ######################
    # CALLBACKS FROM FORM #
    ######################

    ####################
    # GENERAL FUNCTIONS #
    ####################
```

```
    # keyboard jogging from key binding calls
    # double the rate if fast is true
    def kb_jog(self, state, joint, direction, fast = False, linear = True):
        if not STATUS.is_man_mode() or not STATUS.machine_is_on():
            return
        if linear:
            distance = STATUS.get_jog_increment()
            rate = STATUS.get_jograte()/60
        else:
            distance = STATUS.get_jog_increment_angular()
            rate = STATUS.get_jograte_angular()/60
        if state:
            if fast:
                rate = rate * 2
            ACTION.JOG(joint, direction, rate, distance)
        else:
            ACTION.JOG(joint, 0, 0, 0)

    ######################
    # KEY BINDING CALLS #
    ######################

    # Machine control
    def on_keycall_ESTOP(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_ESTOP_STATE(STATUS.estop_is_clear())
    def on_keycall_POWER(self,event,state,shift,cntrl):
        if state:
            ACTION.SET_MACHINE_STATE(not STATUS.machine_is_on())
    def on_keycall_HOME(self,event,state,shift,cntrl):
        if state:
            if STATUS.is_all_homed():
                ACTION.SET_MACHINE_UNHOMED(-1)
            else:
                ACTION.SET_MACHINE_HOMING(-1)
    def on_keycall_ABORT(self,event,state,shift,cntrl):
        if state:
            if STATUS.stat.interp_state == linuxcnc.INTERP_IDLE:
                self.w.close()
            else:
                self.cmnd.abort()

    # Linear Jogging
    def on_keycall_XPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, 1, shift)

    def on_keycall_XNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 0, -1, shift)

    def on_keycall_YPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, 1, shift)

    def on_keycall_YNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 1, -1, shift)

    def on_keycall_ZPOS(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, 1, shift)

    def on_keycall_ZNEG(self,event,state,shift,cntrl):
        self.kb_jog(state, 2, -1, shift)
```

```
    def on_keycall_APOS(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, 1, shift, False)

    def on_keycall_ANEG(self,event,state,shift,cntrl):
        pass
        #self.kb_jog(state, 3, -1, shift, linear=False)

    ############################
    # **** closing event **** #
    ############################

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)

################################
# required handler boiler code #
################################

def get_handlers(halcomp,widgets,paths):
     return [HandlerClass(halcomp,widgets,paths)]
```

#### 12.5.5.2 IMPORT Section

Dieser Abschnitt ist für **Import der erforderlichen Bibliotheksmodule** für Ihren Bildschirm.

Es wäre typisch, die QtVCP-Bibliotheken *keybinding*, *Status* und *Action* zu importieren.

#### 12.5.5.3 Abschnitt INSTANTIATE BIBRARIES

Indem wir die Bibliotheken hier instanziieren, **erzeugen wir eine globale Referenz**.

Sie können dies an den Befehlen erkennen, denen kein ”self” vorangestellt ist.

By convention we *capitalize the names of globally referenced libraries*.

#### 12.5.5.4 HANDLER CLASS Section

Der **angepasste Code** wird *in einer Klasse platziert, damit QtVCP ihn verwenden kann*.

Dies ist die Definition der Handler-Klasse.

#### 12.5.5.5 INITIALIZE Section

Wie alle Python-Bibliotheken wird die **+`__init__`+-Funktion** aufgerufen, wenn die Bibliothek *erstmals* instanziiert wird.

Hier können Sie *Standardwerte*, *Referenzvariablen* und *globale Variablen* einrichten.

Die Referenzen der Widgets sind zu diesem Zeitpunkt nicht verfügbar.

Die Variablen `halcomp`, `widgets` und `paths` ermöglichen den Zugriff auf QtVCP’s HAL-Komponenten, Widgets bzw. Pfadinformationen.

#### 12.5.5.6 Abschnitt zu BESONDEREN FUNKTIONEN

Es gibt mehrere *special functions*, nach denen QtVCP in der Handlerdatei sucht.

Wenn QtVCP diese findet, ruft es sie auf, wenn nicht, ignoriert es sie stillschweigend.

**`class_patch__(self):`**
*Class patching*, auch bekannt als *monkey patching*, erlaubt es, **Funktionsaufrufe in einem importierten Modul zu überschreiben**.
Klassen-Patching muss *vor der Instanziierung des Moduls* durchgeführt werden und *verändert alle danach erstellten Instanzen*.
Ein Beispiel wäre das Patchen von Schaltflächenaufrufen aus dem G-Code-Editor, um stattdessen Funktionen in der Handler-Datei aufzurufen.

**`initialized__(self):`**
Diese Funktion wird *aufgerufen, nachdem die Widgets und HAL-Pins erstellt wurden*.
Sie können hier die Widgets und HAL-Pins manipulieren oder weitere HAL-Pins hinzufügen.
In der Regel gibt es

- Einstellungen überprüft und eingestellt,
- styles applied to widgets,
- Status von LinuxCNC verbunden mit Funktionen.
- Tastenbelegungen würden hinzugefügt.

**`after_override__(self):`**
This function is called after the optional override file is loaded but
before the optional HAL file is loaded or HAL component is set ready.

**`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):`**
Diese Funktion liefert **rohe Tastendruckereignisse**.
Sie hat _Vorrang vor dem `verarbeiteten_Tastenereignis`.

**`keyrelease_event__(receiver, event):`**
Diese Funktion gibt **raw key release events** aus.
Es hat *Vorrang vor* dem `processed_key_event`.

**`before_loop__(self):`**
Diese Funktion wird *kurz vor dem Eintritt in die Qt-Ereignisschleife* aufgerufen. Zu diesem Zeitpunkt sind alle Widgets/Bibliotheken/Initialisierungscodes abgeschlossen und der Bildschirm wird bereits angezeigt.

**`system_shutdown_request__(self):`**
If present, this function **overrides the normal function called for total system shutdown**.
It could be used to do *pre-shutdown housekeeping*.
+
The Linux *system will not shutdown if using this function*, you will have to do that yourself.
QtVCP/LinuxCNC will terminate without a prompt once this function returns.

**`closing_cleanup__(self):`**
Diese Funktion wird *kurz vor dem Schließen des Bildschirms* aufgerufen. Sie kann verwendet werden, um vor dem Schließen aufzuräumen.

#### 12.5.5.7 STATUS CALLBACKS Abschnitt

Konventionell würden Sie hier Funktionen unterbringen, die **Rückrufe von STATUS-Definitionen** sind.

#### 12.5.5.8 CALLBACKS FROM FORM Abschnitt

Konventionell würden Sie hier Funktionen ablegen, die **Rückrufe von den Widgets sind, die mit dem MainWindow** im Qt Designer-Editor verbunden sind.

#### 12.5.5.9 GENERAL FUNCTIONS Section

Konventionell werden hier die **allgemeinen Funktionen** untergebracht.

#### 12.5.5.10 KEY BINDING Section

Wenn Sie die `Keybinding`-Bibliothek_ verwenden, platzieren Sie hier Ihre **benutzerdefinierten Tastenaufrufroutinen**.

The function signature is:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

`KEY` ist der Code (aus der Keybindings-Bibliothek) für den gewünschten Schlüssel.

#### 12.5.5.11 CLOSING EVENT Section

Wenn Sie die Funktion **`closeEvent` hier einfügen, werden Schließungsereignisse** abgefangen.

Dies *replaces eine vordefinierte 'closeEvent'*-Funktion von QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

---

**Anmerkung**
It is usually better to use the special `closing_cleanup__` function.

---

### 12.5.6 Connecting Widgets to Python Code

Es ist möglich, Widgets über **Signale und Slots** mit Python-Code zu verbinden.

Auf diese Weise können Sie:

- *LinuxCNC-Widgets neue Funktionen geben*, oder
- *Standard Qt-Widgets zur Steuerung von LinuxCNC verwenden*.

#### 12.5.6.1 Übersicht

**Im Qt Designer-Editor**:

- You *create user function slots*
- Sie *verbinden die Slots mit Widgets, indem Sie Signale verwenden*.

**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.

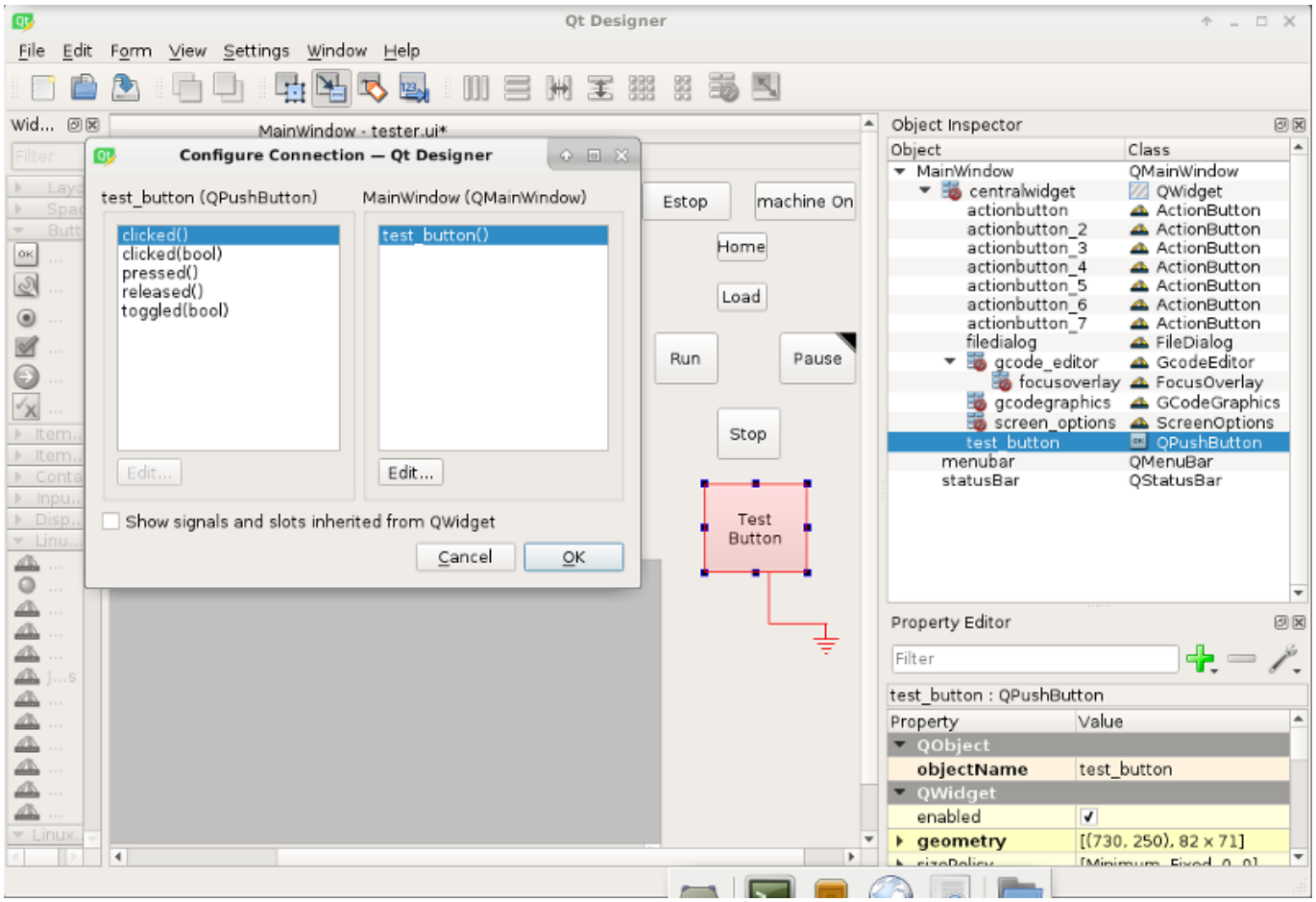



Abbildung 12.68: Qt Designer Signal/Slot Selection

#### 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 Mehr zum Thema

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
```

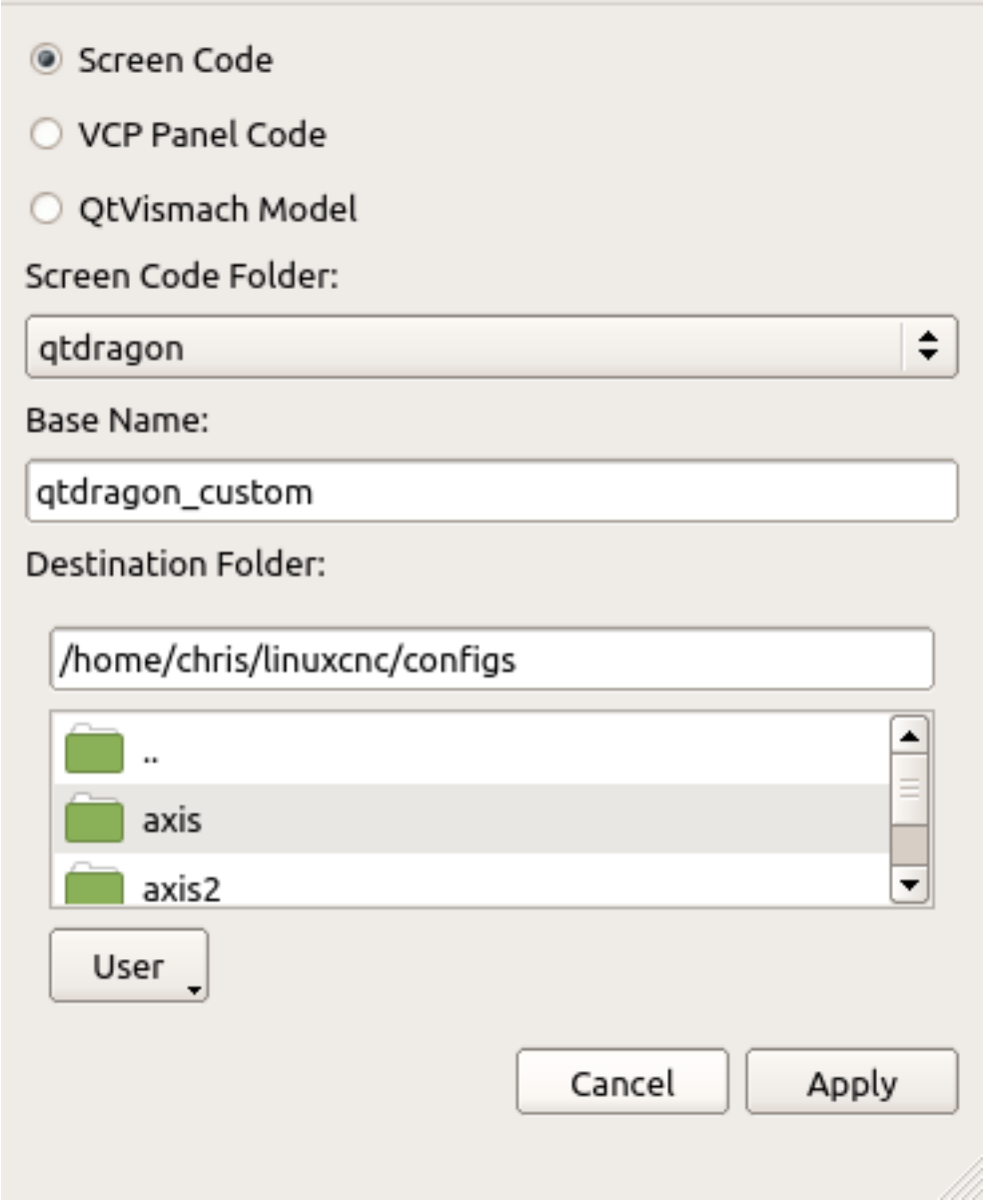
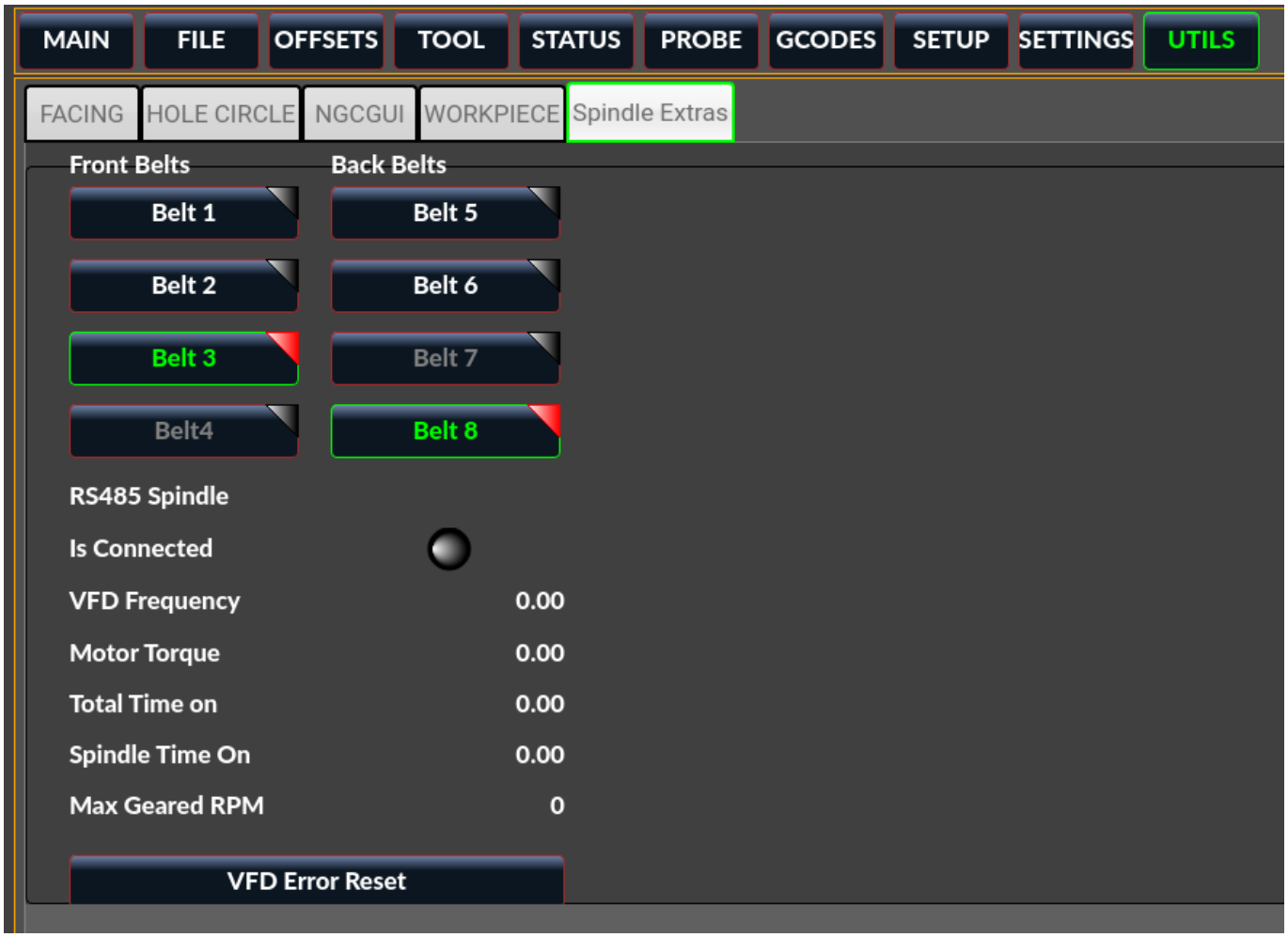



Abbildung 12.69: QtVCP `copy` Dialog - Screen, VCP Panel or QtVismach Code Copy Panel

#### 12.6.1.2 spindle_belts

This panel is designed to display additional RS485 VFD data and also to configure a 4 sheave, 2 belt spindle drive via a series of buttons.




In addition, it is also a useful template to use for your custom panel because it includes:

- Display of additional HAL data
- Buttons and button groups
- Dynamic changes to button enabled/disabled state based on the state of other buttons
- Saving data to the qtdragon.prefs file
- Custom button to reset the VFD

Modify this panel to suit your own requirements. Most common features are used. The advantage of using panels is that it separates your custom display code from the qtdragon core code so upgrading the system will not break your customization.

- A spindle drive (for instance VFDMOD)
- A custom component that scales the VFD frequency to obtain the desired spindle speed.
- A belt driven spindle that uses two belts and an intermediate idler pulley much like a drill press.
- Connect the input pins qtdragon.belts.<pin-name> in your postgui HAL file.

The belts are broken into two button groups, the front belts and the rear belts. These are numbered as per the plate on the machine. Buttons in a group are mutually exclusive, i.e., only one can be selected in the group.

Additionally, it’s not possible to have both belts on the same level with this kind of mechanism because you cannot fit two belts to the one idler pulley sheave. So if a belt is selected, its opposite button is disabled. E.g., if belt 3 is selected, belt 7 is disabled.

Add these lines to the [DISPLAY] section in your .ini file
The example tab_location is for the QtDragon screen.

```
EMBED_TAB_NAME=Spindle Extras
EMBED_TAB_COMMAND=qtvcp spindle_belts
EMBED_TAB_LOCATION=tabWidget_utilities
```

Here is how to load `spindle_belt` from a HAL script:

```
loadusr qtvcp spindle_belts
```

Customizing the panel:

- Copy the files located in /user/share/qtvcp/qtdragon/panels/belts to: ~/linuxcnc/configs/<my_configuratio
(you can use the copy dialog panel to do this)
- Edit belts.ui with designer.
- Edit belts_handler.py with a text editor
- Connect the relevant pins in a postgui.hal file
- Make sure your postgui file is loaded by your .ini file.

For information on the finer points, consult the QtVCP and QtDragon documentation. The Python handler file also provides a useful template for any custom panel.

#### 12.6.1.3 `test_dial`

- Dieses Panel hat ein * Einstellrad, das S32- und Float-HAL-Ausgangspins einstellt *.
- 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
```

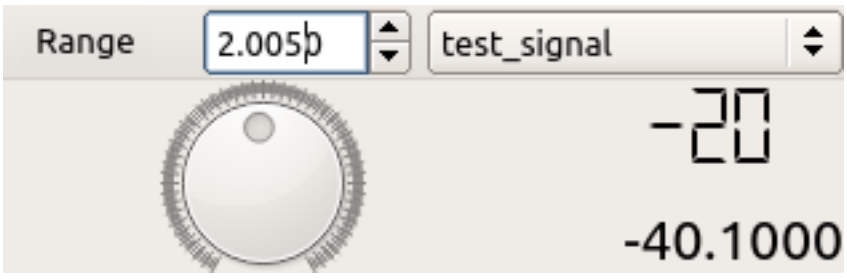



Abbildung 12.70: QtVCP `test_dial` Panel - Test Dial VCP

#### 12.6.1.4 test_button

- Dieses Panel hat eine **Schaltfläche, die einen HAL-Pin** setzt.
- 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`.

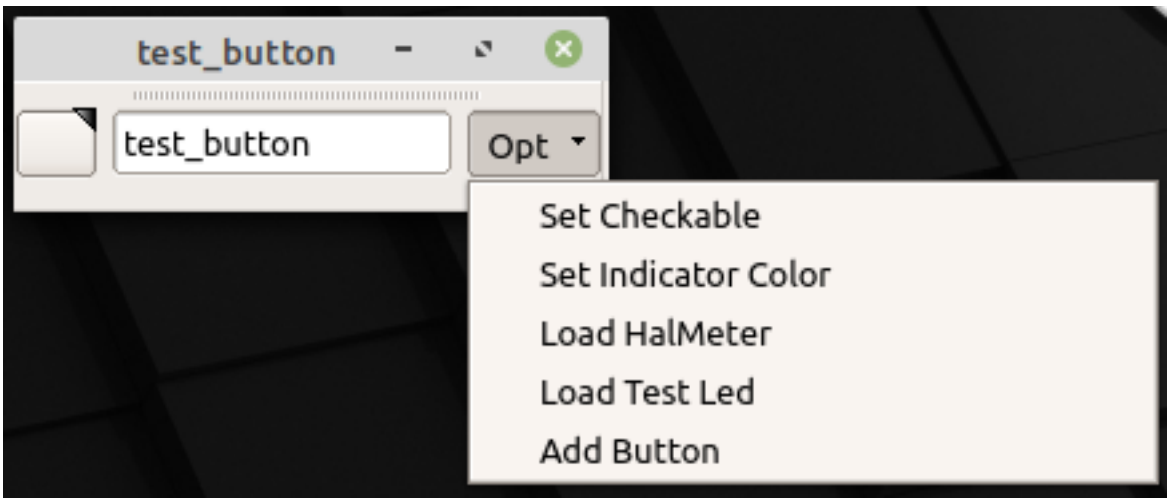



Abbildung 12.71: QtVCP `test_button` - Test Button VCP

#### 12.6.1.5 test_led

- Dieses Panel verfügt über eine *LED, die ausgewählt werden kann, um HAL-Bit-Pins / Signale * zu beobachten.
- 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*.

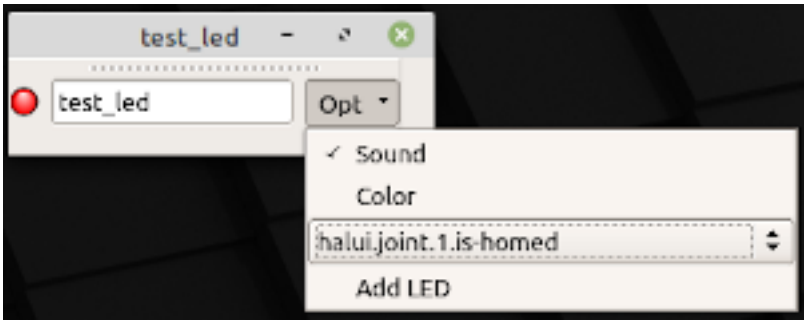



Abbildung 12.72: QtVCP `test_dial` Panel - Test LED VCP

#### 12.6.1.6 `test_panel`

**Collection of useful widgets for testing HAL component**, including speech of LED state.

```
loadusr qtvcp test_panel
```

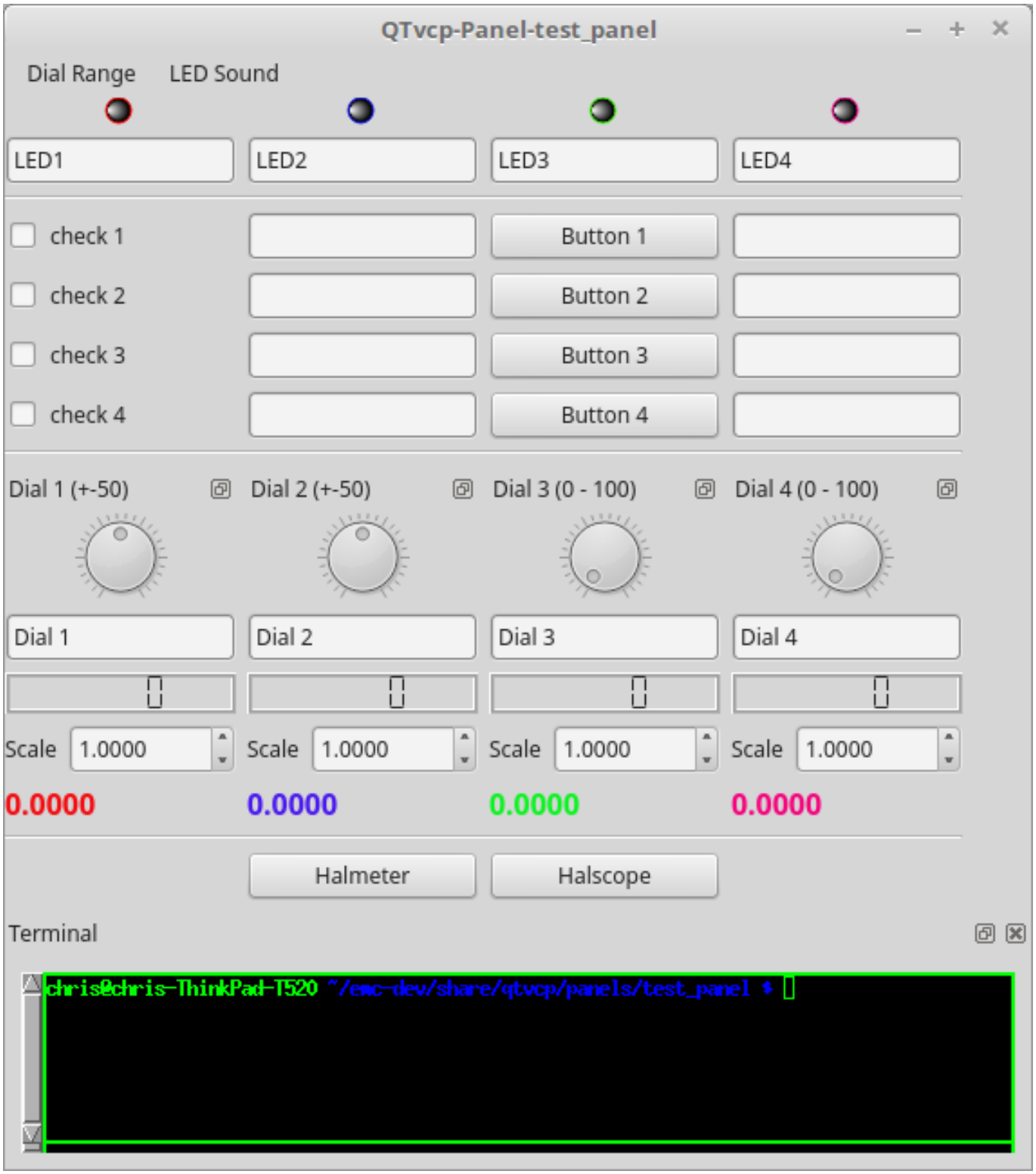



Abbildung 12.73: QtVCP `test_panel` - HAL Component Testing Panel

#### 12.6.1.7 `cam_align`

A **camera display widget for rotational alignment**.

Abbildung 12.74: QtVCP `cam_align` Panel - Camera Based Alignment Panel

**Anwendung** Add these lines to the INI file:

```
[DISPLAY]
EMBED_TAB_NAME = cam_align
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID}  ↩
    cam_align
# The following line is needed if embedding in GMOCCAPY
EMBED_TAB_LOCATION = ntb_preview
```

You can add window width and height size, rotation increment, and camera number from the INI with -o options.

```
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID} -o size ↩
    =400,400 -o rotincr=.2 -o camnumber=0 cam_align
```

Mouse controls:

- left mouse single click - increase cross hair rotation one increment

- right mouse single click - decrease cross hair rotation one increment
- middle mouse single click - cycle through rotation increments
- left mouse hold and scroll - scroll camera zoom
- right mouse hold and scroll - scroll cross hair rotation angle
- mouse scroll only - scroll circle diameter
- left mouse double click - reset zoom
- right mouse double click - reset rotation
- middle mouse double click - reset circle diameter

To use the top buttons you have to assign a command (or a sub-routine). This could look like this:

```
[MDI_COMMAND_LIST]
MDI_COMMAND=G10 L20 P1 X0 Y0,Set XY\nOrigin
MDI_COMMAND=G0 X0 Y0,Go To\nOrigin
```

Where the first command is referring to the button ”SET origin” and the second to the button ”GOTO Origin”.
Note the comma and text after is optional - it will override the default button text.
These buttons are QtVCP action buttons and follow those rules.

#### 12.6.1.8 sim_panel

Small control panel to **simulate MPG jogging controls etc** for simulated configurations.
The MPG, selection buttons and control buttons export HAL pins to connect to linuxcnc.
The selection and control group boxes can be hidden if not needed by using the *-o hide=* option.
*groupBoxControl* and *groupBoxSelection* are the widget names that can be hidden.
If you want to hide both, use a comma between them with no spaces.
The *-a* option will make the panel always-on-top of all windows.

```
loadusr qtvcp sim_panel
```

Here we load the panel with no MPG selection buttons and the always-on-top option.

```
loadusr qtvcp -a -o hide=groupBoxSelection sim_panel
```

Abbildung 12.75: QtVCP `sim_panel` - Simulated Controls Panel For Screen Testing.

#### 12.6.1.9 `tool_dialog`

**Manueller Werkzeugwechsel-Dialog**, der eine Werkzeugbeschreibung enthält.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Optionen:

- -o notify_on` - _verwendet Desktop-Notify-Dialoge anstelle der QtVCP-eigenen Dialoge.
- `-o audio_on` - *play sound on tool change*
- `-o speak_on` - *speak announcement of tool change*

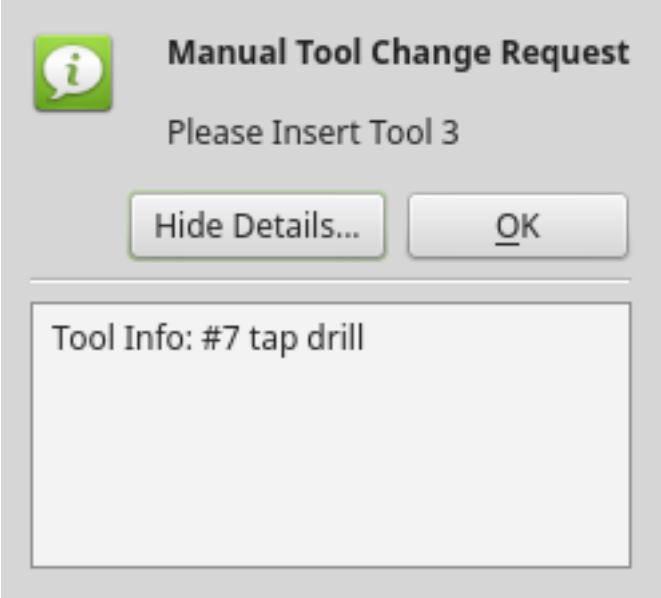



Abbildung 12.76: QtVCP `tool_dialog` - Manual Tool Change Dialog

### 12.6.2 `vismach` 3D Simulation Panels

Diese Tafeln sind vorgefertigte Simulationen gängiger Maschinentypen.
Diese können auch in andere Bildschirme wie AXIS oder GMOCCAPY eingebettet werden.

#### 12.6.2.1 QtVCP `vismach_mill_xyz`

3D OpenGL view of a *3-Axis milling machine*.

```
loadusr qtvcp vismach_mill_xyz
```

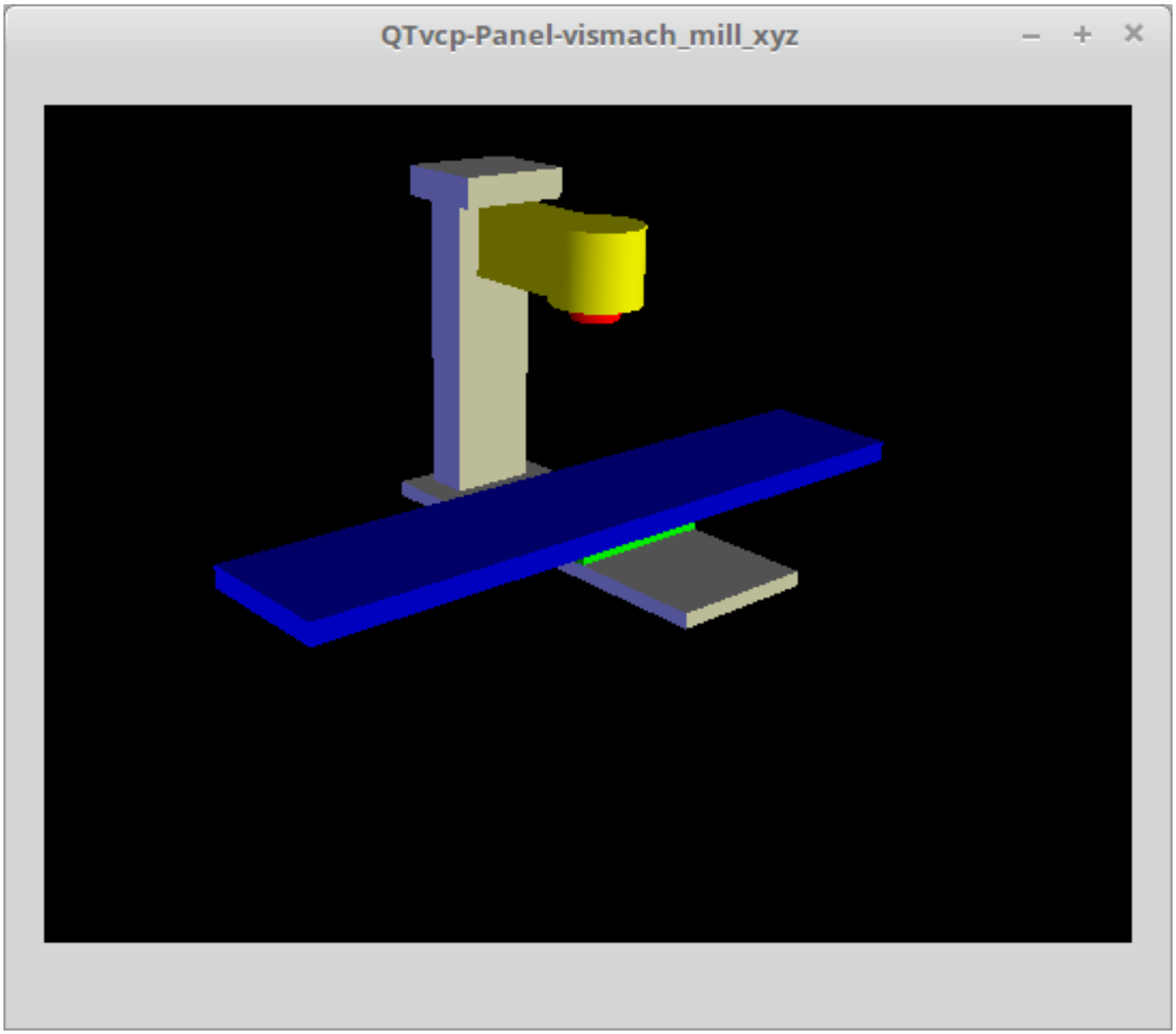



Abbildung 12.77: QtVCP `vismach_mill_xyz` - 3-Achsen-Fräse 3D-Ansichtspanel

#### 12.6.2.2 QtVCP `vismach_router_atc`

3D OpenGL view of a *3-Axis router style, gantry bed milling machine*.
This particular panel shows how to define and connect the model parts in the handler file, rather then importing the pre-built model from QtVCP's vismach library.

```
loadusr qtvcp vismach_router_atc
```

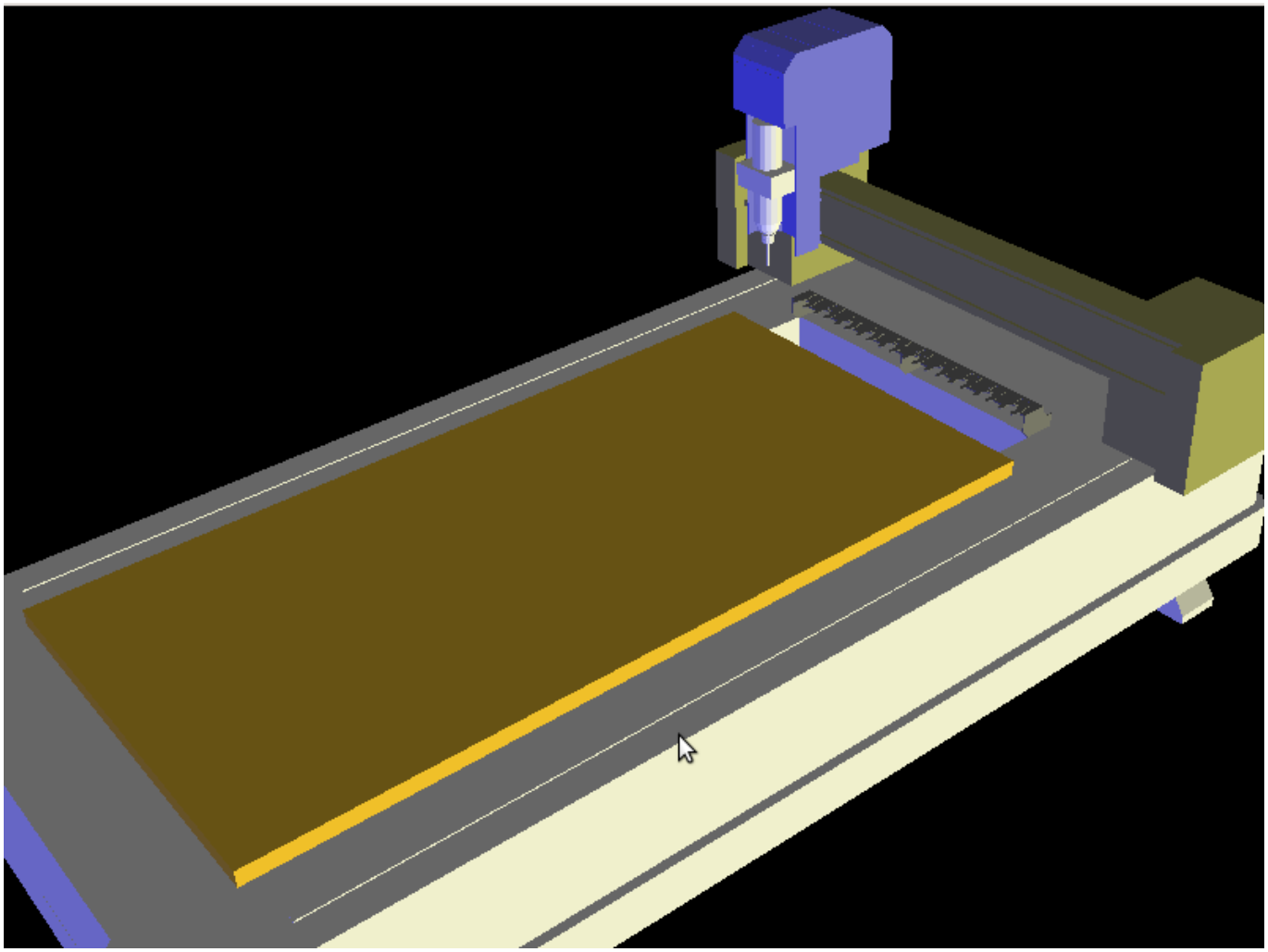


Abbildung 12.78: QtVCP `vismach_router_atc` - 3-Axis Gantry Bed Mill 3D View Panel

#### 12.6.2.3 QtVCP `vismach_scara`

3D-OpenGL-Ansicht einer *SCARA-basierten Fräsmaschine*.

```
loadusr qtvcp vismach_scara
```

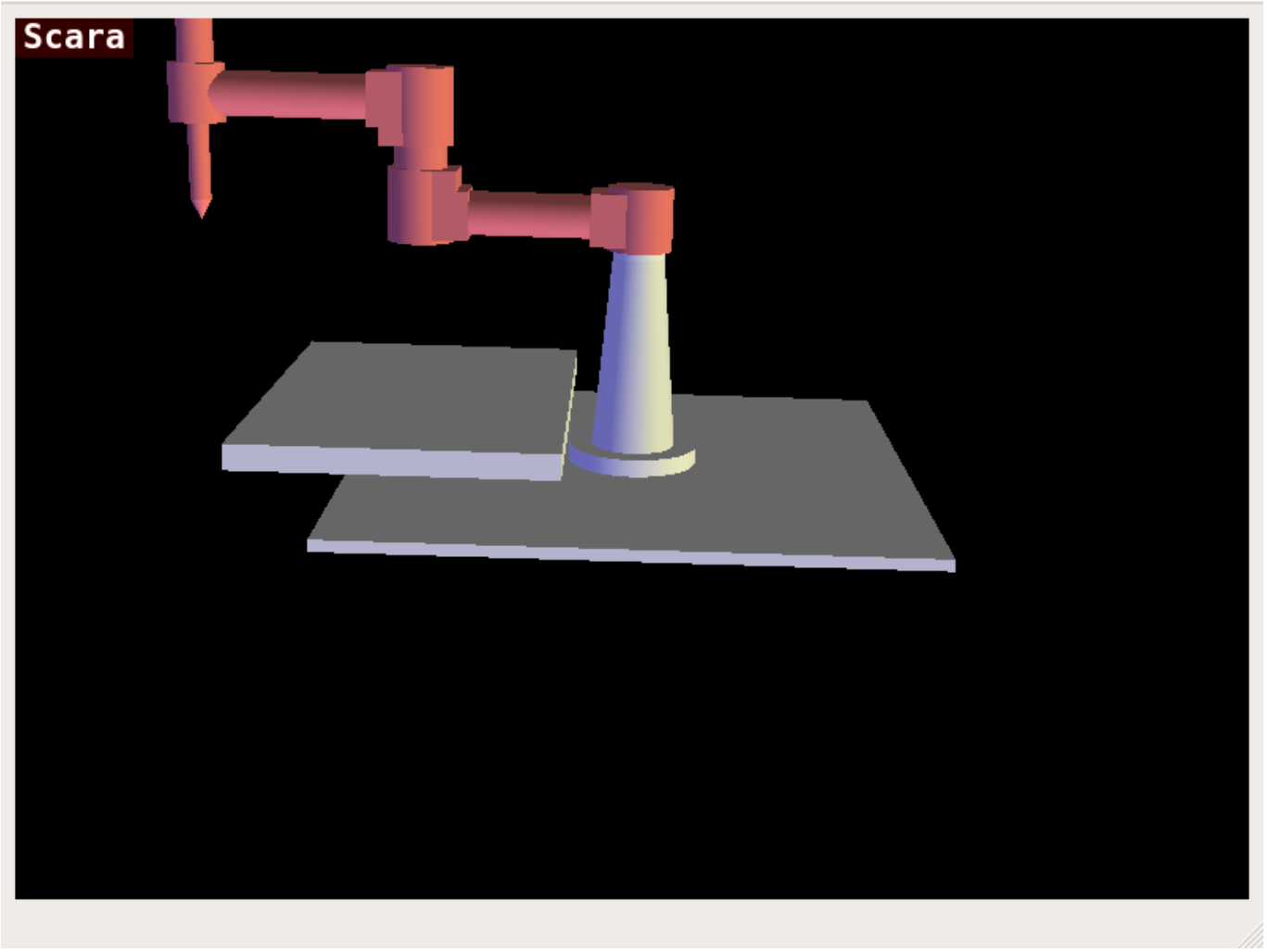



Abbildung 12.79: QtVCP `vismach_scara` - SCARA Fräse 3D-Ansichtsfenster

#### 12.6.2.4 QtVCP `vismach_millturn`

3D OpenGL view of a *3-Axis milling machine with an A axis/spindle*.

```
loadusr qtvcp vismach_millturn
```

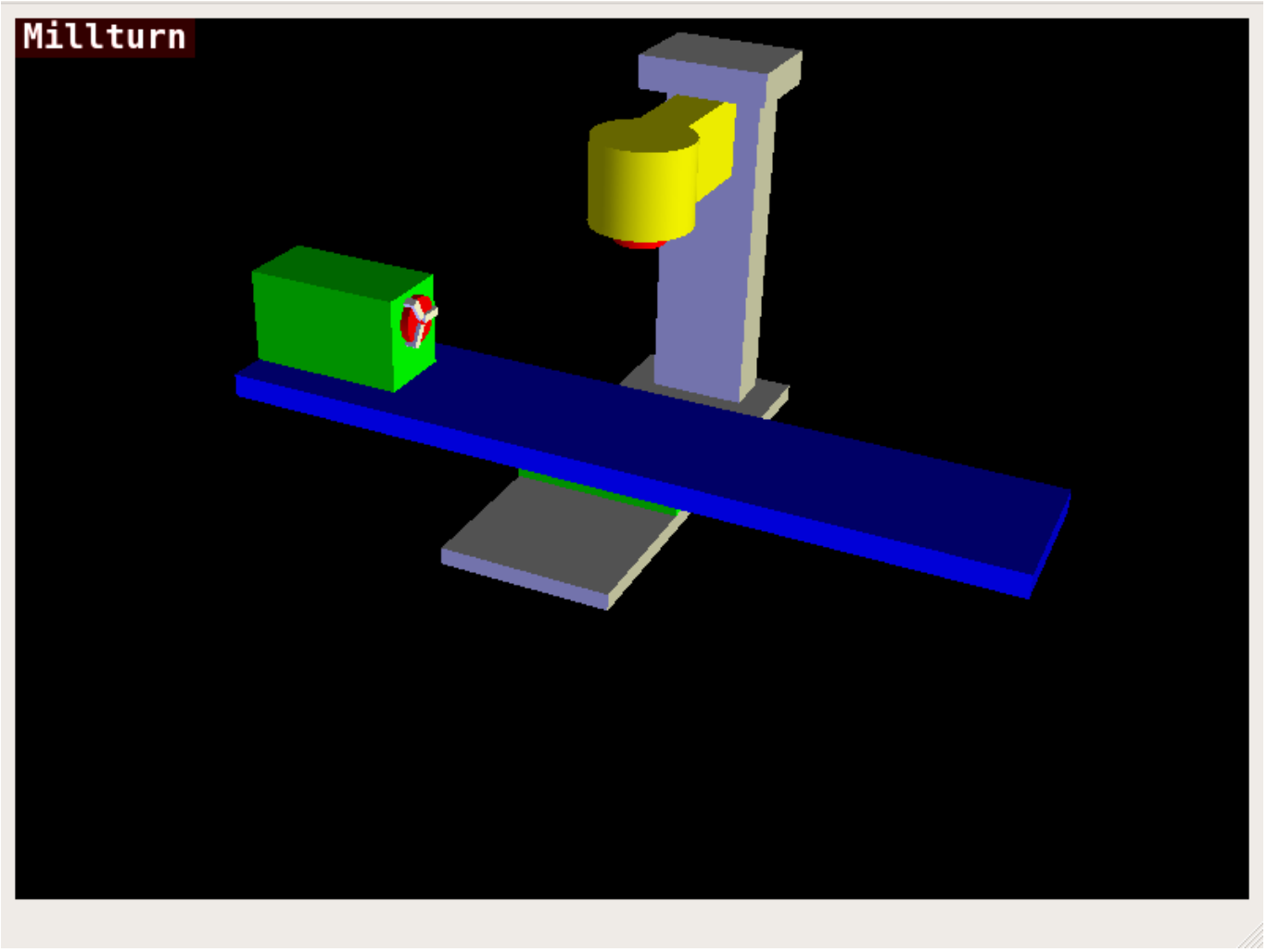



Abbildung 12.80: QtVCP `vismach_millturn` - 4 Axis MillTurn 3D View Panel

#### 12.6.2.5 QtVCP `vismach_mill_5axis_gantry`

3D OpenGL view of a *5-Axis gantry type milling machine*.

```
loadusr qtvcp vismach_mill_5axis_gantry
```

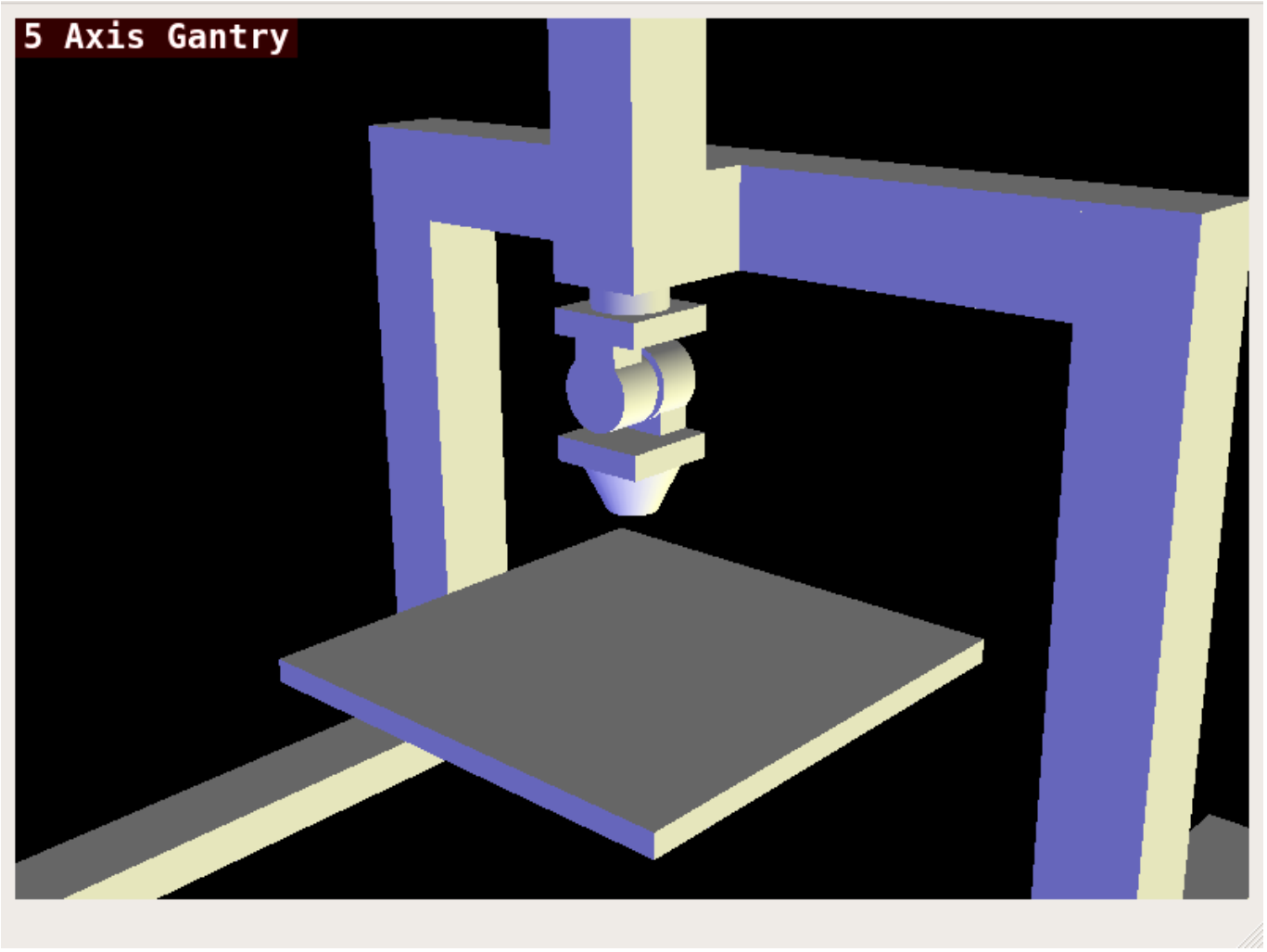



Abbildung 12.81: QtVCP `vismach_mill_5axis_gantry` - 5-AxIs Gantry Mill 3D View Panel

#### 12.6.2.6 QtVCP `vismach_fanuc_200f`

3D openGL view of a *6 joint robotic arm*.

```
loadusr qtvcp vismach_fanuc_200f
```

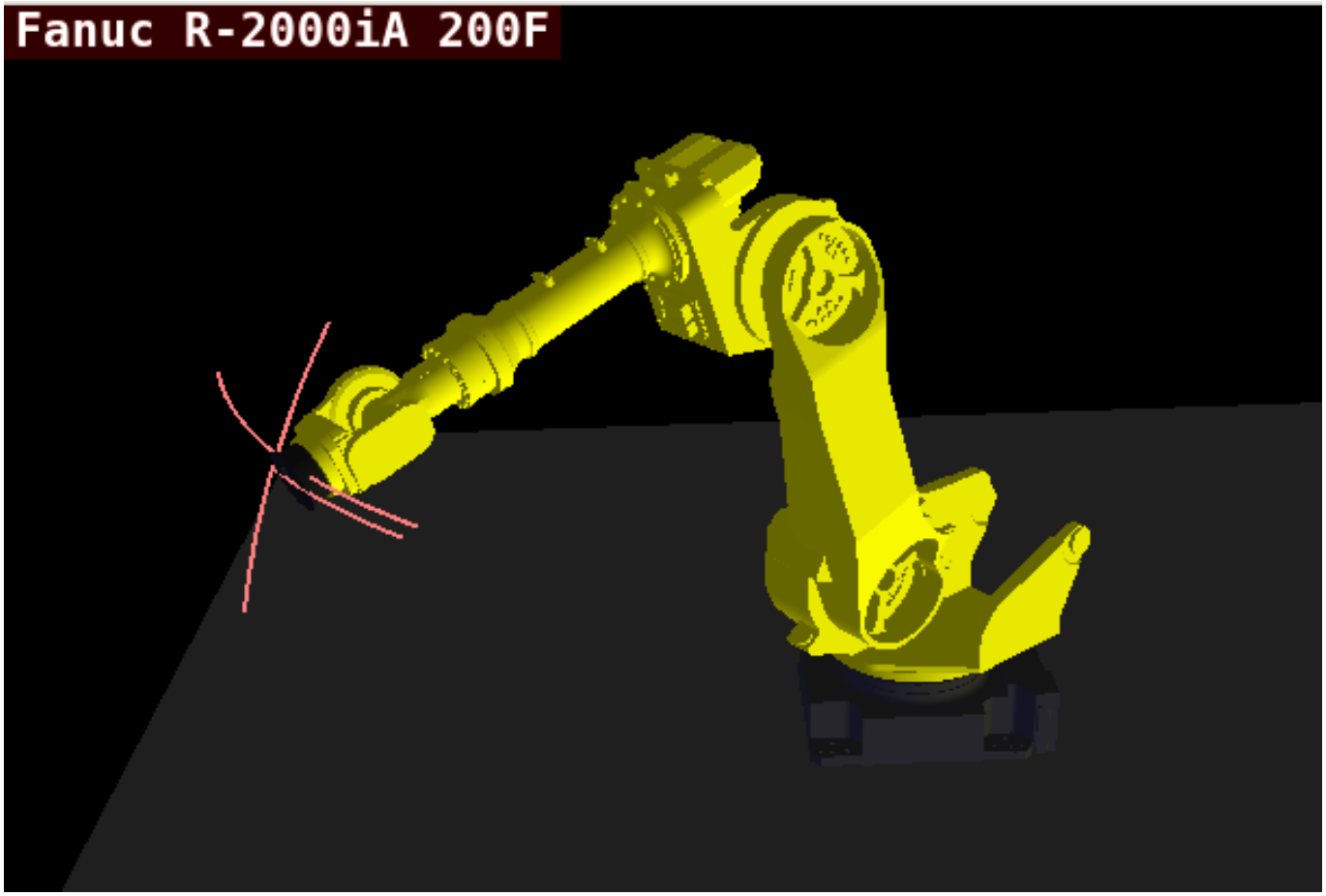



Abbildung 12.82: QtVCP `vismach_fanuc_200f` - 6 Joint Robotic Arm

### 12.6.3 Custom Virtual Control Panels

Sie können natürlich **Ihr eigenes Panel erstellen und laden**.

Wenn Sie eine UI-Datei mit dem Namen `my_panel.ui` und eine HAL-Datei mit dem Namen `my_panel.hal` erstellt haben, würden Sie diese dann von einem Terminal aus laden mit:

```
halrun -I -f my_panel.hal
```

**Beispiel einer HAL-Datei, die ein QtVCP-Panel lädt**

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime 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 diesem Fall laden wir `qtvcp` mit **`-Wn`**, das wartet, bis das Panel das Laden beendet hat, bevor es mit der Ausführung des nächsten HAL-Befehls fortfährt.
Damit soll *gewährleistet werden, dass die vom Panel erstellten HAL-Pins tatsächlich fertig sind*, falls sie im Rest der Datei verwendet werden.

## 12.6.4 Embedding QtVCP Virtual Control Panels into QtVCP Screens

QtVCP panels can be embedded into most QtVCP screens and avoids problems such as focus transferring that can be a problem in non-native embedding.

### 12.6.4.1 Embedding Commands

A typical screen such as QtDragon will search the INI file under the heading [DISPLAY] for commands to embed a panel.

```
[DISPLAY]
EMBED_TAB_NAME=Embedding demo
EMBED_TAB_COMMAND=qtvcp simple_hal
EMBED_TAB_LOCATION=tabWidget_utilities
```

***EMBED_TAB_NAME***
: will typically be the title of the tab.

***EMBED_TAB_LOCATION***
: will be specific to the screen and specifies the tabWidget or stackedWidget to embed into.

***EMBED_TAB_COMMAND***
: is the command used to invoke loading of the panel. For native embedded panels the first word will always be *qtvcp*, the last will be the panel name to load. You can also pass options to the panel with -o switches in the command line between *qtvcp* and the panel name. The panel will follow the debugging mode setting of the main screen.

### 12.6.4.2 Location of builtin Panels

There are panels available that are included with LinuxCNC. To see a list open a terminal and type *qtvcp* and press return.
You will get a help printout and a list of builtin screen and panels.
Pick any of the names from the panel list and add that to the COMMAND entry after *qtvcp*.
The builtin panel search path is *share/qtvcp/panels/PANELNAME*.
Run-In-Place and installed versions of LinuxCNC have these in different locations on the system.

#### 12.6.4.3 Location of Custom Panels

Custom panels can be embedded too -either a modified builtin panel or a new user-built one.
When loading panels, QtVCP looks in the configuration folders path for *qtvcp/panels/PANELNAME/-PANELNAME.ui*.
*PANNELNAME* being any valid string with no spaces. If no path is found there, then looks in the builtin file path.
QtVCP will do the same process for the optional handler file: *qtvcp/panels/PANELNAME/PANELNAME_handler.py*

#### 12.6.4.4 Handler Programming Tips

In a screen handler file, the reference used for the window is *self.w*.
In QtVCP panels, that reference will refers to the panel's window.
To reference the main window use *self.w.MAIN* If your panel is to be able to run independently and embedded, you must trap errors from referencing objects not available. (Note, main screen objects are not available in an independent panel.)

E.g., this would use the panel's preference file if there is one.

```
try:
    belt_en = self.w.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

This would use the main screen preference file if there is one.

```
try:
    belt_en = self.w.MAIN.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

#### 12.6.4.5 Designer Widget Tips

When using Python command option in Action Button widgets of an embedded panel:

**`INSTANCE`**
: refers to the panel window. E.g., `INSTANCE.my_panel_handler_function_call(True)`

***`MAIN_INSTANCE`***
: refers to the main screen window. E.g., `MAIN_INSTANCE.my_main_screen_handler_function_call(Tru`

If the panel is not embedded, both refer to the panel window.

#### 12.6.4.6 Handler Patching - Subclassing Builtin Panels

We can have QtVCP load a subclassed version of the standard handler file. in that file we can manipulate the original functions or add new ones.
Subclassing just means our handler file first loads the original handler file and adds our new code on top of it - so a patch of changes.
This is useful for changing/adding behaviour while still retaining standard handler updates from LinuxCNC repositories.

You may still need to use the handler copy dialog to copy the original handler file to decide how to patch it.

There should be a folder in the config folder; for panel: named *<CONFIG FOLDER>/qtvcp/panels/<PANEL NAME>/*
add the handle patch file there, named like so <ORIGINAL PANEL NAME>_handler.py
ie for cam_align the file would be called *cam_align_handler.py*

Here is a sample to change the circle color in cam_align:

```
import sys
import os
import importlib
from PyQt5.QtCore import Qt
from qtvcp.core import Path

PATH = Path()

# get reference to original handler file so we can subclass it
sys.path.insert(0, PATH.PANELDIR)
panel = os.path.splitext(os.path.basename(os.path.basename(__file__)))[0]
base = panel.replace('_handler','')
module = "{}.{}".format(base,panel)
mod = importlib.import_module(module, PATH.PANELDIR)
sys.path.remove(PATH.PANELDIR)
HandlerClass = mod.HandlerClass

# return our subclassed handler object to Qtvcp
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# subclassed from HandlerClass which was imported above
class UserHandlerClass(HandlerClass):
    print('Custom subclassed panel handler loaded\n')

    def initialized__(self):
        # call original handler initialized function
        super().initialized__()

        # add our customization
        self.w.camview.circle_color = Qt.green
```

## 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.

There are also **special widgets** made for LinuxCNC that make integration easier. These are split in two, heading on the right side of the editor:

- 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

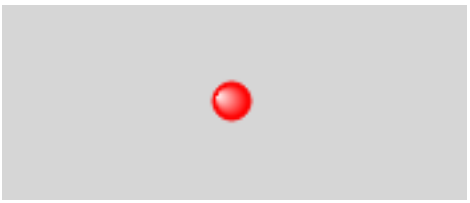


Abbildung 12.83: QtVCP `LED`: LED-Anzeige-Widget

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**.

Er basiert auf dem *QCheckButton* von PyQt.

#### 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

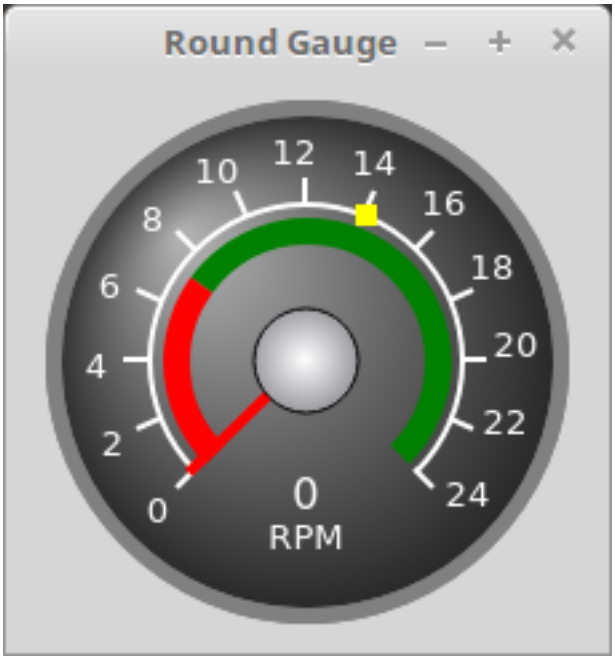



Abbildung 12.84: QtVCP `Gauge`: Round Dial Gauge Widget

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`.

If this option is not set, then `value` and `setpoint` must be connected programmatically, i.e., in the handler file.

**`max_reading`**
This value determines the *highest number displayed* on the gauge face.

**`max_value`**
Dies ist der *maximal zu erwartende Wert des Werteingangssignals*.
Mit anderen Worten, es ist der Skalenendwert.

**`num_ticks`**
Dies ist die *Anzahl der Ticks/Anzeigewerte* auf der Anzeigefläche.
Sie sollte auf eine Zahl eingestellt werden, die sicherstellt, dass die Textanzeigen auf der Anzeigefläche lesbar sind.
Der minimal zulässige Wert ist 2.

**zone1_color**
Zone1 erstreckt sich *vom maximalen Messwert bis zum Schwellenwert*.
Sie kann auf eine beliebige RGB-Farbe eingestellt werden.

**zone2_color**
Zone2 erstreckt sich *vom Schwellenwert bis zum Mindestwert*, der 0 ist.
Sie kann auf eine beliebige RGB-Farbe eingestellt werden.

**bezel_color**
This is the color of the *outer ring of the gauge*.

**threshold**
Der Schwellenwert ist der *Übergangspunkt zwischen den Zonen*.
Er sollte auf einen Wert zwischen 0 und dem Maximalwert gesetzt werden.
Der höchstzulässige Wert wird auf den ”Maximalwert” des Messgeräts gesetzt, der Mindestwert ist 0.

**gauge_label**
Dies ist der *Text unter der Wertanzeige*, nahe dem unteren Rand des Messgeräts.
Die Funktion des Messgeräts ist dann leicht erkennbar.

**Non Customizable Parameters** There are 2 inputs that are not customizable. They can be set via HAL pins, programmatically or via signals from other widgets:

**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 *HalBar* - HAL Bar Level Indicator

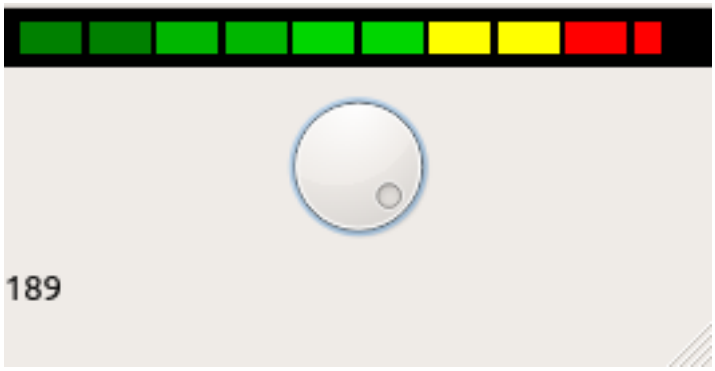



Abbildung 12.85: QtVCP `HalBar`: Panel demonstrating the HAL Bar Level Indicator

This widget is used to indicate level or value, usually of a HAL s32/float pin.
you can also disable the HAL pin and use Qt signals or python commands to change the level.

HalBar is a subclass of the Bar widget, so it inherits these properties

- *stepColorList*: a list of color strings, the number of colors defines the number of bars.
- *backgroundColor*: a QColor definition of the background color.
- *setMaximum*: an integer that defines the maximum level of indication.
- *setMinimum*: an integer that defines the lowest level of indication.
- *pinType*: to select **HAL pins type**:
  - `NONE` no HAL pin will be added
  - `S32` A S32 integer pin will be added
  - `FLOAT` A Float pin will be added
- *pinName*: to change the **HAL pin name** otherwise the widget base name is used.

The above Bar properties could be set in *styles sheets*.
pinType and pinName properties can not be changed in stylesheets.

**Anmerkung**
In style sheets, stepColorList is a single string of color names separated by commas.

```
HalBar{
    qproperty-backgroundColor: #000;
    qproperty-stepColorList: 'green,green,#00b600,#00b600,#00d600,#00d600,yellow,yellow,red ↩
        ,red';
}
```

#### 12.7.1.8 `HALPad` - HAL Buttons Joypad

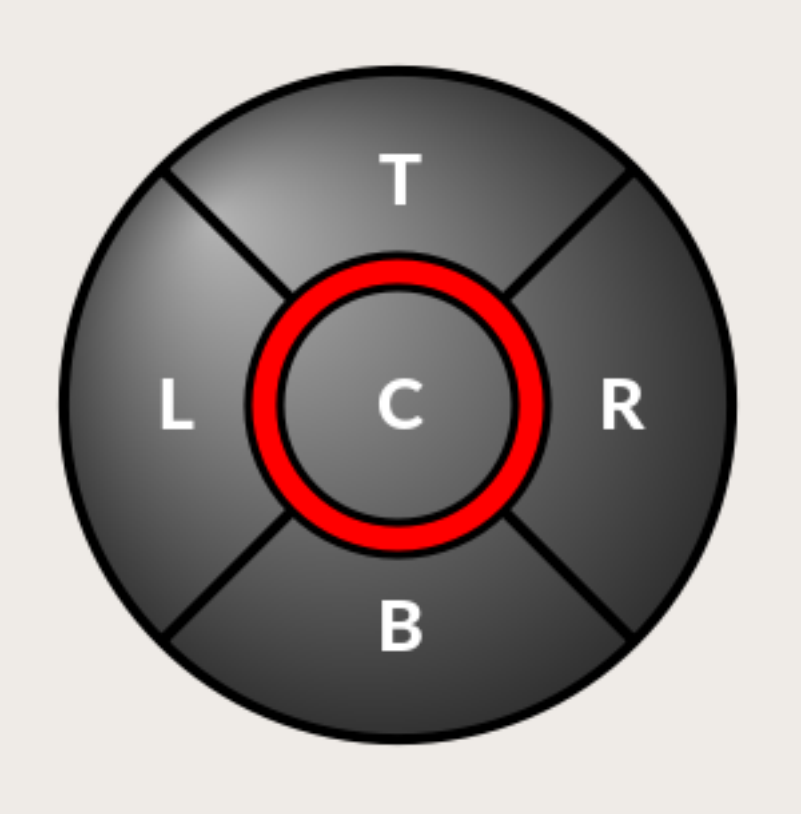



Abbildung 12.86: QtVCP `HALPad`: HAL Buttons Joypad

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.
The LED center ring has selectable colors for off and on and is controlled by a bit HAL pin.
**HALPad ENUMS** Es werden *numerierte Konstanten* verwendet:

- Um **Indikatorpositionen** zu referenzieren:
  - `NONE`
  - `LEFT`
  - `RIGHT`
  - `CENTER`
  - `TOP`
  - `BOTTOM`
  - `LEFTRIGHT`
  - `TOPBOTTOM`
- Für **HAL-Pins Typ**:
  - `NONE`
  - `BIT`

- S32
- FLOAT

Sie verwenden den Namen des Widgets im Qt Designer plus die Referenzkonstante:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad Properties

**pin_name**
Optional name to use for the *HAL pins basename*. If left blank, the Qt Designer widget name will be used.

**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**
Farbauswahl für den mittleren LED-Ring, der angezeigt werden soll, wenn der BASENAME.light.center HAL-Pin True oder False ist.

**text_color**
Auswahl der Farbe für den Text des Buttons.

**text_font**
Auswahl der Schriftart für den Text des Buttons.

**HALPad Styles** The above properties could be set in *styles sheets*.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

#### 12.7.1.9 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*.

#### 12.7.1.10 `focusOverlay` - Focus Overlay Widget

Dieses Widget legt ein **farbiges Overlay über den Bildschirm**, normalerweise während ein Dialog angezeigt wird.

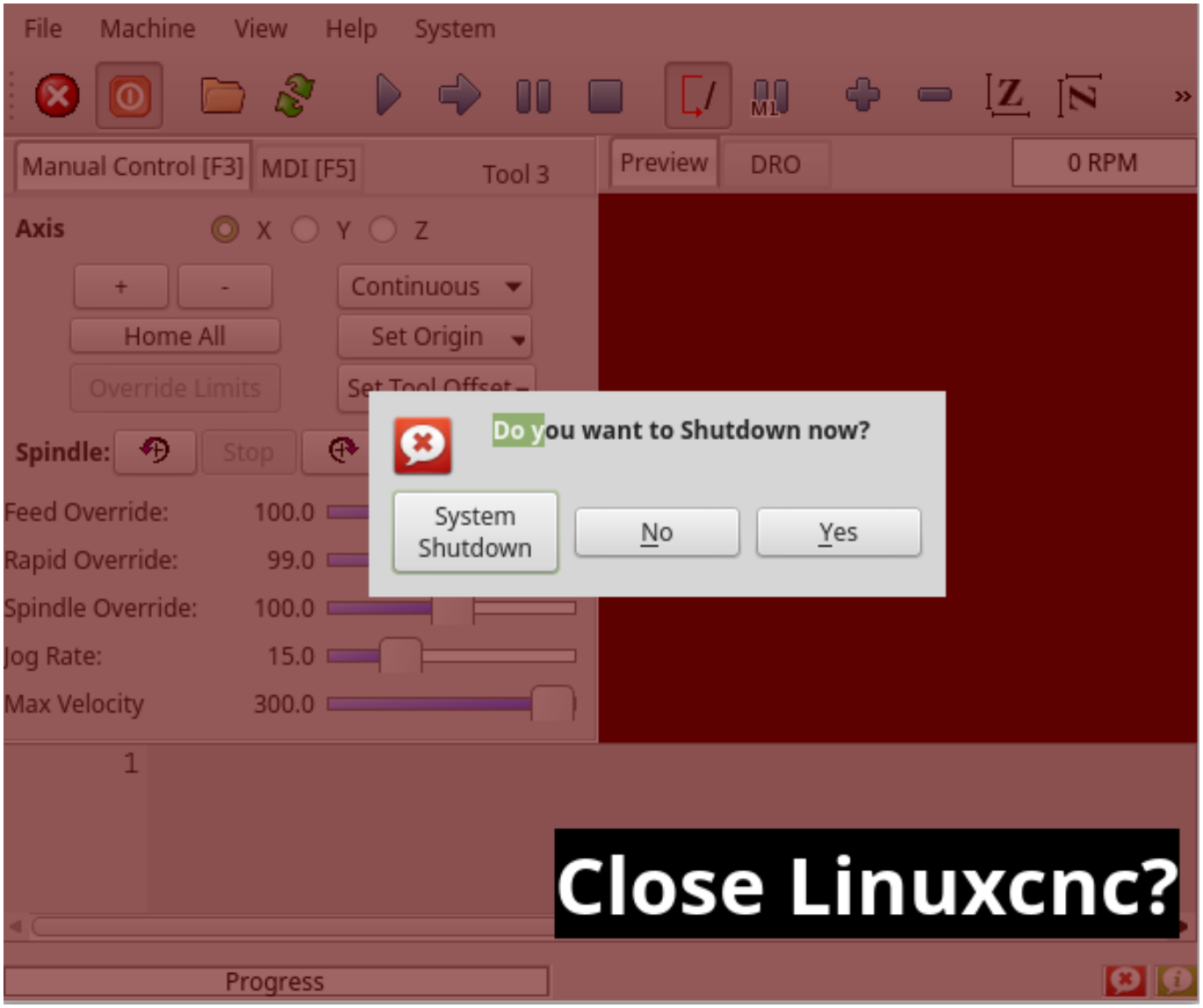



Abbildung 12.87: Beispiel für ein Fokus-Overlay zur Bestätigung der Abschlussaufforderung

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*.

#### 12.7.1.11 `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`.

#### 12.7.1.12 `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.

Es basiert auf PyQts *QLabel*.

#### 12.7.1.13 `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`**
: Optionsstring, der als HAL-Pin-Name verwendet werden soll.
Bei einem leeren String wird der Name des Widgets verwendet.

**`bit_pin_type`**
: Selects the input pin as type 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, e.g., `{:.2f}` will display a float rounded to 2 numbers after the decimal.
A blank setting will allow the decimal to move as required.

Es basiert auf PyQts *QLCDNumber*.

#### 12.7.1.14 `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 an integer, or a 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*.

#### 12.7.1.15 `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*.

#### 12.7.1.16 `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*.

#### 12.7.1.17 `WidgetSwitcher` - Multi-widget Layout View Switcher Widget

This is used to switch the view of a multi-widget layout to show just one widget, i.e. 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 its 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(`**

Dies zeigt das ”Seite 0”-Layout und stellt alle anderen Widgets wieder so ein, wie sie ursprünglich in Qt Designer erstellt wurden.

**`[_WidgetSwitcherName_].show_next()`**
Nächstes Widget anzeigen.

Es basiert auf dem *QStack*-Widget.

### 12.7.2 Widgets für Maschinensteuerungen

Diese Widgets **interagieren mit dem Zustand der Maschinensteuerung**.

#### 12.7.2.1 `ActionButton` - Aktionssteuerungs-Widget der Maschinensteuerung

Diese Tasten werden für **Steuerungsaktionen an der Maschinensteuerung** verwendet.

Sie sind auf `IndicatedPushButton` aufgebaut und können daher mit LEDs überlagert werden.

---

**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.

---

**Aktionen** 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**
Nullt die aktuellen Offsets des Benutzerkoordinatensystems.

**zero G92**
Nullt die optionalen G92-Offsets.

**zero Z rotational**
Zeros the rotation offset.

**jog joint positive**
Legt die Gelenknummer fest.

**jog joint negative**
Legt die Gelenknummer fest.

**jog selected positive**
Ausgewählt mit einem anderen Widget oder STATUS.

**jog selected negative**
Ausgewählt mit einem anderen Widget oder STATUS.

**jog increment**
Set metric/imperial/angular numbers.

**jog rate**
Festlegen Sie die float/alt-Gleitkommanummer.

**feed override**
Festlegen Sie die float/alt-Gleitkommanummer.

**rapid override**
Festlegen Sie die float/alt-Gleitkommanummer.

**spindle override**
Festlegen Sie die float/alt-Gleitkommanummer.

**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`, i.e.,calls a hard coded MDI command

**INI mdi number**
Set `ini_mdi_number`, i.e., 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** Diese setzen *Attribute* der ausgewählten Aktion (Verfügbarkeit hängt vom Widget ab):

**toggle float option**
Allows jog rate and overrides to toggle between two rates.

**joint number**
Wählt das Gelenk/die Achse aus, das/die von der Schaltfläche gesteuert wird.

**incr imperial number**
Legt das imperiale Jog-Inkrement fest (negativ setzen, um zu ignorieren).

**incr mm number**
Legt die metrische Schrittweite fest (zum Ignorieren negativ setzen).

**incr angular number**
Legt die Winkelschrittweite fest (zum Ignorieren negativ einstellen).

**float number**
Wird für `jograte` und overrides verwendet.

**float alternate number**
Für `jograte` und overrides, die zwischen zwei Fließkommazahlen wechseln können.

**view type string**
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
```

Action buttons are subclassed from Abschnitt 12.7.5.1`[IndicatedPushButton]`. See the following sections for more information about:

- 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`.

Es basiert auf PyQts *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.

Die aufgezeichnete Position wird beim Herunterfahren gespeichert, wenn eine Einstellungsdateioption eingerichtet ist.

---

**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 its 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 must have selected an entry dialog that corresponds to the dialog_code_string, usually this is selected from the screenOptions widget.

You can select the property *halpin_option*, it will then set a HAL pin true when the axis is selected. The property *joint_number* should be set to the appropriate joint number. The property *axis_letter* should be set to the appropriate axis letter.

The property *dialog_code_string* can be changed to *ENTRY* or *CALCULATOR* to call a typing only entry dialog or a touch/typing calculator type entry dialog.

Es basiert auf PyQts *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`**
: Gelenknummer des anzuzeigenden Offsets (10 gibt den Rotationsoffset an).

**`Qreference_type`**
: Tatsächlich, relativ oder noch zu fahrende Entfernung (0,1,2).

**`metric_template`**
: Format of display, e.g. `%10.3f`.

**`imperial_template`**
: format of display, e.g. `%9.4f`.

**`angular_template`**
: Format of display, e.g. `%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 its objectName in Qt Designer:

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

Es basiert auf PyQts *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.
- **Einträge in Präferenzdateien**, wenn Sie PREFERENCE in Großbuchstaben in das Widget MDILine eingeben.

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.

The GcodeDisplay properties can be set in a stylesheet with the following code added to the .qss file (the following color choices are random).

```
EditorBase{
    qproperty-styleColorBackground: lightblue;
    qproperty-styleColor0: black;
    qproperty-styleColor1: #000000; /* black */
    qproperty-styleColor2: blue;
    qproperty-styleColor3: red;
    qproperty-styleColor4: green;
    qproperty-styleColor5: darkgreen;
    qproperty-styleColor6: darkred;
    qproperty-styleColor7: deeppink;
    qproperty-styleColorMarginText: White;
    qproperty-styleColorMarginBackground: blue;
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont1: ”Times,18,-1,0,90,1,0,0,0,0”;
    qproperty-styleFont2: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont3: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont4: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont5: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont6: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont7: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFontMargin: ”Times,14,-1,0,90,0,0,0,0,0”;
}
```

Für den *Standard-G-Code-Lexer* des Widgets `GcodeDisplay`:

- **styleColor0 = Default**: Everything not part of the groups below
- **styleColor1 = LineNo and Comments**: Nxxx and comments (characters inside of and including *()* or anything after *;* (when used outside of parenthesis) with the exception of the note below)
- **styleColor2 = G-code**: G and the digits after
- **styleColor3 = M-code**: M and the digits after
- **styleColor4 = Axis**: XYZABCUVW
- **styleColor5 = Other**: EFHIJKDQLRPST (feed, rpm, radius, etc.)
- **styleColor6 = AxisValue**: Values following XYZABCUVW
- **styleColor7 = OtherValue**: Values following EFHIJKDQLRPST$

---

**Anmerkung**
For comments, the ”OtherValue” color (Color 5) can be used to highlight ”print,” ”debug,” ”msg,” ”logopen,” ”logappend,” ”logclose” ”log,” ”pyrun,” ”pyreload” ”abort,” ”probeopen” ”probeclose” inside of a parenthesis comment in a line of G-code. As well as ”py,” if a line that starts with ”;py,”. Examples: (print, text), (log, text), (msg, text), or (debug, text). Only the last of the examples will be highlighted if there are more than one on the same line.

---

*Schriftdefinitionen*:

```
”style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0”
```

Es basiert auf PyQts *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

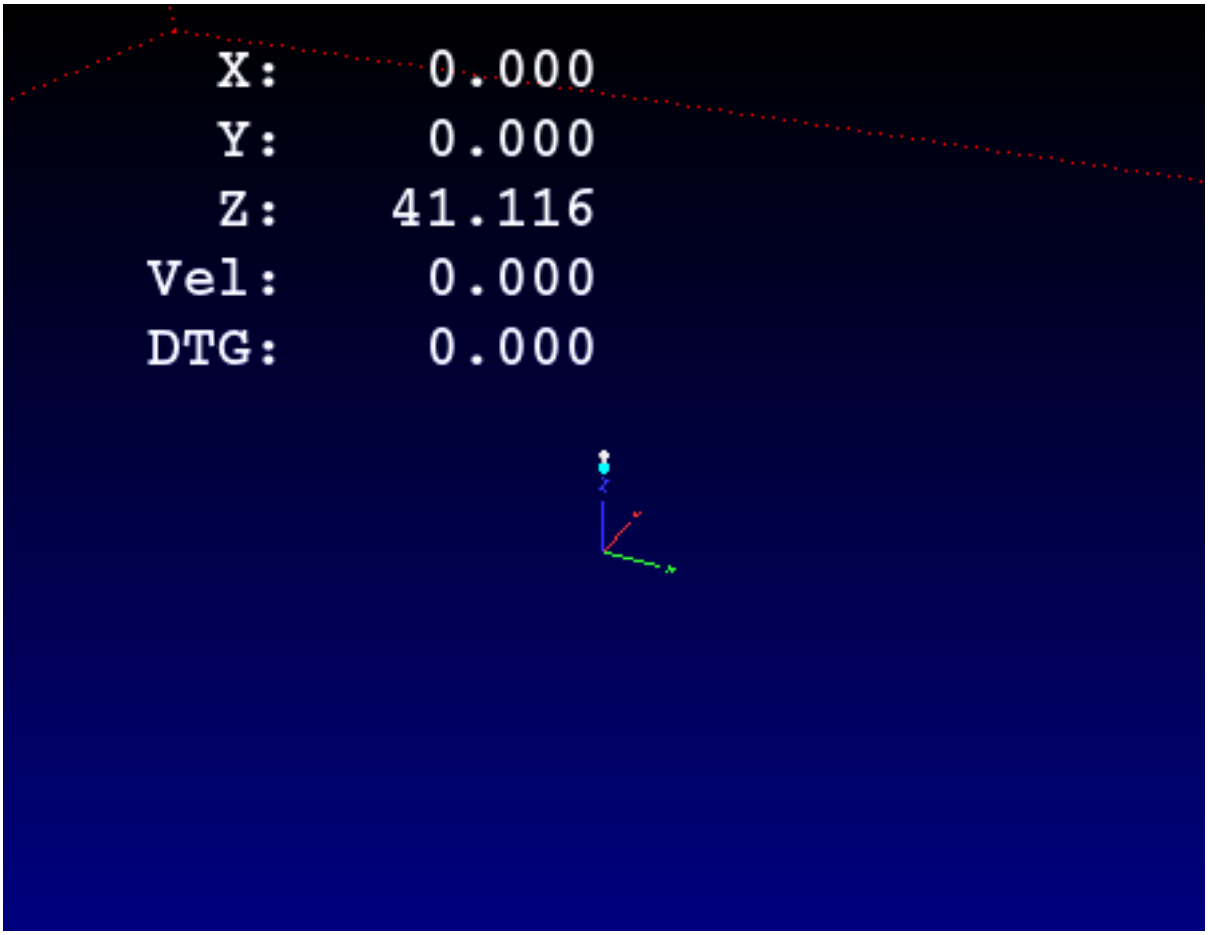



Abbildung 12.88: QtVCP GcodeGraphics: G-code Graphic Backplot Widget

This **displays the current G-code in a graphical form**.

Stylesheets Properties

**`dro-font/dro-large-font` *(string)***
Sets the small and large DRO font properties
Here we reference with the widget base name; GCodeGraphics

```
GCodeGraphics{
    qproperty-dro_font:”monospace  bold 12”;
}
GCodeGraphics{
    qproperty-dro_large_font:”Times 25”;
}
```

**_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 (referenced using the widget user selected name):

```
#gcodegraphics{
    qproperty-_view: z;
}
```

**_dro *(bool)***
Legt fest, ob *die DRO* angezeigt werden soll oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

**_dtg *(bool)***
Legen Sie fest, ob die *Reststrecke angezeigt* werden soll.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

**_metric *(bool)***
Legt fest, ob die_ Einheiten standardmäßig_ in *metrischen* Einheiten angezeigt werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

**_overlay *(bool)***
Legt fest, ob das Overlay standardmäßig angezeigt werden soll oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

**_offsets *(bool)***
Legt fest, ob die *Offsets standardmäßig angezeigt* werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

**_small_origin *(bool)***
Legt fest, ob *standardmäßig der kleine Ursprung angezeigt wird*.
Das folgende Beispiel zeigt, wie diese Eigenschaft festgelegt wird:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

**`overlay_color` *(Primär-, Sekundär- oder RGBA-formatierte Farbe)***
Legt die *Standard-Overlayfarbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

**`background_color` Hintergrundfarbe, *(Primär-, Sekundär- oder RGBA-formatierte Farbe)***

Legt die *Standard-Hintergrundfarbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

**`+_use_gradient_background+` *(bool)***
Legt fest, ob *Standardmäßig ein Hintergrund mit Farbverlauf verwendet wird*.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
}
```

**`jog_color` *(primary, secondary, or RGBA formatted color)***
Legt die *Standard-Jog-Farbe* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

**`Feed_color` *(primary, secondary, or RGBA formatted color)***
Legt die *Standard-Farbe für den Vorschub* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

**`Rapid_color` *(primary, secondary, or RGBA formatted color)***
Legt die *Standard-Farbe für den Eilgang* fest.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

**`InhibitControls` *(bool)***
Legt fest, ob *externe Steuerelemente standardmäßig gesperrt* werden sollen oder nicht.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

**`MouseButtonMode` *(int)***
Ändert das Verhalten der *Maustaste* zum Drehen, Verschieben oder Zoomen innerhalb der Vorschau.
Im Folgenden wird ein Beispiel für die Einstellung dieser Eigenschaft gezeigt:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

There are 12 valid modes:

| **Mode** | **Move** | **Zoom** | **Rotate** (engl. drehen) |
|---|---|---|---|
| 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 whether 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
- *set-large-dro*
- *set-small-dro*

**ACTION.ADJUST_PAN(_X,Y_)**
Legen Sie direkt den relativen Blickwinkel in x- und y-Richtung fest.

**ACTION.ADJUST_ROTATE(_X,Y_)**
Legen Sie direkt die relative Drehung der Ansicht in x- und y-Richtung fest.

It is based on PyQt's *OpenGL* widget.

#### 12.7.2.10 *StateLabel* - Controller-Modi Statusbeschriftungsanzeige-Widget

Dadurch wird eine **Beschriftung basierend auf den Zuständen der Maschinensteuerung true/false** angezeigt.

Sie können zwischen verschiedenen Texten wählen, die auf wahr oder falsch basieren.

**Eigenschaften der Zustandsauswahl** Die Zustände sind über diese Eigenschaften wählbar:

**css_mode_status**
*True*, wenn sich die Maschine in *G96* befindet *Constant Surface Speed Mode*.

**diameter_mode_status**
*True*, wenn sich die Maschine in *G7* befindet *Drehmaschine Durchmesser Modus*.

**fpr_mode_status**
: `True`, wenn die Maschine im `G95` *Vorschub je Umdrehung Modus* (engl. Feed per Revolution Mode) ist.

**metric_mode_status**
: `True`, wenn sich die Maschine im `G21` *Metrischen Modus* befindet.

Text templates properties

**true_textTemplate**
: Dies ist der Text, der gesetzt wird, wenn die Option `True` ist.
Sie können *Qt rich text* code für verschiedene Schriftarten/Farben usw. verwenden.
Eine typische Vorlage für den metrischen Modus im wahren Zustand könnte sein: *Metrischer Modus*

**false_textTemplate**
: Dies ist der Text, der gesetzt wird, wenn die Option `False` ist.
Sie können *Qt rich text* code für verschiedene Schriftarten/Farben usw. verwenden.
Eine typische Vorlage für den metrischen Modus im falschen Zustand könnte sein: *Imperialer Modus*.

Es basiert auf PyQts *QLabel*.

#### 12.7.2.11 `StatusLabel` - Anzeige-Widget für Controller-Variablen-Zustandsbeschriftung

This will display a label based on selectable status of the machine controller.
You can change how the status will be displayed by substituting python formatting code in the text template. You can also use rich text for different fonts/colors etc.

**Selectable States** These states are selectable:

**actual_spindle_speed_status**
: Dient zur Anzeige der tatsächlichen Spindeldrehzahl, wie sie vom HAL-Pin ”spindle.0.speed-i” gemeldet wird.
Sie wird in *RPM* umgewandelt.
Normalerweise wird ein ”textTemplate” von ”%d” verwendet.

**actual_surface_speed_status**
: 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**
: Zeigt den Namen der zuletzt geladenen Datei an.

**filepath_status**
Shows the last loaded full file path name.

**gcode_status**
Shows all active G-codes.

**gcode_selected_status**
Zeigt die aktuell ausgewählte G-Code-Zeile an.

**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 machine_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*.

***motion_type_status***
Shows current type of machine motion using the text described from the motion_type_list.

- *None*
- *Rapid*
- *Feed*
- *Arc*
- *Tool Change*
- *Probe*
- *Rotary Index*

**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**
Zeigt die aktuelle Spindel-Override-Einstellung in Prozent an.

**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**
Gibt den Kommentartext des aktuell geladenen Werkzeugs zurück.

**tool diameter_status**
Gibt den Durchmesser des aktuell geladenen Werkzeugs zurück.

**tool_number_status**
Gibt die Werkzeugnummer des aktuell geladenen Werkzeugs zurück.

**tool_offset_status**
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 to describe different machine states.

***motion_label_list***
List of labels used to describe different motion types.

**halpin_names**
Name of a halpin to monitor (must be the complete name, including the 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`.

Es basiert auf PyQts *QLabel*.

#### 12.7.2.12 `StatusImageSwitcher` - Controller Status Image Switcher

Status image switcher will **switch between images based on LinuxCNC states**.

***watch_spindle**
Wechselt zwischen *3 Bildern*: `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**
Würde zwischen *2 Bildern oder einem pro Gelenk* umschalten.

Here is an example of using it to display an icon of Z axis homing state:

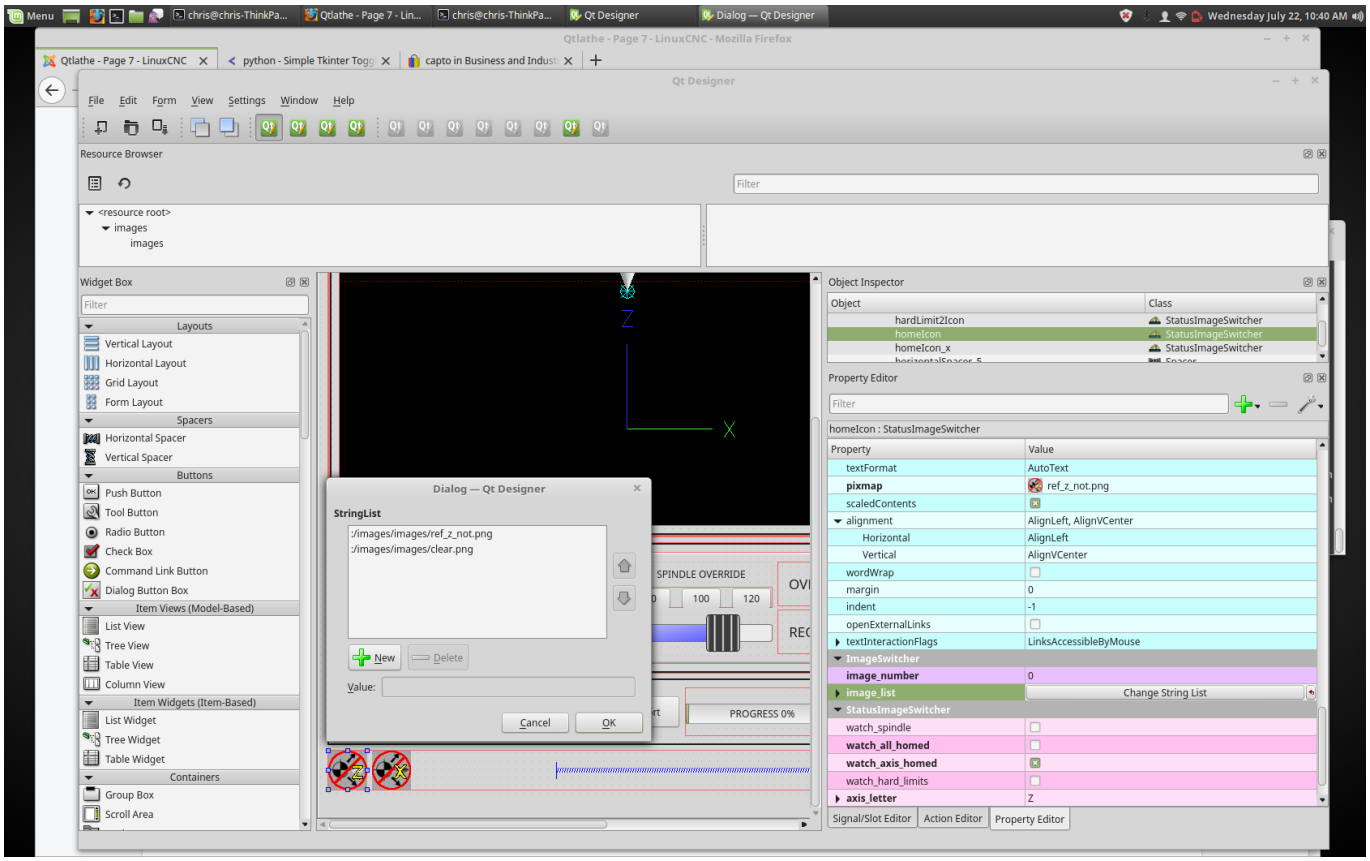



Abbildung 12.89: QtVCP `StatusImageSwitcher`: Controller Status Image Switcher

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.*

Sie können die *Anzeige der Bilder* in der Bilderliste *testen*, indem Sie die `image number` (engl. für Bildnummer) ändern. In diesem Fall ist `0` unhomed und `1` würde homed sein.
Dies ist nur für die Testanzeige und wird außerhalb von Qt Designer ignoriert.

#### 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.

Es basiert auf PyQts *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 property 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**
Farbe der transparenten Ebene, die beim Verlassen angezeigt wird.

**catch_error_option**
*Überwachung des LinuxCNC-Fehlerkanals.*
Dies sendet auch die Nachricht über `STATUS` an alles, das sich registriert.

**play_sounds_option**
Abspielen von Sounds mit `beep`, `espeak` und dem Systemsound.

**use_pref_file_option**
Setting up a *preferences file path*.
Using the magic word `WORKINGFOLDER` in the preference file path will be replaced with the launched configuration path, e.g. `WORKINFOLDER/my_preferences`.

**use_send_zmq_option**
Wird verwendet, um *ZMQ-basierte ausgehende Nachrichten* zu initiieren.

**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**
In der *INI* definierte Programme einbetten.

**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 legt ein transparentes Bild oder ein farbiges Feld über den Hauptbildschirm, um den Fokus auf ein externes Ereignis zu betonen - normalerweise ein Dialog.

**messageDialog_option**
Richtet den Nachrichtendialog ein - wird für allgemeine Nachrichten verwendet.

**message_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Nachrichtendialog eingeblendet wird.

**closeDialog_option**
Richtet den Standarddialog zum Schließen des Bildschirms ein.

**entryDialog_option**
Richtet den numerischen Eingabedialog ein.

**entryDialogSoftKey_option**
Richtet eine schwebende Softwaretastatur ein, wenn der Eingabedialog fokussiert ist.

**entry_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Eingabedialog angezeigt wird.

**toolDialog_option**
Richtet den manuellen Werkzeugwechsel-Dialog ein, inklusive HAL-Pin.

**tool_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Werkzeugdialog angezeigt wird.

**ToolUseDesktopNotify**
Option zur Verwendung von Desktop-Benachrichtigungsdialogen für manuelle Werkzeugwechseldialoge.

**ToolFrameless**
Framesless-Dialoge können von Benutzern nicht einfach verschoben werden.

**fileDialog_option**
Richtet den Dateiauswahldialog ein.

**file_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Dateidialog angezeigt wird.

***keyboardDialog_option***
Richtet ein Tastatureingabe-Widget ein.

**keyboard_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Tastaturdialog angezeigt wird.

**vesaProbe_option**
Richtet den Versa-Style-Probe-Dialog ein.

**versaProbe_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Dialog `versaProbe` angezeigt wird.

**macroTabDialog_option**
legt den Makro-Auswahldialog fest.

**macroTab_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der `macroTab`-Dialog angezeigt wird.

**camViewDialog_option**
Richtet den Kameraausrichtungsdialog ein.

**camView_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der Dialog `camView` angezeigt wird.

**toolOffset_option**
Sets up the tool offset display/editor dialog.

**toolOffset_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der `toolOffset`-Dialog angezeigt wird.

**originOffset_option**
Richtet das Dialogfeld für die Anzeige/Editierung des Ursprungs ein.

**originOffset_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der `originOffset`-Dialog angezeigt wird.

**calculatorDialog_option**
Sets up the calculator entry dialog.

**calculator_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn das Dialogfeld ”Rechner” angezeigt wird.

**machineLogDialog_option**
Richtet einen Dialog ein, um Protokolle von der Maschine und QtVCP anzuzeigen.

**machineLog_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der machineLog-Dialog angezeigt wird.

**runFromLineDialog_option**
Richtet einen Dialog ein, der die Startoptionen anzeigt, wenn die Maschinenausführung von einer beliebigen Zeile aus gestartet wird.

**runFromLine_overlay_color**
Farbe der transparenten Ebene, die angezeigt wird, wenn der runFromLine-Dialog angezeigt wird.

**Setting Properties Programmatically** Der Screendesigner wählt die **Standardeinstellungen des Widgets screenOptions aus**.

Once chosen, most won't ever need to be changed. But if needed, some can be changed in the handler file or in stylesheets.

- **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 style sheets**:
  Here we can reference the widget by Qt Designer user defined name or by widget class name.

```
/* 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)
}
```

**Some settings are only checked on startup** so will not cause changes after startup. In these cases you would need to *make the changes in Qt Designer only*.

**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)***
Anzahl der angezeigten Fehler zu einem Zeitpunkt.

**shutdown_check *(bool)***
Ob ein Bestätigungsdialog erscheinen soll.

**sound_player_on** ***(bool)***
Schaltet alle Töne ein oder aus.

**[MCH_MSG_OPTIONS]**

**mchnMsg_play_sound** ***(bool)***
Um Warntöne wiederzugeben, wenn ein Dialogfeld angezeigt wird.

**mchnMsg_speak_errors** ***(bool)***
Verwendung von Espeak, um Fehlerbotschaften zu sprechen.

**mchnMsg_speak_text** ***(bool)***
Espeak verwenden, um alle anderen Nachrichten zu sprechen.

**mchnMsg_sound_type** ***(str)***
Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

**[USER_MSG_OPTIONS]**

**usermsg_play_sound** ***(bool)***
Um Warntöne wiederzugeben, wenn ein Dialogfeld angezeigt wird.

**userMsg_sound_type** ***(str)***
Ton, der abgespielt wird, wenn Benutzermeldungen angezeigt werden. Siehe Hinweise unten.

**userMsg_use_focusOverlay** ***(bool)***

**[SHUTDOWN_OPTIONS]**

**shutdown_play_sound** ***(bool)*** **, shutdown_alert_sound_type** ***(str)***
Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

**shutdown_exit_sound_type** ***(str)***
Ton, der abgespielt wird, wenn Nachrichten angezeigt werden. Siehe Hinweise unten.

**shutdown_msg_title** ***(str)***
Kurzer Titelstring, der im Dialog angezeigt wird.

**shutdown_msg_focus_text** ***(str)***
Großer Text, der in die Fokusebene eingeblendet wird.

**shutdown_msg_detail** ***(str)***
Longer descriptive string to display in dialog.

**NOTIFY_OPTIONS**

**notify_start_greeting** ***(bool)***
Whether to display a greeting dialog on start-up.

**notify_start_title** ***(str)***
Kurzer Titelstring.
Wenn die Option ”Sprechen” ebenfalls ausgewählt ist, wird der Titel mit Espeak gesprochen.

**notify_start_detail** ***(str)***
Längere Zeichenfolge zur Beschreibung.

**notify_start_timeout** ***(int)***
Zeit in Sekunden bis zur Anzeige vor dem Schließen.

*_sound_type Einträge

- **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**
  If the *Espeak* module (`python3-espeak`) is installed, you can use the `SPEAK` entry to pronounce text:
- **`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

This widget allows you to **manually select a G5x user system by pressing and holding**.

Wenn Sie den Text der Schaltfläche nicht festlegen, wird sie automatisch auf das aktuelle System aktualisiert.

Es basiert auf PyQts *QToolButton*.

#### 12.7.2.20 `MacroTab` - Spezielles Makro-Widget

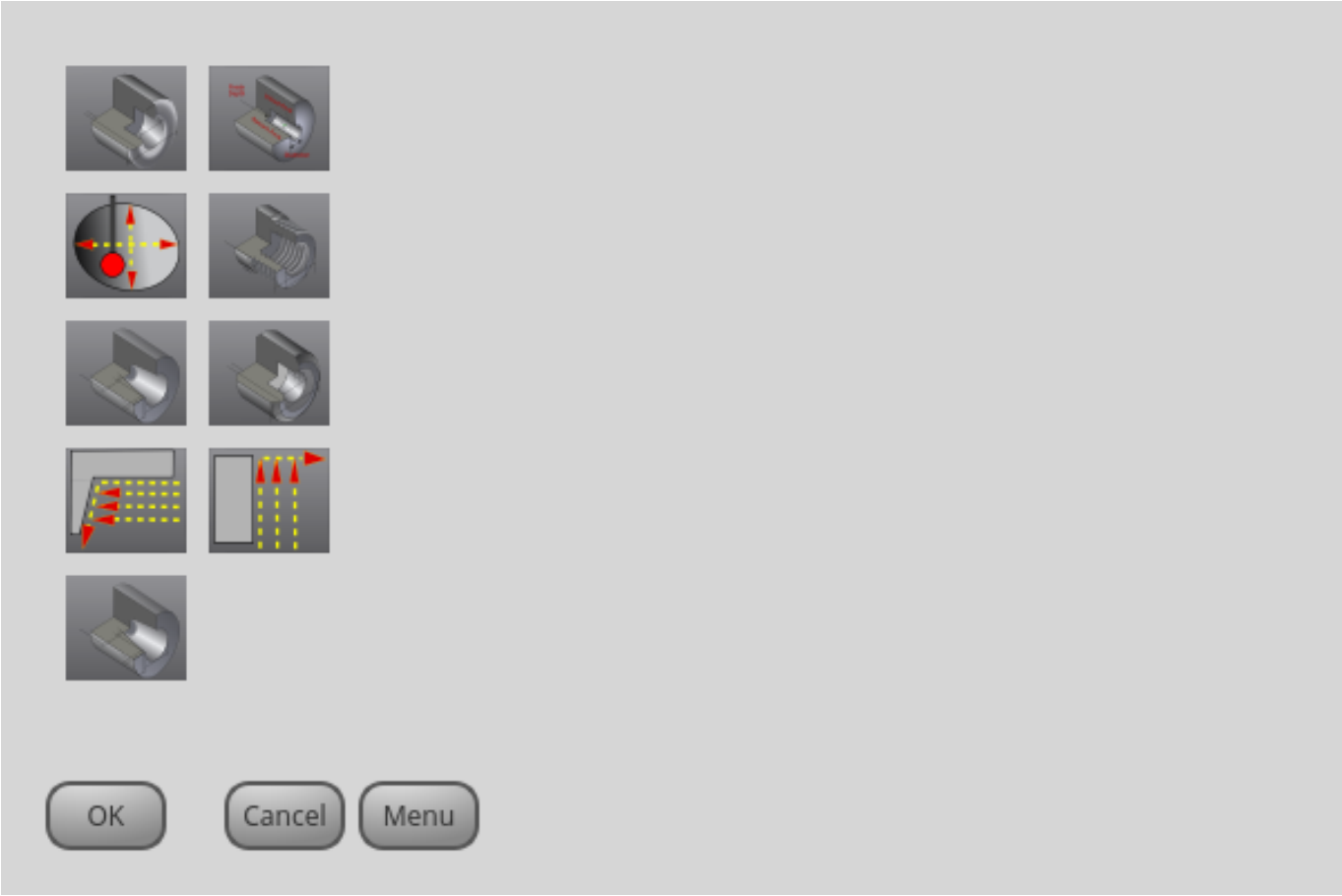



Abbildung 12.90: QtVCP `MacroTab`: Spezielles Makro-Widget

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)**
  : **Icon image file name** as optional second field.
    If the second entry is missing the same image will be used for macro and image.

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** There are also **embedded commands** available from this widget.

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**
Starts LinuxCNC Calibration

**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.

An edit line is embedded for MDI commands. The same MDILine embedded commands may be accessed from this widget.

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

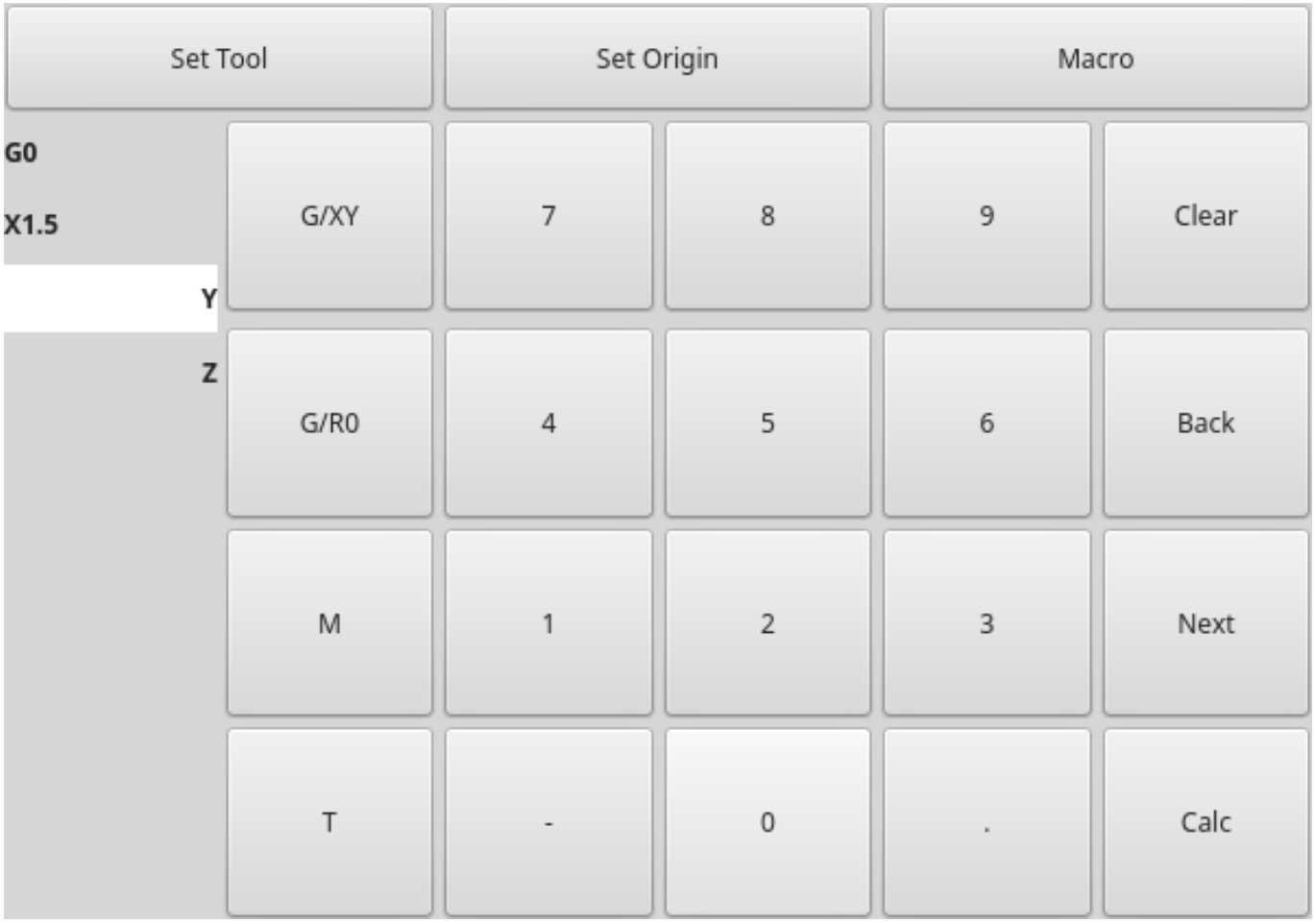



Abbildung 12.91: QtVCP `MDITouchy`: Touchscreen-MDI-Eingabe-Widget

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`:

- First press one of the `G/XY`, `G/R0`, `M` or `T` button. On the left will show the entry fields that can be filled out.
- Drücken Sie dann „Weiter" und „Zurück", um zwischen den Feldern zu navigieren.
- `Calc` öffnet einen Taschenrechnerdialog.
- `Clear` clears the current entry.
- `Set Tool` will call for a tool change.
- `Set Origin` will allow setting the origin of the current G6x system.
- `Macro` will call any available macro ngc programs.

The widget *requires an explicit call to MDITouchy Python code* to *actually run the MDI command*:

- **For handler file code**
  If the widget was named *mditouchy* in Qt Designer, the command below would run the displayed MDI command:

```
self.w.mditouchy.run_command()
```

- **For action button use**
  If the widget was named *mditouchy* in Qt Designer, use the action button's *Call Python commands* option and enter:

```
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
```

Now, place the macro in a file named `macro_name.ngc` in the `PROGRAM_PREFIX` directory, or into any directory in the `SUBROUTINE_PATH` specified in the INI file.

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.

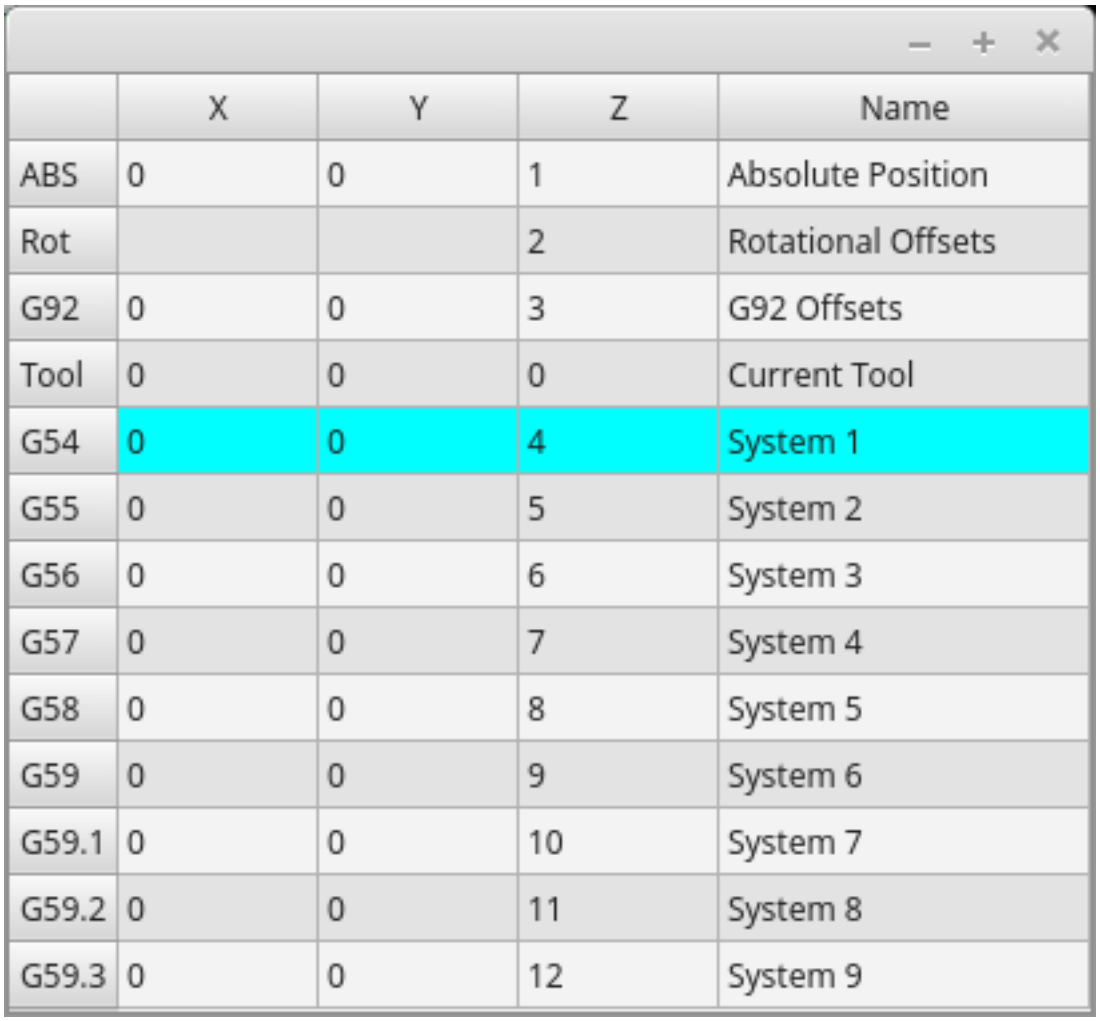
When you invoke the macro by pressing the Macro button you can enter values for parameters (`xinc` and `yinc` in our example).
These are passed to the macro as positional parameters: `#1`, `#2`… `#N` respectively.
Parameters you leave empty are passed as value `0`.

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 12.92: 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
- E-stop 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

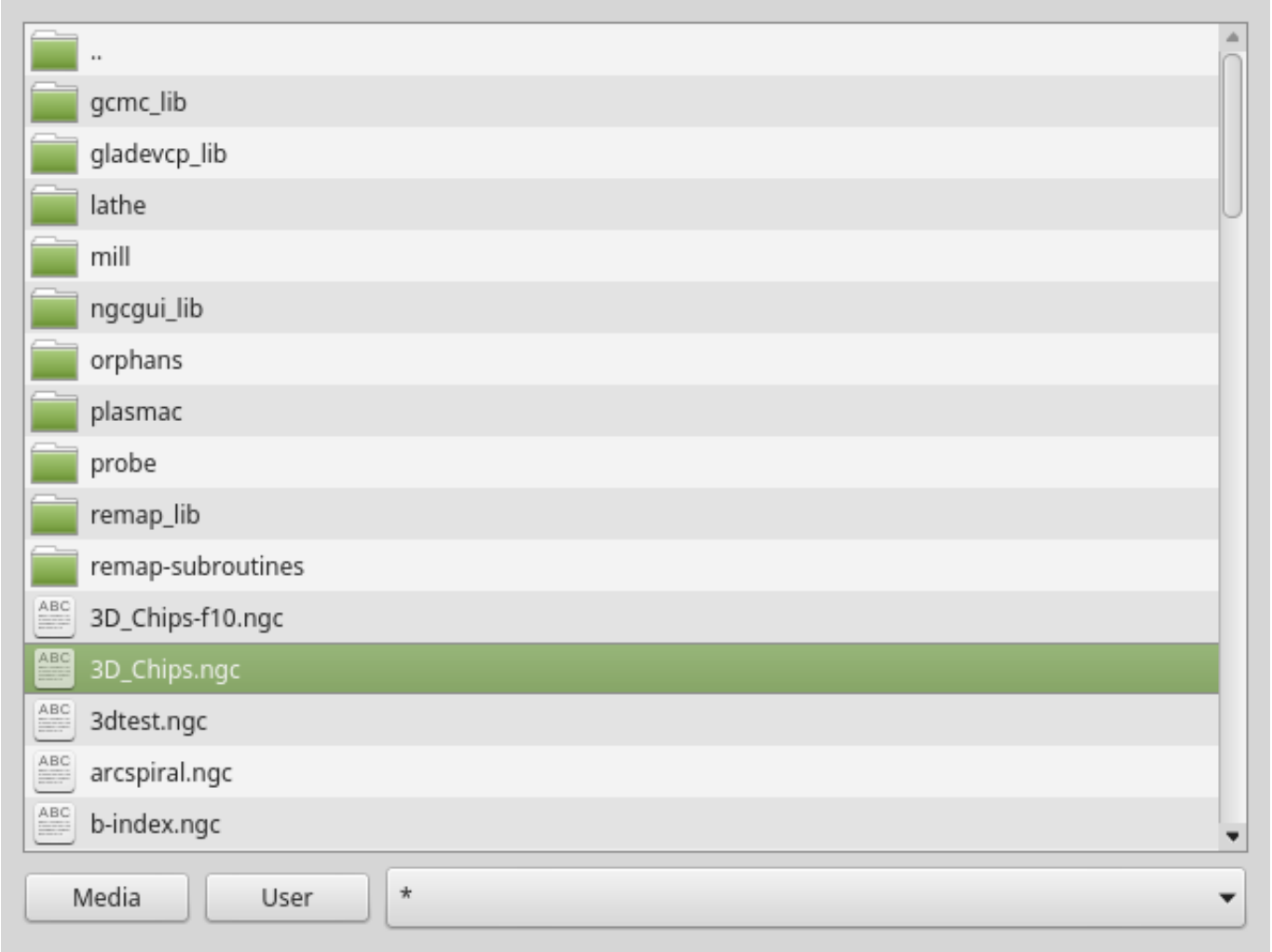



Abbildung 12.93: QtVCP `FileManager`: File Loading Selector Widget

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 is a file.

```
temp = FILEMANAGER.getCurrentSelected()
print('filepath={}'.format(temp[0]))
if temp[1]:
    print('Is a file')
```

Stylesheets Properties

**`doubleClickSelection` *(bool)***
Determines whether or not to *require double clicking on a folder*.
Single clicking a folder (False) is enabled by default and is intended for touch screen users.
The following shows an example of how to set this property:

```
#filemanager {
    qproperty-doubleClickSelection: True;
}
```

**`showListView` *(bool)***
Determines whether or not to *show the file/folder structure in list form*.
Table view (False) is enabled by default.
The following shows an example of how to set this property:

```
#filemanager {
    qproperty-showListView: True;
}
```

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

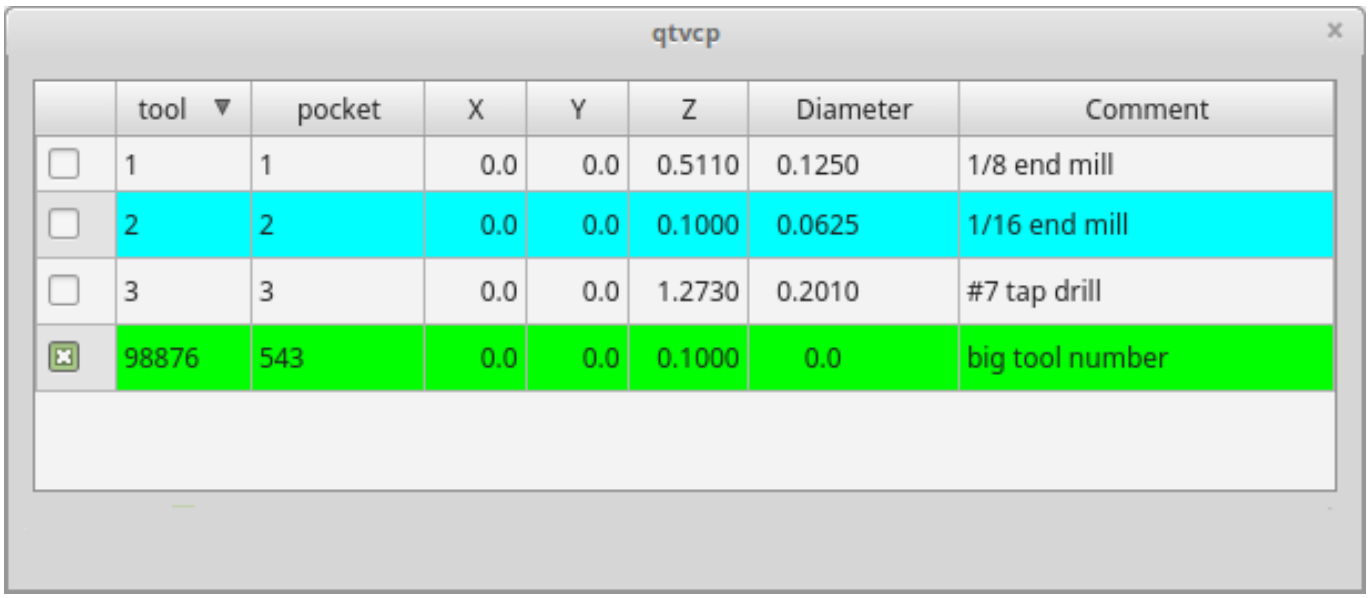



Abbildung 12.94: QtVCP `ToolOffsetView`: Tools Offsets View And Edit Widget

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')
```

und in Stylesheets:

```
ToolOffsetView{
  qproperty-styleColorHighlist: lightblue;
  qproperty-styleColorSelected: #444;
}
```

**Funktionen** Die Funktion `ToolOffsetView` hat einige Funktionen, die für Screenbuilder nützlich sind, um Aktionen hinzuzufügen:

**add_tool()**
Fügt ein leeres Dummy-Werkzeug (99) hinzu, das der Benutzer nach Belieben bearbeiten kann.

**delete_tools()**
Löscht die in der Checkbox ausgewählten Werkzeuge.

**get_checked_list()**
Gibt eine Liste der durch Ankreuzfelder ausgewählten Werkzeuge zurück.

**set_all_unchecked()**
Hebt die Markierung aller ausgewählten Werkzeuge auf.

**Example for handler file executing aforementioned functions.**

```
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

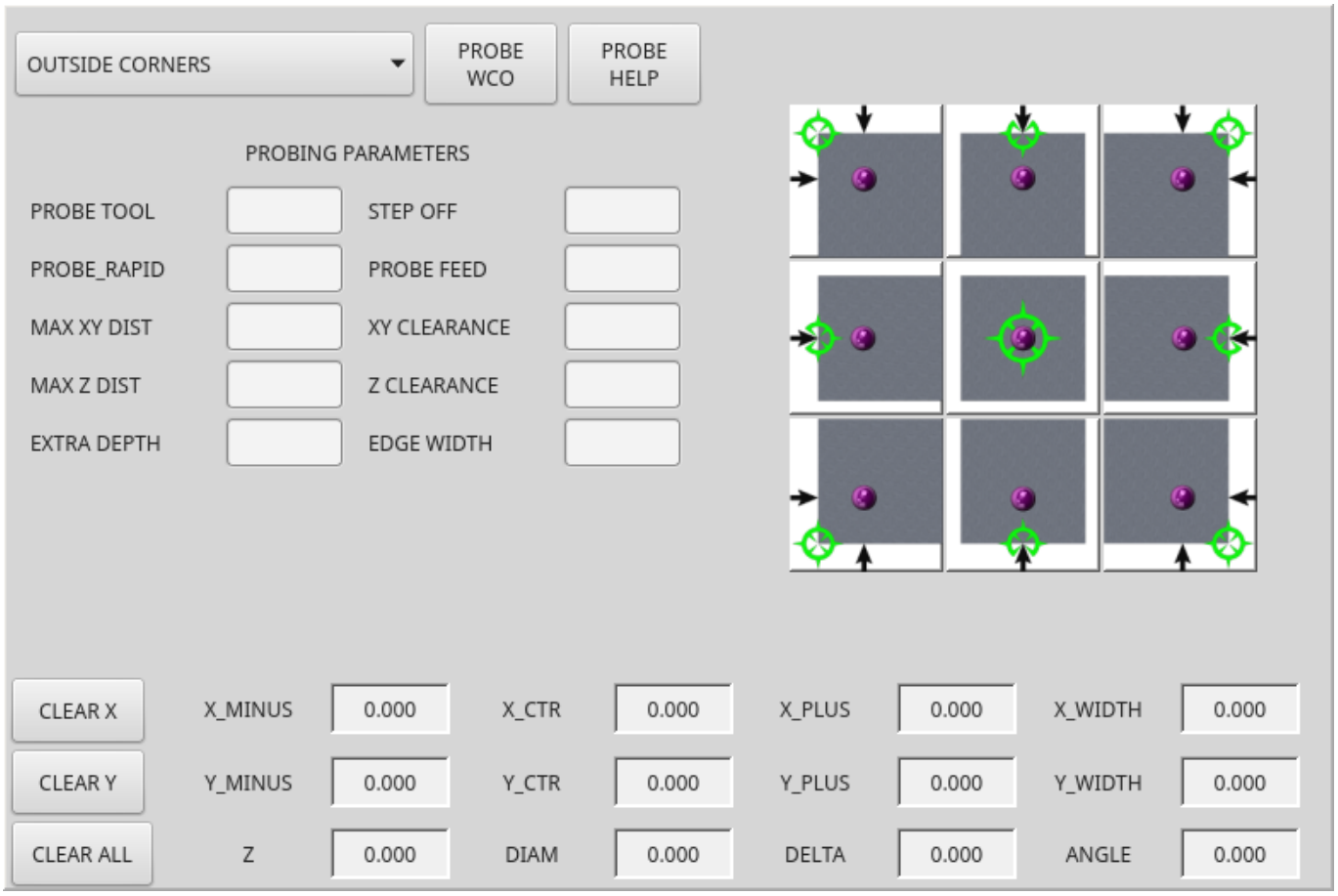



Abbildung 12.95: QtVCP *BasicProbe*: Einfaches Fräs-Tast-Widget

Widget zum **Sondieren auf einer Fräse**. Wird vom *QtDragon*-Bildschirm verwendet.

#### 12.7.2.33 `VersaProbe` - Mill Probing Widget

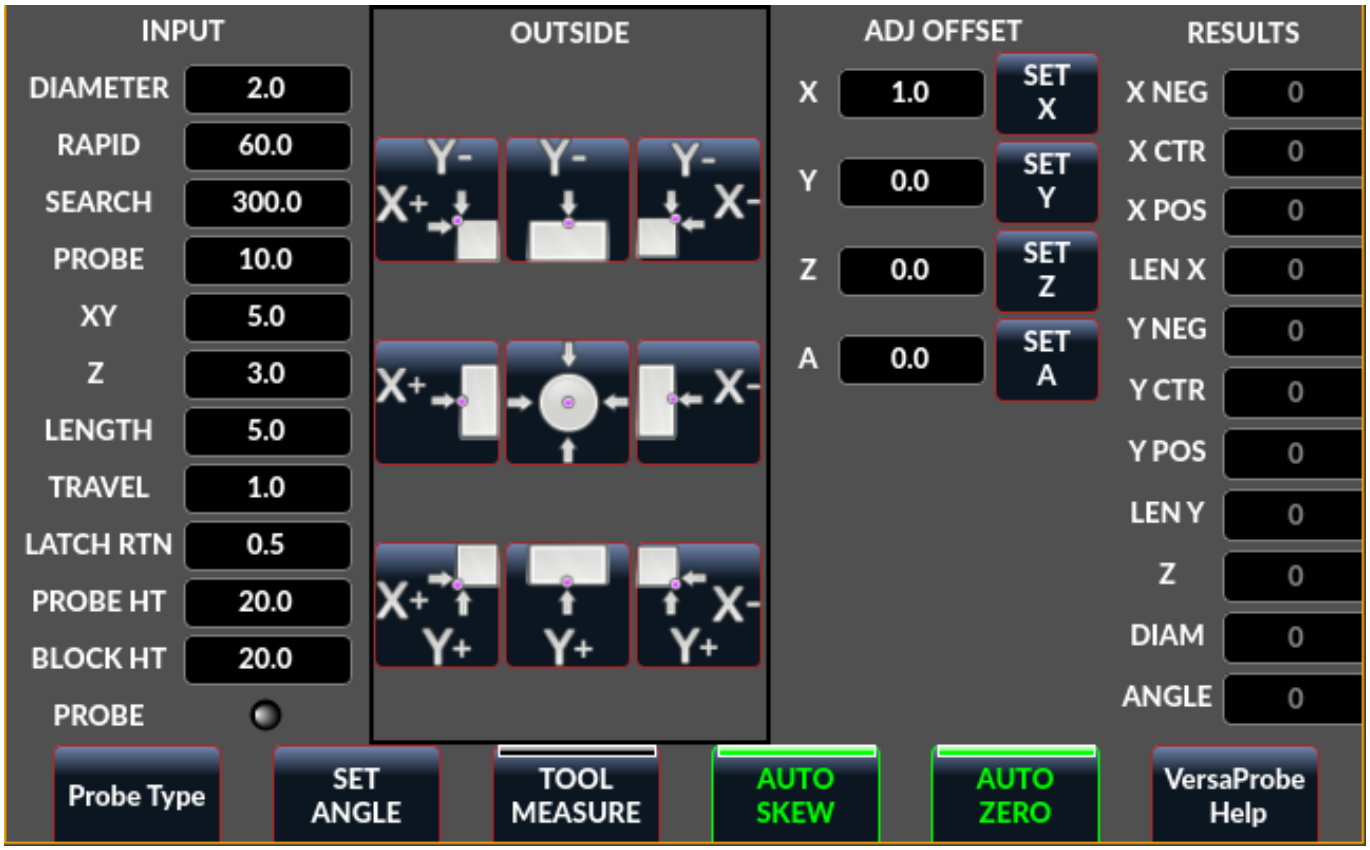



Abbildung 12.96: QtVCP `VersaProbe`: Mill Probing Widget

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** You can show dialogs directly with *Python code*, but a safer way is to **use `STATUS` messages** to request the dialog to launch and to return the gathered information.

- **Register to `STATUS` channel**:
  To set this up, first register to catch the `general` message from `STATUS`:

```
STATUS.connect('general',self.return_value)
```

- **Add a function to call a dialog**:
  This function must *build a message `dict` to send to the dialog*.
  This message will be passed back in the general message with the addition of the *`return` variable*.
  It is possible to add *extra user information* to the message. The dialog will ignore these and pass them back.

**NAME**
Launches code name of dialog to show.

**ID**
A unique id so we process only a dialog that we requested.

**TITLE**
The title to use on the dialog.

```
def show_dialog(self):
    mess = {'NAME':'ENTRY','ID':'__test1__',
            'TITLE':'Test Entry'}
    ACTION.CALL_DIALOG, mess)
```

- **Hinzufügen einer Callback-Funktion, um die allgemeine Nachricht zu verarbeiten**: Beachten Sie, dass diese Funktion *alle allgemeinen Nachrichten* abruft, so dass die `dict`-Keynamen sind nicht garantiert vorhanden sind. Es ist ratsam, die Funktion `.get()` zu benutzen und/oder `try`/`except` zu verwenden. Diese Funktion sollte:
  - Prüfen Sie, ob der Name und die Kennung mit den Angaben übereinstimmen, die wir gesendet haben,
  - Extrahieren Sie dann den Rückgabewert und alle Benutzervariablen.

```
# Verarbeitung der STATUS return message
def return_value(self, w, message):
    rtn = message.get('RETURN')
    code = bool(message.get('ID') == '__test1__')
    name = bool(message.get('NAME') == 'ENTRY')
    if code and name and not rtn is None:
        print('Entry return value from {} = {}'.format(code, rtn))
```

#### 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

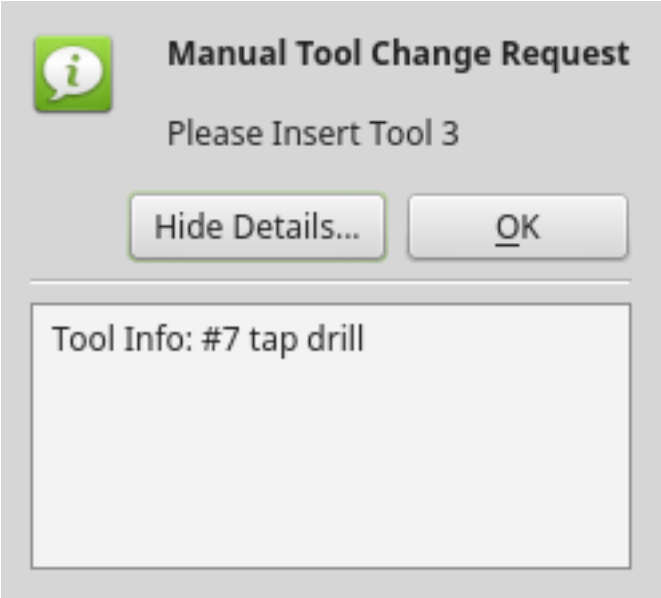



Abbildung 12.97: QtVCP `ToolDialog`: Dialog zum manuellen Werkzeugwechsel

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

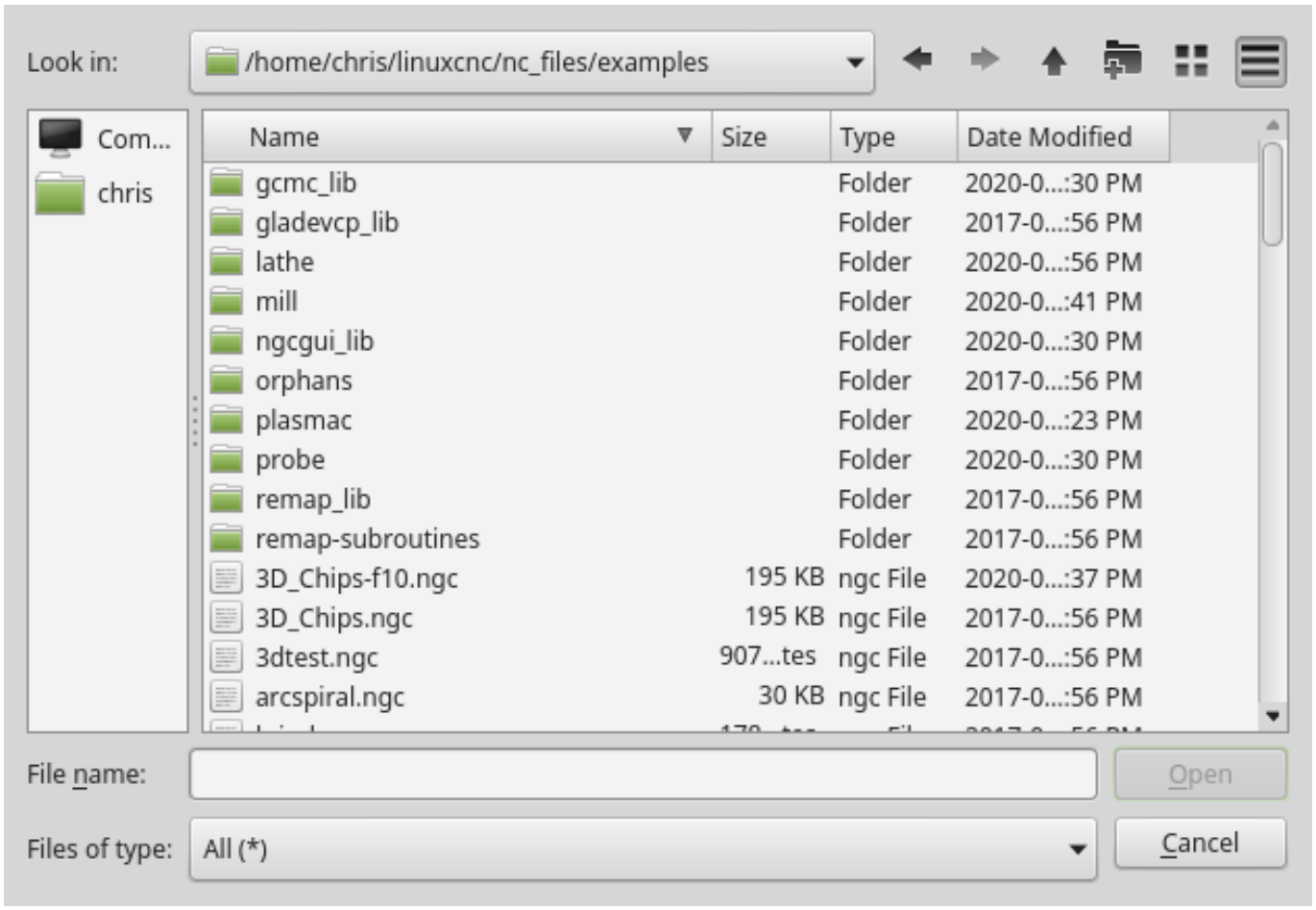



Abbildung 12.98: QtVCP `FileDialog`: Widget für das Laden und Speichern von Dateien

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

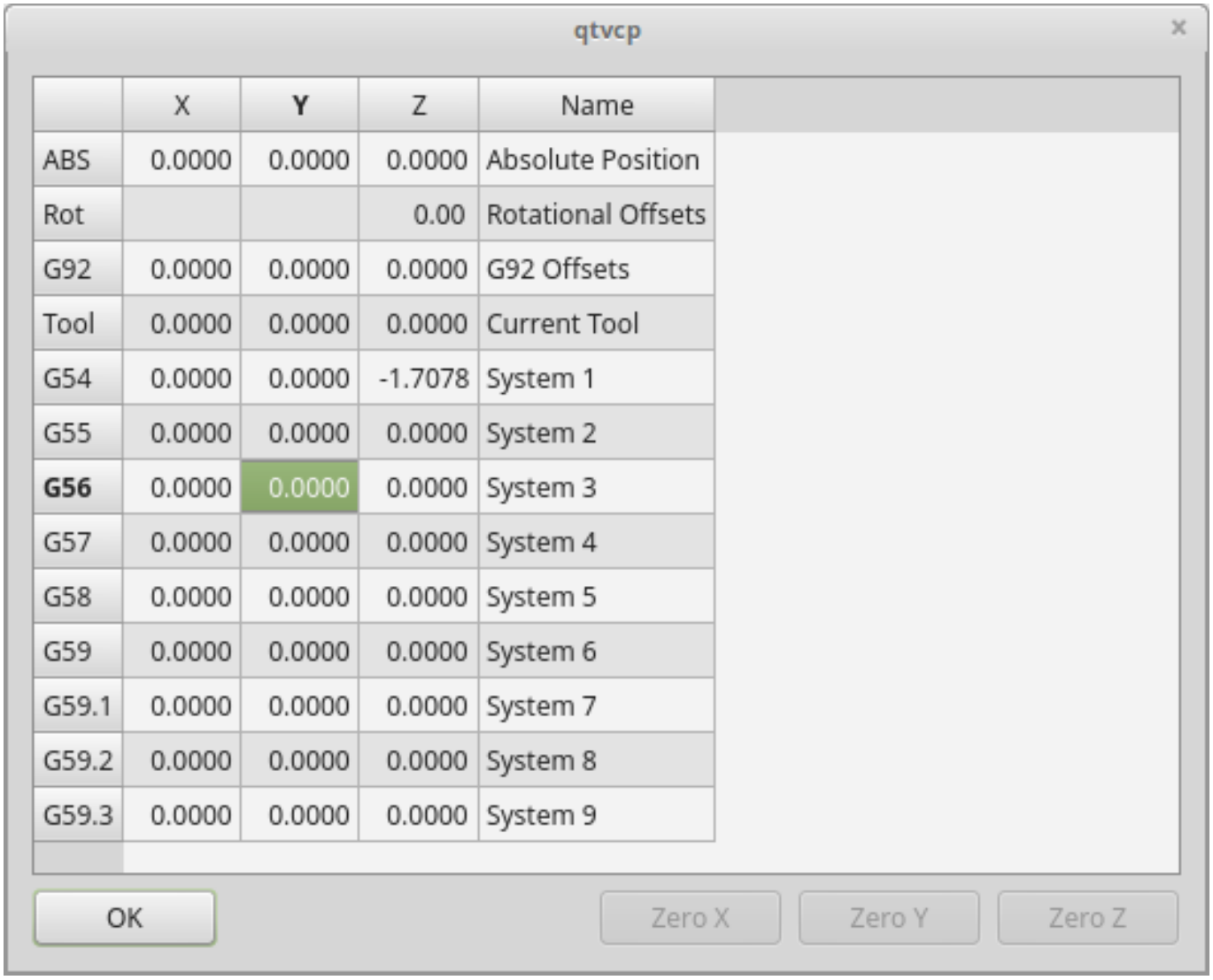



Abbildung 12.99: QtVCP `OriginOffsetDialog`: Widget zur Einstellung des Ursprungsversatzes

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

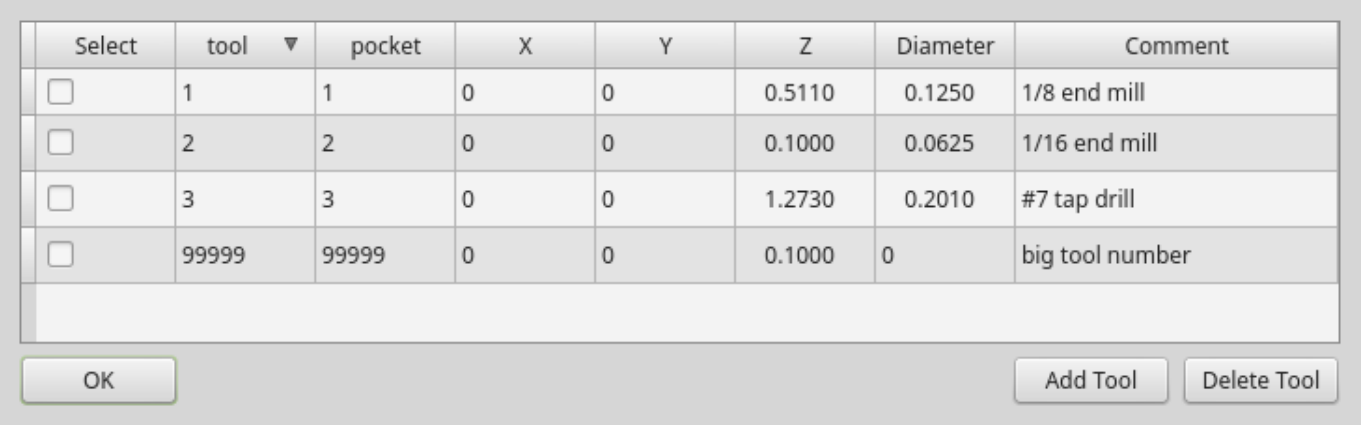



Abbildung 12.100: QtVCP `ToolOffsetDialog`: Werkzeug-Offset-Einstellungsdialog-Widget

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

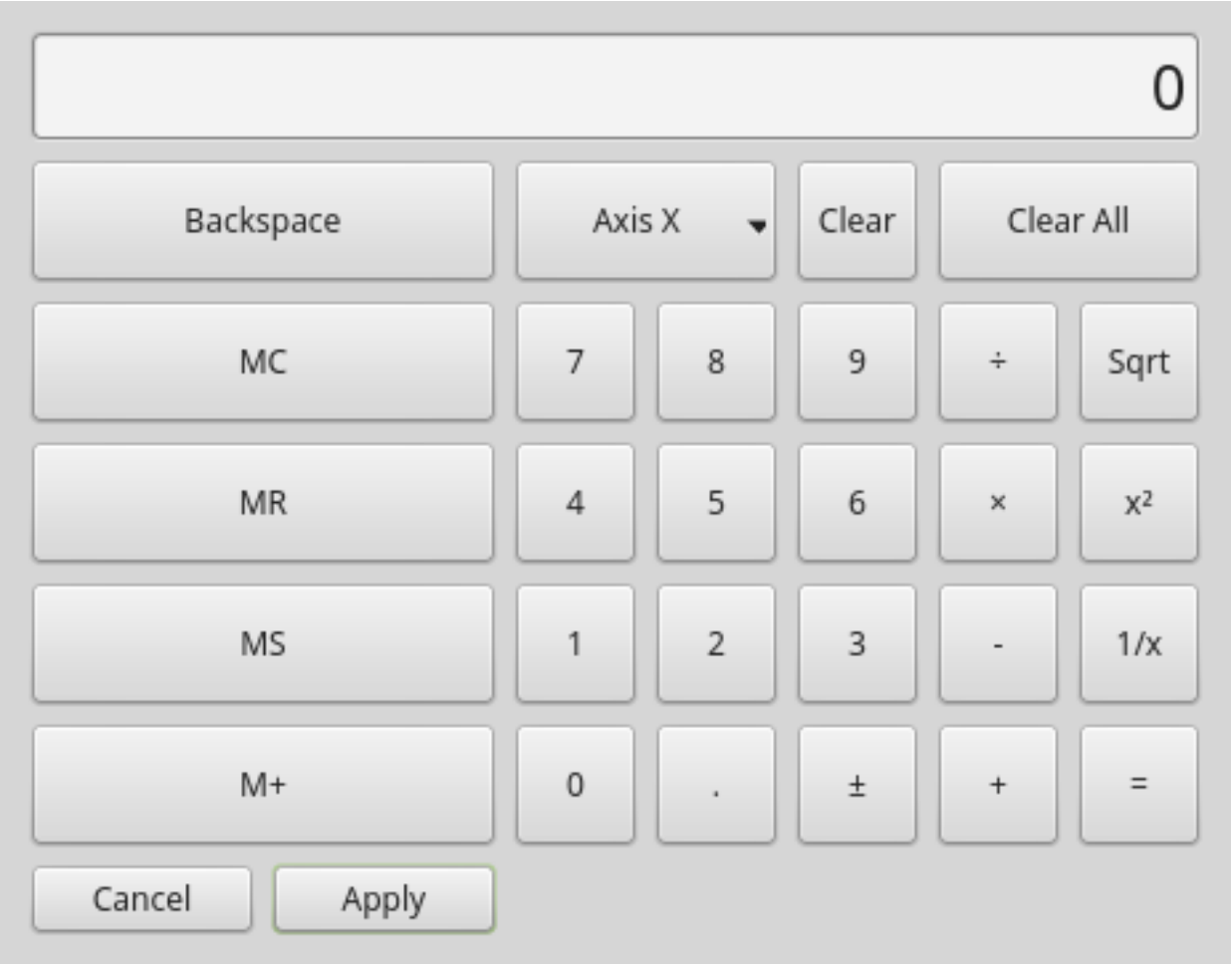



Abbildung 12.101: QtVCP `CalculatorDialog`: Calculator Dialog Widget

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 12.102: 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

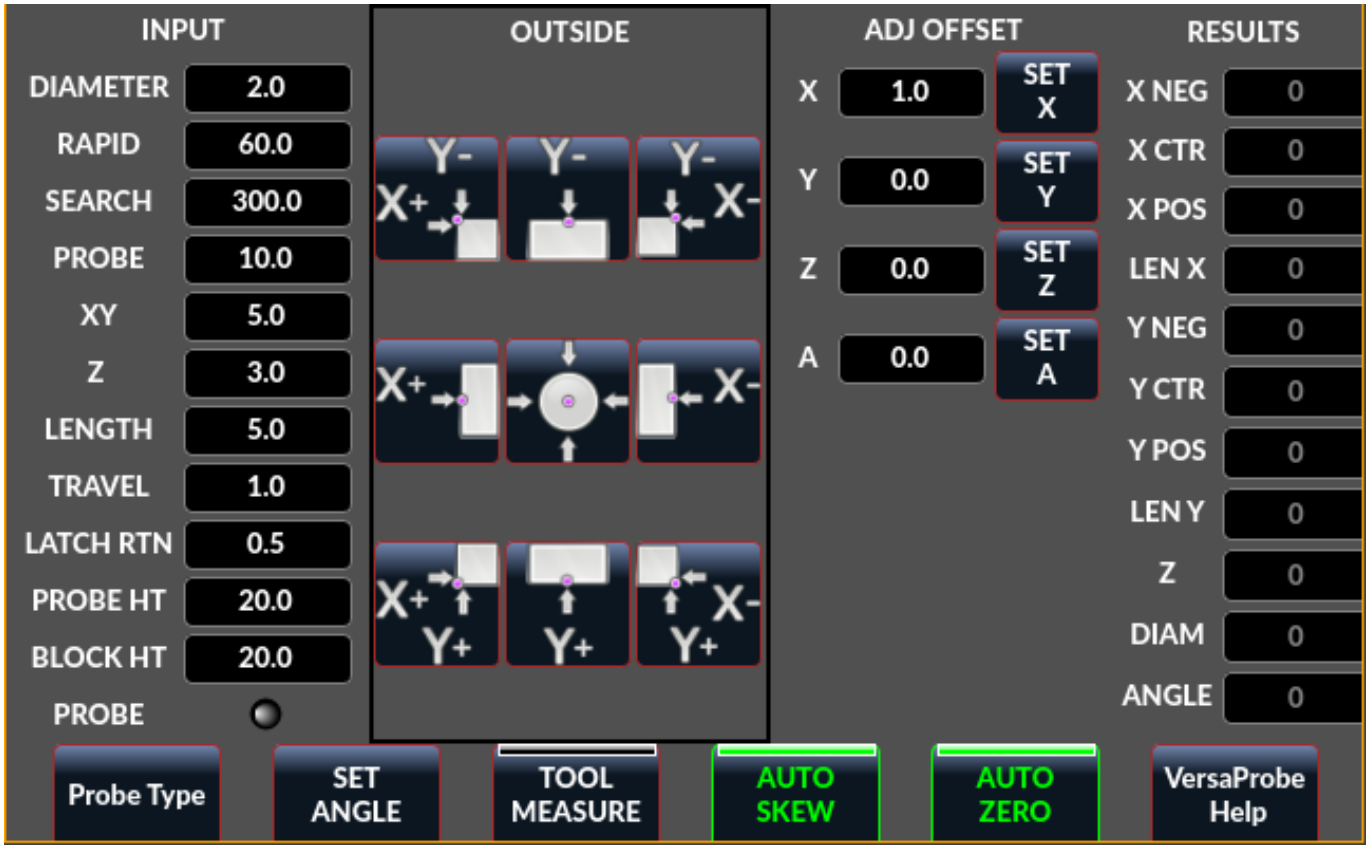



Abbildung 12.103: QtVCP `VersaProbeDialog`: Part Touch Probing Dialog Widget

This is a dialog to display a **part probing screen based on Verser Probe v2**.
Es basiert auf PyQts *QDialog*.

#### 12.7.3.12 `MachineLogDialog` - Machine and Debugging Logs Dialog Widget

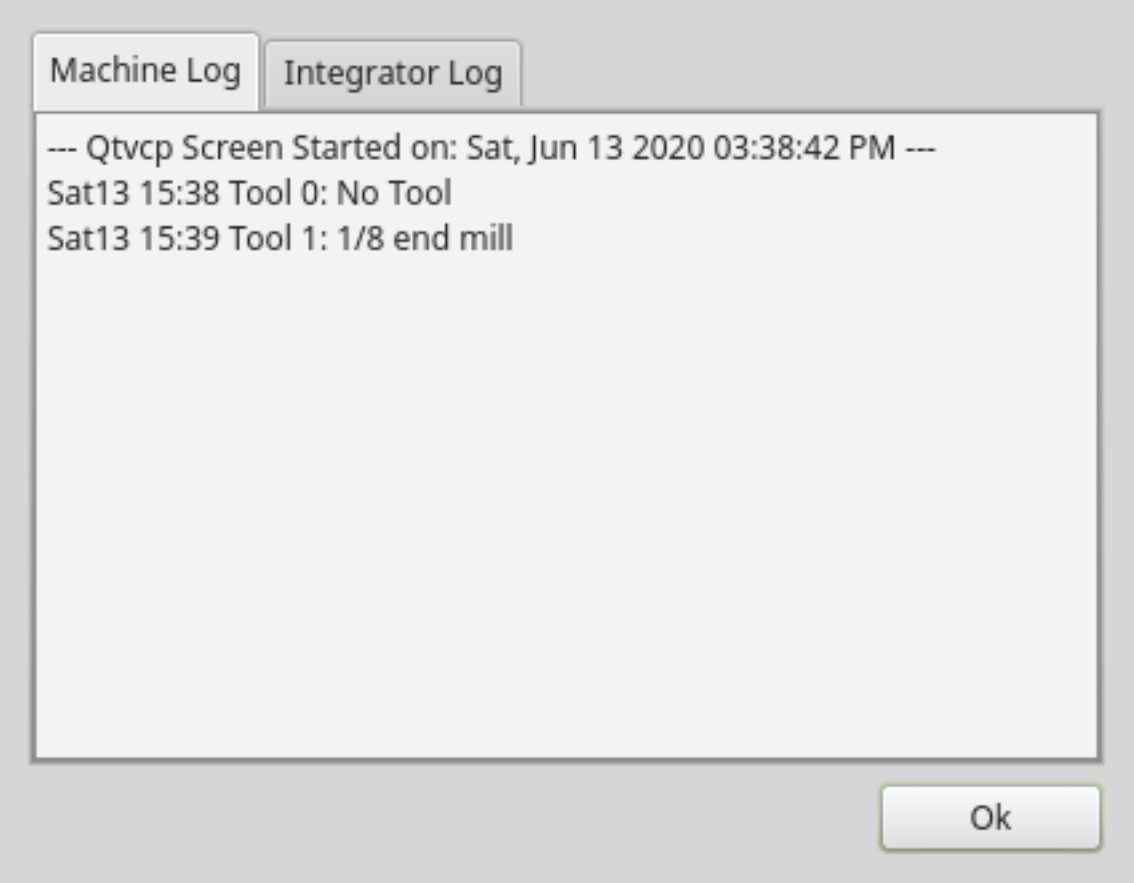



Abbildung 12.104: QtVCP `MachineLogDialog`: Machine and Debugging Logs Dialog Widget

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

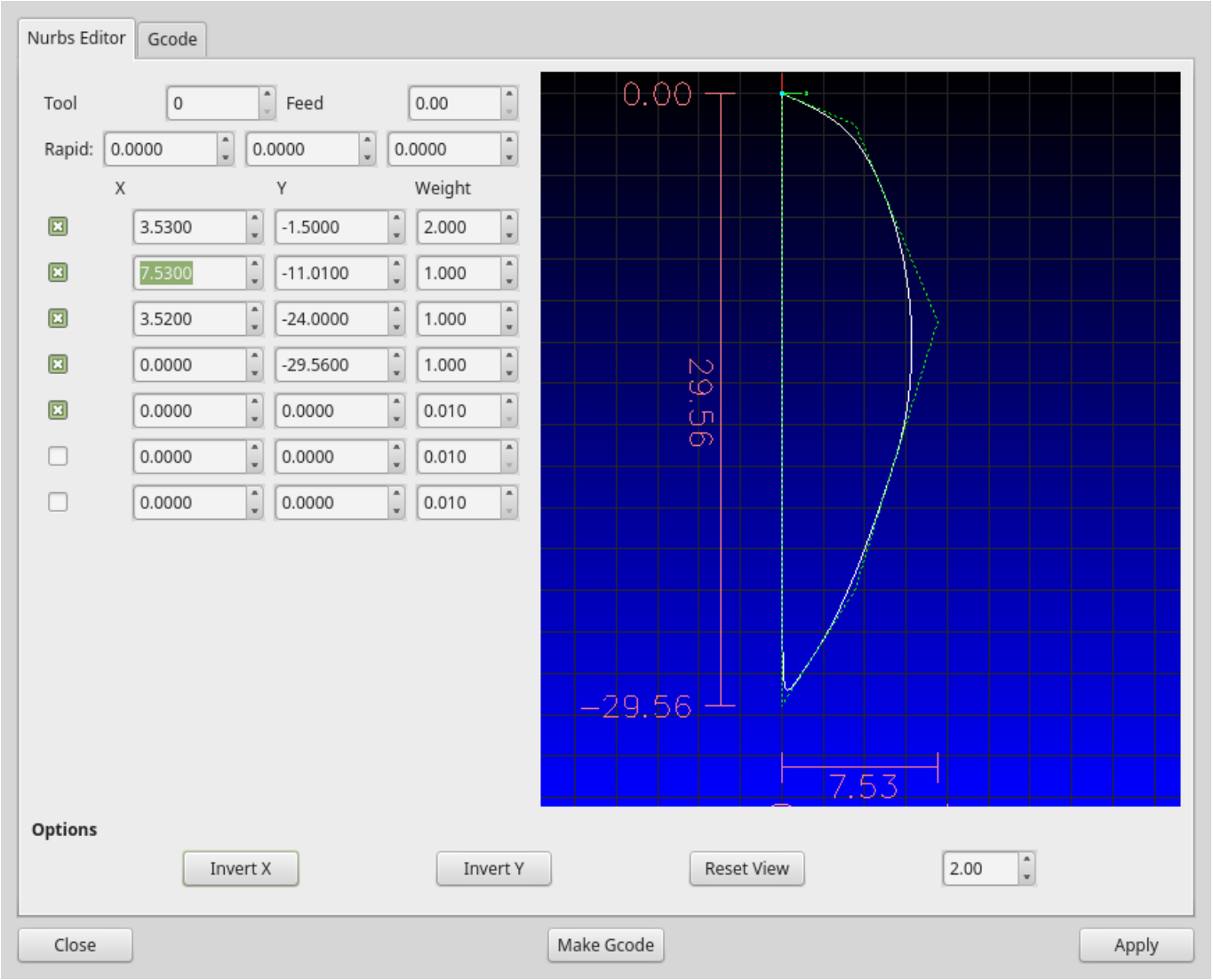



Abbildung 12.105: QtVCP `NurbsEditor`: NURBS Editing Widget

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** Es gibt **aufgezählte Konstanten, die zur Referenzierung von Indikatorpositionen verwendet werden**.
Sie werden im Eigenschaftseditor des Qt Designer-Editors oder im Python-Code verwendet.

**NONE , LEFT, L , RIGHT, R , CENTER, C , TOP, T , BOTTOM, B , LEFTRIGHT, X , TOPBOTTOM, A**
: Für Python-Handler-Code verwenden Sie den Widget-Namen in Qt Designer plus die Referenzkonstante:

```
self.w.joypadname.set_highlight(self.w.joypadname.LEFT)
```

### Useful Override-able Functions

```
def _pressedOutput(self, btncode):
    self.joy_btn_pressed.emit(btncode)
    self['’’.format(btncode.lower())].emit(True)

def _releasedOutput(self, btncode):
    self.joy_btn_released.emit(btncode)
    self['joy_{}_pressed’.format(btncode.lower())].emit(False)
```

As coded these function *issue (emit) PyQt5 signals (joy_btn_pressed and joy*<letter>_pressed) for the any button pressed or released_.

Signal *joy_btn_pressed* outputs a string code for the button.
Signal *joy_<letter>_pressed* outputs a bool value.

You could override the functions to do something else if making a custom widget:

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`**
: Color selection for the center LED ring to be displayed, when the `BASENAME.light.center` *HAL pin* is `True` or `False`.

**`text_color`**
: Auswahl der Farbe für den Text des Buttons.

**`button_font`**
: Auswahl der Schriftart für den Text des Buttons.

Die obigen Eigenschaften könnten gesetzt werden in:

- **Stylesheets**:
  You would usually use the Qt Designer widget name with *# prefix* to set individual widget properties, otherwise you would use the `JoyPad` *class name* to set all `JoyPad` widgets the same:

```
#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')
```

#### 12.7.4.3 *WebWidget*

This widget will create a html/pdf viewing page using the QtWebKit or QtWebEngine libraries. The newer QtWebEngine is preferred if both are on the system.
If the QtWebEngine library is used with the Qt Designer editor, a placeholder QWidget will show in Qesigner. This will be replaced with the QtWebEngine widget at run time.

### 12.7.5 BaseClass/Mixin-Widgets

These widgets are used to **combine different properties and behaviours into other widgets**.

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 12.106: 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* (engl. Nebel)
- *Block Delete*
- *Optionaler Stop*
- *Manual*
- *MDI*
- *Auto*
- *Spindle Stopped*
- *Spindel vorwärts*
- *Spindel rückwärts*
- *On Limits*

Einige `indicator_status_options` enthält eine Eigenschaft, die mit einem *stylesheet* verwendet werden kann, um die Farbe der Schaltfläche basierend auf dem Zustand der Eigenschaft in LinuxCNC zu ändern.
Derzeit sind diese Status-Eigenschaften können verwendet werden, um Auto-Stil Schaltflächen:

- `is_estopped_status` schaltet die Eigenschaft `isEstop` um
- `is_on_status` schaltet die Eigenschaft `isStateOn` um
- `is_manual_status`, `is_mdi_status`, `is_auto_status` schalten die Eigenschaften `isManual`, `isMDI`, und `isAuto` um.
- `is_homed_status` schaltet die Eigenschaft *isAllHomed* um

Hier ist ein Beispiel-Stylesheet-Eintrag, der den Hintergrund von Mode-Button-Widgets festlegt, wenn sich LinuxCNC in diesem Modus befindet:

```
ActionButton[isManual=true] {
    background: red;
}
ActionButton[isMdi=true] {
    background: blue;
}
ActionButton[isAuto=true] {
    background: green;
}
```

Here is how you specify a particular widget by its objectName in Qt Designer:

```
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.

Choosing the **checked_state_text_option** allows a *checkable* button to *change the text based on its checked state*.

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.

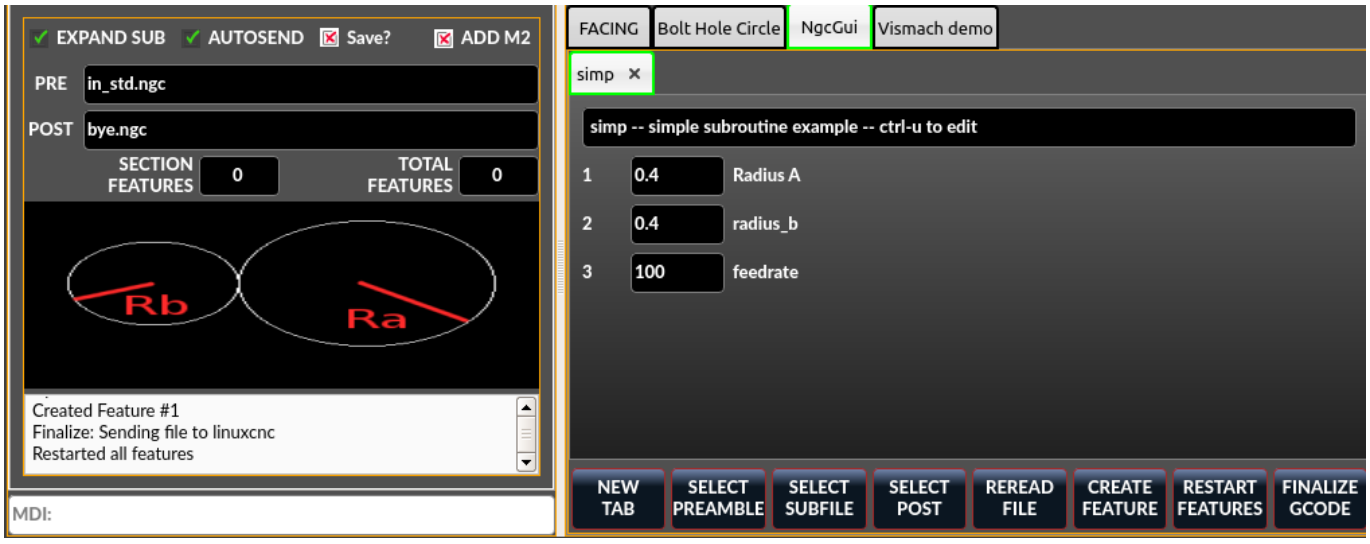
#### 12.7.6.3 Facing

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 of NGC subroutine selector (Shown as used in QtDragon).

LinuxCNC needs to know where to look to run the subroutines.
If the subroutine calls other subroutines or custom M codes, those paths must be added too.

```
[RS274NGC]
SUBROUTINE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ ↩
    ngcgui_lib/utilitysubs
```

QtVCP needs to know where to open subroutines from.
You can also specify subroutines to be pre-opened in tabs.

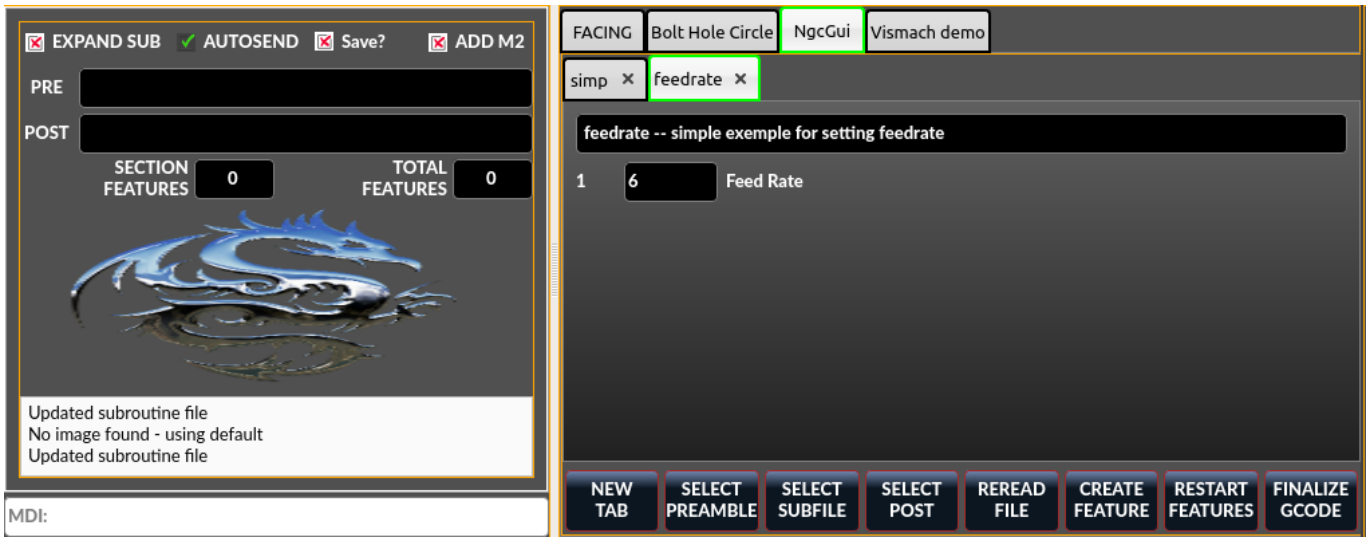
```
[DISPLAY]
# NGCGUI subroutine path.
# This path must also be in [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# pre selected programs tabs
# specify filenames only, files must be in the NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

- *NEW TAB* - add new blank tab to NGCGUI
- *SELECT PREAMBLE* - select a file that add preamble G-code
- *SELECT SUBFILE* - select a NGCGUI subroutine file
- *SELECT POST* - select a file that add post G-code
- *REREAD FILE* - reload the subroutine file
- *CREATE FEATURE* - add feature to the list
- *RESTART FEATURE* - remove all features from the list
- *FINALIZE GCODE* - create the full G-code and send it to LinuxCNC/a file

You can create your own subroutines for use with NGCGUI. They must follow these rules:

- Um ein Unterprogramm für die Verwendung mit NGCGUI zu erstellen, müssen der Dateiname und der Name des Unterprogramms identisch sein.
- The subroutine must be in a folder within LinuxCNC’s INI designated search path.
- On the first line there may be a comment of type info:
- The subroutine must be surrounded by the sub and endsub tags.
- Die verwendeten Variablen müssen nummerierte Variablen sein und dürfen keine Nummer überspringen.
- Kommentare und Voreinstellungen können enthalten sein.
- If an image file of the same name is in the folder, it will be shown.

```
(info: feedrate -- simple example for setting feedrate)
o<feedrate> sub
    #<feedrate>       = #1 (= 6 Feed Rate) ; comments in brackets will be shown in ngcui
    f#<feedrate>
o<feedrate> endsub
```




#### 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())
```

  Wenn sich LinuxCNC in *Zustand ”machine on” befindet, wird in diesem Beispielcode state* die Funktion *self.on_state_on* aufgerufen.
  Wenn LinuxCNC in *Zustand ”machine off”* ist, wird die Funktion *self.on_state_off* aufgerufen.

- 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

- **Importiere `Info` Modul**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** 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`** Bibliothek wird verwendet, um **die Bewegungssteuerung von LinuxCNC** zu steuern.

It tries to hide incidental details and add convenience methods for developers.

#### 12.8.3.1 Helpers

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(_filena`**

Öffnen Sie das G-Code-Filterprogramm.

#### 12.8.3.2 Anwendung

- **Importiere `Action` Modul**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Action
```

- **Instantiierung des Moduls `Action`**
  Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
###########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
###########################################

ACTION = Action()
```

- **Zugriff auf `ACTION`-Befehle**
  Verwenden Sie eine allgemeine Syntax wie die folgende:

```
ACTION.SET_ESTOP_STATE(state)
ACTION.SET_MACHINE_STATE(state)

ACTION.SET_MACHINE_HOMING(joint)
ACTION.SET_MACHINE_UNHOMED(joint)

ACTION.SET_LIMITS_OVERRIDE()

ACTION.SET_MDI_MODE()
ACTION.SET_MANUAL_MODE()
ACTION.SET_AUTO_MODE()

ACTION.SET_LIMITS_OVERRIDE()

ACTION.CALL_MDI(code)
ACTION.CALL_MDI_WAIT(code)
ACTION.CALL_INI_MDI(number)

ACTION.CALL_OWORD()

ACTION.OPEN_PROGRAM(filename)
ACTION.SAVE_PROGRAM(text_source, fname):

ACTION.SET_AXIS_ORIGIN(axis,value)
ACTION.SET_TOOL_OFFSET(axis,value,fixture = False)

ACTION.RUN()
```

```
ACTION.ABORT()
ACTION.PAUSE()

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 information*
- **[1]** will be *wear tools information* (tool numbers will be over 10000; Fanuc style tool wear)

By default, adds a blank 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. it does not feature 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 Referenzierte Pfade

**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-Datei Pfad.

**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**
LinuxCNCs Python-Bibliothek.

**PATH.LIBDIR**
Der Ordner der QtVCP-Bibliothek.

**PATH.WIDGET**
Der QtVCP-Widget-Ordner.

**PATH.PLUGIN**
Der QtVCP-Widget-Plugin-Ordner.

**PATH.VISMACHDIR**
Verzeichnis, in dem sich die vorgefertigten Vismach-Dateien befinden.

Derzeit nicht verwendet:

**PATH.LOCALEDIR**
Ordner für Übersetzungen.

**PATH.DOMAIN**
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

- **Importiere Path Modul**
  Fügen Sie diesen Python-Code in Ihren Import-Abschnitt ein:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Path
```

- **Instantiierung des Moduls Path**
  Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

PATH = Path()
```

## 12.8.6 VCPWindow

Das **VCPWindow** Modul gibt **Referenz auf das MainWindow und Widgets**.

Typically this would be used for a library (e.g., the 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* Modul**
  Fügen Sie diesen Python-Code zu Ihrem Importabschnitt hinzu:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **VCPWindow-Modul instanziieren**+ Fügen Sie diesen Python-Code in Ihren instantiate-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

WIDGETS = VCPWindow()
```

## 12.8.7 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* wird zur **Anzeige von Daten eines HAL-Pins** verwendet.
Laden Sie ein Halmeter mit:

```
AUX_PRGM.load_halmeter()
```

**load_ladder()**
SPS-Programm *ClassicLadder* laden:

```
AUX_PRGM.load_ladder()
```

**load_status()**
Laden des LinuxCNC status Programms:

```
AUX_PRGM.load_status()
```

**load_halshow()**
*HALshow* laden, Anzeigeprogramm konfigurieren:

```
AUX_PRGM.load_halshow()
```

**load_halscope()**
Laden des *HALscope* Programms:

```
AUX_PRGM.load_halscope()
```

**load_tooledit()**
Programm *Tooledit* laden:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

**load_calibration()**
Programm zur *Kalibrierung* laden:

```
AUX_PRGM.load_calibration()
```

**keyboard_onboard()**
Load *onboard/Matchbox keyboard*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

#### 12.8.7.2 Anwendung

- **Modul `Aux_program_loader` importieren**
  Fügen Sie diesen Python-Code zu Ihrem Importabschnitt hinzu:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Instantiierung des Moduls "Aux_program_loader "**
  Fügen Sie diesen Python-Code in Ihren "instantiate"-Abschnitt ein:

```
########################################
# **** BIBLIOTHEKEN INSTANZIIEREN **** #
########################################

AUX_PRGM = Aux_program_loader()
```

### 12.8.8 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
```

**Instanziieren des Moduls Keylookup** 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 der Handler-Datei, unter der *initialisierten Funktion* verwenden Sie diese allgemeine Syntax, um **Tastenbindungen** zu erstellen:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Hier fügen wir eine Tastenbindung für `F10`, `F11` und `F12` hinzu:

```
########################################
# Spezielle Funktionen, die von QtVCP aufgerufen werden
########################################

# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    KEYBIND.add_call('Key_F10','on_keycall_F10',None)
    KEYBIND.add_call('Key_F11','on_keycall_override',10)
    KEYBIND.add_call('Key_F12','on_keycall_override',20)
```

Und dann müssen wir **die Funktionen hinzufügen, die aufgerufen werden**.
Fügen Sie in der Handler-Datei unter dem Abschnitt `KEY BINDING CALLS` Folgendes hinzu:

```
####################
# KEY BINDING CALLS #
####################

def on_keycall_F12(self,event,state,shift,cntrl,value):
    if state:
        print('F12 pressed')

def on_keycall_override(self,event,state,shift,cntrl,value):
    if state:
        print('value = {}'.format(value))
```

#### 12.8.8.2 Key Defines

Here is a list of recognized key words. Use the quoted text.
Letter keys use *Key_* with the upper or lower letter added.
e.g., *Key_a* and *Key_A*.

```
keys = {
    Qt.Key_Escape: "Key_Escape",
    Qt.Key_Tab: "Key_Tab",
    Qt.Key_Backtab: "Key_Backtab",
    Qt.Key_Backspace: "Key_Backspace",
    Qt.Key_Return: "Key_Return",
    Qt.Key_Enter: "Key_Enter",
    Qt.Key_Insert: "Key_Insert",
    Qt.Key_Delete: "Key_Delete",
    Qt.Key_Pause: "Key_Pause",
    Qt.Key_Print: "Key_Print",
    Qt.Key_SysReq: "Key_SysReq",
    Qt.Key_Clear: "Key_Clear",
    Qt.Key_Home: "Key_Home",
    Qt.Key_End: "Key_End",
    Qt.Key_Left: "Key_Left",
    Qt.Key_Up: "Key_Up",
    Qt.Key_Right: "Key_Right",
    Qt.Key_Down: "Key_Down",
    Qt.Key_PageUp: "Key_PageUp",
    Qt.Key_PageDown: "Key_PageDown",
    Qt.Key_Shift: "Key_Shift",
    Qt.Key_Control: "Key_Control",
    Qt.Key_Meta: "Key_Meta",
    # Qt.Key_Alt: "Key_Alt",
    Qt.Key_AltGr: "Key_AltGr",
    Qt.Key_CapsLock: "Key_CapsLock",
    Qt.Key_NumLock: "Key_NumLock",
    Qt.Key_ScrollLock: "Key_ScrollLock",
    Qt.Key_F1: "Key_F1",
    Qt.Key_F2: "Key_F2",
    Qt.Key_F3: "Key_F3",
    Qt.Key_F4: "Key_F4",
    Qt.Key_F5: "Key_F5",
    Qt.Key_F6: "Key_F6",
    Qt.Key_F7: "Key_F7",
    Qt.Key_F8: "Key_F8",
    Qt.Key_F9: "Key_F9",
    Qt.Key_F10: "Key_F10",
    Qt.Key_F11: "Key_F11",
    Qt.Key_F12: "Key_F12",
    Qt.Key_F13: "Key_F13",
    Qt.Key_F14: "Key_F14",
    Qt.Key_F15: "Key_F15",
    Qt.Key_F16: "Key_F16",
    Qt.Key_F17: "Key_F17",
    Qt.Key_F18: "Key_F18",
    Qt.Key_F19: "Key_F19",
    Qt.Key_F20: "Key_F20",
    Qt.Key_F21: "Key_F21",
    Qt.Key_F22: "Key_F22",
    Qt.Key_F23: "Key_F23",
    Qt.Key_F24: "Key_F24",
    Qt.Key_F25: "Key_F25",
    Qt.Key_F26: "Key_F26",
    Qt.Key_F27: "Key_F27",
    Qt.Key_F28: "Key_F28",
    Qt.Key_F29: "Key_F29",
    Qt.Key_F30: "Key_F30",
    Qt.Key_F31: "Key_F31",
    Qt.Key_F32: "Key_F32",
```

```
    Qt.Key_F33: "Key_F33",
    Qt.Key_F34: "Key_F34",
    Qt.Key_F35: "Key_F35",
    Qt.Key_Super_L: "Key_Super_L",
    Qt.Key_Super_R: "Key_Super_R",
    Qt.Key_Menu: "Key_Menu",
    Qt.Key_Hyper_L: "Key_HYPER_L",
    Qt.Key_Hyper_R: "Key_Hyper_R",
    Qt.Key_Help: "Key_Help",
    Qt.Key_Direction_L: "Key_Direction_L",
    Qt.Key_Direction_R: "Key_Direction_R",
    Qt.Key_Space: "Key_Space",
    Qt.Key_Any: "Key_Any",
    Qt.Key_Exclam: "Key_Exclam",
    Qt.Key_QuoteDbl: "Key_QuoteDdl",
    Qt.Key_NumberSign: "Key_NumberSign",
    Qt.Key_Dollar: "Key_Dollar",
    Qt.Key_Percent: "Key_Percent",
    Qt.Key_Ampersand: "Key_Ampersand",
    Qt.Key_Apostrophe: "Key_Apostrophe",
    Qt.Key_ParenLeft: "Key_ParenLeft",
    Qt.Key_ParenRight: "Key_ParenRight",
    Qt.Key_Asterisk: "Key_Asterisk",
    Qt.Key_Plus: "Key_Plus",
    Qt.Key_Comma: "Key_Comma",
    Qt.Key_Minus: "Key_Minus",
    Qt.Key_Period: "Key_Period",
    Qt.Key_Slash: "Key_Slash",
    Qt.Key_0: "Key_0",
    Qt.Key_1: "Key_1",
    Qt.Key_2: "Key_2",
    Qt.Key_3: "Key_3",
    Qt.Key_4: "Key_4",
    Qt.Key_5: "Key_5",
    Qt.Key_6: "Key_6",
    Qt.Key_7: "Key_7",
    Qt.Key_8: "Key_8",
    Qt.Key_9: "Key_9",
    Qt.Key_Colon: "Key_Colon",
    Qt.Key_Semicolon: "Key_Semicolon",
    Qt.Key_Less: "Key_Less",
    Qt.Key_Equal: "Key_Equal",
    Qt.Key_Greater: "Key_Greater",
    Qt.Key_Question: "Key_Question",
    Qt.Key_At: "Key_At",
    Qt.Key_BracketLeft: "Key_BracketLeft",
    Qt.Key_Backslash: "Key_Backslash",
    Qt.Key_BracketRight: "Key_BracketRight",
    Qt.Key_AsciiCircum: "Key_AsciiCircum",
    Qt.Key_Underscore: "Key_Underscore",
    Qt.Key_QuoteLeft: "Key_QuoteLeft",
    Qt.Key_BraceLeft: "Key_BraceLeft",
    Qt.Key_Bar: "Key_Bar",
    Qt.Key_BraceRight: "Key_BraceRight",
    Qt.Key_AsciiTilde: "Key_AsciiTilde",

}
```

## 12.8.9 Messages

Das Modul **Messages** dient zur **Anzeige von Pop-up-Dialogmeldungen auf dem Bildschirm**.

These messages are:

- *definiert in der INI-Datei unter der Überschrift `[DISPLAY]`*, und
- *gesteuert durch 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**
: Gibt an, ob es sich um Folgendes handelt: **Statusmeldung** - angezeigt in der *Statusleiste und im Benachrichtigungsdialog*.
Erfordert keinen Benutzereingriff. **OK-Meldung** - *die den Benutzer auffordert, auf OK zu klicken, um den Dialog zu schließen*.
OK-Meldungen haben *zwei HAL-Pins*:

  - Ein HAL-Pin, um den Dialog zu starten, und
  - Einen, um anzuzeigen, dass auf eine Antwort gewartet wird. **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.

By default it will send `STATUS` messages for `focus_overlay` and alert sound.

### 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

**`Notify`** module is used to **send messages that are integrated into the desktop**.

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

**Preferences** module allows one to **load and save preference data permanently to storage media**.

Das Widget ScreenOptions kann das Einstellungssystem automatisch einrichten.

QtVCP searches for the ScreenOptions widget first and, if found, calls **_pref_init()**.
This will *create the preferences object* and return it to QtVCP to pass to all the widgets and add it to the window object attributes.
In this case the preferences object would be accessible from the handler file's initialized_ method as **self.w.PREFS_**. 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()
```

  Die Funktion **`_register_messages()`** verbindet den Audioplayer mit der `STATUS`-Bibliothek, so dass Klänge mit dem `STATUS`-Meldungssystem abgespielt werden können.

#### 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
- *zugewiesene Aktionen/Untermenüs in der Handler-Datei*.

#### 12.8.14.1 Aktionen

**`estop` , `power` , `load` , `reload` , `gcode_properties` , `run` , `pause` , `abort` , `block_delete` , `optional_stop`**

Schaltet die Anzeige der Dimensionen um.

#### 12.8.14.2 Untermenüs

**recent_submenu , home_submenu , unhome_submenu , zero_systems_submenu , grid_size_submenu**

Menu to set graphic grid size

#### 12.8.14.3 Anwendung

Hier ist der typische Code, der zu den entsprechenden Abschnitten der *Handler-Datei* hinzuzufügen ist:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

#########################################
**** Bibliotheken instanziieren Abschnitt **** #
#########################################

TOOLBAR = ToolBarActions()
```

#### 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*.

Beispiel aus dem mill_xyz-Beispiel:

```
toolshape = CylinderZ(0)
toolshape = Color([1, .5, .5, .5], [toolshape])
tool = Collection([
    Translate([HalTranslate([tooltip], None, "motion.tooloffset.z", 0, 0, - ↩
        MODEL_SCALING)], 0, 0, 0),
    HalToolCylinder(toolshape)
])
```

**`Track`**
**Move and rotate an object to point from one `capture()` 'd coordinate system to another**.

Base object to *hold coordinates for primitive shapes*.

**`CylinderX, CylinderY, CylinderZ`**
**Build a cylinder on the X, Y or Z axis** by giving *endpoint* (X, Y, or Z) and *radii* coordinates.

**`Sphere`**
**Baue eine Kugel** aus den Koordinaten *center* und *radius*.

**`TriangleXY, TriangleXZ, TriangleYZ`**
**Build a triangle** in the *specified plane* by giving the *corners Z coordinates* for each side.

**`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`**
**Build a box centered in X and Y, and running from Z=0**, by specifying the *width in X and Y*, and running up or down to the specified *height in Z*.

**`Capture`**
**Capture current transformation matrix of a collection**.

---

**Anmerkung**
This *transforms from the current coordinate system to the viewport system*, NOT to the world system.

---

**`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:

```
########################################
# 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):
    machine = MILL.Window()
    self.w.mainLayout.addWidget(machine)
```

#### 12.8.15.4 Mehr zum Thema

Weitere Informationen über die Erstellung einer benutzerdefinierten Maschinensimulation finden Sie im Kapitel zu Qt Vismach.

## 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, also see: https://sa-cnc.com/linuxcnc-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**.

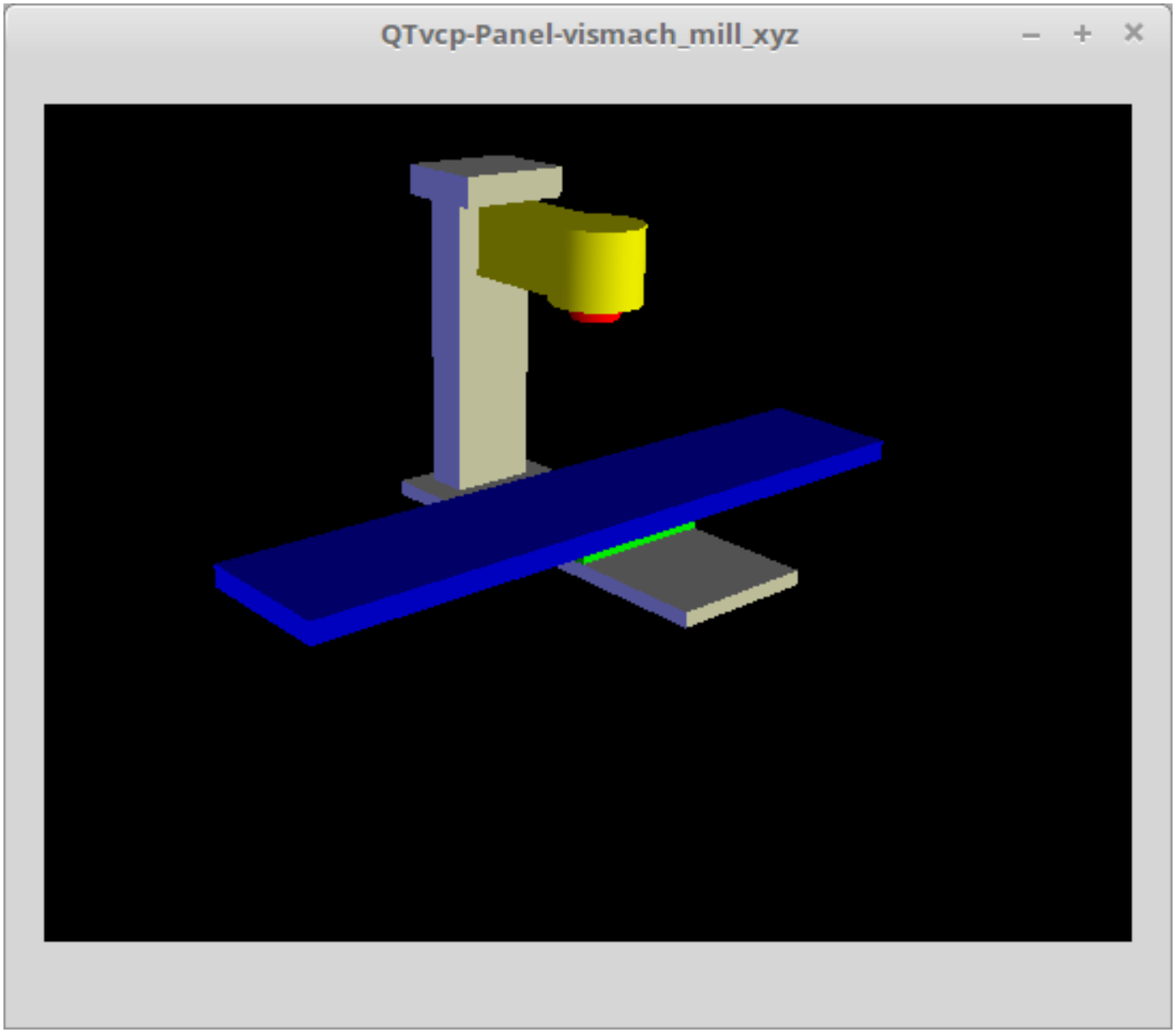



Abbildung 12.107: QtVismach 3D Viewport

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 the smaller branch, the 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, the 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, the *lowest parts must exist* first *before those can be grouped with others into an assembly*. So you *build upwards* from lowest point in tree and assemble them together.

The same is applicable for any design of machine: look at the machine arm example and you will see that it starts with the tip and adds to the larger part of the arm, then it finally groups with the base.

### 12.9.3 Start the script

It is useful for testing to include the `#!/usr/bin/env python3` shebang line to _allow the file to be executed directly from the command line.

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()".

Both functions can take one of two named arguments, either a *filename* or *data*:

```
part = AsciiSTL(filename="path/to/file.stl")
part = AsciiSTL(data="solid part1 facet normal ...")
part = AsciiOBJ(filename="path/to/file.obj")
part = AsciiOBJ(data="v 0.123 0.234 0.345 1.0 ...")
```

- STL model parts are added to the Vismach space in the *same locations as they were created in the STL or OBJ space*, i.e. ideally with a rotational point at their origin.

---

**Anmerkung**
It is much easier to move while building if the origin of the model is at a rotational pivot point.

---

#### 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

Parts may need to be moved in the Vismach space to assemble the model. The origin does not move - Translate() and Rotate() move the Collection as you add parts, relative to a stationary origin. 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 [degrees] 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 four functions: `HalTranslate`, `HalRotate`, `HalToolCylinder` and `HalToolTriangle`.

*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**
A *collection or part*.
It can be pre-created earlier in the script, or could be created at this point if preferred, e.g.,

```
'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 ( = 1 cm/1 inch = 1 cm /2.54 cm ) as the scale.

#### 12.9.7.2 HalRotate

**part = HalRotate([part], hal_comp, hal_pin, angle_scale, x, y, z)**
This command is similar in its operation to `HalTranslate`, except that it is typically necessary to move the part to the origin first to define the axis.

**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.

#### 12.9.7.3 *HalToolCylinder*

**tool = HalToolCylinder()**
Make a cylinder to represent a cylindrical mill tool, based on the tool table and current loaded tool.

```
tool = HalToolCylinder()
toolshape = Color([1, .5, .5, .5],[tool])

# or more compact:
toolshape = Color([1, .5, .5, .5], [HalToolCylinder()])
```

#### 12.9.7.4 *HalToolTriangle*

**tool = HalToolTriangle()**
Make a triangle to represent a triangular lathe tool, based on the tool table and current loaded tool.

```
tool = HalToolTriangle()
toolshape = Color([1, 1, 0, 1],[tool])

# or more compact:
toolshape = Color([1, 1, 0, 1],[HalToolTriangle()])
```

### 12.9.8 Zusammenbau des Modells

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])
```

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])
```

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()**
Erzeugt ein *Heads-up-Display* in der Vismach-GUI, um Elemente wie Achsenpositionen, Titel oder Meldungen anzuzeigen.

```
myhud = Hud()
myhud.show(”Mill_XYZ”)‘
```

**myhud = HalHud()**
A more advanced version of the Hud that allows HAL pins to be displayed:

```
myhud = HalHud()
myhud.set_background_color(0,.1,.2,0)
myhud.show_top(”Mill_XYZ”)
myhud.show_top(”------------”)
myhud.add_pin(’axis-x: ’,”{:10.4f}”,”axis.x.pos-cmd”)
myhud.add_pin(’axis-y: ’,”{:10.4f}”,”axis.y.pos-cmd”)
myhud.add_pin(’axis-z: ’,”{:10.4f}”,”axis.z.pos-cmd”)
myhud.show(”-------------”)
```

**part = Capture()**
Damit wird die aktuelle Position im Modell festgelegt.

**main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)**
This is the command that makes it all happen, creates the display, etc. if invoked directly from Python. Usually this file is imported by QtVCP and the `window()` object is instantiated and embedded into another screen.

**_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 Tips

Create an axes origin marker to be able to see parts relative to it, for construction purposes. You can remove it when you are done.

```
# build axis origin markers
X = CylinderX(-500,1,500,1)
X = Color([1, 0, 0, 1], [X])
Y = CylinderY(-500,1,500,1)
Y = Color([0, 1, 0, 1], [Y])
Z = CylinderZ(-500,1,500,1)
Z = Color([0, 0, 1, 1], [Z])
origin = Collection([X,Y,Z])
```

Add it to the Window class Collection so it is never moved from the origin.

```
v.model = Collection([origin, model, world])
```

Start from the cutting tip and work your way back. Add each collection to the model at the origin and run the script to confirm the location, then rotate/translate and run the script to confirm again.

### 12.9.11 Grundstruktur eines `QtVismach`-Skripts

```
# 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)
```

### 12.9.12 Builtin Vismach Sample Panels

QtVCP builtin Vismach Panels

## 12.10 QtVCP: Building Custom Widgets

### 12.10.1 Übersicht

Die Erstellung von benutzerdefinierten Widgets ermöglicht es, **den Qt Designer-Editor zu verwenden, um ein benutzerdefiniertes Widget zu platzieren**, *anstatt dies manuell in einer Handler-Datei zu tun*.

Ein nützliches benutzerdefiniertes Widget wäre eine großartige Möglichkeit, zu LinuxCNC beizutragen.

#### 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.

Alle diese Widgets können *mit Qt Designer Editor* platziert werden - so dass man *das Ergebnis* sehen kann, bevor man das Panel in LinuxCNC lädt.

#### 12.10.1.2 Qt Designer

**Qt Designer** is a *WYSIWYG (What You See is What You Get) editor for placing PyQt widgets*.

Die ursprüngliche Absicht war es, die grafischen Widgets für Programme zu erstellen.
Wir nutzen es, um **Bildschirme und Panels für LinuxCNC** zu bauen.

In Qt Designer, auf der linken Seite des Editors, finden Sie **drei Kategorien von LinuxCNC Widgets**:

- *HAL only widgets*.
- *LinuxCNC-Controller-Widgets*.
- *Dialog-Widgets*.

Damit Qt Designer benutzerdefinierte Widgets zu seinem Editor *hinzufügen* kann, muss ein **Plugin** zum richtigen Ordner hinzugefügt werden.

#### 12.10.1.3 Initialization Process

QtVCP macht *extra setup* für **Widgets, die Spezialisierungen** (Unterklassen) **von `_HALWidgetBase`** sind, auch bekannt als ”HAL-ified” Widgets.

This includes:

- Injecting *important variables*,
- Aufruf einer *extra Setup-Funktion*
- Aufruf einer *schließenden Aufräumfunktion* beim Herunterfahren.

Diese Funktionen werden nicht aufgerufen, wenn der Qt Designer Editor die Widgets anzeigt.

Wenn QtVCP einen Bildschirm aus der *.ui*-Datei erstellt:

1. Es wird nach allen HAL-ifizierten Widgets gesucht.
2. Er findet das Widget `ScreenOptions`, um Informationen zu sammeln, die es in die anderen Widgets einspeisen muss
3. Er instanziiert jedes Widget und ruft, wenn es ein HAL-ifiziertes Widget ist, die Funktion `hal_init()` auf.
   **`hal_init()`** ist in der Basisklasse definiert und sie:
   a. Fügt Variablen wie die Einstellungsdatei zu jedem HAL-ifizierten Widget hinzu.
   b. Ruft `+_hal_init()+` für das Widget auf.
   `+_hal_init()+` erlaubt es dem Widget-Designer, Einstellungen vorzunehmen, die den Zugriff auf zusätzliche Variablen erfordern.

Hier finden Sie eine Beschreibung der zusätzlichen Variablen, die in ”HAL-ifizierte” Widgets eingefügt werden:

**`self.HAL_GCOMP`**
The *HAL component instance*

**`self.HAL_NAME`**
This *widget’s name* as a string

**`self.QT_OBJECT_`**
Diese *Objektinstanz des Widgets*

**`self.QTVCP_INSTANCE_`**
Die *übergeordnete Ebene* des Bildschirms

**`self.PATHS_`**
Die Instanz der *QtVCP-Pfadbibliothek*

**`self.PREFS_`**
Die *optionale Instanz der Einstellungsdatei*

**`self.SETTINGS_`**
Die `Qsettings` *Objektinstanz*

#### 12.10.1.4 Aufräum-Prozess

Wenn QtVCP geschlossen wird, ruft es die **`+_hal_cleanup()+`** Funktion *auf allen HAL-ifizierten Widgets* auf.

Die Basisklasse erstellt eine leere Funktion `+_hal_cleanup()+`, die in der Unterklasse des benutzerdefinierten Widgets neu definiert werden kann.

Damit können Sie z. B. Präferenzen aufzeichnen usw.

Diese Funktion wird nicht aufgerufen, wenn der Qt Designer Editor die Widgets anzeigt.

### 12.10.2 Custom HAL Widgets

HAL-Widgets sind die einfachsten, um ein Beispiel zu zeigen.
Die Datei `qtvcp/widgets/simple_widgets.py` enthält viele reine HAL-Widgets.

Sehen wir uns einen Ausschnitt aus `simple_widgets.py` an:

**Im Abschnitt "Imports"** Hier importieren wir die Bibliotheken, die unsere Widget-Klasse benötigt.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5 import QtWidgets                         # (1)
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase        # (2)
import hal                                          # (3)
```

In diesem Fall benötigen wir Zugang zu:

(1) Die QtWidgets-Bibliothek von PyQt,

(2) 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` erlaubt es,:

- *Objekte in einem Raster zu positionieren*.
- *Alle darin enthaltenen Widgets zu aktivieren/deaktivieren*, basierend auf einem **HAL-Pin-Status**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):    # (1)
    def __init__(self, parent = None):                          # (2)
        super(GridLayout, self).__init__(parent)                # (3)
```

Zeile für Zeile:

(1) Dies definiert den *Klassennamen* und die *Bibliotheken, von denen sie erbt*.
Diese Klasse, genannt `Lcnc_GridLayout`, erbt die Funktionen von `QWidget` und `+_HalSensitiveBase+`.
`+_HalSensitiveBase+` ist eine *Unterklasse* von **`+_HalWidgetBase+`**, der *Basisklasse der meisten QtVCP-Widgets*, d.h. sie hat alle Funktionen von `+_HalWidgetBase+` plus die Funktionen von `+_HalSensitiveBase+`.
Sie fügt die Funktion hinzu, um das Widget basierend auf einem HAL-Eingangs-BIT-Pin zu aktivieren oder zu deaktivieren.

(2) Dies ist die Funktion, die *aufgerufen wird, wenn das Widget zum ersten Mal erstellt wird* (d.h. instanziiert wird) - das ist ziemlich standardmäßig.

(3) Diese Funktion initialisiert die **Super-Klassen** unseres Widgets.
Super" bedeutet einfach die *vererbten Basisklassen*, d.h. `QWidget` und `_HalSensitiveBase`.
Außer dem Namen des Widgets wird sich nichts ändern.

### 12.10.3 Benutzerdefinierte Controller-Widgets mit STATUS

Widget, die mit LinuxCNC-Controller interagieren sind nur ein wenig komplizierter und sie erfordern einige *extra Bibliotheken*.

In diesem reduzierten Beispiel werden wir Eigenschaften hinzufügen, die im Qt Designer geändert werden können.

Dieses LED-Anzeige-Widget reagiert auf wählbare Zustände der LinuxCNC-Steuerung.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5.QtCore import pyqtProperty
from qtvcp.widgets.led_widget import LED
from qtvcp.core import Status

###########################################
# **** instantiate libraries section **** #
###########################################
STATUS = Status()

########################################
# custom widget class definition
########################################
class StateLED(LED):
    def __init__(self, parent=None):
        super(StateLED, self).__init__(parent)
        self.has_hal_pins = False
        self.setState(False)
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False

    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))

    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)

    ######################################################################
    # Qt Designer properties setter/getters/resetters
    ######################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
```

```
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False

    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

#### 12.10.3.1 In The *Imports* Section

Hier importieren wir die Bibliotheken, die unsere Widget-Klasse benötigt.

```
#!/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`, damit wir mit dem Qt Designer-Editor interagieren können,

❷ `LED`, weil unser benutzerdefiniertes Widget darauf basiert,

❸ `Status` weil es uns Statusmeldungen von LinuxCNC gibt.

#### 12.10.3.2 Im Abschnitt *Bibliotheken instanziieren*

Here we create the `Status` library instance:

```
###########################################
# **** instantiate libraries section **** #
###########################################
STATUS = Status()
```

Typischerweise haben wir die Bibliothek *außerhalb der Widget-Klasse* instanziiert, so dass der Verweis auf sie **global** ist - was bedeutet, dass Sie nicht `self.` davor verwenden müssen.

Konventionell verwenden wir *große* Buchstaben im Namen für globale Verweise.

#### 12.10.3.3 Im Abschnitt ”Benutzerdefinierte Widget-Klassendefinition”

Dies ist das Herzstück unseres benutzerdefinierten Widgets.

**Klassendefinition und Instanzinitialisierungsfunktion**

```
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
```

❶ Definiert den **Namen** *unseres benutzerdefinierten Widgets* und von welcher anderen *Klasse es erbt*.
In diesem Fall erben wir `LED` - ein QtVCP-Widget, das eine Statusleuchte darstellt.

❷ Typisch für die meisten Widgets - wird aufgerufen, wenn das Widget zum ersten Mal erstellt wird.

❸ Typisch für die meisten Widgets - ruft den Initialisierungscode des übergeordneten (Super-)Widgets auf.
Dann setzen wir einige Attribute:

❹ Geerbt von `Lcnc_Led` - wir setzen ihn hier, damit kein HAL-Pin erstellt wird.

❺ Geerbt von `Lcnc_led` - wir setzen es, um sicherzustellen, dass die LED aus ist.

The other attributes are for the selectable options of our widget.

**Die HAL-Initialisierungsfunktion des Widgets**

```
    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))
```

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.

Wir haben zwei Zustände, zwischen denen wir wählen können: `is_estopped` oder `is_on`.
Je nachdem, welcher Zustand aktiv ist, wird unser Widget mit den entsprechenden STATUS-Meldungen verbunden.

**`+_hal_init()+`** 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 dieser Funktion haben Sie Zugang zu einigen zusätzlichen Informationen (obwohl wir sie in diesem Beispiel nicht verwenden):

**`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_**
Die *übergeordnete Ebene* des Bildschirms

**self.PATHS_**
Die _Instanz der Pfadbibliothek von QtVCP

**self.PREFS_**
die *Instanz einer optionalen Präferenzdatei*

**self.SETTINGS_**
das `Qsettings` *Objekt*

Wir könnten diese Informationen nutzen, um HAL-Pins zu erstellen oder Bildpfade nachzuschlagen usw.

```
STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Schauen wir uns diese Zeile genauer an:

- `STATUS` ist ein sehr verbreitetes Thema beim Erstellen von Widgets.
  `STATUS` verwendet das `GObject`-Nachrichtensystem, um Nachrichten an Widgets zu senden, die sich dafür registrieren.
  Diese Zeile ist der Registrierungsprozess.
- `state-estop` is the message we wish to listen for and act on. There are many messages available.
- lambda w:self._flip_state(True)` ist das, was passiert, wenn die Nachricht abgefangen wird.
  Die Lambda-Funktion akzeptiert die Widget-Instanz (`w`), die GObject ihr sendet und ruft dann die Funktion `self._flip_state(True)` auf.
  Lambda wurde verwendet, um das (`w`) Objekt vor dem Aufruf der Funktion `self._flip_state` zu strippen.
  Es erlaubte auch die Verwendung, um `self._flip_state()` den Zustand True zu senden.

```
    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

Dies ist die Funktion, die den Zustand der LED tatsächlich umschaltet.
Sie wird aufgerufen, wenn die entsprechende STATUS-Meldung akzeptiert wird.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

The function called looks like this:

```
    def _set_feedrate_text(self, widget, data):
```

in dem das Widget und alle Daten von der Funktion akzeptiert werden müssen.

```
    ########################################################################
    # Qt Designer properties setter/getters/resetters
    ########################################################################

    # invert status
    def set_invert_state(self, data):
```

```
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False
```

Dies ist *die Art und Weise, wie Qt Designer die Attribute des Widgets setzt*.
Dies *kann auch direkt* im Widget aufgerufen werden.

```
    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

Dies ist die **Registrierung von Eigenschaften in Qt Designer**.

Der **Eigenschaftsname**:

- ist der *Text, der in Qt Designer* verwendet wird,
- *kann nicht mit den Attributen identisch sein*, die sie darstellen.

Diese Eigenschaften werden im Qt Designer in der Reihenfolge angezeigt, in der sie hier erscheinen.

### 12.10.4 Benutzerdefinierte Controller-Widgets mit Aktionen

Hier ist ein Beispiel für ein Widget, welches das Benutzerreferenzsystem festlegt.

It changes:

- den Zustand der Maschinensteuerung mit Hilfe der Bibliothek `ACTION`,
- ob die Schaltfläche mit Hilfe der `STATUS`-Bibliothek angeklickt werden kann oder nicht.

```
import os
import hal

from PyQt5.QtWidgets import QWidget, QToolButton, QMenu, QAction
from PyQt5.QtCore import Qt, QEvent, pyqtProperty, QBasicTimer, pyqtSignal
from PyQt5.QtGui import QIcon
```

```
from qtvcp.widgets.widget_baseclass import _HalWidgetBase
from qtvcp.widgets.dialog_widget import EntryDialog
from qtvcp.core import Status, Action, Info

# Instanziieren Sie die Bibliotheken mit einer globalen Referenz.
# STATUS gibt uns Statusmeldungen von LinuxCNC
# INFO enthält INI-Details
# ACTION gibt Befehle an LinuxCNC
STATUS = Status()
INFO = Info()
ACTION = Aktion()

class SystemToolButton(QToolButton, _HalWidgetBase):
    def __init__(self, parent=None):
        super(SystemToolButton, self).__init__(parent)
        self._joint = 0
        self._last = 0
        self._block_signal = False
        self._auto_label_flag = True
        SettingMenu = QMenu()
        for system in('G54', 'G55', 'G56', 'G57', 'G58', 'G59', 'G59.1', 'G59.2', 'G59.3'):

            Button = QAction(QIcon('exit24.png'), system, self)
            Button.triggered.connect(self[system.replace('.','_')])
            SettingMenu.addAction(Button)

        self.setMenu(SettingMenu)
        self.dialog = EntryDialog()

    def _hal_init(self):
        if not self.text() == '':
            self._auto_label_flag = False
        def homed_on_test():
            return (STATUS.machine_is_on()
                    and (STATUS.is_all_homed() or INFO.NO_HOME_REQUIRED))

        STATUS.connect('state-off', lambda w: self.setEnabled(False))
        STATUS.connect('state-estop', lambda w: self.setEnabled(False))
        STATUS.connect('interp-idle', lambda w: self.setEnabled(homed_on_test()))
        STATUS.connect('interp-run', lambda w: self.setEnabled(False))
        STATUS.connect('all-homed', lambda w: self.setEnabled(True))
        STATUS.connect('not-all-homed', lambda w, data: self.setEnabled(False))
        STATUS.connect('interp-paused', lambda w: self.setEnabled(True))
        STATUS.connect('user-system-changed', self._set_user_system_text)

    def G54(self):
        ACTION.SET_USER_SYSTEM('54')

    def G55(self):
        ACTION.SET_USER_SYSTEM('55')

    def G56(self):
        ACTION.SET_USER_SYSTEM('56')

    def G57(self):
        ACTION.SET_USER_SYSTEM('57')

    def G58(self):
        ACTION.SET_USER_SYSTEM('58')

    def G59(self):
```

```
        ACTION.SET_USER_SYSTEM('59')

    def G59_1(self):
        ACTION.SET_USER_SYSTEM('59.1')

    def G59_2(self):
        ACTION.SET_USER_SYSTEM('59.2')

    def G59_3(self):
        ACTION.SET_USER_SYSTEM('59.3')

    def _set_user_system_text(self, w, data):
        convert = { 1:"G54", 2:"G55", 3:"G56", 4:"G57", 5:"G58", 6:"G59", 7:"G59.1", 8:"G59 ↩
            .2", 9:"G59.3"}
        if self._auto_label_flag:
            self.setText(convert[int(data)])

    def ChangeState(self, joint):
        if int(joint) != self._joint:
            self._block_signal = True
            self.setChecked(False)
            self._block_signal = False
            self.hal_pin.set(False)

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

### 12.10.5 Stylesheet-Eigenschaftsänderungen auf der Grundlage von Ereignissen

Es ist möglich, **Widgets bei Ereignisänderungen neu zu gestalten**. Sie müssen das Widget explizit _"polieren", damit PyQt den Stil wiederherstellt.
Dies ist eine relativ teure Funktion und sollte daher sparsam verwendet werden.

Dieses Beispiel setzt eine "isHomed"-Eigenschaft basierend auf dem "homed"-Zustand von LinuxCNC und verwendet diese wiederum, um Stylesheet-Eigenschaften zu ändern:

**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))

    # ishomed-Eigenschaft aktualisieren
    # Widget polieren, damit Stylesheet die Änderung der Eigenschaft sieht
    # einige Stylesheets färben den Text bei home/unhome
```

```
    def _home_status_polish(self, d, state):
        if self.joint_number = d:
            self.setProperty('isHomed', state)
            self.style().unpolish(self)
            self.style().polish(self)

    # Qproperty getter and setter
    def getisHomed(self):
        return self._isHomed
    def setisHomed(self, data):
        self._isHomed = data

    # Qproperty
    isHomed = QtCore.pyqtProperty(bool, getisHomed, setisHomed)
```

Hier ist ein Beispiel-Stylesheet zum Ändern der Textfarbe basierend auf dem Home-Status.

In diesem Fall wird jedes Widget, das auf dem obigen HomeLabel-Widget basiert, die Textfarbe ändern.
Normalerweise würden Sie bestimmte Widgets mit `HomeLabel #specific_widget_name[homed=true]` auswählen:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

### 12.10.6 Verwenden von Stylesheets zum Ändern benutzerdefinierter Widget-Eigenschaften

```
class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        alternateFont0 = self.font

    # Qproperty getter and setter
    def getFont0(self):
        return self.aleternateFont0
    def setFont0(self, value):
        self.alternateFont0(value)
    # Qproperty
    styleFont0 = pyqtProperty(QFont, getFont0, setFont0)
```

Beispiel-Stylesheet, das eine benutzerdefinierte Widget-Eigenschaft festlegt.

```
Label{
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
}
```

### 12.10.7 Widget-Plugins

Wir müssen unsere benutzerdefinierten Widgets *registrieren*, damit Qt Designer sie verwenden kann.

Hier sind ein paar typische Beispiele.
Sie müssten zu `qtvcp/plugins/` hinzugefügt werden.
Dann müsste `qtvcp/plugins/qtvcp_plugin.py` angepasst werden, um sie zu *importieren*.

#### 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 Erstellen eines Plugins mit einem Dialogfeld "MenuEntry"

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)
```

The function

- 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-Funktionen

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
- eine benutzerdefinierte Funktionsaktion.

Der `objectName` der Symbolleistenschaltfläche wird verwendet, um die Schaltfläche bei der Konfiguration zu identifizieren - *deskriptive Namen helfen*.

Klicken Sie mit der rechten Maustaste auf das Menü des Aktionseditors und wählen Sie Bearbeiten. Editieren Sie den Objektnamen, den Text und den Schaltflächentyp für eine geeignete Aktion.

In this example the:

- Der Name des Untermenüs (engl. submenu) muss `menuRecent` lauten,
- Aktionsnamen müssen `actionAbout`, `actionQuit`, `actionMyFunction` sein

**Lädt die Bibliothek toolbar_actions in die IMPORT SECTION**

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

**Instanziieren Sie das Modul ToolBarActions im Abschnitt INSTANTIATE LIBRARY SECTION**

```
TOOLBAR = ToolBarActions()
```

**Konfigurieren Sie Untermenüs und Aktionen im Abschnitt "BESONDERE FUNKTIONEN"**
Under the `def initialized__(self)` 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*.

**Lädt die Bibliothek Qhal in die IMPORT SECTION**

```
from qtvcp.core import Qhal
```

This is to allow access to **QtVCP's HAL component**.

**Instanziieren von Qhal in der INSTANTIATE LIBRARY SECTION**

```
QHAL = Qhal()
```

**Hinzufügen einer Funktion, die aufgerufen wird, wenn sich der Zustand des Pins ändert**
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))
```

**Definieren der Funktion, die durch Pin-Statusänderung im GENERAL FUNCTIONS SECTION aufgerufen wird**

```
#####################################
# allgemeine (engl. general) functions #
#####################################

def cycleStart(self, state):
    if state:
        tab = self.w.mainTab.currentWidget()
        if  tab in( self.w.tab_auto,  self.w.tab_graphics):
            ACTION.RUN(line=0)
        elif tab == self.w.tab_files:
                self.w.filemanager.load()
        elif tab == self.w.tab_mdi:
            self.w.mditouchy.run_command()
```

Diese Funktion geht davon aus, dass es ein Tab-Widget mit dem Namen `mainTab` gibt, das Tabs mit den Namen `tab_auto`, `tab_graphics`, `tab_filemanager` und `tab_mdi` hat.

Auf diese Weise funktioniert der Zyklusstart-Button je nach angezeigter Registerkarte unterschiedlich.

Dies ist vereinfacht - *Zustandskontrolle und Fehlerverfolgung könnten hilfreich sein*.

### 12.11.8 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.

Der **`STATUS`** Befehl stellt sicher, dass der Schieberegler sich anpasst, wenn LinuxCNC die aktuelle maximale Geschwindigkeit ändert.

**`valueChanged.connect()`** *ruft eine Funktion auf, wenn der Schieberegler bewegt wird*.

Fügen Sie im Qt Designer ein **QSlider**-Widget mit dem Namen `mvPercent` hinzu und fügen Sie dann den folgenden Code in die Handler-Datei ein:

```
############################
# SPECIAL FUNCTIONS SECTION #
############################

def initialized__(self):
    self.w.mvPercent.setMaximum(100)
    STATUS.connect('max-velocity-override-changed', \
        lambda w, data: self.w.mvPercent.setValue( \
            (data / INFO.MAX_TRAJ_VELOCITY)*100 \
            )
        )
    self.w.mvPercent.valueChanged.connect(self.setMVPercentValue)

####################
# GENERAL FUNCTIONS #
####################

def setMVPercentValue(self, value):
    ACTION.SET_MAX_VELOCITY_RATE(INFO.MAX_TRAJ_VELOCITY * (value/100.0))
```

### 12.11.9 Kontinuierlichen Jog ein- und ausschalten

Generally selecting continuous jogging is a momentary button, that requires you to select the previous jog increment after.

Wir werden eine Schaltfläche erstellen, die zwischen kontinuierlichem Joggen und der bereits ausgewählten Schrittweite umschaltet.

Im Qt-Designer:

- Hinzufügen eines `ActionButton` ohne Aktion
- Nennen Sie ihn "btn_toggle_continuous".
- Setzen Sie die Eigenschaft `AbstractButton checkable` auf `True`.
- Setzen Sie die Eigenschaften `ActionButton incr_imperial_number` und `incr_mm_number` auf `0`.
- 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 information.

Dann fügen Sie diesen Code-Schnipsel in die Handler-Datei unter der Funktion `initialized__` ein:

```
# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    STATUS.connect('jogincrement-changed', \
        lambda w, d, t: self.record_jog_incr(d,t) \
        )
    # ein Standardinkrement festlegen, zu dem zurückgeschaltet werden soll
    self.L_incr = 0,01
    self.L_text = "0.01in"
```

In the `GENERAL FUNCTIONS SECTION` add:

```
####################
# GENERAL FUNCTIONS #
####################

# Wenn es nicht kontinuierlich ist, zeichnen Sie das letzte Jog-Inkrement auf.
# und deaktiviere die Schaltfläche ”kontinuierlich”.
def record_jog_incr(self,d, t):
    if d != 0:
        self.L_incr = d
        self.L_text = t
        self.w.btn_toggle_continuous.safecheck(False)
```

Im Abschnitt `CALLBACKS FROM STATUS SECTION` hinzufügen:

```
###########################
# CALLBACKS VOM FORMULAR #
###########################

def toggle_continuous_clicked(self, state):
    if state:
        # set continuous (call the actionbutton’s function)
        self.w.btn_toggle_continuous.incr_action()
    else:
        # reset previously recorded increment
        ACTION.SET_JOG_INCR(self.L_incr, self.L_text)
```

### 12.11.10 Class Patch The File Manager Widget

**Anmerkung**
Class Patching (Monkey Patching) ist ein wenig wie *schwarze Magie* - also verwenden Sie es *nur bei Bedarf*.

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. *Behalten Sie einen Verweis auf die ursprüngliche Funktion* (1), damit wir sie weiterhin aufrufen können
2. Leiten Sie die Klasse um, damit sie stattdessen unsere benutzerdefinierte Funktion (2) in der Handler-Datei aufruft.

```
########################################
# 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. Schreiben Sie eine benutzerdefinierte Funktion, um das Original zu ersetzen:
   Diese Funktion muss die **gleiche Signatur wie die Originalfunktion** haben.
   In diesem Beispiel werden wir immer noch die ursprüngliche Funktion aufrufen, indem wir den Verweis auf sie verwenden, den wir zuvor aufgezeichnet haben.
   Sie *erfordert, dass das erste Argument die Widget-Instanz ist*, was in diesem Fall `self.w.filemanager` ist (der Name, der im Qt Designer-Editor angegeben wurde).

```
####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

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 manchen Situationen ist es nur möglich, Widgets mit Python-Code hinzuzufügen, anstatt den Qt Designer-Editor zu verwenden.

Wenn QtVCP-Widgets programmatisch hinzugefügt werden, müssen manchmal *zusätzliche Schritte* unternommen werden.

Here we are going to add a spindle speed indicator bar and up-to-speed LED to a tab widget corner. Qt Designer does not support adding corner widgets to tabs but PyQt does.

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** For the spindle speed indicator we need to know the current spindle speed. For this we *register* with `STATUS` to:

- *Catch* the `actual-spindle-speed-changed` *signal*
- *Aufruf* der _Funktion `self.update_spindle()`

```
########################
# **** INITIALIZE **** #
########################
# Widgets allow 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** We need to *make sure the Qt Designer widgets are already built* before we try to add to them. For this, we add a call to `self.make_corner_widgets()` function to build our extra widgets at the right time, i.e. under the `initialized__()` function:

```
########################################
# 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 let's code the function to build the widgets and add them in the tab widget. We are assuming there is a tab widget built with Designer called *rightTab*.

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_**
Die *übergeordnete Ebene* des Bildschirms

**self.PATHS_**
Die _Instanz der Pfadbibliothek von QtVCP

**self.PREFS_**
die *Instanz einer optionalen Präferenzdatei*

**self.SETTINGS_**
das `Qsettings` *Objekt*

5 Initialisiert einen PyQt5 `QProgressBar`.

6 Setzt den maximalen Bereich des Fortschrittsbalkens auf den in der `INI` angegebenen Maximalwert.

7 Wir erstellen ein QWidget
Da man nur ein Widget in die Tab-Ecke einfügen kann und wir dort zwei haben wollen, müssen wir beide in einen **Container** einfügen.

8 ein QHBoxLayout zum QWidget hinzufügen.

9, 10 Dann fügen wir unseren QProgress-Balken und die LED in das Layout ein.

11 Schließlich fügen wir das QWidget (mit unserem QProgress-Balken und der LED darin) in die Ecke des Tab-Widgets ein.

**Erstellen Sie die Überwachungsfunktion STATUS** Jetzt erstellen wir die Funktion, die den `QProgressBar` aktualisiert, wenn `STATUS` die Spindeldrehzahl aktualisiert:

```
####################
# callbacks von STATUS #
####################
def update_spindle(self, data):
    self.w.rpm_bar.setInvertedAppearance(bool(data<0))       # 1
    self.w.rpm_bar.setFormat('{0:d} RPM'.format(int(data)))  # 2
    self.w.rpm_bar.setValue(abs(data))                       # 3
```

1 In diesem Fall haben wir uns für die Darstellung von links nach rechts oder von rechts nach links entschieden, je nachdem, ob wir uns im oder gegen den Uhrzeigersinn drehen.

2 Dadurch wird die Schrift in der Leiste formatiert.

3 This sets the length of the colored bar.

### 12.11.12 Objekte periodisch aktualisieren/auslesen

Manchmal muss man **ein Widget aktualisieren oder regelmäßig einen Wert auslesen**, der von den normalen Bibliotheken nicht abgedeckt wird.

Here we update an LED based on a watched HAL pin every 100 ms.

Wir nehmen an, dass in der Qt Designer UI Datei eine LED mit dem Namen `led` vorhanden ist.

**Laden Sie die Qhal-Bibliothek für den Zugriff auf die HAL-Komponente von QtVCP** In the `IMPORT SECTION` add:

```
from qtvcp.core import Qhal
```

**Instanziiere 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:

**Registrierung einer Funktion, die im Zeitraum CYCLE_TIME aufgerufen wird** This is usually every 100 ms.

```
########################
# **** 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 100 ms)
    STATUS.connect('periodic', lambda w: self.update_periodic())
```

**Erstellen Sie die benutzerdefinierte Funktion, die periodisch aufgerufen werden soll**

```
####################
# general functions #
####################
def update_periodic(self):
    data = QHAL.getvalue('spindle.0.is-oriented')
    self.w.led.setState(data)
```

## 12.11.13 External Control With ZMQ

*QtVCP kann automatisch* ein **ZMQ-Messaging** *einrichten, um Remote-Nachrichten von externen Programmen zu senden und/oder zu empfangen.*

It uses ZMQ's **publish/subscribe messaging pattern**.

Wie immer sollte man die **Sicherheit** im Auge behalten, bevor man Programmen eine Schnittstelle für Nachrichtenübermittlung einräumt.

### 12.11.13.1 ZMQ Messages Reading

Manchmal möchte man den **Bildschirm mit einem separaten Programm steuern**.

**Aktivieren des Empfangs von ZMQ-Nachrichten** Im Widget `ScreenOptions` können Sie die Eigenschaft **`use_receive_zmq_option`** auswählen.
Sie können diese Eigenschaft auch direkt *in der Handler-Datei* einstellen, wie in diesem Beispiel.

Wir nehmen an, dass das `ScreenOptions`-Widget in Qt Designer `screen_options` genannt wird:

```
########################
# **** INITIALIZE **** #
########################
# widgets 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** Let's add a specific function for testing. You will need to run LinuxCNC from a terminal to see the printed text.

```
####################
# general functions #
####################
def test_zmq_function(self, arg1, arg2):
    print('zmq_test_function called: ', arg1, arg2)
```

**Erstellen eines externen Programms, das ZMQ-Nachrichten sendet, die einen Funktionsaufruf auslösen** 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 Legen Sie die **aufzurufende Funktion** und die **zu sendenden Argumente** auf diese Funktion fest.

You will need to know the *signature* of the function you wish to call. Also note that the *message is converted to a JSON object*. This is because ZMQ sends byte messages not Python objects. `json` converts Python objects to bytes and will be converted back when received.

#### 12.11.13.2 ZMQ Messages Writing

Sie können auch **mit einem externen Programm vom Bildschirm aus kommunizieren**.

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.

Wir nehmen an, dass das `ScreenOptions`-Widget in Qt Designer `screen_options` genannt wird:

**Senden von ZMQ-Nachrichten aktivieren**

```
########################
# **** INITIALIZE **** #
########################
# widgets erlaubt den Zugriff auf Widgets aus den QtVCP-Dateien.
# zu diesem Zeitpunkt sind die Widgets und Hal-Pins noch nicht instanziiert
def __init__(self, halcomp,widgets,paths):
    # directly select ZMQ message sending
    self.w.screen_options.setProperty('use_send_zmq_option',True)
```

Dies ermöglicht das Senden von Nachrichten an ein separates Programm.
Welche Nachricht gesendet wird, hängt davon ab, was das externe Programm erwartet.

**Erstellen einer Funktion zum Senden von ZMQ-Nachrichten** Lassen Sie uns eine spezielle Funktion zum Testen hinzufügen.
Sie müssen LinuxCNC von einem Terminal aus starten, um den gedruckten Text zu sehen.
Außerdem muss etwas hinzugefügt werden, um diese Funktion aufzurufen, wie z.B. ein Tastenklick.

```
####################
# 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)
```

**Verwenden oder erstellen Sie ein Programm, das ZMQ-Nachrichten empfangen kann** Hier ist ein Beispielprogramm, das die Nachricht empfängt und auf dem Terminal ausgibt:

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Definition des Sockets mit Hilfe des "Context".
sock = context.socket(zmq.SUB)

# Definieren des Abonnements und der zu akzeptierenden Nachrichten ohne Einschränkung der ↩
    Themen.
topic = "" # alle Themen
sock.setsockopt(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**
Nicht alle Bildschirme haben eine Statusleiste.

---

#### Status bar usage example

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` ist in Sekunden und wir nehmen an, dass ”statusbar” der Name des Qt Designer Widgets ist.

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`**
: Anzeigen der Nachricht nur für eine kurze Zeit.

**`STATUS.OPERATOR_ERROR`**, **`STATUS.OPERATOR_TEXT`**, **`STATUS.NML_ERROR`**, **`STATUS.NML_TEXT`**

#### Beispiel für das Versenden einer Bedienermeldung:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

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 Fokusänderungen abfangen

Der Fokus wird verwendet, um Benutzeraktionen wie Tastatureingaben auf das richtige Widget zu lenken.

#### Aktuell fokussiertes Widget abrufen

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

#### Fokussiertes Widget erhalten, wenn sich der Fokus ändert

```
# zu diesem Zeitpunkt:
# sind die Widgets instanziiert.
# die HAL-Pins sind gebaut, aber HAL ist nicht bereit
def initialized__(self):
    QtWidgets.QApplication.instance().event_filter.focusIn.connect(self.focusInChanged)
```

```
#####################################
# allgemeine (engl. general) functions #
#####################################

def focusInChanged(self, widget):
    if isinstance(widget.parent(),type(self.w.gcode_editor.editor)):
        print('G-code Editor')
    elif isinstance(widget,type(self.w.gcodegraphics)):
        print('G-code Display')
    elif isinstance(widget.parent(),type(self.w.mdihistory) ):
        print('MDI History')
```

Beachten Sie, dass wir manchmal mit `widget`, manchmal mit `widget.parent()` vergleichen.

Das liegt daran, dass *einige QtVCP-Widgets aus mehreren **Sub-Widgets*** aufgebaut sind und letztere tatsächlich den Fokus erhalten; daher müssen wir **den Elternteil** dieser Sub-Widgets überprüfen.

Other times the main widget is what gets the focus, e.g., the G-code display widget can be set to accept the focus. 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.

### 12.11.16 Read Command Line Load Time Options

Some panels need information at load time for setup/options. QtVCP covers this requirement with *-o* options.
The *-o* argument is good for a few, relatively short options, that can be added to the loading command line.
For more involved information, reading an INI or preference file is probably a better idea.

Multiple *-o* options can be used on the command line so you must decode them.
*self.w.USEROPTIONS_* will hold any found *-o* options as a list of strings. You must parse and define what is accepted and what to do with it.

**Example code to get -o options for camera number and window size**

```
    def initialized__(self):

        # set a default camera number
        number = 0

        # check if there are any -o options at all
        if self.w.USEROPTIONS_ is not None:

            # if in debug mode print the options to the terminal
            LOG.debug('cam_align user options: {}'.format(self.w.USEROPTIONS_))

            # go through the found options one by one
            for num, i in enumerate(self.w.USEROPTIONS_):

                # if the -o option has 'size=' in it, assume it's width and height of  ↩
                    window
                # override the default width and height of the window
                if 'size=' in self.w.USEROPTIONS_[num]:
                    try:
                        strg = self.w.USEROPTIONS_[num].strip('size=')
                        arg = strg.split(',')
                        self.w.resize(int(arg[0]),int(arg[1]))
                    except Exception as e:
                        print('Error with cam_align size setting:',self.w.USEROPTIONS_[num ↩
                            ])
```

```
            #  # if the -o option has 'camnumber=' in it, assume it's the camera number ↩
                 to use
            elif 'camnumber=' in self.w.USEROPTIONS_[num]:
                try:
                    number = int(self.w.USEROPTIONS_[num].strip('camnumber='))
                except Exception as e:
                    print('Error with cam_align camera selection - not a number - using ↩
                        0')

        # set the camera number either as default or if -o option changed the 'number'  ↩
            variable, to that number.
        self.w.camview._camNum = number
```

## 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** and
- 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 vom Startup bis zum Herunterfahren

**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 Herunterfahren (engl. shutdown)

Wenn QtVCP schließlich zum Herunterfahren aufgefordert wird:

1. Sie ruft die Abschaltfunktionen in der Handler-Datei auf,
2. STATUS monitoring is shut down
3. HAL component gets killed

## 12.12.4 Path Information

When QtVCP loads it collects paths information.

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 information.

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_`**
Die *übergeordnete Ebene* des Bildschirms

**`self.PATHS_`**
Die _Instanz der Pfadbibliothek von QtVCP

**`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 non-realtime 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 its 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]
  # The double bracket section(s) define the group(s) of radio buttons.
  # The group name must be unique and is case sensitive.
  # Groups output is controlled by what button is active not directly by keycode.
  # DEFAULT references a button in the group by name and is case sensitive.
  [[group1_name]]
    KEY = NONE
    OUTPUT = FLOAT
    DEFAULT = small
    # The triple bracket sections define the buttons in this group.
    # button names must be unique and are case sensitive.
    # There must be at least two buttons in a group.
    #
    # This button, named 'small'is controller by the row 0 column 1 key.
    # It will cause the group output to be .0001 when it is pressed.
    # It has no output of its own, but has a status
    # pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[small]]]
      KEY = R0C1
      GROUP = group1_name
      GROUP_OUTPUT = .0001
      OUTPUT = NONE
      STATUS_PIN = True
      TRUE_STATE = TRUE
      FALSE_STATE = FALSE
      TRUE_COMMAND = NONE, NONE
      FALSE_COMMAND = NONE, NONE
      DEFAULT = false
    # This button, named 'large' is controller by the row 0 column 2 key.
    # It will cause the group output to be 1000 when it is pressed.
    # It has a S32 output of its own, will be 20 on true and 0 on false.
    # It also has a status pin which will follow its current state.
    # since this button is in a group, DEFAULT has no bearing.
    # since OUTPUT in not 'COMMAND' _COMMAND entries are ignored.
    [[[large]]]
      KEY = R0C2
```

```
      GROUP = group1_name
      GROUP_OUTPUT = 1000
      OUTPUT = 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]
  # Each button name inside double brackets, must be unique and is case sensitive.
  # This button, named 'tool_change'is controller by the row 2 column 5 key.
  # It has a BIT output, will output 1 on true state and 0 on false state.
  # It also has a status pin which will follow its current state.
  # DEFAULT sets this to true when first initialized.
  # The _COMMAND are not used since OUTPUT is not set to COMMAND but validation will
  # add the lines regardless
  [[tool_change]]
    KEY = R2C5
    OUTPUT = 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]
  # Each button name inside double brackets, must be unique and is case sensitive.
  # This button, named 'spindle_rev'is controller by the row 2 column 3 key.
  # It has a COMMAND output, so will use TRUE_COMMAND and FALSE_COMMAND.
  # It also has a status pin which will follow its current state.
  # COMMANDs will have a command name and then any required arguments
  # This TRUE_COMMAND calls an internal command to start the spindle in reverse at 200 rpm
  # If the spindle is already started, it will increase the rpm.
  # DEFAULT is not used with Momentary buttons.
  # The _STATE are not used since OUTPUT is set to COMMAND but validation will
  # add the lines regardless
  [[spindle_rev]]
    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 can send ZMQ based messages on button presses.
In this way panelui can interact will other programs such as QtVCP screens.

```
[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

A special file can be used to add custom python code that will be available as commands. panelui_handler.py must be written in python and be placed in the configuration folder. If panelui finds a file there it will add its function calls to the available commands. Here is an example of a handler file that adds two functions - hello_world and 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 The LinuxCNC Python module

Diese Dokumentation beschreibt das `linuxcnc` Python-Modul, das eine Python-API für die Kommunikation mit LinuxCNC bereithält.

### 13.2.1 Einführung

Benutzeroberflächen steuern LinuxCNC-Aktivitäten durch Senden von NML-Nachrichten an die LinuxCNC-Task-Controller, und überwachen die Ergebnisse durch die Beobachtung der LinuxCNC-Status-Struktur, sowie des Fehlerberichterstattung Kanals.

Der programmatische Zugriff auf NML erfolgt über eine C++-API; die wichtigsten Teile der NML-Schnittstelle zu LinuxCNC sind jedoch auch für Python-Programme über das Modul `linuxcnc` verfügbar.

Neben der NML-Schnittstelle zu den Befehls-, Status- und Fehlerkanälen enthält das Modul `linuxcnc` auch:

- Unterstützung für das Lesen von Werten aus INI-Dateien

### 13.2.2 Verwendungsmuster für die LinuxCNC NML-Schnittstelle

Das allgemeine Muster für die Verwendung von `linuxcnc` ist in etwa wie folgt:

- 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.

### 13.2.3 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.

#### 13.2.3.1 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)**
Removed since version 2.9 use axis_mask.bit_count() to get the number of axes configured

**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**
*(returns tuple of integers)* - Active G-codes for each modal group.
The integer values reflect the nominal G-code numbers multiplied by 10. (Examples: 10 = *G1*, 430 = *G43*, 923 = *G92.3*)

**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.

#### 13.2.3.2 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.

#### 13.2.3.3 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.

#### 13.2.3.4 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)* - spindle speed value, rpm, > 0: clockwise, < 0: counterclockwise.
With G96 active this reflects the maximum speed set by the optional G96 *D*-word or, if the *D*-word was missing, the default values +/-1e30

### 13.2.4 Vorbereitung des Sendens von Befehlen

Einige Befehle können immer gesendet werden, unabhängig von Modus und Zustand; zum Beispiel kann die Methode `linuxcnc.command.abort()` immer aufgerufen werden.

Andere Befehle können nur in einem geeigneten Zustand gesendet werden, und diese Tests können etwas knifflig sein. Zum Beispiel kann ein MDI-Befehl nur gesendet werden, wenn:

- NOTAUS (engl. ESTOP) nicht ausgelöst wurde und
- die Maschine eingeschaltet ist und
- Referenzfahrten an den Achsen durchgeführt wurden
- der Interpreter nicht läuft und
- der Modus ist auf ”MDIModus” eingestellt ist

Ein geeigneter Test vor dem Senden eines MDI-Befehls durch `linuxcnc.command.mdi()` könnte 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”)
```

### 13.2.5 Senden von Befehlen über *linuxcnc.command*

Initialisieren Sie vor dem Senden eines Befehls einen Befehlskanal wie folgt:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# Anwendungsbeispiele für einige der unten aufgeführten Befehle:
c.abort()

c.auto(linuxcnc.AUTO_RUN, program_start_line)
c.auto(linuxcnc.AUTO_STEP)
c.auto(linuxcnc.AUTO_PAUSE)
c.auto(linuxcnc.AUTO_RESUME)

c.brake(linuxcnc.BRAKE_ENGAGE)
c.brake(linuxcnc.BRAKE_RELEASE)

c.flood(linuxcnc.FLOOD_ON)
c.flood(linuxcnc.FLOOD_OFF)

c.home(2)

c.jog(linuxcnc.JOG_STOP,        jjogmode, joint_num_or_axis_index)
c.jog(linuxcnc.JOG_CONTINUOUS,  jjogmode, joint_num_or_axis_index, velocity)
c.jog(linuxcnc.JOG_INCREMENT,   jjogmode, joint_num_or_axis_index, velocity, increment)

c.load_tool_table()

c.maxvel(200.0)

c.mdi("G0 X10 Y20 Z30")

c.mist(linuxcnc.MIST_ON)
c.mist(linuxcnc.MIST_OFF)

c.mode(linuxcnc.MODE_MDI)
c.mode(linuxcnc.MODE_AUTO)
c.mode(linuxcnc.MODE_MANUAL)

c.override_limits()

c.program_open("foo.ngc")
c.reset_interpreter()

c.tool_offset(toolno, z_offset,  x_offset, diameter, frontangle, backangle, orientation)
```

#### 13.2.5.1 `linuxcnc.command` Attribute

**`serial`**
: die Seriennummer des aktuellen Befehls

#### 13.2.5.2 `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)**
set the feedrate override, 1.0 = 100%.

**flood(int)**
Ein-/ausschalten der Kühlmittel-Flut.

**Syntax**
flood(command)
flood(linuxcnc.FLOOD_ON)
flood(linuxcnc.FLOOD_OFF)

**Konstanten**
FLOOD_ON
FLOOD_OFF

**home(int)**
ein bestimmtes Gelenk zu Referenzpunkt fahren.

**jog(command-constant, bool, int[, float[, float]])**

**Syntax**
jog(command, jjogmode, joint_num_or_axis_index, velocity[, distance])
jog(linuxcnc.JOG_STOP, jjogmode, joint_num_or_axis_index)
jog(linuxcnc.JOG_CONTINUOUS, jjogmode, joint_num_or_axis_index, velocity)
jog(linuxcnc.JOG_INCREMENT, jjogmode, joint_num_or_axis_index, velocity, distance)

**Befehlskonstanten**
linuxcnc.JOG_STOP
linuxcnc.JOG_CONTINUOUS
linuxcnc.JOG_INCREMENT

**jjogmode**

**True**
Einzelne Gelenkbewegungen anfordern (erfordert teleop_enable(0))

**False**
Abfrage der kartesischen Achsenkoordinaten (erfordert teleop_enable(1))

**joint_num_or_axis_index**

**Für gemeinsames Joggen (Jogmode=1)**
joint_number

**Für kartesisches Joggen der Achse (jjogmode=0)**
Nullbasierter Index der Achsenkoordinate in Bezug auf die bekannten Koordinatenbuchstaben XYZABCUVW (x=>0,y=>1,z=>2,a=>3,b=>4,c=>5,u=>6,v=>7,w=>8)

**load_tool_table()**
die Werkzeugtabelle neu laden.

**maxvel(float)**
set maximum velocity

**mdi(string)**
einen MDI-Befehl senden. Maximal 254 Zeichen.

**mist(int)**
Kühlnebel ein-/ausschalten.

**Syntax**
mist(command)
mist(linuxcnc.MIST_ON)
mist(linuxcnc.MIST_OFF)

**Konstanten**
MIST_ON
MIST_OFF

**mode(int)**
Modus einstellen (MODE_MDI, MODE_MANUAL, MODE_AUTO).

**override_limits()**
Flag für die Überschreitung der Achsengrenzen setzen.

**program_open(string)**
eine NGC-Datei öffnen.

**rapidrate()**
set rapid override factor

**reset_interpreter()**
den RS274NGC-Interpreter zurücksetzen

**set_adaptive_feed(int)**
adaptiven Vorschub (engl. adaptive feed)-Flag setzen

**set_analog_output(int, float)**
Wert auf analogen Ausgangs-Pin legen

**set_block_delete(int)**
setze Block-löschen-Markierung (engl. flag)

**set_digital_output(int, int)**
Digitalen Ausgangspin auf Wert setzen

**set_feed_hold(int)**
Vorschubfreigabe ein/auseinstellen

**set_feed_override(int)**
Vorschub-Neufestsetzung ein/ausschalten

**set_max_limit(int, float)**
Einstellung der maximalen Positionsgrenze für eine bestimmte Achse

**set_min_limit()**
Legen Sie die minimale Positionsgrenze für eine bestimmte Achse fest

**set_optional_stop(int)**
optionalen Stopp ein-/ausschalten

**set_spindle_override(int [, int])**
Spindel-Neufestsetzung (engl. override) einstellen aktiviert. Standardmäßig ist die Spindel 0 eingestellt.

**spindle(direction: int, speed: float=0, spindle: int=0, wait_for_speed: int=0)**

- Direction (engl. für Richtung): [SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECREASE, oder SPINDLE_CONSTANT]
- Speed: Drehzahl in U/min (engl. RPM), Standardwert ist 0.
- Spindle: Spindelnummer für den Befehl ist standardmäßig 0.
- Wait_for_speed: wenn auf 1 gesetzt warten Bewegung auf die Geschwindigkeit bevor sie fortgesetzt werden, die Standardeinstellung verlang dies nicht.

**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.

### 13.2.6 Lesen des Fehlerkanals

Um Fehlermeldungen zu behandeln, stellen Sie eine Verbindung zum Fehlerkanal her und pollen (durch Aufruf von poll() ) Sie ihn regelmäßig.

Beachten Sie, dass der NML-Kanal für Fehlermeldungen über eine Warteschlange verfügt (anders als die Befehls- und Statuskanäle), was bedeutet, dass der erste Empfänger einer Fehlermeldung diese aus der Warteschlange löscht; ob ein anderer Empfänger einer Fehlermeldung (z. B. Axis) die Meldung sieht, hängt vom Timing ab. Es wird empfohlen, nur eine Fehlerkanal-Leser-Task in einem Setup zu haben.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
e = linuxcnc.error_channel()

error = e.poll()

if error:
    kind, text = error
    if kind in (linuxcnc.NML_ERROR, linuxcnc.OPERATOR_ERROR):
        typus = "error"
    else:
        typus = "info"
    print(typus, text)
```

### 13.2.7 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
```

### 13.2.8 Der Typ `linuxcnc.positionlogger`

Einige Verwendungshinweise können von *src/emc/usr_intf/gremlin/gremlin.py* entnommen werden.

#### 13.2.8.1 Members

**`npts`**
Anzahl der Punkte.

#### 13.2.8.2 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

## 13.3 The HAL Python module

This documentation describes the `hal` python module, which provides a Python API for creating and accessing HAL pins and signals.

### 13.3.1 Grundlegende Verwendung (engl. basic usage)

```
#!/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()
```

### 13.3.2 Funktionen

**component**
: + The component itself is created by a call to the constructor `hal.component`. The arguments are the HAL component name and (optionally) the prefix used for pin and parameter names. If the prefix is not specified, the component name is used.
.Beispiel

```
h = hal.component("passthrough")
```

**newpin**
: +
Create new pin.
Arguments: pin name suffix, pin type, and pin direction. For parameters, the arguments are: parameter name suffix, parameter type, and parameter direction.
.Beispiel:

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

**ready**
: Sagt dem HAL-System, dass die Komponente initialisiert ist. Sperrt das Hinzufügen von Pins.

**unready**
: Ermöglicht es einer Komponente, Pins hinzuzufügen, nachdem *ready()* aufgerufen wurde. Man sollte *ready()* auf der Komponente danach aufrufen.

**component_exists**
: Existiert die angegebene Komponente zu diesem Zeitpunkt.

**Beispiel**

```
hal.component_exists("testpanel")
```

**component_is_ready**
: Ist die angegebene Komponente zu diesem Zeitpunkt bereit.

**Beispiel**

```
hal.component_is_ready(”testpanel”)
```

**get_msg_level**
Abfrage der aktuellen Echtzeit-Nachrichten (engl. kurz msg)-Stufe.

**set_msg_level**
Legen Sie die aktuelle Echtzeit-MSG-Stufe fest. Wird zum Debuggen von Informationen verwendet.

**connect**
Verbinden Sie einen Pin mit einem Signal.

**Beispiel**

```
hal.connect(”pinname”,”signal_name”)
```

**disconnect**
Trennen Sie einen Pin von einem Signal.

**Beispiel**

```
hal.disconnect(”Pinname”)
```

**get_value**
Lesen Sie einen Pin, einen Parameter oder ein Signal direkt.

**Beispiel**

```
value = hal.get_value(”iocontrol.0.emc-enable-in”)
```

**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')
```

**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')
```

**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')
```

**new_sig**
Erstellt ein neues Signal des angegebenen Typs.

**Beispiel**

```
hal.new_sig(”signalname”,hal.HAL_BIT)
```

**pin_has_writer**
Ist der angegebene Pin mit einem treibenden Pin verbunden.
Gibt True oder False zurück.

```
h.in.pin_has_writer()
```

**get_name**
Ermittelt den Namen des HAL-Objekts.
Rückgabe einer Zeichenkette.

```
h.in.get_name()
```

**get_type**
Ermittelt den Typ des HAL-Objekts’.
Gibt eine ganze Zahl zurück.

```
h.in.get_type()
```

**get_dir**
Ermittelt den Richtungstyp des HAL-Objekts.
Gibt eine ganze Zahl zurück.

```
h.in.get_dir()
```

**get**
Ermittelt den Wert des HAL-Objekts.

```
h.in.get()
```

**set**
Setzt den Wert des HAL-Objekts.

```
h.out.set(10)
```

**is_pin**
Ist das Objekt ein Pin oder Parameter?
Gibt True oder False zurück.

```
h.in.is_pin()
```

**sampler_base**
TODO

**stream_base**
TODO

**stream**
TODO

**set_p**
Einstellen eines Pin-Wertes eines beliebigen Pins im HAL-System.

**Beispiel**

```
hal.set_p(”pinname”,”10”)
```

## 13.4 GStat Python Module

### 13.4.1 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 checks LinuxCNC’s status every 100 ms and if there are differences from the last check, it will send a callback message to all the connected programs with the current 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):
```

### 13.4.2 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.

#### 13.4.2.1 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
```

This would be loaded with `loadusr python PATH-TO-FILE/FILENAME.py` or if you need to wait for the pins to be made before continuing:
`loadusr python -Wn metric_status PATH-TO-FILE/FILENAME.py`
The pins would be: `metric_status.g20` and `metric_status.g21`.

#### 13.4.2.2 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('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.builder.get_object('state_label').set_label("State: %s" % (text))

    def update_estate_label(self,text):
        self.builder.get_object('e_state_label').set_label("E State: %s" % (text))

    def update_interp_label(self,text):
        self.builder.get_object('interp_state_label').set_label("Interpreter State: %s" % ( ↩
            text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

#### 13.4.2.3 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('MACHINE ON'))
        GSTAT.connect("state-off",lambda w: self.update_state_label('MACHINE OFF'))

        GSTAT.connect("interp-paused",lambda w: self.update_interp_label('Paused'))
        GSTAT.connect("interp-run",lambda w: self.update_interp_label('Run'))
        GSTAT.connect("interp-idle",lambda w: self.update_interp_label('Idle'))

    def update_state_label(self,text):
        self.w.state_label.setText("State: %s" % (text))
```

```
    def update_estate_label(self,text):
        self.w.e_state_label.setText("E State: %s" % (text))

    def update_interp_label(self,text):
        self.winterp_state_label.setText("Interpreter State: %s" % (text))

def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,widgets,paths)]
```

### 13.4.3 Nachrichten

**periodic (engl. für periodisch)**
*(returns nothing)* - sent every 100 ms.

**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)* - Sent when LinuxCNC's interpreter is idle.

**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)* - Sent every 100 ms.

**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**
*(returns Python dict)* - Sent when G-code is loaded.
The dict contains the following keys:

- `name` (*string*): Name of the loaded file
- `size` (*string*): Size in bytes and lines
- `g0` (*string*): Total rapid distance
- `g1` (*string*): Total feed distance
- `run` (*string*): Estimated program run time
- `toollist` (*list*): List of used tools
- `x` (*string*): X extents (bounds) [1]
- `x_zero_rxy` (*string*): X extents without rotation around z (bounds) [1]
- `y` (*string*): Y extents (bounds) [1]
- `y_zero_rxy` (*string*): Y extents without rotation around z (bounds) [1]
- `z` (*string*): Z extents (bounds) [1]
- `z_zero_rxy` (*string*): Z extents without rotation around z (bounds) [1]
- `machine_unit_sys` (*string*): Machine units (*Metric* or *Imperial*)
- `gcode_units` (*string*): Units in G-code file (*mm* or *in*)

---

**Anmerkung**

1. See the images extends non-rotated and extends rotated 30 degrees for a better understanding.

---

**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:

- Kennung: *UNIQUE_ID_CODE*
  Das Dikt hat normalerweise mehr Schlüsselnamenpaar - es hängt von der Implementierung ab.

**erzwungene Aktualisierung (engl. forced update)**
(gibt keine zurück)" - soll gesendet werden, wenn ein Objekt initialisiert oder willkürlich aktualisiert werden soll.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

**Fortschritt (engl. progress)**
*(liefert Integer, Python-Objekt)* - soll gesendet werden, um den Fortschritt eines Filterprogramms anzuzeigen.
Dies hängt von den verwendeten Widgets/Bibliotheken ab.

**following-error**
*(gibt eine Python-Liste zurück)* - gibt eine Liste aller Gelenke zurück, die aktuell einem Fehler folgen.

### 13.4.4 Funktionen

Dies sind Komfortfunktionen, die in der Programmierung häufig verwendet werden.

**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-cha`
signal.

**get_jograte()**
*(Nothing)* - x = GSTAT.get_jograte() würde die aktuelle interne Jograte von GSTAT (float) zurückgeben.

**set_jograte_angular**
*(float)* -

**get_jograte_angular**
*(None)* -

**set_jog_increment_angular**
*(float, string)* -

**get_jog_increment_angular**
*(None)* -

**set_jog_increments**
*(float, string)* -

**get_jog_increments**
*(None)* -

**ist_all_homed**
*(nothing)* - Dies gibt den aktuellen Status von all_homed (BOOL) zurück.

**machine_is_on**
*(nothing)* - Dies gibt den aktuellen Zustand der Maschine zurück (BOOL).

**estop_is_clear**
*(nichts)* - Gibt den Status von Estop zurück (BOOL)

**set_tool_touchoff**
*(Werkzeug,Achse,Wert)* - Dieser Befehl wird

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. den MDI-Befehl aufrufen: G10 L10 P[WERKZEUG] [ACHSE] [WERT],
4. warten, bis der Vorgang abgeschlossen ist,
5. G43 aufrufen,
6. warten, bis der Vorgang abgeschlossen ist,
7. zurück in den ursprünglichen Modus wechseln.

**set_axis_origin**
*(Achse,Wert)* - Dieser Befehl wird

1. den aktuellen Modus aufzeichnen,
2. Wechsel zu MDI-Modus,
3. den MDI-Befehl aufzurufen: G10 L20 P0 [ACHSE] [WERT],
4. warten, bis der Vorgang abgeschlossen ist,
5. zurück in den ursprünglichen Modus wechseln,
6. ein *Reload-Display*-Signal aussenden.

**do_jog**
*(Achsennummer, Richtung, Abstand)* - Diese Funktion bewegt eine Achse kontinuierlich oder in einem bestimmten Abstand.
Sie müssen sich im richtigen Modus befinden, um zu joggen.

**check_for_modes**
*(mode)* - Diese Funktion prüft, ob der LinuxCNC-Modus erforderlich ist.
Es gibt ein Python-Tupel zurück (Status, Modus)
-Modus wird in dem Modus eingestellt, in dem sich das System befindet
wird wie folgt festgelegt:

- false, wenn mode 0 ist
- false, wenn die Maschine beschäftigt ist
- wahr, wenn LinuxCNC im angeforderten Modus ist
- Keine, wenn eine Änderung möglich ist, aber nicht im gewünschten Modus

**get_current_mode**
*(nothing)* - gibt eine ganze Zahl zurück: den aktuellen LinuxCNC-Modus.

**set_selected_joint**
*(Integer)* - speichert intern die ausgewählte Gelenknummer.
fordert die Auswahl des Gelenks an, indem er die Meldung
Nachricht ”Gelenkauswahl geändert”.

**get_selected_joint**
*(None)* - 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)* - gibt eine Zeichenkette zurück, die den intern ausgewählten Achsenbuchstaben darstellt.

**is_man_mode**
*(None)* -

**is_mdi_mode**
*(None)* -

**is_auto_mode**
*(None)* -

**is_on_and_idle**
*(None)* -

**is_auto_running**
*(None)* -

**is_auto_paused**
*(None)* -

**is_file_loaded**
*(None)* -

**is_metric_mode**
*(None)* -

**is_spindle_on**
*(None)* -

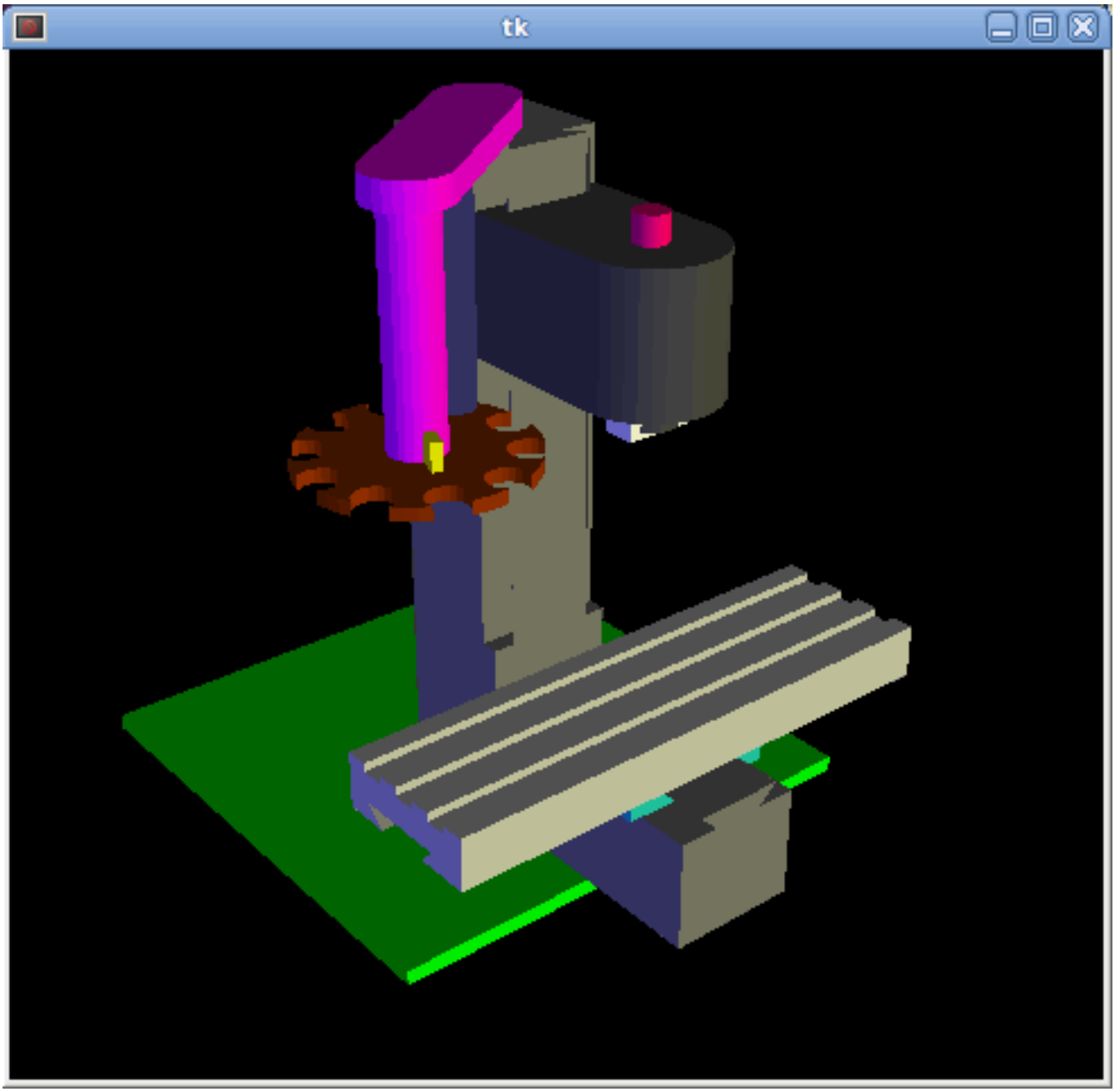
**shutdown**
*(None)* -

### 13.4.5 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.

## 13.5 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.

### 13.5.1 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
```

### 13.5.2 Erstellen der HAL-Pins.

HAL pins are created with the normal Python ”hal” library, and are not specific to Vismach. Further details can be found in the Creating Non-realtime Components in Python section. A component should be created with a name that matches the script file name and then the HAL pins are added to that component. They will be referenced by their component handle and short name when used to animate the Vismach model.

```
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.

### 13.5.3 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])
```

### 13.5.4 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.

### 13.5.5 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.

### 13.5.6 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])
```

### 13.5.7 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.

### 13.5.8 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.

Copyright © 2000-2020 LinuxCNC.org

Es wird die Erlaubnis erteilt, dieses Dokument unter den Bedingungen der GNU Free Documentation License, Version 1.1 oder einer späteren Version, die von der Free Software Foundation veröffentlicht wurde, zu kopieren, zu verbreiten und/oder zu verändern; ohne unveränderliche Abschnitte, ohne Texte auf der Vorderseite und ohne Texte auf der Rückseite des Umschlags. Eine Kopie der Lizenz ist in dem Abschnitt ”GNU Free Documentation License” enthalten.

Wenn Sie die Lizenz nicht finden, können Sie eine Kopie bei uns bestellen:

```
Free Software Foundation, Inc.
51 Franklin Street
Fifth Floor
Boston, MA 02110-1301 USA.
```

(Maßgeblich ist die englische Sprachfassung, deswegen wurde sie hier nicht übersetzt, im Fall von Verständnisproblemen siehe zur Anregung http://www.gnu.de/documents/gpl-3.0.de.html und lassen Sie sich beraten )

LINUX® ist das eingetragene Warenzeichen von Linus Torvalds in den USA und anderen Ländern. Die eingetragene Marke Linux® wird im Rahmen einer Unterlizenz von LMI, dem exklusiven Lizenznehmer von Linus Torvalds, dem Eigentümer der Marke auf weltweiter Basis, verwendet.

Das LinuxCNC-Projekt ist nicht mit Debian® verbunden. Debian_ ist ein eingetragenes Warenzeichen im Besitz von Software in the Public Interest, Inc.

Das LinuxCNC-Projekt ist nicht mit UBUNTU® verbunden. UBUNTU ist eine eingetragene Marke im Besitz von Canonical Limited.

# 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.

**AXIS(GUI)**
Eine der grafischen Benutzeroberflächen, die den Benutzern von LinuxCNC zur Verfügung stehen. Es verfügt über die moderne Verwendung von Menüs und Maustasten während der Ausführung und versteckt einige der mehr traditionellen LinuxCNC Kontrollen. Es ist die einzige Open-Source-Schnittstelle, die den gesamten Werkzeugpfad zeigt, sobald eine Datei geöffnet wird.

**GMOCCAPY (GUI)**
Eine grafische Benutzeroberfläche, die den Benutzern von LinuxCNC zur Verfügung steht. Es bietet die Verwendung und das Gefühl einer industriellen Steuerung und kann mit Touchscreen, Maus und Tastatur verwendet werden. Es unterstützt eingebettete Tabs und von HAL ausgelöste Benutzer-Nachrichten, bietet es eine Menge HAL beens mit Hardware gesteuert werden. GMOCCAPY ist in hohem Maße anpassbar.

**Umkehrspiel**
Das Spiel oder der Bewegungsverlust, 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**
Code running inside the kernel, as opposed to code running in userspace. Some realtime systems (like RTAI) run realtime code in the kernel and non-realtime code in userspace, while other realtime systems (like Preempt-RT) run both realtime and non-realtime code in userspace.

**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**
The linear and angular units used for machine configuration. These units are specified and used in the INI file. HAL pins and parameters are also generally in machine units.

**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**
General-purpose, very high-level programming language. Used in LinuxCNC for the Axis GUI, the StepConf configuration tool, and several G-code programming scripts.

**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 that is intended to meet very strict timing deadlines. On Linux, in order to meet these requirements it is necessary to install a realtime kernel such as RTAI or Preempt-RT, and build the LinuxCNC software to run in the special real-time environment. Realtime software can run in the kernel or in userspace, depending on the facilities offered by the system.

**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

**Copyright (c) 2000-2022 LinuxCNC.org**

Es wird die Erlaubnis erteilt, dieses Dokument unter den Bedingungen der GNU Free Documentation License, Version 1.1 oder einer späteren Version, die von der Free Software Foundation veröffentlicht wurde, zu kopieren, zu verbreiten und/oder zu verändern; ohne unveränderliche Abschnitte, ohne Texte auf der Vorderseite und ohne Texte auf der Rückseite des Umschlags. Eine Kopie der Lizenz ist in dem Abschnitt ”GNU Free Documentation License” enthalten.

### 16.1.2 GNU Free Documentation License

**GNU Free Documentation License Version 1.1, March 2000**

Copyright © 2000 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Es ist jedermann gestattet, wortwörtliche Kopien dieses Lizenzdokuments zu kopieren und zu verbreiten, aber es ist nicht erlaubt, es zu verändern.

**0. PREAMBEL**

Der Zweck dieser Lizenz ist es, ein Handbuch, ein Lehrbuch oder ein anderes schriftliches Dokument ”frei” im Sinne von Freiheit zu machen: jedem die effektive Freiheit zu sichern, es zu kopieren und weiterzugeben, mit oder ohne Modifikation, entweder kommerziell oder nicht-kommerziell. In zweiter Linie bewahrt diese Lizenz dem Autor und dem Herausgeber eine Möglichkeit, Anerkennung für ihre Arbeit zu erhalten, während sie nicht für die von anderen vorgenommenen Änderungen verantwortlich gemacht werden.

Diese Lizenz ist eine Art ”Copyleft”, was bedeutet, dass abgeleitete Werke des Dokuments selbst im gleichen Sinne frei sein müssen. Sie ergänzt die GNU General Public License, die eine Copyleft-Lizenz für freie Software ist.

Wir haben diese Lizenz entworfen, um sie für Handbücher für freie Software zu verwenden, weil freie Software freie Dokumentation braucht: ein freies Programm sollte mit Handbüchern geliefert werden, welche die gleichen Freiheiten bieten wie die Software. Aber diese Lizenz ist nicht auf Software-Handbücher beschränkt; sie kann für jedes textliche Werk verwendet werden, unabhängig vom Thema

oder ob es als gedrucktes Buch veröffentlicht wird. Wir empfehlen diese Lizenz in erster Linie für Werke, deren Zweck die Anleitung oder das Nachschlagen ist.

**1. ANWENDBARKEIT UND DEFINITIONEN**

Diese Lizenz gilt für jedes Handbuch oder andere Werk, das einen Hinweis des Urheberrechtsinhabers enthält, der besagt, dass es unter den Bedingungen dieser Lizenz verbreitet werden darf. Das ”Dokument”, unten, bezieht sich auf ein solches Handbuch oder Werk. Jedes Mitglied der Öffentlichkeit ist ein Lizenznehmer und wird als ”Sie” angesprochen.

Eine ”modifizierte Version” des Dokuments ist jedes Werk, welches das Dokument oder einen Teil davon enthält, entweder wortwörtlich kopiert oder mit Änderungen und/oder in eine andere Sprache übersetzt.

Ein ”sekundärer Abschnitt” ist ein benannter Anhang oder ein vorderer Abschnitt des Dokuments, der sich ausschließlich mit der Beziehung der Herausgeber oder Autoren des Dokuments zum Gesamtthema des Dokuments (oder zu verwandten Themen) befasst und nichts enthält, was direkt in dieses Gesamtthema fallen könnte. (Wenn das Dokument zum Beispiel teilweise ein Lehrbuch der Mathematik ist, darf ein sekundärer Abschnitt keine Mathematik erklären). Die Beziehung könnte eine Frage des historischen Zusammenhangs mit dem Thema oder mit verwandten Themen oder der rechtlichen, kommerziellen, philosophischen, ethischen oder politischen Position dazu sein.

Die unveränderlichen Abschnitte” sind bestimmte sekundäre Abschnitte, deren Titel in der Mitteilung, die besagt, dass das Dokument unter dieser Lizenz freigegeben ist, als die der unveränderlichen Abschnitte bezeichnet werden.

Die ”Covertexte” sind bestimmte kurze Textpassagen, die als Front-Cover-Texte oder Back-Cover-Texte in dem Hinweis aufgeführt sind, der besagt, dass das Dokument unter dieser Lizenz freigegeben ist.

Eine ”transparente” Kopie des Dokuments ist eine maschinenlesbare Kopie, die in einem Format dargestellt wird, dessen Spezifikation der Allgemeinheit zur Verfügung steht, dessen Inhalt direkt und unkompliziert mit allgemeinen Texteditoren oder (für Bilder, die aus Pixeln bestehen) mit allgemeinen Malprogrammen oder (für Zeichnungen) mit einem weit verbreiteten Zeichnungseditor betrachtet und bearbeitet werden kann, und die für die Eingabe in Textformatierer oder für die automatische Übersetzung in eine Vielzahl von Formaten geeignet ist, die für die Eingabe in Textformatierer geeignet sind. Eine Kopie, die in einem ansonsten transparenten Dateiformat erstellt wurde, dessen Markup so gestaltet wurde, dass eine nachträgliche Änderung durch Leser vereitelt oder erschwert wird, ist nicht transparent. Eine Kopie, die nicht ”Transparent” ist, wird als ”Opak” bezeichnet.

Geeignete Formate für transparente Kopien sind z. B. ASCII ohne Markup, Texinfo-Eingabeformat, LaTeX-Eingabeformat, SGML oder XML mit einer öffentlich zugänglichen DTD und standardkonformes einfaches HTML, das für die Bearbeitung durch den Menschen ausgelegt ist. Zu den undurchsichtigen Formaten gehören PostScript, PDF, proprietäre Formate, die nur von proprietären Textverarbeitungsprogrammen gelesen und bearbeitet werden können, SGML oder XML, für die eine DTD und/oder die Verarbeitungswerkzeuge nicht allgemein verfügbar sind, und das maschinell erzeugte HTML, das von einigen Textverarbeitungsprogrammen nur zu Ausgabezwecken erzeugt wird.

Die ”Titelseite” bedeutet bei einem gedruckten Buch die Titelseite selbst sowie die Folgeseiten, die benötigt werden, um das Material, das nach dieser Lizenz auf der Titelseite erscheinen soll, lesbar zu halten. Für Werke in Formaten, die kein Titelblatt als solches haben, bedeutet ”Titelblatt” den Text in der Nähe des auffälligsten Erscheinens des Werktitels, der dem Beginn des Textes vorausgeht.

**2. WORTWÖRTLICHES KOPIEREN**

Sie dürfen das Dokument in jedem beliebigen Medium kopieren und verbreiten, sei es kommerziell oder nicht kommerziell, vorausgesetzt, dass diese Lizenz, die Urheberrechtsvermerke und der Lizenzvermerk, der besagt, dass diese Lizenz für das Dokument gilt, in allen Kopien wiedergegeben werden, und dass Sie keine weiteren Bedingungen zu denen dieser Lizenz hinzufügen. Sie dürfen keine technischen Maßnahmen anwenden, um das Lesen oder weitere Kopieren der von Ihnen erstellten oder verbreiteten Kopien zu behindern oder zu kontrollieren. Sie dürfen jedoch eine Vergütung im Austausch für Kopien annehmen. Wenn Sie eine ausreichend große Anzahl von Kopien verbreiten, müssen Sie auch die Bedingungen in Abschnitt 3 einhalten.

Sie können auch Kopien unter den oben genannten Bedingungen ausleihen und öffentlich ausstellen.

**3. MASSENHAFTES KOPIEREN**

Wenn Sie mehr als 100 gedruckte Exemplare des Dokuments veröffentlichen und der Lizenzhinweis des Dokuments Umschlagtexte verlangt, müssen Sie die Exemplare in Umschläge einlegen, die deutlich und lesbar alle diese Umschlagtexte enthalten: Vorderseitentexte auf dem vorderen Umschlag und Rückseitentexte auf dem hinteren Umschlag. Auf beiden Umschlägen müssen Sie außerdem deutlich und leserlich als Verleger dieser Exemplare ausgewiesen sein. Der vordere Umschlag muss den vollständigen Titel enthalten, wobei alle Wörter des Titels gleichmäßig hervorgehoben und sichtbar sein müssen. Sie können die Umschläge zusätzlich mit anderem Material versehen. Kopien mit Änderungen, die sich auf die Umschläge beschränken, können als wortgetreue Kopien behandelt werden, solange der Titel des Dokuments erhalten bleibt und diese Bedingungen erfüllt sind.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

**4. 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 https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

**ADDENDUM**: 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 Origin

EMC (the Enhanced Machine Controller) was created by NIST , the National Institute of Standards and Technology, which is an agency of the Commerce Department of the United States government.

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.

Early considerations focused on replacing the expensive and temperamental ”real time” Windows NT system. It was proposed that a relatively new (at the time) real time extension of the Linux operating system be tried. This idea was pursued with success. Next up was the issue of the expensive intelligent motion control boards. By this time the processing power of a PC was considered great enough to directly take control of the motion routines. A quick search of available hardware resulted in the selection of a ”Servo-To-Go” interface board as the first platform for letting the PC directly control the motors. Software for trajectory planning and PID loop control was added to the existing user interface and RS274 interpreter. Matt successfully used this version to upgrade a couple of machines with dead controls and this became the EMC system that first caught the attention of the outside world. Mention of EMC on the rec.crafts.metalworking USENET newsgroup resulted in early adopters like Jon Elson building systems to take advantage of EMC.

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.

By this time interest in EMC as beginning to pick up substantially. As more and more people attempted installation of EMC, the difficulty of patching a Linux kernel with the real time extensions and of compiling the EMC code became glaringly obvious. Many attempts to document the process and write scripts were attempted, some with moderate success. The problem of matching the correct version of the patches and compilers with the selected version of Linux kept cropping up. Paul Corner came to the rescue with the BDI (brain dead install) which was a CD from which a complete working system (Linux, patches, and EMC) could be installed. The BDI approach opened the world of EMC to a much larger user community. As this community continued to grow, the EMC mailing list and code archives were moved to SourceForge and the LinuxCNC web site was established.

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

NIST published a paper describing the RS274NGC language and the abstract machining center it controls, as well as an early implementation of EMC. The paper is also available at https://linuxcnc.org/-files/RS274NGCv3.pdf .

NIST also published a paper on the history of EMC and its transition to open source. The paper is also available at https://linuxcnc.org/files/Use-of-Open-Source-Distribution-for-a-Machine-Tool-Controller.pdf

# Kapitel 18

# Index

_
.axisrc, 603
/ Block Delete, 873
[APPLICATIONS] Section, 158
[AXIS_<letter>] Sections, 162
[DISPLAY] Section, 149
[EMCIO] Section, 170
[EMCMOT] Section, 156
[EMC] Section, 149
[FILTER] Section, 153
[HALUI] Section, 158
[HAL] Section, 157
[JOINT_<num>] Sections, 163
[RS274NGC] Section, 155
[SPINDLE_<num>] Section(s), 169
[TASK] Section, 157
[TRAJ] Section, 159
0-10 Volt Spindle Speed Example, 413
3 & 4, 59
5-Axis Kinematics, 492
7 & 8, 59

**Numbers**
2
  3 & 4, 59
6
  7 & 8, 59

**A**
About LinuxCNC, 2
Achse), 1311
Achsenkonfiguration, 87
Achsenschnittstelle, 338
Acme-Schraube, 1311
addf, 209
Analog Input, 315
Analog Input Functions, 316
Analog Input Parameters, 316
Analog Input Pins, 315
Analog Output, 316
  Functions, 316
  Parameters, 316
  Pins, 316
and2, 216
ANGULAR UNITS, 161
Anzeigeeinheiten, 1312
Arc Distance Mode, 941
Automatic Login, 22
axes, 51
  primary linear, 52
  rotational, 52
  secondary linear, 52
Axis
  Configuration, 92
AXIS GUI, 579
  .axisrc, 603
  axisui pins, 605
  Jogwheel, 603
  Keyboard Shortcuts, 593
  Lathe Mode, 597
  Manual Tool Change, 596
  Manual Toolchange Window, 596
  Preview Control, 604
  Python Modules, 596
  Soft Limits, 588
  Tool Touch Off, 584
  Virtual Control Panel, 604
axisui pins, 605

**B**
Backlash, 164
Base Period Maximum Jitter, 88
Basics, 208
Binary Operators, 883
bit, 214

**C**
Calling Files, 962
Canned Cycle Errors, 929
Canonical Device Interfaces, 314
cd, 24
Changing Directories, 24
CL Programming, 423
ClassicLadder Programming
  CL Programming, 423
CNC, 202

CNC), 1312
Commands, 208
addf, 209
loadrt, 209
loadusr, 210
net, 211
setp, 212
sets, 213
unlinkp, 213
Comment Parameters, 890
Comments, 146, 874, 889, 957
comp), 1312
Compensation, 164
Compensation End Point, 865
Complete Machine Configuration, 99
Component Generator, 284
Components, 146
Comments, 146
Custom sections and variables, 148
Include, 148
Sections, 147
Variables, 147
Conditional loops, 961
Configuration, 92
Configuration Launcher, 13
Configuration Selection, 50
connecting-rs485, 350
Controlled Point, 53
Conversion Components, 218
weighted_sum, 218
coolant, 52
Cooling, 54
Coordinate Systems, 851
Core components, 253
Creating Non-realtime Python Components, 311
Custom sections and variables, 148
Cutter Radius Compensation, 865

**D**
Data, 214
bit, 214
Files, 214
float, 214
s32, 214
u32, 214
debounce, 281
Debug Messages, 890
Determining Spindle Calibration, 97
DH parameters Examples, 471
Diese Achse testen, 93
Digital Input, 315
Digital Input Functions, 315
Digital Input Parameters, 315
Digital Input Pins, 315
Digital Output, 315
Digital Output Functions, 315
Digital Output Parameters, 315
Digital Output Pins, 315
DRO, 1312
Dwell
Feed Out, 940
dwell, 54

**E**
EDM, 1312
Eilgang), 1315
Eilgang-Geschwindigkeit, 1315
Einheiten), 1316
EMC, 1313
EMCIO, 1313
EMCMOT, 1313
Encoder, 275, 1313
Encoder Block Diagram, 275
End of Line Marker, 874
Entfernung zur Beschleunigung auf Höchstgeschwindigkeit, 93
Examples, 218
Expressions, 883
external, 568

**F**
F: Set Feed Rate, 964
Feed and Speed Interaction, 56
Feed Out, 939, 940
feed override, 52
Feed Rate, 54
FERROR, 165
File Requirements, 891
File Size, 891
Files, 214
Finding Maximum Acceleration, 95
Finding Maximum Velocity, 94
Five Phase, 272
float, 214
Four Phase, 272
Functions, 316, 884

**G**
G-Code, 1313
G-code, 46
G-code Best Practices, 892
G-code Order of Execution, 891
G-Code Table, 894
G0 Rapid Move, 896
G1 Linear Move, 896
G10 L0 Reload Tool Table Data, 906
G10 L1 Tool Table, 906
G10 L10 Set Tool Table, 908
G10 L11 Set Tool Table, 909
G10 L2 Coordinate System, 907
G10 L20 Set Coordinate System, 909
G17 - G19.1 Plane Select, 910
G2
G3 Arc Move, 897

G20 Units, 910
G28 Go/Set Predefined Position, 910
G3 Arc Move, 897
G30 Go/Set Predefined Position, 911
G33 Spindle Synchronized Motion, 911
G33.1 Rigid Tapping, 912
G38.n Probe, 914
G4 Dwell, 902
G40 Cutter Compensation Off, 915
G41 G42 Cutter Compensation, 916
G41.1 G42.1 Dynamic Compensation, 917
G43 Tool Length Offset, 917
G43.1 Dynamic Tool Length Offset, 918
G43.2 Apply additional Tool Length Offset, 918
G49 Cancel Tool Length Offset, 919
G5 Cubic spline, 902
G5.1 Quadratic spline, 903
G5.2 G5.3 NURBS Block, 904
G53 Machine Coordinates, 919
G54-G59.3 Select Coordinate System, 920
G61 Exact Path Mode, 921
G61.1 Exact Stop Mode, 921
G64 Path Blending, 921
G7 Lathe Diameter Mode, 905
G70 Lathe finishing cycle, 921
G71 G72 Lathe roughing cycles, 922
G73 Drilling Cycle with Chip Break, 923
G74 Left-hand Tapping Cycle with Dwell, 924
G76 Threading Cycle, 925
G8 Lathe Radius Mode, 905
G80 Cancel Modal Motion, 932
G80-G89 Canned Cycles, 928
G81 Drilling Cycle, 933
G82 Drilling Cycle Dwell, 938
G83 Peck Drilling, 938
G84 Right-hand Tapping Cycle Dwell, 939
G85 Boring
  Feed Out, 939
G86 Boring
  Spindle Stop
    Rapid Move Out, 940
G87 Back Boring Cycle, 940
G88 Boring Cycle
  Spindle Stop
    Manual Out, 940
G89 Boring
  Dwell
    Feed Out, 940
G90
  G91 Distance Mode, 941
G91 Distance Mode, 941
G92 Coordinate System Offset, 941
G92.1
  G92.2 Reset G92 Offsets, 942
G92.2 Reset G92 Offsets, 942
G92.3 Restore G92 Offsets, 942
G93
  G94
    G95 Feed Rate Mode, 942
G94
  G95 Feed Rate Mode, 942
G95 Feed Rate Mode, 942
G96
  G97 Spindle Control Mode, 943
G97 Spindle Control Mode, 943
G98
  G99 Canned Cycle Return, 944
G99 Canned Cycle Return, 944
Ganzzahl mit Vorzeichen, 1316
Ganzzahl ohne Vorzeichen, 1316
Gelenkkoordinaten, 1314
Getting Help, 3
Gewindespindelsteigung, 92
GladeVCP: Glade Virtual Control Panel, 1012
Graphical User Interfaces, 31
GS2 VFD Driver, 354
GUI, 1311, 1313

**H**

HAL, 199, 1314
  Analog Input, 315
  Analog Input Functions, 316
  Analog Input Parameters, 316
  Analog Input Pins, 315
  Analog Output, 316
    Functions, 316
    Parameters, 316
    Pins, 316
  Basics, 208
  Canonical Device Interfaces, 314
  Commands, 208
    addf, 209
    loadrt, 209
    loadusr, 210
    net, 211
    setp, 212
    sets, 213
    unlinkp, 213
  Component Generator, 284
  Conversion Components, 218
    weighted_sum, 218
  Data, 214
    bit, 214
    Files, 214
    float, 214
    s32, 214
    u32, 214
  Digital Input, 315
  Digital Input Functions, 315
  Digital Input Parameters, 315
  Digital Input Pins, 315
  Digital Output, 315
  Digital Output Functions, 315
  Digital Output Parameters, 315

Digital Output Pins, 315
Logic Components, 216
and2, 216
Examples, 218
not, 216
or2, 217
xor2, 217
Parameters, 215
pins
INI settings, 259
iocontrol, 258
motion), 254
spindle, 257
HAL Basics Summary, 204
HAL Component, 205
HAL Components, 207
HAL Concepts, 205
HAL Examples, 247
HAL Introduction, 199
HAL Parameter, 206
HAL Part selection, 203
HAL Physical Pin, 206
HAL Pin, 206
HAL pins and INI values, 882
HAL Signal, 206
Hal stepgen Functions, 272
HAL stepgen parameters, 271
HAL stepgen pins, 270
HAL stepgen Step Types, 271
HAL System Design, 202
HAL time, 216
HAL Timing Issues, 207
HAL tmax, 216
HAL Tools, 317
HAL Tutorial, 223
HAL Type, 206
HAL,Component List, 260
HAL: Implementation, 204
HAL: Interconnections Design, 204
HAL: Testing, 204
HAL:Function, 206
HAL:Geschwindigkeitsbeispiel, 250
HAL:Thread, 207
Halcmd Tutorial, 223
Halmeter, 317
Tutorial Halmeter, 229
Halui Examples, 309
History, 1322
HOME, 177
Home, 1314
HOME ABSOLUTE ENCODER, 178
HOME IGNORE LIMITS, 177
HOME INDEX NO ENCODER RESET, 177
HOME IS SHARED, 178
Home Latch Direction, 93
HOME LATCH VEL, 176
HOME OFFSET, 177
HOME SEARCH VEL, 165, 176
Home Search Velocity, 93
HOME SEQUENCE, 178
HOME USE INDEX, 177

**I**
Immediate Homing, 180
Include, 148
Indirection, 962
Inhibiting Homing, 180
INI, 1314
INI Configuration, 146
INI File
Components, 146
Comments, 146
Custom sections and variables, 148
Include, 148
Sections, 147
Variables, 147
Sections, 149
[APPLICATIONS] Section, 158
[AXIS_<letter>] Sections, 162
[DISPLAY] Section, 149
[EMCIO] Section, 170
[EMCMOT] Section, 156
[EMC] Section, 149
[FILTER] Section, 153
[HALUI] Section, 158
[HAL] Section, 157
[JOINT_<num>] Sections, 163
[RS274NGC] Section, 155
[SPINDLE_<num>] Section(s), 169
[TASK] Section, 157
[TRAJ] Section, 159
KINS Section, 162
INI settings, 259
INI settings (HAL pins, 259
Instanz, 1314
iocontrol, 258

**J**
Joggen, 1314
Jogwheel, 603

**K**
kartesische Maschinen, 465
Keyboard Shortcuts, 593
Kinematics, 465
Kinematik, 1314
KINS Section, 162
Koordinatenmessmaschine, 1312
Kugelmutter, 1312
Kugelumlaufspindel, 1312

**L**
Latency Test, 88, 138
Latency Testing, 137

Latency Tests, 137
Lathe Mode, 597
Lathe Tool Orientation, 58
Lathe Tool Table, 58
Lathe User Information, 57
Leitspindel, 1314
Line number, 873
LINEAR UNITS, 161
Linux FAQ, 22
LinuxCNC User Introduction
    User Introduction, 29
loadrt, 209
loadusr, 210
Local Offsets, 919
LOCKING INDEXER, 179
Logging, 890
Logic Components, 216
    and2, 216
    Examples, 218
    not, 216
    or2, 217
    xor2, 217
loop, 1316
Looping, 960
lut5, 283

**M**
M0
    M1 Program Pause, 945
M0 Mandatory Program Pause, 945
M1 Optional Program Pause, 945
M1 Program Pause, 945
M100-M199 User Defined Commands, 955
M19 Orient Spindle, 948
M2 Program End, 945
M3
    M4
      M5 Spindle Control, 946
M30 Pallet Exchange and Program End, 945
M4
    M5 Spindle Control, 946
M48
    M49 Speed and Feed Override Control, 948
M49 Speed and Feed Override Control, 948
M5 Spindle Control, 946
M50 Feed Override Control, 949
M51 Spindle Speed Override, 949
M52 Adaptive Feed Control, 949
M53 Feed Stop Control, 949
M6-Tool-Change, 946
M60 Pallet Change Pause, 946
M61 Set Current Tool, 949
M62 - M65 Digital Output Control, 950
M66 Wait on Input, 950
M67 Analog Output
    Synchronized, 951
M68 Analog Output, 952
M7
    M8
      M9 Coolant Control, 947
M70 Save Modal State, 952
M70 Saved state, 952
M71 Invalidate Stored Modal State, 953
M72 Restore Modal State, 953
M73 Save and Autorestore Modal State, 954
M8
    M9 Coolant Control, 947
M9 Coolant Control, 947
M98-M99, 959
Machine Overview, 51
machine units, 1314
Man Pages, 23
Manual Out, 940
Manual Tool Change, 596
Manual Toolchange Window, 596
Maschineneinheiten, 88
Maschinenname, 87
MAX ACCELERATION, 161
MAX LIMIT, 162, 164
Max Step Rate, 88
MAX VELOCITY, 161
Maximale Beschleunigung, 93
Maximale Geschwindigkeit, 92
MDI, 1314
mdro GUI, 847
Messages, 889
Microstepping des Treibers, 92
Min Base Period, 88
MIN FERROR, 165
MIN LIMIT, 162, 164
Modal Groups, 887
Modal Groups: G-codes, 887
Modal Groups: M-codes, 888
Motion, 253
motion), 254
Motorschritte pro Umdrehung, 92

**N**
named, 878
    predefined, 879
net, 211
NGCGUI, 709
NIST), 1314
NML, 1315
not, 216
numbered, 876
Numbers, 874

**O**
O-Code Errors, 963
Offsets, 1315
    external, 568
Operating without Home Switches, 100
Operators Precedence, 883

or2, 217
ORIENT OFFSET, 155
Other Codes, 964

**P**
Panelui, 1265
PARAMETER FILE, 155
Parameters, 56, 215, 316, 874, 875
  named, 878
    predefined, 879
  numbered, 876
Parport-Blockdiagramm, 328
Path Control Mode, 55
pci-card connectors, 335
PID, 277
PID Block Diagram, 277
PID Tuning, 518
pin-numbering-endsw, 347
pin-numbering-gpio, 335
Pin-Nummerierung-Achse, 337
Pins, 316
pins
  INI settings, 259
  iocontrol, 258
  motion), 254
  spindle, 257
Plasma Cutting Primer, 66
pluto-servo pinout, 398
pluto-step, 399
pluto-step pinout, 400
Pluto-Step-Timings, 401
Polar Coordinates, 886
predefined, 879
Preview Control, 604
primary linear, 52
Print Messages, 890
Probe Logging, 890
Programmeinheiten, 1315
Pulsfrequenz bei maximaler Geschwindigkeit, 93
PWM Spindle Speed Example, 413
PWMgen, 273
Python Modules, 596
PyVCP Widgets Reference, 981
PyVCP with AXIS, 978

**Q**
QtDragon, 675
QtVCP Overview, 1086

**R**
Rapid Move Out, 940
real-time, 1315
Referenzpunkt, 93
Referenzpunkt), 1314
Referenzschalter-Position, 93
Referenzsignal-Zeitdiagramme, 343
Repeat loop, 962
Return Values, 963
Riemenscheibenverhältnis, 92
rotational, 52
RS274NGC, 1315
RS274NGC STARTUP CODE, 155
RTAI, 1315
RTAPI, 1315
RTLINUX, 1315
Running LinuxCNC, 12
Rückkopplung, 1313

**S**
s32, 214
S: Set Spindle Speed, 964
Schrittmotor, 1316
secondary linear, 52
Sections, 147, 149
  [APPLICATIONS] Section, 158
  [AXIS_<letter>] Sections, 162
  [DISPLAY] Section, 149
  [EMCIO] Section, 170
  [EMCMOT] Section, 156
  [EMC] Section, 149
  [FILTER] Section, 153
  [HALUI] Section, 158
  [HAL] Section, 157
  [JOINT_<num>] Sections, 163
  [RS274NGC] Section, 155
  [SPINDLE_<num>] Section(s), 169
  [TASK] Section, 157
  [TRAJ] Section, 159
  KINS Section, 162
Servomotor, 1316
setp, 212
sets, 213
Sherline, 186
SigGen, 282
Simulated Encoder, 280
Soft Limits, 588
Spindel, 1316
spindle, 52, 257
Spindle At Speed Example, 416
Spindle Direction Example, 414
Spindle Enable Example, 413
spindle override, 52
Spindle Soft Start Example, 414
Spindle Speed Control, 96
Spindle Stop
  Manual Out, 940
  Rapid Move Out, 940
Spindle Synchronized Motion Example, 415
Spindle-synchronized motion, 97
Starting LinuxCNC, 49
stepgen, 268
Stepgen Block Diagram, 268
stepgen Example, 232

Stepper Configuration, 188
Stepper Configuration Wizard, 85
Stepper Diagnostics, 192
Subroutine Codes, 874
Subroutine Parameters, 878
SUBROUTINE PATH, 155
Subroutines, 958
  Conditional loops, 961
  Looping, 960
  M98-M99, 959
  Repeat loop, 962
Synchronized, 951
System Parameters, 881
System Requirements, 4

**T**
T: Select Tool, 964
TASK, 1316
Tischverfahrweg, 93
Tk, 1316
TkLinuxCNC Common Keyboard Shortcuts, 731
TkLinuxCNC Interpreter, 729
Tool Changers, 864
Tool Compensation, 859
Tool IO, 862
Tool Length Compensation, 864
Tool Positions 1
  2
    3 & 4, 59
Tool Positions 5
  6
    7 & 8, 59
Tool Table, 56, 860
Tool Table Format, 860
Tool Touch Off, 584
Touch Off, 859
touchygui, 656
TP, 43
Trajectory Control, 43, 921
Trajectory Planning
  TP, 43
Trajectory Planning:Path Following, 44
Trajectory Planning:Planning Moves, 45
Trajectory Planning:Programming the Planner, 44
Traverse Bewegung, 1316
Treiber Typ, 88
Trivialkinematik, 465
Tutorial Halmeter, 229
Tutorial Halscope, 236
Two and Three Phase, 272

**U**
u32, 214
Umkehrspiel, 1311
Umkehrspiel-Kompensation, 1311
Unary operations, 884
UNITS, 163
units, 54
unlinkp, 213
Updates to LinuxCNC, 10
User Concepts, 43
User Foreword, 28
User Introduction, 29
USER M PATH, 155

**V**
Variables, 147
Versatz, 1315
Virtual Control Panel, 604
VOLATILE HOME, 179
Vorschub, 1313
Vorschub-Override, 1313
Vorschubgeschwindigkeit, 1313

**W**
weighted_sum, 218
Weltkoordinaten, 1316
Werkstück Programm, 1315
Words, 873

**X**
xor2, 217
Xylotex, 186

**Z**
Zeit bis zum Erreichen der Maximalgeschwindigkeit, 93