# LinuxCNC V2.9.8-4205-g8893513972

# Contents

**I Начало работы и настройка** **1**

**1 Начало работы с LinuxCNC** **2**

1.1 О LinuxCNC . . . 2
1.1.1 Architecture - Context diagram . . . 3
1.1.2 The Operating System . . . 4
1.1.3 Получение помощи . . . 4
1.1.3.1 Web Forum . . . 4
1.1.3.2 IRC . . . 4
1.1.3.3 Mailing List . . . 5
1.1.3.4 Web Forum . . . 5
1.1.3.5 LinuxCNC Wiki . . . 5
1.1.3.6 Bug Reports . . . 5
1.2 Системные требования . . . 6
1.2.1 Минимальные требования . . . 6
1.2.2 Требования к ядру и версии . . . 6
1.2.2.1 Preempt-RT с пакетом «linuxcnc-uspace» . . . 7
1.2.2.2 RTAI с пакетом «linuxcnc» . . . 7
1.2.2.3 Xenomai с пакетом «linuxcnc-uspace» . . . 7
1.2.2.4 RTAI с пакетом «linuxcnc-uspace» . . . 7
1.2.3 Проблемное оборудование . . . 7
1.2.3.1 Ноутбуки . . . 7
1.2.3.2 Видеокарты . . . 7
1.3 Получение LinuxCNC . . . 7
1.3.1 Загрузка образа . . . 8
1.3.1.1 Обычная загрузка . . . 8
1.3.1.2 Загрузка с помощью zsync . . . 8
1.3.1.3 Проверка образа . . . 9
1.3.2 Запишите образ на загрузочное устройство . . . 9

1.3.2.1 Raspberry Pi Образ . . . 9
1.3.2.2 AMD-64 (x86-64, PC) Образ использующий инструменты ГИП . . . 9
1.3.2.3 Командная строка - Linux . . . 10
1.3.2.4 Командная строка - MacOS . . . 10
1.3.3 Тестирование LinuxCNC . . . 11
1.3.4 Установка LinuxCNC . . . 11
1.3.5 Обновления LinuxCNC . . . 11
1.3.6 Проблемы с установкой . . . 11
1.3.7 Альтернативные методы установки . . . 12
1.3.7.1 Installing on Debian Trixie (with Preempt-RT kernel) . . . 12
1.3.7.2 Installing on Debian Trixie (with experimental RTAI kernel) . . . 13
1.3.7.3 Установка на Raspbian 12 . . . 14
1.4 Running LinuxCNC . . . 14
1.4.1 Вызов LinuxCNC . . . 14
1.4.2 Конфигурационный загрузчик . . . 14
1.4.3 Следующие шаги по настройке . . . 17
1.4.4 Конфигурации симулятора . . . 17
1.4.5 Ресурсы по настройке . . . 18
1.5 Обновление LinuxCNC . . . 18
1.5.1 Обновление до новой версии . . . 18
1.5.1.1 Конфигурация источников Apt . . . 19
1.5.1.2 Обновление до новой версии . . . 21
1.5.1.3 Ubuntu . . . 21
1.5.2 Обновление без сети . . . 21
1.5.3 Обновление файлов конфигурации для версии 2.9 . . . 22
1.5.3.1 Более строгое обращение с подключаемыми интерпретаторами . . . 22
1.5.3.2 Canterp . . . 22
1.5.3.3 Ограничения шпинделя в INI . . . 22
1.5.4 Updating Configuration Files for 2.10.y . . . 22
1.5.5 New HAL components . . . 22
1.5.5.1 Не в реальном времени . . . 22
1.5.5.2 В реальном времени . . . 23
1.5.6 Новые драйверы . . . 23
1.6 Linux FAQ . . . 23
1.6.1 Автоматический вход . . . 23
1.6.1.1 Debian . . . 23
1.6.1.2 Ubuntu . . . 23
1.6.2 Automatic Startup . . . 24
1.6.3 Терминал . . . 24

1.6.4 Страницы руководства . . . 24
1.6.5 List Modules . . . 24
1.6.6 Editing a Root File . . . 25
1.6.6.1 The Command Line Way . . . 25
1.6.6.2 The GUI Way . . . 25
1.6.6.3 Root Access . . . 25
1.6.7 Terminal Commands . . . 25
1.6.7.1 Working Directory . . . 25
1.6.7.2 Изменение каталогов . . . 25
1.6.7.3 Listing files in a directory . . . 26
1.6.7.4 Finding a File . . . 26
1.6.7.5 Searching for Text . . . 26
1.6.7.6 Diagnostic Messages . . . 26
1.6.8 Convenience Items . . . 27
1.6.8.1 Terminal Launcher . . . 27
1.6.9 Hardware Problems . . . 27
1.6.9.1 Hardware Info . . . 27
1.6.9.2 Разрешение монитора . . . 27
1.6.10Пути . . . 27

**2 Общая информация для пользователя 28**
2.1 User Foreword . . . 28
2.2 Пользовательское введение в LinuxCNC . . . 29
2.2.1 Введение . . . 29
2.2.2 How LinuxCNC Works . . . 29
2.2.3 Графические интерфейсы пользователя . . . 31
2.2.4 Интерфейсы пользователя . . . 39
2.2.5 Виртуальные панели управления . . . 39
2.2.6 Языки . . . 42
2.2.7 Think Like a CNC Operator . . . 42
2.2.8 Modes of Operation . . . 43
2.3 Важные пользовательские концепции . . . 43
2.3.1 Управление траекторией . . . 43
2.3.1.1 Планировщик траектории . . . 43
2.3.1.2 Path Following . . . 44
2.3.1.3 Программирование планировщика . . . 44
2.3.1.4 Планирование перемещений . . . 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 Homing . . . 46
2.3.4 Tool Changes . . . 46
2.3.5 Coordinate Systems . . . 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 Начиная с LinuxCNC . . . 49
2.4.1 Running LinuxCNC . . . 49
2.4.1.1 Селектор конфигурации . . . 50
2.5 CNC Machine Overview . . . 51
2.5.1 Mechanical Components . . . 51
2.5.1.1 Axes . . . 51
2.5.1.2 Spindle . . . 52
2.5.1.3 Coolant . . . 52
2.5.1.4 Переопределение подачи и скорости . . . 52
2.5.1.5 Block Delete Switch . . . 52
2.5.1.6 Optional Program Stop Switch . . . 52
2.5.2 Control and Data Components . . . 52
2.5.2.1 Linear Axes . . . 52
2.5.2.2 Rotational Axes . . . 53
2.5.2.3 Контрольная точка . . . 53
2.5.2.4 Coordinated Linear Motion . . . 53
2.5.2.5 Feed Rate . . . 53
2.5.2.6 Охлаждение . . . 54
2.5.2.7 Dwell . . . 54
2.5.2.8 Units . . . 54
2.5.2.9 Current Position . . . 54
2.5.2.10 Selected Plane . . . 54
2.5.2.11 Tool Carousel . . . 54
2.5.2.12 Tool Change . . . 54
2.5.2.13 Pallet Shuttle . . . 55
2.5.2.14 Speed Override . . . 55
2.5.2.15 Path Control Mode . . . 55
2.5.3 Interpreter Interaction with Switches . . . 55
2.5.3.1 Клавиши переопределения подачи и скорости . . . 55
2.5.3.2 Block Delete Switch . . . 55

2.5.3.3 Optional Program Stop Switch . . . 55
2.5.4 Tool Table . . . 56
2.5.5 Параметры . . . 56
2.6 Общая информация для пользователя токарного станка . . . 57
2.6.1 Режим токарного станка . . . 57
2.6.2 Таблица инструментов токарного станка . . . 57
2.6.3 Ориентация токарного инструмента . . . 57
2.6.4 Инструмент Touch Off . . . 59
2.6.4.1 X Touch Off . . . 59
2.6.4.2 Z Touch Off . . . 60
2.6.4.3 Z Смещение станка . . . 60
2.6.5 Синхронизированное движение шпинделя . . . 61
2.6.6 Дуги . . . 61
2.6.6.1 Дуги и конструкция токарного станка . . . 61
2.6.6.2 Режим радиуса и диаметра . . . 61
2.6.7 Путь инструмента . . . 62
2.6.7.1 Контрольная точка . . . 62
2.6.7.2 Углы резки без компенсации резца . . . 62
2.6.7.3 Радиусное резание . . . 64
2.6.7.4 Использование компенсации резца . . . 66
2.7 Plasma Cutting Primer for LinuxCNC Users . . . 66
2.7.1 What Is Plasma? . . . 66
2.7.2 Arc Initialisation . . . 67
2.7.2.1 High Frequency Start . . . 67
2.7.2.2 Blowback Start . . . 68
2.7.3 CNC Plasma . . . 68
2.7.4 Choosing a Plasma Machine for CNC operations . . . 69
2.7.5 Types Of Torch Height Control . . . 70
2.7.6 Arc OK Signal . . . 70
2.7.7 Initial Height Sensing . . . 71
2.7.7.1 Float Switches . . . 71
2.7.7.2 Ohmic Sensing . . . 71
2.7.7.3 Hypersensing with a MESA THCAD-5 . . . 72
2.7.7.4 Example HAL Code for Hypersensing . . . 73
2.7.8 THC Delay . . . 74
2.7.9 Torch Voltage Sampling . . . 74
2.7.10 Torch Breakaway . . . 74
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
2.7.14.1 Arc OK (input) . . . 76
2.7.14.2 Torch On (output) . . . 77
2.7.14.3 Float switch (input) . . . 77
2.7.14.4 Ohmic Sensor enable (output) . . . 77
2.7.14.5 Ohmic Sensing (input) . . . 77
2.7.14.6 Torch Breakaway Sensor . . . 78
2.7.15 G-code For Plasma Controllers . . . 78
2.7.15.1 Enable/Disable THC Operation: . . . 78
2.7.16 External Offsets and Plasma Cutting . . . 79
2.7.17 Reading Arc Voltage With The Mesa THCAD . . . 79
2.7.17.1 THCAD Connections . . . 80
2.7.17.2 THCAD Initial Testing . . . 80
2.7.17.3 Which Model THCAD To Use? . . . 80
2.7.18 Post Processors And Nesting . . . 81
2.7.19 Designing For Noisy Electrical Environments . . . 81
2.7.20 Water Tables . . . 82
2.7.21 Downdraft Tables . . . 82
2.7.22 Designing For Speed And Acceleration . . . 82
2.7.23 Distance Travelled Per Motor Revolution . . . 83
2.7.24 QtPlasmaC LinuxCNC Plasma Configuration . . . 83
2.7.25 Hypertherm RS485 Control . . . 83
2.7.26 Post Processors For Plasma Cutting . . . 83

**3 Мастера настройки 85**

3.1 Мастер настройки шагового двигателя . . . 85
3.1.1 Введение . . . 85
3.1.2 Start Page . . . 86
3.1.3 Basic Information . . . 87
3.1.4 Parallel Port Setup . . . 89
3.1.5 Parallel Port 2 Setup . . . 90
3.1.6 Axis Configuration . . . 91
3.1.6.1 Нахождение максимальной скорости . . . 93
3.1.6.2 Finding Maximum Acceleration . . . 94
3.1.7 Spindle Configuration . . . 94
3.1.7.1 Контроль скорости шпинделя . . . 95
3.1.7.2 Spindle-synchronized motion . . . 95
3.1.7.3 Determining Spindle Calibration . . . 95

3.1.8 Варианты . . . 96
3.1.9 Полная конфигурация станка . . . 97
3.1.10 Axis Travels and Homes . . . 97
3.1.10.1 Operating without Limit Switches . . . 98
3.1.10.2 Работа без концевиков исходного положения . . . 98
3.1.10.3 Home and Limit Switch wiring options . . . 98
3.2 Mesa Configuration Wizard . . . 99
3.2.1 Step by Step Instructions . . . 100
3.2.2 Create or Edit . . . 100
3.2.3 Basic Machine Information . . . 101
3.2.4 External Configuration . . . 103
3.2.5 GUI Configuration . . . 105
3.2.6 Mesa Configuration . . . 108
3.2.7 Mesa I/O Setup . . . 110
3.2.8 Parallel port configuration . . . 114
3.2.9 Axis Configuration . . . 115
3.2.10 Spindle Configuration . . . 122
3.2.11 Advanced Options . . . 124
3.2.12 HAL Components . . . 125
3.2.13 Advanced Usage Of PnCconf . . . 126

**4 Конфигурация 129**
4.1 Концепции системного разработчика . . . 129
4.1.1 Расплоложение файлов . . . 129
4.1.1.1 На установленном пакете . . . 129
4.1.1.2 Командная строка . . . 130
4.1.2 Файлы . . . 130
4.1.3 Системы шаговых двигателей . . . 130
4.1.3.1 Base Period . . . 130
4.1.3.2 Параметры шагового импульса . . . 131
4.1.4 Серво системы . . . 131
4.1.4.1 Принцип работы . . . 131
4.1.4.2 Пропроциональный член . . . 133
4.1.4.3 Интегральный член . . . 133
4.1.4.4 Производный член . . . 134
4.1.4.5 Настройка обратной связи . . . 134
4.1.4.6 Ручная настройка . . . 134
4.1.5 S-Curve Trajectory Planning . . . 134
4.1.5.1 Enabling . . . 134

4.1.5.2 Tuning . . . 135
4.1.6 RTAI . . . 135
4.1.6.1 ACPI . . . 135
4.1.7 Варианты оборудования интерфейса компьютера/станка . . . 135
4.1.7.1 litehm2/rv901t . . . 135
4.2 Тестирование задержки . . . 136
4.2.1 Что такое задержка (latency)? . . . 136
4.2.2 Тесты задержки . . . 136
4.2.2.1 Latency Test . . . 136
4.2.2.2 График задержки . . . 138
4.2.2.3 Гистограмма задержки . . . 139
4.2.3 Настройка задержки . . . 140
4.2.3.1 Настройка BIOS на задержку . . . 140
4.2.3.2 Настройка Preempt-RT для задержки . . . 141
4.3 Stepper Tuning . . . 141
4.3.1 Getting the most out of Software Stepping . . . 141
4.3.1.1 Run a Latency Test . . . 142
4.3.1.2 Figure out what your drives expect . . . 142
4.3.1.3 Choose your BASE_PERIOD . . . 143
4.3.1.4 Use steplen, stepspace, dirsetup, and/or dirhold . . . 144
4.3.1.5 No Guessing! . . . 144
4.4 INI Configuration . . . 144
4.4.1 Компоненты INI-файла . . . 144
4.4.1.1 Comments . . . 145
4.4.1.2 Разделы . . . 145
4.4.1.3 Переменные . . . 146
4.4.1.4 Пользовательские разделы и переменные . . . 148
4.4.1.5 Включить файлы . . . 149
4.4.2 Разделы INI-файла . . . 149
4.4.2.1 [EMC] раздел . . . 149
4.4.2.2 [DISPLAY] Section . . . 150
4.4.2.3 [FILTER] раздел . . . 154
4.4.2.4 [RS274NGC] раздел . . . 156
4.4.2.5 [EMCMOT] раздел . . . 158
4.4.2.6 [TASK] раздел . . . 158
4.4.2.7 [HAL] раздел . . . 159
4.4.2.8 [HALUI] раздел . . . 160
4.4.2.9 [APPLICATIONS] раздел . . . 160
4.4.2.10 [TRAJ] раздел . . . 161

4.4.2.11 [KINS] раздел . . . 164
4.4.2.12 [AXIS_<letter>] Section . . . 164
4.4.2.13 [JOINT_<num>] разделы . . . 165
4.4.2.14 [SPINDLE_<num>] раздел(ы)) . . . 172
4.4.2.15 [EMCIO] раздел . . . 173
4.5 Конфигурация исходного положения . . . 174
4.5.1 Обзор . . . 174
4.5.2 Предварительные условия . . . 174
4.5.3 Пример схемы отдельного концевика исходного положения . . . 175
4.5.4 Пример схемы общего концевика предела/исходного положения . . . 176
4.5.5 Последовательность приведения к исходному положению . . . 177
4.5.6 Конфигурация . . . 179
4.5.6.1 HOME_SEARCH_VEL . . . 179
4.5.6.2 HOME_LATCH_VEL . . . 179
4.5.6.3 HOME_FINAL_VEL . . . 180
4.5.6.4 HOME_IGNORE_LIMITS . . . 180
4.5.6.5 HOME_USE_INDEX . . . 180
4.5.6.6 HOME_INDEX_NO_ENCODER_RESET . . . 180
4.5.6.7 HOME_OFFSET . . . 180
4.5.6.8 HOME . . . 181
4.5.6.9 HOME_IS_SHARED . . . 181
4.5.6.10HOME_ABSOLUTE_ENCODER . . . 181
4.5.6.11HOME_SEQUENCE . . . 182
4.5.6.12VOLATILE_HOME . . . 183
4.5.6.13LOCKING_INDEXER . . . 183
4.5.6.14Немедленно исходная позиция . . . 183
4.5.6.15Inhibiting Homing . . . 184
4.6 Конфигурация токарного станка . . . 185
4.6.1 Default Plane . . . 185
4.6.2 INI Settings . . . 185
4.7 Stepper Quickstart . . . 186
4.7.1 Latency Test . . . 186
4.7.2 Sherline . . . 186
4.7.3 Xylotex . . . 186
4.7.4 Machine Information . . . 186
4.7.5 Pinout Information . . . 187
4.7.6 Mechanical Information . . . 187
4.8 Конфигурация шагового двигателя . . . 188
4.8.1 Введение . . . 188

4.8.2 Maximum step rate . . . 189
4.8.3 Pinout . . . 189
4.8.3.1 Standard Pinout HAL . . . 189
4.8.3.2 Обзор . . . 191
4.8.3.3 Changing the standard_pinout.hal . . . 191
4.8.3.4 Changing polarity of a signal . . . 191
4.8.3.5 Adding PWM Spindle Speed Control . . . 192
4.8.3.6 Adding an enable signal . . . 192
4.8.3.7 External ESTOP button . . . 192
4.9 Диагностика шагового двигателя . . . 192
4.9.1 Common Problems . . . 192
4.9.1.1 Stepper Moves One Step . . . 192
4.9.1.2 No Steppers Move . . . 192
4.9.1.3 Distance Not Correct . . . 193
4.9.2 Error Messages . . . 193
4.9.2.1 Following Error . . . 193
4.9.2.2 RTAPI Error . . . 193
4.9.3 Testing . . . 194
4.9.3.1 Параметры шагового импульса . . . 194
4.10 Фильтрующие программы . . . 194
4.10.1 Введение . . . 194
4.10.2 Настройка INI для программных фильтров . . . 194
4.10.3 Создание программ фильтров на основе Python . . . 195

**5 HAL (Уровень аппаратной абстракции) 198**
5.1 HAL Введение . . . 198
5.1.1 HAL Обзор . . . 198
5.1.2 Коммуникация . . . 200
5.1.3 HAL System Design . . . 202
5.1.3.1 Выбор детали . . . 203
5.1.3.2 Проектирование межсоединений . . . 204
5.1.3.3 Реализация . . . 204
5.1.3.4 Testing . . . 204
5.1.3.5 Краткое содержание . . . 204
5.1.4 HAL Концепции . . . 205
5.1.5 HAL компоненты . . . 207
5.1.6 Проблемы с синхронизацией в HAL . . . 207
5.2 HAL Основы . . . 208
5.2.1 HAL команды . . . 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 Устаревшие команды . . . 213
5.2.2 HAL Data . . . 214
5.2.2.1 Bit . . . 214
5.2.2.2 Float . . . 214
5.2.2.3 s32 . . . 214
5.2.2.4 u32 . . . 214
5.2.2.5 s64 . . . 214
5.2.2.6 u64 . . . 215
5.2.3 HAL файлы . . . 215
5.2.4 HAL Параметр . . . 215
5.2.5 Основные логические компоненты . . . 215
5.2.5.1 and2 . . . 215
5.2.5.2 not . . . 216
5.2.5.3 or2 . . . 216
5.2.5.4 xor2 . . . 217
5.2.6 Примеры логики . . . 218
5.2.7 Компоненты преобразования . . . 218
5.2.7.1 weighted_sum . . . 218
5.3 HAL Двойной проход . . . 219
5.3.1 Двойной проход . . . 219
5.3.2 Post GUI . . . 221
5.3.3 Исключение файлов .hal . . . 221
5.3.4 Examples . . . 222
5.4 HAL Учебное пособие . . . 222
5.4.1 Введение . . . 222
5.4.2 Halcmd . . . 222
5.4.2.1 Обозначения . . . 222
5.4.2.2 Табуляция-завершение . . . 223
5.4.2.3 Среда RTAPI . . . 223
5.4.3 Простой пример . . . 223
5.4.3.1 Загрузка компонента . . . 223
5.4.3.2 Изучение HAL . . . 224

5.4.3.3 Запуск кода в реальном времени . . . 225
5.4.3.4 Изменение параметров . . . 226
5.4.3.5 Сохранение конфигурации HAL . . . 227
5.4.3.6 Выход из halrun . . . 227
5.4.3.7 Восстановление конфигурации HAL . . . 228
5.4.3.8 Удаление HAL из памяти . . . 228
5.4.4 Halmeter . . . 228
5.4.5 Stepgen Пример . . . 231
5.4.5.1 Установка компонентов . . . 231
5.4.5.2 Соединение контактов с сигналами . . . 233
5.4.5.3 Настройка выполнения в реальном времени — потоки и функции . . . 233
5.4.5.4 Настройка параметров . . . 235
5.4.5.5 Запустить его! . . . 235
5.4.6 Halscope . . . 235
5.4.6.1 Подключение щупов . . . 238
5.4.6.2 Захват наших первых сигналов . . . 241
5.4.6.3 Вертикальная регулировки . . . 242
5.4.6.4 Запуск развертки . . . 243
5.4.6.5 Горизонтальные регулировки . . . 245
5.4.6.6 Больше каналов . . . 246
5.4.6.7 Больше выборок . . . 247
5.5 HAL Примеры . . . 247
5.5.1 Подключение двух выходов . . . 247
5.5.2 Ручная смена инструмента . . . 248
5.5.3 Вычислить скорость . . . 249
5.5.4 Подробности плавного запуска . . . 250
5.5.5 Автономный HAL . . . 252
5.6 Основные компоненты . . . 253
5.6.1 Motion . . . 254
5.6.1.1 Варианты . . . 254
5.6.1.2 Контакты . . . 255
5.6.1.3 Параметры . . . 256
5.6.1.4 Функции . . . 257
5.6.2 Spindle . . . 257
5.6.2.1 Контакты . . . 257
5.6.3 Контакты осей и сочленений и параметры . . . 259
5.6.4 iocontrol . . . 259
5.6.4.1 Контакты . . . 259
5.6.5 настройки INI . . . 259

5.6.5.1 Контакты . . . 259
5.7 HAL Список компонентов . . . 261
5.7.1 Компонентов . . . 261
5.7.1.1 Пользовательские интерфейсы (не в реальном времени) . . . 261
5.7.1.2 Движение (не в реальном времени) . . . 262
5.7.1.3 Драйверы оборудования . . . 262
5.7.1.4 Mesa и другие карты ввода-вывода (в реальном времени) . . . 263
5.7.1.5 Утилиты (не в реальном времени) . . . 263
5.7.1.6 Обработка сигналов (в реальном времени) . . . 264
5.7.1.7 Генерация сигнала (в реальном времени) . . . 266
5.7.1.8 Кинематика (в реальном времени) . . . 267
5.7.1.9 Управление движением (в реальном времени) . . . 268
5.7.1.10 Управление двигателем (в реальном времени) . . . 268
5.7.1.11 Моделирование/Тестирование . . . 268
5.7.1.12 Другое (в реальном времени) . . . 268
5.7.2 Вызовы HAL API . . . 270
5.7.3 RTAPI-вызовы . . . 270
5.8 HAL Component Descriptions . . . 271
5.8.1 StepGen . . . 272
5.8.1.1 Контакты . . . 272
5.8.1.2 Параметры . . . 273
5.8.1.3 Типы шаговых импульсов . . . 274
5.8.1.4 Функции . . . 275
5.8.2 PWMgen . . . 275
5.8.2.1 Типы выходов . . . 276
5.8.2.2 Контакты . . . 276
5.8.2.3 Параметры . . . 277
5.8.2.4 Функции . . . 277
5.8.3 Энкодер . . . 277
5.8.3.1 Контакты . . . 278
5.8.3.2 Параметры . . . 279
5.8.3.3 Функции . . . 279
5.8.4 ПИД . . . 280
5.8.4.1 Контакты . . . 280
5.8.4.2 Функции . . . 282
5.8.5 Имитированный энкодер . . . 282
5.8.5.1 Контакты . . . 282
5.8.5.2 Параметры . . . 282
5.8.5.3 Функции . . . 283

5.8.6 Debounce . . . 283
5.8.6.1 Контакты . . . 283
5.8.6.2 Параметры . . . 284
5.8.6.3 Функции . . . 284
5.8.7 SigGen . . . 284
5.8.7.1 Контакты . . . 284
5.8.7.2 Параметры . . . 285
5.8.7.3 Функции . . . 285
5.8.8 lut5 . . . 285
5.9 HAL Генератор компонентов . . . 287
5.9.1 Введение . . . 287
5.9.2 Installing . . . 287
5.9.3 Компиляция . . . 288
5.9.3.1 Внутри дерева исходников . . . 288
5.9.3.2 Компоненты реального времени вне дерева исходного кода . . . 288
5.9.3.3 Компоненты не реального времени вне дерева исходного кода . . . 288
5.9.4 Использование компонента . . . 289
5.9.5 Определения . . . 289
5.9.6 Создание экземпляра . . . 289
5.9.7 Неявные параметры . . . 290
5.9.8 Syntax . . . 290
5.9.8.1 HAL функции . . . 292
5.9.8.2 Варианты . . . 292
5.9.8.3 Лицензия и авторство . . . 294
5.9.8.4 Поэкземплярное хранилище данных . . . 294
5.9.8.5 Comments . . . 294
5.9.9 Ограничения . . . 294
5.9.10 Удобные макросы . . . 295
5.9.11 Компоненты с одной функцией . . . 296
5.9.12 Индивидуальность компонента . . . 296
5.9.13 Examples . . . 296
5.9.13.1 constant . . . 296
5.9.13.2 sincos . . . 297
5.9.13.3 out8 . . . 297
5.9.13.4 hal_loop . . . 298
5.9.13.5 arraydemo . . . 298
5.9.13.6 rand . . . 298
5.9.13.7 logic (using personality) . . . 299
5.9.13.8 Общие функции . . . 300

5.9.14 Использование командной строки . . . 300
5.10 HALTCL Files . . . 300
5.10.1 Compatibility . . . 301
5.10.2 Haltcl Commands . . . 301
5.10.3 Haltcl INI-file variables . . . 301
5.10.4 Converting HAL files to Tcl files . . . 302
5.10.5 Haltcl Notes . . . 303
5.10.6 Haltcl Examples . . . 303
5.10.7 Haltcl Interactive . . . 304
5.10.8 Haltcl Distribution Examples (sim) . . . 304
5.11 HAL User Interface . . . 304
5.11.1 Введение . . . 304
5.11.2 MDI . . . 304
5.11.3 Пример конфигурации . . . 305
5.11.4 Справочник по контактам Halui . . . 305
5.11.4.1 Прерывание . . . 305
5.11.4.2 E-Stop . . . 305
5.11.4.3 Переопределение подачи . . . 305
5.11.4.4 Туман . . . 306
5.11.4.5 СОЖ . . . 306
5.11.4.6 Homing . . . 306
5.11.4.7 Станок . . . 306
5.11.4.8 Макс скорость . . . 306
5.11.4.9 MDI . . . 307
5.11.4.10 Сочленение . . . 307
5.11.4.11 Медленная подача сочленения . . . 308
5.11.4.12 Axis . . . 309
5.11.4.13 Медленная подача оси . . . 309
5.11.4.14 Mode . . . 310
5.11.4.15 Программа . . . 310
5.11.4.16 Быстрое переназначение . . . 311
5.11.4.17 Переопределение шпинделя . . . 311
5.11.4.18 Spindle . . . 311
5.11.4.19 Tool . . . 312
5.12 Примеры Halui . . . 312
5.12.1 Remote Start . . . 312
5.12.2 Pause & Resume . . . 313
5.13 Создание Python компонентов не реального времени . . . 314
5.13.1 Базовый пример использования . . . 314

5.13.2 Компоненты и задержки, не относящиеся к реальному времени . . . 315
5.13.3 Создание контактов и параметров . . . 315
5.13.3.1 Изменение префикса . . . 316
5.13.4 Чтение и запись контактов и параметров . . . 316
5.13.4.1 Управляющие выходные контакты (HAL_OUT) . . . 316
5.13.4.2 Управление двунаправленными (HAL_IO) контактами . . . 316
5.13.5 Выход . . . 316
5.13.6 Полезные функции . . . 316
5.13.7 Constants . . . 317
5.13.8 Системная информация . . . 317
5.14 Canonical Device Interfaces . . . 317
5.14.1 Введение . . . 317
5.14.2 Цифровой вход . . . 318
5.14.2.1 Контакты . . . 318
5.14.2.2 Параметры . . . 318
5.14.2.3 Функции . . . 318
5.14.3 Цифровой вывод . . . 318
5.14.3.1 Контакты . . . 318
5.14.3.2 Параметры . . . 318
5.14.3.3 Функции . . . 318
5.14.4 Аналоговый вход . . . 318
5.14.4.1 Контакты . . . 318
5.14.4.2 Параметры . . . 319
5.14.4.3 Функции . . . 319
5.14.5 Аналоговый вывод . . . 319
5.14.5.1 Контакты . . . 319
5.14.5.2 Параметры . . . 319
5.14.5.3 Функции . . . 319
5.15 HAL Инструменты . . . 320
5.15.1 Halcmd . . . 320
5.15.2 Halmeter . . . 320
5.15.3 Halshow . . . 322
5.15.4 Halscope . . . 323
5.15.5 Sim Pin . . . 323
5.15.6 Simulate Probe . . . 324
5.15.7 HAL Histogram . . . 325
5.15.8 Halreport . . . 326

**6 Драйверы оборудования 329**
6.1 Драйвер параллельного порта . . . 329
6.1.1 Загрузка . . . 330
6.1.2 Адрес порта PCI . . . 332
6.1.3 Контакты . . . 333
6.1.4 Параметры . . . 333
6.1.5 Функции . . . 333
6.1.6 Распространенные проблемы . . . 334
6.1.7 Использование DoubleStep . . . 334
6.1.8 probe_parport . . . 334
6.1.8.1 Установка probe_parport . . . 335
6.2 AX5214H Driver . . . 335
6.2.1 Installing . . . 335
6.2.2 Контакты . . . 335
6.2.3 Параметры . . . 336
6.2.4 Функции . . . 336
6.3 General Mechatronics Driver . . . 336
6.3.1 I/O connectors . . . 338
6.3.1.1 Контакты . . . 339
6.3.1.2 Параметры . . . 339
6.3.2 Axis connectors . . . 339
6.3.2.1 Axis interface modules . . . 340
6.3.2.2 Энкодер . . . 341
6.3.2.3 StepGen module . . . 343
6.3.2.4 Enable and Fault signals . . . 346
6.3.2.5 Axis DAC . . . 346
6.3.3 CAN-bus servo amplifiers . . . 347
6.3.3.1 Контакты . . . 348
6.3.3.2 Параметры . . . 348
6.3.4 Watchdog timer . . . 348
6.3.4.1 Контакты . . . 348
6.3.4.2 Параметры . . . 348
6.3.5 End-, homing- and E-stop switches . . . 349
6.3.5.1 Контакты . . . 350
6.3.5.2 Параметры . . . 350
6.3.6 Status LEDs . . . 350
6.3.6.1 CAN . . . 350
6.3.6.2 RS485 . . . 351
6.3.6.3 EMC . . . 351

6.3.6.4 Boot . . . 351
6.3.6.5 Error . . . 351
6.3.7 RS485 I/O expander modules . . . 351
6.3.7.1 Relay output module . . . 352
6.3.7.2 Digital input module . . . 353
6.3.7.3 DAC & ADC module . . . 354
6.3.7.4 Teach Pendant module . . . 355
6.3.8 Errata . . . 356
6.3.8.1 GM6-PCI card Errata . . . 356
6.4 GS2 VFD Driver . . . 356
6.4.1 Command Line Options . . . 356
6.4.2 Контакты . . . 357
6.4.3 Параметры . . . 358
6.5 HAL Driver for Raspberry Pi GPIO pins . . . 358
6.5.1 Purpose . . . 358
6.5.2 Использование . . . 358
6.5.3 Контакты . . . 359
6.5.4 Параметры . . . 359
6.5.5 Функции . . . 360
6.5.6 Pin Numbering . . . 360
6.5.7 Known Bugs . . . 360
6.6 Универсальный драйвер для любого GPIO, поддерживаемого gpiod. . . . 360
6.6.1 Purpose . . . 360
6.6.2 Использование . . . 361
6.6.3 Контакты . . . 362
6.6.4 Параметры . . . 362
6.6.5 Функции . . . 362
6.6.6 Идентификация контактов . . . 362
6.6.7 Troubleshooting permissions problems. . . . 363
6.6.8 Автор . . . 363
6.6.9 Known Bugs . . . 363
6.7 Mesa HostMot2 драйвер . . . 363
6.7.1 Введение . . . 363
6.7.2 Бинарные файлы прошивки . . . 364
6.7.3 Установка прошивки . . . 364
6.7.4 Загрузка HostMot2 . . . 364
6.7.5 Сторожевой таймер . . . 365
6.7.5.1 Контакты . . . 365
6.7.5.2 Параметры . . . 365

6.7.6 HostMot2 Функции . . . 365
6.7.7 Распиновка . . . 366
6.7.8 PIN Файлы . . . 367
6.7.9 Прошивка . . . 367
6.7.10 HAL Контакты . . . 367
6.7.11 Конфигурации . . . 368
6.7.12 GPIO . . . 370
6.7.12.1 Контакты . . . 370
6.7.12.2 Параметры . . . 371
6.7.13 StepGen . . . 371
6.7.13.1 Контакты . . . 372
6.7.13.2 Параметры . . . 372
6.7.13.3 Выходные параметры . . . 372
6.7.14 PWMGen . . . 373
6.7.14.1 Контакты . . . 373
6.7.14.2 Параметры . . . 373
6.7.14.3 Выходные параметры . . . 374
6.7.15 Энкодер . . . 374
6.7.15.1 Контакты . . . 374
6.7.15.2 Параметры . . . 375
6.7.16 5I25 Конфигурация . . . 375
6.7.16.1 Прошивка . . . 375
6.7.16.2 Конфигурация . . . 376
6.7.16.3 SSERIAL Конфигурация . . . 376
6.7.16.4 7I77 Пределы . . . 376
6.7.17 Примеры конфигураций . . . 377
6.8 MB2HAL . . . 377
6.8.1 Введение . . . 377
6.8.2 Использование . . . 377
6.8.3 Варианты . . . 378
6.8.3.1 Init Section . . . 378
6.8.3.2 Transaction Sections . . . 378
6.8.3.3 Error codes . . . 379
6.8.4 Example config file . . . 380
6.8.5 Контакты . . . 385
6.8.5.1 fnct_01_read_coils . . . 385
6.8.5.2 fnct_02_read_discrete_inputs . . . 385
6.8.5.3 fnct_03_read_holding_registers . . . 385
6.8.5.4 fnct_04_read_input_registers . . . 385

6.8.5.5 fnct_05_write_single_coil . . . 385
6.8.5.6 fnct_06_write_single_register . . . 386
6.8.5.7 fnct_15_write_multiple_coils . . . 386
6.8.5.8 fnct_16_write_multiple_registers . . . 386
6.9 Mitsub VFD Driver . . . 386
6.9.1 Command Line Options . . . 386
6.9.2 Контакты . . . 387
6.9.3 HAL example . . . 387
6.9.4 Configuring the Mitsubishi VFD for serial usage . . . 388
6.9.4.1 Connecting the Serial Port . . . 388
6.9.4.2 Modbus setup . . . 388
6.10 Motenc Driver . . . 389
6.10.1 Контакты . . . 389
6.10.2 Параметры . . . 390
6.10.3 Функции . . . 390
6.11 Opto22 Driver . . . 390
6.11.1 The Adapter Card . . . 391
6.11.2 The Driver . . . 391
6.11.3 Контакты . . . 391
6.11.4 Параметры . . . 391
6.11.5 FUNCTIONS . . . 392
6.11.6 Configuring I/O Ports . . . 392
6.11.7 Pin Numbering . . . 392
6.12 Pico Drivers . . . 393
6.12.1 Command Line Options . . . 393
6.12.2 Контакты . . . 394
6.12.3 Параметры . . . 395
6.12.4 Функции . . . 396
6.13 Pluto P Driver . . . 396
6.13.1 General Info . . . 396
6.13.1.1 Requirements . . . 396
6.13.1.2 Connectors . . . 397
6.13.1.3 Physical Pins . . . 397
6.13.1.4 LED . . . 397
6.13.1.5 Power . . . 397
6.13.1.6 PC interface . . . 397
6.13.1.7 Rebuilding the FPGA firmware . . . 398
6.13.1.8 For more information . . . 398
6.13.2 Pluto Servo . . . 398

6.13.2.1 Pinout . . . 398
6.13.2.2 Input latching and output updating . . . 400
6.13.2.3 HAL Functions, Pins and Parameters . . . 400
6.13.2.4 Compatible driver hardware . . . 400
6.13.3 Pluto Step . . . 400
6.13.3.1 Pinout . . . 401
6.13.3.2 Input latching and output updating . . . 401
6.13.3.3 Step Waveform Timings . . . 401
6.13.3.4 HAL Functions, Pins and Parameters . . . 402
6.14 Powermax Modbus Driver . . . 402
6.14.1 Контакты . . . 403
6.14.2 Description . . . 403
6.14.3 Reference: . . . 403
6.15 Servo To Go Driver . . . 404
6.15.1 Installing . . . 404
6.15.2 Контакты . . . 404
6.15.3 Параметры . . . 405
6.15.4 Функции . . . 405
6.16 Shuttle . . . 405
6.16.1 Description . . . 405
6.16.2 Setup . . . 406
6.16.3 Контакты . . . 406
6.17 VFS11 VFD Driver . . . 407
6.17.1 Command Line Options . . . 407
6.17.2 Контакты . . . 407
6.17.3 Параметры . . . 409
6.17.4 INI file configuration . . . 409
6.17.5 HAL example . . . 410
6.17.6 Panel operation . . . 411
6.17.7 Error Recovery . . . 411
6.17.8 Configuring the VFS11 VFD for Modbus usage . . . 411
6.17.8.1 Connecting the Serial Port . . . 411
6.17.8.2 Modbus setup . . . 411
6.17.9 Programming Note . . . 412

**7 Примеры оборудования 413**
7.1 PCI Parallel Port . . . 413
7.2 Управление шпинделем . . . 414
7.2.1 Скорость вращения шпинделя 0-10 В . . . 414
7.2.2 Скорость шпинделя ШИМ . . . 414
7.2.3 Разрешить шпиндель . . . 414
7.2.4 Направление вращения шпинделя . . . 415
7.2.5 Плавный пуск шпинделя . . . 415
7.2.6 Обратная связь шпинделя . . . 416
7.2.6.1 Синхронизированное движение шпинделя . . . 416
7.2.6.2 Шпиндель достиг скорости (Spindle At Speed) . . . 417
7.3 MPG Pendant . . . 417
7.4 GS2 Spindle . . . 420
7.4.1 Пример . . . 420

**8 Язык релейных схем 421**
8.1 ClassicLadder введение . . . 421
8.1.1 История . . . 421
8.1.2 Введение . . . 421
8.1.3 Пример . . . 422
8.1.4 Базовая схема включения-выключения с фиксацией . . . 423
8.2 Программирование ClassicLadder . . . 424
8.2.1 Концепции языка релейных схем . . . 424
8.2.2 Языки . . . 424
8.2.3 Компонентов . . . 424
8.2.3.1 Файлы . . . 425
8.2.3.2 Модуль реального времени . . . 425
8.2.3.3 Переменные . . . 425
8.2.4 Загрузка ClassicLadder модуля не в реальном времени . . . 426
8.2.5 ClassicLadder ГИП . . . 427
8.2.5.1 Менеджер разделов . . . 427
8.2.5.2 Отображение раздела . . . 427
8.2.5.3 Окна переменных . . . 429
8.2.5.4 Окно символов . . . 431
8.2.5.5 Окно редактора . . . 432
8.2.5.6 Окно конфигурации . . . 434
8.2.6 Объекты релейной логики . . . 435
8.2.6.1 КОНТАКТЫ . . . 435
8.2.6.2 IEC ТАЙМЕРЫ . . . 435

8.2.6.3 ТАЙМЕРЫ . . . 436
8.2.6.4 МОНОСТАБИЛЬНЫЕ . . . 436
8.2.6.5 СЧЕТЧИКИ . . . 437
8.2.6.6 СРАВНЕНИЕ . . . 437
8.2.6.7 НАЗНАЧЕНИЕ ПЕРЕМЕННОЙ . . . 438
8.2.6.8 КАТУШКИ . . . 440
8.2.7 Переменные ClassicLadder . . . 441
8.2.8 Программирование GRAFCET (конечный автомат) . . . 442
8.2.9 Modbus . . . 444
8.2.10 Настройки MODBUS . . . 448
8.2.10.1 Информация о MODBUS . . . 448
8.2.10.2 Ошибки связи . . . 448
8.2.11 Отладка проблем с Modbus . . . 449
8.2.11.1 Запрос . . . 450
8.2.11.2 Ответ об ошибке . . . 451
8.2.11.3 Ответ данных . . . 452
8.2.11.4 Ошибки MODBUS . . . 453
8.2.12 Настройка ClassicLadder . . . 453
8.2.12.1 Добавление модулей . . . 454
8.2.12.2 Добавление лестничной логики . . . 454
8.3 ClassicLadder примеры . . . 461
8.3.1 Счетчик с предустановкой . . . 461
8.3.2 Отклонить дополнительные импульсы . . . 461
8.3.3 Внешний аварийный останов . . . 462
8.3.4 Timer/Operate пример . . . 465

**9 Расширенные темы 467**

9.1 Kinematics . . . 467
9.1.1 Введение . . . 467
9.1.1.1 Сочленения против осей . . . 467
9.1.2 Тривиальная кинематика . . . 468
9.1.3 Нетривиальная кинематика . . . 469
9.1.3.1 Forward transformation . . . 470
9.1.3.2 Обратное преобразование . . . 471
9.1.4 Детали реализации . . . 472
9.1.4.1 Модуль кинематики с использованием шаблона userkins.comp . . . 473
9.2 Setting up ”modified” Denavit-Hartenberg (DH) parameters for *genserkins* . . . 473
9.2.1 Prelude . . . 473
9.2.2 General . . . 473

9.2.3 Modified DH-Parameters . . . 474
9.2.4 Modified DH-Parameters as used in *genserkins* . . . 474
9.2.5 Numbering of joints and parameters . . . 475
9.2.6 How to start . . . 475
9.2.7 Special cases . . . 475
9.2.8 Detailed Example (RV-6SL) . . . 475
9.2.9 Credits . . . 494
9.3 5-Axis Kinematics . . . 494
9.3.1 Введение . . . 494
9.3.2 5-Axis Machine Tool Configurations . . . 494
9.3.3 Tool Orientation and Location . . . 494
9.3.4 Translation and Rotation Matrices . . . 495
9.3.5 Table Rotary/Tilting 5-Axis Configurations . . . 496
9.3.5.1 Transformations for a xyzac-trt machine tool with work offsets . . . 498
9.3.5.2 Transformations for a xyzac-trt machine with rotary axis offsets . . . 502
9.3.5.3 Transformations for a xyzbc-trt machine with rotary axis offsets . . . 505
9.3.6 Table Rotary/Tilting Examples . . . 508
9.3.6.1 Vismach Simulation Models . . . 508
9.3.6.2 Tool-Length Compensation . . . 508
9.3.7 Custom Kinematics Components . . . 508
9.3.8 Figures . . . 510
9.3.9 REFERENCES . . . 512
9.4 Switchable Kinematics (switchkins) . . . 512
9.4.1 Введение . . . 512
9.4.2 Switchable Kinematic Modules . . . 513
9.4.2.1 назначения идентифицирующей буквы . . . 513
9.4.2.2 Обратная совместимость . . . 514
9.4.3 HAL Контакты . . . 514
9.4.3.1 Сводная информация о контактах HAL . . . 514
9.4.4 Использование . . . 514
9.4.4.1 HAL Connections . . . 514
9.4.4.2 Команды G-/M-кода . . . 515
9.4.4.3 Настройки ограничений INI-файла . . . 515
9.4.4.4 Соображения смещения системы координат . . . 517
9.4.4.5 Внешние соображения смещения . . . 517
9.4.5 Конфигурации моделирования . . . 517
9.4.6 Условия пользовательской кинематики . . . 518
9.4.7 Предупреждения . . . 518
9.4.8 Примечания к коду . . . 519

9.5 PID Настройка . . . 519
9.5.1 ПИД-контроллер . . . 519
9.5.1.1 Основы контура управления . . . 519
9.5.1.2 Теория . . . 520
9.5.1.3 Настройка контура . . . 521
9.5.1.4 Автоматическая настройка ПИД-регулятора . . . 522
9.6 Remap Extending G-code . . . 523
9.6.1 Introduction: Extending the RS274NGC Interpreter by Remapping Codes . . . 523
9.6.1.1 A Definition: Remapping Codes . . . 523
9.6.1.2 Why would you want to extend the RS274NGC Interpreter? . . . 523
9.6.2 Getting started . . . 525
9.6.2.1 Builtin Remaps . . . 525
9.6.2.2 Picking a code . . . 526
9.6.2.3 Parameter handling . . . 526
9.6.2.4 Handling results . . . 526
9.6.2.5 Execution sequencing . . . 527
9.6.2.6 An minimal example remapped code . . . 527
9.6.3 Configuring Remapping . . . 527
9.6.3.1 The REMAP statement . . . 527
9.6.3.2 Useful REMAP option combinations . . . 528
9.6.3.3 The argspec parameter . . . 529
9.6.4 Upgrading an existing configuration for remapping . . . 532
9.6.5 Remapping tool change-related codes: T, M6, M61 . . . 533
9.6.5.1 Обзор . . . 533
9.6.5.2 Understanding the role of `iocontrol` with remapped tool change codes . . . 534
9.6.5.3 Specifying the M6 replacement . . . 534
9.6.5.4 Configuring `iocontrol` with a remapped M6 . . . 536
9.6.5.5 Writing the change and prepare O-word procedures . . . 537
9.6.5.6 Making minimal changes to the built in codes, including `M6` . . . 537
9.6.5.7 Specifying the T (prepare) replacement . . . 538
9.6.5.8 Error handling: dealing with abort . . . 539
9.6.5.9 Error handling: failing a remapped code NGC procedure . . . 540
9.6.6 Remapping other existing codes: . . . 541
9.6.6.1 Automatic gear selection be remapping S (set spindle speed) . . . 541
9.6.6.2 Adjusting the behavior of M0, M1 . . . 541
9.6.6.3 Adjusting the behavior of M7, M8, M9 . . . 541
9.6.7 Creating new G-code cycles . . . 541
9.6.8 Configuring Embedded Python . . . 542
9.6.8.1 Python plugin : INI file configuration . . . 542

9.6.8.2 Executing Python statements from the interpreter . . . 542
9.6.9 Programming Embedded Python in the RS274NGC Interpreter . . . 543
9.6.9.1 The Python plugin namespace . . . 543
9.6.9.2 The Interpreter as seen from Python . . . 543
9.6.9.3 The Interpreter `__init__` and `__delete__` functions . . . 543
9.6.9.4 Calling conventions: NGC to Python . . . 544
9.6.9.5 Calling conventions: Python to NGC . . . 547
9.6.9.6 Built in modules . . . 549
9.6.10 Adding Predefined Named Parameters . . . 549
9.6.11 Standard Glue routines . . . 550
9.6.11.1 T: `prepare_prolog` and `prepare_epilog` . . . 550
9.6.11.2 M6: `change_prolog` and `change_epilog` . . . 550
9.6.11.3 G-code Cycles: `cycle_prolog` and `cycle_epilog` . . . 551
9.6.11.4 S (Set Speed) : `setspeed_prolog` and `setspeed_epilog` . . . 552
9.6.11.5 F (Set Feed) : `setfeed_prolog` and `setfeed_epilog` . . . 552
9.6.11.6 M61 Set tool number : `settool_prolog` and `settool_epilog` . . . 552
9.6.12 Remapped code execution . . . 552
9.6.12.1 NGC procedure call environment during remaps . . . 552
9.6.12.2 Nested remapped codes . . . 552
9.6.12.3 Sequence number during remaps . . . 552
9.6.12.4 Debugging flags . . . 553
9.6.12.5 Debugging Embedded Python code . . . 553
9.6.13 Axis Preview and Remapped code execution . . . 554
9.6.14 Remappable Codes . . . 555
9.6.14.1 Existing codes which can be remapped . . . 555
9.6.14.2 Currently unallocated G-codes: . . . 555
9.6.14.3 Currently unallocated M-codes: . . . 558
9.6.15 A short survey of LinuxCNC program execution . . . 559
9.6.15.1 Interpreter state . . . 559
9.6.15.2 Task and Interpreter interaction, Queuing and Read-Ahead . . . 559
9.6.15.3 Predicting the machine position . . . 560
9.6.15.4 Queue-busters break position prediction . . . 560
9.6.15.5 How queue-busters are dealt with . . . 560
9.6.15.6 Word order and execution order . . . 561
9.6.15.7 Parsing . . . 561
9.6.15.8 Execution . . . 561
9.6.15.9 Procedure execution . . . 561
9.6.15.10 How tool change currently works . . . 561
9.6.15.11 How Tx (Prepare Tool) works . . . 562

9.6.15.12How M6 (Change tool) works . . . 562
9.6.15.13How M61 (Change tool number) works . . . 563
9.6.16Status . . . 564
9.6.17Changes . . . 564
9.6.18Debugging . . . 564
9.7 Компонент Moveoff . . . 564
9.7.1 Modifying an existing configuration . . . 566
9.8 Автономный интерпретатор . . . 569
9.8.1 Использование . . . 569
9.8.2 Пример . . . 570
9.9 Смещения внешней оси . . . 570
9.9.1 INI File Settings . . . 570
9.9.2 HAL Контакты . . . 571
9.9.2.1 Per-Axis Motion HAL Pins . . . 571
9.9.2.2 Other Motion HAL Pins . . . 571
9.9.3 Использование . . . 571
9.9.3.1 Offset Computation . . . 571
9.9.3.2 Machine-off/Machine-on . . . 572
9.9.3.3 Програмные пределы . . . 572
9.9.3.4 Notes . . . 572
9.9.3.5 Warning . . . 573
9.9.4 Related HAL Components . . . 573
9.9.4.1 eoffset_per_angle.comp . . . 573
9.9.5 Testing . . . 573
9.9.6 Examples . . . 574
9.9.6.1 eoffsets.ini . . . 574
9.9.6.2 jwp_z.ini . . . 574
9.9.6.3 dynamic_offsets.ini . . . 574
9.9.6.4 opa.ini (eoffset_per_angle) . . . 575
9.10Tool Database Interface . . . 575
9.10.1Interface . . . 575
9.10.1.1INI file Settings . . . 575
9.10.1.2**db_program** operation (v2.1) . . . 575
9.10.1.3Использование . . . 576
9.10.1.4Example program . . . 577
9.10.1.5Python tooldb module . . . 578
9.10.2Конфигурации моделирования . . . 578
9.10.2.1Notes . . . 579

**II Использование** **580**

**10 Интерфейсы пользователя** **581**

10.1 AXIS ГИП . . . 581
10.1.1 Введение . . . 581
10.1.2 Начало . . . 582
10.1.2.1 настройки INI . . . 583
10.1.2.2 Типичная сессия . . . 583
10.1.3 Окно AXIS . . . 584
10.1.3.1 Пункты меню . . . 584
10.1.3.2 Кнопки панели инструментов . . . 588
10.1.3.3 Область графического отображения . . . 589
10.1.3.4 Область отображения текста . . . 590
10.1.3.5 Ручное управление . . . 591
10.1.3.6 MDI . . . 593
10.1.3.7 Переопределение подачи . . . 594
10.1.3.8 Spindle Speed Override . . . 594
10.1.3.9 Скорость медленной подачи . . . 594
10.1.3.10 Макс скорость . . . 595
10.1.4 Управление клавиатурой . . . 595
10.1.4.1 Клавиши переопределения подачи . . . 595
10.1.5 Показать статус LinuxCNC (`linuxcnctop`) . . . 596
10.1.6 MDI интерфейс . . . 597
10.1.7 axis-remote . . . 598
10.1.8 Manual Tool Change . . . 598
10.1.9 Python Модули . . . 598
10.1.10 Использование AXIS в режиме токарного станка . . . 599
10.1.11 Использование AXIS в режиме резки вспененого материала . . . 602
10.1.12 Расширенная конфигурация . . . 603
10.1.12.1 Программные фильтры . . . 604
10.1.12.2 База данных ресурсов X (X-windows) . . . 605
10.1.12.3 Безупорный регулятор . . . 605
10.1.12.4 ~/.axisrc . . . 605
10.1.12.5 USER_COMMAND_FILE . . . 606
10.1.12.6 user_live_update() . . . 606
10.1.12.7 user_hal_pins() . . . 606
10.1.12.8 Внешний редактор . . . 606
10.1.12.9 Виртуальная панель управления . . . 606
10.1.12.10 Управление предварительным просмотром . . . 606

10.1.12.1 Touch Off using Actual Position . . . 607
10.1.13 Axisui . . . 607
10.1.14 Советы по настройке AXIS . . . 608
10.1.14.1 Функция обновления . . . 608
10.1.14.2 Отключить диалог закрытия . . . 608
10.1.14.3 Изменить шрифт текста . . . 609
10.1.14.4 Измените быструю скорость с помощью сочетаний клавиш . . . 609
10.1.14.5 Чтение INI-файла . . . 609
10.1.14.6 Чтение статуса LinuxCNC . . . 610
10.1.14.7 Изменить текущий вид . . . 610
10.1.14.8 Создание новых контактов AXISUI HAL . . . 610
10.1.14.9 Создание нового HAL компонента и контактов . . . 610
10.1.14.10 Переключение вкладок с помощью контактов HAL . . . 610
10.1.14.11 Добавить кнопку GOTO Home . . . 611
10.1.14.12 Добавить кнопку в ручной фрейм . . . 611
10.1.14.13 Чтение внутренних переменных . . . 612
10.1.14.14 Скрыть виджеты . . . 613
10.1.14.15 Изменить ярлык . . . 613
10.1.14.16 Перенаправить существующую команду . . . 613
10.1.14.17 Изменить цвет УЦИ . . . 614
10.1.14.18 Изменение кнопок панели инструментов . . . 614
10.1.14.19 Изменение цветов отрисовки . . . 615
10.2 GMOCCAPY . . . 615
10.2.1 Введение . . . 615
10.2.2 Requirements . . . 616
10.2.3 Как получить GMOCCAPY . . . 616
10.2.4 Базовая конфигурация . . . 617
10.2.4.1 Раздел DISPLAY . . . 618
10.2.4.2 Раздел TRAJ . . . 620
10.2.4.3 Кнопки макроса . . . 620
10.2.4.4 Встроенные вкладки и панели . . . 622
10.2.4.5 Сообщения, созданные пользователем . . . 625
10.2.4.6 Управление предварительным просмотром . . . 626
10.2.4.7 Пользовательский командный файл . . . 626
10.2.4.8 Пользовательский CSS файл . . . 627
10.2.4.9 Ведение журнала . . . 627
10.2.5 HAL Контакты . . . 628
10.2.5.1 Списки правых и нижних кнопок . . . 628
10.2.5.2 Скорости и переназначения . . . 631

10.2.5.3 HAL Контакты медленной подачи . . . 634
10.2.5.4 Скорость медленной подачи и контакт HAL Turtle-Jog . . . 634
10.2.5.5 HAL Контакты приращения медленной подачи . . . 635
10.2.5.6 Контакт разблокировки оборудования . . . 635
10.2.5.7 Контакты ошибок/предупреждений . . . 635
10.2.5.8 HAL Контакты созданного пользователем сообщения . . . 636
10.2.5.9 Контакты обратной связи шпинделя . . . 636
10.2.5.10 Контакты для отображения информации о ходе выполнения программы 637
10.2.5.11 Контакты, относящиеся к инструменту . . . 637
10.2.6 Автоматическое измерение инструмента . . . 638
10.2.6.1 Предоставленные контакты . . . 639
10.2.6.2 Изменения INI-файла . . . 640
10.2.6.3 Необходимые файлы . . . 641
10.2.6.4 Необходимые соединения HAL . . . 641
10.2.7 Страница настроек . . . 641
10.2.7.1 Внешний вид . . . 642
10.2.7.2 Behaviour . . . 646
10.2.7.3 Оборудование . . . 650
10.2.7.4 Расширенные настройки . . . 652
10.2.8 Тема иконок . . . 653
10.2.8.1 Пользовательская тема значков . . . 654
10.2.8.2 Символические иконки . . . 655
10.2.9 Специальный раздел для токарных станков . . . 655
10.2.10 Специальный раздел плазмы . . . 658
10.2.11 Видео на YouTube . . . 658
10.2.11.1 Базовое использование . . . 659
10.2.11.2 Имитация маховичков медленной подачи . . . 659
10.2.11.3 Страница настроек . . . 659
10.2.11.4 Имитированная аппаратная кнопка . . . 659
10.2.11.5 Пользовательские вкладки . . . 659
10.2.11.6 Видео об измерениях инструментов . . . 659
10.2.12 Известные проблемы . . . 659
10.2.12.1 Странные числа в информационной зоне . . . 659
10.2.12.2 Не завершающийся макрос . . . 660
10.3 Touchy ГИП . . . 660
10.3.1 Panel Configuration . . . 661
10.3.1.1 HAL connections . . . 661
10.3.1.2 Recommended for any setup . . . 662
10.3.2 Setup . . . 662

10.3.2.1 Enabling Touchy . . . 662
10.3.2.2 Preferences . . . 662
10.3.2.3 Macros . . . 663
10.4 Gscreen . . . 663
10.4.1 Введение . . . 663
10.4.1.1 Glade File . . . 668
10.4.1.2 PyGTK . . . 668
10.4.2 GladeVCP . . . 669
10.4.2.1 Обзор . . . 669
10.4.2.2 Build a GladeVCP Panel . . . 670
10.4.3 Building a simple clean-sheet custom screen . . . 671
10.4.4 Handler file example . . . 673
10.4.4.1 Adding Keybindings Functions . . . 674
10.4.4.2 Linuxcnc State Status . . . 674
10.4.4.3 Jogging Keys . . . 675
10.4.5 Gscreen Start Up . . . 675
10.4.6 INI Settings . . . 677
10.4.7 User Dialog Messages . . . 677
10.4.7.1 Copy the Stock Handler/Glade File For Modification . . . 678
10.5 QtDragon GUI . . . 679
10.5.1 Введение . . . 679
10.5.1.1 QtDragon . . . 679
10.5.1.2 QtDragon_lathe . . . 680
10.5.1.3 QtDragon_hd . . . 681
10.5.1.4 QtDragon_hd_vertical . . . 681
10.5.2 Начало работы — INI-файл . . . 681
10.5.2.1 Display . . . 682
10.5.2.2 Preferences . . . 682
10.5.2.3 Ведение журнала . . . 682
10.5.2.4 Override controls . . . 682
10.5.2.5 Spindle controls . . . 682
10.5.2.6 Jogging increments . . . 683
10.5.2.7 Grid Increments . . . 683
10.5.2.8 Jog speed . . . 683
10.5.2.9 User message dialog system . . . 683
10.5.2.10 Embed Custom VCP Panels . . . 684
10.5.2.11 Subroutine Paths . . . 685
10.5.2.12 Управление предварительным просмотром . . . 686
10.5.2.13 Program Extensions/Filters . . . 686

10.5.2.14 Probe/Touchplate/Laser Settings . . . 686
10.5.2.15 Abort detection . . . 687
10.5.2.16 Startup codes . . . 687
10.5.2.17 Кнопки макроса . . . 687
10.5.2.18 Post GUI HAL File . . . 688
10.5.2.19 Post GUI HAL Command . . . 688
10.5.2.20 HAL Bridge . . . 688
10.5.2.21 Builtin Sample Configurations . . . 689
10.5.3 Key Bindings . . . 689
10.5.4 Кнопки . . . 689
10.5.5 Virtual Keyboard . . . 689
10.5.6 HAL Контакты . . . 690
10.5.7 HAL files . . . 691
10.5.8 Manual Tool Changes . . . 691
10.5.9 Spindle . . . 692
10.5.10 Auto Raise Z Axis on Program Pause . . . 692
10.5.11 Z level compensation . . . 693
10.5.11.1 Using G-code Ripper for Z level Compensation . . . 694
10.5.12 Probing . . . 696
10.5.12.1 Versa Probe . . . 697
10.5.12.2 Basic probe . . . 699
10.5.12.3 Customizing Probe Screen Widget . . . 702
10.5.13 Touch plate . . . 702
10.5.14 Автоматическое измерение инструмента . . . 703
10.5.14.0 Обзор . . . 703
10.5.14.2 Обзор рабочего процесса . . . 703
10.5.14.3 Подробный пример рабочего процесса . . . 705
10.5.14.4 Work Piece Height Probing in QtDragon_hd . . . 706
10.5.14.5 Work Piece Height Probing . . . 707
10.5.14.6 Tool Measurement Pins . . . 708
10.5.14.7 Tool Measurement INI File Modifications . . . 709
10.5.14.8 Required HAL Connections . . . 711
10.5.15 Run from Line . . . 711
10.5.16 Laser buttons . . . 711
10.5.17 Tabs Description . . . 711
10.5.17.1 Main tab . . . 711
10.5.17.2 File Tab . . . 712
10.5.17.3 Offsets Tab . . . 712
10.5.17.4 Tool Tab . . . 712

10.5.17.1 Status Tab . . . 712
10.5.17.2 Probe Tab . . . 712
10.5.17.3 Camview Tab . . . 712
10.5.17.4 G-codes Tab . . . 713
10.5.17.5 Setup Tab . . . 713
10.5.17.6 Settings Tab . . . 714
10.5.17.7 Utilities Tab . . . 714
10.5.17.8 User Tab . . . 715
10.5.18 Styles . . . 715
10.5.19 Internationalisation . . . 715
10.5.20 Customization . . . 716
10.5.20.1 Stylesheets . . . 717
10.5.20.2 Qt Designer and Python code . . . 719
10.6 NGCGUI . . . 721
10.6.1 Обзор . . . 721
10.6.2 Demonstration Configurations . . . 722
10.6.3 Library Locations . . . 724
10.6.4 Standalone Usage . . . 725
10.6.4.1 Standalone NGCGUI . . . 725
10.6.4.2 Standalone PyNGCGUI . . . 725
10.6.5 Embedding NGCGUI . . . 726
10.6.5.1 Embedding NGCGUI in AXIS . . . 726
10.6.5.2 Embedding PyNGCGUI as a GladeVCP tab page in a GUI . . . 727
10.6.5.3 Additional INI File items required for NCGUI or PyNGCGUI . . . 727
10.6.5.4 Truetype Tracer . . . 729
10.6.5.5 INI File Path Specifications . . . 729
10.6.5.6 Summary of INI File item details for NGCGUI usage . . . 730
10.6.6 File Requirements for NGCGUI Compatibility . . . 732
10.6.6.1 Single-File Gcode (.ngc) Subroutine Requirements . . . 732
10.6.6.2 Gcode-meta-compiler (.gcmc) file requirements . . . 734
10.6.7 DB25 Example . . . 735
10.6.8 Creating a subroutine . . . 738
10.7 TkLinuxCNC GUI . . . 739
10.7.1 Введение . . . 739
10.7.2 Начало . . . 739
10.7.2.1 A typical session with TkLinuxCNC . . . 739
10.7.3 Elements of the TkLinuxCNC window . . . 740
10.7.3.1 Main buttons . . . 740
10.7.3.2 Offset display status bar . . . 741

10.7.3.3Coordinate Display Area . . . 741
10.7.3.4TkLinuxCNC Interpreter / Automatic Program Control . . . 741
10.7.3.5Ручное управление . . . 741
10.7.3.6Code Entry . . . 742
10.7.3.7Скорость медленной подачи . . . 743
10.7.3.8Переопределение подачи . . . 743
10.7.3.9Spindle speed Override . . . 743
10.7.4Управление клавиатурой . . . 743
10.8QtPlasmaC . . . 744
10.8.1Preamble . . . 744
10.8.2License . . . 744
10.8.3Введение . . . 744
10.8.4Установка LinuxCNC . . . 747
10.8.4.1If The User Does Not Have Linux Installed . . . 748
10.8.4.2Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) . . . 748
10.8.4.3Package Installation (Buildbot) If The User Has Linux on Debian 12 (Bookworm) or Debian 11 (Bullseye) . . . 748
10.8.4.4Run In Place Installation If The User Has Linux Installed . . . 748
10.8.5Creating A QtPlasmaC Configuration . . . 748
10.8.5.1Modes . . . 748
10.8.5.2Available I/Os . . . 749
10.8.5.3Recommended Settings: . . . 750
10.8.5.4Configuring . . . 751
10.8.5.5Qt Dependency Errors . . . 756
10.8.5.6Initial Setup . . . 756
10.8.6Migrating to QtPlasmaC From PlasmaC (AXIS or GMOCCAPY) . . . 760
10.8.7Other QtPlasmaC Setup Considerations . . . 760
10.8.7.1Фильтр нижних частот . . . 760
10.8.7.2Contact Bounce . . . 760
10.8.7.3Contact Load . . . 761
10.8.7.4Desktop Launcher . . . 762
10.8.7.5QtPlasmaC Files . . . 762
10.8.7.6INI File . . . 763
10.8.8QtPlasmaC GUI Overview . . . 765
10.8.8.1Exiting QtPlasmaC . . . 765
10.8.8.2MAIN Tab . . . 765
10.8.8.3Preview Views . . . 772
10.8.8.4CONVERSATIONAL Tab . . . 772

10.8.8.5 PARAMETERS Tab . . . 773
10.8.8.6 SETTINGS Tab . . . 779
10.8.8.7 STATISTICS Tab . . . 783
10.8.9 Using QtPlasmaC . . . 783
10.8.9.1 Units Systems . . . 784
10.8.9.2 Preamble and Postamble Codes . . . 784
10.8.9.3 Mandatory Codes . . . 785
10.8.9.4 Coordinates . . . 785
10.8.9.5 Cut Feed Rate . . . 785
10.8.9.6 Material File . . . 785
10.8.9.7 Manual Material Handling . . . 787
10.8.9.8 Automatic Material Handling . . . 787
10.8.9.9 Material Addition Via Magic Comments In G-code . . . 788
10.8.9.10 Material Converter . . . 789
10.8.9.11 LASER . . . 792
10.8.9.12 CAMERA . . . 794
10.8.9.13 Path Tolerance . . . 795
10.8.9.14 Paused Motion . . . 796
10.8.9.15 Pause At End Of Cut . . . 796
10.8.9.16 Multiple Tools . . . 796
10.8.9.17 Velocity Reduction . . . 797
10.8.9.18 THC (Torch Height Controller) . . . 798
10.8.9.19 Cutter Compensation . . . 799
10.8.9.20 Initial Height Sense (IHS) Skip . . . 799
10.8.9.21 Probing . . . 800
10.8.9.22 Offset Probing . . . 800
10.8.9.23 Cut Types . . . 801
10.8.9.24 Hole Cutting - Intro . . . 802
10.8.9.25 Hole Cutting . . . 802
10.8.9.26 Hole Cutting - Automatic . . . 804
10.8.9.27 Single Cut . . . 805
10.8.9.28 Thick Materials . . . 807
10.8.9.29 Mesh Mode (Expanded Metal Cutting) . . . 807
10.8.9.30 Ignore Arc OK . . . 808
10.8.9.31 Cut Recovery . . . 808
10.8.9.32 Run From Line . . . 809
10.8.9.33 Scribe . . . 811
10.8.9.34 Spotting . . . 813
10.8.9.35 Tube Cutting . . . 814

10.8.9.3.6 Virtual Keyboard Custom Layouts . . . 814
10.8.9.3.7 Keyboard Shortcuts . . . 815
10.8.9.3.8 MDI . . . 816
10.8.10 Conversational Shape Library . . . 817
10.8.10.1 Conversational Settings . . . 819
10.8.10.2 Conversational Lines And Arcs . . . 819
10.8.10.3 Conversational Single Shape . . . 820
10.8.10.4 Conversational Group Of Shapes . . . 822
10.8.10.5 Conversational Block . . . 822
10.8.10.6 Conversational Saving A Job . . . 823
10.8.11 Error Messages . . . 824
10.8.11.1 Error Logging . . . 824
10.8.11.2 Error Message Display . . . 824
10.8.11.3 Critical Errors . . . 824
10.8.11.4 Warning Messages . . . 826
10.8.12 Updating QtPlasmaC . . . 827
10.8.12.1 Standard Update . . . 827
10.8.12.2 Continuous Update . . . 827
10.8.13 Modify An Existing QtPlasmaC Configuration . . . 827
10.8.14 Customizing QtPlasmaC GUI . . . 827
10.8.14.1 Add A Custom Style . . . 828
10.8.14.2 Create A New Style . . . 828
10.8.14.3 Returning To The Default Styling . . . 829
10.8.14.4 Custom Python Code . . . 829
10.8.14.5 Custom G-code Filter . . . 830
10.8.15 QtPlasmaC Advanced Topics . . . 831
10.8.15.1 Custom User Buttons . . . 831
10.8.15.2 Peripheral Offsets (Laser, Camera, Scribe, Offset Probe) . . . 838
10.8.15.3 Keep Z Motion . . . 840
10.8.15.4 External HAL Pins . . . 840
10.8.15.5 Hide Program Buttons . . . 841
10.8.15.6 Tuning Mode 0 Arc OK . . . 842
10.8.15.7 Lost Arc Delay . . . 842
10.8.15.8 Zero Window . . . 843
10.8.15.9 Tuning Void Sensing . . . 843
10.8.15.10 Max Offset . . . 843
10.8.15.11 Enable Tabs During Automated Motion . . . 843
10.8.15.12 Override Jog Inhibit Via Z+ Jog . . . 844
10.8.15.13 QtPlasmaC State Outputs . . . 844

10.8.15.4 QtPlasmaC Debug Print . . . 845
10.8.15.5 Hypertherm PowerMax Communications . . . 845
10.8.15.6 Moving Pierce . . . 846
10.8.16 Internationalisation . . . 849
10.8.17 Appendix . . . 851
10.8.17.1 Примеры конфигураций . . . 851
10.8.17.2 NGC Samples . . . 851
10.8.17.3 QtPlasmaC Specific G-codes . . . 851
10.8.17.4 QtPlasmaC G-code Examples . . . 852
10.8.17.5 Mesa THCAD . . . 855
10.8.17.6 RS485 Connections . . . 856
10.8.17.7 Arc OK With A Reed Relay . . . 858
10.8.17.8 Contact Load Schematics . . . 860
10.8.18 Known Issues . . . 860
10.8.18.1 Keyboard Jogging . . . 860
10.8.18.2 NO_FORCE_HOMING . . . 861
10.8.19 Contributing Code To QtPlasmaC . . . 861
10.8.20 Support . . . 862
10.9 MDRO GUI . . . 862
10.9.1 Введение . . . 862
10.9.2 Начало . . . 863
10.9.2.1 INI File Options . . . 863
10.9.2.2 Command Line Options . . . 864
10.9.2.3 Контакты . . . 864
10.9.3 MDRO Window . . . 864
10.9.4 Index operations . . . 865
10.9.5 Simulation . . . 865

**11 Программирование в G-кодах 866**
11.1 Coordinate Systems . . . 866
11.1.1 Введение . . . 866
11.1.2 Machine Coordinate System . . . 866
11.1.2.1 Machine coordinates moves: G53 . . . 866
11.1.3 Coordinate Systems . . . 867
11.1.3.1 Default Coordinate System . . . 868
11.1.3.2 Setting Coordinate System Offsets . . . 869
11.1.4 Local and Global Offsets . . . 869
11.1.4.1 The G52 command . . . 869
11.1.5 G92 Axes Offsets . . . 870

11.1.5.1The G92 commands . . . 870
11.1.5.2Setting G92 Values . . . 871
11.1.5.3G92 Persistence Cautions . . . 871
11.1.5.4G92 and G52 Interaction Cautions . . . 872
11.1.6Sample Programs Using Offsets . . . 872
11.1.6.1Sample Program Using Workpiece Coordinate Offsets . . . 872
11.1.6.2Sample Program Using G52 Offsets . . . 873
11.2Компенсация инструмента . . . 873
11.2.1Касание . . . 873
11.2.1.1Using G10 L1/L10/L11 . . . 874
11.2.2Tool Table . . . 874
11.2.2.1Tool Table Format . . . 875
11.2.2.2Инструментальный ввод/вывод . . . 876
11.2.2.3Устройства смены инструмента . . . 878
11.2.3Компенсация длины инструмента . . . 878
11.2.4Cutter Radius Compensation . . . 879
11.2.4.1Обзор . . . 880
11.2.4.2Examples . . . 882
11.3Tool Edit GUI . . . 883
11.3.1Обзор . . . 883
11.3.2Column Sorting . . . 884
11.3.3Columns Selection . . . 885
11.3.4Stand Alone Use . . . 885
11.4Обзор программирования G-кода . . . 886
11.4.1Обзор . . . 886
11.4.2Format of a line . . . 886
11.4.2.1/: Block Delete . . . 887
11.4.2.2Необязательный номер строки . . . 887
11.4.2.3Words, Parameters, Subroutines, Comments . . . 887
11.4.2.4End of Line Marker . . . 888
11.4.3Числа . . . 888
11.4.4Параметры . . . 889
11.4.4.1Пронумерованные параметры . . . 890
11.4.4.2Коды и параметры подпрограмм . . . 892
11.4.4.3Именованные параметры . . . 892
11.4.4.4Предопределенные именованные параметры . . . 893
11.4.4.5Параметры системы . . . 895
11.4.5HAL Контакты и INI значения . . . 895
11.4.6Expressions . . . 896

11.4.7 Бинарные операторы . . . 896
11.4.8 Equality and floating-point values . . . 897
11.4.9 Функции . . . 897
11.4.10 Repeated Items . . . 898
11.4.11 Item order . . . 898
11.4.12 Commands and Machine Modes . . . 899
11.4.13 Полярные координаты . . . 899
11.4.14 Modal Groups . . . 901
11.4.15 Comments . . . 902
11.4.16 Messages . . . 903
11.4.17 Ведение журнала зонда . . . 903
11.4.18 Ведение журнала . . . 903
11.4.19 Сообщения об отмене . . . 904
11.4.20 Debug Messages . . . 904
11.4.21 Print Messages . . . 904
11.4.22 Параметры комментариев . . . 904
11.4.23 Требования к файлам . . . 905
11.4.24 Размер файла . . . 905
11.4.25 G-code Order of Execution . . . 905
11.4.26 G-code Best Practices . . . 906
11.4.27 Linear and Rotary Axis . . . 907
11.4.28 Common Error Messages . . . 907
11.5 G-Codes . . . 907
11.5.1 Conventions . . . 907
11.5.2 G-Code Quick Reference Table . . . 908
11.5.3 G0 Rapid Move . . . 909
11.5.3.1 Rapid Velocity Rate . . . 909
11.5.4 G1 Линейное перемещение . . . 910
11.5.5 G2, G3 Arc Move . . . 910
11.5.5.1 Center Format Arcs . . . 911
11.5.5.2 Center Format Examples . . . 913
11.5.5.3 Radius Format Arcs . . . 914
11.5.6 G4 Dwell . . . 915
11.5.7 G5 Cubic Spline . . . 915
11.5.8 G5.1 Quadratic Spline . . . 916
11.5.9 G5.2 G5.3 NURBS Block . . . 917
11.5.10 Режим диаметра токарного станка G7 . . . 918
11.5.11 Радиусный режим токарного станка G8 . . . 918
11.5.12 G10 L0 Перезагрузить данные таблицы инструментов . . . 919

11.5.15 G10 L1 Таблица набора инструментов . . . 919
11.5.16 G10 L2 Set Coordinate System . . . 920
11.5.17 G10 L10 Таблица набора инструментов . . . 921
11.5.18 G10 L11 Таблица набора инструментов . . . 922
11.5.19 G10 L20 Set Coordinate System . . . 922
11.5.20 G17 - G19.1 Plane Select . . . 923
11.5.21 G20, G21 Units . . . 923
11.5.22 G28, G28.1 Go/Set Predefined Position . . . 923
11.5.23 G30, G30.1 Go/Set Predefined Position . . . 924
11.5.24 G33 Spindle Synchronized Motion . . . 924
11.5.25 G33.1 Rigid Tapping . . . 925
11.5.26 G38.*n* Straight Probe . . . 926
11.5.27 G40 Compensation Off . . . 928
11.5.28 G41, G42 компенсация режущего инструмента . . . 928
11.5.29 G41.1, G42.1 Динамическая компенсация режущего инструмента . . . 929
11.5.30 G43 смещение длины инструмента . . . 929
11.5.31 G43.1 Динамическое смещение длины инструмента . . . 930
11.5.32 G43.2 Apply additional Tool Length Offset . . . 931
11.5.33 G49 отмена смещения длины инструмента . . . 931
11.5.34 G52 Local Coordinate System Offset . . . 932
11.5.35 G53 Move in Machine Coordinates . . . 932
11.5.36 G54-G59.3 Select Coordinate System . . . 932
11.5.37 G61 Exact Path Mode . . . 933
11.5.38 G61.1 Exact Stop Mode . . . 933
11.5.39 G64 Path Blending . . . 933
11.5.40 G70 Lathe finishing cycle . . . 936
11.5.41 G71 G72 Lathe roughing cycles . . . 937
11.5.42 G73 Drilling Cycle with Chip Breaking . . . 938
11.5.43 G74 Left-hand Tapping Cycle with Dwell . . . 939
11.5.44 G76 цикл нарезания резьбы . . . 940
11.5.45 G80-G89 Canned Cycles . . . 942
11.5.43.1 Common Words . . . 943
11.5.43.2 Sticky Words . . . 943
11.5.43.3 Repeat Cycle . . . 943
11.5.43.4 Retract Mode . . . 943
11.5.43.5 Canned Cycle Errors . . . 943
11.5.43.6 Preliminary and In-Between Motion . . . 944
11.5.43.7 Why use a canned cycle? . . . 944
11.5.46 G80 Cancel Canned Cycle . . . 946

11.5.45 G81 Drilling Cycle . . . 947
11.5.46 G82 цикл сверления,задержка . . . 952
11.5.47 G83 Цикл сверления с частичным повторением импульсов . . . 952
11.5.48 G84 Right-hand Tapping Cycle, Dwell . . . 953
11.5.49 G85 Цикл растачивания, подача . . . 953
11.5.50 G86 цикл растачивания, остановка шпинделя, быстрый выход . . . 954
11.5.51 G87 цикл обратного растачивания . . . 954
11.5.52 G88 цикл растачивания, остановка шпинделя, ручной отвод . . . 954
11.5.53 G89 цикл развертывания, задержка, вывод на рабочей подаче . . . 954
11.5.54 G90, G91 Distance Mode . . . 954
11.5.55 G90.1, G91.1 Arc Distance Mode . . . 955
11.5.56 G92 Coordinate System Offset . . . 955
11.5.57 G92.1, G92.2 Reset G92 Offsets . . . 956
11.5.58 G92.3 Restore G92 Offsets . . . 956
11.5.59 G93, G94, G95 Feed Rate Mode . . . 956
11.5.60 G96, G97 Spindle Control Mode . . . 957
11.5.61 G98, G99 уровень возврата стандартного цикла . . . 957
11.6 M-Codes . . . 958
11.6.1 M-Code Quick Reference Table . . . 958
11.6.2 M0, M1 Program Pause . . . 958
11.6.3 M2, M30 конец программы . . . 958
11.6.4 M60 Pallet Change Pause . . . 959
11.6.5 M3, M4, M5 Spindle Control . . . 959
11.6.6 M6 Смена инструмента . . . 960
11.6.6.1 Manual Tool Change . . . 960
11.6.6.2 Tool Changer . . . 960
11.6.7 M7, M8, M9 Coolant Control . . . 960
11.6.8 M19 Orient Spindle . . . 961
11.6.9 M48, M49 Speed and Feed Override Control . . . 961
11.6.10 M50 Управление переопределения подачи . . . 962
11.6.11 M51 Управление переопределения скорости шпинделя . . . 962
11.6.12 M52 Адаптивное управление подачей . . . 962
11.6.13 M53 Контроль скорости подачи . . . 962
11.6.14 M61 Set Current Tool . . . 962
11.6.15 M62 - M65 Управление цифрового вывода . . . 963
11.6.16 M66 Wait on Input . . . 963
11.6.17 M67 Analog Output, Synchronized . . . 964
11.6.18 M68 Analog Output, Immediate . . . 964
11.6.19 M70 Save Modal State . . . 965

11.6.20 M71 Invalidate Stored Modal State . . . 966
11.6.21 M72 Restore Modal State . . . 966
11.6.22 M73 Save and Autorestore Modal State . . . 967
11.6.23 M98 and M99 . . . 967
11.6.23.1 Selectively Restoring Modal State . . . 968
11.6.24 M100-M199 определяемые пользователем команды . . . 968
11.7 O Коды . . . 970
11.7.1 Use of O-codes . . . 970
11.7.2 Numbering . . . 970
11.7.3 Comments . . . 971
11.7.4 Subroutines . . . 971
11.7.4.1 Fanuc-Style Numbered Programs . . . 972
11.7.5 Цикл . . . 974
11.7.6 Условный . . . 974
11.7.7 Repeat . . . 975
11.7.8 Indirection . . . 975
11.7.9 Вызов файлов . . . 976
11.7.10 Subroutine return values . . . 976
11.7.11 Ошибки . . . 977
11.8 Другие коды . . . 977
11.8.1 F: Set Feed Rate . . . 977
11.8.2 S: Set Spindle Speed . . . 977
11.8.3 T: Выбрать инструмент . . . 978
11.9 G-Code Examples . . . 978
11.9.1 Mill Examples . . . 978
11.9.1.1 Helical Hole Milling . . . 978
11.9.1.2 Slotting . . . 979
11.9.1.3 Grid Probe . . . 979
11.9.1.4 Smart Probe . . . 980
11.9.1.5 Tool Length Probe . . . 981
11.9.1.6 Hole Probe . . . 981
11.9.1.7 Cutter Compensation . . . 981
11.9.2 Lathe Examples . . . 982
11.9.2.1 Threading . . . 982
11.10 Image to G-Code . . . 982
11.10.1 What is a depth map? . . . 982
11.10.2 Integrating image-to-gcode with the AXIS user interface . . . 983
11.10.3 Using `image-to-gcode` . . . 983
11.10.4 Option Reference . . . 983

11.10.4.1 Units . . . 983
11.10.4.2 Invert Image . . . 983
11.10.4.3 Normalize Image . . . 983
11.10.4.4 Expand Image Border . . . 983
11.10.4.5 Tolerance (units) . . . 983
11.10.4.6 Pixel Size (units) . . . 984
11.10.4.7 Plunge Feed Rate (units per minute) . . . 984
11.10.4.8 Feed Rate (units per minute) . . . 984
11.10.4.9 Spindle Speed (RPM) . . . 984
11.10.4.10 Scan Pattern . . . 984
11.10.4.11 Scan Direction . . . 984
11.10.4.12 Depth (units) . . . 984
11.10.4.13 Step Over (pixels) . . . 985
11.10.4.14 Tool Diameter . . . 985
11.10.4.15 Safety Height . . . 985
11.10.4.16 Tool Type . . . 985
11.10.4.17 Face bounding . . . 985
11.10.4.18 Contact angle . . . 985
11.10.4.19 Roughing offset and depth per pass . . . 986
11.11 RS274/NGC Differences . . . 986
11.11.1 Changes from RS274/NGC . . . 986
11.11.2 Additions to RS274/NGC . . . 987

**12 Виртуальные панели управления 989**
12.1 PyVCP . . . 989
12.1.1 Введение . . . 989
12.1.2 Устройство панели . . . 990
12.1.3 Безопасность . . . 991
12.1.4 AXIS . . . 991
12.1.4.1 Панель для примера . . . 991
12.1.5 Автономность . . . 993
12.1.6 Widgets . . . 994
12.1.6.1 Syntax . . . 994
12.1.6.2 Главные примечания . . . 994
12.1.6.3 Label . . . 995
12.1.6.4 Мульти_метка . . . 996
12.1.6.5 СИДы . . . 996
12.1.6.6 Кнопки . . . 998
12.1.6.7 Числовые индикаторы . . . 1000

12.1.6.8 Ввод чисел . . . 1003
12.1.6.9 Изображения . . . 1006
12.1.6.10 Контейнеры . . . 1008
12.2 Примеры PyVCP . . . 1013
12.2.1 AXIS . . . 1013
12.2.2 Плавающие панели . . . 1013
12.2.3 Пример кнопок подачи . . . 1014
12.2.3.1 Создайте виджеты . . . 1015
12.2.3.2 Сделайте соеденения . . . 1017
12.2.4 Тестер порта . . . 1018
12.2.5 GS2 измеритель оборотов . . . 1021
12.2.5.1 Панель . . . 1021
12.2.5.2 Соединения . . . 1023
12.2.6 Кнопка быстрого приведения в исходное положение . . . 1023
12.3 GladeVCP: Виртуальная панель управления . . . 1025
12.3.1 Что такое GladeVCP? . . . 1025
12.3.1.1 Краткий обзор PyVCP и GladeVCP . . . 1025
12.3.2 Краткий тур с примером панели . . . 1026
12.3.2.1 Изучение примера панели . . . 1029
12.3.2.2 Изучение описания пользовательского интерфейса . . . 1030
12.3.2.3 Изучение обратного вызова Python . . . 1030
12.3.3 Создание и интеграция пользовательского интерфейса Glade . . . 1030
12.3.3.1 Обязательное условие: установка Glade . . . 1030
12.3.3.2 Запуск Glade для создания нового пользовательского интерфейса . . . 1031
12.3.3.3 Тестирование панели . . . 1032
12.3.3.4 Подготовка командного файла HAL . . . 1032
12.3.3.5 Интеграция в AXIS, например PyVCP . . . 1032
12.3.3.6 Встраивание в виде вкладки . . . 1033
12.3.3.7 Интеграция в Touchy . . . 1034
12.3.3.8 Loading Builtin Panels . . . 1034
12.3.4 Параметры командной строки GladeVCP . . . 1035
12.3.5 Понимание процесса запуска GladeVCP . . . 1035
12.3.6 Ссылка на виджет HAL . . . 1036
12.3.6.1 Именование виджетов и контактов HAL . . . 1037
12.3.6.2 Атрибуты Python и методы HAL Widgets . . . 1037
12.3.6.3 Установка значений контактов и виджетов . . . 1038
12.3.6.4 Сигнал hal-pin-changed . . . 1038
12.3.6.5 Кнопки . . . 1039
12.3.6.6 Шкалы . . . 1040

12.3.6.7 Окошко счетчика . . . 1040
12.3.6.8 Hal_Dial . . . 1040
12.3.6.9 Маховичок подачи . . . 1043
12.3.6.10 Контроль скорости . . . 1044
12.3.6.11 Label . . . 1047
12.3.6.12 Контейнеры . . . 1048
12.3.6.13 LED . . . 1048
12.3.6.14 ProgressBar . . . 1049
12.3.6.15 ComboBox . . . 1050
12.3.6.16 Шкалы . . . 1050
12.3.6.17 Meter . . . 1052
12.3.6.18 HAL_Graph . . . 1053
12.3.6.19 Предварительный просмотр пути инструмента Gremlin для файлов NGC 1053
12.3.6.20 HAL_Offset . . . 1056
12.3.6.21 Виджет УЦИ . . . 1057
12.3.6.22 Виджет Combi_DRO . . . 1059
12.3.6.23 IconView (Выбор файла) . . . 1062
12.3.6.24 Виджет калькулятора . . . 1065
12.3.6.25 Виджет редактора инструментов . . . 1066
12.3.6.26 Offsetpage . . . 1068
12.3.6.27 Виджет HAL_sourceview . . . 1070
12.3.6.28 История MDI . . . 1071
12.3.6.29 Анимированные функциональные диаграммы: виджеты HAL в растровом изображении . . . 1072
12.3.7 Справочник по виджетам действий . . . 1073
12.3.7.1 Виджеты VCP Action . . . 1073
12.3.7.2 VCP Action Python . . . 1073
12.3.7.3 Виджеты VCP ToggleAction . . . 1074
12.3.7.4 Виджеты Action_MDI Toggle и Action_MDI . . . 1075
12.3.7.5 Простой пример: выполнение команды MDI при нажатии кнопки . . . 1075
12.3.7.6 Передача параметров с помощью виджетов Action_MDI и ToggleAction_MDI 1076
12.3.7.7 Расширенный пример: передача параметров в подпрограмму с O-словом 1076
12.3.7.8 Подготовка к действию MDI и последующая очистка . . . 1077
12.3.7.9 Использование объекта LinuxCNC Stat для обработки изменений статуса 1078
12.3.8 Программирование GladeVCP . . . 1078
12.3.8.1 Действия, определяемые пользователем . . . 1078
12.3.8.2 Основная библиотека . . . 1079
12.3.8.3 Пример: добавление пользовательских обратных вызовов в Python . . . 1079
12.3.8.4 События изменения значения HAL . . . 1080

12.3.8.5 Модель программирования . . . 1080
12.3.8.6 Последовательность инициализации . . . 1082
12.3.8.7 Несколько обратных вызовов с одним и тем же именем . . . 1083
12.3.8.8 Флаг GladeVCP `-U <useropts>` . . . 1083
12.3.8.9 Постоянные переменные в GladeVCP . . . 1083
12.3.8.10 Использование постоянных переменных . . . 1084
12.3.8.11 Сохранение состояния при завершении работы GladeVCP . . . 1085
12.3.8.12 Сохранение состояния при нажатии Ctrl-C . . . 1085
12.3.8.13 Ручное редактирование файлов INI (.ini) . . . 1085
12.3.8.14 Добавление контактов HAL . . . 1086
12.3.8.15 Добавление таймеров . . . 1086
12.3.8.16 Программная настройка свойств виджета HAL . . . 1086
12.3.8.17 Обратный вызов с изменением значения с помощью hal_glib . . . 1087
12.3.8.18 Примеры и создание собственного приложения GladeVCP . . . 1087
12.3.9 FAQ . . . 1088
12.3.10 Решение проблем . . . 1089
12.3.11 Примечание по реализации: обработка ключей в AXIS . . . 1089
12.3.12 Добавление пользовательских виджетов . . . 1089
12.3.13 Вспомогательные приложения GladeVCP . . . 1089
12.4 Модули библиотеки GladeVCP . . . 1090
12.4.1 Информация . . . 1090
12.4.2 Действие . . . 1092
12.5 QtVCP . . . 1094
12.5.1 Showcase . . . 1094
12.5.2 Обзор . . . 1101
12.5.2.1 QtVCP Widgets . . . 1102
12.5.2.2 INI Settings . . . 1102
12.5.2.3 Qt Designer UI File . . . 1103
12.5.2.4 Handler Files . . . 1103
12.5.2.5 Libraries Modules . . . 1104
12.5.2.6 Themes . . . 1104
12.5.2.7 Local Files . . . 1104
12.5.2.8 Modifying Stock Screens . . . 1105
12.5.3 VCP Panels . . . 1108
12.5.3.1 Builtin Panels . . . 1108
12.5.3.2 Custom Panels . . . 1112
12.5.4 Build A Simple Clean-sheet Custom Screen . . . 1114
12.5.4.1 Обзор . . . 1114
12.5.4.2 Get Qt Designer To Include LinuxCNC Widgets . . . 1114

12.5.4.3 Build The Screen .ui File . . . 1115
12.5.4.4 Handler file . . . 1118
12.5.4.5 INI Configuration . . . 1118
12.5.5 Handler File In Detail . . . 1118
12.5.5.1 Обзор . . . 1119
12.5.5.2 IMPORT Section . . . 1122
12.5.5.3 INSTANTIATE LIBRARIES Section . . . 1122
12.5.5.4 HANDLER CLASS Section . . . 1122
12.5.5.5 INITIALIZE Section . . . 1122
12.5.5.6 SPECIAL FUNCTIONS Section . . . 1123
Bibliography . . . 1124
12.5.5.7 STATUS CALLBACKS Section . . . 1124
12.5.5.8 CALLBACKS FROM FORM Section . . . 1124
12.5.5.9 GENERAL FUNCTIONS Section . . . 1124
12.5.5.10 KEY BINDING Section . . . 1124
12.5.5.11 CLOSING EVENT Section . . . 1124
12.5.6 Connecting Widgets to Python Code . . . 1125
12.5.6.1 Обзор . . . 1125
12.5.6.2 Using Qt Designer to add Slots . . . 1125
12.5.6.3 Python Handler Changes . . . 1126
12.5.7 More Information . . . 1127
12.6 QtVCP Virtual Control Panels . . . 1127
12.6.1 Builtin Virtual Control Panels . . . 1127
12.6.1.1 copy . . . 1127
12.6.1.2 spindle_belts . . . 1128
12.6.1.3 test_dial . . . 1130
12.6.1.4 test_button . . . 1131
12.6.1.5 test_led . . . 1131
12.6.1.6 test_panel . . . 1132
12.6.1.7 cam_align . . . 1133
12.6.1.8 sim_panel . . . 1136
12.6.1.9 tool_dialog . . . 1137
12.6.2 vismach 3D Simulation Panels . . . 1137
12.6.2.1 QtVCP vismach_mill_xyz . . . 1137
12.6.2.2 QtVCP vismach_router_atc . . . 1138
12.6.2.3 QtVCP vismach_scara . . . 1139
12.6.2.4 QtVCP vismach_millturn . . . 1140
12.6.2.5 QtVCP vismach_mill_5axis_gantry . . . 1141
12.6.2.6 QtVCP vismach_fanuc_200f . . . 1142

12.6.3 Custom Virtual Control Panels . . . 1143
12.6.4 Embedding QtVCP Virtual Control Panels into QtVCP Screens . . . 1144
12.6.4.1 Встраивание команд . . . 1144
12.6.4.2 Location of builtin Panels . . . 1144
12.6.4.3 Location of Custom Panels . . . 1145
12.6.4.4 Советы по программированию обработчиков . . . 1145
12.6.4.5 Designer Widget Tips . . . 1145
12.6.4.6 Handler Patching - Subclassing Builtin Panels . . . 1145
12.7 QtVCP Widgets . . . 1146
12.7.1 HAL Only Widgets . . . 1147
12.7.1.1 `CheckBox` Widget . . . 1147
12.7.1.2 *DetachTabWidget* - Container Widget With User Detachable Panels . . . 1147
12.7.1.3 `DoubleScale` - Spin Button Entry Widget . . . 1147
12.7.1.4 `FocusOverlay` - Focus Overlay Widget . . . 1147
12.7.1.5 `Gauge` - Round Dial Gauge Widget . . . 1149
12.7.1.6 `GeneralHALInput` - General Signals/Slots Input Connection Widget . . . 1150
12.7.1.7 `GeneralHALOutput` - General Signals/Slots Output Connection Widget . . . 1150
12.7.1.8 `GridLayout` - Grid Layout Widget . . . 1151
12.7.1.9 *HalBar* - HAL Bar Level Indicator . . . 1151
12.7.1.10 `HALPad` - HAL Buttons Joypad . . . 1152
12.7.1.11 `HALLabel` - HAL Label Widget . . . 1154
12.7.1.12 `LCDNumber` - LCD Style Number Readout Widget . . . 1155
12.7.1.13 `LED` - Indicator Widget . . . 1155
12.7.1.14 `PushButton` - HAL Pin Toggle Widget . . . 1156
12.7.1.15 `RadioButton` Widget . . . 1156
12.7.1.16 `Slider` - HAL Pin Value Adjusting Widget . . . 1156
12.7.1.17 `TabWidget` - Tab Widget . . . 1157
12.7.1.18 `WidgetSwitcher` - Multi-widget Layout View Switcher Widget . . . 1157
12.7.1.19 `XEmbed` - Program Embedding Widget . . . 1158
12.7.2 Machine Controller Widgets . . . 1158
12.7.2.1 `ActionButton` - Machine Controller Action Control Widget . . . 1158
12.7.2.2 `ActionToolButton` - Optional Actions Menu Button Widget . . . 1161
12.7.2.3 `AxisToolButton` - Select and Set Axis Widget . . . 1161
12.7.2.4 `BasicProbe` - Simple Mill Probing Widget . . . 1163
12.7.2.5 `CamView` - Workpiece Alignment and Origin Setting Widget . . . 1163
12.7.2.6 `DROLabel` - Axis Position Display Widget . . . 1163
12.7.2.7 `FileManager` - File Loading Selector Widget . . . 1166
12.7.2.8 `GcodeDisplay` - G-code Text Display Widget . . . 1167
12.7.2.9 `GcodeEditor` - G-code Program Editor Widget . . . 1168

12.7.2.10 GCodeGraphics - G-code Graphic Backplot Widget . . . 1169
12.7.2.11 JointEnableWidget - FIXME . . . 1173
12.7.2.12 JogIncrements - Jog Increments Value Selection Widget . . . 1173
12.7.2.13 MacroTab - Special Macros Widget . . . 1174
12.7.2.14 OperatorValueLine - Operator Value Line Entry Widget . . . 1176
12.7.2.15 MDILine - MDI Commands Line Entry Widget . . . 1177
12.7.2.16 MDIHistory - MDI Commands History Widget . . . 1178
12.7.2.17 MDITouchy - Touch Screen MDI Entry Widget . . . 1179
12.7.2.18 OriginOffsetView - Origins View and Setting Widget . . . 1181
12.7.2.19 RadioAxisSelector - FIXME . . . 1182
12.7.2.20 RoundButton - Round Shapped ActionButton Widget . . . 1182
12.7.2.21 StateLabel - Controller Modes State Label Display Widget . . . 1182
12.7.2.22 StatusLabel - Controller Variables State Label Display Widget . . . 1183
12.7.2.23 StatusImageSwitcher - Controller Status Image Switcher . . . 1185
12.7.2.24 StatusStacked - Mode Status Display Switching Widget . . . 1187
12.7.2.25 ScreenOption - General Options Setting widget . . . 1187
12.7.2.26 StatusSlider - Controller Setting Adjustment Slider Widget . . . 1192
12.7.2.27 StateLED - Controller State LED Widget . . . 1193
12.7.2.28 StatusAdjustmentBar - Controller Value Setting Widget . . . 1194
12.7.2.29 SystemToolButton - User System Selection Widget . . . 1195
12.7.2.30 StateEnableGridlayout - Controller State Enabled Container Widget . . . 1195
12.7.2.31 StatusImageSwitcher - Controller Status Image Switching Widget . . . 1195
12.7.2.32 ToolOffsetView - Tools Offsets View And Edit Widget . . . 1196
12.7.2.33 VersaProbe - Mill Probing Widget . . . 1198
12.7.3 Dialog Widgets . . . 1198
12.7.3.1 LcncDialog - General Message Dialog Widget . . . 1199
12.7.3.2 ToolDialog - Manual Tool Change Dialog Widget . . . 1200
12.7.3.3 FileDialog - Load and Save File Chooser Dialog Widget . . . 1201
12.7.3.4 OriginOffsetDialog - Origin Offset Setting Dialog Widget . . . 1202
12.7.3.5 ToolOffsetDialog - Tool Offset Setting Dialog Widget . . . 1203
12.7.3.6 ToolChooserDialog - Tool Chooser Dialog Widget . . . 1203
12.7.3.7 MachineLog - Machine Events Journal Display Widget . . . 1204
12.7.3.8 MacroTabDialog - Macro Launch Dialog Widget . . . 1205
12.7.3.9 CamViewDialog - WebCam Part Alignment Dialog Widget . . . 1205
12.7.3.10 EntryDialog - Edit Line Dialog Widget . . . 1205
12.7.3.11 CalculatorDialog - Calculator Dialog Widget . . . 1205
12.7.3.12 RunFromLine - Run-From-Line Dialog Widget . . . 1207
12.7.3.13 VersaProbeDialog - Part Touch Probing Dialog Widget . . . 1208
12.7.3.14 MachineLogDialog - Machine and Debugging Logs Dialog Widget . . . 1209

12.7.4 Other Widgets . . . 1209
12.7.4.1 `NurbsEditor` - NURBS Editing Widget . . . 1210
12.7.4.2 `JoyPad` - 5 button D-pad Widget . . . 1210
12.7.4.3 *WebWidget* . . . 1212
12.7.5 BaseClass/Mixin Widgets . . . 1213
12.7.5.1 `IndicatedPushButtons` . . . 1213
12.7.6 Import-Only Widgets . . . 1216
12.7.6.1 Auto Height . . . 1216
12.7.6.2 G-code Utility . . . 1216
12.7.6.3 Facing . . . 1217
12.7.6.4 Hole Circle . . . 1217
12.7.6.5 Hole Enlarge . . . 1218
12.7.6.6 Qt NGCGUI . . . 1218
12.7.6.7 Qt PDF . . . 1220
12.7.6.8 Qt Vismach . . . 1220
12.7.6.9 Окно выбора Hal . . . 1220
12.8 QtVCP Libraries modules . . . 1220
12.8.1 `Status` . . . 1220
12.8.1.1 Использование . . . 1221
12.8.1.2 Пример . . . 1221
12.8.2 `Info` . . . 1221
12.8.2.1 Available data and defaults . . . 1222
12.8.2.2 User message dialog info . . . 1223
12.8.2.3 Embedded program info . . . 1223
12.8.2.4 Helpers . . . 1223
12.8.2.5 Использование . . . 1223
12.8.3 `Action` . . . 1224
12.8.3.1 Helpers . . . 1224
12.8.3.2 Использование . . . 1224
12.8.4 Qhal . . . 1226
12.8.4.1 Attributes . . . 1226
12.8.4.2 Constants . . . 1227
12.8.4.3 Рекомендации . . . 1227
12.8.5 QPin . . . 1227
12.8.5.1 Сигналы . . . 1227
12.8.5.2 Attributes . . . 1227
12.8.5.3 Рекомендации . . . 1228
12.8.5.4 Пример . . . 1228
12.8.6 `Tool` . . . 1228

12.8.6.1 Helpers . . . 1228
12.8.7 Path . . . 1229
12.8.7.1 Referenced Paths . . . 1229
12.8.7.2 Helpers . . . 1231
12.8.7.3 Использование . . . 1231
12.8.8 VCPWindow . . . 1231
12.8.8.1 Использование . . . 1231
12.8.9 Aux_program_loader . . . 1232
12.8.9.1 Helpers . . . 1232
12.8.9.2 Использование . . . 1232
12.8.10 Keylookup . . . 1233
12.8.10.1 Использование . . . 1233
12.8.10.2 Key Defines . . . 1234
12.8.11 Messages . . . 1236
12.8.11.1 Свойства . . . 1236
12.8.11.2 HAL Контакты . . . 1237
12.8.11.3 Examples . . . 1237
12.8.12 multimessages . . . 1238
12.8.12.1 Свойства . . . 1238
12.8.12.2 Examples . . . 1239
12.8.13 Notify . . . 1239
12.8.13.1 Свойства . . . 1240
12.8.14 Preferences . . . 1240
12.8.15 Player . . . 1240
12.8.15.1 Sounds . . . 1240
12.8.15.2 Использование . . . 1241
12.8.15.3 Пример . . . 1241
12.8.16 Virtual Keyboard . . . 1241
12.8.17 Toolbar Actions . . . 1242
12.8.17.1 Actions . . . 1242
12.8.17.2 Submenus . . . 1242
12.8.17.3 Использование . . . 1242
12.8.17.4 Examples . . . 1242
12.8.18 Qt Vismach Machine Graphics library . . . 1243
12.8.18.1 Builtin Samples . . . 1243
12.8.18.2 Primitives Library . . . 1243
12.8.18.3 Использование . . . 1245
12.8.18.4 More Information . . . 1246
12.9 QtVismach . . . 1246

12.9.1 Введение . . . 1246
12.9.2 Hierarchy of Machine Design . . . 1248
12.9.3 Start the script . . . 1249
12.9.4 HAL pins. . . . 1249
12.9.5 Creating Parts . . . 1249
12.9.5.1 Import STL or OBJ Files . . . 1249
12.9.5.2 Build from Geometric Primitives . . . 1250
12.9.6 Moving Model Parts . . . 1251
12.9.6.1 Translating Model parts . . . 1251
12.9.6.2 Rotating Model Parts . . . 1251
12.9.7 Animating Parts . . . 1251
12.9.7.1 `HalTranslate` . . . 1251
12.9.7.2 `HalRotate` . . . 1252
12.9.7.3 *HalToolCylinder* . . . 1252
12.9.7.4 *HalToolTriangle* . . . 1252
12.9.7.5 HAL Adjustable Primitives . . . 1252
12.9.8 Assembling the model . . . 1253
12.9.9 Other functions . . . 1254
12.9.9.1 Цвет . . . 1254
12.9.9.2 HALColorFlip . . . 1254
12.9.9.3 HALColorRGB . . . 1254
12.9.9.4 Heads Up Display . . . 1255
12.9.9.5 HAL Heads Up Display . . . 1255
12.9.9.6 Скрыть коллекцию . . . 1255
12.9.9.7 Plot Color Based on Motion Type . . . 1256
12.9.9.8 Capture . . . 1256
12.9.9.9 main . . . 1256
12.9.10 Tips . . . 1257
12.9.11 Basic structure of a `QtVismach` script . . . 1257
12.9.12 Builtin Vismach Sample Panels . . . 1258
12.10 QtVCP: Building Custom Widgets . . . 1258
12.10.0 Обзор . . . 1258
12.10.1. Widgets . . . 1258
12.10.1. Qt Designer . . . 1259
12.10.1. Initialization Process . . . 1259
12.10.1. Cleanup process . . . 1260
12.10.2 Custom HAL Widgets . . . 1260
12.10.3 Custom Controller Widgets Using `STATUS` . . . 1261
12.10.3.1 In The *Imports* Section . . . 1262

12.10.3.2 In The *Instantiate Libraries* Section . . . 1263
12.10.3.3 In The *Custom Widget Class Definition* Section . . . 1263
12.10.4 Custom Controller Widgets with Actions . . . 1265
12.10.5 Stylesheet Property Changes Based On Events . . . 1267
12.10.6 Use Stylesheets To Change Custom Widget Properties . . . 1268
12.10.7 Widget Plugins . . . 1268
12.10.7.1 Gridlayout Example . . . 1269
12.10.7.2 SystemToolbutton Example . . . 1269
12.10.7.3 Making a plugin with a MenuEntry dialog box . . . 1270
12.11 QtVCP Handler File Code Snippets . . . 1273
12.11.1 Preference File Loading/Saving . . . 1273
12.11.2 Use `QSettings` To Read/Save Variables . . . 1273
12.11.3 Add A Basic Style Editor . . . 1274
12.11.4 Request Dialog Entry . . . 1274
12.11.5 Speak a Startup Greeting . . . 1275
12.11.6 ToolBar Functions . . . 1276
12.11.7 Add HAL Pins That Call Functions . . . 1277
12.11.8 Read/Write System HAL Pins Directly . . . 1277
12.11.9 Add A Special Max Velocity Slider Based On Percent . . . 1278
12.11.10 Toggle Continuous Jog On and Off . . . 1278
12.11.11 Class Patch The File Manager Widget . . . 1279
12.11.12 Adding Widgets Programmatically . . . 1281
12.11.13 Update/Read Objects Periodically . . . 1284
12.11.14 External Control With ZMQ . . . 1285
12.11.14.1 ZMQ Messages Reading . . . 1285
12.11.14.2 ZMQ Messages Writing . . . 1286
12.11.15 Sending Messages To Status Bar Or Desktop Notify Dialogs . . . 1287
12.11.16 Catch Focus Changes . . . 1288
12.11.17 Read Command Line Load Time Options . . . 1288
12.11.18 Gcode to read Qt preferences . . . 1289
12.12 QtVCP Development . . . 1290
12.12.1 Обзор . . . 1290
12.12.2 Builtin Locations . . . 1290
12.12.3 QtVCP Startup To Shutdown . . . 1290
12.12.3.1 QtVCP Startup . . . 1291
12.12.3.2 QtVCP Shutdown . . . 1291
12.12.4 Path Information . . . 1291
12.12.5 Idiosyncrasies . . . 1292
12.12.5.1 Error Code Collecting . . . 1292

12.12.5.1 Log Rate . . . 1292
12.12.5.2 Keybinding . . . 1292
12.12.5.3 Preference File . . . 1293
12.12.5.4 Widget Special Setup Functions . . . 1293
12.12.5.5 Dialogs . . . 1293
12.12.5.6 Styles (Themes) . . . 1293

**13 Программирование интерфейса пользователя 1294**

13.1 Panelui . . . 1294
13.1.1 Введение . . . 1294
13.1.2 Loading Commands . . . 1294
13.1.3 panelui.ini file reference . . . 1295
13.1.4 Internal Command reference . . . 1297
13.1.5 ZMQ Messages . . . 1300
13.1.6 Handler File Extension . . . 1300
13.2 Python-модуль для LinuxCNC . . . 1301
13.2.1 Введение . . . 1301
13.2.2 Usage Patterns for the LinuxCNC NML interface . . . 1302
13.2.3 Чтение статуса LinuxCNC с помощью модуля Python linuxcnc . . . 1302
13.2.3.1 linuxcnc.stat attributes . . . 1302
13.2.3.2 The `axis` dictionary . . . 1308
13.2.3.3 The `joint` dictionary . . . 1308
13.2.3.4 The `spindle` dictionary . . . 1309
13.2.4 Preparing to send commands . . . 1310
13.2.5 Sending commands through `linuxcnc.command` . . . 1311
13.2.5.1 `linuxcnc.command` attributes . . . 1311
13.2.5.2 `linuxcnc.command` methods: . . . 1312
13.2.6 Reading the error channel . . . 1315
13.2.7 Reading INI file values . . . 1315
13.2.8 The `linuxcnc.positionlogger` type . . . 1316
13.2.8.1 members . . . 1316
13.2.8.2 methods . . . 1316
13.3 HAL Python модуль . . . 1317
13.3.1 Basic usage . . . 1317
13.3.2 Функции . . . 1317
13.4 GStat Python Module . . . 1321
13.4.1 Intro . . . 1321
13.4.2 Sample GStat Code . . . 1321
13.4.2.1 Sample HAL component code pattern . . . 1321

13.4.2.2GladeVCP Python extension code pattern . . . 1322
13.4.2.3QtVCP Python extension code pattern . . . 1323
13.4.3Messages . . . 1323
13.4.4Функции . . . 1330
13.4.5Known Issues . . . 1332
13.5Vismach . . . 1332
13.5.1Start the script . . . 1334
13.5.2Create the HAL pins. . . . 1334
13.5.3Creating Parts . . . 1334
13.5.4Moving Parts . . . 1335
13.5.5Animating Parts . . . 1335
13.5.6Assembling the model. . . . 1336
13.5.7Other functions . . . 1337
13.5.8Basic structure of a Vismach script. . . . 1338

**III Глоссарий, авторское право и история 1339**

**14 С обратной стороны 1340**

**15 Glossary 1341**

**16 Авторское право 1347**
16.1Legal Section . . . 1347
16.1.1Copyright Terms . . . 1347
16.1.2GNU Free Documentation License . . . 1347

**17 История LinuxCNC 1352**
17.1Origin . . . 1352
17.1.1Name Change . . . 1353
17.1.2Additional Info . . . 1353

# Part I

# Начало работы и настройка

# Chapter 1

# Начало работы с LinuxCNC

## 1.1 О LinuxCNC

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

To lower the entry-hurdle, LinuxCNC provides: * easy discovery and testing without installation with the Live Image, * easy installation from the Live Image, * easy to use graphical configuration wizards to rapidly create a configuration specific to the machine, * directly availability as regular packages of recent releases of Debian (since Bookworm) and Ubuntu (since Kinetic Kudu).

LinuxCNC provides a graphical user interface with many flavours to choose from to match your personal preferences and technical needs. Advanced users may directly exploit * graphical interface creation tools (Glade, Qt), * the interpreter for *G-code* (the RS-274 machine tool programming language), * operation of low-level machine electronics such as sensors and motor drives, * an easy to use *breadboard* layer for quickly creating a unique configuration for your machine, * a software PLC programmable with ladder diagrams.

Under the hood, LinuxCNC provides * a realtime motion planning system with look-ahead, * support for non-Cartesian motion systems is provided via custom kinematics modules. Available architectures include hexapods (Stewart platforms and similar concepts) and systems with rotary joints to provide motion such as PUMA or SCARA robots. * support for a variety of hardware interfaces. The control can operate true servos (analog or PWM) with the feedback loop closed by the LinuxCNC software at the computer, or open loop with step-servos or stepper motors. * Motion control features include: cutter radius and length compensation, path deviation limited to a specified tolerance, lathe threading, synchronized axis motion, adaptive feedrate, operator feed override, and constant velocity control. * LinuxCNC runs on Linux using real-time extensions.

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

### 1.1.1 Architecture - Context diagram

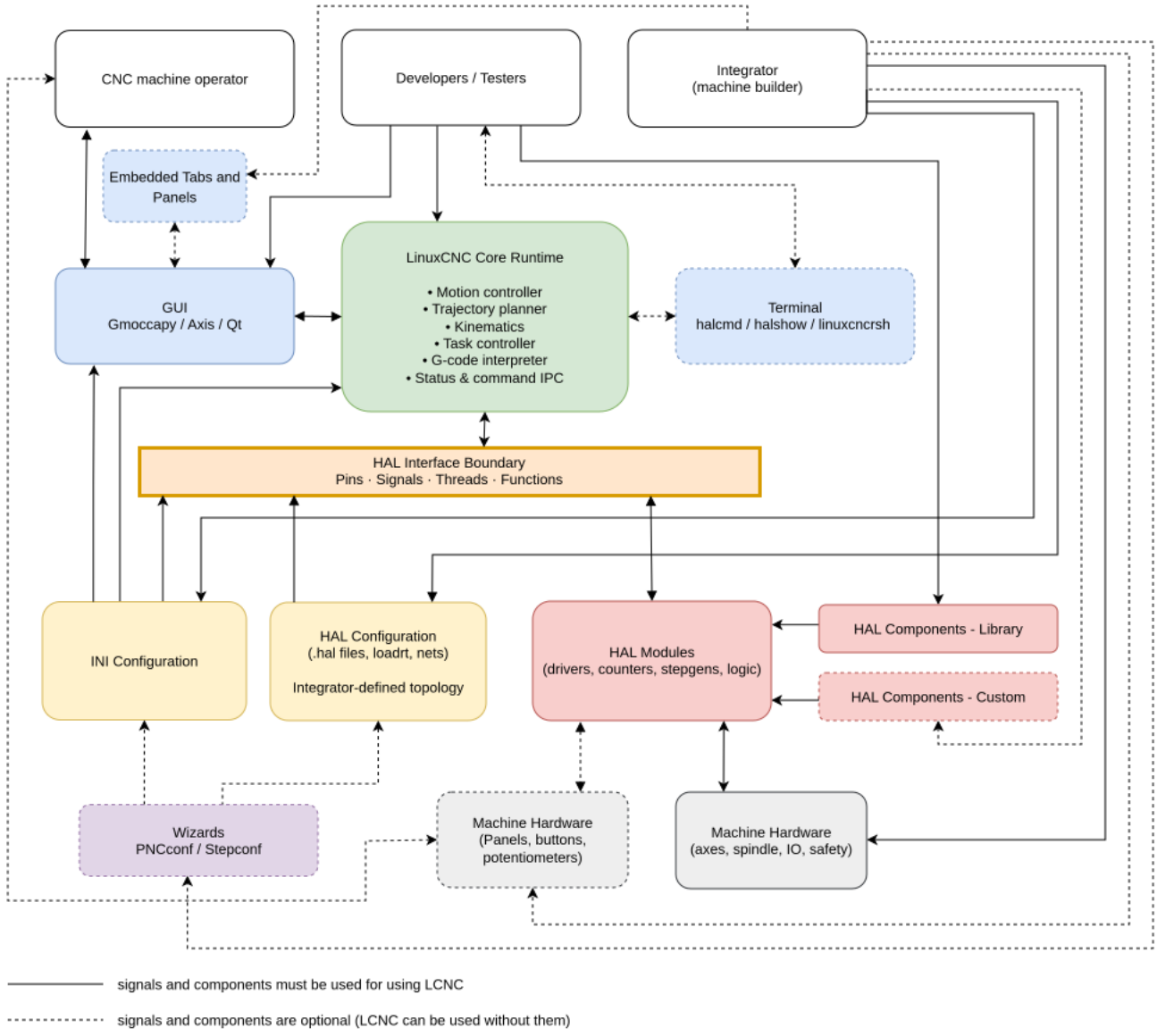



Figure 1.1: Roles of operators, integrators, developers and hardware

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

**Operator**

One a machine is set up, its operator will only use one of the many graphical user interfaces that LinuxCNC and external groups are providing. The requirements for the operator are determined by how the integrator has set up the machine. The integrator has the option of setting up the machine so that the operator only presses one button to start the machining process, or leaves the GUI in its default state and the operator will fully control the CNC machine using the GUI functionality and G,M,O-codes. The integrator may or may not create a physical or virtual panel for the operator with various buttons and various indicators.

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

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

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

### 1.1.2 The Operating System

LinuxCNC is available as ready-to-use packages for Debian distributions.

### 1.1.3 Получение помощи

#### 1.1.3.1 Web Forum

A web forum can be found at https://forum.linuxcnc.org or by following the link at the top of the linuxcnc.org home page.

This is quite active but the demographic is more user-biased than the mailing list. If you want to be sure that your message is seen by the developers then the mailing list is to be preferred.

#### 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.

The simplest way to get on the IRC is to use the embedded web client client from libera.

**Some IRC etiquette**

- Ask specific questions… Avoid questions like ”Can someone help me?”.
- If you’re really new to all this, think a bit about your question before typing it. Make sure you give enough information so someone can answer your question or solve your problem.
- Have some patience when waiting for an answer. Sometimes it takes a while to formulate an answer, or everyone might be busy working or something.
- Set up your IRC account with your unique name so people will know who you are. If you use the java client, use the same name every time you log in. This helps people remember who you are. If you have been on before, many will remember past discussions with you which will save time on both ends.

**Sharing Files**

The most common way to share files on the IRC is to upload the file to one of the following or a similar service and paste the link:

- *For text*: https://pastebin.com/, https://gist.github.com/, https://0bin.net/, https://paste.debian.net/-
- *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 Mailing List

An Internet Mailing List is a way to put questions out for everyone on that list to see and answer at their convenience. You get better exposure to your questions on a mailing list than on the IRC but answers take longer. In a nutshell you e-mail a message to the list and either get daily digests or individual replies back depending on how you set up your account.

You can subscribe to the emc-users mailing list at: https://lists.sourceforge.net/lists/listinfo/emc-users.

#### 1.1.3.4 Web Forum

A web forum can be found at https://forum.linuxcnc.org/ or by following the link at the top of the https://linuxcnc.org/ home page.

This is quite active but the demographic is more user-biased than the mailing list. If you want to be sure that your message is seen by the developers then the mailing list is to be preferred.

#### 1.1.3.5 LinuxCNC Wiki

A Wiki site is a user maintained web site that anyone can add to or edit.

The user maintained LinuxCNC Wiki site contains a wealth of information and tips at: http://wiki.linuxcnc.org

#### 1.1.3.6 Bug Reports

Report bugs on the LinuxCNC Github github bug tracker.

## 1.2 Системные требования

### 1.2.1 Минимальные требования

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.

Дополнительную информацию можно найти на Wiki LinuxCNC site: Hardware Requirements

LinuxCNC и Debian Linux должны достаточно хорошо работать на компьютере со следующими минимальными спецификациями оборудования. Эти цифры не являются абсолютным минимумом, но обеспечивают приемлемую производительность для большинства шаговых систем.

- 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
- Видеокарта с разрешением не менее 1024x768, не использующая собственные драйверы NVidia или ATI fglrx. Современные встроенные графические чипсеты, похоже, в целом работают нормально.
- Интернет-соединение (не обязательно, но очень полезно для обновлений и общения с сообществом LinuxCNC)

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") and/or 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 Требования к ядру и версии

LinuxCNC требует модифицированного ядра для использования в реальном времени для управления реальным машинным оборудованием. Однако он может работать на стандартном ядре в режиме моделирования для таких целей, как проверка G-кода, тестирование файлов конфигурации и изучение системы. Для работы с этими версиями ядра распространяются две версии LinuxCNC. Имена пакетов: «linuxcnc» и «linuxcnc-uspace».

Опции ядра реального времени — preempt-rt, RTAI и Xenomai.

Вы можете узнать версию ядра вашей системы с помощью команды:

```
uname -a
```

Если вы видите (как указано выше) `-rt-` в имени ядра, значит, вы используете ядро preempt-rt и вам следует установить версию LinuxCNC "uspace". Вам также следует установить uspace для конфигураций "sim" на ядрах, не работающих в реальном времени.

Если вы видите `-rtai-` в имени ядра, значит, вы используете RTAI в реальном времени. См. ниже версию LinuxCNC для установки.

#### 1.2.2.1 Preempt-RT с пакетом «linuxcnc-uspace»

Preempt-RT — новейшая из систем реального времени, а также версия, наиболее близкая к основному ядру. Ядра Preempt-RT доступны в виде предварительно скомпилированных пакетов из основных репозиториев. Их можно найти по поисковому запросу «PREEMPT_RT», и его можно загрузить и установить, как и любой другой пакет. Preempt-RT, как правило, обеспечивает лучшую поддержку драйверов и является единственным вариантом для систем, использующих аппаратные карты драйверов Mesa, подключенные к Ethernet. В целом у preempt-rt самая низкая задержка среди доступных систем, но есть исключения.

#### 1.2.2.2 RTAI с пакетом «linuxcnc»

RTAI уже много лет является основой дистрибутивов LinuxCNC. Обычно он обеспечивает наилучшую производительность в реальном времени с точки зрения низкой задержки, но может иметь меньшую поддержку периферийных устройств и меньшее разрешение экрана. Ядро RTAI доступно в репозитории пакетов LinuxCNC. Если вы установили из образа Live/Install, то переключение ядра и версии LinuxCNC описано в [Installing-RTAI].

#### 1.2.2.3 Xenomai с пакетом «linuxcnc-uspace»

Xenomai также поддерживается, но вам придется найти или собрать ядро и скомпилировать LinuxCNC из исходного кода, чтобы использовать его.

#### 1.2.2.4 RTAI с пакетом «linuxcnc-uspace»

Также возможно запустить LinuxCNC с RTAI в режиме пользовательского пространства. Как и в случае с Xenomai, для этого вам придется скомпилировать исходный код.

### 1.2.3 Проблемное оборудование

#### 1.2.3.1 Ноутбуки

Ноутбуки обычно не подходят для генерации шагов программным способом в реальном времени. Опять же, запуск теста задержки в течение длительного времени предоставит вам информацию, необходимую для определения пригодности.

#### 1.2.3.2 Видеокарты

Если ваша установка отображается с разрешением экрана 800 x 600, то, скорее всего, Debian не распознает вашу видеокарту или монитор. Иногда эту проблему можно обойти, установив драйверы или создав или отредактировав файлы Xorg.conf.

## 1.3 Получение LinuxCNC

В этом разделе описан рекомендуемый способ загрузки и новой установки LinuxCNC. Для любителей приключений также существуют Альтернативные методы установки. Если у вас есть существующая установка, которую вы хотите обновить, вместо этого перейдите в раздел Обновление LinuxCNC.

**Note**
Для управления оборудованием LinuxCNC требуется специальное ядро с расширениями реального времени. Здесь есть три возможности: preempt-rt, RTAI или Xenomai. Кроме того, существуют две версии LinuxCNC, работающие с этими ядрами. Подробности смотрите в таблице ниже. Однако для тестирования кода и моделирования можно запустить приложение `linuxcnc-uspace` на стандартном ядре дистрибутива.

Свежие установки LinuxCNC проще всего создать с помощью образа Live/Install. Это гибридный образ файловой системы ISO, который можно записать на USB-накопитель или DVD-диск и использовать для загрузки компьютера. Во время загрузки вам будет предоставлен выбор: загрузить "Live" систему (чтобы запустить LinuxCNC без внесения каких-либо постоянных изменений в ваш компьютер) или загрузить установщик (чтобы установить LinuxCNC и его операционную систему на жесткий диск вашего компьютера).

Схема процесса выглядит так:

1. Загрузите Live/Install образ.
2. Запишите образ на USB-накопитель или DVD.
3. Загрузите систему Live, чтобы протестировать LinuxCNC.
4. Загрузите установщик, чтобы установить LinuxCNC.

## 1.3.1 Загрузка образа

В этом разделе описаны некоторые способы загрузки образа Live/Install.

### 1.3.1.1 Обычная загрузка

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.

Для Raspberry Pi предоставлено несколько образов, позволяющих устранить различия между RPi4 и RPi5.

**Note**
Не используйте обычный дистрибутив Raspbian для LinuxCNC, который может поставляться с вашим стартовым комплектом RPi — в нем не будет ядра реального времени, и вы не сможете перейти с Raspbian на образ ядра Debian.

### 1.3.1.2 Загрузка с помощью zsync

zsync — это приложение для загрузки, которое эффективно возобновляет прерванные загрузки и эффективно переносит большие файлы с небольшими изменениями (если у вас есть старая локальная копия). Обратите внимание: требуется протокол http, а не https. Используйте zsync, если загрузка изображения методом Normal Download часто прерывается.

zsync в Linux

1. Установите zsync с помощью Synaptic или выполнив следующую команду в terminal

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

2. Затем запустите эту команду, чтобы загрузить ISO на свой компьютер

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

Please remember to confirm the checksum of the downloaded iso as described below, since the authenticity of the server is not guaranteed with the http protocol.

**zsync в Windows** Существует порт zsync для Windows. Он работает как консольное приложение и его можно загрузить по адресу https://www.assembla.com/spaces/zsync-windows/documents.

#### 1.3.1.3 Проверка образа

(Этот шаг не нужен, если вы использовали zsync)

1. После загрузки проверьте контрольную сумму образа, чтобы убедиться в его целостности.

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

or

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

1. Затем сравните с этими контрольными суммами

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

**Проверьте md5sum на Windows или 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 Запишите образ на загрузочное устройство

ISO-образ LinuxCNC Live/Install — это гибридный ISO-образ, который можно записать непосредственно на USB-накопитель (флэш-накопитель) или DVD-диск и использовать для загрузки компьютера. Изображение слишком велико для компакт-диска.

#### 1.3.2.1 Raspberry Pi Образ

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) Образ использующий инструменты ГИП

Загрузите и установите Balena Etcher с https://etcher.balena.io/#download-etcher (Linux, Windows, Mac) и запишите загруженный образ на USB-накопитель.

Если ваш образ не загружается, попробуйте также Rufus. Он выглядит сложнее, но кажется более совместимым с различными BIOS.

#### 1.3.2.3 Командная строка - Linux

1. Подключите запоминающее устройство USB (например, флэш-накопитель или устройство типа флэш-накопителя).
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. Используйте команду dd, чтобы записать образ на USB-накопитель. Например, если ваше устройство хранения отображается как `/dev/sde`, используйте следующую команду:

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

#### 1.3.2.4 Командная строка - MacOS

1. Open a terminal and type

```
diskutil list
```

2. Вставьте USB и запишите имя нового диска, которое появится, например /dev/disk5.
3. Размонтируйте USB. Число, указанное выше, следует заменить на N.

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

4. Перенесите данные с помощью dd, как для Linux выше. Обратите внимание, что в начале имени диска добавлена буква ”r”.

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

Запись образа на DVD в Linux

1. Вставьте чистый DVD в записывающее устройство. Появится окно *CD/DVD Creator* или *Choose Disc Type*. Закройте это, так как мы не будем его использовать.
2. Перейдите к загруженному изображению в браузере файлов.
3. Щелкните правой кнопкой мыши файл образа ISO и выберите Write to Disc.
4. Выберите скорость записи. Рекомендуется писать на минимально возможной скорости.
5. Запустите процесс записи.
6. Если появится окно *choose a file name for the disc image*, просто выберите OK.

Запись образа на DVD в Windows

1. Загрузите и установите Infra Recorder, бесплатную программу записи изображений с открытым исходным кодом: https://infrarecorder.org/.
2. Вставьте пустой компакт-диск в привод и выберите Do nothing или Cancel, если появится диалоговое окно автозапуска.
3. Откройте Infra Recorder и выберите меню *Actions*, затем *Burn image*.

Запись образа на DVD в Mac OSX

1. Загрузите файл .iso
2. Щелкните правой кнопкой мыши файл в окне Finder и выберите ”Burn to disc”. (Опция записи на диск появится только в том случае, если на машине установлен или подключен оптический привод.)

### 1.3.3 Тестирование LinuxCNC

Подключив USB-накопитель или DVD-диск в приводе DVD, выключите компьютер, а затем снова включите его. Компьютер загрузится из образа Live/Install и будет выбран вариант загрузки Live.

**Note**
Если система не загружается с DVD-диска или USB-накопителя, возможно, необходимо изменить порядок загрузки в BIOS ПК.

После загрузки компьютера вы можете попробовать LinuxCNC, не устанавливая его. Вы не можете создавать собственные конфигурации или изменять большинство системных настроек в сеансе Live, но вы можете (и должны) запустить тест задержки.

Чтобы опробовать LinuxCNC: в меню Applications/CNC выберите LinuxCNC. Откроется диалоговое окно, в котором вы сможете выбрать одну из множества примеров конфигураций. На этом этапе действительно имеет смысл выбрать конфигурацию ”sim”. Некоторые из примеров конфигураций включают в себя трехмерное моделирование машин на экране. Чтобы увидеть их, найдите ”Vismach”.

Чтобы проверить, подходит ли ваш компьютер для программной пошаговой генерации импульсов, запустите тест задержки, как показано здесь.

На момент написания Live Image доступен только с ядром preempt-rt и соответствующим LinuxCNC. На некотором оборудовании это может не обеспечивать достаточно хорошую задержку. Доступна экспериментальная версия, использующая ядро реального времени RTAI, которая часто обеспечивает лучшую задержку.

### 1.3.4 Установка LinuxCNC

Чтобы установить LinuxCNC с Live CD, при загрузке выберите *Install (Graphical)*.

### 1.3.5 Обновления LinuxCNC

При обычной установке Update Manager будет уведомлять вас об обновлениях LinuxCNC, когда вы подключаетесь к сети, и позволит вам легко выполнить обновление без необходимости знания Linux. По запросу можно обновить все, кроме операционной системы.

**Warning**
Не обновляйте операционную систему на новую версию, если будет предложено это сделать. Однако вам следует принимать *обновления* ОС, особенно обновления безопасности.

### 1.3.6 Проблемы с установкой

В редких случаях вам может потребоваться сбросить настройки BIOS до значений по умолчанию, если во время установки Live CD он не может распознать жесткий диск во время загрузки.

### 1.3.7 Альтернативные методы установки

Самый простой и предпочтительный способ установки LinuxCNC — использовать Live/Install Image, как описано выше. Этот метод настолько прост и надежен, насколько это возможно, и подходит как для начинающих, так и для опытных пользователей. Однако обычно это заменяет любую существующую операционную систему. Если на целевом компьютере есть файлы, которые вы хотите сохранить, воспользуйтесь одним из методов, описанных в этом разделе.

Кроме того, для опытных пользователей, знакомых с системным администрированием Debian (поиск установочных образов, манипулирование источниками apt, изменение версий ядра и т. д.), новые установки поддерживаются на следующих платформах: (”amd64” означает ”64-разрядный” и не специфично для процессоров AMD, он будет работать в любой 64-битной системе x86)

| **Debian Trixie** | **amd64 & arm64** | **preempt-rt** | **linuxcnc-uspace** | **управление станком и моделирование** |
|---|---|---|---|---|
| Debian Trixie | amd64 | RTAI | linuxcnc | управление станком |
| Дистрибутив | Архитектура | Ядро | Имя пакета | Типовое использование |
| Debian Bookworm | amd64 & arm64 | preempt-rt | linuxcnc-uspace | управление станком и моделирование |
| Debian Bookworm | amd64 | RTAI | linuxcnc | управление станком |
| Debian Bullseye | amd64 | preempt-rt | linuxcnc-uspace | управление станком и моделирование |
| Любой | Любой | Оригинал | linuxcnc-uspace | ТОЛЬКО моделирование |

**Note**
LinuxCNC v2.9 не поддерживается в Debian 9 и более ранних версиях.

**Preempt-RT ядра** Ядра Preempt-rt доступны для Debian из обычного архива debian.org. Пакет называется `linux-image-rt-*`. Просто установите пакет так же, как и любой другой пакет, из диспетчера пакетов Synaptic или с помощью apt-get в командной строке.

**RTAI Ядра** Ядра RTAI доступны для загрузки из архива Debian linuxcnc.org. Подходящий источник:

- 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 и ядро RTAI теперь доступны только для 64-битных ОС, но очень мало сохранившихся систем не могут работать на 64-битной ОС.

#### 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.deb
   distrib/

2. После записи ISO и загрузки, если вам не нужен рабочий стол Gnome, выберите *Advanced Options > Alternative desktop environments* и выберите тот, который вам нравится. Затем выберите *Install* or *Graphical Install*.

**Warning**
Не вводите пароль root, если вы это сделаете, sudo будет запрещено, и вы не сможете завершить следующие шаги.

3. Запустите следующую команду в terminal, чтобы обновить машину до последних пакетов.

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

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

4. Установите ядро и модули Preempt-RT

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

5. Перезагрузитесь и выберите ядро Linux 6.1.0-10-rt-amd64. Точная версия ядра может отличаться, обратите внимание на суффикс ”-rt”. Это может быть скрыто в подменю ”Advanced options for Debian Bookworm” в Grub. При входе в систему убедитесь, что следующая команда сообщает о PREEMPT RT.

```
uname -v
```

6. Откройте Applications Menu > System > Synaptic Package Manager, найдите *linux-image*, щелкните правой кнопкой мыши исходный файл без rt и выберите *Mark for Complete Removal*. Перезагрузитесь. Это необходимо для принудительной загрузки системы с ядра RT. Если вы предпочитаете сохранить оба ядра, то другие ядра удалять не нужно, но изменения конфигурации загрузки grub потребуются, выходящие за рамки этого документа.

7. Добавьте ключ подписи архива LinuxCNC в свой набор ключей apt, загрузив [скрипт установщика LinuxCNC] (https://www.linuxcnc.org/linuxcnc-install.sh). Для его запуска вам потребуется сделать скрипт исполняемым:

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

```
b''3b''b''ab''b''тb''b''eb''b''мb'' b''вb''b''ыb'' b''мb''b''ob''b''жb''b''eb''b''тb''b ↩
    ''eb'' b''зb''b''ab''b''пb''b''yb''b''cb''b''тb''b''иb''b''тb''b''ьb'' b''yb''b' ↩
    'cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''щb''b''иb''b''кb'':
```

```
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. Вы можете добавить ключ подписи архива LinuxCNC и информацию о репозитории, загрузив и запустив сценарий установки, как описано выше. Если обнаружено ядро RTAI, оно остановится перед установкой каких-либо пакетов.
3. Обновите список пакетов с 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
```

Перезагрузите компьютер, убедившись, что система загружается с новым ядром 5.4.258-rtai.

#### 1.3.7.3 Установка на Raspbian 12

Не делайте этого. Задержки слишком велики с ядром по умолчанию, а ядро Debian PREEMPT_RT (RT важно) не загружается на Pi (по состоянию на 1 января 2024 г.). Пожалуйста, обратитесь к образам .iso, которые можно найти в Интернете на обычной странице LinuCNC download page. Вы можете создать их самостоятельно, следуя предоставленным скриптам online.

## 1.4 Running LinuxCNC

### 1.4.1 Вызов LinuxCNC

После установки LinuxCNC запускается так же, как и любая другая программа Linux: запустите ее из terminal, введя команду «linuxcnc», или выберите ее в меню «Applications -> CNC».

### 1.4.2 Конфигурационный загрузчик

При запуске LinuxCNC (из меню ЧПУ или из командной строки без указания INI-файла) открывается диалог Configuration Selector.

Диалоговое окно «Configuration Selector» позволяет пользователю выбрать одну из существующих конфигураций (My Configurations) или выбрать новую (из примеров конфигураций) для копирования в свой домашний каталог. Скопированные конфигурации появятся в разделе «My Configurations» при следующем вызове селектора конфигурации.

Configuration Selector предлагает выбор конфигураций, организованных:

- «My Configurations» — пользовательские конфигурации, расположенные в linuxcnc/configs в вашем домашнем каталоге.
- «Sample Configurations» — примеры конфигураций, если они выбраны, копируются в linuxcnc/configs. После того как образец конфигурации будет скопирован в ваш локальный каталог, программа запуска предложит его как «My Configurations». Имена, под которыми представлены эти локальные конфигурации, соответствуют именам каталогов внутри каталога configs/:
  - *sim* — конфигурации, включающие моделируемое оборудование. Их можно использовать для тестирования или изучения работы LinuxCNC.
  - *by_interface* — конфигурации, организованные по признаку ГИП.

- *by_machine* - Конфигурации организованные по признаку станка.
- *apps* — приложения, которые не требуют запуска linuxcnc, но могут быть полезны для тестировани
  или использования таких приложений, как PyVCP или GladeVCP.
- «attic» — устаревшие или исторические конфигурации.

Конфигурации Sim часто являются наиболее полезной отправной точкой для новых пользователей и организованы вокруг поддерживаемых ГИПов:

- *axis* — ГИП клавиатуры и мыши
- «craftsman» — ГИП с сенсорным экраном (больше не поддерживается ???)
- *gmoccapy* - ГИП Сенсорного экрана
- *gscreen* - ГИП Сенсорного экрана
- *pyvcp_demo* - Виртуальная панель управления Python
- *qtaxis* — ГИП с сенсорным экраном, выглядит как Axis
- *qtdragon* - ГИП Сенсорного экрана
- *qtdragon_hd* - ГИП Сенсорного экрана высокого разрешения
- *qtplasmac* - ГИП Сенсорного экрана, для столов плазменной резки
- *qttouchy* - ГИП Сенсорного экрана
- *tklinuxcnc* — ГИП клавиатуры и мыши (больше не поддерживается)
- *touchy* - ГИП для сенсорного экрана
- *woodpecker* - ГИП Сенсорного экрана

Каталог конфигурации ГИП может содержать подкаталоги с конфигурациями, иллюстрирующими особые ситуации или встраивание других приложений.

Конфигурации *by_interface* организованы вокруг общих поддерживаемых интерфейсов, таких как:

- общая мехатроника
- mesa
- parport
- pico
- pluto
- servotogo
- vigilant
- vitalsystems

Для использования этих конфигураций в качестве отправной точки системы может потребоваться соответствующее оборудование.

Конфигурации *by_machine* организованы вокруг полных известных систем, таких как:

- boss
- cooltool

- scortbot erIII
- sherline
- smithy
- tormach

Для использования этих конфигураций может потребоваться полная система.

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 — создает файл с системной информацией, которая может быть полезна для диагностики проблем.
   - Gladevcp — пример приложений GladeVCP.
   - halrun — запускает halrun в terminal.
   - latency — приложения для исследования задержки
     - latency-histogram-1 - гистограмма для одного servo thread
     - latency-histogram - гистограмма
     - latency-test - стандартный тест
     - latency-plot — ленточная диаграмма
   - parport - Приложение для тестирования parport.
   - pyvcp — Примеры приложений pyvcp.
   - xhc-hb04 — приложения для тестирования беспроводного USB-устройства РГИ xhc-hb04

---

**Note**
В каталоге «Приложения» для копирования в каталог пользователя предлагаются только те приложения, которые были изменены пользователем с пользой.

---

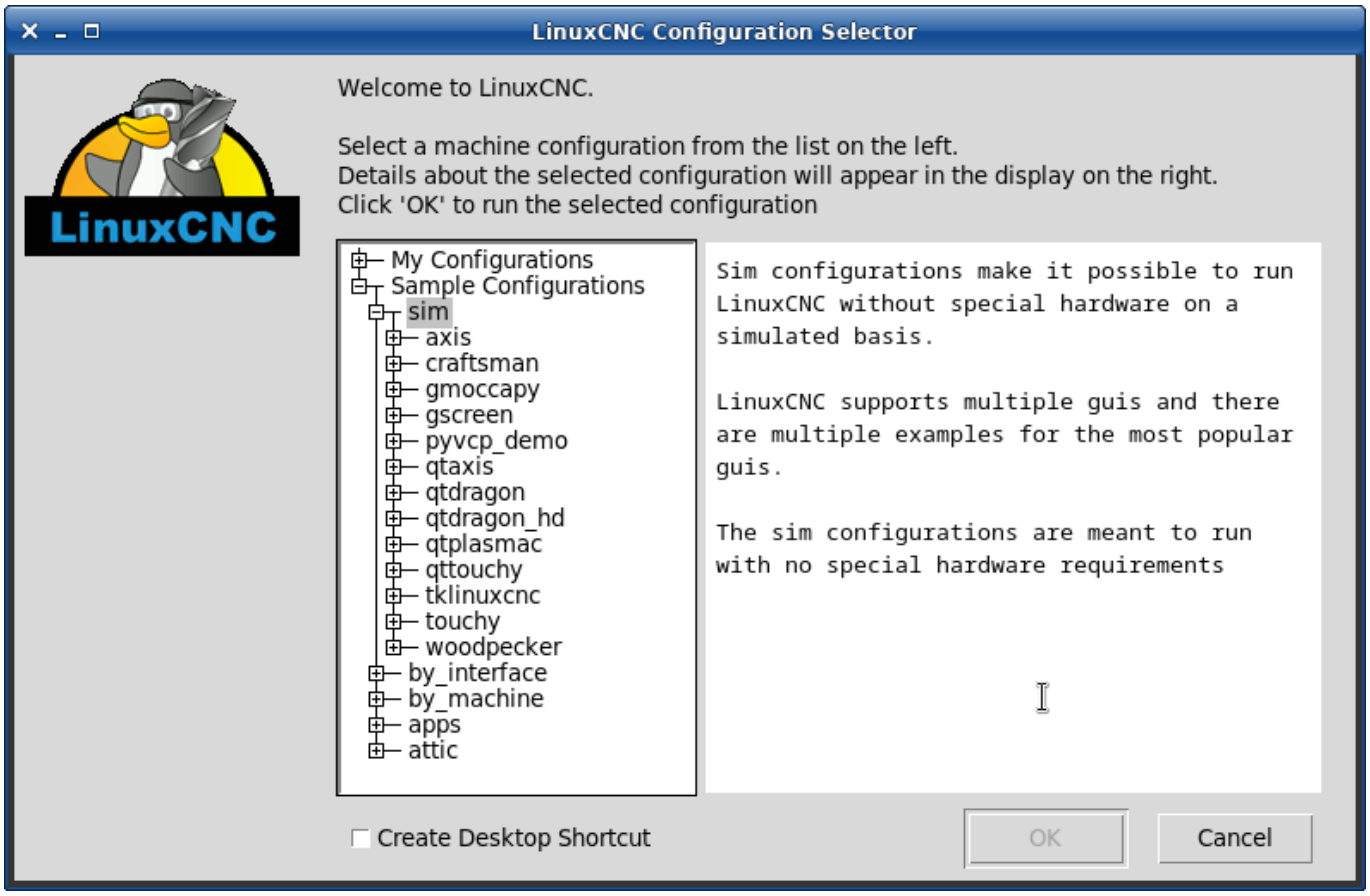



Figure 1.2: Селектор конфигурации LinuxCNC

Кликните на любой из перечисленных конфигураций, чтобы отобразить конкретную информацию о ней. Дважды щелкните конфигурацию или нажмите «OK», чтобы начать настройку.

Выберите *Create Desktop Shortcut*, а затем нажмите «OK», чтобы добавить значок на рабочий стол Ubuntu для прямого запуска этой конфигурации без отображения экрана выбора конфигурации.

Когда вы выбираете конфигурацию в разделе Sample Configurations, она автоматически помещает копию этой конфигурации в каталог ~/linuxcnc/configs.

### 1.4.3 Следующие шаги по настройке

Найдя пример конфигурации, использующий то же интерфейсное оборудование, что и ваш станок (или конфигурацию симулятора), и сохранив копию в своем домашнем каталоге, вы можете настроить ее в соответствии с особенностями вашего станка. См. раздел «Руководство для интегратора», посвященный настройке.

### 1.4.4 Конфигурации симулятора

Все конфигурации, перечисленные в разделе Sample Configurations/sim, предназначены для запуска на любом компьютере. Никакого специального оборудования не требуется и поддержка в режиме реального времени не требуется.

Эти конфигурации полезны для изучения отдельных возможностей или опций. Конфигурации для симуляции организованы в соответствии с ГИП, использованным в демонстрации. Каталог

для Axis содержит больше всего вариантов и подкаталогов, поскольку это наиболее протестированный ГИП. Возможности, продемонстрированные в любом конкретном ГИП, могут быть доступны и в других ГИП.

### 1.4.5 Ресурсы по настройке

Селектор конфигурации копирует все файлы, необходимые для конфигурации, в новый подкаталог ~/linuxcnc/configs (эквивалентно: /home/username/linuxcnc/configs). Каждый созданный каталог будет включать как минимум один INI-файл (iniflename.ini), который используется для описания конкретной конфигурации.

Файловые ресурсы в скопированном каталоге обычно включают один или несколько INI-файлов (filename.ini) для связанных конфигураций и файл таблицы инструментов (toolfilename.tbl). Кроме того, ресурсы могут включать файлы HAL (filename.hal, filename.tcl), файл README для описания каталога и информацию, специфичную для конфигурации, в текстовом файле, названном в честь конкретной конфигурации (inifilename.txt). Последние два файла отображаются при использовании Селектора Конфигурации.

В поставляемых примерах конфигураций может быть указан параметр HALFILE (имя_файла.hal) в INI-файле конфигурации, которого нет в скопированном каталоге, поскольку он находится в системной библиотеке файлов HAL. Эти файлы можно скопировать в каталог конфигурации пользователя и изменить по требованию пользователя для модификации или тестирования. Поскольку при поиске файлов HAL в первую очередь выполняется поиск в каталоге конфигурации пользователя, локальные модификации будут иметь преимущественную силу.

Селектор конфигурации создает символическую ссылку в каталоге конфигурации пользователя (с именем hallib), указывающую на системную библиотеку файлов HAL. Эта ссылка упрощает копирование файла библиотеки. Например, чтобы скопировать файл библиотеки core_sim.hal для внесения локальных изменений:

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

## 1.5 Обновление 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 Обновление до новой версии

В этом разделе описывается, как обновить LinuxCNC с версии 2.8.x до версии 2.9.y. Предполагается, что у вас есть установленная версия 2.8, которую вы хотите обновить.

Чтобы обновить LinuxCNC с версии старше 2.8, вам необходимо сначала обновление старой установки до 2.8, а затем следовать этим инструкциям. для обновления до новой версии.

Если у вас нет старой версии LinuxCNC для обновления, лучше всего выполнить новую установку новой версии, как описано в разделе Getting LinuxCNC.

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).

Чтобы обновить основные версии, такие как 2.8, до 2.9, когда у вас есть сетевое соединение на компьютере, вам необходимо отключить старые источники apt linuxcnc.org в файле /etc/apt/-sources.list и добавить новый источник apt linuxcnc.org для 2.9. , затем обновите LinuxCNC.

Детали будут зависеть от того, на какой платформе вы работаете. Откройте terminal, затем введите `lsb_release -ic`, чтобы найти эту информацию:

```
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.

Вам также нужно будет проверить, какое используется ядро реального времени:

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

Если вы видите (как указано выше) `-rt-` в имени ядра, значит, вы используете ядро preempt-rt и вам следует установить версию LinuxCNC ”uspace”. Вам также следует установить uspace для конфигураций ”sim” на ядрах, не работающих в реальном времени.

Если вы видите `-rtai-` в имени ядра, значит, вы используете RTAI в реальном времени. См. ниже версию LinuxCNC для установки. Пакеты RTAI доступны для Bookworm и Buster, но в настоящее время недоступны для Bullseye.

#### 1.5.1.1 Конфигурация источников Apt

- Откройте окно `Software Sources`. Этот процесс немного отличается на трех поддерживаемых платформах:
  - Debian:
    - Нажмите `Applications Menu`, затем `System`, затем `Synaptic Package Manager`.
    - В Synaptic нажмите меню `Settings`, затем нажмите `Repositories`, чтобы открыть окно `Software Sources`.
  - Ubuntu Precise:
    - Нажмите на значок `Dash Home` в левом верхнем углу.
    - В поле `Search` введите ”software”, затем щелкните значок `Ubuntu Software Center`.
    - В окне Ubuntu Software Center нажмите меню `Edit`, затем нажмите `Software Sources...`, чтобы открыть окно `Software Sources`.
  - Ubuntu Lucid:
    - Откройте меню `System`, затем `Administration`, затем `Synaptic Package Manager`.
    - В Synaptic нажмите меню `Settings`, затем нажмите `Repositories`, чтобы открыть окно `Software Sources`.
- В окне `Software Sources` выберите вкладку `Other Software`.
- Удалите или снимите флажки со всех старых записей linuxcnc.org (оставьте все строки, отличные от linuxcnc.org, как есть).
- Нажмите кнопку `Add` и добавьте новую строку apt. На разных платформах строка будет немного отличаться:

Table 1.2: 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` |

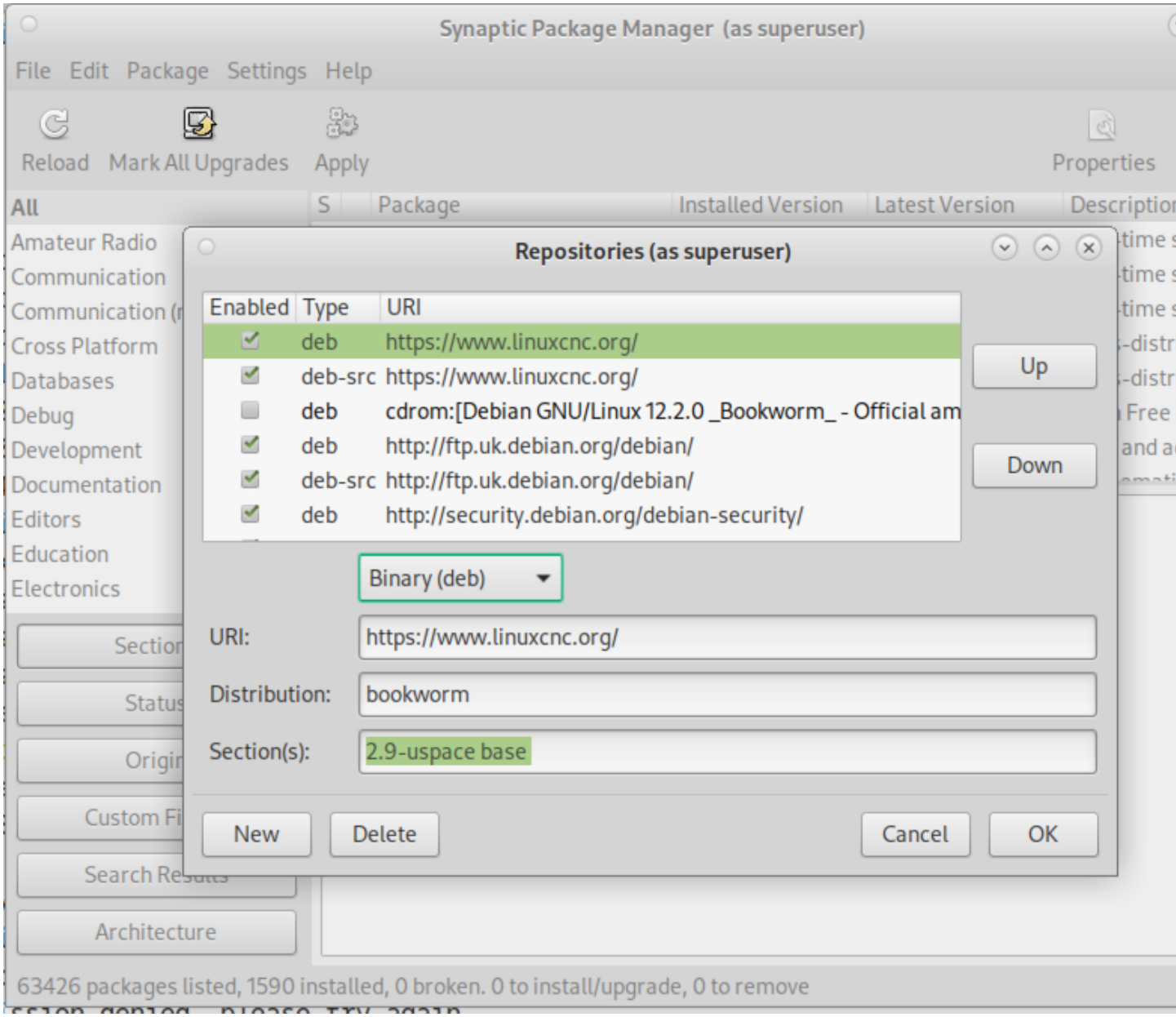



Figure 1.3: Figure with a screenshot of the repository configuration of the synaptic package manager.

- Нажмите `Add Source`, затем `Close` в окне Software Sources. Если появится окно с сообщением о том, что информация о доступном программном обеспечении устарела, нажмите кнопку `Reload`.

#### 1.5.1.2 Обновление до новой версии

Теперь ваш компьютер знает, где взять новую версию программного обеспечения, теперь нам нужно ее установить.

Процесс снова различается в зависимости от вашей платформы.

Debian используют диспетчер пакетов Synaptic.

- Откройте Synaptic, следуя инструкциям в разделе Настройка источников apt выше.
- Нажмите кнопку `Reload`.
- Используйте функцию поиска для поиска `linuxcnc`.
- Пакет называется «linuxcnc» для ядер RTAI и «linuxcnc-uspace» для preempt-rt.
- Установите флажок, чтобы отметить новые пакеты linuxcnc и linuxcnc-doc-* для обновления. Менеджер пакетов может выбрать несколько дополнительных пакетов для установки, чтобы удовлетворить зависимости, которые имеет новый пакет linuxcnc.
- Нажмите кнопку `Apply`, и ваш компьютер установит новый пакет. Старый пакет linuxcnc будет автоматически обновлен до нового.

#### 1.5.1.3 Ubuntu

- Нажмите на значок `Dash Home` в левом верхнем углу.
- В поле `Search` введите «update», затем щелкните значок `Update Manager`.
- Нажмите кнопку `Check`, чтобы просмотреть список доступных пакетов.
- Нажмите кнопку `Install Updates`, чтобы установить новые версии всех пакетов.

### 1.5.2 Обновление без сети

Для обновления без подключения к сети вам необходимо загрузить .deb, а затем установить его с помощью dpkg. Файлы .debs можно найти по адресу https://linuxcnc.org/dists/.

Вам нужно перейти по ссылке выше, чтобы найти правильный deb для вашей установки. Откройте terminal и введите «lsb_release -ic», чтобы найти название выпуска вашей ОС.

```
> 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 Обновление файлов конфигурации для версии 2.9

### 1.5.3.1 Более строгое обращение с подключаемыми интерпретаторами

Если вы просто запускаете обычный G-код и не знаете, что такое подключаемый интерпретатор, то этот раздел вас не касается.

Редко используемая функция LinuxCNC — это поддержка подключаемых интерпретаторов, управляем недокументированной настройкой INI `[TASK]INTERPRETER`.

Версии LinuxCNC до 2.9.0 использовались для обработки неправильной настройки `[TASK]INTERPRETER` путем автоматического возврата к использованию интерпретатора G-кода по умолчанию.

Начиная с версии 2.9.0, неправильное значение `[TASK]INTERPRETER` приведет к тому, что LinuxCNC откажется запускаться. Исправьте это условие, удалив настройку `[TASK]INTERPRETER` из вашего INI-файла, чтобы LinuxCNC использовал интерпретатор G-кода по умолчанию.

### 1.5.3.2 Canterp

Если вы просто запускаете обычный G-код и не используете подключаемый интерпретатор canterp, то этот раздел вас не касается.

В крайне маловероятном случае использования `canterp` знайте, что модуль перемещен из /usr/lib/libcanterp.so в /usr/lib/linuxcnc/canterp.so, и настройку `[TASK]INTERPRETER` соответственно, необходимо изменить с `libcanterp.so` на `canterp.so`.

### 1.5.3.3 Ограничения шпинделя в INI

Теперь можно добавлять настройки в раздел [SPINDLE] INI-файла

MAX_FORWARD_VELOCITY = 20000 The maximum spindle speed (in rpm)

MIN_FORWARD_VELOCITY = 3000 The minimum spindle speed (in rpm)

MAX_REVERSE_VELOCITY = 20000 Если этот параметр опущен, по умолчанию будет установлено значение MAX_FORWARD_VELOCITY.

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 - Принято, но в настоящее время ничего не делает

HOME_SEQUENCE = 0 - Принято, но в настоящее время ничего не делает

## 1.5.4 Updating Configuration Files for 2.10.y

Touchy: the Touchy MACRO entries should now be placed in a [MACROS] section of the INI rather than in the [TOUCHY] section. This is part of a process of commonising the INI setting between GUIs.

## 1.5.5 New HAL components

### 1.5.5.1 Не в реальном времени

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

#### 1.5.5.2 В реальном времени

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

### 1.5.6 Новые драйверы

Была представлена структура для управления устройствами ModBus с использованием последовательных портов на многих картах Mesa. http://linuxcnc.org/docs/2.9/html/drivers/mesa_modbus.html

Теперь включен новый драйвер GPIO для любого GPIO, поддерживаемого библиотекой gpiod: 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 Автоматический вход

#### 1.6.1.1 Debian

Debian Stretch uses the Xfce desktop environment by default, with the lightDM display manager lightDM. To get automatic login with Stretch:

- In a terminal, use the command:

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

- Make a note of the absolute path to the configuration file lightdm.conf.
- Edit that file with a pure text editor (gedit, nano, etc), as root.
- Find and uncomment the lines:

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

- Set autologin-user=your_user_name
- Save and reboot.

#### 1.6.1.2 Ubuntu

When you install LinuxCNC with the Ubuntu LiveCD the default is to have to log in each time you turn the computer on. To enable automatic login go to *System* > *Administration* > *Login Window*. If it is a fresh install the Login Window might take a second or three to pop up. You will have to have your password that you used for the install to gain access to the Login Window Preferences window. In the Security tab check off Enable Automatic Login and pick a user name from the list (that would be you).

### 1.6.2 Automatic Startup

To have LinuxCNC start automatically with your config after turning on the computer go to *System > Preferences > Sessions > Startup Applications*, click Add. Browse to your config and select the .ini file. When the file picker dialog closes, add linuxcnc and a space in front of the path to your .ini file.

Example:

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

В документации соответствующий файл .ini называется INI-файлом.

### 1.6.3 Терминал

Многое требуется сделать из терминала, например, проверить буфер сообщений ядра с помощью *dmesg*. В Ubuntu и Linux Mint есть сочетание клавиш Ctrl
Alt + t. В Debian Stretch не определены сочетания клавиш. Его можно легко создать с помощью «Диспетчера конфигураций». Большинство современных файловых менеджеров поддерживают правую клавишу для открытия терминала, просто убедитесь, что вы щелкаете правой кнопкой мыши по пустой области или каталогу, а не по имени файла. Большинство ОС имеют терминал в качестве пункта меню, обычно в разделе Аксессуары.

### 1.6.4 Страницы руководства

Справочная страница ”man page” (сокращение от manual page) — это форма документации по программному обеспечению, обычно встречающаяся в UNIX или UNIX -подобных операционных системах, таких как Linux.

Чтобы просмотреть справочную страницу, откройте терминал, чтобы узнать что-нибудь о команде find в окне терминала напечатайте:

```
man find
```

Используйте клавиши Page Up и Page Down для просмотра справочной страницы и клавишу Q для выхода из просмотра.

---

**Note**
Viewing the man page from the terminal may not get the expected man page. For example if you type in man abs you will get the C abs not the LinuxCNC abs. It is best to view the LinuxCNC man pages in the HTML documents.

---

### 1.6.5 List Modules

Sometimes when troubleshooting you need to get a list of modules that are loaded. In a terminal window type:

```
lsmod
```

If you want to send the output from lsmod to a text file in a terminal window type:

```
lsmod > mymod.txt
```

The resulting text file will be located in the home directory if you did not change directories when you opened up the terminal window and it will be named mymod.txt or what ever you named it.

## 1.6.6 Editing a Root File

When you open the file browser and you see the owner of the file is root you must do extra steps to edit that file. Editing some root files can have bad results. Be careful when editing root files. Generally, you can open and view most root files, but they will open in *read only* mode.

### 1.6.6.1 The Command Line Way

Open a terminal and type

```
sudo gedit
```

Open the file with File > Open > Edit

### 1.6.6.2 The GUI Way

1. Щелкните правой кнопкой мыши по рабочему столу и выберите Создать панель запуска.
2. Type a name in like sudo edit.
3. Type *gksudo "gnome-open %u"* as the command and save the launcher to your desktop.
4. Drag a file onto your launcher to open and edit.

### 1.6.6.3 Root Access

In Ubuntu you can become root by typing in "sudo -i" in a terminal window then typing in your password. Be careful, because you can really foul things up as root if you don't know what you're doing.

## 1.6.7 Terminal Commands

### 1.6.7.1 Working Directory

To find out the path to the present working directory in the terminal window, type:

```
pwd
```

### 1.6.7.2 Изменение каталогов

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

```
cd ..
```

To move up two levels in the terminal window type:

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

To move directly to your home directory, in the terrminal window use the cd command with no arguments:

```
cd
```

To move down to the linuxcnc/configs subdirectory in the terminal window type:

```
cd linuxcnc/configs
```

#### 1.6.7.3 Listing files in a directory

To view a list of all the files and subdirectories in the terminal window type:

```
dir
```

or

```
ls
```

#### 1.6.7.4 Finding a File

The find command can be a bit confusing to a new Linux user. The basic syntax is:

```
find starting-directory parameters actions
```

For example to find all the .ini files in your linuxcnc directory you first need to use the pwd command to find out the directory.

Open a new terminal window and type:

```
pwd
```

And pwd might return the following result:

```
/home/joe
```

With this information put the command together like this:

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

The -name is the name of the file your looking for and the -print tells it to print out the result to the terminal window. The *.ini tells find to return all files that have the .ini extension. The backslash is needed to escape the shell meta-characters. See the find man page for more information on find.

#### 1.6.7.5 Searching for Text

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

This will find all the files that contain the *text to search for* in the current directory and all the subdirectories below it, while ignoring the case. The -i is for ignore case and the -r is for recursive (include all subdirectories in the search). The -l option will return a list of the file names, if you leave the -l off you will also get the text where each occurrence of the "text to search for" is found. The * is a wild card for search all files. See the grep man page for more information.

#### 1.6.7.6 Diagnostic Messages

To view the diagnostic messages use "dmesg" from the command window. To save the diagnostic messages to a file use the redirection operator >, like this:

```
dmesg > bootmsg.txt
```

The contents of this file can be copied and pasted on line to share with people trying to help you diagnose your problem.

To clear the message buffer type this:

```
sudo dmesg -c
```

Это может быть полезно сделать непосредственно перед запуском LinuxCNC, чтобы была только запись информации, связанной с текущим запуском LinuxCNC.

Чтобы найти адрес встроенного параллельного порта, используйте grep для фильтрации информации из dmesg.

After boot up open a terminal and type:

```
dmesg|grep parport
```

## 1.6.8 Convenience Items

### 1.6.8.1 Terminal Launcher

If you want to add a terminal launcher to the panel bar on top of the screen you typically can right click on the panel at the top of the screen and select ”Add to Panel”. Select Custom Application Launcher and Add. Give it a name and put gnome-terminal in the command box.

## 1.6.9 Hardware Problems

### 1.6.9.1 Hardware Info

To find out what hardware is connected to your motherboard in a terminal window type:

```
lspci -v
```

### 1.6.9.2 Разрешение монитора

Во время установки Ubuntu пытается определить настройки монитора. Если это не сработает, у вас останется обычный монитор с максимальным разрешением 800x600.

Инструкции по устранению этого находятся здесь:

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

## 1.6.10 Пути

**Относительные пути** Относительные пути базируются на каталоге запуска, который является каталогом, содержащим INI-файл. Использование относительных путей может облегчить перемещени конфигураций, но требует хорошего понимания спецификаторов пути Linux.

```
./f0 b''cb''b''ob''b''вb''b''пb''b''ab''b''дb''b''ab''b''eb''b''тb'' b''cb'' f0, b''нb''b' ↩
    'ab''b''пb''b''pb''b''иb''b''мb''b''eb''b''pb'', b''фb''b''ab''b''йb''b''лb'' b''cb'' b' ↩
    'иb''b''мb''b''eb''b''нb''b''eb''b''мb'' f0 b''вb'' b''кb''b''ab''b''тb''b''ab''b''лb''b ↩
    ''ob''b''гb''b''eb'' b''зb''b''ab''b''пb''b''уb''b''cb''b''кb''b''ab''
../f1 b''cb''b''cb''b''ыb''b''лb''b''ab''b''eb''b''тb''b''cb''b''яb'' b''нb''b''ab'' b' ↩
    'фb''b''ab''b''йb''b''лb'' f1 b''вb'' b''pb''b''ob''b''дb''b''иb''b''тb''b''eb''b''лb''b ↩
    ''ьb''b''cb''b''кb''b''ob''b''мb'' b''кb''b''ab''b''тb''b''ab''b''лb''b''ob''b''гb''b' ↩
    'eb''
../../f2 b''cb''b''cb''b''ыb''b''лb''b''ab''b''eb''b''тb''b''cb''b''яb'' b''нb''b''ab'' b' ↩
    'фb''b''ab''b''йb''b''лb'' f2 b''вb'' b''pb''b''ob''b''дb''b''иb''b''тb''b''eb''b''лb''b ↩
    ''ьb''b''cb''b''кb''b''ob''b''мb'' b''кb''b''ab''b''тb''b''ab''b''лb''b''ob''b''гb''b' ↩
    'eb''
../../../f3 b''иb'' b''тb''.b''дb''.
```

# Chapter 2

# Общая информация для пользовател

## 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 Пользовательское введение в LinuxCNC

### 2.2.1 Введение

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

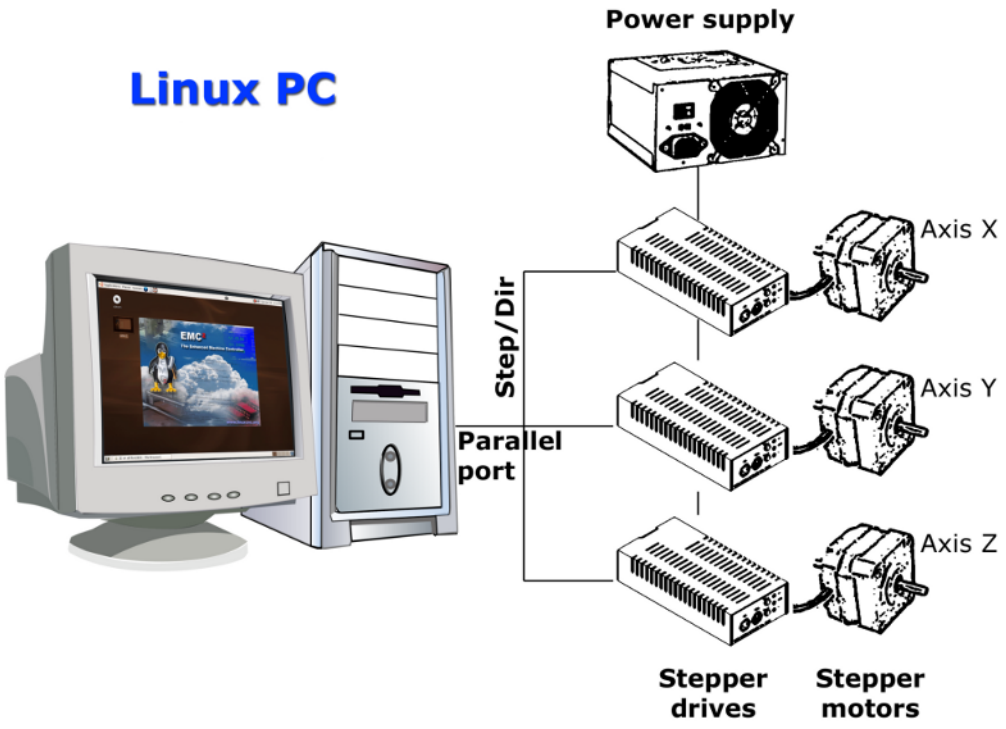



Figure 2.1: Simple LinuxCNC Controlled Machine

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 Графические интерфейсы пользователя

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:

**AXIS**
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:

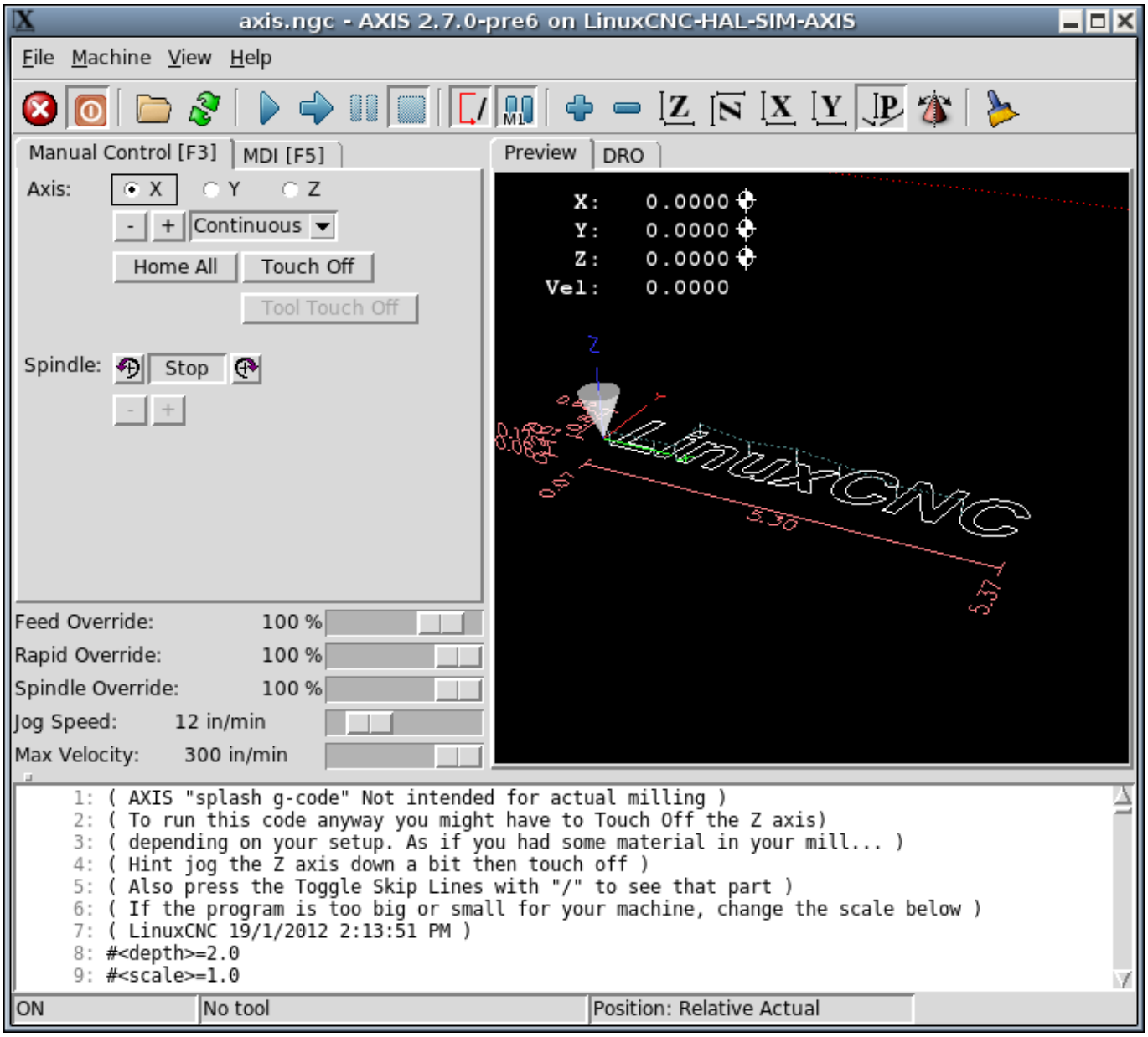



Figure 2.2: AXIS, the standard keyboard GUI interface

**Touchy**

Touchy, a touch screens GUI:

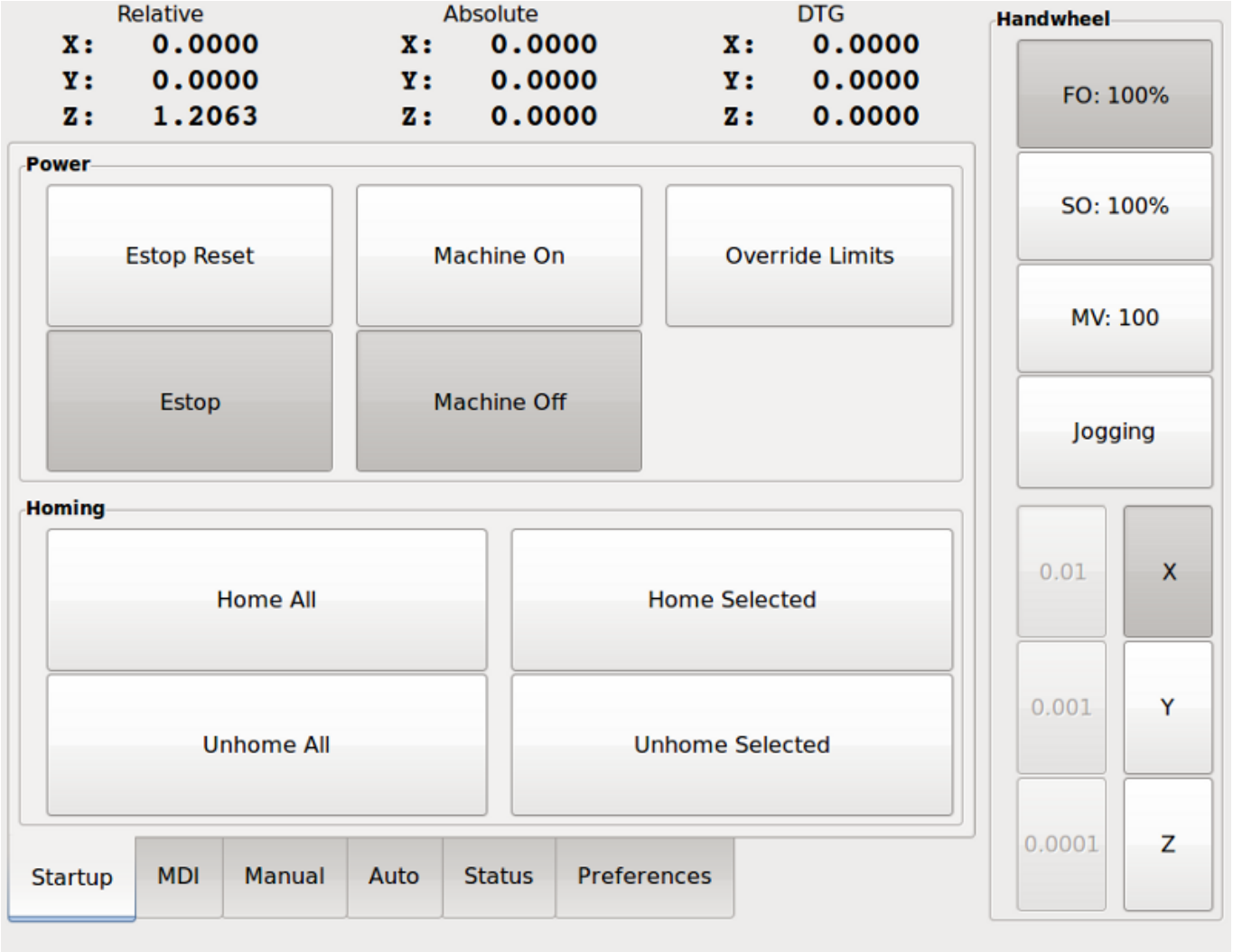



Figure 2.3: Touchy, a touch screen GUI

**Gscreen**

Gscreen, a user-configurable touch screen GUI:

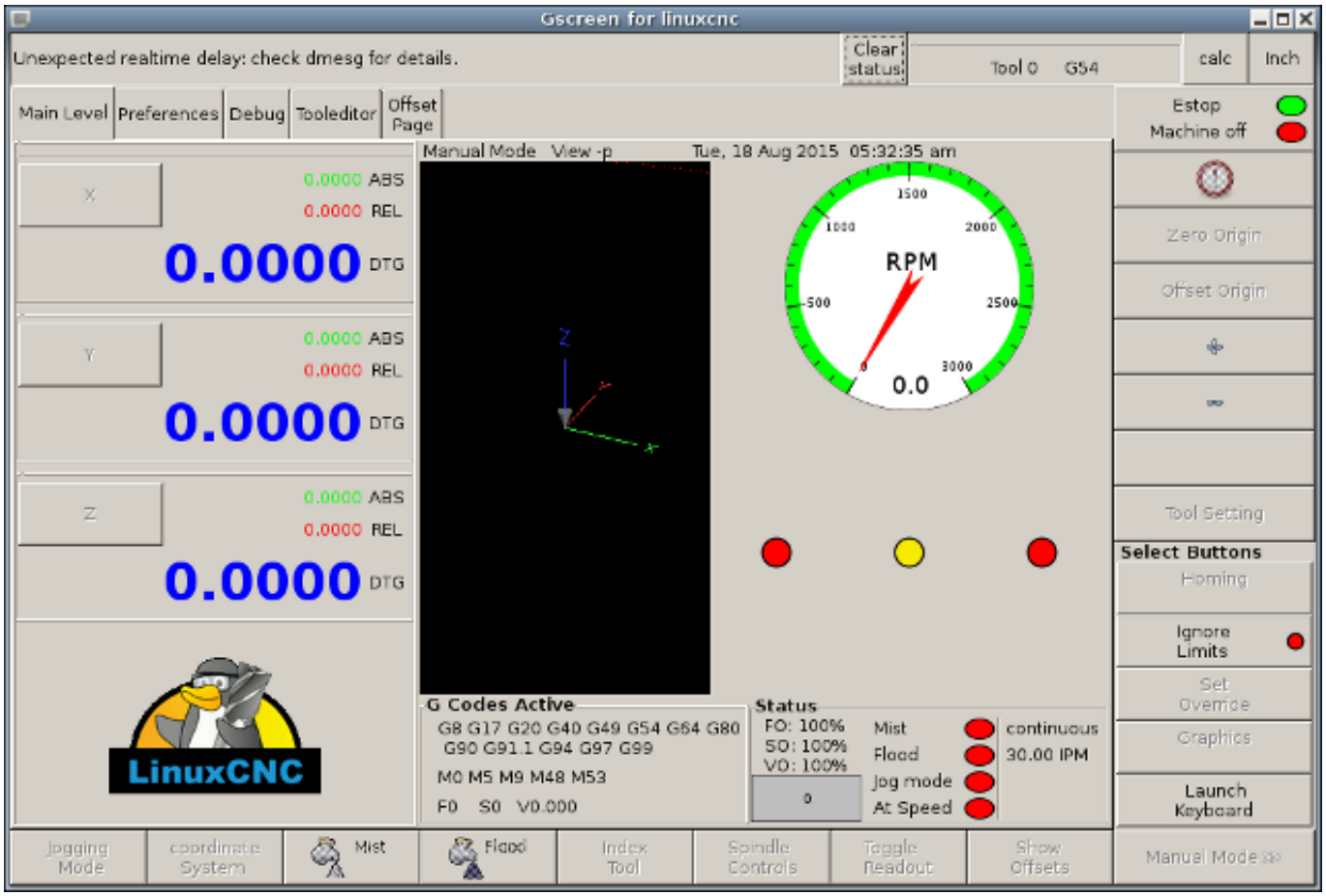



Figure 2.4: Gscreen, a configurable base touch screen GUI

**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:

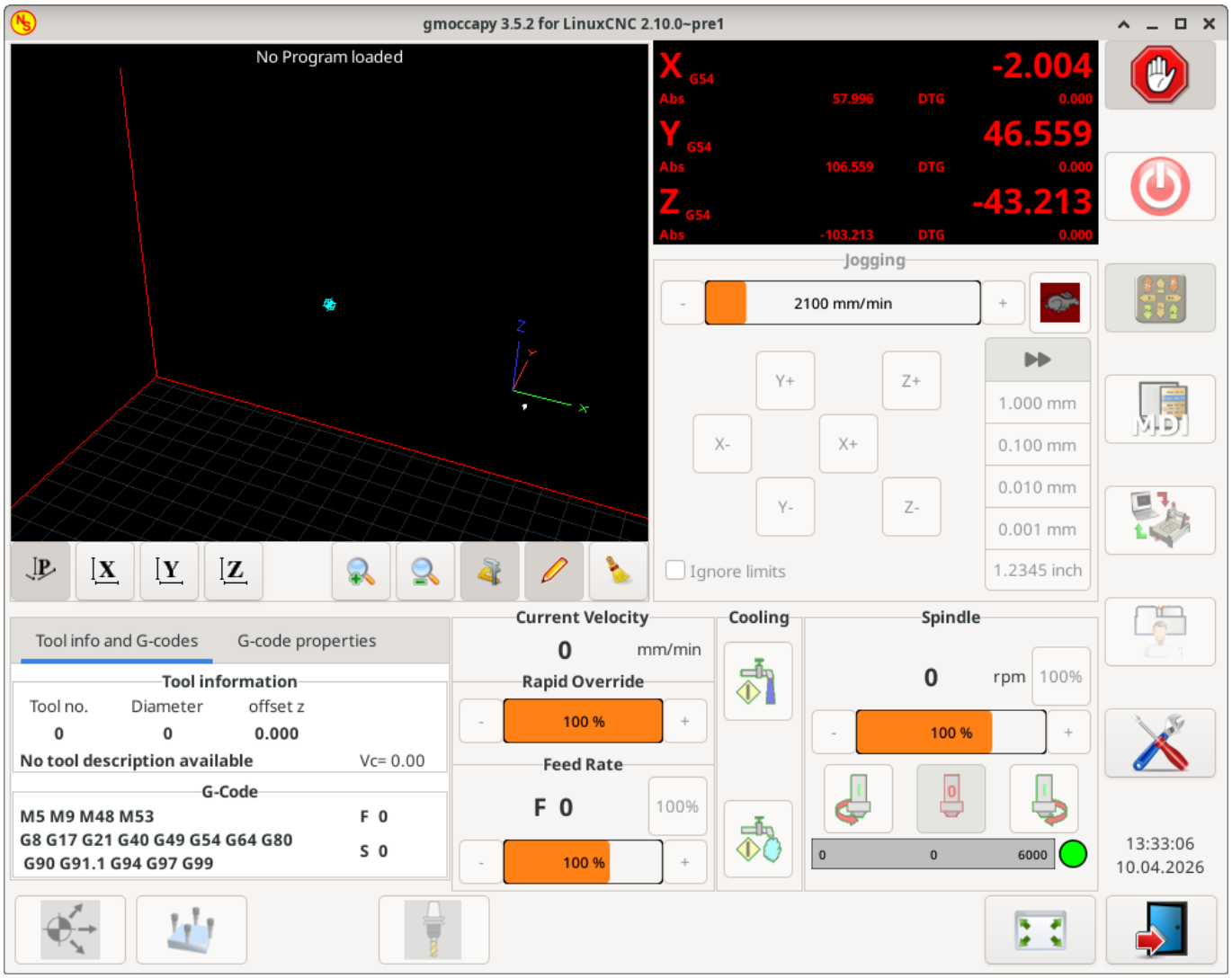



Figure 2.5: GMOCCAPY, a touch screen GUI based on Gscreen

**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:

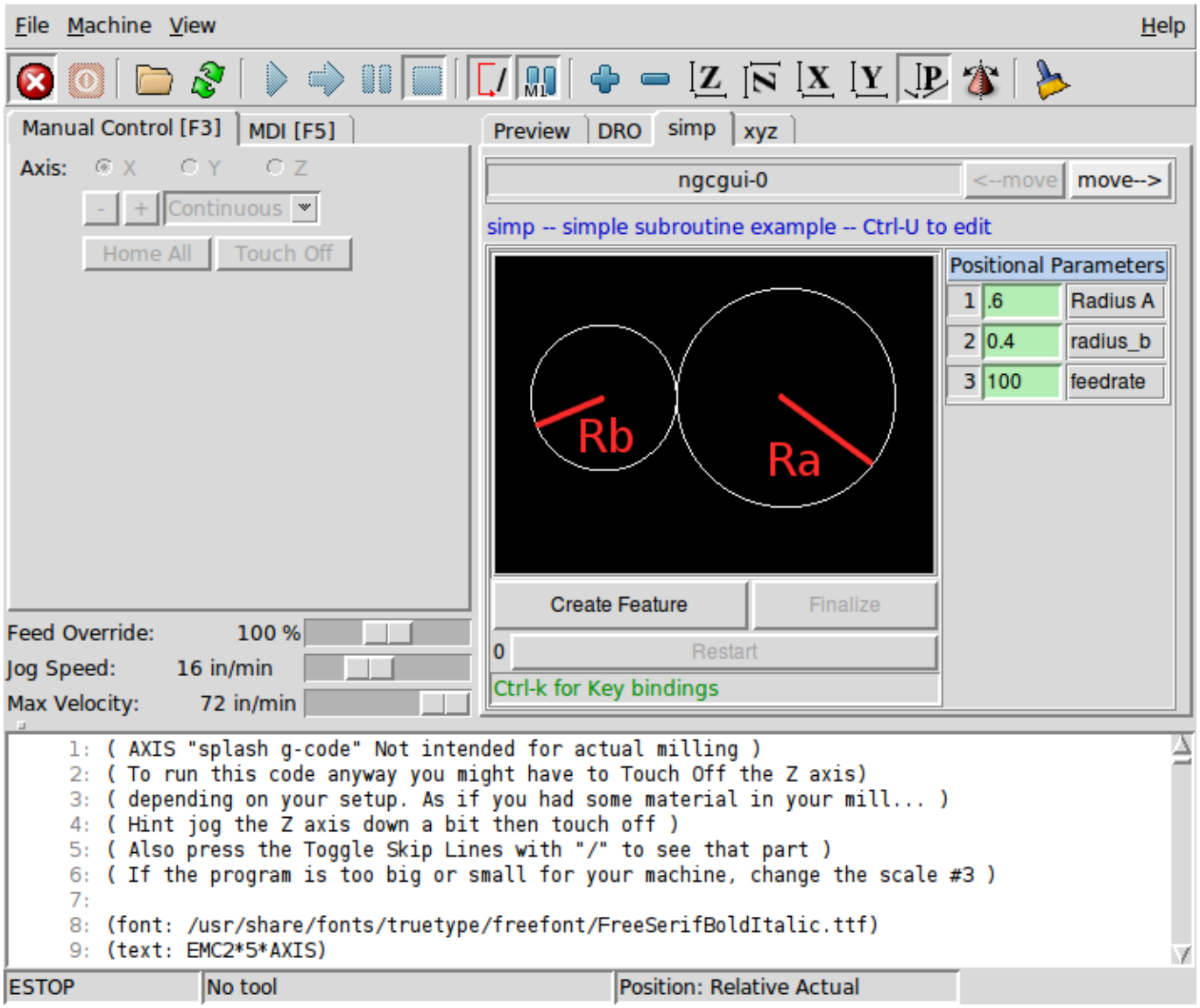



Figure 2.6: NGCGUI, a graphical interface integrated into AXIS

**TkLinuxCNC**

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

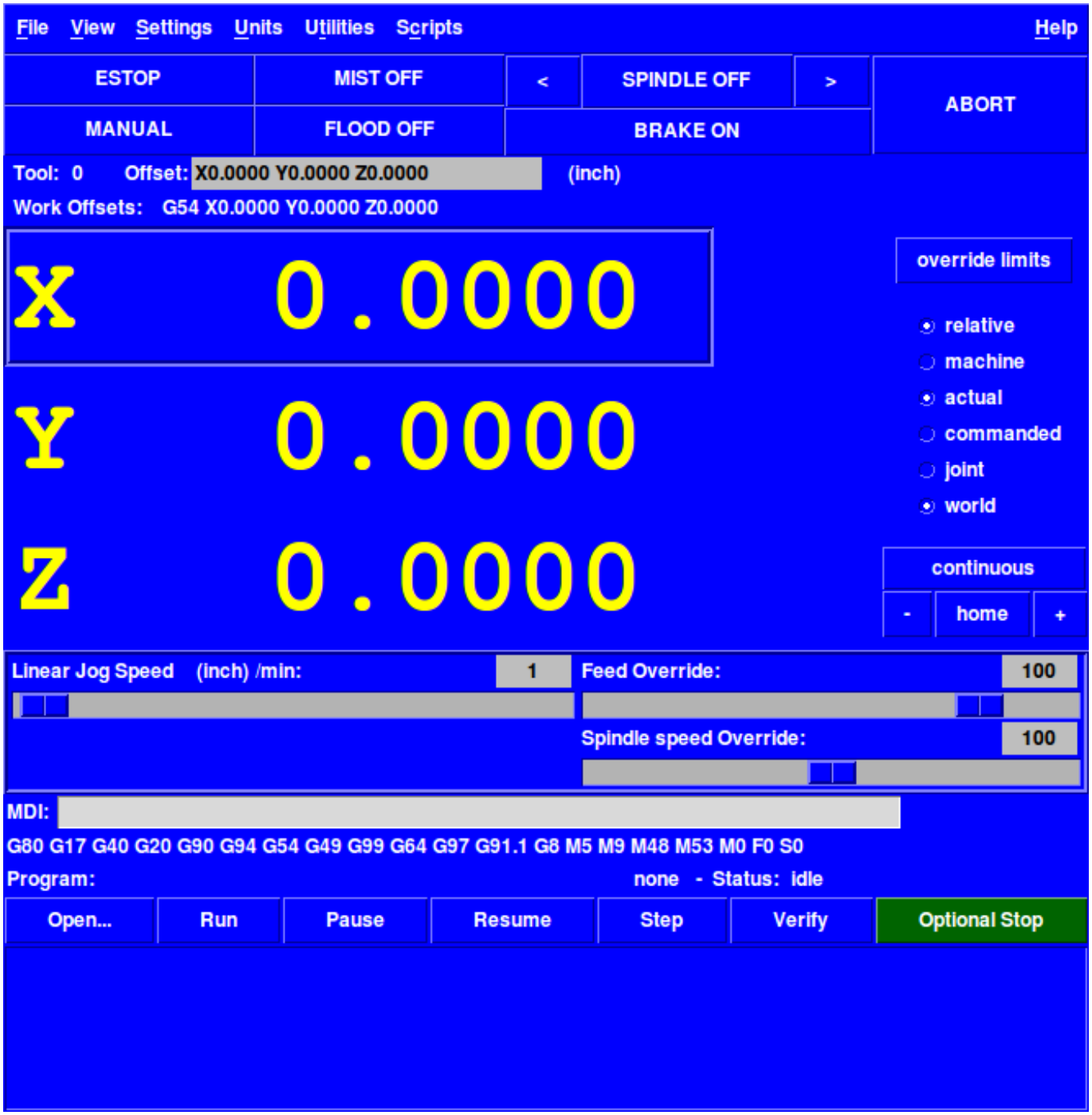



Figure 2.7: TkLinuxCNC graphical interface

**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.

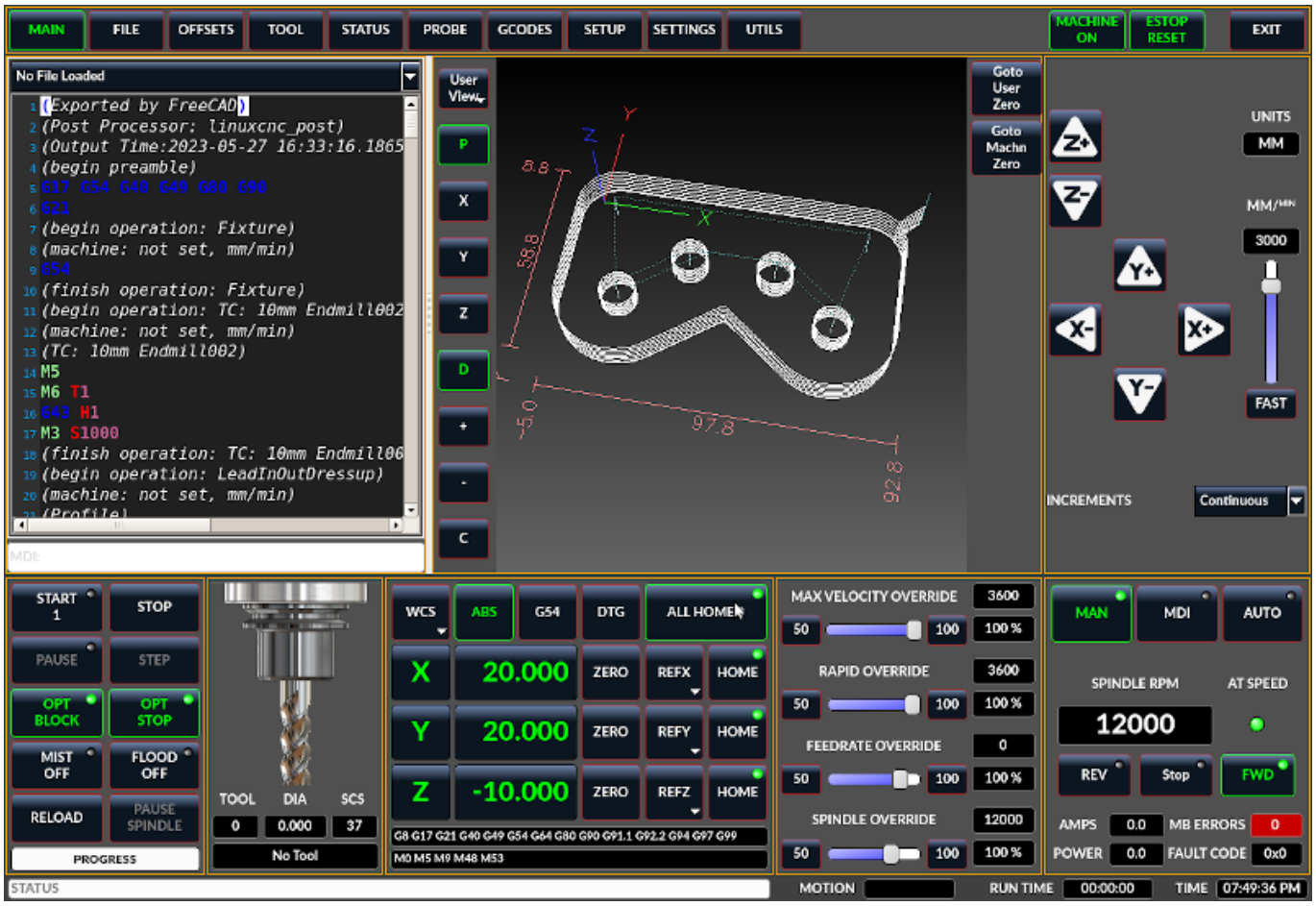



Figure 2.8: QtDragon, a touch screen GUI based on QtVCP

**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.

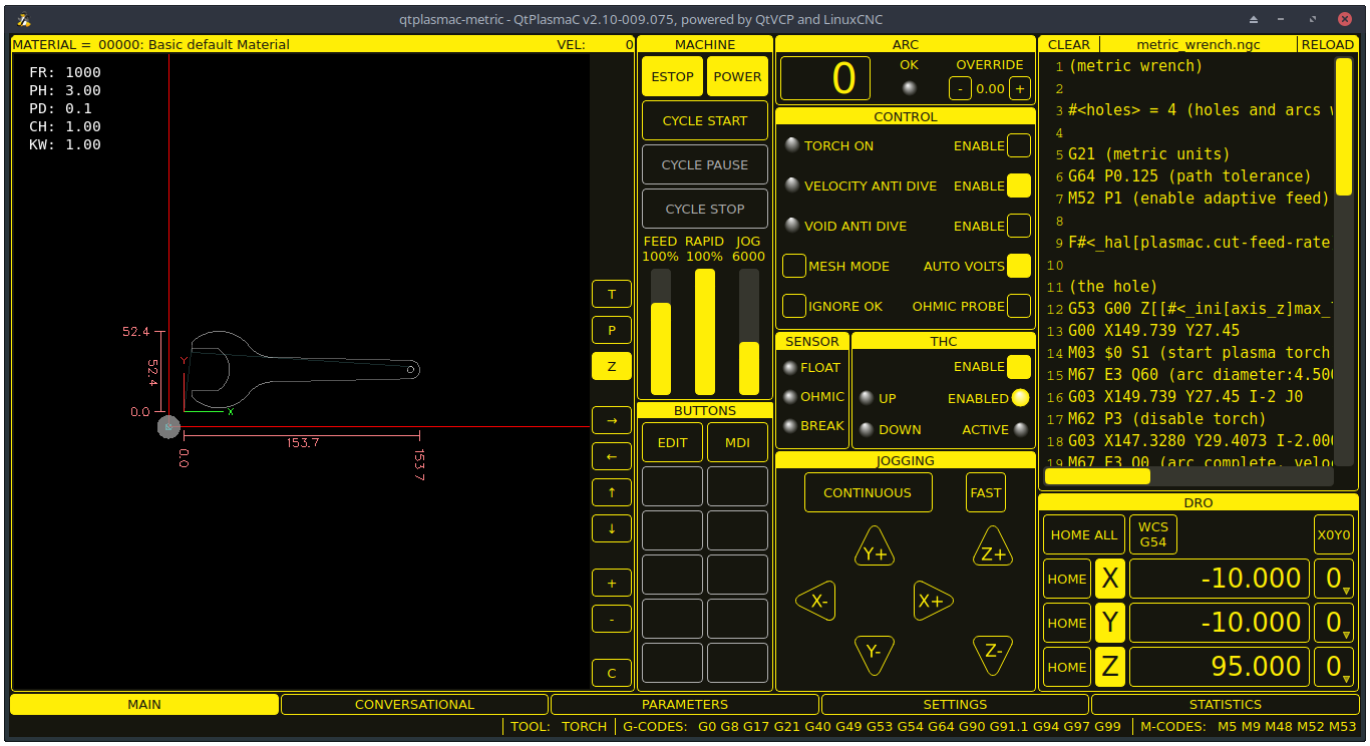



Figure 2.9: QtPlasmaC, a touch screen plasma cutting GUI based on QtVCP

### 2.2.4 Интерфейсы пользователя

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 Виртуальные панели управления

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.

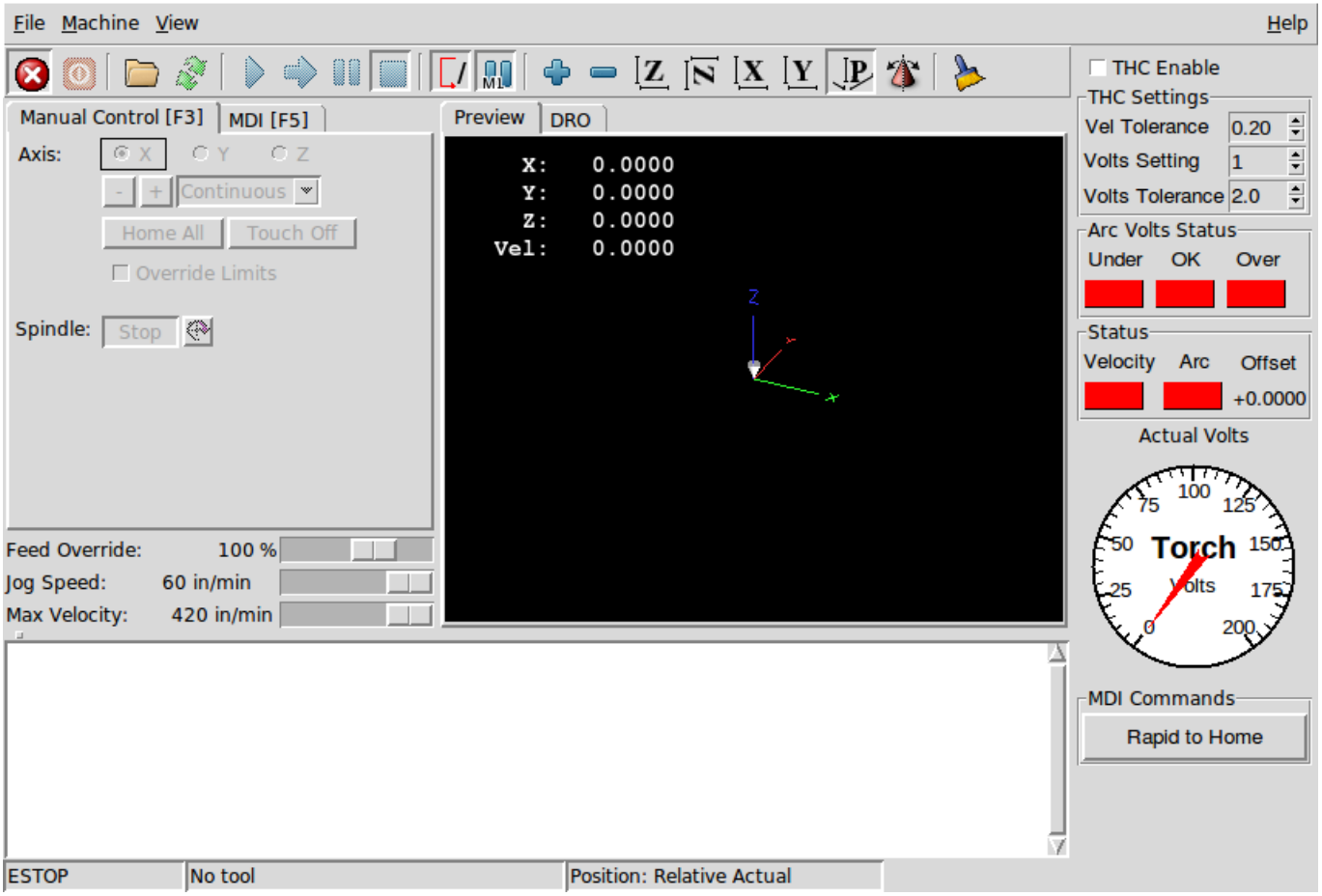



Figure 2.10: PyVCP Example Embedded Into AXIS GUI

**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:

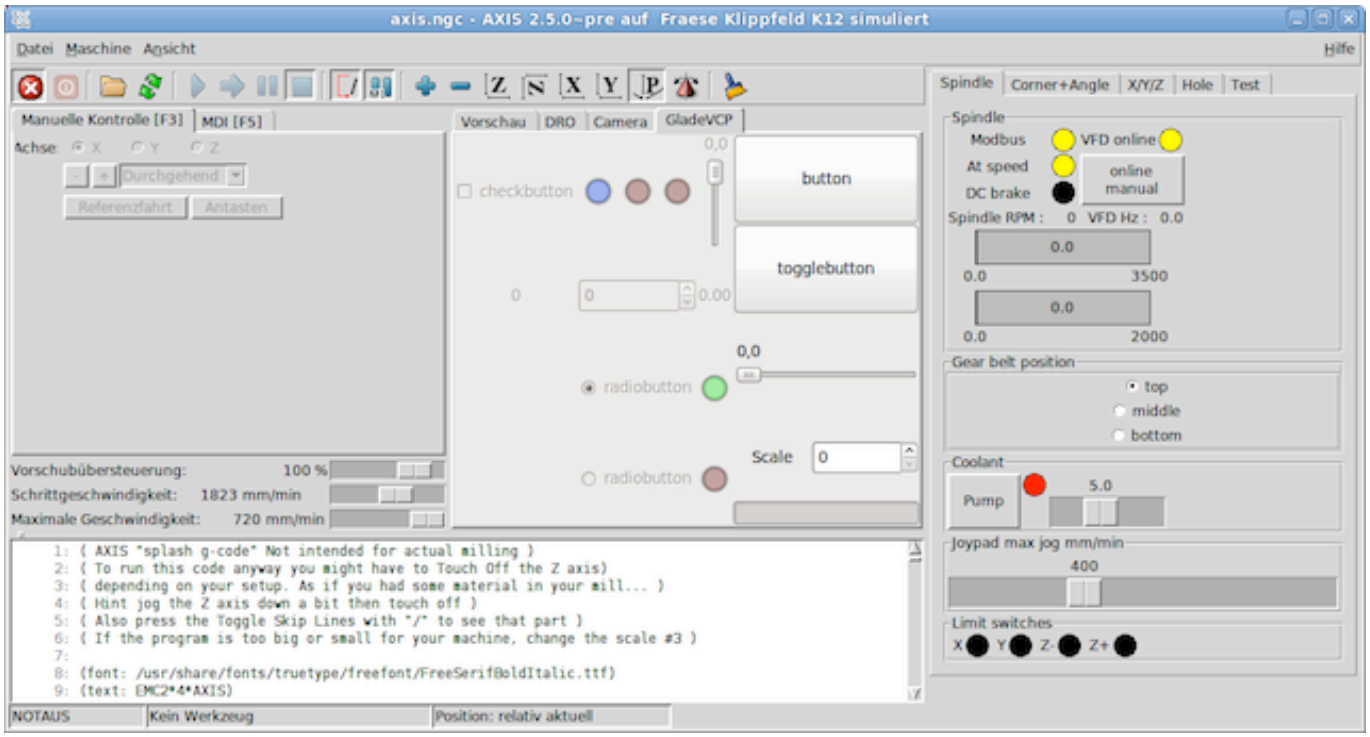



Figure 2.11: GladeVCP Example Embedded Into AXIS GUI

**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:

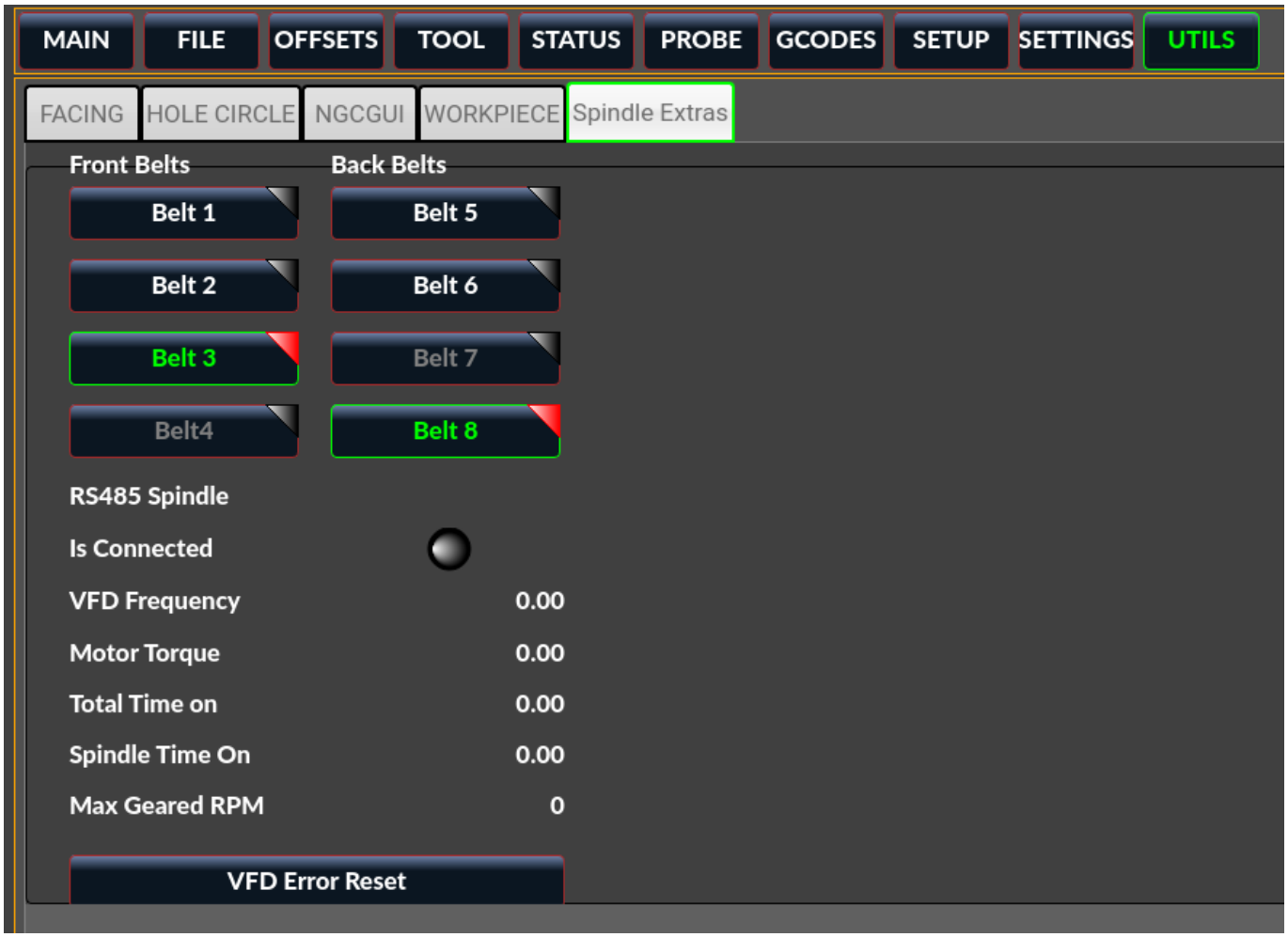



Figure 2.12: QtVCP Example Embedded Into QtDragon GUI

### 2.2.6 Языки

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 Важные пользовательские концепции

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

### 2.3.1 Управление траекторией

#### 2.3.1.1 Планировщик траектории

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 is linear move, X1 is the destination, F10 is the speed)
```

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.

For more information on the Trajectory Planner INI options see the Trajectory Section in the INI chapter.

#### 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:

- It can decelerate to a stop exactly at the coordinates of the corner, and then accelerate in the new direction.
- It can also do what is called blending, which is to keep the feed rate up while going through the corner, making it necessary to round the corner off in order to obey machine constraints.

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 Программирование планировщика

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.

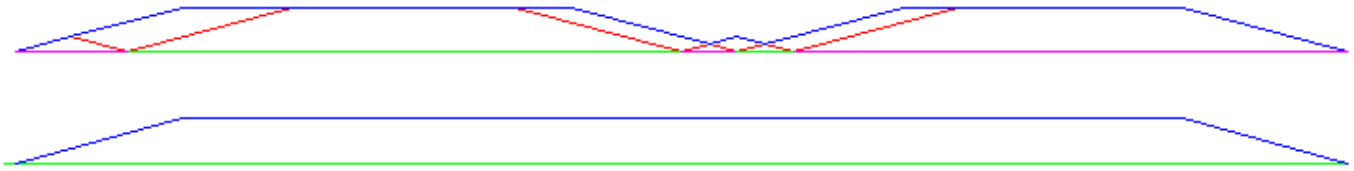


Figure 2.13: Naive CAM Detector

#### 2.3.1.4 Планирование перемещений

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 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 Coordinate Systems

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.

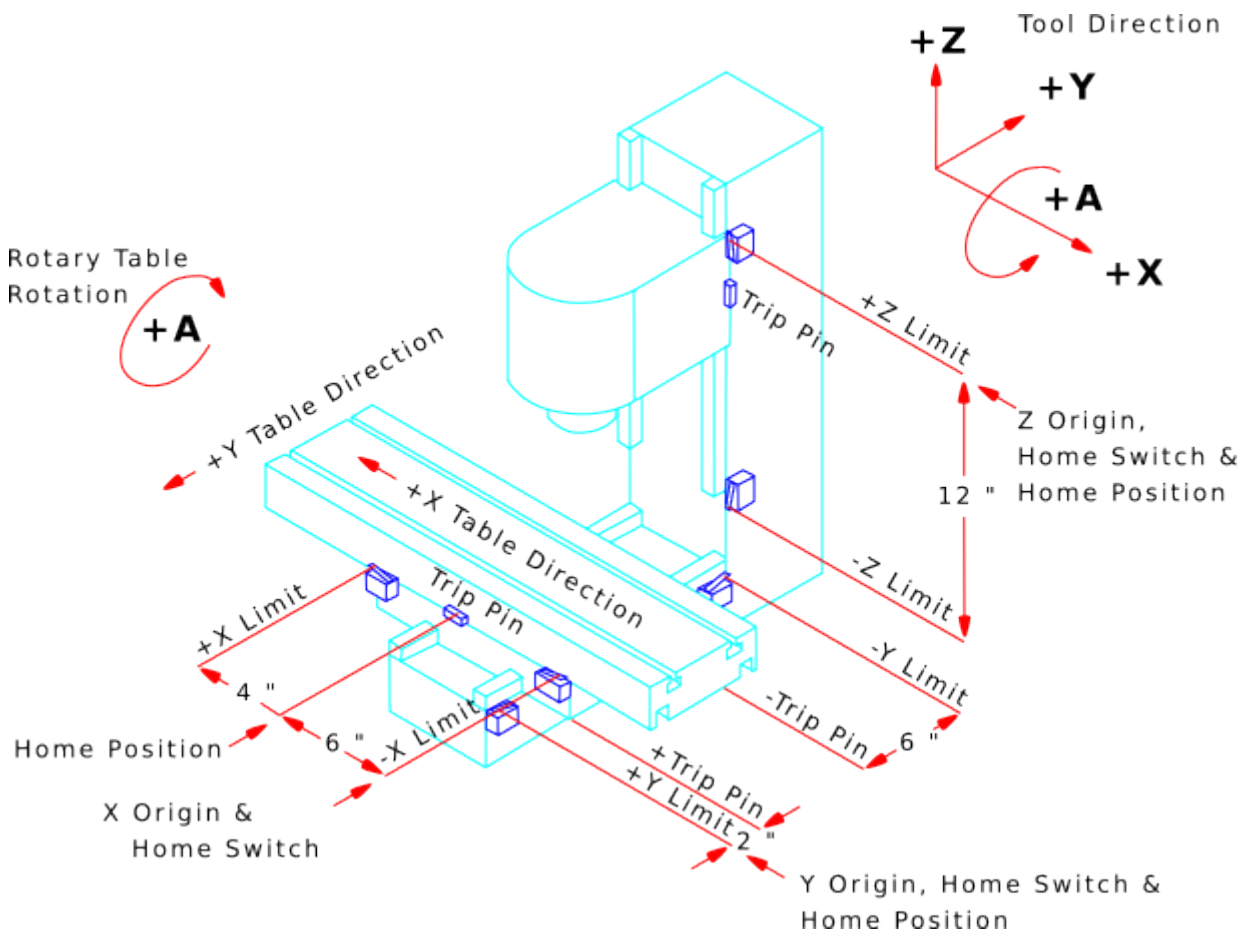



Figure 2.14: Typical Mill Configuration

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

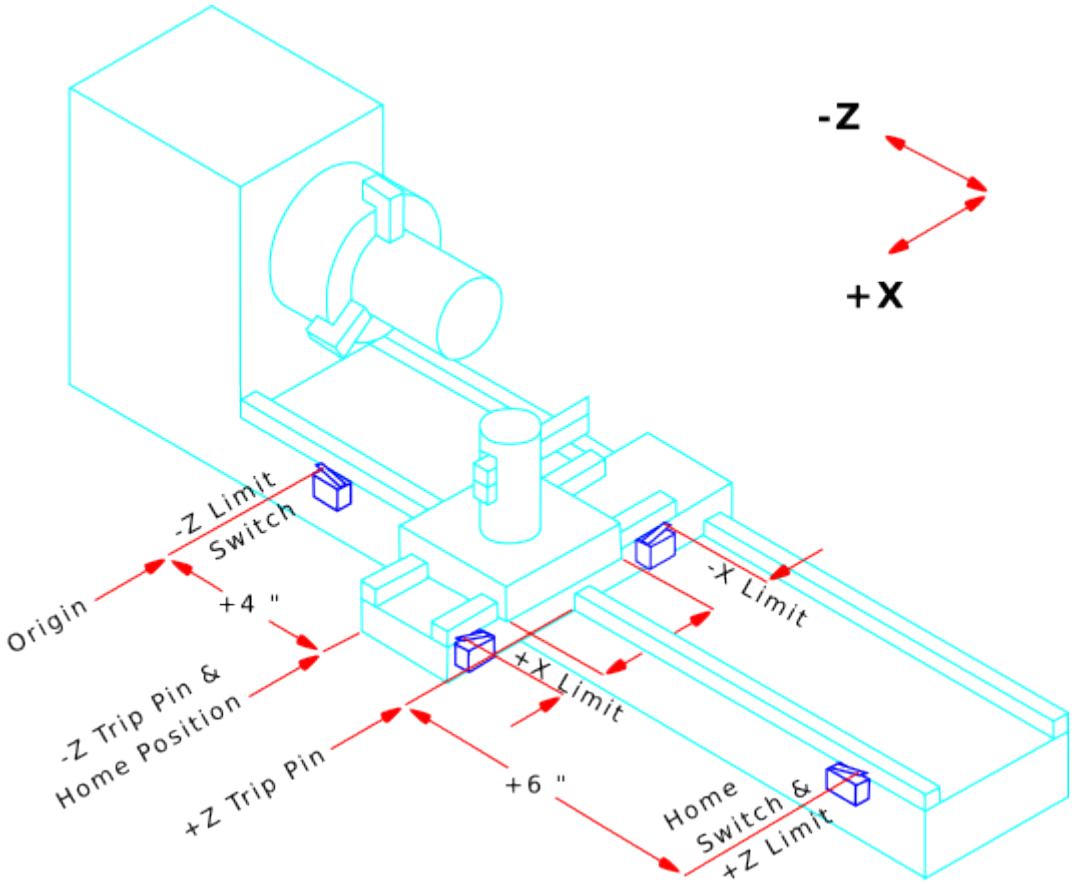



Figure 2.15: Typical Lathe Configuration

## 2.4 Начиная с LinuxCNC

### 2.4.1 Running LinuxCNC

LinuxCNC is started with the script file *linuxcnc*.

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

**linuxcnc script options**

```
linuxcnc: b''3b''b''ab''b''nb''b''yb''b''cb''b''kb'' 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

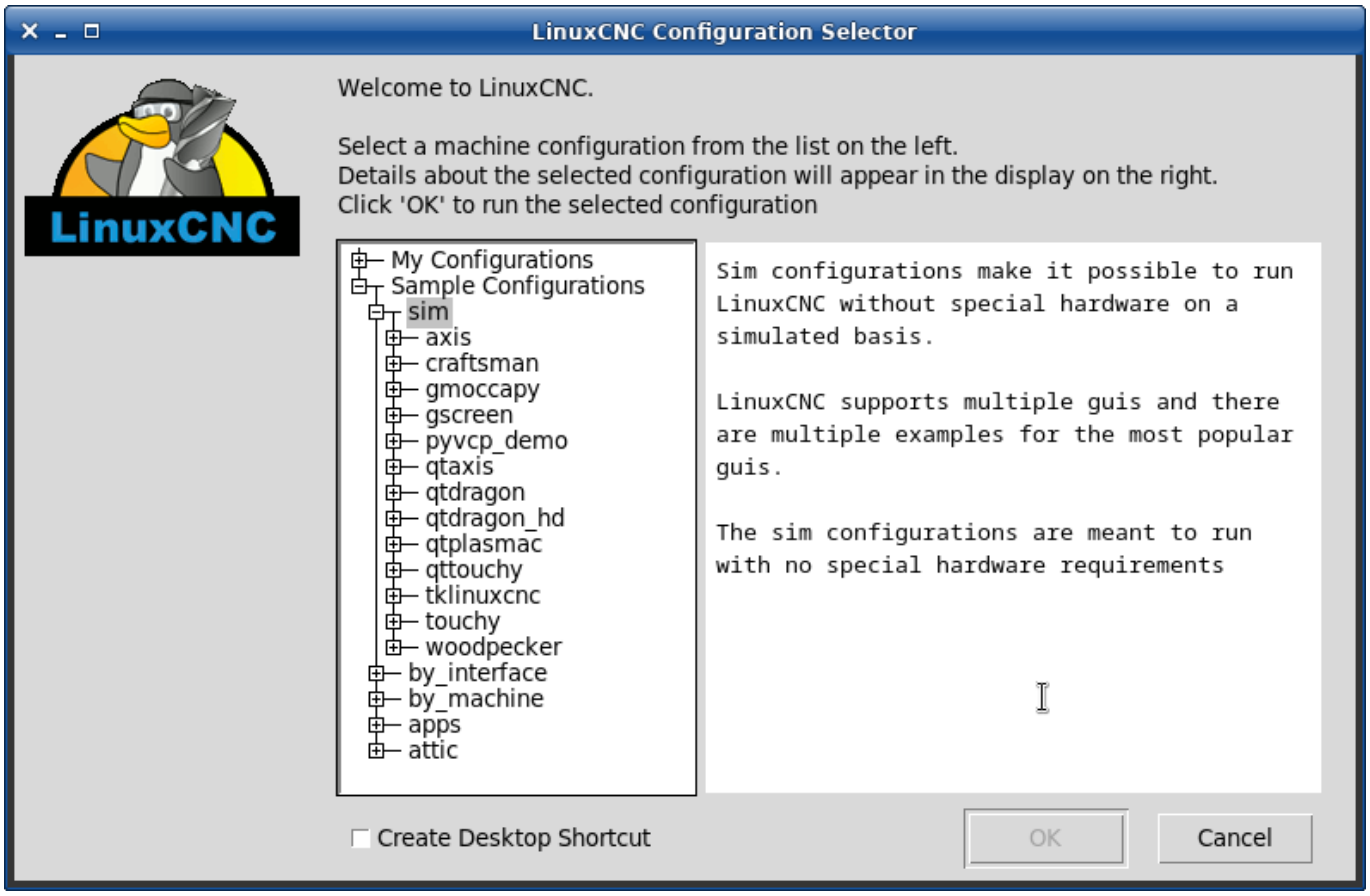
  $ linuxcnc [Options] -l
    Use the previously used configuration INI file
```

```
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 Селектор конфигурации

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 Mechanical Components

A CNC machine has many mechanical components that may be controlled or may affect the way in which control is exercised. This section describes the subset of those components that interact with the Interpreter. Mechanical components that do not interact directly with the Interpreter, such as the jog buttons, are not described here, even if they affect control.

#### 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.

---

**Note**
If the motion of mechanical components is not independent, as with hexapod machines, the RS274/NGC language and the canonical machining functions will still be usable, as long as the lower levels of control know how to control the actual mechanisms to produce the same relative motion of tool and workpiece as would be produced by independent axes. This is called *kinematics*.

---

**Note**
With LinuxCNC, the case of the XYYZ gantry machine with two motors for one axis is better handled by the kinematics than by an additional linear axis.

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

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

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

#### 2.5.1.2 Spindle

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 Coolant

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 Переопределение подачи и скорости

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 Block Delete Switch

A CNC machine can have a block delete switch. See the Block Delete section.

#### 2.5.1.6 Optional Program Stop Switch

A CNC machine can have an optional program stop switch. See the Optional Program Stop section.

### 2.5.2 Control and Data Components

#### 2.5.2.1 Linear Axes

The X, Y, and Z axes form a standard right-handed coordinate system of orthogonal linear axes. Positions of the three linear motion mechanisms are expressed using coordinates on these axes.

The U, V and W axes also form a standard right-handed coordinate system. X and U are parallel, Y and V are parallel, and Z and W are parallel (when A, B, and C are rotated to zero).

#### 2.5.2.2 Rotational Axes

The rotational axes are measured in degrees as wrapped linear axes in which the direction of positive rotation is counterclockwise when viewed from the positive end of the corresponding X, Y, or Z-axis. By *wrapped linear axis*, we mean one on which the angular position increases without limit (goes towards plus infinity) as the axis turns counterclockwise and deceases without limit (goes towards minus infinity) as the axis turns clockwise. Wrapped linear axes are used regardless of whether or not there is a mechanical limit on rotation.

Clockwise or counterclockwise is from the point of view of the workpiece. If the workpiece is fastened to a turntable which turns on a rotational axis, a counterclockwise turn from the point of view of the workpiece is accomplished by turning the turntable in a direction that (for most common machine configurations) looks clockwise from the point of view of someone standing next to the machine. [3]

#### 2.5.2.3 Контрольная точка

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 Coordinated Linear Motion

To drive a tool along a specified path, a machining center must often coordinate the motion of several axes. We use the term *coordinated linear motion* to describe the situation in which, nominally, each axis moves at constant speed and all axes move from their starting positions to their end positions at the same time. If only the X, Y, and Z axes (or any one or two of them) move, this produces motion in a straight line, hence the word *linear* in the term. In actual motions, it is often not possible to maintain constant speed because acceleration or deceleration is required at the beginning and/or end of the motion. It is feasible, however, to control the axes so that, at all times, each axis has completed the same fraction of its required motion as the other axes. This moves the tool along same path, and we also call this kind of motion coordinated linear motion.

Coordinated linear motion can be performed either at the prevailing feed rate, or at traverse rate, or it may be synchronized to the spindle rotation. If physical limits on axis speed make the desired rate unobtainable, all axes are slowed to maintain the desired path.

#### 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. If any of XYZ are moving, F is in units per minute in the XYZ cartesian system, and all other axes (ABCUVW) move so as to start and stop in coordinated fashion.
2. Otherwise, if any of UVW are moving, F is in units per minute in the UVW cartesian system, and all other axes (ABC) move so as to start and stop in coordinated fashion.

[3] If the parallelism requirement is violated, the system builder will have to say how to distinguish clockwise from counter-clockwise.

3. Otherwise, the move is pure rotary motion and the F word is in rotary units in the ABC *pseudo-cartesian* system.

#### 2.5.2.6 Охлаждение

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

#### 2.5.2.7 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 Units

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 Current Position

The controlled point is always at some location called the *current position*, and the controller always knows where that is. The numbers representing the current position must be adjusted in the absence of any axis motion if any of several events take place:

1. Length units are changed.
2. Tool length offset is changed.
3. Coordinate system offsets are changed.

#### 2.5.2.10 Selected Plane

There is always a *selected plane*, which must be the XY-plane, the YZ-plane, or the XZ-plane of the machining center. The Z-axis is, of course, perpendicular to the XY-plane, the X-axis to the YZ-plane, and the Y-axis to the XZ-plane.

#### 2.5.2.11 Tool Carousel

Zero or one tool is assigned to each slot in the tool carousel.

#### 2.5.2.12 Tool Change

A machining center may be commanded to change tools.

#### 2.5.2.13 Pallet Shuttle

The two pallets may be exchanged by command.

#### 2.5.2.14 Speed Override

The speed override buttons can be activated (they function normally) or rendered inoperative (they no longer have any effect). The RS274/NGC language has a command that activates all the buttons and another that disables them. See inhibition and activation speed correctors. See also here for further details.

#### 2.5.2.15 Path Control Mode

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

**exact stop mode**
: In exact stop mode, the machine stops briefly at the end of each programmed move.

**exact path mode**
: In exact path mode, the machine follows the programmed path as exactly as possible, slowing or stopping if necessary at sharp corners of the path.

**continuous mode**
: In continuous mode, sharp corners of the path may be rounded slightly so that the feed rate may be kept up (but by no more than the tolerance, if specified).

See sections G61 and G64.

### 2.5.3 Interpreter Interaction with Switches

The Interpreter interacts with several switches. This section describes the interactions in more detail. In no case does the Interpreter know what the setting of any of these switches is.

#### 2.5.3.1 Клавиши переопределения подачи и скорости

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 reacts to the speed and feed override settings when these switches are enabled.

See the M48 M49 Override section for more information.

#### 2.5.3.2 Block Delete Switch

If the block delete switch is on, lines of G-code which start with a slash (the block delete character) are not interpreted. If the switch is off, such lines are interpreted. Normally the block delete switch should be set before starting the NGC program.

#### 2.5.3.3 Optional Program Stop Switch

If this switch is on and an M1 code is encountered, program execution is paused.

### 2.5.4 Tool Table

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]
# tool table file
TOOL_TABLE = tooltable.tbl
```

The default filename probably looks something like the above, but you may prefer to give your machine its own tool table, using the same name as your INI file, but with a tbl extension:

```
TOOL_TABLE = acme_300.tbl
```

or:

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

For more information on the specifics of the tool table format, see the Tool Table Format section.

### 2.5.5 Параметры

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.

All parameters are available for use in G-code programs.

The format of a parameter file is shown in the following table. The file consists of any number of header lines, followed by one blank line, followed by any number of lines of data. The Interpreter skips over the header lines. It is important that there be exactly one blank line (with no spaces or tabs, even) before the data. The header line shown in the following table describes the data columns, so it is suggested (but not required) that that line always be included in the header.

The Interpreter reads only the first two columns of the table. The third column, *Comment*, is not read by the Interpreter.

Each line of the file contains the index number of a parameter in the first column and the value to which that parameter should be set in the second column. The value is represented as a double-precision floating point number inside the Interpreter, but a decimal point is not required in the file. All of the parameters shown in the following table are required parameters and must be included in any parameter file, except that any parameter representing a rotational axis value for an unused axis may be omitted. An error will be signaled if any required parameter is missing. A parameter file may include any other parameter, as long as its number is in the range 1 to 5400. The parameter numbers must be arranged in ascending order. An error will be signaled if not. Any parameter included in the file read by the Interpreter will be included in the file it writes as it exits. The original file is saved as a backup file when the new file is written. Comments are not preserved when the file is written.

Table 2.1: Parameter File Format

| Parameter Number | Parameter Value | Comment |
|---|---|---|
| 5161 | 0.0 | G28 Home X |
| 5162 | 0.0 | G28 Home Y |

See the Parameters section for more information.

## 2.6 Общая информация для пользователя токарного станка

В этой главе будет предоставлена информация, касающаяся токарных станков.

### 2.6.1 Режим токарного станка

Если ваш станок с ЧПУ — токарный, вам, вероятно, придется внести некоторые изменения в INI-файл, чтобы получить наилучшие результаты от LinuxCNC.

Если вы используете дисплей AXIS, заставьте AXIS правильно отображать ваши токарные инструменты Подробнее см. в разделе INI Configuration.

Настройка AXIS для режима токарной обработки.

```
[DISPLAY]

# Tell the AXIS GUI our machine is a lathe.
LATHE = TRUE
```

Lathe Mode в AXIS не устанавливает вашу плоскость по умолчанию на G18 (XZ). Вы должны запрограммировать это в преамбуле каждого файла G-кода или (лучше) добавить это в ваш INI-файл, например:

```
[RS274NGC]

# G-code modal codes (modes) that the interpreter is initialized with
# on startup
RS274NGC_STARTUP_CODE = G18 G20 G90
```

Если вы используете GMOCCAPY, тогда смотрите раздел the GMOCCAPY Lathe.

### 2.6.2 Таблица инструментов токарного станка

”Таблица инструментов” — это текстовый файл, содержащий информацию о каждом инструменте. Файл находится в том же каталоге, что и ваша конфигурация, и по умолчанию называется ”tool.tbl”. Инструменты могут находиться в устройстве смены инструмента или просто быть изменены вручную. Файл можно редактировать с помощью текстового редактора или обновлять с помощью G10 L1, L10, L11. В дисплее AXIS также есть встроенный редактор таблиц инструментов. Максимальное количество записей в таблице инструментов — 56. Максимальное количество инструментов и карманов — 99999.

В более ранних версиях LinuxCNC было два разных формата таблицы инструментов для фрезерных и токарных станков, но с версии 2.4.x для всех станков используется один формат таблицы инструментов. Просто игнорируйте те части таблицы инструментов, которые не относятся к вашему станку или которые вам не нужно использовать. Для получения дополнительной информации о специфике формата таблицы инструментов см. раздел Tool Table.

### 2.6.3 Ориентация токарного инструмента

На следующем рисунке показаны ориентации токарного инструмента с углом центральной линии каждой ориентации и информация о ПЕРЕДНЕМ УГЛЕ и ЗАДНЕМ УГЛЕ.

ПЕРЕДНИЙ УГОЛ и ЗАДНИЙ УГОЛ отсчитываются по часовой стрелке, начиная с линии, параллельно Z+.

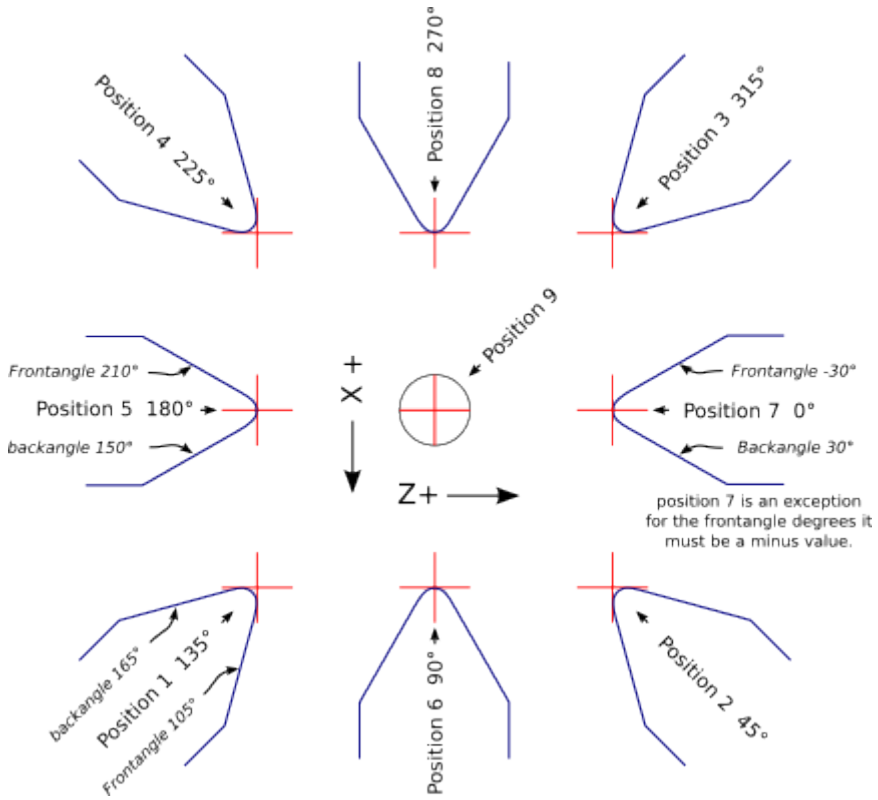
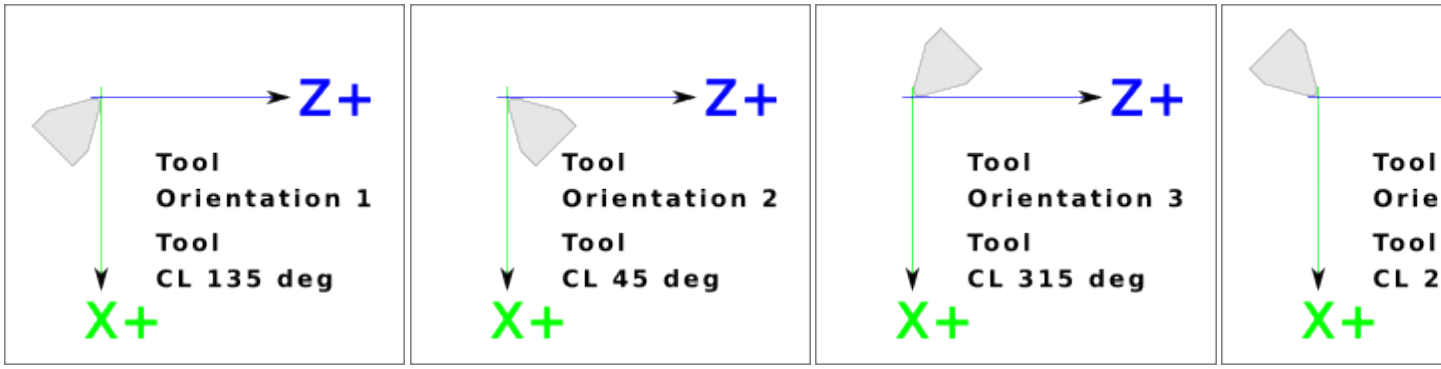


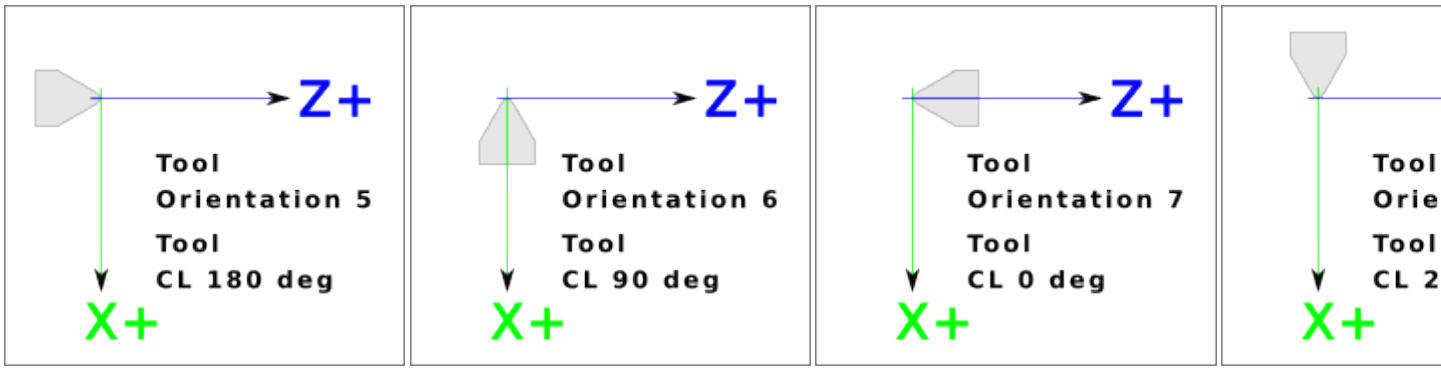

Figure 2.16: Ориентация токарного инструмента

На следующих рисунках в AXIS показано, как выглядят положения инструментов, введенные в таблицу инструментов.

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




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




## 2.6.4 Инструмент Touch Off

При работе в режиме токарного станка в AXIS вы можете задать X и Z в таблице инструментов с помощью окна Touch Off. Если у вас есть инструментальная револьверная головка, то при настройке револьверной головки обычно выбирается *Touch off to fixture* . При установке нулевой точки Z материала выбирается *Touch off to material*. Для получения дополнительной информации о G-кодах, используемых для инструментов, см. M6, Tn и G43. Для получения дополнительной информации о параметрах touch off инструмента в AXIS см. Tool Touch Off.

### 2.6.4.1 X Touch Off

Смещение оси X для каждого инструмента обычно представляет собой смещение от центральной линии шпинделя.

Один из методов — взять обычный токарный инструмент и обточить часть заготовки до известного диаметра. Используя окно Tool Touch Off, введите измеренный диаметр (или радиус, если вы находитесь в режиме радиуса) для этого инструмента. Затем, используя немного жидкости для разметки или маркера для покрытия детали, поднимите каждый инструмент так, чтобы он только коснулся красителя, и установите его смещение по оси X на диаметр используемой детали, используя tool touch off. Убедитесь, что все инструменты в угловых квадрантах имеют правильно установленный радиус носа в таблице инструментов, чтобы контрольная точка была правильной. Tool touch off автоматически добавляет G43, поэтому текущий инструмент является текущим смещением.

Типичная сессия может быть такой:

1. Верните каждую ось в исходное положение, если она не установлена.
2. Установите текущий инструмент с помощью *T*n *M6 G43*, где *n* — номер инструмента.
3. Выберите ось X в окне *Manual Control window*.
4. Переместите X в известное положение или сделайте пробный надрез и измерьте диаметр.
5. Выберите Touch Off и подберите Tool Table, затем введите положение или диаметр.
6. Выполните ту же последовательность для коррекции оси Z.

Примечание: если вы находитесь в Radius Mode, вам необходимо ввести радиус, а не диаметр.

#### 2.6.4.2 Z Touch Off

Смещения оси Z могут поначалу немного сбивать с толку, поскольку в смещении Z есть два элемента. Это смещение таблицы инструментов и смещение координат станка. Сначала мы рассмотрим смещения таблицы инструментов. Один из методов — использовать фиксированную точку на вашем токарном станке и задать смещение Z для всех инструментов от этой точки. Некоторые используют торец шпинделя или торец патрона. Это дает вам возможность перейти на новый инструмент и задать его смещение Z без необходимости перенастройки всех инструментов.

Типичная сессия может быть такой:

1. Верните каждую ось в исходное положение, если она не установлена.
2. Убедитесь, что для текущей системы координат не действуют смещения.
3. Установите текущий инструмент с помощью *T*n *M6 G43*, где *n* — номер инструмента.
4. Выберите ось Z в окне Manual Control.
5. Подведите инструмент близко к контрольной поверхности.
6. Используя цилиндр, отведите Z от контрольной поверхности до тех пор, пока цилиндр не пройдет между инструментом и контрольной поверхностью.
7. Выберите Touch Off, подберите Tool Table и установите положение на 0.0.
8. Повторите процедуру для каждого инструмента, используя тот же цилиндр.

Теперь все инструменты смещены на одинаковое расстояние от стандартного положения. Если вы меняете инструмент, например сверло, вы повторяете вышеописанное, и теперь он синхронизирова с остальными инструментами для смещения по оси Z. Некоторые инструменты могут потребовать немного шифровки для определения контрольной точки от точки касания. Например, если у вас есть отрезной инструмент шириной 0,125” и вы касаетесь левой стороны, но хотите, чтобы правая была Z0, введите 0,125” в окне касания.

#### 2.6.4.3 Z Смещение станка

После того, как смещение по оси Z для всех инструментов будет введено в таблицу инструментов, вы можете использовать любой инструмент для установки смещения станка с использованием системы координат станка.

Типичная сессия может быть такой:

1. Верните каждую ось в исходное положение, если она не установлена.
2. Установите текущий инструмент с помощью *T*n *M6*, где *n* — номер инструмента.
3. Введите команду G43, чтобы применить текущее смещение инструмента.
4. Подведите инструмент к заготовке и установите смещение станка по оси Z.

Если вы забыли установить G43 для текущего инструмента при установке смещения системы координат станка, вы не получите ожидаемого, поскольку смещение инструмента будет добавлено к текущему смещению при использовании инструмента в вашей программе.

### 2.6.5 Синхронизированное движение шпинделя

Для синхронизированного движения шпинделя требуется квадратурный энкодер, подключенный к шпинделю с одним индексным импульсом на оборот. Для получения дополнительной информации см. страницу руководства по движению и Пример управления шпинделем.

**Threading** Цикл резьбонарезания G76 используется как для внутренней, так и для внешней резьбы. Для получения дополнительной информации см. раздел G76.

**Постоянная скорость поверхности** CSS или Constant Surface Speed использует начало координат X станка, измененное смещением инструмента по оси X, для вычисления скорости шпинделя в об/мин. CSS будет отслеживать изменения в смещениях инструмента. X machine origin должен быть тогда, когда эталонный инструмент (с нулевым смещением) находится в центре вращения. Для получения дополнительной информации см. раздел G96.

**Подача за оборот** Подача за оборот переместит ось Z на величину F за оборот. Это не для резьбонарезания, используйте G76 для резьбонарезания. Для получения дополнительной информации см. раздел G95.

### 2.6.6 Дуги

Расчет дуг может быть достаточно сложным, даже если не учитывать режим радиуса и диаметра на токарных станках, а также ориентацию системы координат станка. Следующее относится к дугам формата центра. На токарном станке вы должны включить G18 в преамбулу, так как по умолчанию используется G17, даже если вы находитесь в режиме токарного станка, в пользовательско интерфейсе AXIS. Дуги в плоскости G18 XZ используют смещения I (ось X) и K (ось Z).

#### 2.6.6.1 Дуги и конструкция токарного станка

Типичный токарный станок имеет шпиндель слева от оператора и инструменты на стороне оператора от центральной линии шпинделя. Обычно это настраивается с воображаемой осью Y (+), направленной на пол.

Для этого типа настройки будет справедливо следующее:

- Ось Z (+) направлена вправо, в сторону от шпинделя.
- Ось X (+) направлена на оператора, и когда находится на стороне шпинделя со стороны оператора, значения X положительны.

У некоторых токарных станков с инструментами на задней стороне воображаемая ось Y (+) направлена вверх.

G2/G3 Направления дуг основаны на оси, вокруг которой они вращаются. В случае токарных станков это воображаемая ось Y. Если ось Y (+) указывает на пол, вам нужно посмотреть вверх, чтобы дуга выглядела идущей в правильном направлении. Поэтому, глядя сверху, вы инвертируете G2/G3, чтобы дуга выглядела идущей в правильном направлении.

#### 2.6.6.2 Режим радиуса и диаметра

При расчете дуг в радиусном режиме вам нужно помнить только направление вращения, применимое к вашему токарному станку.

При расчете дуг в режиме диаметра X — это диаметр, а смещение X (I) — это радиус, даже если вы находитесь в режиме диаметра G7.

## 2.6.7 Путь инструмента

### 2.6.7.1 Контрольная точка

Контрольная точка для инструмента следует запрограммированной траектории. Контрольная точка является пересечением линии, параллельной осям X и Z и касательной к диаметру режущей кромки инструмента, как определено при касании осей X и Z для этого инструмента. При повороте или торцевании прямолинейных деталей траектория резания и кромка инструмента следуют одной и той же траектории. При повороте радиуса и углов кромка режущей кромки инструмента не будет следовать запрограммированной траектории, если только не действует компенсация резца. На следующих рисунках вы можете увидеть, как контрольная точка не следует за кромкой инструмента, как вы могли бы предположить.

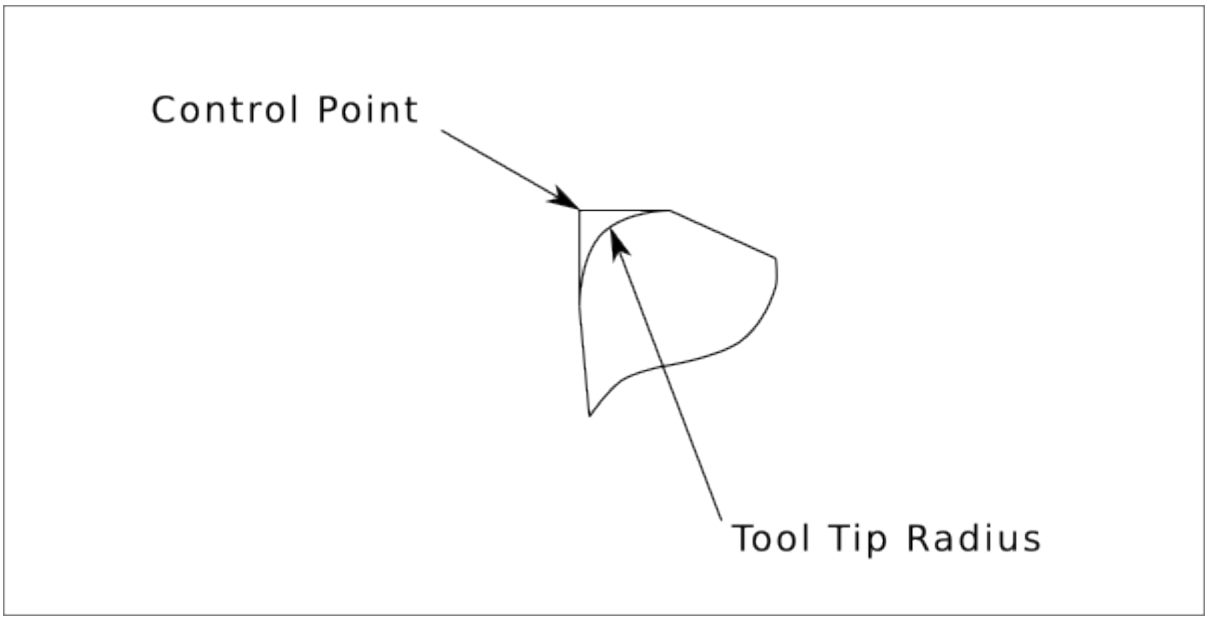



Figure 2.17: Контрольная точка

### 2.6.7.2 Углы резки без компенсации резца

Теперь представьте, что мы программируем рампу без компенсации резака. Запрограммированный путь показан на следующем рисунке. Как вы можете видеть на рисунке, запрограммированный путь и желаемый путь реза — это одно и то же, пока мы движемся только в направлении X или Z.

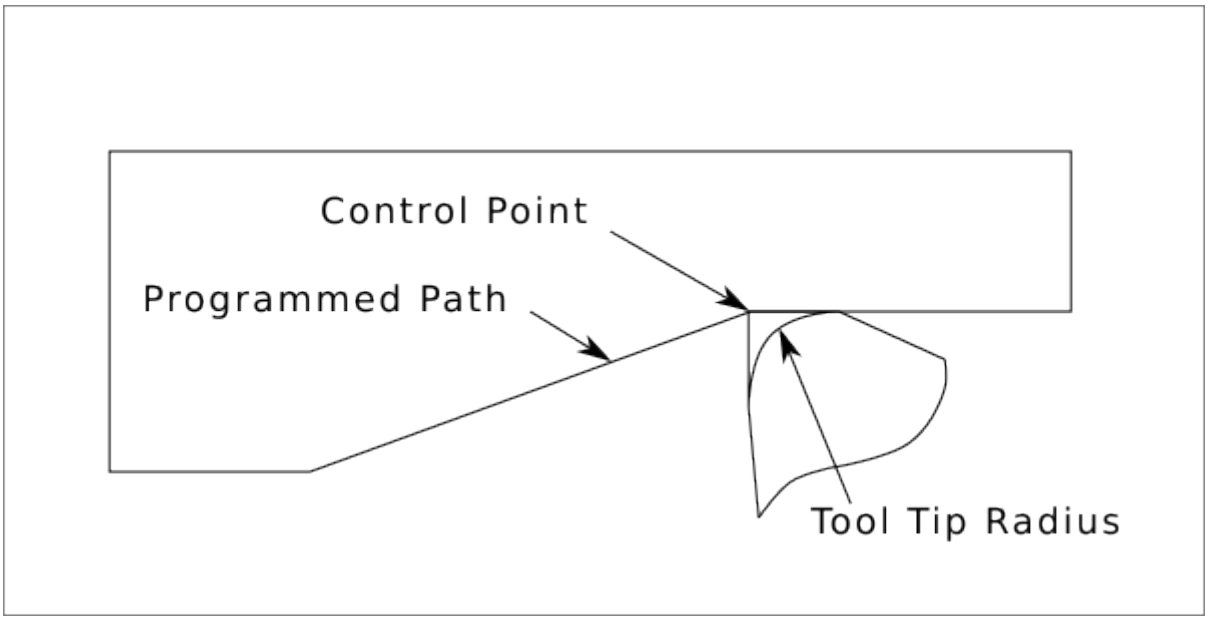



Figure 2.18: Наклонный заход

Теперь, когда контрольная точка продвигается по запрограммированному пути, фактическая режущая кромка не следует запрограммированному пути, как показано на следующем рисунке. Есть два способа решить эту проблему: компенсация резца и корректировка запрограммированного пути для компенсации радиуса резца.

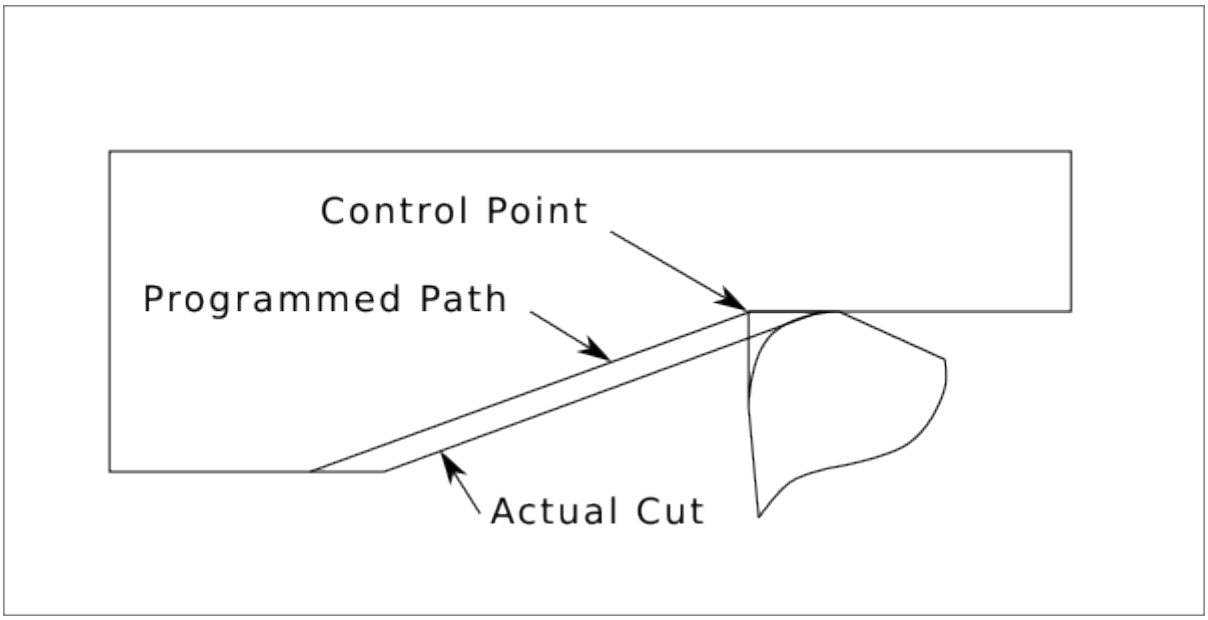



Figure 2.19: Наклонная траектория

В приведенном выше примере простым упражнением является корректировка запрограммированной траектории для получения желаемой фактической траектории путем перемещения запрограммирова траектории для наклона влево на радиус кончика резца.

#### 2.6.7.3 Радиусное резание

В этом примере мы рассмотрим, что происходит во время радиусной резки без компенсации резца. На следующем рисунке вы видите, как резец проходит внешний диаметр детали. Контрольная точка инструмента следует запрограммированной траектории, а инструмент касается внешнего диаметра детали.

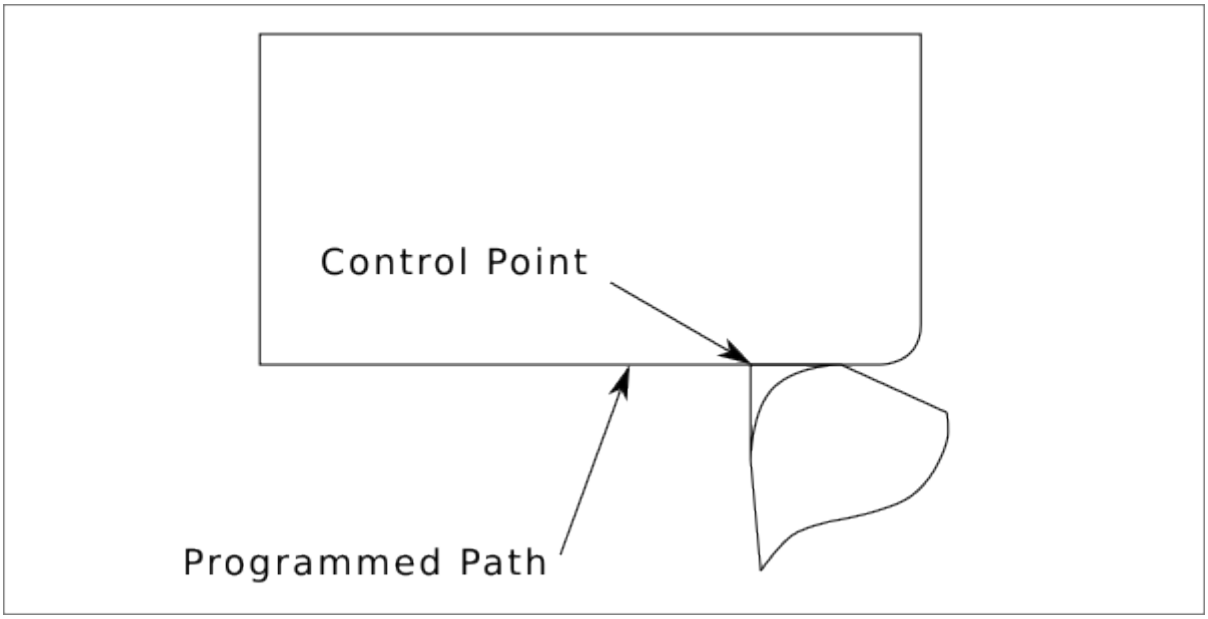



Figure 2.20: Токарная резка

На следующем рисунке вы можете видеть, что по мере приближения инструмента к концу детали контрольная точка все еще следует по траектории, но кончик резца покинул деталь и режет воздух. Вы также можете видеть, что, хотя радиус был запрограммирован, деталь фактически получит квадратный угол.

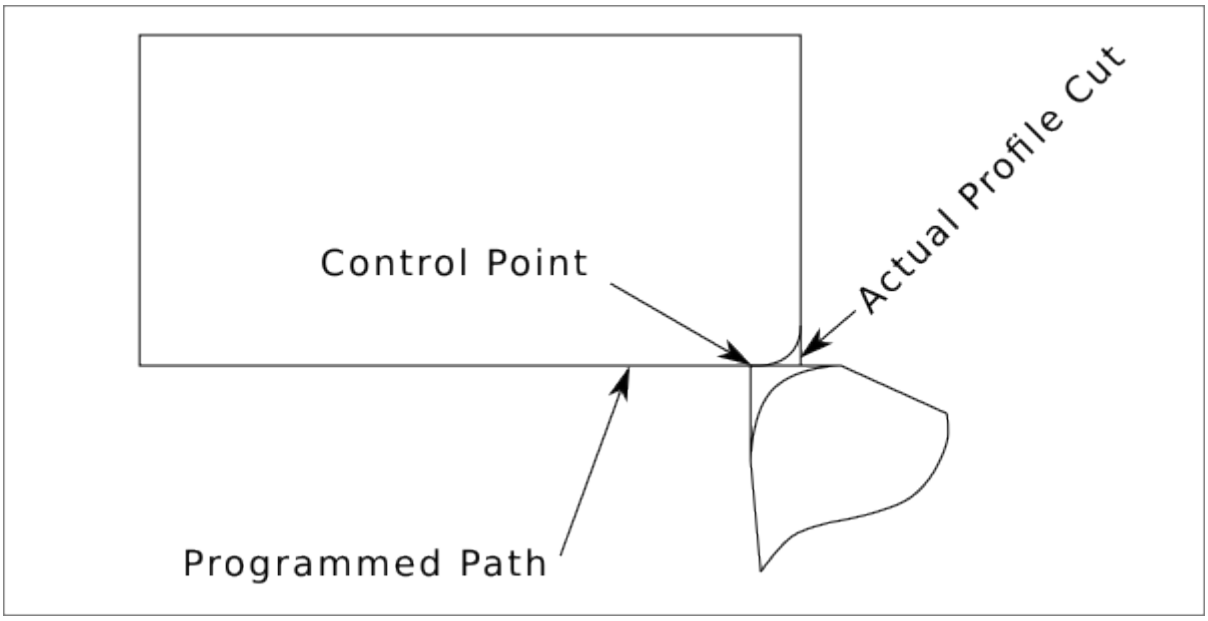



Figure 2.21: Радиусный рез

Теперь вы можете видеть, как контрольная точка следует запрограммированному радиусу, а кончик инструмента покинул деталь и теперь режет воздух.

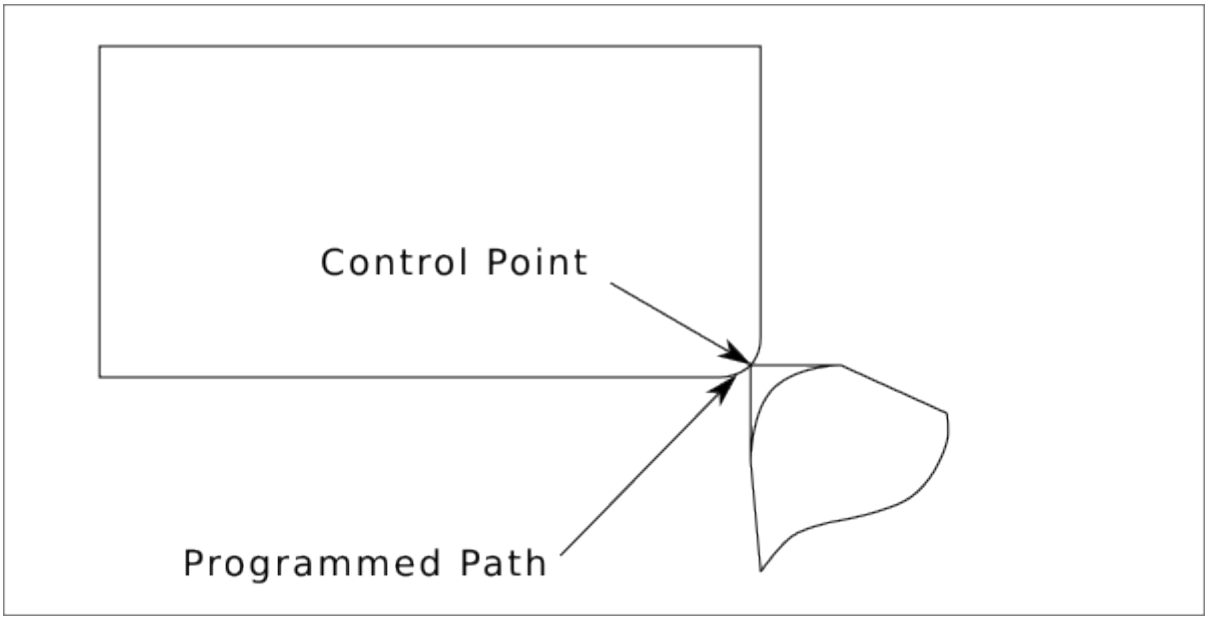



Figure 2.22: Радиусный рез

На последнем рисунке мы видим, что кончик инструмента закончит резку грани, но оставит квадратный угол вместо хорошего радиуса. Обратите внимание также, что если вы запрограммируете резку так, чтобы она заканчивалась в центре детали, небольшое количество материала останется от радиуса инструмента. Чтобы закончить резку грани до центра детали, вам нужно запрограммирова инструмент так, чтобы он прошел мимо центра, по крайней мере, на радиус носика инструмента.

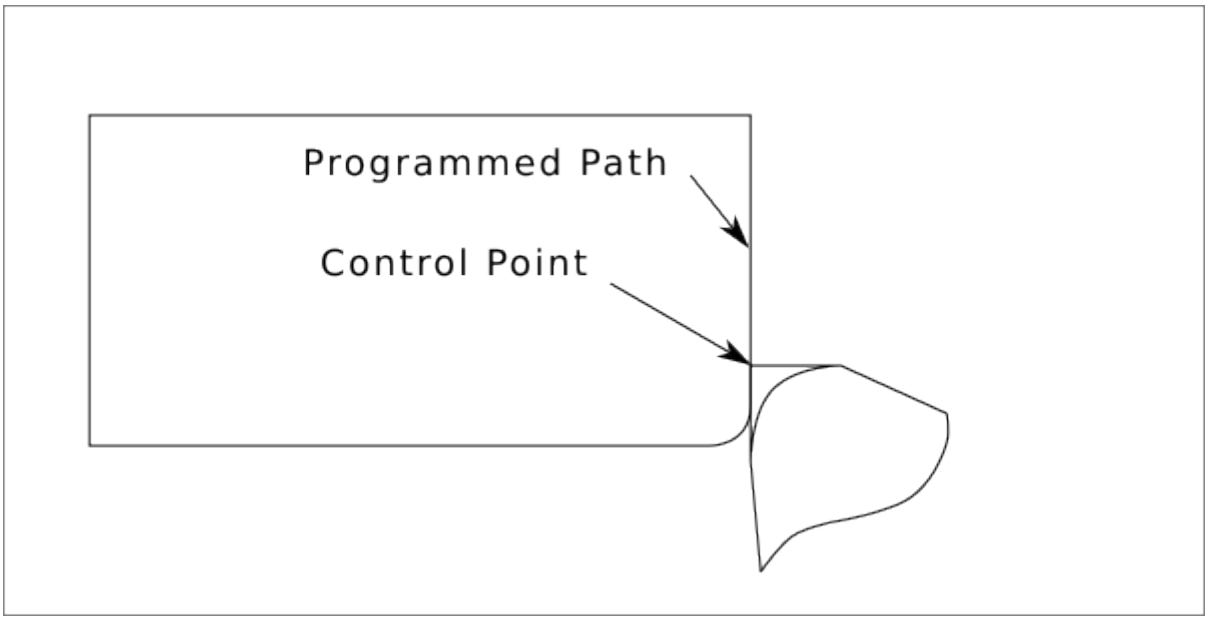



Figure 2.23: Торцевой рез

#### 2.6.7.4 Использование компенсации резца

- При использовании компенсации резца на токарном станке представьте радиус режущей кромки инструмента как радиус круглой фрезы.
- Когда используется компенсация резца траектория должна быть достаточно большой для круглого инструмента, который не будет врезаться в следующую линию.
- При резке прямых линий на токарном станке вы можете не захотеть использовать компенсацию резца. Например, при расточке отверстия с помощью плотно прилегающей расточной оправки у вас может не хватить места для выполнения выходного движения.
- Входное движение в дугу компенсации резца важно для получения правильных результатов.

## 2.7 Plasma Cutting Primer for LinuxCNC Users

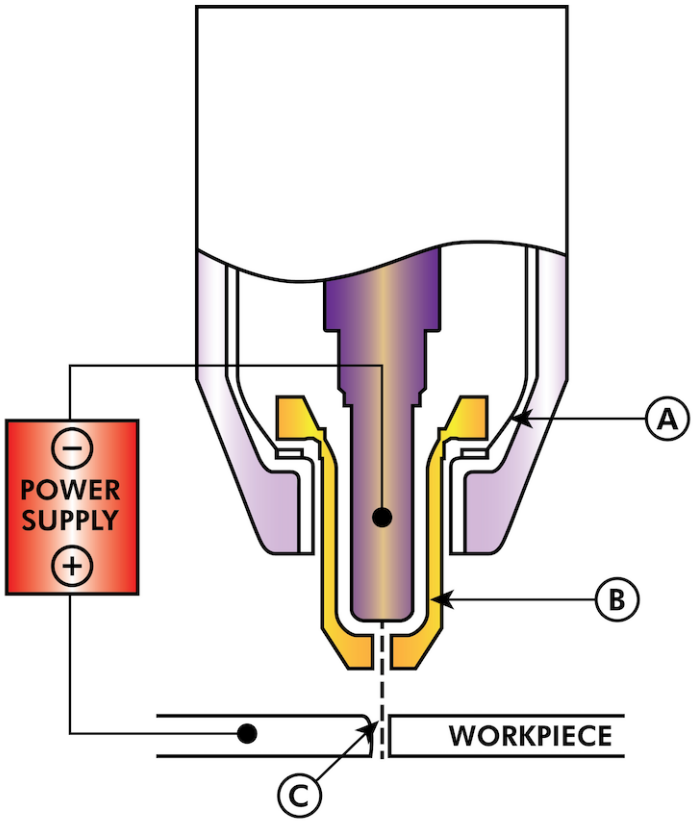
### 2.7.1 What Is Plasma?

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 Arc Initialisation

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 High Frequency Start

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.

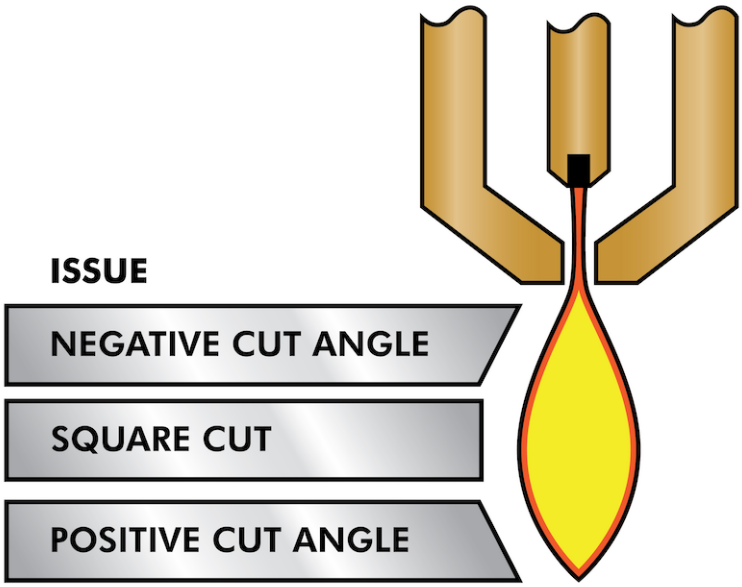
For entry level air plasma CNC systems, the blowback style is much preferred to minimise electrical interference with electronics and standard PCs, but the High frequency start still rules supreme in larger machines from 200 A and up. These require industrial level PCs and electronics, and even commercial 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.

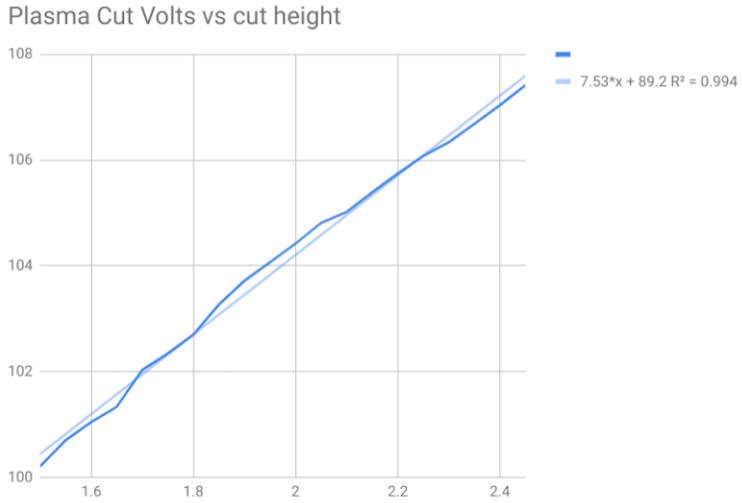
- **Torch to work distance can impact edge bevel**




- **Negative cut angle:** torch too low, increase torch to work distance.
- **Positive cut angle:** torch too high, decrease torch to work distance.

**Note**
A slight variation in cut angles may be normal, as long as it is within tolerance.

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 Choosing a Plasma Machine for CNC operations

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.

Whilst rules are made to be broken if you fully understand the reasons the rule apply, we consider a new plasma table builder should select a machine with the following features:

- Blowback start to minimise electrical noise to simplify construction
- A Machine torch is preferred but many have used hand torches.
- A fully shielded torch tip to allow ohmic sensing

If you have the budget, a higher end machines will supply:

- Manufacturer provided cut charts which will save many hours and material waste calibrating cut parameters

- 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.
- Higher duty cycles

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 Types Of Torch Height Control

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 Arc 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 Initial Height Sensing

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 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.

Regardless of the probing method used, it is strongly recommended that float switch is implemented so that there is a fallback or secondary signal to avoid damage to the torch from a crash.

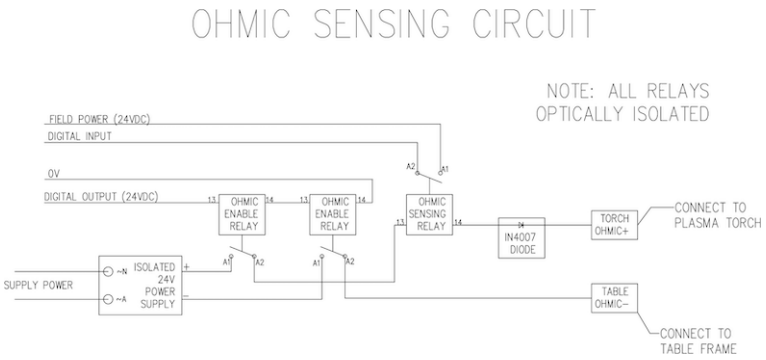
### 2.7.7.2 Ohmic Sensing

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. Ohmic sensing only be implemented where the torch has a shield that is isolated from the torch tip that conveys the cutting arc.

2. The ohmic circuit uses a totally separate isolated power supply that activates an opto-isolated relay to enable the probing signal to be transmitted to the CNC controller.
3. The positive side of the circuit should be at the torch
4. Both sides of the circuit needs to be isolated by opto-isolated relays until probing is being undertaken
5. Blocking diodes be used to prevent arc voltage entering the ohmic sensing circuit.

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

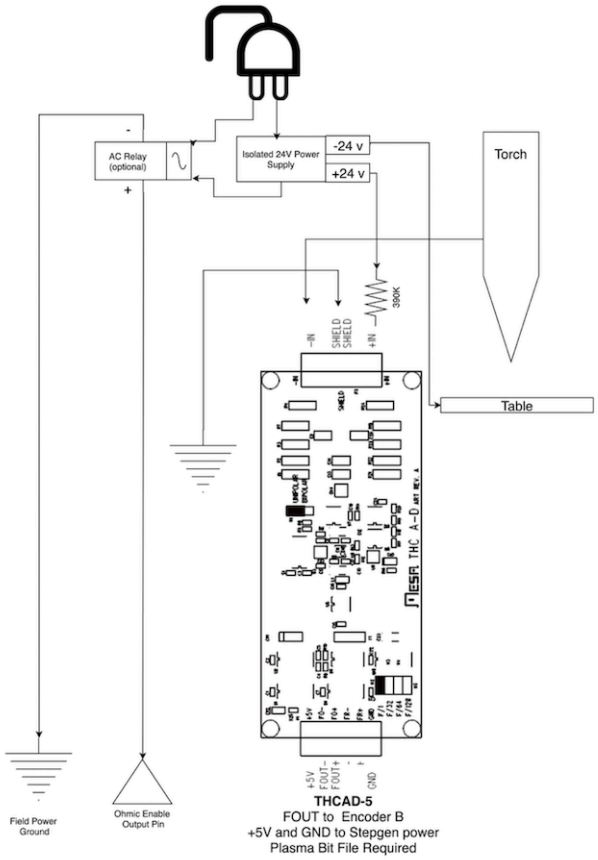
A more sophisticated method of material sensing that eliminates the relays and diodes is to use another THCAD-5 to monitor the material sensing circuit voltage from an isolated power supply. The advantage this has is the THCAD is designed for the hostile plasma electrical environment and totally and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

To implement this method, a second encoder input is required.

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.

The following circuit diagram shows how to implement a hypersensing circuit.




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 Example HAL Code for 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

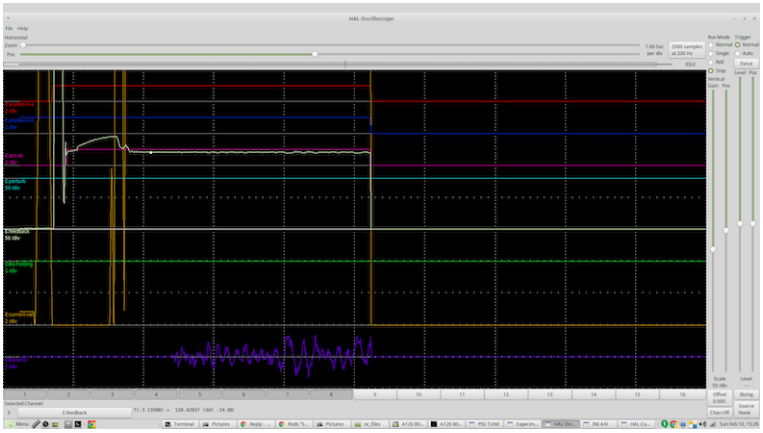
# --- Configure the component ---
setp ohmicsense.thcad-0-volt-freq    140200
setp ohmicsense.thcad-max-volt-freq  988300
setp ohmicsense.thcad-divide         32
setp ohmicsense.thcad-fullscale      5
```

```
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 db_ohmic.in
net ohmic-true ohmicsense.ohmic-on => db_ohmic.in
net plasmac:ohmic-enable    =>  ohmicsense.is-probing
```

### 2.7.8 THC Delay

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.

---

**Note**
If the cut velocity is not near the desired cut speed at the end of this delay, the controller should wait until this is achieved before enabling the THC.

---

### 2.7.9 Torch Voltage Sampling

Rather than relying on the manufacturer's cut charts to set the desired torch voltage, many people (the writer included) prefer to sample the voltage as the THC is enabled and use that as a set point.

### 2.7.10 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.

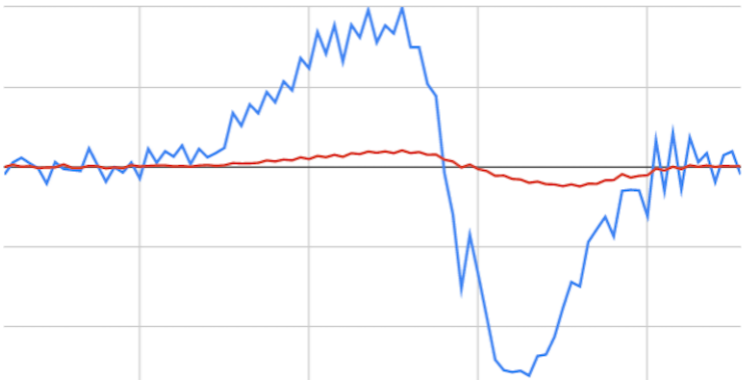
### 2.7.11 Corner Lock / Velocity Anti-Dive

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 Torch On (output)

- 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.
- Connect the other side of the coil to Field Power ground.
- 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.
- In some circumstances, the onboard spindle relay on a Mesa card can be used instead of an external relay.
- Usually connected to `spindle.0.on`.

**Warning**
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 Float switch (input)

- Used for surface probing. A sensor or switch that is activated if the torch slides up when it hits the material.
- 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 Ohmic Sensor enable (output)

- See the ohmic sensing schematic.
- Connect output pin to one side of the isolation relays and the other side to field power ground.
- 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 Ohmic Sensing (input)

- Take care to follow the ohmic sensing schematic shown previously.
- An isolated power supply triggers a relay when the torch shield contacts the material.
- Connect field power to one output terminal and the other to the input.
- Take care to observe relay polarity if opto-coupled solid State relays are used.
- 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 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
```

---

**Note**
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 Enable/Disable THC Operation:

```
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 would set the velocity to 100% of requested~speed
M67 E3 Q40 would set the velocity to 40% of requested~speed
M67 E3 Q60 would set the velocity to 60% of requested~speed
M67 E3 Q100 would set the velocity to 100% of requested~speed
```

**Cutter Compensation:**

```
G41.1 D#<_hal[plasmac_run.kerf-width-f]> ; for left of programmed path
G42.1 D#<_hal[plasmac_run.kerf-width-f]> for right of programmed path
G40 to turn compensation off
```

---

**Note**
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.

# Chapter 3

# Мастера настройки

## 3.1 Мастер настройки шагового двигателя

### 3.1.1 Введение

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

StepConf is a program that generates configuration files for LinuxCNC for a specific class of CNC machine: those that are controlled via a *standard parallel port*, and controlled by signals of type *step & direction*.

StepConf is installed when you install LinuxCNC and is in the CNC menu.

StepConf places a file in the linuxcnc/config directory to store the choices for each configuration you create. When you change something, you need to pick the file that matches your configuration name. The file extension is .stepconf.

The StepConf Wizard works best with at least 800 x 600 screen resolution.

### 3.1.2 Start Page

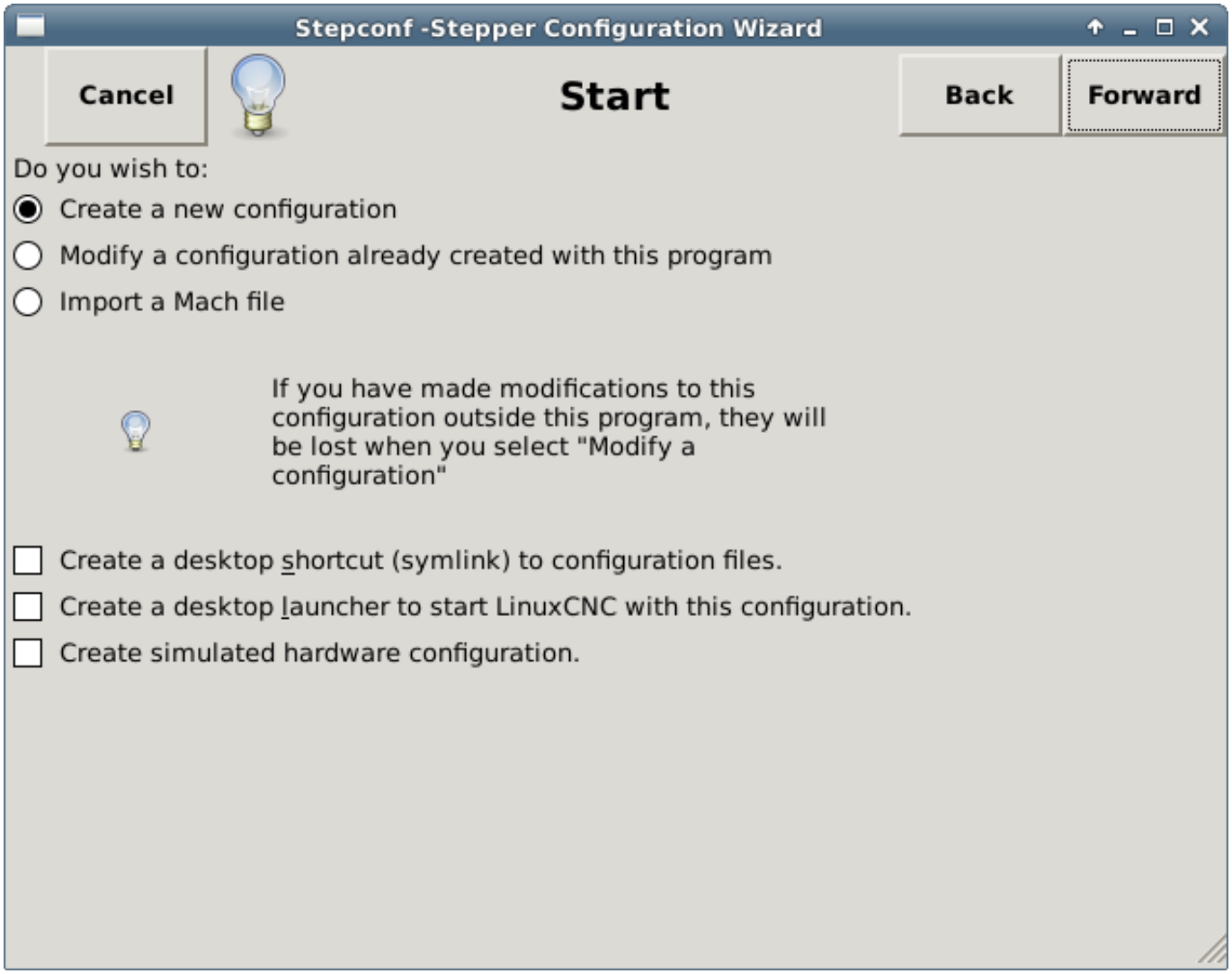



Figure 3.1: StepConf Entry Page

The three first radio buttons are self-explanatory:

- *Create New* - Creates a fresh configuration.
- *Modify* - Modify an existing configuration. After selecting this a file picker pops up so you can select the .stepconf file for modification. If you made any modifications to the main HAL or the INI file these will be lost. Modifications to custom.hal and custom_postgui.hal will not be changed by the StepConf Wizard. StepConf will highlight the lastconf that was built.
- *Import* - Import a Mach configuration file and attempt to convert it to a LinuxCNC config file. After the import, you will go though the pages of StepConf to confirm/modify the entries. The original mach XML file will not be changed.

These next options will be recorded in a preference file for the next run of StepConf.

- *Create Desktop Shortcut* - This will place a link on your desktop to the files.

- *Create Desktop Launcher* - This will place a launcher on your desktop to start your application.
- *Create Simulated Hardware* - This allows you to build a config for testing, even if you don't have the actual hardware.

### 3.1.3 Basic Information

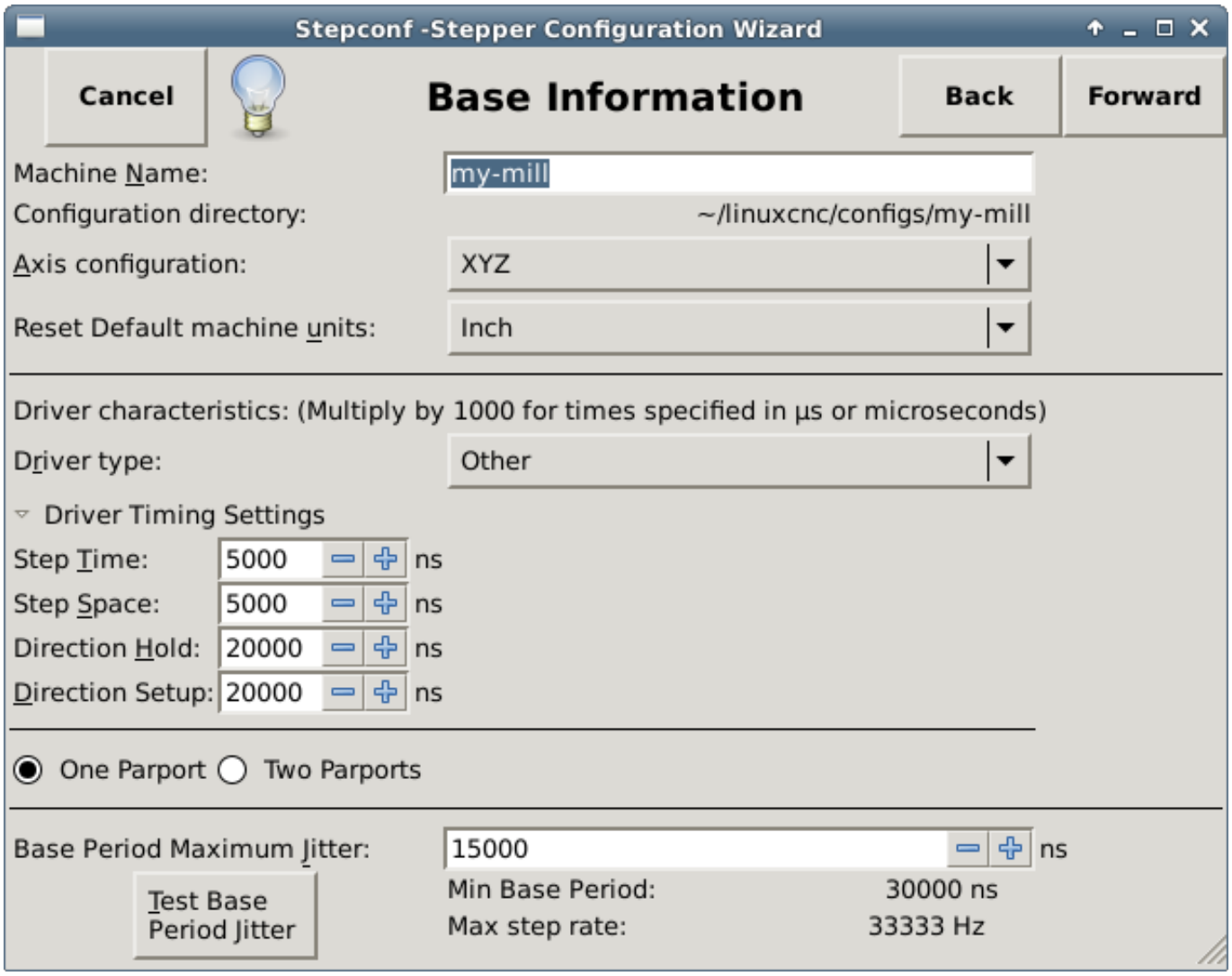



Figure 3.2: Basic Information Page

- *Create Simulated Hardware* - This allows you to build a config for testing, even if you don't have the actual hardware.
- *Machine Name* - Choose a name for your machine. Use only uppercase letters, lowercase letters, digits, - and _.
- *Axis Configuration* - Choose XYZ (Mill), XYZA (4-axis mill) or XZ (Lathe).
- *Machine Units* - Choose Inch or mm. All subsequent entries will be in the chosen units. Changing this also changes the default values in the Axes section. If you change this after selecting values in any of the axes sections, they will be over-written by the default values of the selected units.

- *Driver Type* - If you have one of the stepper drivers listed in the pull down box, choose it. Otherwise, select *Other* and find the timing values in your driver's data sheet and enter them as *nano seconds* in the *Driver Timing Settings*. If the data sheet gives a value in microseconds, multiply by 1000. For example, enter 4.5 µs as 4500 ns.

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

  Additional signal conditioning or isolation such as optocouplers and RC filters on break out boards can impose timing constraints of their own, in addition to those of the driver. You may find it necessary to add some time to the drive requirements to allow for this.

  The LinuxCNC Configuration Selector has configs for Sherline already configured. * *Step Time* - How long the step pulse is *on* in nano seconds. If your not sure about this setting a value of 20,000 will work with most drives. * *Step Space* - Minimum time between step pulses in nano seconds. If your not sure about this setting a value of 20,000 will work with most drives. * *Direction Hold* - How long the direction pin is held after a change of direction in nanoseconds. If your not sure about this setting a value of 20,000 will work with most drives. * *Direction Setup* - How long before a direction change after the last step pulse in nanoseconds. If your not sure about this setting a value of 20,000 will work with most drives. * *One / Two Parport* - Select how many parallel port are to be configured. * *Base Period Maximum Jitter* - Enter the result of the Latency Test here. To run a latency test press the *Test Base Period Jitter* button. See the Latency Test section for more details.

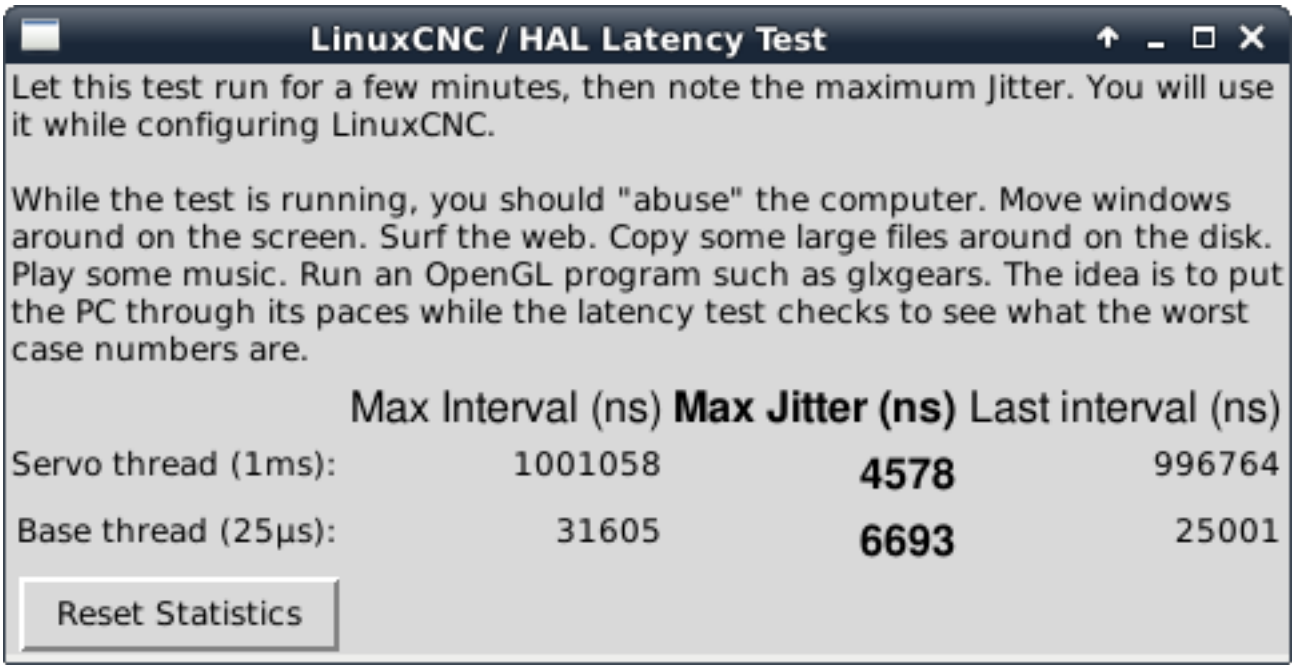



Figure 3.3: Latency Test

- *Max Step Rate* -StepConf automatically calculates the Max Step Rate based on the driver characteristics entered and the latency test result.
- *Min Base Period* - StepConf automatically determines the Min Base Period based on the driver characteristics entered and latency test result.

The important number from the result of the Latency Test is the *max jitter*. In the example above, 9075 nanoseconds (ns), or 9.075 microseconds (µs), is the highest jitter. Enter the max jitter it in the Base Period Maximum Jitter box.

### 3.1.4 Parallel Port Setup

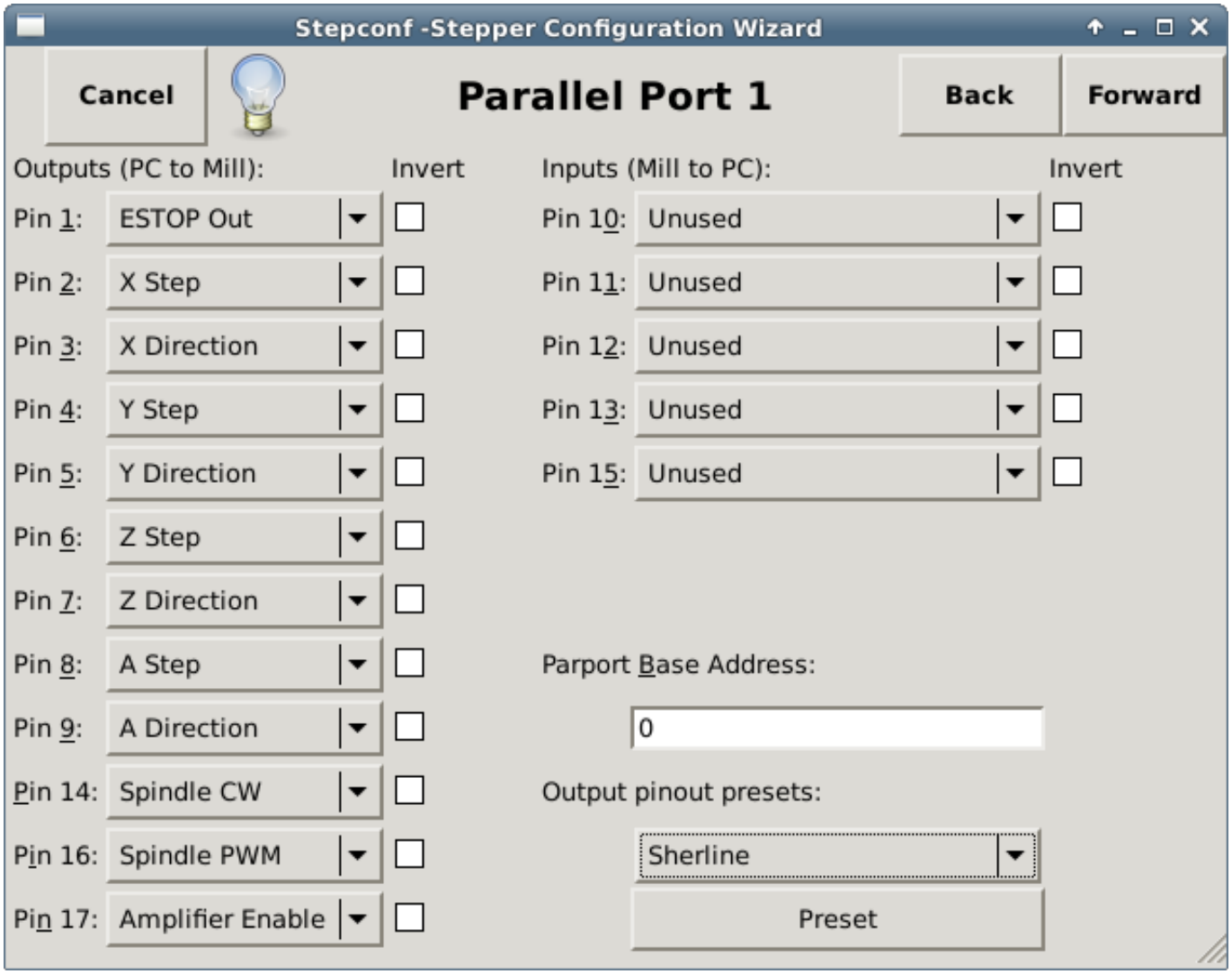



Figure 3.4: Parallel Port Setup Page

You may specify the address as a hexadecimal (often 0x378) or as linux's default port number (probably 0)

For each pin, choose the signal which matches your parallel port pinout. Turn on the *invert* check box if the signal is inverted (0V for true/active, 5V for false/inactive).

- *Output pinout presets* - Automatically set pins 2 through 9 according to the Sherline standard (Direction on pins 2, 4, 6, 8) or the Xylotex standard (Direction on pins 3, 5, 7, 9).
- *Inputs and Outputs* - If the input or output is not used set the option to *Unused*.
- *External E-Stop* - This can be selected from an input pin drop down box. A typical E-Stop chain uses all normally closed contacts.
- *Homing & Limit Switches* - These can be selected from an input pin drop down box for most configurations.

- *Charge Pump* - If your driver board requires a charge pump signal select Charge Pump from the drop down list for the output pin you wish to connect to your charge pump input. The charge pump output is connected to the base thread by StepConf. The charge pump output will be about 1/2 of the maximum step rate shown on the Basic Machine Configuration page.
- *Plasma Arc Voltage* - If you require a Mesa THCAD to input a plasma arc voltage then select Plasma Arc Voltage from the list of output pins. This will enable a THCAD page during the setup procedure for the entry of the card parameters.

### 3.1.5 Parallel Port 2 Setup

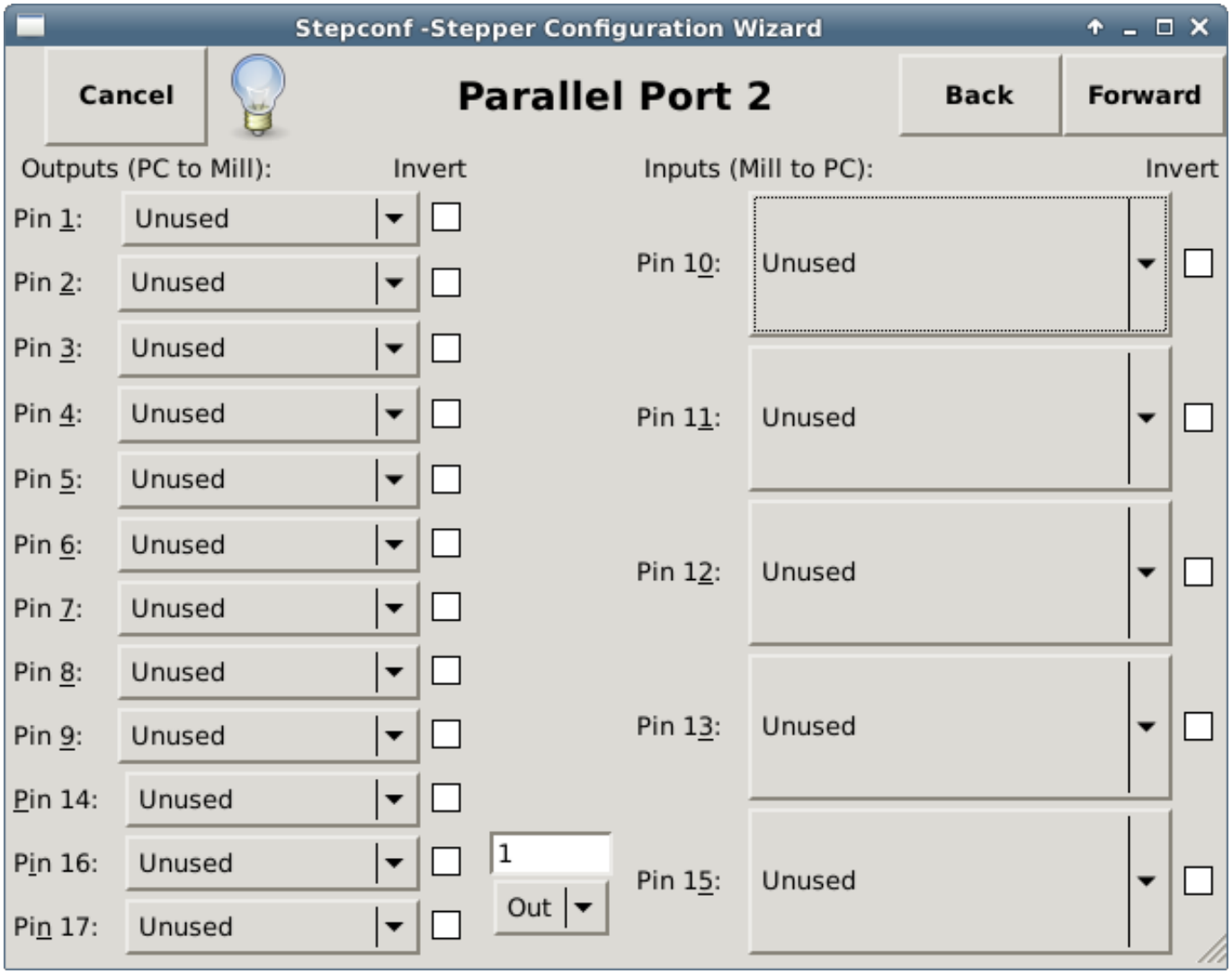



Figure 3.5: Parallel Port 2 Setup Page

The second Parallel port (if selected) can be configured and It's pins assigned on this page. No step and direction signals can be selected. You may select in or out to maximizes the number of input/output pins that are available. You may specify the address as a hexadecimal (often 0x378) or as linux's default port number (probably 1).

### 3.1.6 Axis Configuration

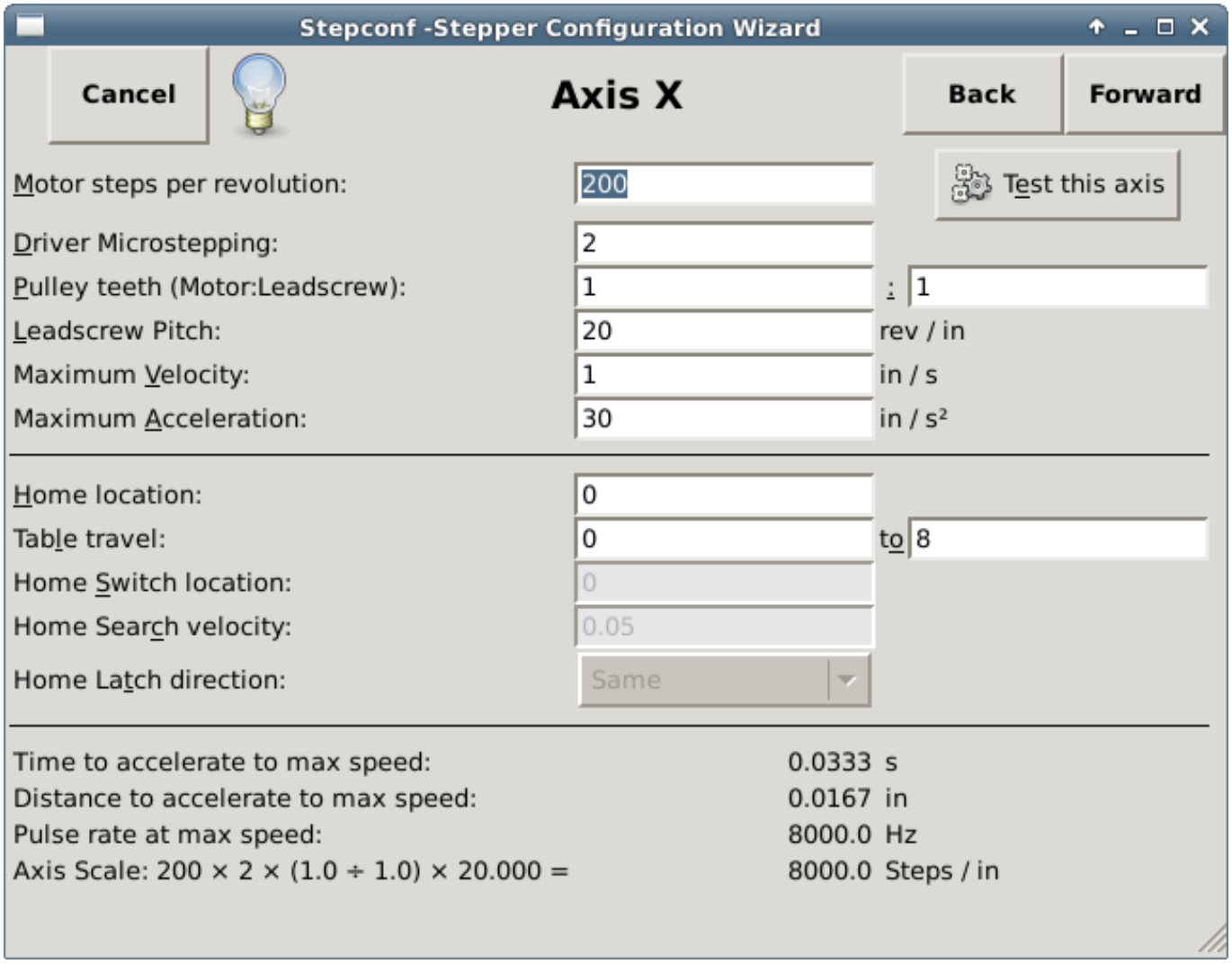



Figure 3.6: Axis Configuration Screen

- *Motor Steps Per Revolution* - The number of full steps per motor revolution. If you know how many degrees per step the motor is (e.g., 1.8 degree), then divide 360 by the degrees per step to find the number of steps per motor revolution.
- *Driver Microstepping* - The amount of microstepping performed by the driver. Enter *2* for half-stepping.
- *Pulley Ratio* - If your machine has pulleys between the motor and leadscrew, enter the ratio here. If not, enter *1:1*.
- *Leadscrew Pitch* - Enter the pitch of the leadscrew here. If you chose *Inch* units, enter the number of threads per inch. If you chose *mm* units, enter the number of millimeters per revolution (e.g., enter 2 for 2mm/rev). If the machine travels in the wrong direction, enter a negative number here instead of a positive number, or invert the direction pin for the axis.
- *Максимальная скорость* — введите максимальную скорость для оси в единицах в секунду.

- *Maximum Acceleration* - The correct values for these items can only be determined through experimentation. See Finding Maximum Velocity to set the speed and Finding Maximum Acceleration to set the acceleration.
- *Home Location* - The position the machine moves to after completing the homing procedure for this axis. For machines without home switches, this is the location the operator manually moves the machine to before pressing the Home button. If you combine the home and limit switches you must move off of the switch to the home position or you will get a joint limit error.
- *Table Travel* - The range of travel for that axis based on the machine origin. The home location must be inside the *Table Travel* and not equal to one of the Table Travel values.
- *Home Switch Location* - The location at which the home switch trips or releases relative to the machine origin. This item and the two below only appear when Home Switches were chosen in the Parallel Port Pinout. If you combine home and limit switches the home switch location can not be the same as the home position or you will get a joint limit error.
- *Home Search Velocity* - The velocity to use when searching for the home switch. If the switch is near the end of travel, this velocity must be chosen so that the axis can decelerate to a stop before hitting the end of travel. If the switch is only closed for a short range of travel (instead of being closed from its trip point to one end of travel), this velocity must be chosen so that the axis can decelerate to a stop before the switch opens again, and homing must always be started from the same side of the switch. If the machine moves the wrong direction at the beginning of the homing procedure, negate the value of *Home Search Velocity*.
- *Home Latch Direction* - Choose *Same* to have the axis back off the switch, then approach it again at a very low speed. The second time the switch closes, the home position is set. Choose *Opposite* to have the axis back off the switch and when the switch opens, the home position is set.
- *Time to accelerate to max speed* - Time to reach maximum speed calculated from *Max Acceleration* and *Max Velocity*.
- *Distance to accelerate to max speed* - Distance to reach maximum speed from a standstill.
- *Pulse rate at max speed* - Information computed based on the values entered above. The greatest *Pulse rate at max speed* determines the *BASE_PERIOD*. Values above 20000Hz may lead to slow response time or even lockups (the fastest usable pulse rate varies from computer to computer)
- *Axis SCALE* - The number that will be used in the INI file [SCALE] setting. This is how many steps per user unit.
- *Test this axis* - This will open a window to allow testing for each axis. This can be used after filling out all the information for this axis.

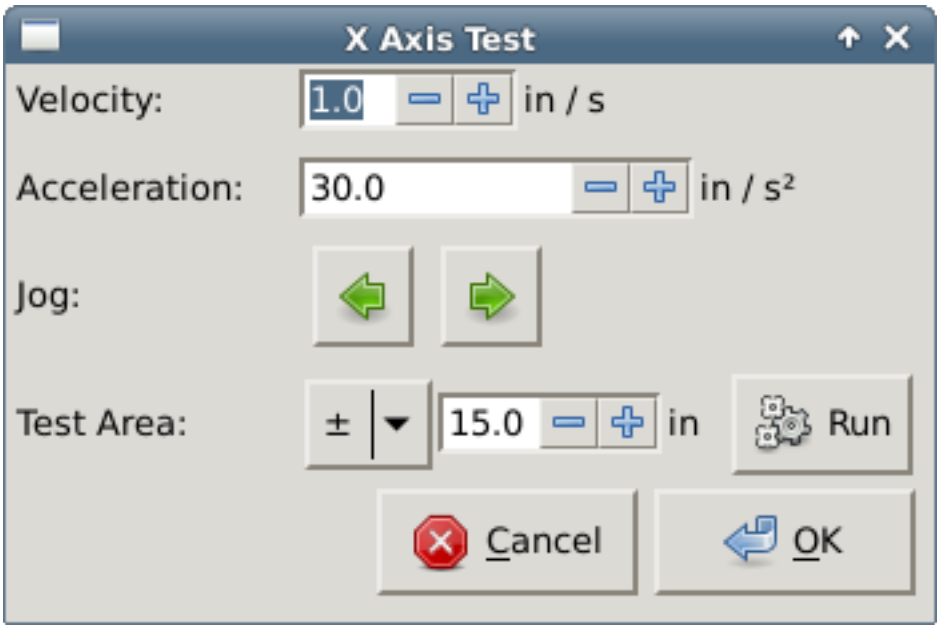



Figure 3.7: Axis Test

Test this axis is a basic tester that only outputs step and direction signals to try different values for acceleration and velocity.

**Important**
In order to use test this axis you have to manually enable the axis if this is required. If your driver has a charge pump you will have to bypass it. Test this axis does not react to limit switch inputs. Use with caution.

#### 3.1.6.1 Нахождение максимальной скорости

Begin with a low Acceleration (for example, **2 inches/s²** or **50 mm/s²**) and the velocity you hope to attain. Using the buttons provided, jog the axis to near the center of travel. Take care because with a low acceleration value, it can take a surprising distance for the axis to decelerate to a stop.

After gauging the amount of travel available, enter a safe distance in Test Area, keeping in mind that after a stall the motor may next start to move in an unexpected direction. Then click Run. The machine will begin to move back and forth along this axis. In this test, it is important that the combination of Acceleration and Test Area allow the machine to reach the selected Velocity and *cruise* for at least a short distance — the more distance, the better this test is. The formula **d = 0.5 * v * v/a** gives the minimum distance required to reach the specified velocity with the given acceleration. If it is convenient and safe to do so, push the table against the direction of motion to simulate cutting forces. If the machine stalls, reduce the speed and start the test again.

If the machine did not obviously stall, click the *Run* button off. The axis now returns to the position where it started. If the position is incorrect, then the axis stalled or lost steps during the test. Reduce Velocity and start the test again.

If the machine doesn't move, stalls, or loses steps, no matter how low you turn Velocity, verify the following:

- Correct step waveform timings

- Correct pinout, including *Invert* on step pins
- Correct, well-shielded cabling
- Physical problems with the motor, motor coupling, leadscrew, etc.

Once you have found a speed 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 Velocity*.

#### 3.1.6.2 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.

### 3.1.7 Spindle Configuration

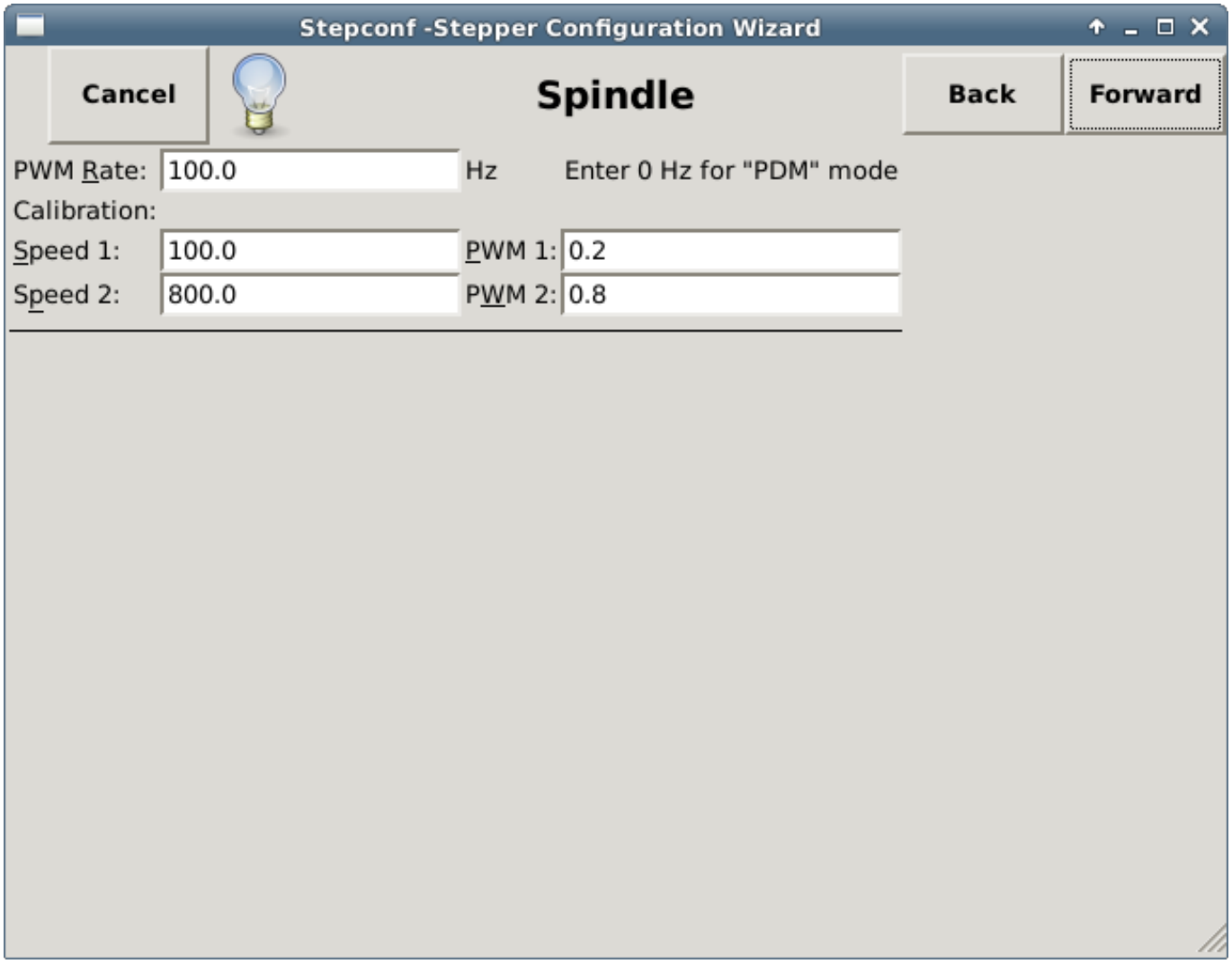



Figure 3.8: Spindle Configuration Page

This page only appears when *Spindle PWM* is chosen in the *Parallel Port Pinout* page for one of the outputs.

#### 3.1.7.1 Контроль скорости шпинделя

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

- *PWM Rate* - The *carrier frequency* of the PWM signal to the spindle. Enter *0* for PDM mode, which is useful for generating an analog control voltage. Refer to the documentation for your spindle controller for the appropriate value.
- *Speed 1 and 2, PWM 1 and 2* - The generated configuration file uses a simple linear relationship to determine the PWM value for a given RPM value. If the values are not known, they can be determined. For more information see Determining Spindle Calibration.

#### 3.1.7.2 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* - Is a pulse that occurs once per revolution of the spindle.
- *Spindle Phase A* - This is a pulse that occurs in multiple equally-spaced locations as the spindle turns.
- *Spindle Phase B (optional)* - This is a second pulse that occurs, but with an offset from Spindle Phase A. The advantages to using both A and B are direction sensing, increased noise immunity, and increased resolution.

If *Spindle Phase A* and *Spindle Index* appear on the pinout, the following information should be entered:

- *Use Spindle-At-Speed* - With encoder feedback one can choose to have LinuxCNC wait for the spindle to reach the commanded speed before feed moves. Select this option and set the *close enough* scale.
- *Speed Display Filter Gain* - Setting for adjusting the stability of the visual spindle speed display.
- *Cycles per revolution* - The number of cycles of the *Spindle A* signal during one revolution of the spindle. This option is only enabled when an input has been set to *Spindle Phase A*
- *Maximum speed in thread* - The maximum spindle speed used in threading. For a high spindle RPM or a spindle encoder with high resolution, a low value of *BASE_PERIOD* is required.

#### 3.1.7.3 Determining Spindle Calibration

Enter the following values in the Spindle Configuration page:

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

Finish the remaining steps of the configuration process, then launch LinuxCNC with your configuration. Turn the machine on and select the MDI tab. Start the spindle turning by entering: *M3 S100*. Change the spindle speed by entering a different S-number: *S800*. Valid numbers (at this point) range from 1 to 1000.

For two different S-numbers, measure the actual spindle speed in RPM. Record the S-numbers and actual spindle speeds. Run StepConf again. For *Speed* enter the measured speed, and for *PWM* enter the S-number divided by 1000.

Because most spindle drivers are somewhat nonlinear in their response curves, it is best to:

- Make sure the two calibration speeds are not too close together in RPM.
- Make sure the two calibration speeds are in the range of speeds you will typically use while milling.

For instance, if your spindle will go from 0 RPM to 8000 RPM, but you generally use speeds from 400 RPM (10%) to 4000 RPM (100%), then find the PWM values that give 1600 RPM (40%) and 2800 RPM (70%).

### 3.1.8 Варианты

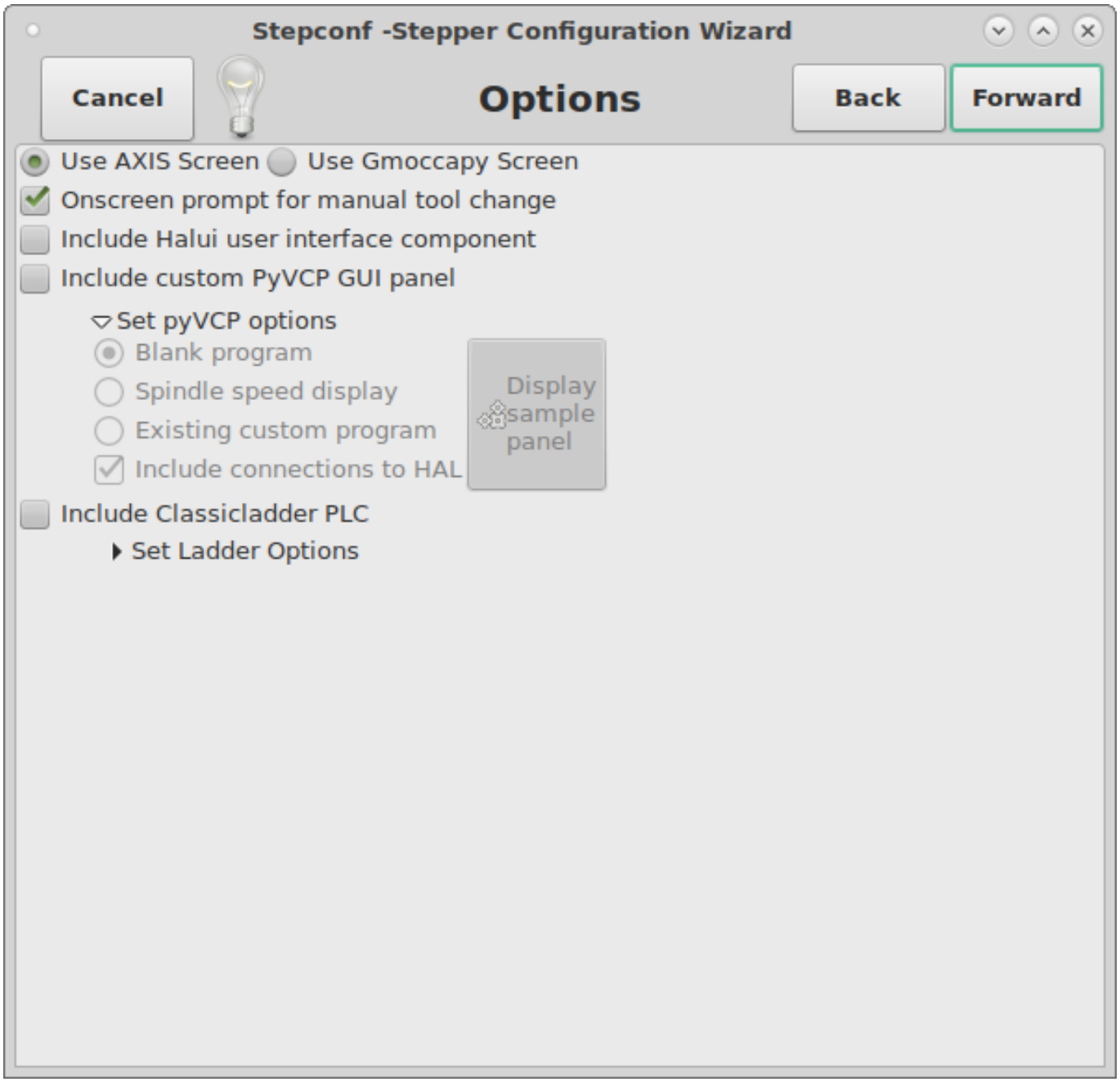



Figure 3.9: Advanced Options Configuration

- *Include Halui* - This will add the Halui user interface component. See the HALUI Chapter for more information on.
- *Include PyVCP* - This option adds the PyVCP panel base file or a sample file to work on. See the PyVCP Chapter for more information.
- *Include ClassicLadder PLC* - This option will add the ClassicLadder PLC (Programmable Logic Controller). See the ClassicLadder Chapter for more information.
- *Onscreen Prompt For Tool Change* - If this box is checked, LinuxCNC will pause and prompt you to change the tool when *M6* is encountered. This feature is usually only useful if you have presettable tools.

### 3.1.9 Полная конфигурация станка

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

### 3.1.10 Axis Travels and Homes

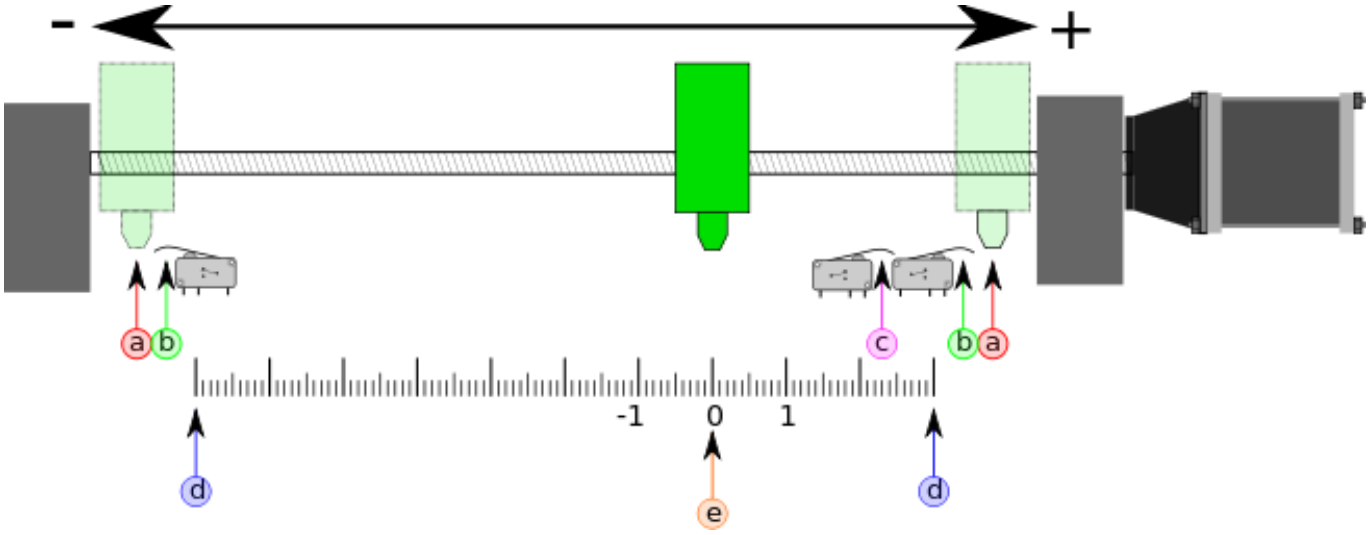



Figure 3.10: Axis Travel and Home

For each axis, there is a limited range of travel. The physical end of travel is called the *hard stop*.

**Warning**
If a mechanical hard stop were to be exceeded, the screw or the machine frame would be damaged!

Before the *hard stop* there is a *limit switch*. If the limit switch is encountered during normal operation, LinuxCNC shuts down the motor amplifier. The distance between the *hard stop* and *limit switch* must be long enough to allow an unpowered motor to coast to a stop.

Before the *limit switch* there is a *soft limit*. This is a limit enforced in software after homing. If a MDI command or G-code program would pass the soft limit, it is not executed. If a jog would pass the soft limit, it is terminated at the soft limit.

The *home switch* can be placed anywhere within the travel (between hard stops). As long as external hardware does not deactivate the motor amplifiers when the limit switch is reached, one of the limit switches can be used as a home switch.

The *zero position* is the location on the axis that is 0 in the machine coordinate system. Usually the *zero position* will be within the *soft limits*. On lathes, constant surface speed mode requires that machine *X=0* correspond to the center of spindle rotation when no tool offset is in effect.

The *home position* is the location within travel that the axis will be moved to at the end of the homing sequence. This value must be within the *soft limits*. In particular, the *home position* should never be exactly equal to a *soft limit*.

#### 3.1.10.1 Operating without Limit Switches

A machine can be operated without limit switches. In this case, only the soft limits stop the machine from reaching the hard stop. Soft limits only operate after the machine has been homed.

#### 3.1.10.2 Работа без концевиков исходного положения

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.

#### 3.1.10.3 Home and Limit Switch wiring options

The ideal wiring for external switches would be one input per switch. However, the PC parallel port only offers a total of 5 inputs, while there are as many as 9 switches on a 3-axis machine. Instead, multiple switches are wired together in various ways so that a smaller number of inputs are required.

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;.

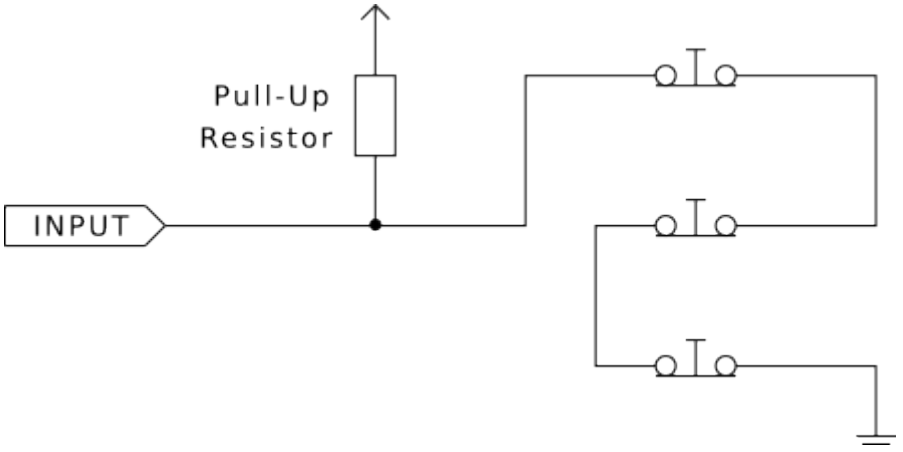



Figure 3.11: Normally Closed Switches (N/C) wiring in series (simplified diagram)

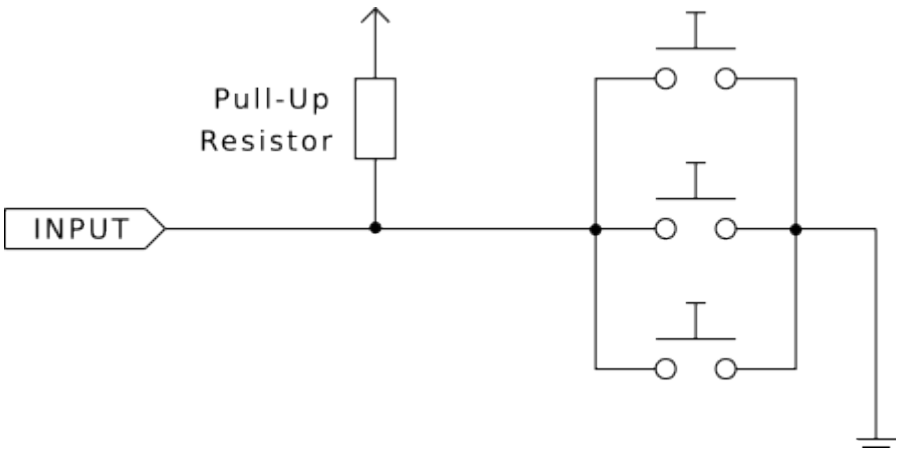



Figure 3.12: Normally Open Switches (N/O) wiring in parallel (simplified diagram)

The following combinations of switches are permitted in StepConf:

- Combine home switches for all axes
- Combine limit switches for all axes
- Combine both limit switches for one axis
- Combine both limit switches and the home switch for one axis
- Combine one limit switch and the home switch for one axis

The last two combinations are also appropriate when the type contact + home is used.

## 3.2 Mesa Configuration Wizard

PnCconf is made to help build configurations that utilize specific Mesa *Anything I/O* products.

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 is still in a development stage (Beta) so there are some bugs and lacking features. Please report bugs and suggestions to the LinuxCNC forum page or mailing list.

There are two trains of thought when using PnCconf:

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.

You navigate the wizard pages with the forward, back, and cancel buttons there is also a help button that gives some help information about the pages, diagrams and an output page.

**Tip**
PnCconf's help page should have the most up to date info and has additional details.

### 3.2.1 Step by Step Instructions

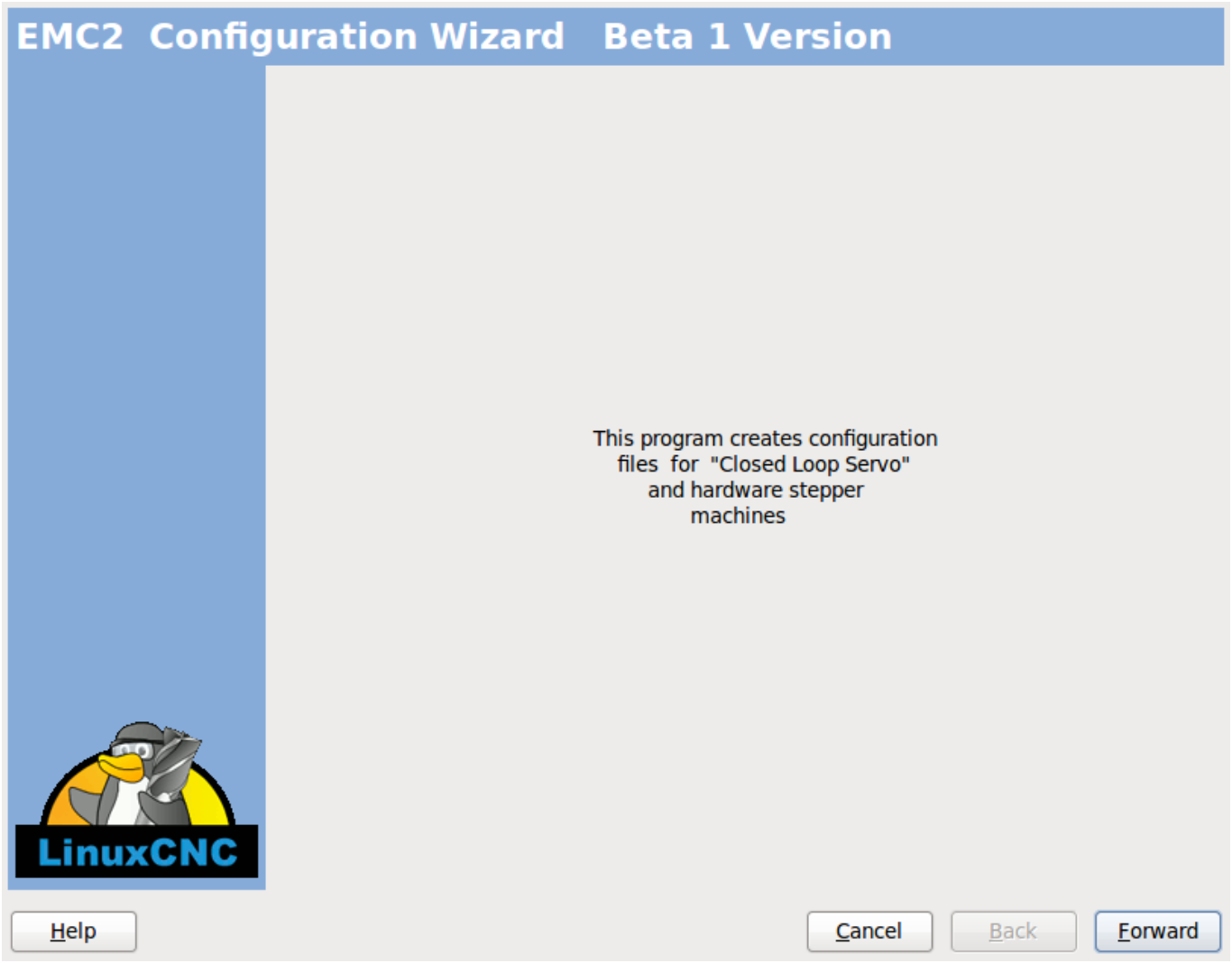



Figure 3.13: PnCconf Splash

### 3.2.2 Create or Edit

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 overwrite** 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 Basic Machine Information

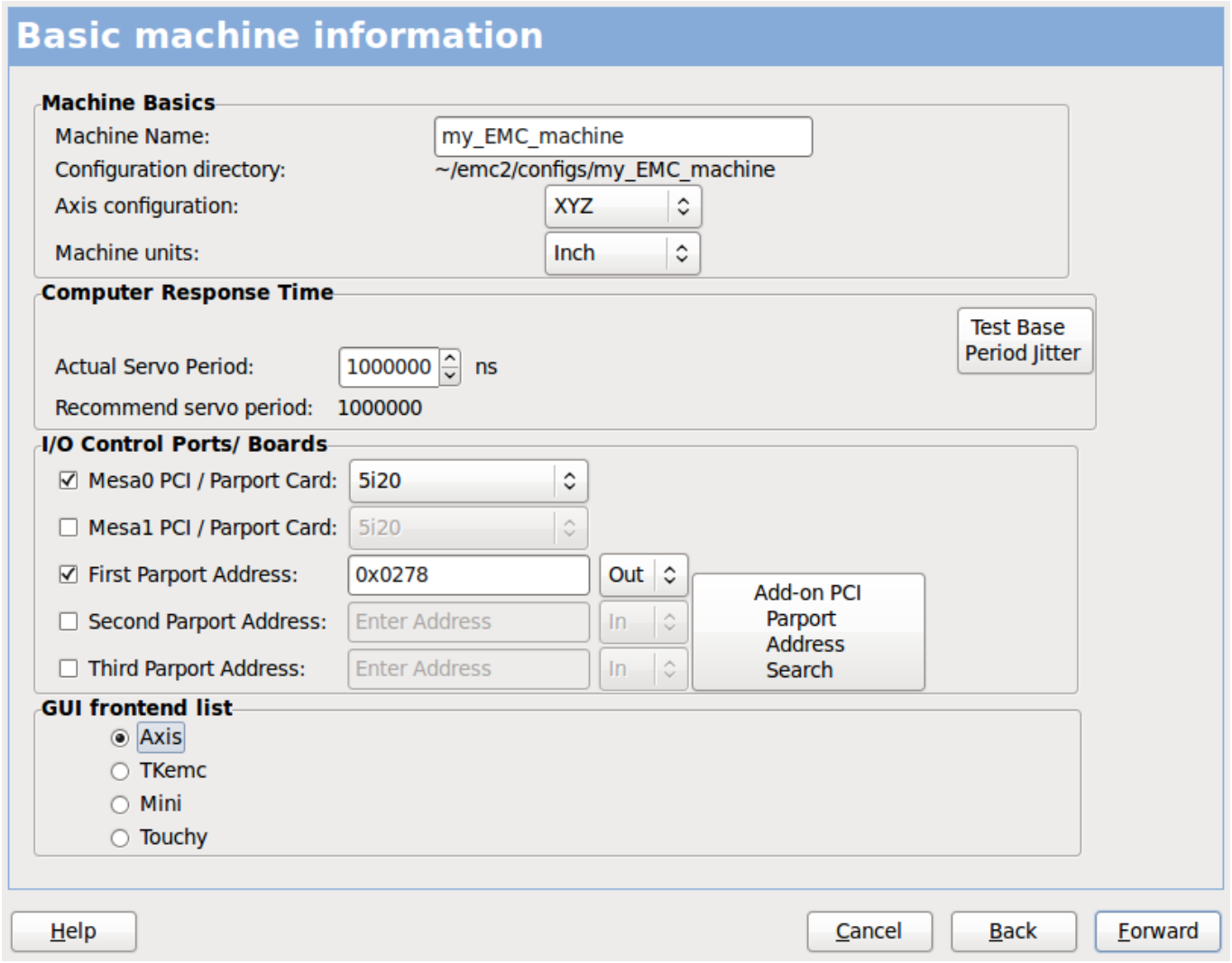



Figure 3.14: PnCconf Basic

**Machine Basics**

If you use a name with spaces PnCconf will replace the spaces with underscores (as a loose rule Linux doesn't like spaces in names). Picking an axis configuration 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 later steps in the configuration process.

---

**Tip**

Defaults are not converted when using metric so make sure they are sane values!

---

**Computer Response Time**

The servo period sets the heart beat of the system. Latency describes the difference between the time that the system is scheduled to perform and action and the time that it actually does

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

Testing the latency is crucial and a key thing to check before proceeding further. Please follow the directions on the Latency Test page before proceeding further.

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 than 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.

**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.

**Parallel Port**
В качестве простого ввода-вывода можно использовать до трех параллельных портов (называемых parports). Необходимо указать адрес parport. Вы можете либо ввести систему нумерации параллельных портов Linux (0,1 или 2), либо ввести фактический адрес. Адрес встроенного порта на плате часто равен 0x0278 или 0x0378 (записан в шестнадцатеричном формате), но его можно найти на странице BIOS. Страница BIOS находится при первом запуске компьютера. Для входа на нее необходимо нажать клавишу (например, F2). На странице BIOS вы можете найти адрес параллельного порта и установить режим, например SPP, EPP и т. д. На некоторых компьютерах эта информация отображается в течение нескольких секунд во время запуска. Для карт параллельного порта PCI адрес можно найти, нажав кнопку «поиск адреса порта». Откроется страница вывода справки со списком всех обнаруженных PCI-устройств. Там должна быть ссылка на устройство параллельного порта со списком адресов. Один из этих адресов должен работать. Не все параллельные порты PCI работают должным образом. Любой тип можно выбрать как «вход» (максимальное количество входных контактов) или «выход» (максимальное количество выходных контактов).

**GUI Front-end list**
This specifies the graphical display screens LinuxCNC will use. Each one has different option.

AXIS

- fully supports lathes.
- is the most developed and used front-end
- is designed to be used with mouse and keyboard

- is tkinter based so integrates PyVCP (Python based virtual control panels) naturally.
- has a 3D graphical window.
- allows VCP integrated on the side or in center tab

TkLinuxCNC

- hi contrast bright blue screen
- separate graphics window
- no VCP integration

Touchy

- Touchy was designed to be used with a touchscreen, some minimal physical switches and a MPG wheel.
- requires cycle-start, abort, and single-step signals and buttons
- It also requires shared axis MPG jogging to be selected.
- is GTK based so integrates GladeVCP (virtual control panels) naturally.
- allows VCP panels integrated in the center Tab
- has no graphical window
- look can be changed with custom themes

QtPlasmaC

- fully featured plasmac configuration based on the QtVCP infrastructure.
- mouse/keyboard operation or touchscreen operation
- no VCP integration

### 3.2.4 External Configuration

This page allows you to select external controls such as for jogging or overrides.

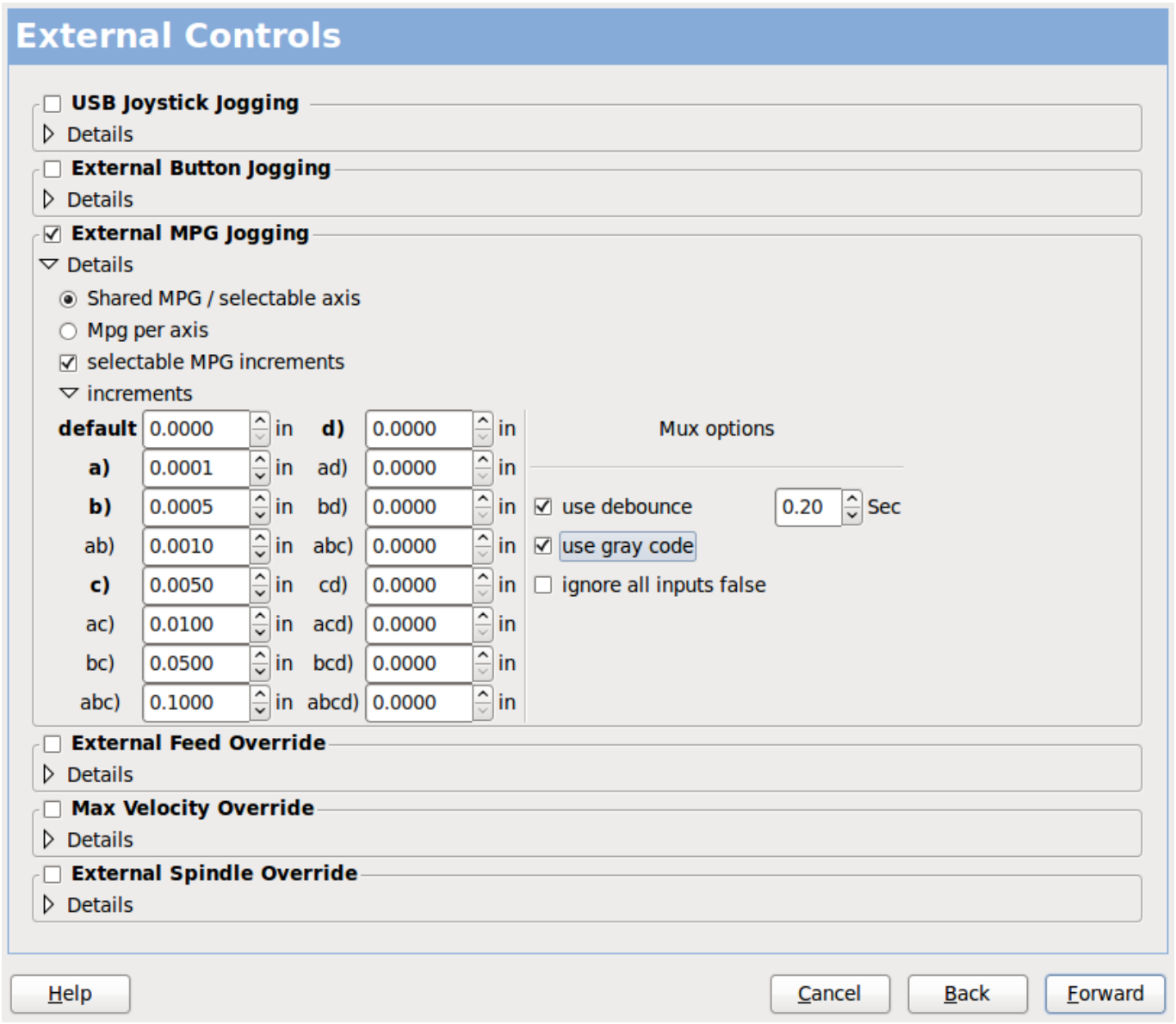



Figure 3.15: External Controls

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 jogging**

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.

- *Search for device rule* will search the system for rules, you can use this to find the name of devices you have already built with PnCconf.
- *Add a device rule* will allow you to configure a new device by following the prompts. You will need your device available.

- *test device* allows you to load a device, see its pin names and check its functions with halmeter.

joystick jogging uses HALUI and hal_input components.

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

**MPG Jogging**
Allows you to use a Manual Pulse Generator to jog the machine's axis.

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.

**Overrides**
PnCconf allows overrides of feed rates and/or spindle speed using a pulse generator (MPG) or switches (eg. rotary).

### 3.2.5 GUI Configuration

Here you can set defaults for the display screens, add virtual control panels (VCP), and set some LinuxCNC options..

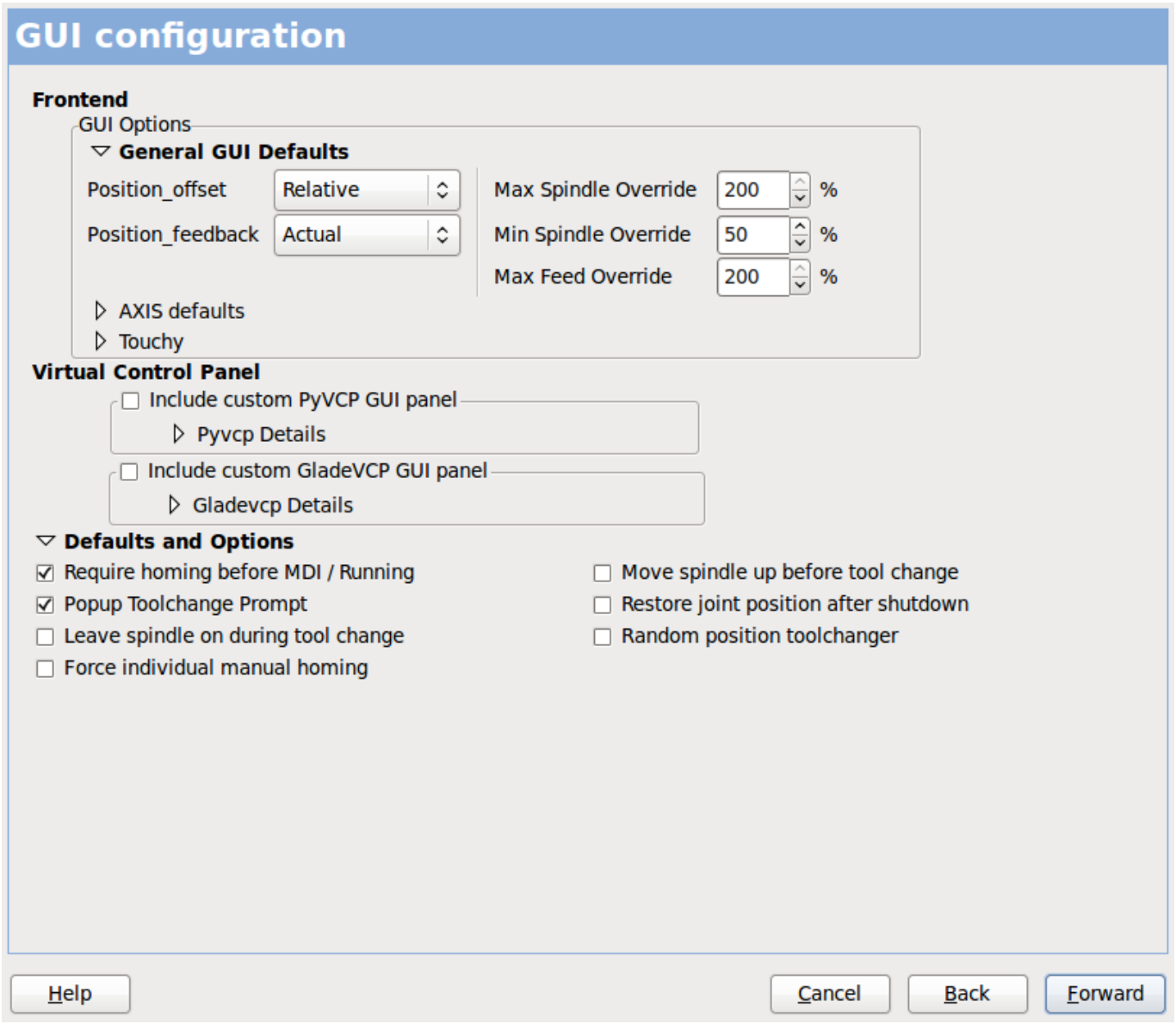



Figure 3.16: GUI Configuration

**Front-end GUI Options**

The default options allows general defaults to be chosen for any display screen.

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 options are specific to QtPlasmac, any common options that are not required will be disabled. If QtPlasmac is selected then the following screen will be a user button setup screen that is specific to QtPlasmaC and VCP options will not be available.

**VCP options**
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.

Конатткты HAL создаются для того, чтобы пользователь мог подключиться к своему собственному файлу HAL. Имеется образец панели отображения шпинделя, которую пользователь может использова как есть или дополнить. Вы можете выбрать пустой файл, в который позже можно будет добавить «widgets» элементов управления, или выбрать образец отображения шпинделя, который будет отображать скорость шпинделя и указывать, находится ли шпиндель на запрошенной скорости.

PnCconf подключит за вас соответствующие контакты HAL дисплея шпинделя. Если вы используете AXIS, панель будет интегрирована с правой стороны. Если вы не используете AXIS, панель будет автономной от основного экрана.

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 fit naturally inside of Touchy screen as they both use GTK to draw them, but by changing GladeVCP's theme it can be made to blend pretty well in AXIS (try Redmond).

Для создания XML-файлов он использует графический редактор. Контакты HAL создаются для подключения пользователя внутри его пользовательского файла HAL.

GladeVCP also allows much more sophisticated (and complicated) programming interaction, which PnCconf currently doesn't leverage (see GladeVCP in the manual).

PnCconf has sample panels for the user to use as-is or build on. With GladeVCP PnCconf will allow you to select different options on your sample display.

Under *sample options* select which ones you would like. The zero buttons use HALUI commands which you could edit later in the HALUI section.

Auto Z touch-off also requires the classic ladder touch-off program and a probe input selected. It requires a conductive touch-off plate and a grounded conductive tool. For an idea on how it works see:

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

Under *Display Options*, size, position, and force max can be used on a *stand-alone* panel for such things as placing the screen on a second monitor if the system is capable.

You can select a GTK theme which sets the basic look and feel of the panel. You Usually want this to match the front-end screen. These options will be used if you press the *Display sample button*. With GladeVCP depending on the front-end screen, you can select where the panel will display.

You can force it to be stand-alone or with AXIS it can be in the center or on the right side, with Touchy it can be in the center.

**Defaults and Options**

- Require homing before MDI / Running
  - If you want to be able to move the machine before homing uncheck this checkbox.
- Popup Tool Prompt
  - Choose between an on screen prompt for tool changes or export standard signal names for a User supplied custom tool changer HAL file
- Leave spindle on during tool change:
  - Used for lathes
- Force individual manual homing
- Move spindle up before tool change
- Restore joint position after shutdown
  - Used for non-trivial kinematics machines
- Random position tool changers
  - Used for tool changers that do not return the tool to the same pocket. You will need to add custom HAL code to support tool changers.

### 3.2.6 Mesa Configuration

The Mesa configuration pages allow one to utilize different firmwares. On the basic page you selected a Mesa card here you pick the available firmware and select what and how many components are available.

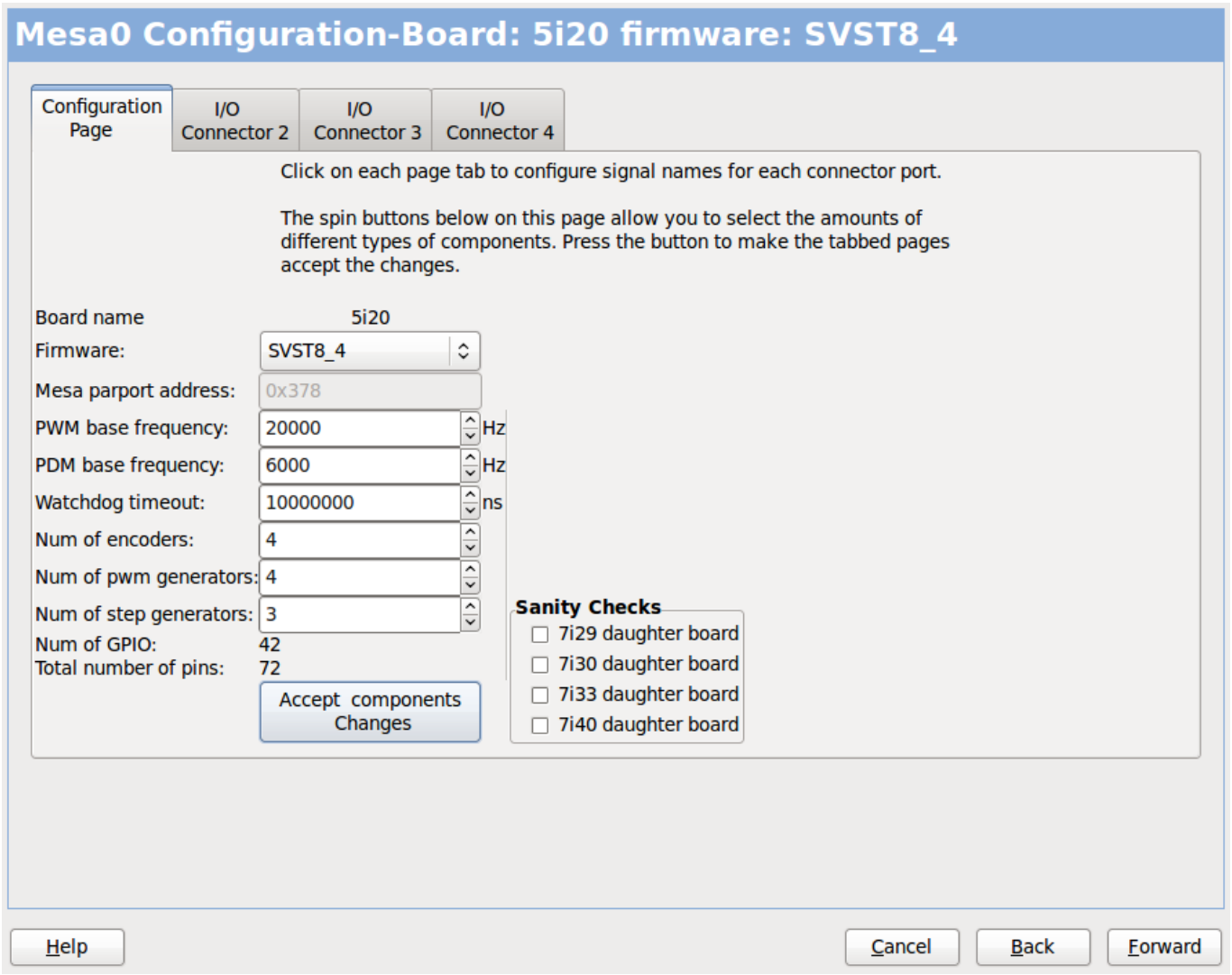



Figure 3.17: Mesa Board Configuration

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.

**Note**
Many PCI cards do not support the EPP protocol properly.

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.

**Important**
It's important to follow these to avoid damage and get the best performance.

```
The 7i33 requires PDM and a PDM base frequency of 6 MHz
The 7i29 requires PWM and a PWM base frequency of 20 kHz
The 7i30 requires PWM and a PWM base frequency of 20 kHz
The 7i40 requires PWM and a PWM base frequency of 50 kHz
The 7i48 requires UDM and a PWM base frequency of 24 kHz
```

**Watchdog time out**
: 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.

**Number of coders/PWM generators/STEP generators**
: You may choose the number of available components by deselecting unused ones. Not all component types are available with all firmware.

Выбор меньшего максимального количества компонентов позволяет получить больше контактов GPIO. При использовании дочерних плат имейте в виду, что нельзя отменять выбор контактов, которые использует карта. Например, некоторые прошивки поддерживают две карты 7i33. Если у вас есть только одна, вы можете отменить выбор достаточного количества компонентов, чтобы использовать разъем, поддерживающий вторую карту 7i33. Выбор компонентов отменяется численно сначала по наибольшему номеру, а затем вниз, без пропуска номера. Если при этом компоненты находятся не там, где вы хотите, вам придется использовать другую прошивку. Прошивка определяет, где, какие и максимальное количество компонентов. Возможна кастомная прошивка, спрашивайте при обращении к разработчикам LinuxCNC и Mesa. Использование кастомной прошивки в PnCconf требует специальных процедур и не всегда возможно — хотя я стараюсь сделать PnCconf максимально гибким.

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 Setup

The tabs are used to configure the input and output pins of the Mesa boards. PnCconf allows one to create custom signal names for use in custom HAL files.

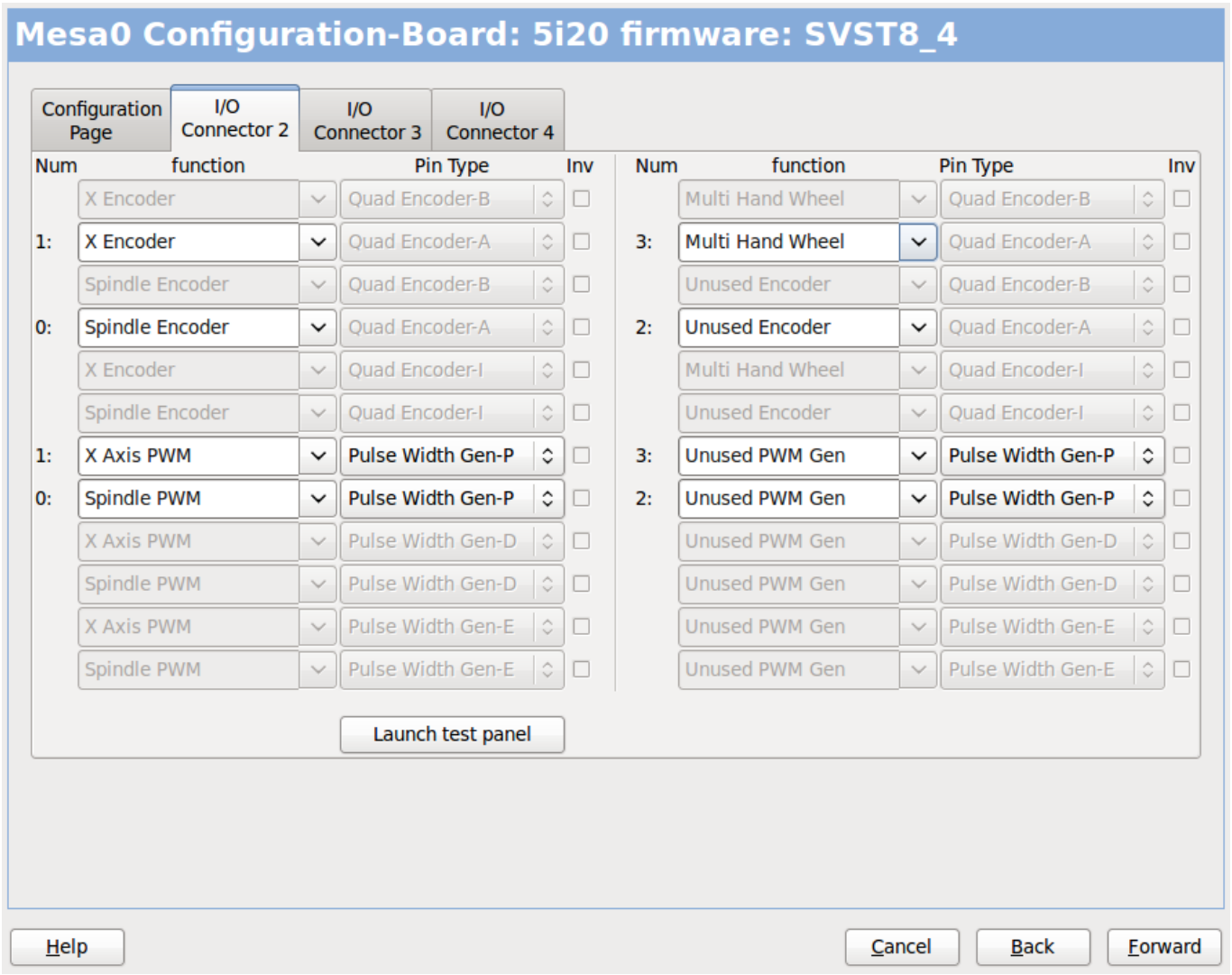



Figure 3.18: Mesa I/O C2 Setup

On this tab with this firmware the components are setup for a 7i33 daughter board, usually used with closed loop servos. Note the component numbers of the encoder counters and PWM drivers are not in numerical order. This follows the daughter board requirements.

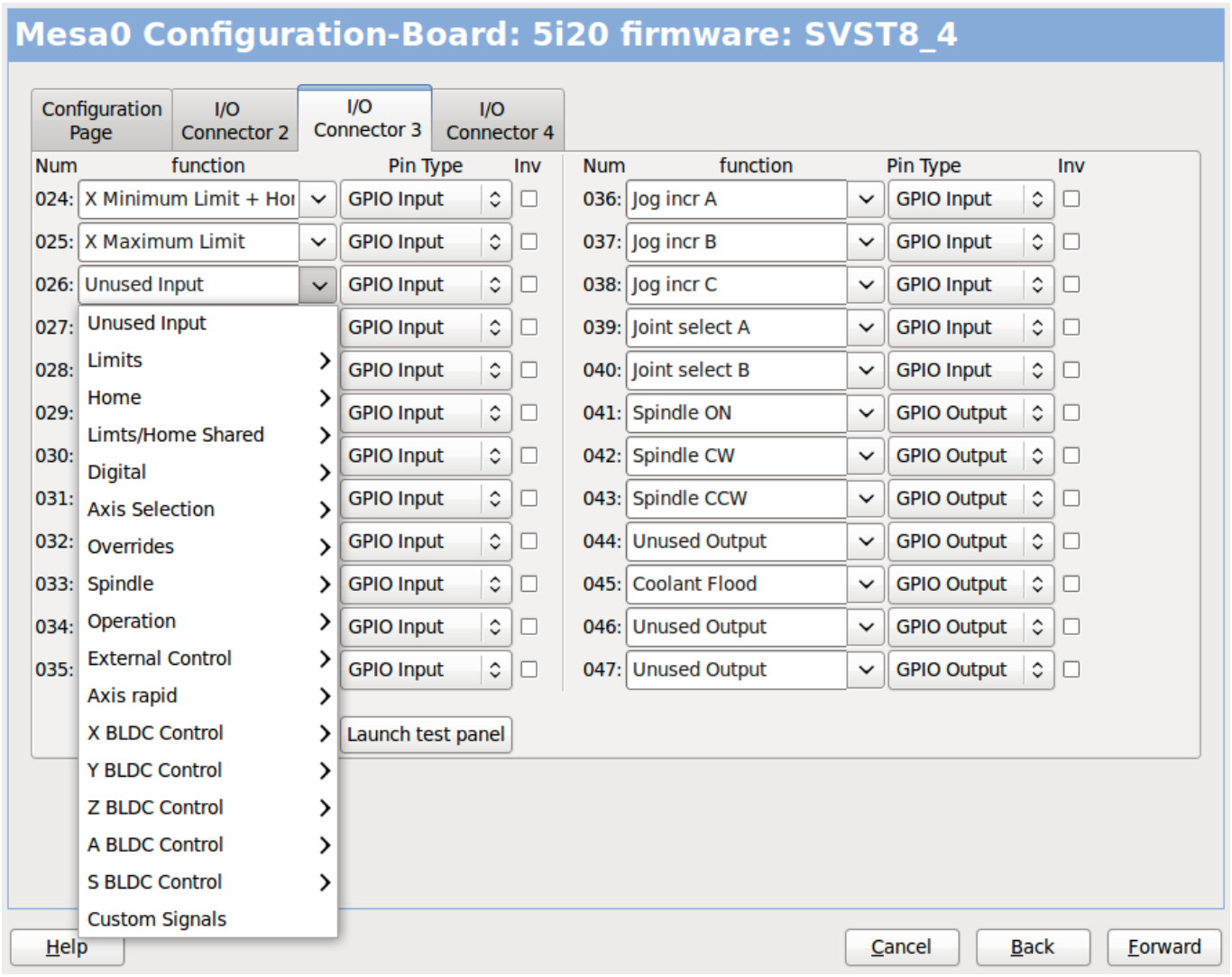



Figure 3.19: Mesa I/O C3 Setup

On this tab all the pins are GPIO. Note the 3 digit numbers - they will match the HAL pin number. GPIO pins can be selected as input or output and can be inverted.

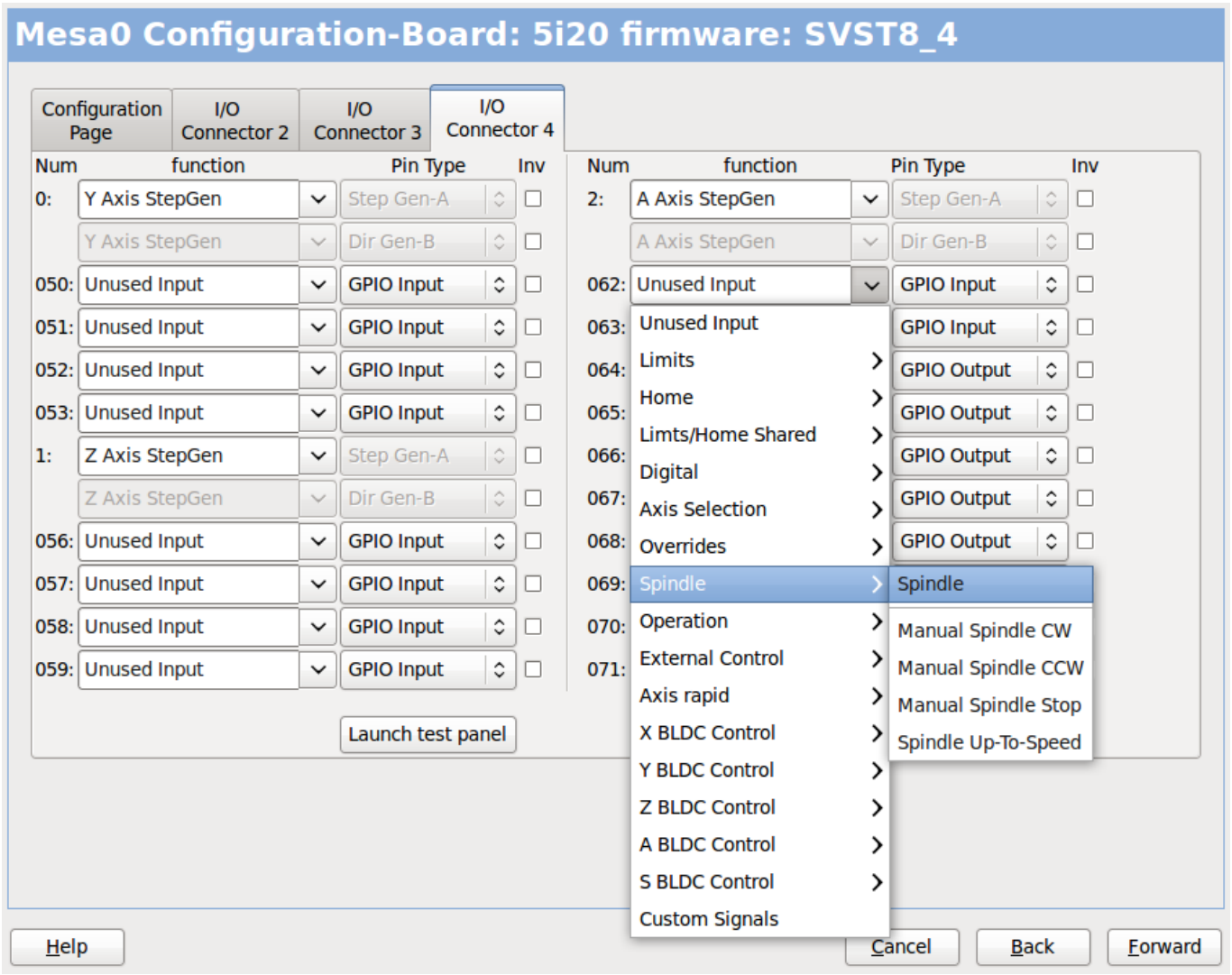
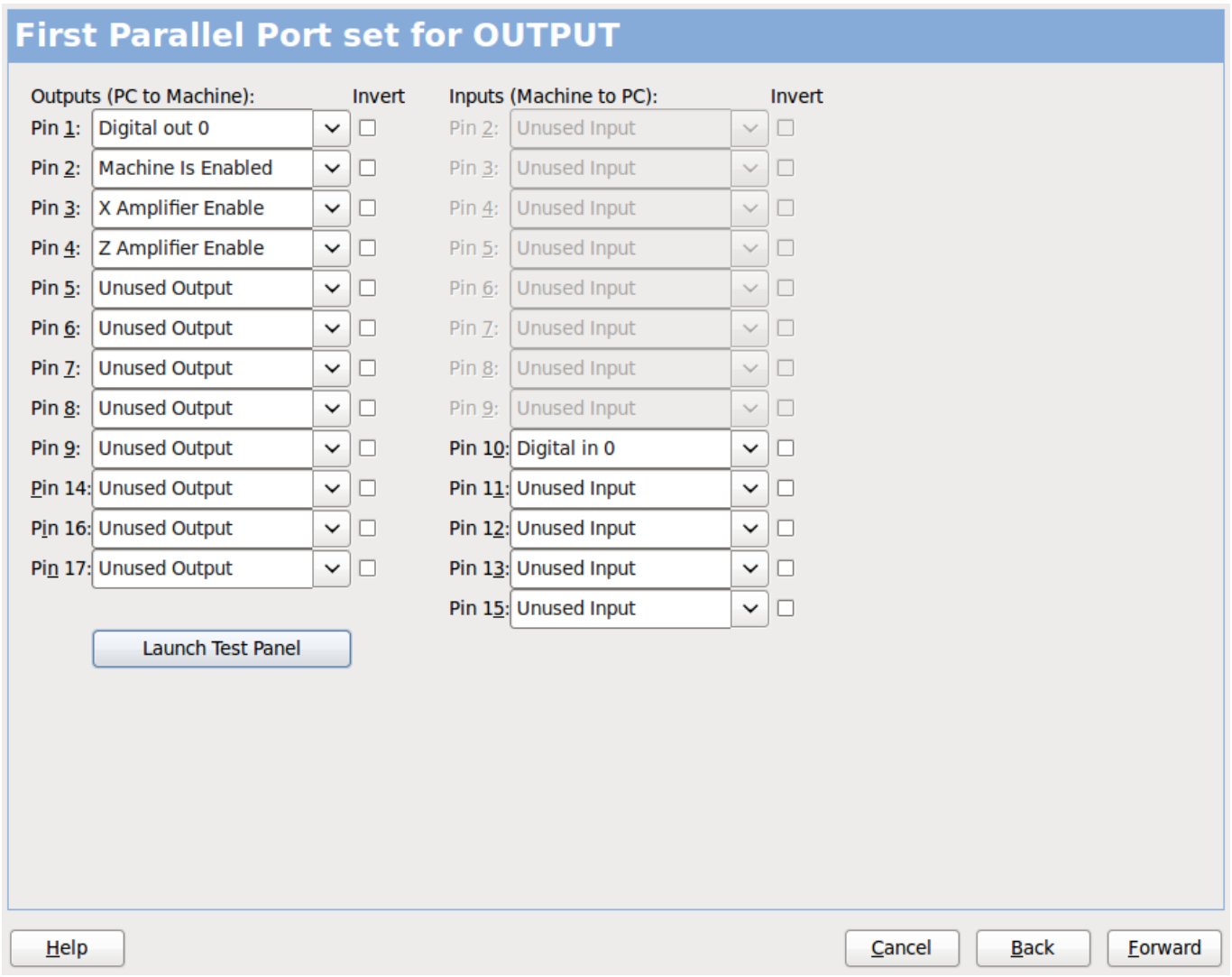



Figure 3.20: Mesa I/O C4 Setup

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 Parallel port configuration




The parallel port can be used for simple I/O similar to Mesa’s GPIO pins.

### 3.2.9 Axis Configuration

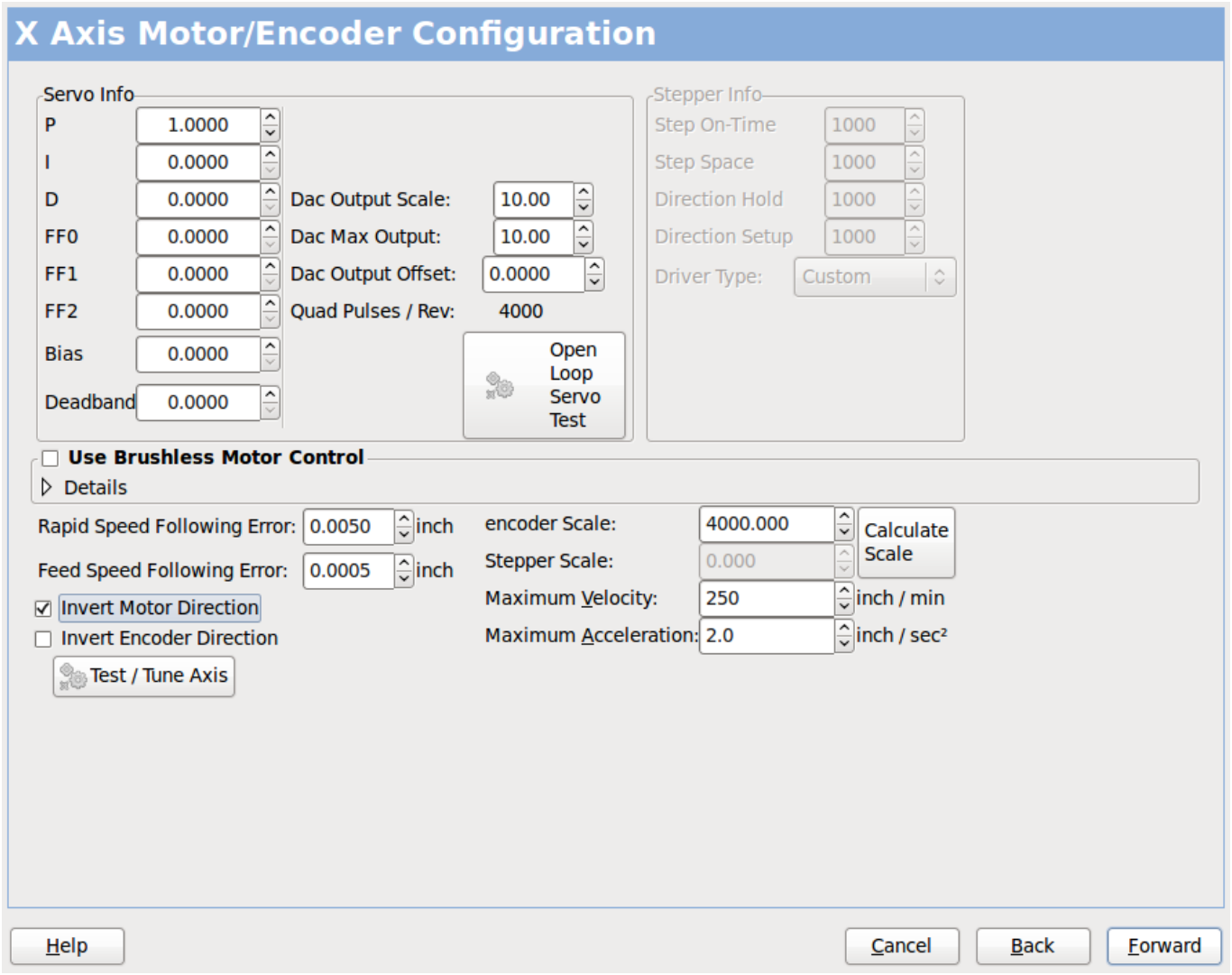



Figure 3.21: Axis Drive Configuration

This page allows configuring and testing of the motor and/or encoder combination. If using a servo motor an open loop test is available, if using a stepper a tuning test is available.

**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.

**Warning**
If the motor and encoder do not agree on counting direction then the servo will run away when using PID control.

Since at the moment PID settings can not be tested in PnCconf the settings are really for when you re-edit a config - enter your tested PID settings.

**DAC scale**
DAC scaling, max output and offset are used to tailor the DAC output.

**Compute DAC**
These two values are the scale and offset factors for the axis output to the motor amplifiers. 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: 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.

The scale and offset can be used to linearize the DAC as well, resulting in values that reflect the combined effects of amplifier gain, DAC non-linearity, DAC units, etc. To do this, follow this procedure:

- Build a calibration table for the output, driving the DAC with a desired voltage and measuring the result:

Table 3.2: Измерения выходного напряжения

| **Необработанные** | **Измеренные** |
|---:|---:|
| -10 | **-9.93** |
| -9 | **-8.83** |
| 0 | **-0.96** |
| 1 | **-0.03** |
| 9 | **9.87** |
| 10 | **10.07** |

- Do a least-squares linear fit to get coefficients a, b such that meas=a*raw+b
- Обратите внимание: нам нужен необработанный выходной сигнал, чтобы результат измерения был идентичен заданному выходному сигналу. Это означает
  - cmd=a*raw+b
  - raw=(cmd-b)/a
- В результате коэффициенты a и b из линейной аппроксимации можно использовать непосредственно в качестве масштаба и смещения для контроллера.

**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.

**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.

**Brushless Motor Control**
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.

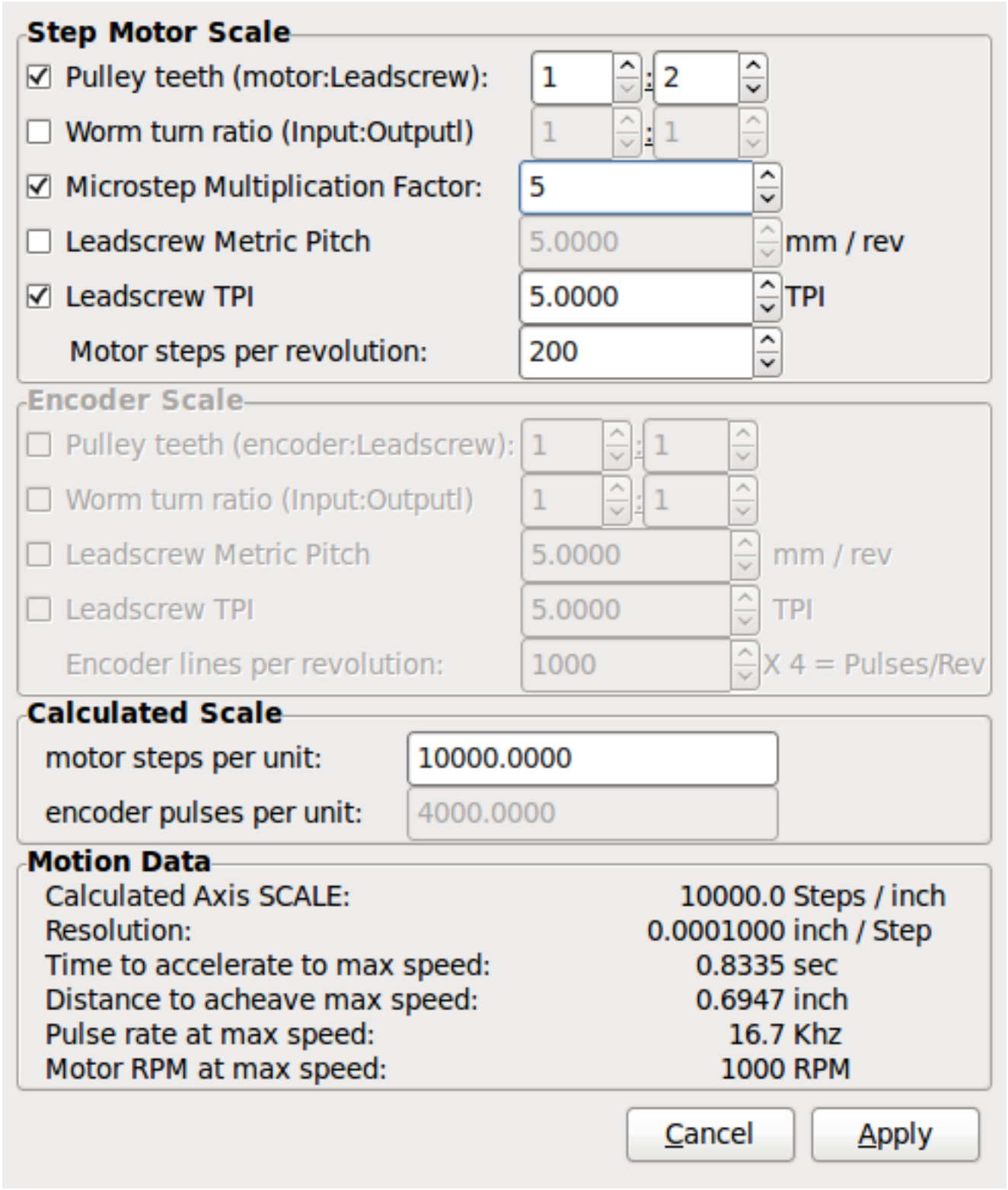



Figure 3.22: Axis Scale Calculation

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.

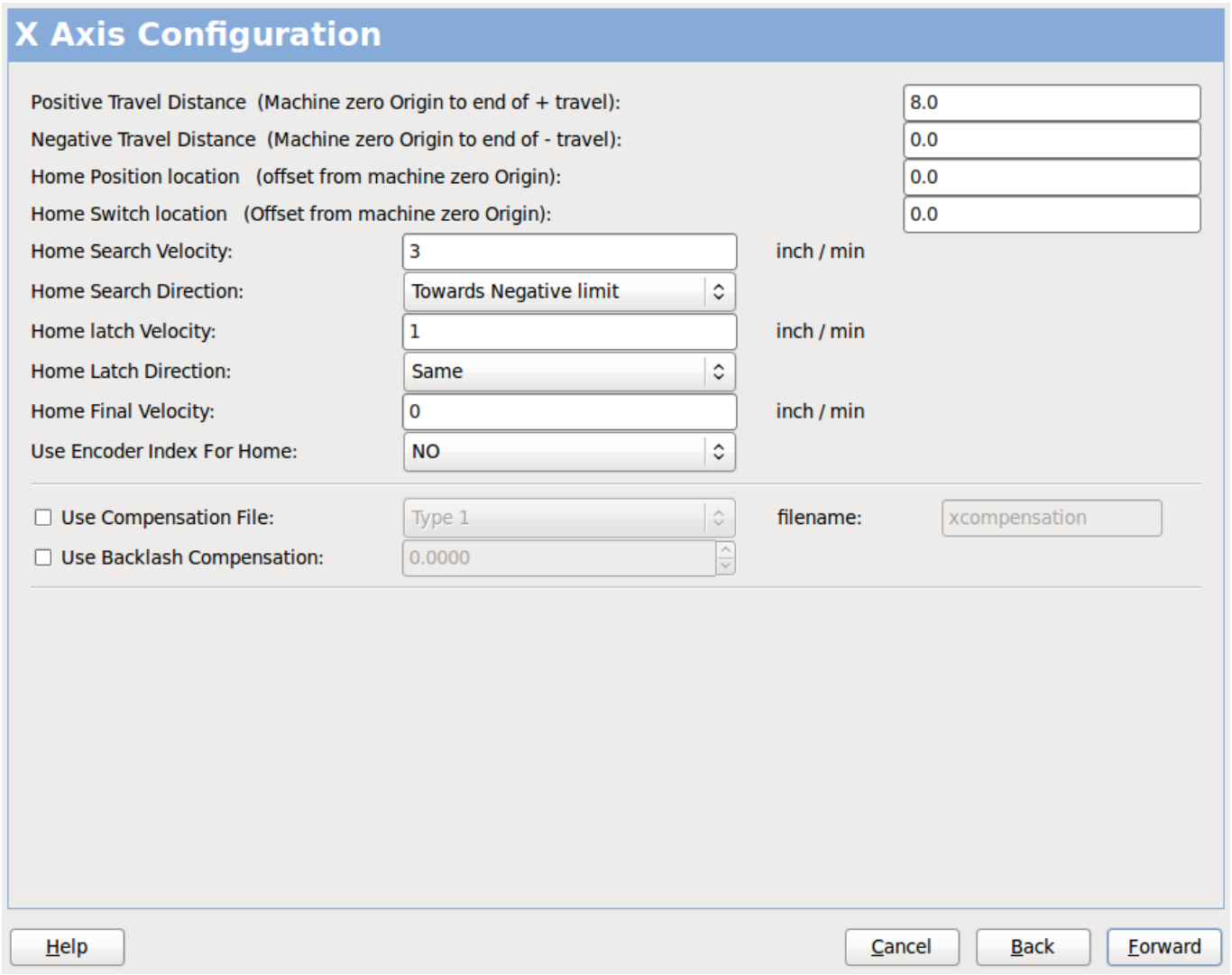



Figure 3.23: Axis Configuration

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.

**Important**
It is very important to start with the axis moving in the right direction or else getting homing right is very difficult!

Remember positive and negative directions refer to the TOOL not the table as per the Machinists handbook.

**On a typical knee or bed mill**

- when the TABLE moves out that is the positive Y direction
- when the TABLE moves left that is the positive X direction
- when the TABLE moves down that is the positive Z direction

- when the HEAD moves up that is the positive Z direction

**On a typical lathe**

- when the TOOL moves right, away from the chuck
- that is the positive Z direction
- when the TOOL moves toward the operator
- that is the positive X direction. Some lathes have X opposite (e.g., tool on back side), that will work fine but AXIS graphical display can not be made to reflect this.

When using homing and / or limit switches LinuxCNC expects the HAL signals to be true when the switch is being pressed / tripped. If the signal is wrong for a limit switch then LinuxCNC will think the machine is on end of limit all the time. If the home switch search logic is wrong LinuxCNC will seem to home in the wrong direction. What it actually is doing is trying to BACK off the home switch.

**Decide on limit switch location**
: Limit switches are the back up for software limits in case something electrical goes wrong, e.g., in case of a servo runaway. Limit switches should be placed so that the machine does not hit the physical end of the axis movement. Remember the axis will coast past the contact point if moving fast. Limit switches should be *active low* on the machine, i.e., power runs through the switches all the time - a loss of power (open switch) trips. While one could wire them the other way, this is fail safe. This may need to be inverted so that the HAL signal in LinuxCNC in *active high* - a TRUE means the switch was tripped. When starting LinuxCNC if you get an on-limit warning, and axis is NOT tripping the switch, inverting the signal is probably the solution. (use HALMETER to check the corresponding HAL signal eg. joint.0.pos-lim-sw-in X axis positive limit switch)

**Decide on the home switch location**
: If you are using limit switches You may as well use one as a home switch. A separate home switch is useful if you have a long axis that in use is usually a long way from the limit switches or moving the axis to the ends presents problems of interference with material. Note, a long shaft in a lathe makes it hard to home to limits with out the tool hitting the shaft, so a separate home switch closer to the middle may be better. If you have an encoder with index then the home switch acts as a course home and the index will be the actual home location.

**Decide on the MACHINE ORIGIN position**
: MACHINE ORIGIN is what LinuxCNC uses to reference all user coordinate systems from. I can think of little reason it would need to be in any particular spot. There are only a few G-codes that can access the MACHINE COORDINATE system.( G53, G30 and G28 ) If using tool-change-at-G30 option having the origin at the tool change position may be convenient. By convention, it may be easiest to have the ORIGIN at the home switch.

**Decide on the (final) HOME POSITION**
: this just places the carriage at a consistent and convenient position after LinuxCNC figures out where the ORIGIN is.

**Measure / calculate the positive / negative axis travel distances**
: Move the axis to the origin. Mark a reference on the movable slide and the non-movable support (so they are in line) move the machine to the end of limits. Measure between the marks that is one of the travel distances. Move the table to the other end of travel. Measure the marks again. That is the other travel distance. If the ORIGIN is at one of the limits then that travel distance will be zero.

**(machine) ORIGIN**
: The Origin is the MACHINE zero point. (not the zero point you set your cutter / material at). LinuxCNC uses this point to reference everything else from. It should be inside the software limits. LinuxCNC uses the home switch location to calculate the origin position (when using home switches or must be manually set if not using home switches.

**Travel distance**
This is the maximum distance the axis can travel in each direction. This may or may not be able to be measured directly from origin to limit switch. The positive and negative travel distances should add up to the total travel distance.

**POSITIVE TRAVEL DISTANCE**
This is the distance the Axis travels from the Origin to the positive travel distance or the total travel minus the negative travel distance. You would set this to zero if the origin is positioned at the positive limit. The will always be zero or a positive number.

**NEGATIVE TRAVEL DISTANCE**
This is the distance the Axis travels from the Origin to the negative travel distance. or the total travel minus the positive travel distance. You would set this to zero if the origin is positioned at the negative limit. This will always be zero or a negative number. If you forget to make this negative PnCconf will do it internally.

**(Final) HOME POSITION**
This is the position the home sequence will finish at. It is referenced from the Origin so can be negative or positive depending on what side of the Origin it is located. When at the (final) home position if you must move in the Positive direction to get to the Origin, then the number will be negative.

**HOME SWITCH LOCATION**
This is the distance from the home switch to the Origin. It could be negative or positive depending on what side of the Origin it is located. When at the home switch location if you must move in the Positive direction to get to the Origin, then the number will be negative. If you set this to zero then the Origin will be at the location of the limit switch (plus distance to find index if used).

**Home Search Velocity**
Course home search velocity in units per minute.

**Home Search Direction**
Sets the home switch search direction either negative (i.e., towards negative limit switch) or positive (i.e., towards positive limit switch).

**Home Latch Velocity**
Fine Home search velocity in units per minute.

**Home Final Velocity**
Velocity used from latch position to (final) home position in units per minute. Set to 0 for max rapid speed.

**Home latch Direction**
Allows setting of the latch direction to the same or opposite of the search direction.

**Use Encoder Index For Home**
LinuxCNC will search for an encoder index pulse while in the latch stage of homing.

**Use Compensation File**
Allows specifying a Comp filename and type. Allows sophisticated compensation. See AXIS Section of the INI chapter.

**Use Backlash Compensation**
Allows setting of simple backlash compensation. Can not be used with Compensation File. See AXIS Section of the INI chapter.

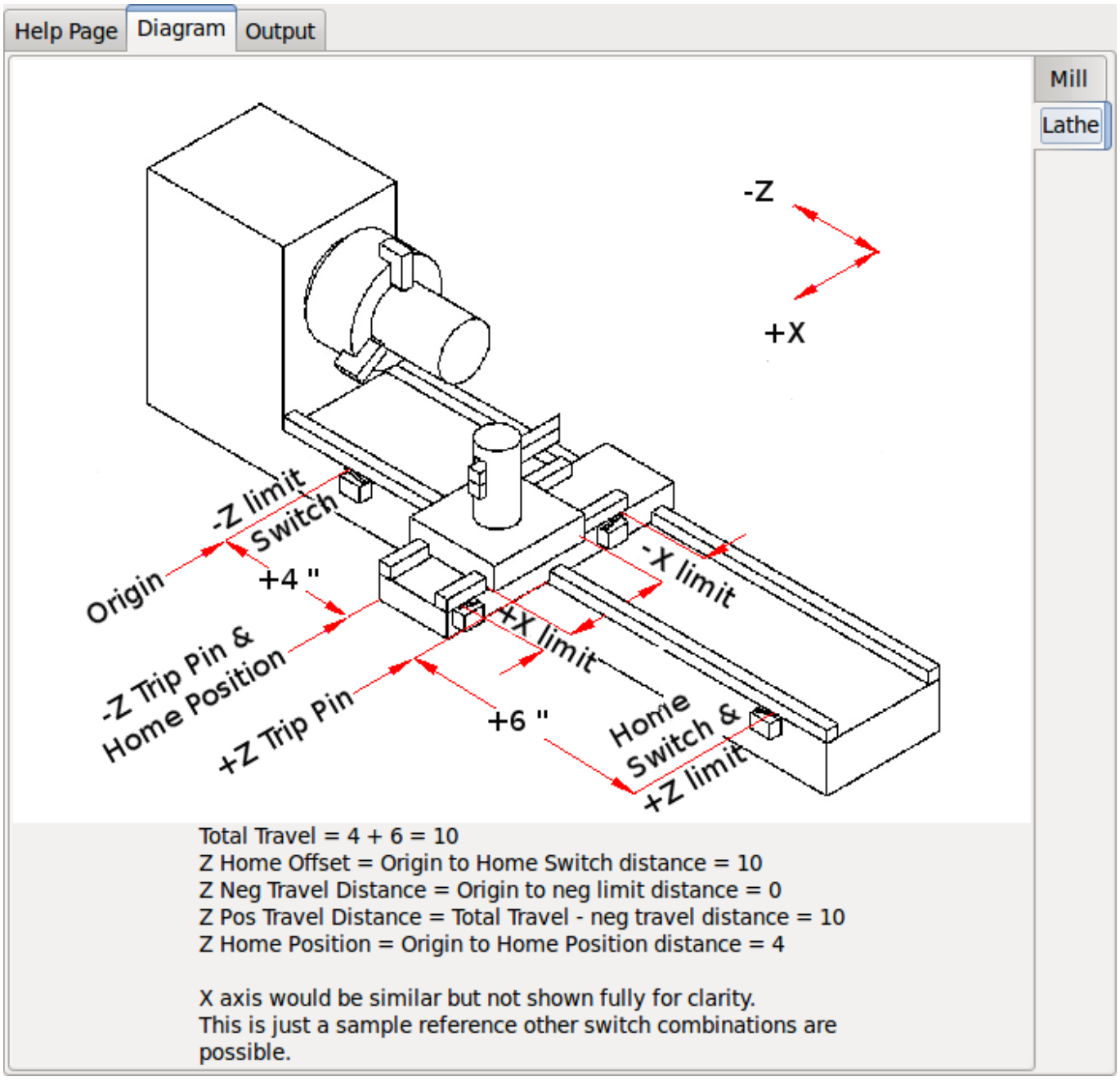



Figure 3.24: AXIS Help Diagram

The diagram should help to demonstrate an example of limit switches and standard axis movement directions. In this example the Z axis was two limit switches, the positive switch is shared as a home switch. The MACHINE ORIGIN (zero point) is located at the negative limit. The left edge of the carriage is the negative trip pin and the right the positive trip pin. We wish the FINAL HOME POSITION to be 4 inches away from the ORIGIN on the positive side. If the carriage was moved to the positive limit we would measure 10 inches between the negative limit and the negative trip pin.

### 3.2.10 Spindle Configuration

If you select spindle signals then this page is available to configure spindle control.

**Tip**
Many of the option on this page will not show unless the proper option was selected on previous pages!

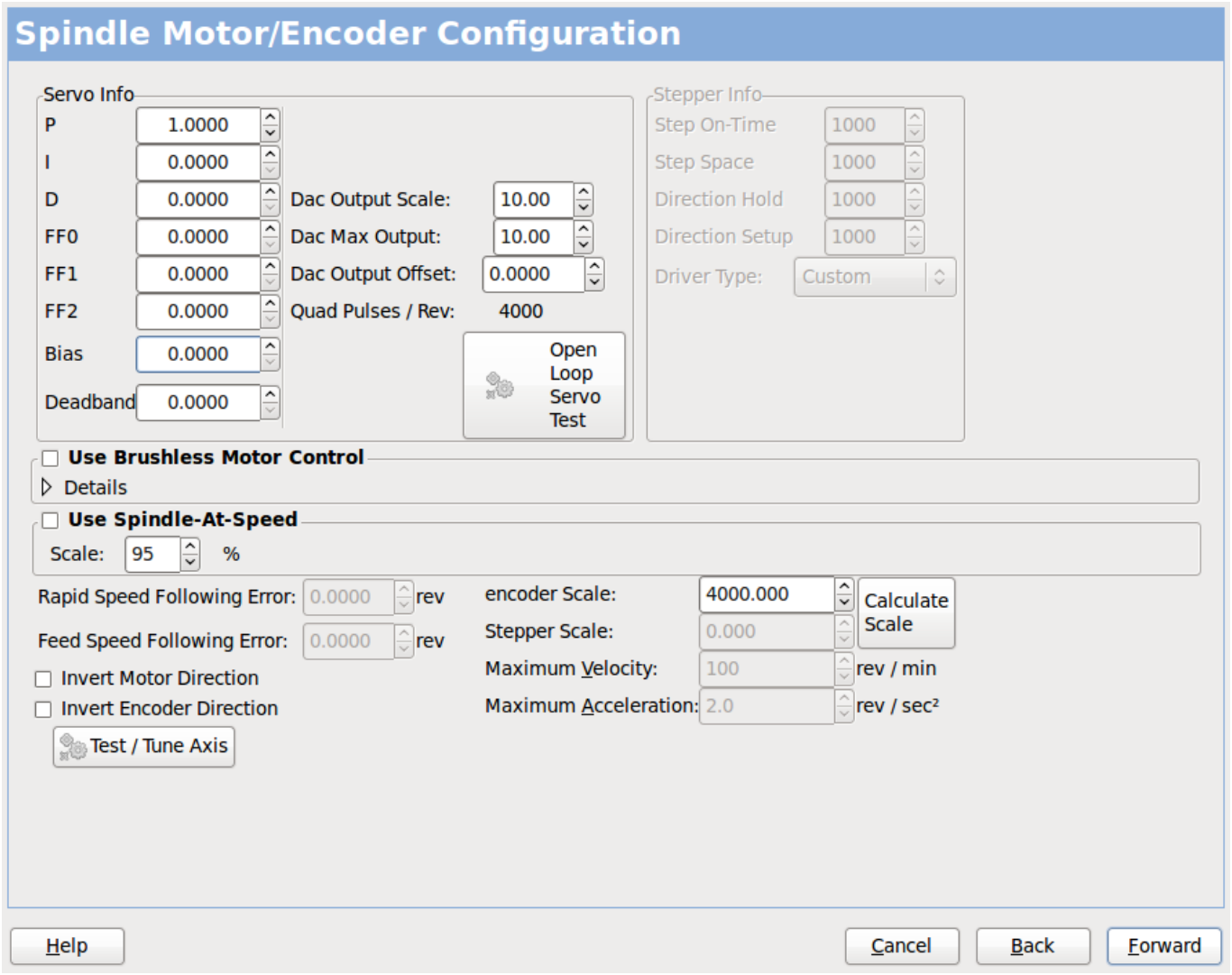



Figure 3.25: Spindle Motor/Encoder Configuration

This page is similar to the axis motor configuration page.

There are some differences:

- Unless one has chosen a stepper driven spindle there is no acceleration or velocity limiting.
- There is no support for gear changes or ranges.
- If you picked a VCP spindle display option then spindle-at-speed scale and filter settings may be shown.
- Spindle-at-speed allows LinuxCNC to wait till the spindle is at the requested speed before moving the axis. This is particularly handy on lathes with constant surface feed and large speed diameter changes. It requires either encoder feedback or a digital spindle-at-speed signal typically connected to a VFD drive.

- If using encoder feedback, you may select a spindle-at-speed scale setting that specifies how close the actual speed must be to the requested speed to be considered at-speed.
- If using encoder feedback, the VCP speed display can be erratic - the filter setting can be used to smooth out the display. The encoder scale must be set for the encoder count / gearing used.
- If you are using a single input for a spindle encoder you must add the line: setp hm2_7i43.0.encoder.00.cou mode 1 (changing the board name and encoder number to your requirements) into a custom HAL file. See the Encoders Section in Hostmot2 for more info about counter mode.

### 3.2.11 Advanced Options

Это позволяет настраивать команды HALUI и загружать ClassicLadder и примеры программ релейной логики. Если вы выбрали параметры GladeVCP, такие как обнуление оси, будут отображаться команды. См. HALUI Chapter для получения дополнительной информации об использовании пользовательских команд halcmds. Существует несколько параметров программ релейной логики. Программа Estop позволяет внешнему переключателю ESTOP или ГИП подавать сигнал Estop. Она также имеет синхронизированный сигнал насоса смазки. Автоматическое касание по оси Z осуществляется с помощью контактной пластины, кнопки касания GladeVCP и специальных команд HALUI для установки текущего пользовательского начала координат в ноль и быстрой очистки. Программа последовательной шины Modbus по сути представляет собой пустой шаблон программы, который настраивает ClassicLadder для последовательной шины Modbus. См. ClassicLadder Chapter в руководстве.

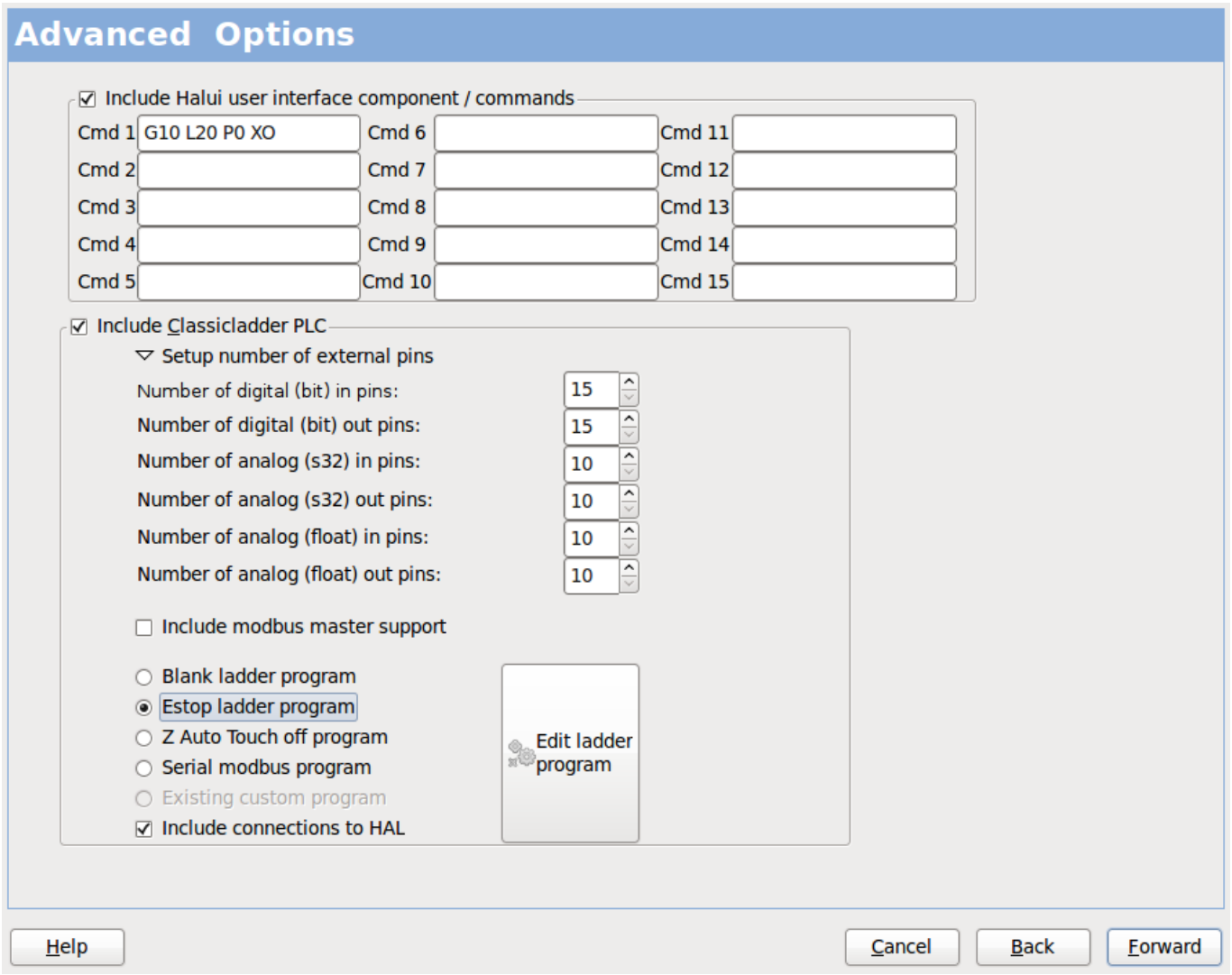



Figure 3.26: PnCconf, advanced options

### 3.2.12 HAL Components

On this page you can add additional HAL components you might need for custom HAL files. In this way one should not have to hand edit the main HAL file, while still allowing user needed components.

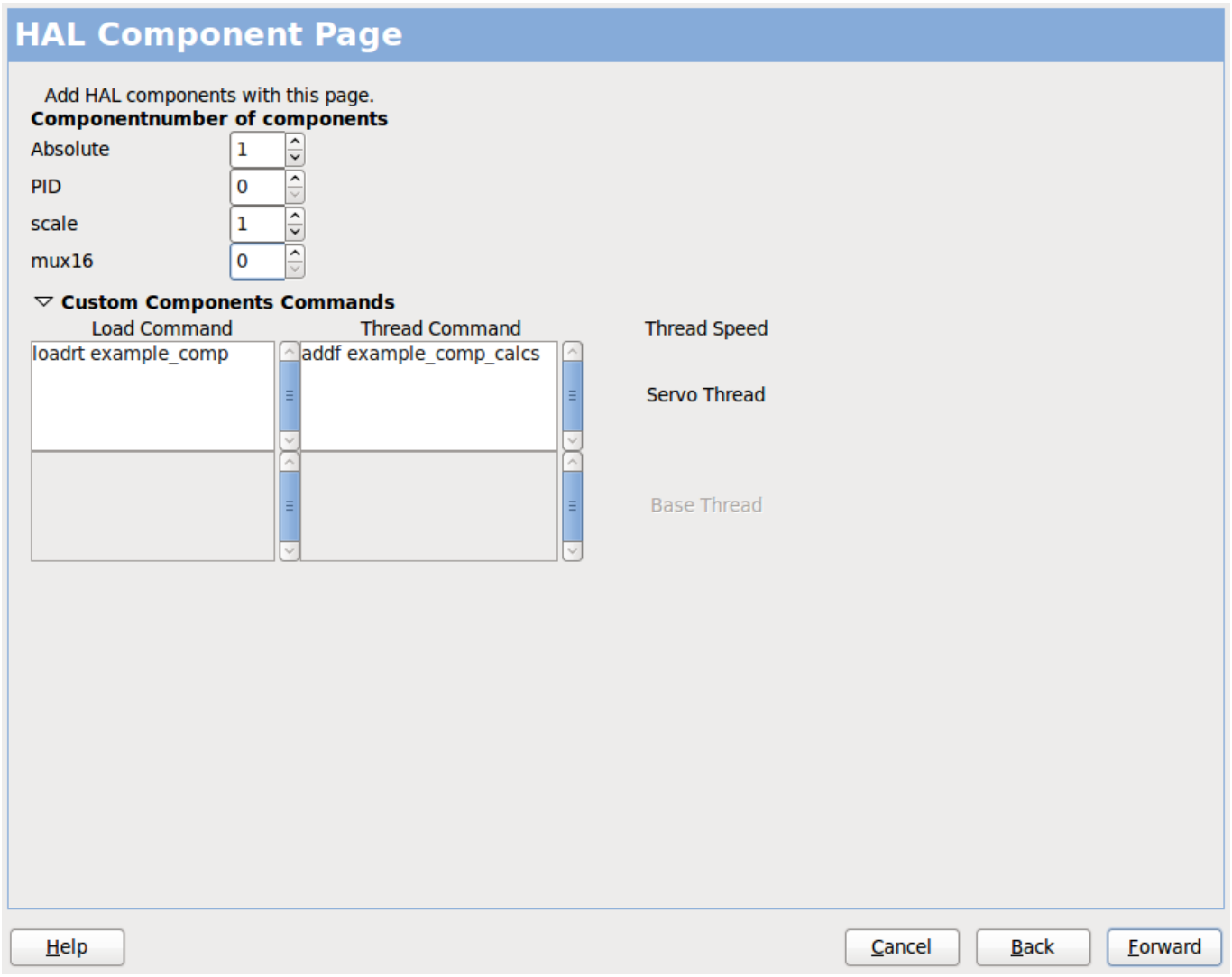



Figure 3.27: HAL Components

The first selection is components that pncconf uses internally. You may configure pncconf to load extra instances of the components for your custom HAL file.

Select the number of instances your custom file will need, PnCconf will add what it needs after them.

Meaning if you need 2 and PnCconf needs 1 PnCconf will load 3 instances and use the last one.

**Custom Component Commands**

This selection will allow you to load HAL components that PnCconf does not use. Add the loadrt or loadusr command, under the heading *loading command* Add the addf command under the heading *Thread command*. The components will be added to the thread between reading of inputs and writing of outputs, in the order you write them in the *thread command*.

### 3.2.13 Advanced Usage Of PnCconf

PnCconf does its best to allow flexible customization by the user. PnCconf has support for custom signal names, custom loading of components, custom HAL files and custom firmware.

There are also signal names that PnCconf always provides regardless of options selected, for user's custom HAL files With some thought most customizations should work regardless if you later select different options in PnCconf.

Eventually if the customizations are beyond the scope of PnCconf's frame work you can use PnCconf to build a base config or use one of LinuxCNC's sample configurations and just hand edit it to what ever you want.

**Custom Signal Names**
If you wish to connect a component to something in a custom HAL file write a unique signal name in the combo entry box. Certain components will add endings to your custom signal name:

Encoders will add < customname > +:

- position
- count
- velocity
- index-enable
- reset

Steppers add:

- enable
- counts
- position-cmd
- position-fb
- velocity-fb

PWM add:

- enable
- value

GPIO pins will just have the entered signal name connected to it

In this way one can connect to these signals in the custom HAL files and still have the option to move them around later.

**Custom Signal Names**
The HAL Components page can be used to load components needed by a user for customization.

**Loading Custom Firmware**
PnCconf searches for firmware on the system and then looks for the XML file that it can convert to what it understands. These XML files are only supplied for officially released firmware from the LinuxCNC team. To utilize custom firmware one must convert it to an array that PnCconf understands and add its file path to PnCconf's preference file. By default this path searches the desktop for a folder named custom_firmware and a file named firmware.py.

The hidden preference file is in the user's home file, is named .pncconf-preferences and require one to select *show hidden files* in your file manager to see and edit it or on the command line you use *ls* with the *-a* option. The contents of this file can be seen when you first load PnCconf - press the help button and look at the output page.

Ask on the LinuxCNC mail-list or forum for info about converting custom firmware. Not all firmware can be utilized with PnCconf.

**Custom HAL Files**

There are four custom files that you can use to add HAL commands to:

- custom.hal is for HAL commands that don't have to be run after the GUI frontend loads. It is run after the configuration-named HAL file.
- custom_postgui.hal is for commands that must be run after AXIS loads or a standalone PyVCP display loads.
- custom_gvcp.hal is for commands that must be run after GladeVCP is loaded.
- shutdown.hal is for commands to run when LinuxCNC shuts down in a controlled manner.

# Chapter 4

# Конфигурация

## 4.1 Концепции системного разработчика

### 4.1.1 Расплоложение файлов

LinuxCNC ищет файлы конфигурации и G-кода в определенном месте. Местоположение зависит от того, как вы используете LinuxCNC.

#### 4.1.1.1 На установленном пакете

Если вы используете LinuxCNC с Live CD или установили его через .deb и используете средство выбора конфигурации *LinuxCNC* из меню, LinuxCNC просматривает следующие каталоги:

- Каталог LinuxCNC находится по адресу */home/user-name/linuxcnc*.
- Каталоги конфигурации расположены по адресу */home/user-name/linuxcnc/configs*.
  - Файлы конфигурации расположены по адресу */home/user-name/linuxcnc/configs/name-of-config*.
- Файлы G-кода расположены по адресу */home/user-name/linuxcnc/nc_files*.

Например, для конфигурации под названием Mill и имени пользователя Fred структура каталогов и файлов будет выглядеть следующим образом.

- */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 Командная строка

Если вы запустите LinuxCNC из командной строки и укажете имя и расположение INI-файла, расположение файлов может находиться в другом месте. Чтобы просмотреть параметры запуска LinuxCNC из командной строки, выполните *linuxcnc -h*.

---

**Note**
Дополнительные места для некоторых файлов можно настроить в INI-файле. См. раздел <<sub:ini:sec:display,[DISPLAY]>> и раздел <<sub:ini:sec:rs274ngc,[RS274NGC]>>.

---

### 4.1.2 Файлы

Для каждого каталога конфигурации требуются как минимум следующие файлы:

- INI-файл .ini
- Файл HAL .hal или файл HALTCL .tcl, указанный в разделе HAL INI-файла.

---

**Note**
Для некоторых ГИП могут потребоваться другие файлы.

---

При желании вы также можете иметь:

- Файл переменных .var
  - Если вы не включите файл .var в каталоге, но включите <<sub:ini:sec:rs274ngc,[RS274NGC]>> PARAMETER_FILE=somefilename.var, файл будет создан для вас при запуске LinuxCNC.
  - Если вы не включаете файл .var и элемент [RS274NGC] PARAMETER_FILE, при запуске LinuxCNC будет создан файл var с именем rs274ngc.var. Если [RS274NGC]PARAMETER_FILE не включен, могут появиться некоторые сбивающие с толку сообщения.
- Файл таблицы инструментов .tbl, если в INI-файле указано <<sub:ini:sec:emcmot,[EMCMOT]>> TOOL_TABLE. Для некоторых конфигураций таблица инструментов не требуется.

### 4.1.3 Системы шаговых двигателей

#### 4.1.3.1 Base Period

BASE_PERIOD — это *пульс* вашего компьютера с LinuxCNC. footnote: [В этом разделе речь идет об использовании **stepgen**, встроенного в LinuxCNC генератора шаговых импульсов. Некоторые аппаратные устройства имеют собственный генератор шаговых импульсов и не используют встроенный в LinuxCNC. В этом случае обратитесь к руководству по оборудованию.] Каждый период программный генератор шаговых импульсов решает, пришло ли время для следующего шагового импульса. Более короткий период позволит вам генерировать больше импульсов в секунду в определенных пределах. Но если вы сделаете слишком короткий импульс, ваш компьютер будет тратить так много времени на генерацию импульсов шагов, что все остальное замедлится или, возможно, даже заблокируется. Требования к задержке и приводу шагового двигателя влияют на самый короткий период, который вы можете использовать.

В худшем случае задержки могут возникать всего несколько раз в минуту, и вероятность того, что плохая задержка произойдет, когда двигатель меняет направление, невелика. Таким образом,

вы можете получить очень редкие ошибки, которые время от времени портят деталь и которые невозможно устранить.

Самый простой способ избежать этой проблемы — выбрать BASE_PERIOD, который представляет собой сумму наибольшего требования к времени вашего диска и наихудшей задержки вашего компьютера. Это не всегда лучший выбор. Например, если вы используете привод с требованием времени удержания сигнала направления 20 мкс, и ваш тест на задержку показал, что максимальная задержка составляет 11 мкс, то если вы установите BASE_PERIOD на 20+11 = 31 мкс, вы получите не-очень-хорошие 32 258 шагов в секунду в одном режиме и 16 129 шагов в секунду в другом режиме.

Проблема заключается в требованиях к времени удержания 20 мкс. Это плюс задержка в 11 мкс заставляет нас использовать медленный период в 31 мкс. Но программный генератор шагов LinuxCNC имеет некоторые параметры, которые позволяют увеличивать различные времена с одного периода до нескольких. Например, если *steplen* footnote: [steplen относится к параметру, который регулирует производительность встроенного в LinuxCNC генератора шаговых импульсов «stepgen», который является компонентом HAL. Этот параметр регулирует длину самого шагового импульса. Продолжайте читать, со временем все будет объяснено.] изменяется с 1 на 2, тогда между началом и концом шагового импульса будет два периода. Аналогично, если *dirhold* footnote: [dirhold относится к параметру, который регулирует продолжительность времени удержания направления.] изменена с 1 на 3, между шаговым импульсом и изменением контакта направления будет как минимум три периода. .

Если мы сможем использовать *dirhold*, чтобы удовлетворить требованиям времени удержания 20 мкс, то следующим по продолжительности временем будет время высокого уровня 4,5 мкс. Добавьте задержку 11 мкс к высокому времени 4,5 мкс, и вы получите минимальный период 15,5 мкс. Когда вы попробуете 15,5 мкс, вы обнаружите, что компьютер работает медленно, поэтому вы остановитесь на 16 мкс. Если мы оставим *dirhold* равным 1 (по умолчанию), то минимальное время между шагом и направлением составит период 16 мкс минус задержка 11 мкс = 5 мкс, чего недостаточно. Нам нужно еще 15 мкс. Поскольку период равен 16 мкс, нам нужен еще один период. Поэтому мы меняем *dirhold* с 1 на 2. Теперь минимальное время от окончания шагового импульса до вывода изменения направления составляет 5+16=21 мкс, и нам не нужно беспокоиться о том, что привод сделает шаг в неправильном направлении из-за задержки.

Дополнительную информацию о Stepgen см. в разделе stepgen.

#### 4.1.3.2 Параметры шагового импульса

Время импульса и паузы на некоторых приводах различаются. В этом случае точка Step (шага) становится важной. Если привод делает ”шаг” по срезу, тогда выходной контакт должен быть инвертирован.

### 4.1.4 Серво системы

#### 4.1.4.1 Принцип работы

Сервосистемы обладают большей скоростью и точностью, чем эквивалентные шаговые системы, но они более дороги и сложны. В отличие от шаговых систем, сервосистемам требуется какой-то тип устройства обратной связи по положению, и их необходимо отрегулировать или *настроить*, поскольку они не совсем работают сразу после установки, как это могла бы сделать шаговая система. Эти различия существуют потому, что сервоприводы представляют собой систему с *обратной связью*, в отличие от шаговых двигателей, которые обычно работают без *обратной связи*. Что означает с *обратной связью*? Давайте рассмотрим упрощенную схему подключения сервомоторной системы.

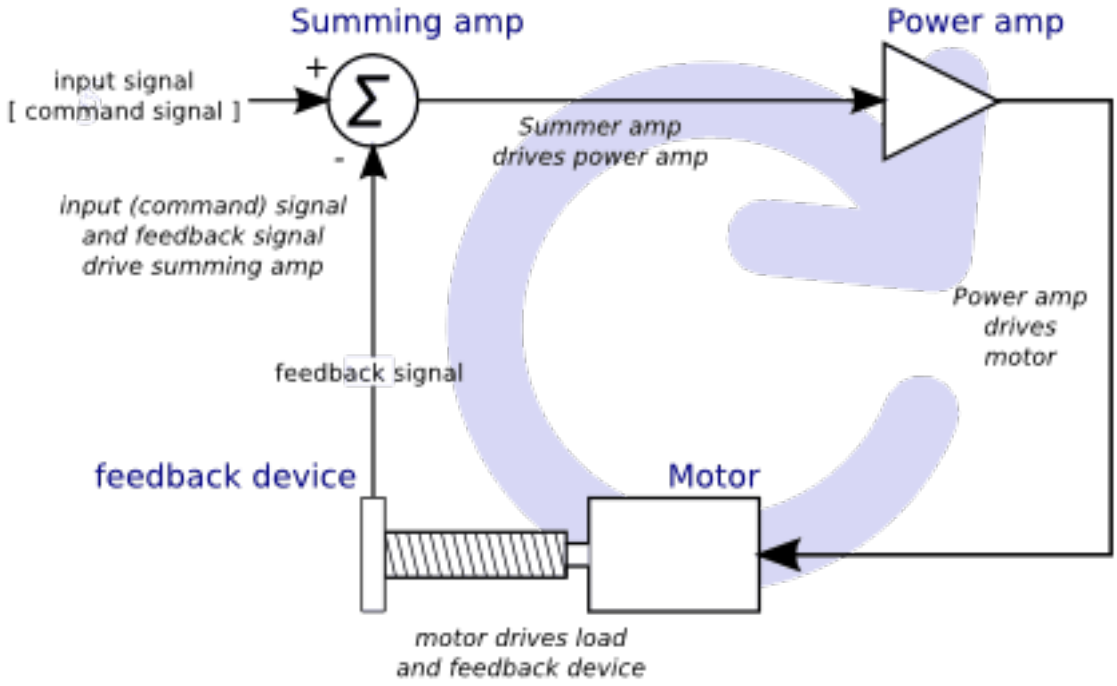



Figure 4.1: Servo Loop

На этой диаграмме показано, что входной сигнал (и сигнал обратной связи) управляет суммирующим усилителем, суммирующий усилитель управляет усилителем мощности, усилитель мощности управляет двигателем, двигатель управляет нагрузкой (и устройством обратной связи) и устройство обратной связи ( и входной сигнал) управляют двигателем. Это очень похоже на круг (обратную связь), где A контролирует B, B контролирует C, C контролирует D, а D контролирует A.

Если вы раньше не работали с сервосистемами, на первый взгляд это, без сомнения, покажется очень странной идеей, особенно по сравнению с более обычными электронными схемами, где входы плавно переходят к выходам и никогда не возвращаются назад. [1] Если *все* контролирует *все остальное*, как это вообще может работать, кто главный? Ответ в том, что LinuxCNC *может* управлять этой системой, но он должен делать это, выбирая один из нескольких методов управления. Метод управления, который использует LinuxCNC, один из самых простых и лучших, называется ПИД.

ПИД означает «пропорциональный», «интегральный» и «производный». Пропорциональное значение определяет реакцию на текущую ошибку, интегральное значение определяет реакцию на основе суммы недавних ошибок, а производное значение определяет реакцию на основе скорости изменения ошибки. Это три распространенных математических метода, которые применяются для достижения заданного значения рабочего процесса. В случае LinuxCNC, процессом, которым мы хотим управлять, является фактическое положение оси, а заданной точкой является заданное положение оси.

[1] Если это поможет, ближайшим эквивалентом этого в цифровом мире являются *конечные автоматы*, *машины последовательностей* и т. д., где то, что выходные данные делают *сейчас*, зависит от того, что входы (и выходы) делали *раньше*. Если это не поможет, тогда неважно.

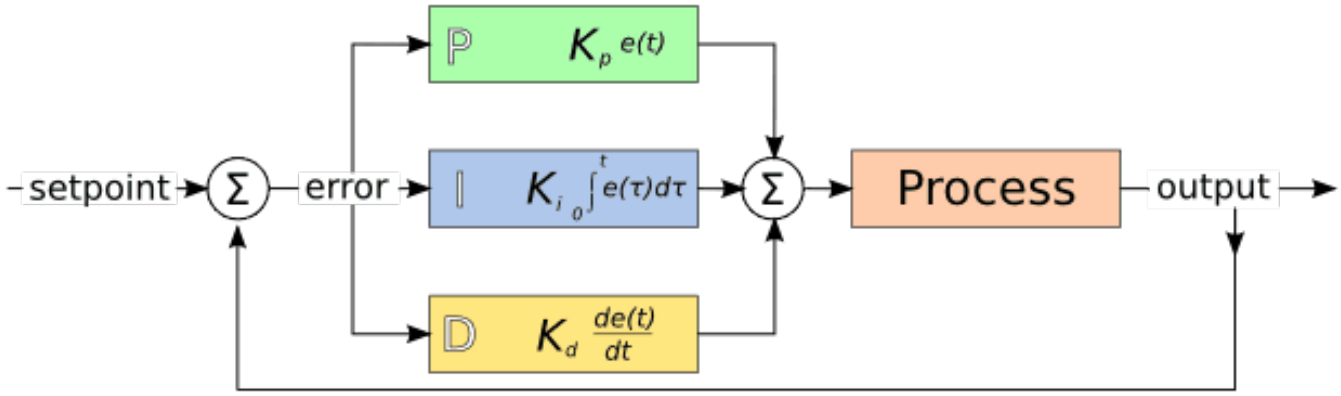



Figure 4.2: ПИД-контур

*Настраивая* три константы в алгоритме ПИД-регулятора, контроллер может обеспечить управляющее действие, разработанное для конкретных требований процесса. Реакцию контроллера можно описать с точки зрения реакции контроллера на ошибку, степени превышения контроллером заданного значения и степени колебаний системы.

#### 4.1.4.2 Пропроциональный член

Пропорциональный член (иногда называемый коэффициентом усиления) вносит изменения в выходной сигнал, пропорциональные текущему значению ошибки. Высокий пропорциональный коэффициент усиления приводит к значительному изменению выходного сигнала при заданном изменении ошибки. Если пропорциональное усиление слишком велико, система может стать нестабильной. Напротив, небольшой коэффициент усиления приводит к небольшому выходному отклику на большую входную ошибку. Если пропорциональное усиление слишком низкое, управляющ воздействие может быть слишком малым при реагировании на возмущения в системе.

При отсутствии помех чисто пропорциональное управление не установится на заданном значении, но сохранит установившуюся ошибку, которая является функцией пропорционального усиления и коэффициента усиления процесса. Несмотря на установившееся смещение, как теория настройки, так и промышленная практика показывают, что именно пропорциональный член должен вносить основной вклад в изменение выхода.

#### 4.1.4.3 Интегральный член

Вклад интегрального члена (иногда называемый сбросом) пропорционален как величине ошибки, так и ее продолжительности. Суммирование мгновенной ошибки по времени (интегрирование ошибки) дает накопленное смещение, которое должно было быть исправлено ранее. Накопленная ошибка затем умножается на интегральный коэффициент усиления и добавляется к выходному сигналу контроллера.

Интегральный член (при добавлении к пропорциональному) ускоряет движение процесса к заданному значению и устраняет остаточную установившуюся ошибку, которая возникает при использовании только пропорционального регулятора. Однако, поскольку интегральная составляющая реагирует на накопленные в прошлом ошибки, это может привести к тому, что текущее значение выйдет за пределы заданного значения (пересечет заданное значение и затем создаст отклонение в другом направлении).

#### 4.1.4.4 Производный член

Скорость изменения ошибки процесса рассчитывается путем определения наклона ошибки во времени (т. е. ее первой производной по времени) и умножения этой скорости изменения на коэффициент усиления производной.

Производный член замедляет скорость изменения выходного сигнала контроллера, и этот эффект наиболее заметен вблизи уставки контроллера. Следовательно, управление по производной используется для уменьшения величины перерегулирования, создаваемого интегральным компонент и улучшения общей стабильности процесса контроллера.

#### 4.1.4.5 Настройка обратной связи

Если параметры ПИД-регулятора (коэффициенты пропорционального, интегрального и производного членов) выбраны неправильно, вход управляемого процесса может быть нестабильным, т. е. его выходной сигнал варируется с осцилляциями или без них и ограничивается только насыщением или механическим прерыванием. Настройка контура управления представляет собой настройку его параметров управления (усиление/зона пропорциональности, интегральное усиление/сброс, производное усиление/скорость) до оптимальных значений для желаемой реакции управления.

#### 4.1.4.6 Ручная настройка

Простой метод настройки заключается в том, чтобы сначала установить значения I и D на ноль. Увеличивайте P до тех пор, пока выходной сигнал контура не начнет колебаться, затем P следует установить равным примерно половине этого значения для отклика типа *затухание на четверть амплитуды*. Затем увеличивайте I до тех пор, пока любое смещение не станет правильным за достаточное для процесса время. Однако слишком много I вызовет нестабильность. Наконец, при необходимости увеличивайте D до тех пор, пока контур не будет достаточно быстро достигать заданного значения после нарушения нагрузки. Однако слишком большое значение D приведет к чрезмерному отклику и перерегулированию. Быстрая настройка контура ПИД-регулятора обычно слегка выходит за пределы диапазона, чтобы быстрее достичь заданного значения; однако некоторые системы не допускают перерегулирования, и в этом случае требуется система ч*резмерного демпфирования* с обратной связью для которой потребуется настройка P, значительно меньшая, чем половина настройки P, вызывающей колебания.

### 4.1.5 S-Curve Trajectory Planning

S-curve trajectory planning limits jerk (the rate of change of acceleration) to provide smoother motion. This can reduce machine vibration and improve surface finish, but requires tuning additional parameters.

#### 4.1.5.1 Enabling

Указывается в INI-файле:

```
[TRAJ]
PLANNER_TYPE = 1              # 0=trapezoidal (default), 1=S-curve
MAX_LINEAR_JERK = 1000.0      # Machine units/s^3

[JOINT_n]
MAX_JERK = 1000.0
```

S-curve planning is only active when `PLANNER_TYPE = 1` and `MAX_LINEAR_JERK > 0`.

**Note**
If MAX_LINEAR_JERK is not specified, it defaults to 1e9 (1 billion), which effectively disables jerk limiting while maintaining S-curve calculations. This produces motion similar to trapezoidal planning but not identical. The maximum allowed value is 1e9 to prevent numerical instability.

#### 4.1.5.2 Tuning

Start with a conservative jerk value and increase gradually:

```
MAX_JERK b''≈b'' 10 to 100 × MAX_ACCELERATION
```

Typical values: 100-100,000 units/s$^3$ depending on machine rigidity and units (mm values are typically 1000x larger than inch values).

Increase MAX_LINEAR_JERK until motion becomes sluggish or following errors increase, then reduce slightly. Test with coordinated moves and arcs.

Values above 1e9 are automatically clamped to 1e9 to avoid numerical issues in the S-curve trajectory calculations.

### 4.1.6 RTAI

Интерфейс приложений реального времени (RTAI) используется для обеспечения наилучшей производительности в реальном времени (RT). Исправленное ядро RTAI позволяет писать приложения со строгими ограничениями по времени. RTAI дает вам возможность использовать такие вещи, как генерация шаговых импульсов, которые требуют точных временных параметров.

#### 4.1.6.1 ACPI

Расширенный интерфейс конфигурации и питания (ACPI) имеет множество различных функций, большинство из которых влияют на производительность RT (например: управление питанием, отключение питания ЦП, масштабирование частоты ЦП и т. д.). В ядре LinuxCNC (и, возможно, во всех ядрах с исправлениями RTAI) ACPI отключен. ACPI также заботится об отключении системы после начала выключения, поэтому вам может потребоваться нажать кнопку питания, чтобы полностью выключить компьютер. Группа RTAI улучшила это в последних выпусках, поэтому ваша система LinuxCNC в конце концов может отключиться сама по себе.

### 4.1.7 Варианты оборудования интерфейса компьютера/станка

#### 4.1.7.1 litehm2/rv901t

Litehm2 — это независимый от платы порт прошивки HostMot2 FPGA. Первой поддерживаемой платой является linsn rv901t, которая изначально была построена как плата контроллера светодиодов, но благодаря наличию ввода-вывода она хорошо подходит для работы в качестве контроллера машины. Она предлагает около 80 портов ввода-вывода с буферизацией 5 В и может переключаться между всеми входами и всеми выходами. его также легко модифицировать, чтобы разделить порты пополам между входом и выходом. Rv901t подключается к компьютеру через Gigabit или 100Mbit Ethernet.

Litehm2 основан на платформе LiteX, которая поддерживает широкий спектр плат FPGA. В настоящее время поддерживается только rv901t, но поддержка большего количества плат находится в стадии разработки.

Дополнительную информацию можно найти по адресу https://github.com/sensille/litehm2.

# 4.2 Тестирование задержки

## 4.2.1 Что такое задержка (latency)?

Задержка — это время, которое требуется ПК, чтобы остановить то, что он делает, и ответить на внешний запрос, например, запустить один из периодических потоков реального времени LinuxCNC. Чем меньше задержка, тем быстрее вы можете запустить потоки реального времени, и тем более плавным будет движение (и потенциально более быстрым в случае программного шагового перемещения).

Задержка гораздо важнее скорости ЦП. Скромный Pentium II, который реагирует на прерывания в течение 10 микросекунд каждый раз, может дать лучшие результаты, чем новейший и самый быстрый монстр P4 Hyperthreading.

Процессор — не единственный фактор, определяющий задержку. Материнские платы, видеокарты, порты USB и ряд других вещей могут повлиять на задержку. Лучший способ узнать, с чем вы имеете дело, — запустить тест на задержку.

Генерация шаговых импульсов в программном обеспечении имеет одно очень большое преимущество - это бесплатно. Почти каждый ПК имеет параллельный порт, который способен выводить шаговые импульсы, генерируемые программным обеспечением.

However, software step pulses also have some disadvantages:

- ограниченная максимальная скорость шага
- джиттер в генерируемых импульсах
- нагружает процессор

## 4.2.2 Тесты задержки

LinuxCNC включает несколько тестов на задержку. Все они выдают эквивалентную информацию. Запуск этих тестов поможет определить, подходит ли компьютер для управления станком с ЧПУ.

**Note**
Не запускайте LinuxCNC или StepConf во время выполнения теста задержки.

### 4.2.2.1 Latency Test

The latency test can be run a few different ways.

If you are using PnCconf to configure your machine, you can launch the Latency Test by clicking the ”Test Base Period Jitter button’ during the 2nd step of the process.

If you are using StepConf to configure your machine, you can launch the Latency Test by clicking the ”Test Base Period Jitter button’ during the 2nd step of the process.

If you want to run the test from the command line, open a terminal window (in Ubuntu, from Applications → Accessories → Terminal) and run the following command:

```
latency-test
```

Это запустит тест задержки с периодом базового потока 25мкс и периодом сервопотока 1мс. Длительность периода можно указать в командной строке:

```
latency-test 50000 1000000
```

Это запустит тест задержки с периодом базового потока 50мкс и периодом сервопотока 1мс.

Для получения доступных параметров введите в командной строке:

```
latency-test -h
```

После запуска теста задержки вы должны увидеть что-то вроде этого:

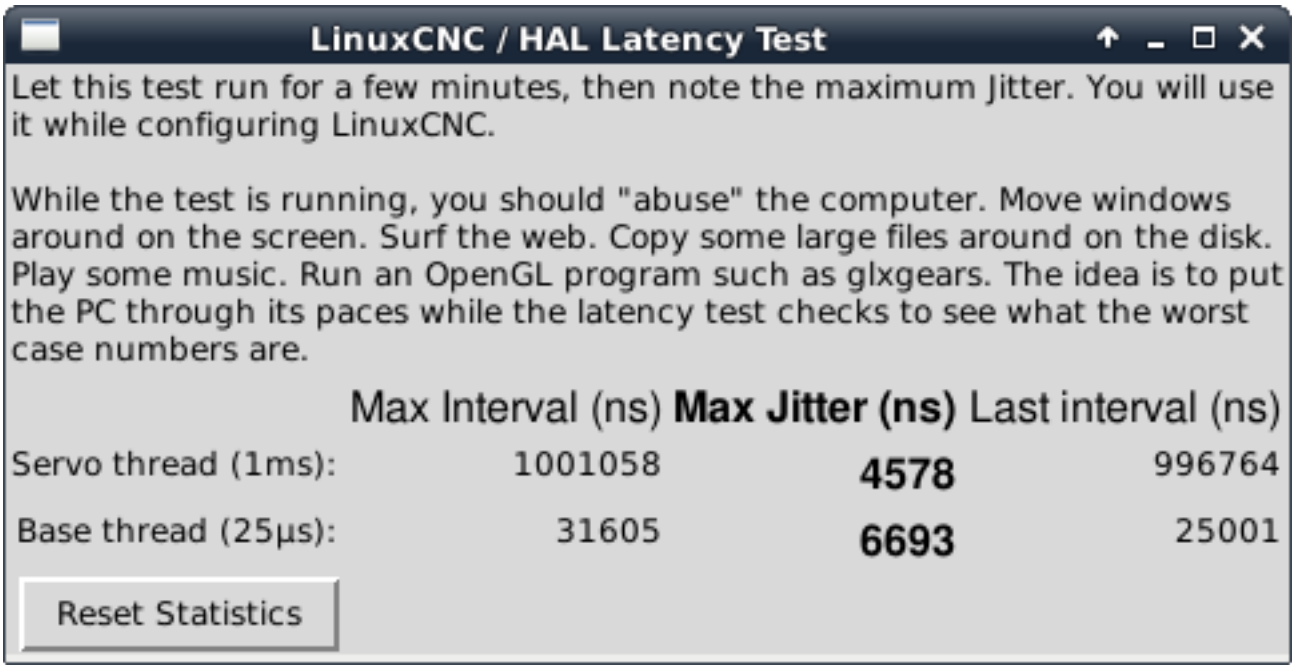



Figure 4.3: Тест задержки HAL

Во время выполнения теста вам следует *издеваться* над компьютером. Перемещать окна по экрану. Полазить по веб-страницам. Скопировать несколько больших файлов на диск. Воспроизвести музыку. Запустить программу OpenGL, например glxgears. Идея состоит в том, чтобы подвергнуть ПК испытаниям, пока тест на задержку проверяет, каковы наихудшие показатели.

Важным числом для программного степпинга является *максимальный джиттер* базового потока. В приведенном выше примере это 6693 наносекунды (нс) или 6,693 микросекунды (мкс). Запишите это число и введите его в StepConf, когда оно будет запрошено.

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 µs latency. Fortunately that was fixable, see [?].

So, what do the results mean?

If your Max Jitter number is less than about 20,000 nanoseconds, the computer should give very nice results with software stepping or a dedicated hardware card such as a Mesa *Anything I/O* card.

If the Max Jitter number is between 20,000 and 50,000 nanoseconds, you can still get good results with software stepping, but your maximum step rate might be a little disappointing, especially if you use microstepping or have very fine pitch leadscrews. You can, however, achieve excellent results using a hardware card.

If the Max Jitter number is between 50,000 and 500,000 nanoseconds, you cannot use software stepping. You can, however, achieve acceptable results using a hardware card.

If the Max Jitter number is above 500,000 nanoseconds, you cannot use software stepping or a hardware card with LinuxCNC and achieve acceptable results.

**Note**
If you get high numbers, there may be ways to improve them. Another PC had very bad latency (several million nanoseconds) when using the onboard video. But a $5 used video card solved the problem. LinuxCNC does not require bleeding-edge hardware.

Для получения дополнительной информации о настройке шагового двигателя см. главу Stepper Tuning.

**Tip**
Доступны дополнительные инструменты командной строки для проверки задержки, когда LinuxCNC не запущен.

#### 4.2.2.2 График задержки

latency-plot создает ленточную диаграмму для базового и сервопотока. Может быть полезно увидеть пики задержки при запуске или использовании других приложений. Использование:

```
latency-plot --help

Usage:
      latency-plot --help | -?
      latency-plot --hal [Options]

Options:
      --base ns  (base  thread interval in nanoseconds, default:   25000)
      --servo ns (servo thread interval in nanoseconds, default: 1000000)
      --time ms  (report interval in milliseconds, default: 1000)
      --relative (relative clock time (default))
      --actual   (actual clock time)
```

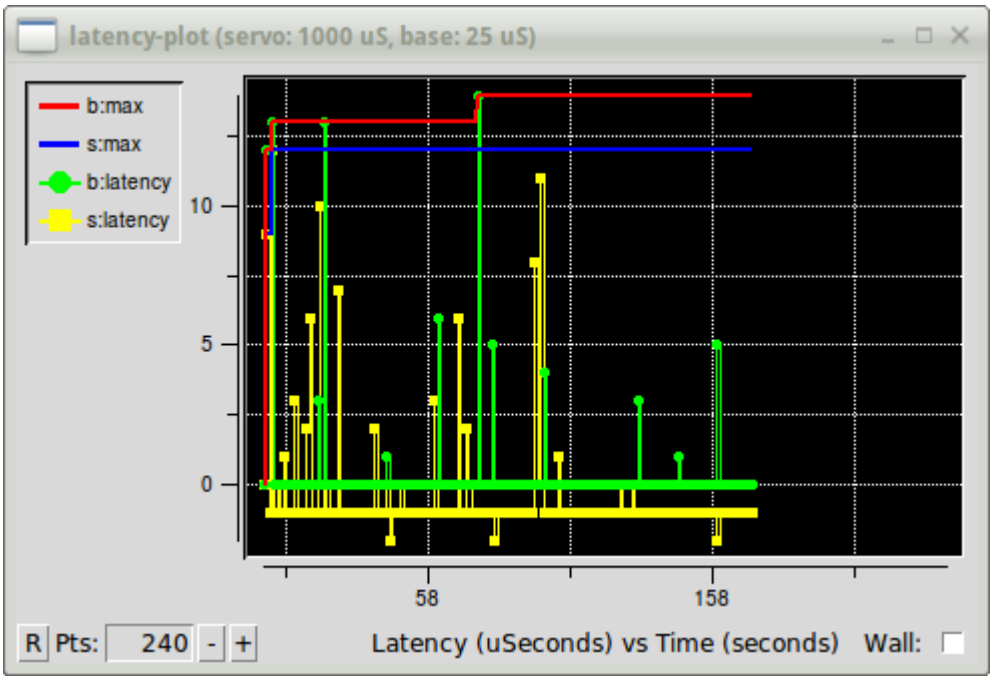



Figure 4.4: `latency-plot` Window

#### 4.2.2.3 Гистограмма задержки

Гистограмма задержки приложения отображает гистограмму задержки (jitter) для базового и сервопотока.

```
Usage:
   latency-histogram --help | -?
   latency-histogram [Options]

Options:
  --base      ns   (base  thread interval in nanoseconds, default:   25000, min:  5000)
  --servo     ns   (servo thread interval in nanoseconds, default: 1000000, min: 25000)
  --bbinsize  ns   (base  bin size in nanoseconds, default: 100
  --sbinsize  ns   (servo bin size in nanoseconds, default: 100
  --bbins     n    (base  bins, default: 200
  --sbins     n    (servo bins, default: 200
  --logscale  0|1  (y axis log scale, default: 1)
  --text      note (additional note, default: "" )
  --show           (show count of undisplayed bins)
  --nobase         (servo thread only)
  --verbose        (progress and debug)
  --nox            (no gui, display elapsed,min,max,sdev for each thread)
```

---

**Note**
When determining the latency, 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 the +/- bin range are reported with special end bars. Use `--show` to show count for the off-chart [pos|neg] bin.

---

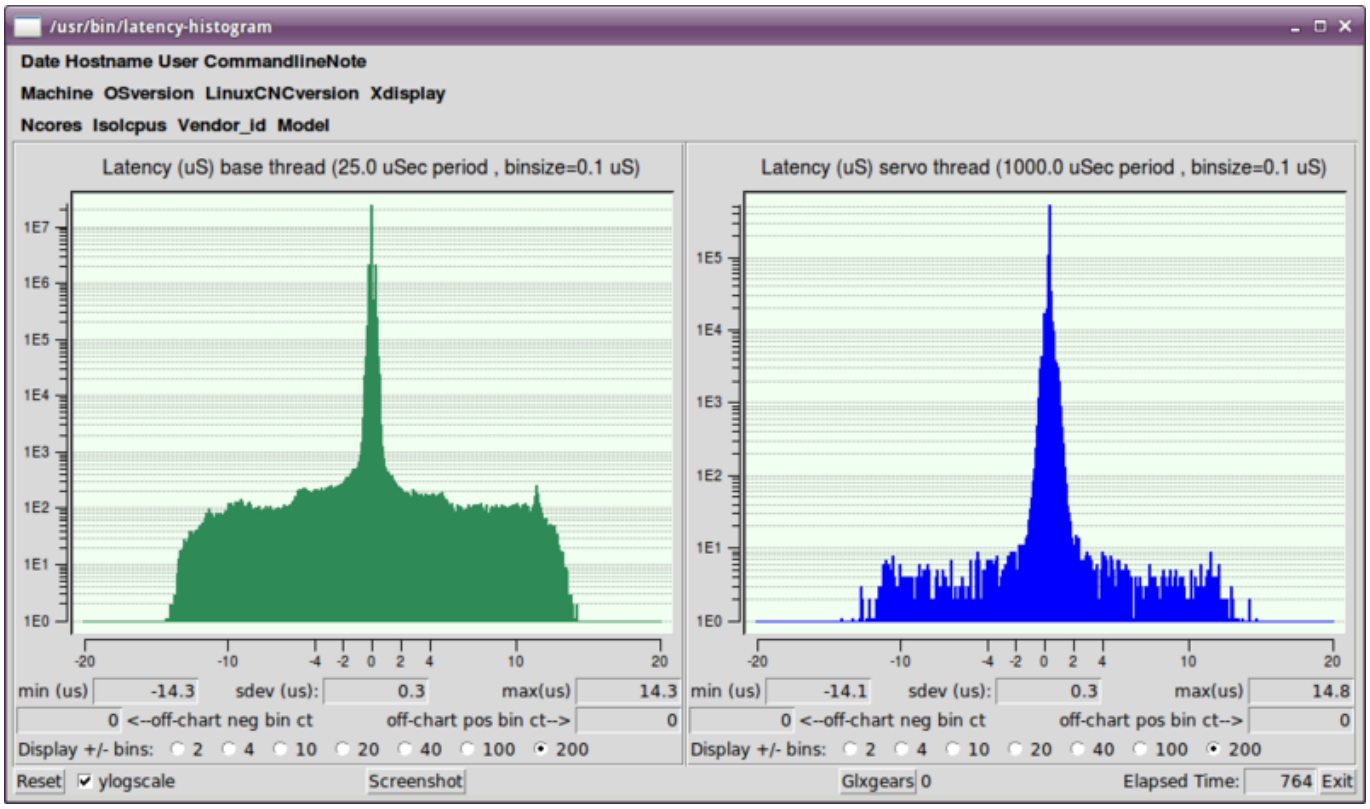



Figure 4.5: Окно `latency-histogram`

### 4.2.3 Настройка задержки

LinuxCNC может работать на многих различных аппаратных платформах и с различными ядрами реального времени, и все они могут выиграть от настройки на оптимальную задержку.

Основной целью настройки системы для LinuxCNC является резервирование ЦП для исключительного использования задач реального времени LinuxCNC, чтобы другие задачи (как пользовательские программы, так и потоки ядра) не мешали доступу LinuxCNC к этому ЦП.

Когда определенные параметры настройки считаются универсально полезными, LinuxCNC выполняет эту настройку автоматически при запуске, но многие параметры настройки зависят от станка и не могут быть выполнены автоматически. Человеку, устанавливающему LinuxCNC, необходимо экспериментально определить оптимальную настройку для своей системы.

#### 4.2.3.1 Настройка BIOS на задержку

BIOS ПК существенно различаются по своим показателям задержки.

Настройка BIOS утомительна, поскольку вам нужно перезагрузить компьютер, сделать одну небольшую настройку в BIOS, загрузить Linux и запустить тест задержки (потенциально в течение длительного времени), чтобы увидеть, как повлияло изменение BIOS. Затем повторите для всех остальных настроек BIOS, которые вы хотите попробовать.

Поскольку все BIOS разные и нестандартные, предоставление подробного руководства по настройке BIOS нецелесообразно. В общем, вот некоторые вещи, которые можно попробовать настроить в BIOS:

- Отключите ACPI, APM и любые другие функции энергосбережения. Сюда входит все, что связано с энергосбережением, приостановкой, состояниями сна ЦП, масштабированием частоты ЦП и т. д.
- Отключить режим ”турбо” процессора.
- Отключить гиперпоточность ЦП.
- Отключить (или иным образом контролировать) прерывание управления системой (SMI).
- Отключите все оборудование, которое вы не собираетесь использовать.

#### 4.2.3.2 Настройка Preempt-RT для задержки

Ядро Preempt-RT может выиграть от настройки, чтобы обеспечить наилучшую задержку для LinuxCNC. Настройка может быть выполнена через командную строку ядра, sysctl и через файлы в `/proc` и `/sys`.

Некоторые параметры настройки, на которые следует обратить внимание:

**Командная строка ядра**
Подробности здесь: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

- `isolcpus`: запрещает большинству процессов, не относящихся к LinuxCNC, использовать эти процессоры, оставляя больше процессорного времени доступным для LinuxCNC.
- `irqaffinity`: Выберите, какие ЦП обслуживают прерывания, чтобы ЦП, зарезервированные для LinuxCNC realtime, не приходилось выполнять эту задачу.
- `rcu_nocbs`: запретить запуск обратных вызовов RCU на этих процессорах.
- `rcu_nocb_poll`: Опрос обратных вызовов RCU вместо использования режима сна/пробуждения.
- `nohz_full`: Отключить тактовую частоту на этих процессорах.

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

- `sysctl.kernel.sched_rt_runtime_us`: установите значение -1, чтобы снять ограничение на время, которое могут использовать задачи реального времени.

## 4.3 Stepper Tuning

### 4.3.1 Getting the most out of Software Stepping

Generating step pulses in software has one very big advantage - it’s free. Just about every PC has a parallel port that is capable of outputting step pulses that are generated by the software. However, software step pulses also have some disadvantages:

- ограниченная максимальная скорость шага
- джиттер в генерируемых импульсах
- нагружает процессор

This chapter has some steps that can help you get the best results from software generated steps.

#### 4.3.1.1 Run a Latency Test

The CPU is not the only factor determining latency. Motherboards, graphics cards, USB ports and many other things can degrade it. The best way to know what to expect from a PC is to run the RT latency tests.

Run the latency test as described in the Latency Test chapter.

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.

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

So, what do the results mean? If your *Max Jitter* number is less than about 15-20 microseconds (15000-20000 nanoseconds), the computer should give very nice results with software stepping. If the max latency is more like 30-50 microseconds, 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 nanoseconds), then the PC is not a good candidate for software stepping. Numbers over 1 millisecond (1,000,000 nanoseconds) mean the PC is not a good candidate for LinuxCNC, regardless of whether you use software stepping or not.

Note that if you get high numbers, there may be ways to improve them. For example, one 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.

#### 4.3.1.2 Figure out what your drives expect

Different brands of stepper drives have different timing requirements on their step and direction inputs. So you need to dig out (or Google for) the data sheet that has your drive's specs.

From the Gecko G202 manual:

```
Step Frequency: 0 to 200 kHz
Step Pulse ”0” Time: 0.5 µs min (Step on falling edge)
Step Pulse ”1” Time: 4.5 µs min
Direction Setup: 1 µs min (20 µs min hold time after Step edge)
```

From the Gecko G203V manual:

```
Step Frequency: 0 to 333 kHz
Step Pulse ”0” Time: 2.0 µs min (Step on rising edge)
Step Pulse ”1” Time: 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
```

From the Xylotex datasheet:

```
Minimum DIR setup time before rising edge of STEP Pulse 200 ns Minimum
DIR hold time after rising edge of STEP pulse 200 ns
Minimum STEP pulse high time 2.0 µs
Minimum STEP pulse low time 1.0 µs
Step happens on rising edge
```

Once you find the numbers, write them down too - you need them in the next step.

#### 4.3.1.3 Choose your BASE_PERIOD

BASE_PERIOD is the *heartbeat* of your LinuxCNC computer. Every period, the software step generator decides if it is time for another step pulse. A shorter period will allow you to generate more pulses per second, within limits. But if you go too short, your computer will spend so much time generating step pulses that everything else will slow to a crawl, or maybe even lock up. Latency and stepper drive requirements affect the shortest period you can use, as we will see in a minute.

Let's look at the Gecko example first. The G202 can handle step pulses that go low for 0.5 µs and high for 4.5 µs, it needs the direction pin to be stable 1 µs before the falling edge, and remain stable for 20 µs after the falling edge. The longest timing requirement is the 20 µs hold time. A simple approach would be to set the period at 20 µs. That means that all changes on the STEP and DIR lines are separated by 20 µs. All is good, right?

Wrong! If there was ZERO latency, then all edges would be separated by 20 µs, and everything would be fine. But all computers have some latency. Latency means lateness. If the computer has 11 µs of latency, that means sometimes the software runs as much as 11 µs later than it was supposed to. If one run of the software is 11 µs late, and the next one is on time, the delay from the first to the second is only 9 µs. If the first one generated a step pulse, and the second one changed the direction bit, you just violated the 20 µs G202 hold time requirement. That means your drive might have taken a step in the wrong direction, and your part will be the wrong size.

The really nasty part about this problem is that it can be very very rare. Worst case latencies might only happen a few times a minute, and the odds of bad latency happening just as the motor is changing direction are low. So you get very rare errors that ruin a part every once in a while and are impossible to troubleshoot.

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. If you are running a Gecko with a 20 µs 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 (31000 nano-seconds in the ini file), you are guaranteed to meet the drive's timing requirements.

But there is a tradeoff. Making a step pulse requires at least two periods. One to start the pulse, and one to end it. Since the period is 31 µs, it takes 2x31 = 62 µs to create a step pulse. That means the maximum step rate is only 16,129 steps per second. Not so good. (But don't give up yet, we still have some tweaking to do in the next section.)

For the Xylotex, the setup and hold times are very short, 200 ns each (0.2 µs). The longest time is the 2 µs high time. If you have 11 µs latency, then you can set the BASE_PERIOD as low as 11+2=13 µs. Getting rid of the long 20 µs hold time really helps! With a period of 13 µs, a complete step takes 2x13 = 26 µs, and the maximum step rate is 38,461 steps per second!

But you can't start celebrating yet. Note that 13 µs is a very short period. If you try to run the step generator every 13 µs, there might not be enough time left to run anything else, and your computer will lock up. If you are aiming for periods of less than 25 µs, you should start at 25 µs or more, run LinuxCNC, and see how things respond. If all is well, you can gradually decrease the period. If the mouse pointer starts getting sluggish, and everything else on the PC slows down, your period is a little too short. Go back to the previous value that let the computer run smoothly.

In this case, suppose you started at 25 µs, trying to get to 13 µs, but you find that around 16 µs is the limit - any less and the computer doesn't respond very well. So you use 16 µs. With a 16 µs period and

11 µs latency, the shortest output time will be 16-11 = 5 µs. The drive only needs 2 µs, so you have some margin. Margin is good - you don't want to lose steps because you cut the timing too close.

What is the maximum step rate? Remember, two periods to make a step. You settled on 16 µs for the period, so a step takes 32 µs. That works out to a not bad 31,250 steps per second.

#### 4.3.1.4 Use steplen, stepspace, dirsetup, and/or dirhold

In the last section, we got the Xylotex drive to a 16 µs period and a 31,250 step per second maximum speed. But the Gecko was stuck at 31 µs and a not-so-nice 16,129 steps per second. The Xylotex example is as good as we can make it. But the Gecko can be improved.

The problem with the G202 is 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 time from one period to several. For example, if steplen is changed from 1 to 2, then it there will be two periods between the beginning and end of the step pulse. Likewise, if dirhold 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 Gecko stepping the wrong direction because of latency.

If the computer has a latency of 11 µs, then a combination of a 16 µs base period, and a dirhold value of 2 ensures that we will always meet the timing requirements of the Gecko. For normal stepping (no direction change), the increased dirhold value has no effect. It takes two periods totalling 32 µs to make each step, and we have the same 31,250 step per second rate that we got with the Xylotex.

The 11 µs latency number used in this example is very good. If you work through these examples with larger latency, like 20 or 25 µs, the top step rate for both the Xylotex and the Gecko will be lower. But the same formulas apply for calculating the optimum BASE_PERIOD, and for tweaking dirhold or other step generator parameters.

#### 4.3.1.5 No Guessing!

For a fast AND reliable software based stepper system, you cannot just guess at periods and other configuration parameters. You need to make measurements on your computer, and do the math to ensure that your drives get the signals they need.

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.

I've added a few things to the spreadsheet to calculate max speed and stepper electrical calculations.

## 4.4 INI Configuration

### 4.4.1 Компоненты INI-файла

Типичный INI-файл имеет довольно простую структуру, которая включает в себя:

- комментарии
- разделы
- переменные

Каждый из этих элементов разделен на отдельные строки. Каждый конец строки или символ новой строки создает новый элемент.

#### 4.4.1.1 Comments

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

```
; This is my mill configuration file.
# I set it up on January 12, 2012
```

Комментарии также можно использовать для *отключения* переменной. Это облегчает выбор между различными переменными.

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

В этом списке переменной DISPLAY будет присвоено значение axis, поскольку другая закомментирова
Если кто-то неосторожно отредактирует такой список и оставит две строки без комментариев, будет использована первая встреченная.

Note that inside a variable's value, the "#" and ";" characters are still comments. You should use double or single quoted strings if you need to embed # or ; characters:

```
# Below results in INCORRECT=value
# because comments are stripped
INCORRECT = value # and a comment

# Correct embedding
CORRECT = "value # and an embedded # char"
```

#### 4.4.1.2 Разделы

Related parts of an INI file are separated into sections. A section name is enclosed in brackets like this: `[THIS_SECTION]`. The order of sections is unimportant. Sections begin at the section name and end at the next section name. Section identifiers are case senstive and can only contain letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, section identifiers cannot start with a digit.

Следующие разделы используются LinuxCNC:

- [EMC] общие сведения
- [DISPLAY] настройки, связанные с графическим интерфейсом пользователя
- [FILTER] настройки программ входного фильтра
- [RS274NGC] настройки, используемые интерпретатором G-кода
- [EMCMOT] настройки, используемые контроллером движения в реальном времени
- [TASK] настройки, используемые контроллером задач

- [HAL] указывает файлы .hal
- [HALUI] Команды MDI, используемые HALUI
- [APPLICATIONS] Другие приложения, запускаемые LinuxCNC
- [TRAJ] дополнительные настройки, используемые контроллером движения в реальном времени
- [JOINT_n] переменные отдельного сочленения
- [AXIS_l] переменные отдельной оси
- [KINS] кинематические переменные
- [EMCIO] настройки, используемые контроллером ввода-вывода

#### 4.4.1.3 Переменные

A variable line is made up of a variable name, an equals sign (=), and a value. A variable identifier is case sensitive and may only consist of letters A-Z, a-z, digits 0-9 and underscore (_). Additionally, variable identifiers cannot start with a digit. White space at the beginning of the line and after the variable identifier, up to the equals sign, is ignored.

**Пример переменной**

```
MACHINE = My Machine
```

A variable line may be extended to multiple lines with a terminal backslash (\) character. There may be white space following the trailing backslash character, but this is strongly discouraged.

Идентификаторы разделов не могут быть расширены на несколько строк.

**Пример переменной с расширенной строкой**

```
APP = sim_pin \
ini.0.max_acceleration \
ini.1.max_acceleration \
ini.2.max_acceleration \
ini.0.max_velocity \
ini.1.max_velocity \
ini.2.max_velocity
```

A specific variable in a specific sections is often denoted in the documentation as [SECTION]VARIABLE. This specification mirrors the same way they are specified in HAL files for expansion.

Variable values may be quoted to ensure proper space and special character embedding in literal or escaped forms. Both single and double quoted values are allowed. Mutiple quoted value segments are merged into one value. Embedding quotes in quoted values must use the backslash \\ character to escape the embedded quote if it is the same kind as the enclosing quotes. Double quotes values also support all common escape formats and full Unicode:

- control: \\[abfnrtv]
- octal: \\[0-2][0-7]{0,2}
-
-
-

The resulting value is always converted into UTF-8 and checked for validity. It is not allowed to embed NUL characters either literally or by using an escape.

**Value Quote and Escape Example**

```
STRING = ”Hello World!”
STRING = ’Hello World Too!’

# The following would become: Hello World!
STRING = ”Hello” \
” ” \
’World’ \
”!”

EMBED = ”Literal single ’ in double”
EMBED = ’Literal double ” in single’
EMBED = ”Literal double \” in double”
EMBED = ’Literal single \’ in single’

SMILE = ”\\370\\237\\230\\200 = b’’□b’’”
SMILE = ”\\xf0\\x9f\\x98\\x80 = b’’□b’’”
SMILE = ”\\ud83d\\ude00 = b’’□b’’”
SMILE = ”\\U0001f600 = b’’□b’’”
```

Variables’ value can have types associated when they are read by LinuxCNC. The types are enforced by the INI file reader when the appropriate function calls are performed. All values may always be read as *string*. Conversion into other types has restrictions. Possible types are:

- *string* - the value is taken as-is
- *boolean* - only boolean words are accepted
- *signed integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *unsigned integer* - whole numbers in decimal, hexadecimal, octal or binary bases
- *real* - floating point values (always using decimal point)
- *enumeration* - a restricted set of keys to represent a value or function

Signed and unsigned integers are read and converted as 64-bit numbers when marked as `s64` and `u64`. Plain old 32-bit integers are marked `int` and are always signed. The default number base is decimal. Alternative bases may be specified using a prefix. The complete list of valid integer numbers:

- \[0-9]+ - decimal
- 0x\[0-9A-Fa-f]+ - hexadecimal
- 0o\[0-7]+ - octal
- 0b\[01]+ - binary

Signed integers may be preceded by a plus (+) or minus (-) sign, regardless number base. Unsigned integers will generate a warning if they are preceded by a minus (-) sign upon conversion.

Boolean values are case insensitive and allow the following words:

- true/enabled - `TRUE`, `YES` `ON` or `1`
- false/disabled - `FALSE`, `NO` `OFF` or `0`

Enumerations are a set of keywords defined by LinuxCNC and interpreted to mean a setting, value or functionality. The exact values are declared in the individual variable description and the variables are marked with `enum`. Most enumerations are interpreted without case. It is noted in the variable description if case matters.

Some LinuxCNC variables are special in that their format is interpreted as a multi-valued entry. These variables have an appropriate description below of the format expected.

Variables that are used by LinuxCNC must always use the section names and variable names as shown. Any custom or private variables and sections are up to the user.

#### 4.4.1.4 Пользовательские разделы и переменные

В большинстве примеров конфигураций используются настраиваемые разделы и переменные, позволяющие для удобства разместить все настройки в одном месте.

Чтобы добавить пользовательскую переменную в существующий раздел LinuxCNC, просто включите переменную в этот раздел.

**Пример пользовательской переменной: присвоение значения LINEAR переменной TYPE и значения 16000 переменной SCALE.**

```
[JOINT_0]
TYPE = LINEAR
...
SCALE = 16000
```

Чтобы ввести пользовательский раздел с собственными переменными, добавьте раздел и переменные в INI-файл.

**Пример пользовательского раздела**

```
[PROBE]
Z_FEEDRATE = 50
Z_OFFSET = 12
Z_SAFE_DISTANCE = -10
```

Чтобы использовать пользовательские переменные в файле HAL, поместите раздел и имя переменной вместо значения.

**HAL Пример**

```
setp offset.1.offset [PROBE]Z_OFFSET
setp stepgen.0.position-scale [JOINT_0]SCALE
```

---

**Note**
Значение, хранящееся в переменной, должно соответствовать типу, указанному контактом компонента.

---

Чтобы использовать пользовательские переменные в G-коде, используйте синтаксис глобальных переменных `#<_ini[section]variable>`. В следующем примере показана простая процедура касания оси Z для фрезерного станка с использованием зондовой пластины.

Please note that G-code embedded ini variables are converted to upper case before they are searched in the INI file. The `#<_ini[section]variable>` should be called [SECTION]VARIABLE in the INI file.

**Пример G-code**

```
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 Включить файлы

Файл INI может включать содержимое другого файла с помощью директивы #INCLUDE.

**#INCLUDE Формат**

```
#INCLUDE filename
```

Имя файла может быть указано как:

- файл в том же каталоге, что и файл INI
- файл, расположенный относительно рабочего каталога
- файл, расположенный относительно рабочего каталога
- a user-home-relative file name (starts with a ~/)

Поддерживается несколько директив #INCLUDE.

**#INCLUDE Примеры**

```
#INCLUDE joint_0.inc
#INCLUDE ../parallel/joint_1.inc
#INCLUDE below/joint_2.inc
#INCLUDE /home/myusername/myincludes/display.inc
#INCLUDE ~/linuxcnc/myincludes/rs274ngc.inc
```

The #INCLUDE directives are supported up to 16 levels — an included file may include additional files up to 16 levels deep. Recursive inclusion of files is detected and flagged as an error. The recommended file extension is *.inc*. Do *not* use a file extension of *.ini* for included files.

### 4.4.2 Разделы INI-файла

Variables in each section have an associated type as denoted in parentheses after the variable.

#### 4.4.2.1 [EMC] раздел

- `VERSION = 1.1` - (string) The format version of this configuration. Any value other than 1.1 will cause the configuration checker to run and try to update the configuration to the new style joint axes type of configuration.
- `MACHINE = My Controller` - (string) This is the name of the controller, which is printed out at the top of most graphical interfaces. You can put whatever you want here as long as you make it a single line long.
- `DEBUG = 0` - (u64) Debug level 0 means no messages will be printed when LinuxCNC is run from a terminal. Debug flags are usually only useful to developers. See src/emc/nml_intf/debugflags.h for other settings.
- `RCS_DEBUG = 1` (u64) RCS debug messages to show. Print only errors (1) by default if EMC_DEBUG_RCS and EMC_DEBUG_RCS bits in `DEBUG` are unset, otherwise print all (-1). Use this to select RCS debug messages. See src/libnml/rcs/rcs_print.hh for all MODE flags.
- `RCS_DEBUG_DEST = STDOUT` - (enum) how to output RCS_DEBUG messages (NULL, STDOUT, STDERR, FILE, LOGGER, MSGBOX).
- `RCS_MAX_ERR = -1` - (int) Number after which RCS errors are not reported anymore (-1 = infinite).
- `NML_FILE = /usr/share/linuxcnc/linuxcnc.nml` - (string) Set this if you want to use a non-default NML configuration file.

#### 4.4.2.2 [DISPLAY] Section

Различные программы пользовательского интерфейса используют разные параметры, и не каждый параметр поддерживается каждым пользовательским интерфейсом. Существует несколько интерфейс таких как AXIS, GMOCCAPY, Touchy, QtVCP's QtDragon и Gscreen. AXIS — это интерфейс для использования с обычным компьютером и монитором, Touchy — для использования с сенсорными экранами. GMOCCAPY можно использовать обоими способами, а также предлагает множество соединений для управления оборудованием. Описания интерфейсов находятся в разделе Интерфейсы Руководства пользователя.

- `DISPLAY = axis` - (string) The file name of the executable providing the user interface to use. Prominent valid options are (all in lower case): `axis`, `touchy`, `gmoccapy`, `gscreen`, `tklinuxcnc`, `qtvcp`, `qtvcp qtdragon` or `qtvcp qtplasmac`.
- `POSITION_OFFSET = RELATIVE` - (enum) The coordinate system (`RELATIVE` or `MACHINE`) to show on the DRO when the user interface starts. The RELATIVE coordinate system reflects the G92 and G5*x* coordinate offsets currently in effect.
- `POSITION_FEEDBACK = COMMANDED` - (enum) The coordinate value (`COMMANDED` or `ACTUAL`) to show on the DRO when the user interface starts. In AXIS this can be changed from the View menu. The COMMANDED position is the position requested by LinuxCNC. The ACTUAL position is the feedback position of the motors if they have feedback like most servo systems. Typically the COMMANDED value is used.
- `DRO_FORMAT_MM = %+08.6f` - (string) Override the default DRO formatting in metric mode (normally 3 decimal places, padded with spaces to 6 digits to the left). The example above will pad with zeros, display 6 decimal digits and force display of a + sign for positive numbers. Formatting follows Python practice: https://docs.python.org/2/library/string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `DRO_FORMAT_IN = % 4.1f` - (string) Override the default DRO formatting in imperial mode (normally 4 decimal places, padded with spaces to 6 digits to the left). The example above will display only one decimal digit. Formatting follows Python practice: https://docs.python.org/2/library/-string.html#format-specification-mini-language . An error will be raised if the format can not accept a floating-point value.
- `CONE_BASESIZE = .25` - (real) Override the default cone/tool base size of .5 in the graphics display. Valid values are between 0.025 and 2.0.
- `DISABLE_CONE_SCALING = TRUE` - (bool) Any non-empty value (including "0") will override the default behavior of scaling the cone/tool size using the extents of the currently loaded G-code program in the graphics display.
- `MAX_FEED_OVERRIDE = 1.2` - (real) The maximum feed override the user may select. 1.2 means 120% of the programmed feed rate.
- `MIN_SPINDLE_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.)
- `MIN_SPINDLE_0_OVERRIDE = 0.5` - (real) The minimum spindle override the user may select. 0.5 means 50% of the programmed spindle speed. (This is used to set the minimum spindle speed.) On multi spindle machine there will be entries for each spindle number. Only used by the QtVCP based user interfaces.
- `MAX_SPINDLE_OVERRIDE = 1.0` - (real) The maximum spindle override the user may select. 1.0 means 100% of the programmed spindle speed.
- `MAX_SPINDLE_0_OVERRIDE = 1.0` - (real) The maximum 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 - Число оборотов шпинделя по умолчанию, когда шпиндель запускается в ручном режиме. Если этот параметр отсутствует, по умолчанию он равен 1 об/мин для AXIS и 300 об/мин для GMOCCAPY.

  - *deprecated* - вместо этого используйте раздел [SPINDLE_n]

- DEFAULT_SPINDLE_0_SPEED = 100 - Число оборотов шпинделя по умолчанию, когда шпиндель запускается в ручном режиме. На многошпиндельном станке будут записи для каждого номера шпинделя. Используется только пользовательскими интерфейсами на основе QtVCP.

  - *deprecated* - вместо этого используйте раздел [SPINDLE_n].

- SPINDLE_INCREMENT = 200 - Приращение, используемое при нажатии кнопок увеличения/уменьшени Используется только пользовательскими интерфейсами на основе QtVCP.

  - *deprecated* - вместо этого используйте раздел [SPINDLE_n].

- MIN_SPINDLE_0_SPEED = 1000 - Минимальная частота вращения, которую можно выбрать вручную. На многошпиндельном станке будут записи для каждого номера шпинделя. Используется только пользовательскими интерфейсами на основе QtVCP.

  - *deprecated* - вместо этого используйте раздел [SPINDLE_n].

- MAX_SPINDLE_0_SPEED = 20000 - Максимальное число оборотов в минуту, которое можно выбрать вручную. На многошпиндельном станке будут записи для каждого номера шпинделя. Используется только пользовательскими интерфейсами на основе QtVCP.

  - *deprecated* - вместо этого используйте раздел [SPINDLE_n].

- PROGRAM_PREFIX = ~/linuxcnc/nc_files - Каталог по умолчанию для файлов G-кода, именованных подпрограмм и определяемых пользователем M-кодов. Каталог PROGRAM_PREFIX ищется перед каталогами, перечисленными в [RS274]SUBROUTINE_PATH и [RS274]USER_M_PATH.

- INTRO_GRAPHIC = emc2.gif - Изображение, показанное на заставке.

- INTRO_TIME = 5 - Максимальное время отображения заставки в секундах.

- CYCLE_TIME = 100 - Время цикла ГИП. В зависимости от экрана это может быть в секундах или мс (предпочтительно мс). Зачастую это частота обновления, а не время ожидания между обновлениями. Если время обновления установлено неправильно, экран может перестать отвечать на запросы или сильно дергаться. Значение 100 мс (0,1 с) является общепринятой настройкой, хотя можно использовать диапазон 50–200 мс (0,05–0,2 с). Процессор с недостаточной мощностью может работать лучше при более длительной настройке. Обычно подходит значение по умолчанию.

- PREVIEW_TIMEOUT = 5 - Таймаут (в секундах) для загрузки графического предварительного просмотра G-кода. На данный момент только в AXIS.

- HOMING_PROMPT = TRUE - Any non-empty value (including ”0”) will enable showing a prompt message with homing request, when the Power On button is pressed in AXIS GUI. Pressing the ”Ok” button in prompt message is equivalent to pressing the ”Home All” button(or the Ctrl-HOME key).

- FOAM_W = 1.5 sets the foam W height.

- FOAM_Z = 0 sets the foam Z height.

- GRAPHICAL_MAX_FILE_SIZE = 20 largest size (in mega bytes) that will be displayed graphically. If the program is bigger than this setting, a bounding box will be displayed. By default, this setting is at 20 MB or 1/4 of the system memory, which ever is smaller. A negative value is interpreted as unlimited.

**Note**
Следующие элементы [DISPLAY] используются GladeVCP и PyVCP, дополнительную информацию см. в разделе embedding a tab главы GladeVCP или в главе PyVCP.

- `EMBED_TAB_NAME = GladeVCP demo`
- `EMBED_TAB_COMMAND = halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID\} -u ./glad ./gladevcp/manual-example.ui`

  **Note**
  Различные программы пользовательского интерфейса используют разные параметры, и не каждый параметр поддерживается каждым пользовательским интерфейсом. Подробную информацию об AXIS см. в документе AXIS GUI. Подробности о GMOCCAPY см. в документе GMOCCAPY.

- `DEFAULT_LINEAR_VELOCITY = .25` - Скорость по умолчанию для линейной медленной подачи в machine unit в секунду.
- `MIN_VELOCITY = .01` - Примерное минимальное значение шага ползунка медленной подачи.
- `MAX_LINEAR_VELOCITY = 1.0` - Максимальная скорость для линейной медленной подачи, в единицах станка в секунду.
- `MIN_LINEAR_VELOCITY = .01` - Примерное минимальное значение шага ползунка медленной подачи.
- `DEFAULT_ANGULAR_VELOCITY = .25` - Скорость по умолчанию для угловой медленной подачи в единицах станка в секунду.
- `MIN_ANGULAR_VELOCITY = .01` - Примерное минимальное значение углового шага ползунка медленно подачи.
- `MAX_ANGULAR_VELOCITY = 1.0` - Максимальная скорость угловой медленной подачи, в единицах станка в секунду.
- `INCREMENTS = 1 mm, .5 in, ...` - Определяет приращения, доступные для дискретной медленной подачи. INCREMENTS можно использовать для переопределения значения по умолчанию. Значения могут быть десятичными (например, 0,1000) или натуральными дробями (например, 1/16), за которыми необязательно следует единица измерения (cm, mm, um, inch, in или mil). Если единица измерения не указана, подразумевается единица измерения станка. Метрические и имперские расстояния могут быть смешанными: INCREMENTS = 1 inch, 1 mil, 1 cm, 1 mm, 1 µm — допустимые значения.
- `GRIDS = 10 mm, 1 in, ...` - Определяет предустановленные значения для линий сетки. Значение интерпретируется так же, как `INCREMENTS`.
- `OPEN_FILE = /full/path/to/file.ngc` - Файл, который будет отображаться на заставке при запуске AXIS. При использовании пустой строки "" при запуске файл не будет загружен. GMOCCAPY не будет использовать этот параметр, поскольку предлагает соответствующую запись на своей странице настроек.
- `EDITOR = gedit` - Редактор, используемый при выборе File > Edit для редактирования G-кода из меню AXIS. Этот параметр необходимо настроить, чтобы работал этот пункт меню. Другая допустимая запись — `gnome-terminal -e vim`. Эта запись не относится к GMOCCAPY, поскольку GMOCCAPY имеет встроенный редактор.
- `TOOL_EDITOR = tooledit` - Редактор, используемый при редактировании таблицы инструментов (например, выбрав "File > Edit tool table..." в AXIS). Другие допустимые записи: `gedit`, `gnome-termina -e vim`, and `gvim`. Эта запись не относится к GMOCCAPY, поскольку GMOCCAPY имеет встроенный редактор.

- `PYVCP = /filename.xml` - Файл описания панели PyVCP. Дополнительную информацию см. в разделе PyVCP.
- `PYVCP_POSITION = BOTTOM` - Размещение панели PyVCP в пользовательском интерфейсе AXIS. Если эта переменная опущена, панель по умолчанию будет располагаться справа. Единственная допустимая альтернатива — `BOTTOM`. Дополнительную информацию см. в главе PyVCP.
- `LATHE = 1` - Любое непустое значение (включая ”0”) заставляет AXIS использовать ”lathe mode” с видом сверху и радиусом и диаметром на УЦИ.
- `BACK_TOOL_LATHE = 1` - Любое непустое значение (включая «0») заставляет AXIS использовать ”back tool lathe mode” с инвертированной осью X.
- `FOAM = 1` - Любое непустое значение (включая ”0”) приводит к тому, что AXIS переключает отображение на режим резака вспененых материалов.
- `GEOMETRY = XYZABCUVW` - Управляет **preview** и **backplot** движения. Этот элемент состоит из последовательности букв оси и управляющих символов, которым может предшествовать знак ”-”:

  1. Буквы X, Y, Z обозначают перемещение вдоль названной координаты.
  2. Буквы A, B, C обозначают вращение вокруг соответствующих осей X, Y, Z.
  3. Буквы U, V, W обозначают перемещение по соответствующим осям X, Y, Z.
  4. Каждая указанная буква должна встречаться в `[TRAJ]COORDINATES`, чтобы иметь эффект.
  5. Символ ”**-**”, предшествующий любой букве, инвертирует направление операции.
  6. Операции перемещения и вращения оцениваются **right-to-left**. Таким образом, использование `GEOMETRY=XYZBC` задает вращение C, за которым следует вращение B, за которым следуют перемещения Z, Y, X. Порядок последовательно следующих букв не имеет значения.
  7. Правильная строка GEOMETRY зависит от конфигурации станка и кинематики, используемой для его управления. Порядок букв важен. Например, вращение вокруг C, а затем B отличается от вращения вокруг B, а затем C.
  8. Повороты по умолчанию применяются относительно исходной точки станка. Пример: `GEOMETRY=CXYZ` сначала переводит контрольную точку в X, Y, Z, а затем выполняет вращение C вокруг оси Z с центром в начале координат станка.
  9. Пример перемещения UVW: `GEOMETRY=XYZUVW` заставляет UVW перемещаться в системе координат инструмента, а XYZ перемещаться в системе координат материала.
  10. Для станка резки вспененных материалов (`FOAM = 1`) необходимо указать ”XY;UV” или оставить значение пустым, даже если это значение в настоящее время игнорируется в режиме резки вспененных материалов. В будущей версии может быть определено, что означает ”;” но если это так, «XY;UV» будет означать то же самое, что и текущие вспененные материалы по умолчанию.
  11. Экспериментально: если в строку GEOMETRY включен символ восклицательного знака (!), точки отображения для поворотов A, B, C учитывают смещения X, Y, Z, установленные кодами G5x, G92. Пример: использование `GEOMETRY = !CXZ` для станка с `[TRAJ]COORDINATES=XZC` Это положение применимо только к живым графикам — предварительный просмотр G-кода должен выполняться с нулевыми смещениями G5x, G92. Этому можно способствовать, устанавливая смещения в программах только тогда, когда задача выполняется, как указано `#<_task> == 1`. Если при запуске существуют ненулевые смещения из-за постоянства, смещения следует обнулить и перезагрузить предварительный просмотр.

---

**Note**
Если в INI-файле нет `[DISPLAY]GEOMETRY`, значение по умолчанию предоставляется `[DISPLAY]DISPLAY` программой ГИП (обычно ”XYZABCUVW”).

---

- ARCDIVISION = 64 - Установите качество предварительного просмотра дуг. Дуги предварительно просматриваются путем разделения их на несколько прямых линий; полукруг разделен на части **ARCDIVISION**. Большие значения обеспечивают более точный предварительный просмотр, но загрузка занимает больше времени и приводит к более медленному отображению. Меньшие значения дают менее точный предварительный просмотр, но загрузка занимает меньше времени и может привести к более быстрому отображению. Значение по умолчанию 64 означает, что круг размером до 3 дюймов будет отображаться с точностью до 1 мил (0,03%).
- MDI_HISTORY_FILE = - Имя локального файла истории MDI. Если это не указано, AXIS сохранит историю MDI в **.axis_mdi_history** в домашнем каталоге пользователя. Это полезно, если у вас есть несколько конфигураций на одном компьютере.
- JOG_AXES = - Порядок, в котором клавиши медленной подачи назначаются буквам осей. Стрелки влево и вправо соответствуют первой букве оси, вверх и вниз — второй, page up/page down — третьей, а левая и правая скобки — четвертой. Если не указано, значение по умолчанию определяется значениями [TRAJ]COORDINATES, [DISPLAY]LATHE и [DISPLAY]FOAM.
- JOG_INVERT = - Для каждой буквы оси направление перемещения инвертируется. По умолчанию для токарных станков установлено значение "X", в противном случае поле пустое.

---

**Note**
Настройки для JOG_AXES и JOG_INVERT применяются к медленной подаче в глобальном режиме по букве координаты оси и действуют в глобальном режиме после успешного приведения в исходное положение. При работе в режиме сочленения до приведения в исходное положение клавиши перемещения на клавиатуре назначаются в фиксированной последовательности: стрелки влево/вправо: сочленение 0, стрелки вверх/вниз: сочленение 1, page up/page down: сочленение 2, левая/правая скобка: сочленение 3

---

- USER_COMMAND_FILE = mycommands.py - Имя дополнительного файла Python, зависящего от конфигур полученного из графического интерфейса AXIS вместо пользовательского файла ~/.axisrc.

---

**Note**
Следующий элемент [DISPLAY] используется только интерфейсом TKLinuxCNC.

---

- HELP_FILE = tklinucnc.txt - Путь к файлу справки.

#### 4.4.2.3 [FILTER] раздел

AXIS и GMOCCAPY имеют возможность отправлять загруженные файлы через программу-фильтр. Этот фильтр может выполнять любую желаемую задачу: что-то простое, например, проверка того, что файл заканчивается на M2, или что-то более сложное, например, определение того, являются ли входные данные образом глубины, и генерируют G-код для фрезерования определяемой им формы. Раздел [FILTER] INI-файла управляет работой фильтров. Сначала для каждого типа файла напишите строку PROGRAM_EXTENSION. Затем укажите программу, которая будет выполняться для каждого типа файла. Этой программе присваивается имя входного файла в качестве первого аргумента, и она должна записать код RS274NGC в стандартный вывод. Этот вывод будет отображаться в текстовой области, просматриваться в области отображения и выполняться LinuxCNC при запуске.

- PROGRAM_EXTENSION = .extension Description

Если ваш постпроцессор выводит файлы заглавными буквами, вы можете добавить следующую строку:

```
PROGRAM_EXTENSION = .NGC XYZ Post Processor
```

Следующие строки добавляют поддержку конвертера изображение-в-G-код, включенного в LinuxCNC.

```
PROGRAM_EXTENSION = .png,.gif,.jpg # Greyscale Depth Image
  png = image-to-gcode
  gif = image-to-gcode
  jpg = image-to-gcode
```

Пример пользовательского конвертера G-кода, расположенного в каталоге linuxcnc.

```
PROGRAM_EXTENSION = .gcode 3D Printer
  gcode = /home/mill/linuxcnc/convert.py
```

---

**Note**
Программный файл, связанный с расширением, должен иметь либо полный путь к программе, либо находиться в каталоге, расположенном на системном пути.

---

Также возможно указать интерпретатор:

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

Таким образом, любой скрипт Python может быть открыт, а его выходные данные обрабатываются как G-код. Один из таких примеров скрипта доступен в nc_files/holecircle.py. Этот скрипт создает G-код для сверления серии отверстий по окружности. Многие другие генераторы G-кода находятся на вики-сайте LinuxCNC https://wiki.linuxcnc.org/.

Фильтры Python должны использовать функцию печати для вывода результата в AXIS.

Этот пример программы фильтрует файл и добавляет ось W, соответствующую оси Z. Работа зависит от наличия пробела между каждым словом оси.

```
#!/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
  Если установлена переменная среды AXIS_PROGRESS_BAR, то строки, записанные в stderr формы выше, устанавливают индикатор выполнения AXIS в заданный процент. Эту функцию следует использовать любому фильтру, работающему в течение длительного времени.

#### 4.4.2.4 [RS274NGC] раздел

- PARAMETER_FILE = myfile.var - Файл, расположенный в том же каталоге, что и файл INI, содержащий параметры, используемые интерпретатором (сохраняемые между запусками).
- ORIENT_OFFSET = 0 - Значение с плавающей запятой, добавленное к параметру слова R операции M19 Orient Spindle. Используется для определения произвольного нулевого положения независимо от ориентации крепления энкодера.
- RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92.1 G94 G97 G98 - Строка кодов ЧПУ, с помощью которой инициализируется интерпретатор. Это не заменяет указание модальных G-кодов в верхней части каждого файла NGC, поскольку модальные коды станков различаются и могут быть изменены G-кодом, интерпретированным ранее в сеансе.
- SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs - Указывает список до 10 каталогов, разделенных двоеточиями (:), в которых осуществляется поиск, если в G-коде указаны однофайловые подпрограммы. Поиск в этих каталогах осуществляется после поиска [DISPLAY]PROGRAM_PREFIX (если он указан) и перед поиском [WIZARD]WIZARD_ROOT (если он указан). Пути ищутся в том порядке, в котором они перечислены. Используется первый соответствующий файл подпрограммы, найденный при поиске. Каталоги указываются относительно текущего каталога для INI-файла или как абсолютные пути. Список не должен содержать промежуточных пробелов.
- G64_DEFAULT_TOLERANCE = *n* (Default: 0) Default P value for G64 if P is not called out.
- G64_DEFAULT_NAIVETOLERANCE = *n* (Default: 0) Default Q value for G64 if Q is not called out.
- CENTER_ARC_RADIUS_TOLERANCE_INCH = *n* (по умолчанию: 0.00005)
- CENTER_ARC_RADIUS_TOLERANCE_MM = *n* (по умолчанию: 0.00127)
- USER_M_PATH = myfuncs:/tmp/mcodes:experimentalmcodes - Указывает список каталогов, разделенн двоеточием (:), для пользовательских функций. Каталоги указываются относительно текущего каталога для INI-файла или как абсолютные пути. Список не должен содержать промежуточных пробелов.

  Поиск производится для каждой возможной определяемой пользователем функции, обычно (M100-M199). Порядок поиска следующий:

  1. [DISPLAY]PROGRAM_PREFIX (если указан)
  2. Если [DISPLAY]PROGRAM_PREFIX не указан, выполните поиск в папке по умолчанию: nc_files
  3. Затем выполните поиск в каждом каталоге в списке [RS274NGC]USER_M_PATH.
     Первый исполняемый M1*xx*, найденный при поиске, используется для каждого M1*xx*.

**Note**
Максимальное количество каталогов USER_M_PATH определяется во время компиляции (тип: USER_DEFINED_FUNCTION_MAX_DIRS == 5).

- INI_VARS = 1 (Default: 1)
  Позволяет программам G-кода считывать значения из файла INI в формате #<_ini[section]name>. См. Параметры G-кода.
- HAL_PIN_VARS = 1 (по умолчанию: 1)
  Позволяет программам G-кода считывать значения контактов HAL в формате #<_hal[элемент HAL]>. Доступ к переменным возможен только для чтения. См. G-коды параметры для получения более подробной информации и важного предостережения.
- RETAIN_G43 = 0 (Default: 0)
  Когда установлен, вы можете включить G43 после загрузки первого инструмента и потом не беспокоиться о нем в программе. Когда вы наконец выгрузите последний инструмент, режим G43 отменяется.
- OWORD_NARGS = 0 (по умолчанию: 0)
  Если эта функция включена, то вызываемая подпрограмма может определить количество переданны фактических позиционных параметров, проверив параметр #<n_args>.
- NO_DOWNCASE_OWORD = 0 (по умолчанию: 0)
  Сохранять регистр в именах O-слов в комментариях, если этот параметр установлен, разрешает читать элементы HAL со смешанным регистром в структурированных комментариях, например (debug, #<_hal[MixedCaseItem]).
- OWORD_WARNONLY = 0 (по умолчанию: 0)
  Скорее предупреждение, чем ошибка, в случае ошибок в подпрограммах O-слов.
- DISABLE_G92_PERSISTENCE = 0 (по умолчанию: 0) Разрешить автоматическую очистку смещения G92 при запуске конфигурации.
- DISABLE_FANUC_STYLE_SUB = 0 (по умолчанию: 0) Если есть причина запретить подпрограммы Fanuc, установите значение 1.
- *G73_PECK_CLEARANCE = .020* (default: Metric machine: 1mm, imperial machine: .050 inches) Chip breaking back-off distance in machine units
- *G83_PECK_CLEARANCE = .020* (default: Metric machine: 1mm, imperial machine: .050 inches) Clearance distance from last feed depth when machine rapids back to bottom of hole, in machine units.

**Note**
Вышеупомянутые шесть опций контролировались битовой маской FEATURES в версиях LinuxCNC до 2.8. Этот тег INI больше не будет работать.
Для справки:

```
FEATURES & 0x1  -> RETAIN_G43
FEATURES & 0x2  -> OWORD_NARGS
FEATURES & 0x4  -> INI_VARS
FEATURES & 0x8  -> HAL_PIN_VARS
FEATURES & 0x10 -> NO_DOWNCASE_OWORD
FEATURES & 0x20 -> OWORD_WARNONLY
```

**Note**
[WIZARD]WIZARD_ROOT является допустимым путем поиска, но Wizard не реализован полностью, и результаты его использования непредсказуемы.

- `LOG_LEVEL = 0` Укажите log_level (по умолчанию: 0)
- `LOG_FILE = file-name.log`
  Для указания файла, используемого для записи данных.
- `REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure` Подробности см. в главе Remap Extending G-code.
- `ON_ABORT_COMMAND=O <on_abort> call` Подробности см. в главе Remap Extending G-code.

#### 4.4.2.5 [EMCMOT] раздел

Этот раздел является настраиваемым и не используется LinuxCNC напрямую. В большинстве конфигураций значения из этого раздела используются для загрузки контроллера движений. Дополнительную информацию о контроллере движения см. в разделе Motion.

- `EMCMOT = motmod` - здесь обычно используется имя контроллера движения.
- `BASE_PERIOD = 50000` - период выполнения задачи *Base* в наносекундах.
- `SERVO_PERIOD = 1000000` - Это период задачи ”Servo” в наносекундах.
- `TRAJ_PERIOD = 100000` - Это период задачи *Trajectory Planner* в наносекундах.
- `COMM_TIMEOUT = 1.0` - Количество секунд ожидания, пока Motion (часть контроллера движения, работающая в режиме реального времени) подтвердит получение сообщений от Task (часть контроллера движения, работающая не в реальном времени).
- `HOMEMOD` = *alternate_homing_module* [home_parms=value] Переменная HOMEMOD является необязат Если указана, используйте указанный (созданный пользователем) модуль вместо модуля по умолчанию (homemod). Параметры модуля (home_parms) могут быть включены, если они поддерживаются указанным модулем. Эту настройку можно переопределить из командной строки с помощью опции -m ($ linuxcnc -h).

#### 4.4.2.6 [TASK] раздел

- `TASK = milltask` - Указывает имя исполняемой *task*. Исполняемый файл *task* выполняет различные действия, например:
  - общаться с пользовательскими интерфейсами через NML,
  - взаимодействовать с планировщиком движений в реальном времени через общую память, отличную от HAL, и
  - интерпретировать G-код. В настоящее время существует только один исполняемый файл задачи, который имеет смысл для 99,9% пользователей — milltask.
- `CYCLE_TIME = 0.010` - The period, in seconds, at which TASK will run. This parameter affects the polling interval when waiting for motion to complete, when executing a pause instruction, and when accepting a command from a user interface. There is usually no need to change this number. Defaults to 0.100 if omitted.

#### 4.4.2.7 [HAL] раздел

- `HALFILE = example.hal` - Запустите файл *example.hal* при запуске.

  Если `HALFILE` указан несколько раз, файлы интерпретируются в том порядке, в котором они появляются в INI-файле. Файлы HAL являются описательными, выполнение того, что описано в файлах HAL, инициируется потоками, в которые встроены функции, а не чтением файла HAL. Почти все конфигурации будут иметь по крайней мере один HALFILE, а в шаговых системах обычно есть два таких файла, то есть один, который определяет общую конфигурацию шагового привода (*core_stepper.hal*), и другой, который определяет расположение контактов станка (*xxx_pinou*

  Файлы HAL, указанные в переменной `HALFILES`, находятся с помощью поиска. Если указанный файл найден в каталоге, содержащем файл INI, он используется. Если указанный файл не найден в этом каталоге файлов INI, поиск выполняется с использованием системной библиотеки файлов HAL.

  Если LinuxCNC запускается со сценарием `linuxcnc` с использованием опции "`-H` *dirname*", указанное имя каталога добавляется к описанному выше поиску, так что сначала выполняется поиск *dirname*. Опцию "`-H` *dirname*" можно указать несколько раз, каталоги добавляются по порядку.

  HALFILE также может быть указан как абсолютный путь (когда имя начинается с символа /). Абсолютные пути использовать не рекомендуется, поскольку их использование может ограничить перемещение конфигураций.

- `HALFILE = texample.tcl` [*arg1* [*arg2*] ...] - Выполните файл tcl *texample.tcl* при запуске с arg1, arg2 и т. д. в качестве списка argv. Файлы с суффиксом .tcl обрабатываются, как указано выше, но для обработки используется остановка. Дополнительную информацию смотрите в HALTCL Chapter.

- `HALFILE = LIB:sys_example.hal` - Запустите файл системной библиотеки *sys_example.hal* при запуске. Явное использование префикса LIB: приводит к использованию системной библиотеки HALFILE без поиска в каталоге файлов INI.

- `HALFILE = LIB:sys_texample.tcl` [*arg1* [*arg2* ...]] - Запустите файл системной библиотеки *sys_texamp* при запуске. Явное использование префикса LIB: приводит к использованию системной библиотеки HALFILE без поиска в каталоге файлов INI.

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

1. отсутствие оператора addf, необходимого для добавления функции(й) компонента в поток,
2. неполные спецификаторы сигнала (net).

Пропуск необходимых операторов addf почти всегда является ошибкой. Сигналы обычно включают одно или несколько входных соединений и один выход (но оба не являются строго обязательными). Для проверки этих условий предоставляется файл системной библиотеки и отчет на stdout и во всплывающий ГИП:

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

---

**Note**
Строка LIB:halcheck.tcl должна быть последней [HAL]HALFILE. Укажите опцию *nopopup*, чтобы подавить всплывающее сообщение и разрешить немедленный запуск. Соединения, выполненные с использованием POSTGUI_HALFILE, не проверяются.

---

- `TWOPASS = ON` - Используйте двухпроходную обработку для загрузки компонентов HAL. При обработке TWOPASS строки файлов, указанные в `[HAL]HALFILE`, обрабатываются за два прохода. На первом проходе (pass0) считываются все HALFILES и накапливаются множественные появления команд loadrt и loadusr. Эти накопленные команды загрузки выполняются в конце pass0. Такое накопление позволяет указывать строки нагрузки более одного раза для данного компонента (предоставляет names= имена уникальны при каждом использовании). На втором проходе (pass1) HALFILES считываются заново и выполняются все команды, кроме ранее выполненных команд загрузки.
- `TWOPASS = nodelete verbose` - Функцию `TWOPASS` можно активировать с помощью любой ненулевой строки, включая ключевые слова verbose и nodelete. Ключевое слово verbose вызывает вывод сведений на stdout. Ключевое слово nodelete сохраняет временные файлы в /tmp.

Для получения дополнительной информации см. главу HAL TWOPASS.

- `HALCMD` = *command* - Выполните *command* как одну команду HAL. Если `HALCMD` указан несколько раз, команды выполняются в том порядке, в котором они появляются в INI-файле. Строки `HALCMD` выполняются после всех строк `HALFILE`.
- `SHUTDOWN = shutdown.hal` - Запустите файл *shutdown.hal* при выходе LinuxCNC. В зависимости от используемых драйверов оборудования это может позволить установить выходные значения на определенные значения при обычном выходе LinuxCNC. Однако, поскольку нет никакой гарантии, что этот файл будет выполнен (например, в случае сбоя компьютера), он не заменяет правильную физическую цепочку аварийного останова или другие средства защиты от сбоя программного обеспечения.
- ‘POSTGUI_HALFILE = example2.hal` — выполнить `example2.hal’ после того, как ГИП создаст свои контакты HAL. Некоторые ГИП создают контакты HAL и поддерживают использование postgui halfile для их использования. К ГИП, поддерживающим файлы HAL postgui, относятся Touchy, AXIS, Gscreen и GMOCCAPY.
  Дополнительную информацию см. в разделе PyVCP with AXIS.
- `HALUI = halui` — добавляет контакты пользовательского интерфейса HAL.
  Для получения дополнительной информации см. главу HAL User Interface.

#### 4.4.2.8 [HALUI] раздел

- `MDI_COMMAND = G53 G0 X0 Y0 Z0` - Команду MDI можно выполнить с помощью `halui.mdi-command-00` Увеличьте число для каждой команды, указанной в разделе [HALUI]. Также можно запускать подпрограммы. `MDI_COMMAND = o<yoursub> CALL [#<yourvariable>]`

#### 4.4.2.9 [APPLICATIONS] раздел

LinuxCNC может запускать другие приложения до запуска указанного ГИП. Приложения можно запускать после заданной задержки, чтобы обеспечить возможность действий, зависящих от ГИП (например, создание контактов HAL для конкретного ГИП).

- `DELAY` = *value* - секунды ожидания перед запуском других приложений. Задержка может потребоваться, если приложение зависит от действий `[HAL]POSTGUI_HALFILE` или контактов HAL, созданных ГИП (по умолчанию `DELAY=0`).
- ‘APP =` *appname* [arg1 [arg2 …]]’ - Приложение, которое нужно запустить. Эту спецификацию можно включать несколько раз. Имя приложения может быть явно указано как абсолютное имя файла или имя файла, указанное с помощью тильды (первый символ / или ~), относительное имя файла (первые символы имени файла — ./) или как файл в каталоге файлов INI. Если по этим именам исполняемый файл не найден, то для поиска приложения используется пользовательск поиск PATH.
  Примеры:

- Имитируйте входные данные для контактов HAL для тестирования (используя sim_pin — простой ГИП для установки ввода для параметров, неподключенных контактов или сигналов без записывающих устройств):

```
APP = sim_pin motion.probe-input halui.abort motion.analog-in-00
```

- Вызовите halshow с ранее сохраненным watchlist. Поскольку LinuxCNC устанавливает в качестве рабочего каталога каталог для файла INI, вы можете ссылаться на файлы в этом каталоге (пример: my.halshow):

```
APP = halshow my.halshow
```

- В качестве альтернативы можно указать файл watchlist, идентифицированный по полному пути:

```
APP = halshow ~/saved_shows/spindle.halshow
```

- Откройте halscope, используя ранее сохраненную конфигурацию:

```
APP = halscope -i my.halscope
```

#### 4.4.2.10 [TRAJ] раздел

**Warning**
Новый планировщик траектории (TP) включен по умолчанию. Если у вас нет настроек TP в разделе [TRAJ] — LinuxCNC по умолчанию использует:
ARC_BLEND_ENABLE = 1
ARC_BLEND_FALLBACK_ENABLE = 0
ARC_BLEND_OPTIMIZATION_DEPTH = 50
ARC_BLEND_GAP_CYCLES = 4
ARC_BLEND_RAMP_FREQ = 100

Раздел [TRAJ] содержит общие параметры модуля планирования траектории в режиме *motion*.

- `ARC_BLEND_ENABLE = 1` - Включите новый TP. Если установлено значение 0, TP использует параболическое смешивание (1 сегмент вперед) (по умолчанию: 1).
- `ARC_BLEND_FALLBACK_ENABLE = 0` - При необходимости вернитесь к параболическим смесям, если расчетная скорость выше. Однако эта оценка приблизительна, и кажется, что простое ее отключение дает лучшую производительность (по умолчанию: 0).
- `ARC_BLEND_OPTIMIZATION_DEPTH = 50` - Посмотрите вперед на глубину в количестве сегментов.

  Чтобы немного подробнее рассказать об этом, вы можете выбрать это значение несколько произвольно. Вот формула, позволяющая оценить, какая *depth* вам нужна для конкретной конфигурации:

```
# n = v_max / (2.0 * a_max * t_c)
# where:
# n = optimization depth
# v_max = max axis velocity (UU / sec)
# a_max = max axis acceleration (UU / sec)
# t_c = servo period (seconds)
```

  Итак, станку с максимальной скоростью оси 10 IPS, максимальным ускорением 100 $IPS^2$ и периодом сервопривода 0,001 s потребуется:

10 / (2,0 * 100 * 0,001) = 50 сегментов, чтобы всегда достигать максимальной скорости вдоль самой быстрой оси.

На практике это число не так важно для настройки, поскольку для просмотра вперед редко требуется полная глубина, если только у вас нет большого количества очень коротких сегментов. Если во время тестирования вы заметили странные подтормаживания и не можете понять, откуда они берутся, сначала попробуйте увеличить эту глубину по формуле выше.

Если вы по-прежнему наблюдаете странные замедления, возможно, это связано с короткими сегментами программы. В этом случае попробуйте добавить небольшой допуск для обнаружения Naive CAM. Хорошее практическое правило таково:

```
# min_length ~= v_req * t_c
# where:
# v_req = desired velocity in UU / sec
# t_c = servo period (seconds)
```

Если вы хотите путешествовать по пути со скоростью 1 IPS = 60 IPM, а период вашего сервопривода составляет 0,001 s, то любые сегменты короче min_length будут замедлять путь. Если вы установите допуск Naive CAM примерно на эту минимальную длину, слишком короткие сегменты будут объединены вместе, чтобы устранить это узкое место. Конечно, установка слишком высокого допуска означает большие отклонения от траектории, поэтому вам придется немного поиграть с ним, чтобы найти хорошее значение. Я бы начал с 1/2 min_length, а затем увеличивал бы по мере необходимости. * `ARC_BLEND_GAP_CYCLES = 4` Насколько коротким должен быть предыдущий сегмент, прежде чем планировщик траектории *потребит* его.

Часто при переходе по дуге окружности между переходами остаются короткие сегменты линий. Поскольку геометрия должна быть круглой, мы не можем растушевать всю линию, если следующая немного короче. Поскольку планировщик траектории должен коснуться каждого сегмента хотя бы один раз, это означает, что очень маленькие сегменты будут значительно замедлять движение. Мое исправление этого способа - ”поглотить” короткий сегмент, сделав его частью дуги сглаживания. Поскольку линия+переход представляет собой один сегмент, нам не нужно замедляться, чтобы попасть на очень короткий сегмент. Скорее всего, вам не придется трогать эту настройку. * `ARC_BLEND_RAMP_FREQ = 20` - Это *граничная* частота для использования линейной скорости.

*Нарастающая скорость* в данном случае просто означает постоянное ускорение на всем участке. Это менее оптимально, чем трапециевидный профиль скорости, поскольку ускорение не максимальное. Однако, если сегмент достаточно короткий, у нас не будет достаточно времени, чтобы сильно ускориться, прежде чем мы достигнем следующего сегмента. Вспомните короткие отрезки линий из предыдущего примера. Поскольку это линии, ускорения на поворотах нет, поэтому мы можем разогнаться до требуемой скорости. Однако если эта линия находится между двумя дугами, то ей придется снова быстро замедлиться, чтобы оказаться в пределах максимальной скорости следующего сегмента. Это означает, что у нас есть резкий скачок ускорения, а затем резкий скачок замедления, вызывающий сильный рывок, при очень небольшом приросте производительности. Эта настройка позволяет устранить этот рывок на коротких сегментах.

По сути, если сегмент будет завершен за меньшее время, чем 1 / ARC_BLEND_RAMP_FREQ, мы не заморачиваемся с трапециевидным профилем скорости на этом сегменте и используем постоянное ускорение. (Установка `ARC_BLEND_RAMP_FREQ = 1000` эквивалентна всегда использовани трапециевидного ускорения, если частота серво потока составляет 1 кГц).

Вы можете охарактеризовать потерю производительности в худшем случае, сравнив скорость, которую достигает трапециевидный профиль, со скоростью нарастания:

```
# v_ripple = a_max / (4.0 * f)
# where:
# v_ripple = average velocity ”loss” due to ramping
# a_max = max axis acceleration
# f = cutoff frequency from INI
```

Для вышеупомянутого станка пульсация для частоты среза 20 Гц составляет 100 / (4 * 20) = 1,25 IPS. Эта цифра кажется высокой, но имейте в виду, что это лишь оценка наихудшего случая. В действительности профиль трапециевидного движения ограничен другими факторами, такими как нормальное ускорение или требуемая скорость, поэтому фактическая потеря производит должна быть намного меньше. Увеличение частоты среза может повысить производительность, но сделать движение более грубым из-за скачков ускорения. Для начала будет разумным значение в диапазоне от 20 до 200 Гц.

Наконец, никакие настройки не ускорят траекторию инструмента с множеством маленьких, крутых углов, поскольку вы ограничены ускорением на поворотах.

- `SPINDLES = 3` - Количество поддерживаемых шпинделей. Крайне важно, чтобы это число соответствовало параметру ”num_spindles”, передаваемому модулю движения.
- `COORDINATES = X Y Z` - Имена управляемых осей. Допустимы только X, Y, Z, A, B, C, U, V, W. В G-коде принимаются только оси, указанные в КООРДИНАТАХ. Разрешено писать имя оси более одного раза (например, X Y Y Z для портального станка). Для обычной *trivkins kinematics* номера соединений назначаются последовательно в соответствии с параметром trivkins *coordinates=*. Итак, для trivkins *coordinates=xz*, joint0 соответствует X, а joint1 соответствует Z. См. справочную страницу по кинематике (*$ man kins*) для получения информации о trivkins и других модулях кинематики.
- `LINEAR_UNITS = <units>_` - Указывает *единицы станка* для линейных осей. Возможные варианты: мм или дюйм. Это не влияет на линейные единицы в коде УП (это делают слова G20 и G21).
- `ANGULAR_UNITS = <units>` - Указывает *единицы станка* для осей вращения. Возможные варианты: *deg*, *degree* (360 на круг), *rad*, *radian* (2*п на круг), *grad*, или *gon* (400 на круг). Это не влияет на угловые единицы кода УП. В RS274NGC слова A, B и C всегда выражаются в градусах.
- `DEFAULT_LINEAR_VELOCITY = 0.0167` - Начальная скорость ступенчатого перемещения линейных осей, в машинных единицах в секунду. Значение, отображаемое в *AXIS*, равно машинным единицам в минуту.
- `DEFAULT_LINEAR_ACCELERATION = 2.0` - На станках с nontrivial kinematics ускорение, используемое для ”teleop” (декартово пространство) медленных подач, выражается в *единицах станка* в секунду.
- `MAX_LINEAR_VELOCITY = 5.0` - Максимальная скорость для любой оси или скоординированного перемещения в *единицах станках* в секунду. Отображаемое значение равно 300 единицам в минуту.
- `MAX_LINEAR_ACCELERATION = 20.0` - Максимальное ускорение для любой оси или перемещение согласованной оси, в *единицах станка* в секунду.
- `PLANNER_TYPE = 0` - Selects the trajectory planner type: 0 = trapezoidal (default), 1 = S-curve with jerk limiting. S-curve planning is only active when `PLANNER_TYPE = 1` AND `MAX_LINEAR_JERK > 0`.
- `MAX_LINEAR_JERK = 10000.0` - The maximum jerk (rate of change of acceleration) for coordinated moves, in *machine units* per second cubed. Default is 1e9 (1 billion) if not specified, which effectively disables jerk limiting while avoiding numerical instability. Values are clamped to a maximum of 1e9 to prevent numerical issues in S-curve calculations. When `PLANNER_TYPE = 1`, this enables S-curve trajectory planning. Note: Not specifying MAX_LINEAR_JERK (defaulting to 1e9) produces motion similar to trapezoidal planning (PLANNER_TYPE = 0) but not identical, as extremely high jerk still uses S-curve calculations.
- `POSITION_FILE = position.txt` - Если установлено непустое значение, позиции сочленений сохраняют между запусками в этом файле. Это позволяет станку стартовать с теми же координатами, которые были у него при выключении. Это предполагает, что при выключенном питании станок не двигался. Если не установлено, позиции сочленений не сохраняются и будут начинаться с 0 при каждом запуске LinuxCNC. Это может помочь на небольших станках без переключателей

исходной позиции. При использовании интерфейса преобразователя Mesa этот файл можно использовать для эмуляции абсолютных энкодеров и устранения необходимости возврата в исходное положение (без потери точности). Дополнительную информацию см. на странице руководства hostmot2.

- `NO_FORCE_HOMING = 1` - (bool) The default behavior is for LinuxCNC to force the user to home the machine before any MDI command or a program is run. Normally, only jogging is allowed before homing. For configurations using identity kinematics, setting `NO_FORCE_HOMING = 1` allows the user to make MDI moves and run programs without homing the machine first. Interfaces using identity kinematics without homing ability will need to have this option set to 1.

**Warning**
LinuxCNC не будет знать ваши ограничения перемещения сочленения при использовании NO_FORCE_HOMING = 1.

- `HOME = 0 0 0 0 0 0 0 0 0` - Глобальная исходная позиция необходима для модулей кинематики, которые вычисляют глобальные координаты с помощью kinematicsForward() при переключении из режима сочленения в teleop режим. Можно указать до девяти значений координат (X Y Z A B C U V W), неиспользуемые конечные элементы можно опустить. Это значение используется только для станков с nontrivial kinematics. На станках с trivial kinematics (фрезерные, токарные, портальные) это значение игнорируется. Примечание. Для конфигурации sim hexapod требуется ненулевое значение координаты Z.
- `TPMOD` = *alternate_trajectory_planning module* [tp_parms=value]
  Переменная `TPMOD` является необязательной. Если указано, используйте указанный (созданный пользователем) модуль вместо модуля по умолчанию (tpmod). Параметры модуля (tp_parms) могут быть включены, если они поддерживаются указанным модулем. Эту настройку можно переопределить из командной строки с помощью опции -t ($ linuxcnc -h).
- `NO_PROBE_JOG_ERROR = 0` - Разрешить обход проверки срабатывания датчика когда вы используете медленную подачу в ручном режиме.
- `NO_PROBE_HOME_ERROR = 0` - Позволяет обходить проверку срабатывания датчика во время возврата в исходное положение.

#### 4.4.2.11 [KINS] раздел

- `JOINTS = 3` - Указывает количество сочленений (двигателей) в системе. Например, станок trivkins XYZ с одним двигателем на каждую ось имеет 3 сочленения. Портальный станок с одним двигателем на каждой из двух осей и двумя двигателями на третьей оси имеет 4 сочленения. (Эта переменная конфигурации может использоваться ГИП для установки количества сочленений (num_joints), указанных для модуля motion (motmod).)
- `KINEMATICS = trivkins` - Укажите модуль kinematics для модуля motion. ГИПы могут использовать эту переменную для указания строки `loadrt` в файлах HAL для модуля motmod. Дополнительную информацию о модулях кинематики смотрите на странице руководства: `$ man kins`.

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

<*letter*> указывает одну из: X Y Z A B C U V W

- `TYPE = LINEAR` - (enum) The type of this axis, either `LINEAR` or `ANGULAR`. Required if this axis is not a default axis type. The default axis types are X,Y,Z,U,V,W = LINEAR and A,B,C = ANGULAR. This setting is effective with the AXIS GUI but note that other GUI's may handle things differently.

- `MAX_VELOCITY = 1.2` - (real) Maximum velocity for this axis in machine units per second.
- `MAX_ACCELERATION = 20.0` - (real) Maximum acceleration for this axis in machine units per second squared.
- `MAX_JERK = 0.0` - (real) Maximum jerk for this axis in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-axis jerk limiting is applied.
- `MIN_LIMIT = -1000` - (real) The minimum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before `MIN_LIMIT` is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MIN_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of -1e99 is used.
- `MAX_LIMIT = 1000` - (real) The maximum limit (soft limit) for axis motion, in machine units. When this limit is exceeded, the controller aborts axis motion. The axis must be homed before MAX_LIMIT is in force. For a rotary axis (A,B,C typ) with unlimited rotation having no `MAX_LIMIT` for that axis in the `[AXIS_<letter>]` section a value of 1e99 is used.
- `WRAPPED_ROTARY = 1` - (bool) When this is set to 1 for an ANGULAR axis the axis will move 0-359.999 degrees. Positive Numbers will move the axis in a positive direction and negative numbers will move the axis in the negative direction.
- `LOCKING_INDEXER_JOINT = 4` - (int) This value selects a joint to use for a locking indexer for the specified axis <*letter*>. In this example, the joint is 4 which would correspond to the B axis for a XYZAB system with trivkins (identity) kinematics. When set, a G0 move for this axis will initiate an unlock with the `joint.4.unlock pin` then wait for the `joint.4.is-unlocked` pin then move the joint at the rapid rate for that joint. After the move the `joint.4.unlock` will be false and motion will wait for `joint.4.is-unlocked` to go false. Moving with other joints is not allowed when moving a locked rotary joint. To create the unlock pins, use the motmod parameter:

  ```
  unlock_joints_mask=jointmask
  ```

  Биты маски соединения: (LSB)0:joint0, 1:joint1, 2:joint2, ...

  Пример: `loadrt motmod ... unlock_joints_mask=0x38` создает разблокирующие контакты для сочленений 3,4,5.
- `OFFSET_AV_RATIO = 0.1` - (real) If nonzero, this item enables the use of HAL input pins for external axis offsets:

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

Информацию об использовании смотрите в главе: External Axis Offsets.

#### 4.4.2.13 [JOINT_<num>] разделы

<*num*> указывает номер сочленения 0 ... (num_joints-1). Значение *num_joints* устанавливается с помощью `[KINS]JOINTS=`.

Разделы `[JOINT_0]`, `[JOINT_1]` и т. д. содержат общие параметры для отдельных компонентов в модуле управления сочленением. Имена секций сочленений начинаются с 0 и заканчиваются количеством сочленений, указанным в записи `[KINS]JOINTS` минус 1.

Обычно (для систем, использующих *trivkins kinematics*, существует соответствие 1:1 между буквой координат сочленения и оси):

- 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

Другие кинематические модули с identity kinematics доступны для поддержки конфигураций с частичными наборами осей. Например, при использовании trivkins с `coordinates=XZ` соотношения сочленение-ось будут такими:

- JOINT_0 = X
- JOINT_1 = Z

Для получения дополнительной информации о модулях кинематики см. справочную страницу *kins* (в терминале UNIX введите `man kins`).

- `TYPE = LINEAR` - (enum) The type of joint, either `LINEAR` or `ANGULAR`.
- `UNITS = INCH` - (enum) If specified, this setting overrides the related `[TRAJ] UNITS` setting, e.g., `[TRAJ]LINEAR_UNITS` if the `TYPE` of this joint is `LINEAR`, `[TRAJ]ANGULAR_UNITS` if the `TYPE` of this joint is `ANGULAR`.
- `MAX_VELOCITY = 1.2` - (real) Maximum velocity for this joint in machine units per second.
- `MAX_ACCELERATION = 20.0` - (real) Maximum acceleration for this joint in machine units per second squared.
- `MAX_JERK = 0.0` - (real) Maximum jerk for this joint in machine units per second cubed. Used when S-curve trajectory planning is enabled. When set to 0 (default), no per-joint jerk limiting is applied.
- `BACKLASH = 0.0000` - (real) Backlash in machine units. Backlash compensation value can be used to make up for small deficiencies in the hardware used to drive an joint. If backlash is added to an joint and you are using steppers the `STEPGEN_MAXACCEL` must be increased to 1.5 to 2 times the `MAX_ACCELERATION` for the joint. Excessive backlash compensation can cause an joint to jerk as it changes direction. If a COMP_FILE is specified for a joint BACKLASH is not used.
- `COMP_FILE =` *file.extension* - (string) The compensation file consists of map of position information for the joint. Compensation file values are in machine units. Each set of values are are on one line separated by a space. The first value is the nominal value (the commanded position). The second and third values depend on the setting of `COMP_FILE_TYPE`. Points in between nominal values are interpolated between the two nominals. Compensation files must start with the smallest nominal and be in ascending order to the largest value of nominals. File names are case sensitive and can contain letters and/or numbers. Currently the limit inside LinuxCNC is for 256 triplets per joint.

  Если для соединения указан `COMP_FILE`, `BACKLASH` не используется.
- `COMP_FILE_TYPE = 0` or `1` - (int) Specifies the type of compensation file. The first value is the nominal (commanded) position for both types.
  A `COMP_FILE_TYPE` must be specified for each `COMP_FILE`.

  - *Type 0:* Второе значение определяет фактическое положение при движении сочленения в положительном направлении (возрастающее значение). Третье значение определяет фактическое положение, когда сочленение движется в отрицательном направлении (убывающее значение).

    **Type 0 Пример**

```
-1.000 -1.005 -0.995
0.000 0.002 -0.003
1.000 1.003 0.998
```

- *Type 1:* Второе значение определяет положительное смещение от номинала при движении в положительном направлении. Третье значение определяет отрицательное смещение от номинала при движении в отрицательном направлении.

  **Type 1 Пример**

```
-1.000 0.005 -0.005
0.000 0.002 -0.003
1.000 0.003 -0.004
```

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

---

**Note**
Для **identity** kinematics настройки `[JOINT_N]MIN_LIMIT/MAX_LIMIT` должны равняться или превышать соответствующие пределы `[AXIS_L]` (идентичность один к одному). Эти настройки проверяются при запуске, когда указаны модули trivkins kinematics.

---

---

**Note**
Настройки `[JOINT_N]MIN_LIMIT/MAX_LIMIT` применяются при медленной подачи в режиме сочленения перед возвратом в исходное положение. После возврата в исходное положение пределы координат `[AXIS_L]MIN_LIMIT/MAX_LIMIT` используются в качестве ограничений для перемещения оси (буквы координат) и при планировании траектории, используемой для перемещений G-кода (программы и команды MDI). Планировщик траектории работает в декартовом пространстве (XYZABCUVW) и не имеет информации о движении сочленений, реализуемом **каким-либо** кинематическим модулем. Возможно возникновение нарушений предела сочленения для G-кода, который подчиняется пределу позиции планирования траектории, когда используется identity kinematics. Модуль движения всегда обнаруживает нарушения и неисправности предельных положений сочленения, если они возникают во время выполнения команд G-кода. См. также соответствующую https://github.com/LinuxCNC/-linuxcnc/issues/97 [GitHub issue #97].

---

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

- LOCKING_INDEXER = 1 - (bool) Indicates the joint is used as a locking indexer.

Эти параметры связаны с возвращением в исходное положение. Для более подробного объяснения прочтите главу Homing Configuration.

- HOME = 0.0 - (real) The position that the joint will go to upon completion of the homing sequence.
- HOME_OFFSET = 0.0 - (real) The joint position of the home switch or index pulse, in machine units. When the home point is found during the homing process, this is the position that is assigned to that point. When sharing home and limit switches and using a home sequence that will leave the home/limit switch in the toggled state, the home offset can be used define the home switch position to be other than 0 if your HOME position is desired to be 0.
- HOME_SEARCH_VEL = 0.0 - (real) Initial homing velocity in machine units per second. Sign denotes direction of travel. A value of zero means assume that the current location is the home position for the machine. If your machine has no home switches you will want to leave this value at zero.
- HOME_LATCH_VEL = 0.0 - (real) Homing velocity in machine units per second to the home switch latch position. Sign denotes direction of travel.
- HOME_FINAL_VEL = 0.0 - (real) Velocity in machine units per second from home latch position to home position. If left at 0 or not included in the joint rapid velocity is used. Must be a positive number.
- HOME_USE_INDEX = NO - (bool) If the encoder used for this joint has an index pulse, and the motion card has provision for this signal you may set it to yes. When it is yes, it will affect the kind of home pattern used. Currently, you can't home to index with steppers unless you're using StepGen in velocity mode and PID.
- HOME_INDEX_NO_ENCODER_RESET = NO - (bool) Use YES if the encoder used for this joint does not reset its counter when an index pulse is detected after assertion of the joint index_enable HAL pin. Applicable only for HOME_USE_INDEX = YES.
- HOME_IGNORE_LIMITS = NO - (bool) When you use the limit switch as a home switch and the limit switch this should be set to YES. When set to YES the limit switch for this joint is ignored when homing. You must configure your homing so that at the end of your home move the home/limit switch is not in the toggled state you will get a limit switch error after the home move.
- HOME_IS_SHARED = NO - (bool) If the home input is shared by more than one joint set to true to prevent homing from starting if the one of the shared switches is already closed. Set to false (the default) to permit homing if a switch is closed.
- HOME_ABSOLUTE_ENCODER = 0 | 1 | 2 - (int) Used to indicate the joint uses an absolute encoder. At a request for homing, the current joint value is set to the HOME_OFFSET value. If the HOME_ABSOLUTE_ENCODER setting is 1, the machine makes the usual final move to the HOME value. If the HOME_ABSOLUTE_ENCODER setting is 2, no final move is made.
- HOME_SEQUENCE = <*n*> - (int) Used to define the "Home All" sequence. <*n*> must start at 0 or 1 or -1. Additional sequences may be specified with numbers increasing by 1 (in absolute value). Skipping of sequence numbers is not allowed. If a HOME_SEQUENCE is omitted, the joint will not be homed by the "Home All" function. More than one joint can be homed at the same time by specifying the same sequence number for more than one joint. A negative sequence number is used to defer the final move for all joints having that (negative or positive) sequence number. For additional info, see: HOME SEQUENCE.
- VOLATILE_HOME = 0 - (bool) When enabled (set to 1) this joint will be unhomed if the Machine Power is off or if E-Stop is on. This is useful if your machine has home switches and does not have position feedback such as a step and direction driven machine.

Эти параметры относятся к сочленениям, управляемыми сервоприводами.

**Warning**
Ниже приведены пользовательские записи INI-файла, которые вы можете найти в образце INI-файла или файле, созданном мастером. Они не используются программным обеспечением LinuxCNC. Они нужны только для того, чтобы собрать все настройки в одном месте. Дополнительную информацию о пользовательских записях INI-файла см. в подразделе Custom Sections and Variables.

Следующие элементы могут использоваться компонентом ПИД-регулятора, при этом предполагается, что выходной сигнал - уровень напряжения.

- `DEADBAND = 0.000015` - (real) How close is close enough to consider the motor in position, in machine units.

  Часто оно устанавливается на расстояние, эквивалентное 1, 1,5, 2 или 3 отсчетам энкодера, но строгих правил не существует. Более свободные (бОльшие) настройки позволяют уменьшить *hunting* сервопривода за счет более низкой точности. Более жесткие (меньшие) настройки обеспечивают более высокую точность за счет большей *hunting* сервопривода. Действительно ли это более точно, если оно также и более неопределенно? Как правило, лучше избегать или, по крайней мере, ограничивать *hunting* сервоприводов, если это возможно.

  Будьте осторожны при уменьшении отсчетов энкодера ниже 1, так как вы можете создать условие, при котором ваш сервопривод не будет работать. Это может варьироваться от *hunting* (медленного) до *nervous* (быстрого) и даже до *squealing*, которое легко спутать с колебаниями, вызванными неправильной настройкой. Лучше поначалу пропустить отсчет другой, пока вы не пройдете хотя бы *gross tuning*.

  Пример расчета единиц станка на импульс энкодера для использования при определении значения `DEADBAND`:

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

- `BIAS = 0.000` - Этот параметр используется hm2-servo и некоторыми другими. Смещение — это постоянная величина, которая добавляется к выходным данным. В большинстве случаев его следует оставить равным нулю. Однако, иногда может быть полезно компенсировать смещения в сервоусилителях или сбалансировать вес объекта, который движется вертикально. Смещение выключается при отключении контура ПИД, как и все остальные компоненты выхода.
- `P = 50` - Пропорциональное усиление для сервопривода сочленения. Это значение умножает ошибку между заданным и фактическим положением в единицах станка, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления P — это вольты на единицу станка, например $\frac{volts}{unit}$
- `I = 0` - Интегральный коэффициент усиления сервопривода сочленения. Это значение умножает совокупную ошибку между заданным и фактическим положением в единицах станка, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления I – вольты на единицу станка в секунду, например: $\frac{volts}{unit\ second}$
- `D = 0` - Производное усиление для сервопривода сочленения. Это значение умножает разницу между текущей и предыдущей ошибками, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления D — это вольты на единицу станка в секунду, например $\frac{volts}{unit\ second}$

- `FF0 = 0` - Усиление прямой связи 0$^{го}$ порядка. Это число умножается на заданное положение, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления FF0 — это вольты на единицу станка, например $\frac{volts}{unit}$
- `FF1 = 0` - Усиление прямой связи 1^го порядка. Это число умножается на изменение заданного положения в секунду, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления FF1 — это вольты на единицу станка в секунду, например $\frac{volts}{unit\ second}$
- `FF2 = 0` - Усиление прямой связи 2$^{го}$ порядка. Это число умножается на изменение заданного положения в секунду за секунду, что приводит к увеличению расчетного напряжения для усилителя двигателя. Единицы усиления FF2 — это вольты на единицу станка в секунду в секунду, например $\frac{volts}{unit\ second^2}$
- `OUTPUT_SCALE = 1.000`
- `OUTPUT_OFFSET = 0.000`

  Эти два значения являются коэффициентами масштабирования и смещения для выхода сочленения на усилители двигателей.

  Второе значение (смещение) вычитается из вычисленного выходного сигнала (в вольтах) и делится на первое значение (масштабный коэффициент) перед записью в цифро-аналоговые преобразователи. Единицы на значениях шкалы выражаются в истинных вольтах на выходное напряжение ЦАП. Единицы измерения значения смещения указаны в вольтах. Их можно использовать для линеаризации ЦАП. В частности, при записи выходных данных LinuxCNC сначала преобразует желаемый выходной сигнал в единицах квази СИ-единицы в необработанные значения исполнительного механизма, например, вольты для ЦАП усилителя. Это масштабирование выглядит так: $raw = \frac{output - offset}{scale}$

  Значение масштаба можно получить аналитически, выполнив анализ единиц измерения, т. е. единицами измерения являются [выходные единицы СИ]/[единицы привода]. Например, на станке с усилителем в режиме скорости, где напряжение 1В приводит к скорости 250 мм/с.

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

  Обратите внимание, что единицы смещения указаны в единицах станка, например: мм/с, и они предварительно вычитаются из показаний датчика. Значение этого смещения получается путем нахождения значения вашего выхода, которое дает 0,0 для выхода привода. Если ЦАП линеаризован, это смещение обычно равно 0,0.

  Масштаб и смещение также можно использовать для линеаризации ЦАП, в результате чего получаются значения, отражающие совокупное влияние коэффициента усиления усилителя, нелинейности ЦАП, блоков ЦАП и т. д.

  Для этого выполните следующую процедуру.

  1. Постройте калибровочную таблицу для выхода, подавая на ЦАП нужное напряжение и измеряя результат.
  2. Выполните линейную аппроксимацию методом наименьших квадратов, чтобы получить коэффициенты a, b такие, что $measured = a * raw + b$
  3. Обратите внимание: нам нужен необработанный выходной сигнал, чтобы результат измерения был идентичен заданному выходному сигналу. Это означает
     a. $command = a * raw + b$

b. $raw = (command - b)/a$

4. В результате коэффициенты a и b из линейной аппроксимации можно использовать непосредств в качестве масштаба и смещения для контроллера.

В следующей таблице приведен пример измерения напряжения.

Table 4.1: Измерения выходного напряжения

| Необработанные | Измеренные |
|---|---|
| -10 | -9.93 |
| -9 | -8.83 |
| 0 | -0.03 |
| 1 | 0.96 |
| 9 | 9.87 |
| 10 | 10.87 |

- `MAX_OUTPUT = 10` - Максимальное значение выходного сигнала ПИД-компенсации, записываемого в усилитель двигателя, в вольтах. Вычисленное выходное значение ограничивается этим пределом. Ограничение применяется перед масштабированием до необработанных выходных единиц. Значение применяется симметрично как к плюсовой, так и к минусовой стороне.
- `INPUT_SCALE = 20000` - в примерах конфигураций
- `ENCODER_SCALE = 20000` - во встроенных конфигах PnCconf

Указывает количество импульсов, соответствующее перемещению одного узла станка, как установлен в разделе `[TRAJ]`. Для линейного сочленения одна единица станка будет равна настройке `LINEAR_UNITS` Для углового сочленения одна единица равна настройке в `ANGULAR_UNITS`. Второе число, если оно указано, игнорируется. Например, для энкодера с частотой 2000 отсчетов на оборот, передачей 10 об/дюйм и желаемыми единицами измерения в дюймах мы имеем:

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

Эти параметры актуальны для сочленений, управляемых шаговыми двигателями.

**Warning**
Ниже приведены пользовательские записи INI-файла, которые вы можете найти в образце INI-файла или файле, созданном мастером. Они не используются программным обеспечением LinuxCNC и предназначены только для размещения всех настроек в одном месте. Дополнительную информацию о пользовательских записях INI-файла см. в подразделе Custom Sections and Variables.

Следующие элементы могут использоваться компонентом StepGen.

- `SCALE = 4000` - в примерах конфигураций
- `STEP_SCALE = 4000` - во встроенных конфигах PnCconf

Указывает количество импульсов, соответствующее перемещению одного узла станка, как установлен в разделе [TRAJ]. Для шаговых систем это количество шаговых импульсов, выдаваемых на единицу станка. Для линейного сочленения одна единица станка будет равна настройке LINEAR_UNITS. Для углового соединения одна единица равна настройке в ANGULAR_UNITS. Для сервосистем это количество импульсов обратной связи на единицу станка. Второе число, если оно указано, игнорируется.

Например, для шагового двигателя 1,8 градуса с полушагом, передачей 10 об/дюйм и желаемым machine unit в дюймах мы имеем:

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

**Note**
В старых файлах INI и HAL для этого значения использовалось INPUT_SCALE.

- ENCODER_SCALE = 20000 (Дополнительно используется во встроенных конфигурациях PnCconf) — указывает количество импульсов, соответствующее перемещению одной единицы станка, как установлено в разделе [TRAJ]. Для линейного сочленения одна единица станка будет равна настройке LINEAR_UNITS. Для углового соединения одна единица равна настройке в ANGULAR_UNITS. Второе число, если оно указано, игнорируется. Например, для энкодера с частотой 2000 отсчетов на оборот, передачей 10 об/дюйм и желаемыми единицами измерения в дюймах мы имеем:

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

- STEPGEN_MAXACCEL = 21.0 - Предел ускорения для генератора шагов. Оно должно быть на 1–10 % больше, чем MAX_ACCELERATION сочленения. Это значение улучшает настройку ”position loop” StepGen. Если вы добавили компенсацию люфта в сочленение, то оно должно быть в 1,5–2 раза больше, чем MAX_ACCELERATION.
- STEPGEN_MAXVEL = 1.4 - Более старые файлы конфигурации также имеют ограничение скорости для генератора шагов. Если указано, оно также должно быть на 1–10% больше, чем MAX_VELOCITY сочленения. Последующее тестирование показало, что использование STEPGEN_MAXVEL не улучшает настройку цикла позиционирования StepGen.

#### 4.4.2.14 [SPINDLE_<num>] раздел(ы))

*<num>* указывает номер шпинделя 0... (num_spindles-1)

Значение *num_spindles* устанавливается [TRAJ]SPINDLES= .

По умолчанию максимальная скорость шпинделя в прямом и обратном направлении составляет примерно 2147483000 об/мин.

По умолчанию минимальная скорость шпинделя в прямом и обратном направлении равна 0 об/мин.

По умолчанию приращение составляет 100 об/мин.

Вы можете изменить эти значения по умолчанию, установив следующие переменные INI:

**Note**
Эти настройки предназначены для компонента контроллера движения. Экраны управления могут дополнительно ограничить эти настройки.

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

#### 4.4.2.15 [EMCIO] раздел

- `TOOL_TABLE = tool.tbl` - (string) The file which contains tool information, described in the User Manual.
- `DB_PROGRAM = db_program` - (string) Path to an executable program that manages tool data. When a DB_PROGRAM is specified, a TOOL_TABLE entry is ignored.
- `TOOL_CHANGE_POSITION = 0 0 2` - Указывает местоположение XYZ, к которому необходимо перейти при смене инструмента, если используются три цифры. Указывает местоположение XYZABC при использовании 6 цифр. Указывает местоположение XYZABCUVW при использовании 9 цифр. Замены инструмента можно комбинировать. Например, если вы комбинируете подъем пиноли с изменением положения, вы можете сначала переместить Z, затем X и Y.
- `TOOL_CHANGE_WITH_SPINDLE_ON = 1` - (bool) The spindle will be left on during the tool change when the value is 1. Useful for lathes or machines where the material is in the spindle, not the tool.
- `TOOL_CHANGE_QUILL_UP = 1` - (bool) The Z axis will be moved to machine zero prior to the tool change when the value is 1. This is the same as issuing a `G0 G53 Z0`.
- `TOOL_CHANGE_AT_G30 = 1` - (bool) The machine is moved to reference point defined by parameters 5181-5186 for G30 if the value is 1. For more information see G-code Parameters and G-code G30-G30.1.
- `RANDOM_TOOLCHANGER = 1` - (bool) This is for machines that cannot place the tool back into the pocket it came from. For example, machines that exchange the tool in the active pocket with the tool in the spindle.

# 4.5 Конфигурация исходного положения

## 4.5.1 Обзор

Приведение к исходному положению устанавливает нулевое начало координат станка G53. Программ
пределы определяются относительно исходной точки станка. Программные пределы автоматически замедляют и останавливают оси до того, как они достигнут концевых выключателей. Правильно сконфигурированный и функционирующий станок не выйдет за пределы программных пределов, и исходная точка станка будет установлена с такой же повторяемостью, как и механизм исходного концевика/метки. Linuxcnc можно установить на глаз (метки совмещения), с помощью концевиков, концевиков и метки энкодера или с помощью абсолютных энкодеров. Возврат в исходное положение кажется достаточно простым — просто переместите каждое сочленение в известное место и соответствующим образом установите внутренние переменные LinuxCNC. Однако у разных станков разные требования, и приведение в исходное положение на самом деле довольно сложное.

**Note**
Хотя можно использовать LinuxCNC без концевиков/процедур приведения в исходное положение или концевиков пределов, это нарушает дополнительную безопасность программных пределов.

## 4.5.2 Предварительные условия

Возврат в исходное положение опирается на некоторые фундаментальные предположения о станке.

- Отрицательное и положительное направления основаны на Tool Movement, которые могут отличаться от фактического движения станка. То есть на фрезерном станке обычно движется стол, а не инструмент.
- Все происходит от нулевой точки станка G53, начало координат может быть где угодно (даже за пределами места, куда вы можете перемещаться)
- Начало координат станка G53 обычно находится внутри области программных пределов, но не обязательно.
- Смещение концевика исходного положения устанавливает начало координат, но даже на него ссылаются из начала координат.
- При использовании для возврата в исходное положение индекса энкодера, смещение концевика исходного положения рассчитывается на основе референтного положения энкодера после срабатывания концевика исходного положения.
- Отрицательные пределы программных пределов — это максимум, который вы можете изменить в отрицательном направлении после возврата в исходное положение. (но они могут не быть отрицательными в абсолютном смысле)
- Положительные программные пределы — это максимум, на который вы можете двигаться в положительном направлении после возврата в исходное положение. (но они могут не быть положительными в абсолютном смысле, хотя обычно их определяют как положительное число)
- Программные пределы находятся внутри области концевого выключателя.
- (Окончательное) Исходное положение находится внутри области программного предела

- (При использовании приведения в исходное положение с помощью концевика) концевик(и) возврата в исходное положение либо используют концевые выключатели пределов (общий концевик предела/исходного положения), либо при использовании отдельного концевика исходного положения находятся внутри области концевика предела.
- При использовании отдельного концевика возврата в исходное положение можно начать возвращени к исходному состоянию не с той стороны переключателя исходного положения, что в сочетании с опцией HOME_IGNORE_LIMITS приведет к серьезному сбою. Вы можете избежать этого, заставив концевик исходного положения переключать свое состояние, когда ограничитель находится на определенной стороне, пока он снова не пройдет точку срабатывания. Другими словами, состояние концевика исходного положения должно отражать положение ограничителя относительно концевика(т.е. *до* или *после* переключателя) и должно оставаться таким, даже если ограничитель проезжает мимо переключателя в том же направлении.

---

**Note**
Хотя можно использовать LinuxCNC с началом координат станка G53 за пределами программных пределов станка, если вы используете G28 или G30 без установки параметров, он по умолчанию переходит в начало координат. Это приведет к отключению концевиков пределов еще до достижения положения.

---

### 4.5.3 Пример схемы отдельного концевика исходного положения

В этом примере показаны минимальные и максимальные концевики пределов с отдельным концевиком исходного положения.

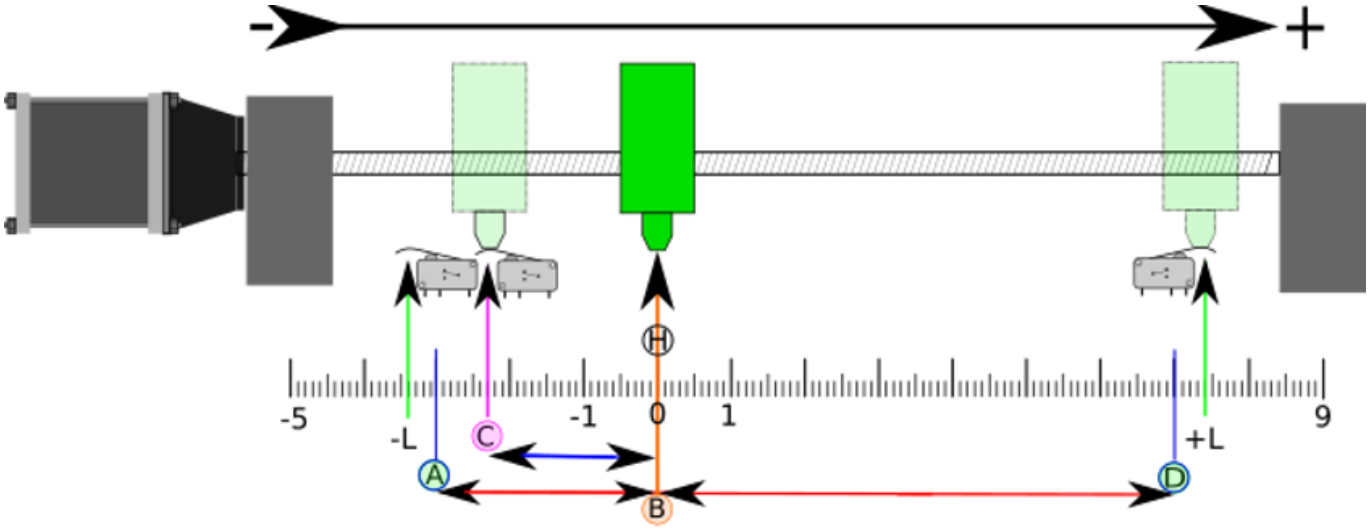



Figure 4.6: Демонстрационная схема отдельного концевика

- A — отрицательный программный предел
- B — это G53 начало координат станка
- C — точка срабатывания концевика исходного положения
- D — положительный программный предел
- H — конечное исходное положение (HOME) = 0 единиц
- -L и +L — точки срабатывания концевиков

- A<->B — отрицательные программные пределы (MIN_LIMITS) = -3 единицы
- B<->C — смещение home_offset (HOME_OFFSET) = -2,3 единицы
- B<->D — положительные программные пределы (MAX_LIMITS) = 7 единиц
- A<->D — общий ход = 10 единиц
- Расстояние между концевиками и программными пределами (-L<->A и D<-+L) в этом примере увеличено
- Обратите внимание, что между концевиками и реальным физическим жестким контактом существует расстояние для выбега после отключения усилителя.

**Note**
При приведении в исходное положение задается система координат G53, тогда как начало координат станка (нулевая точка) может находиться где угодно, а установка нулевой точки в отрицательном программном пределе делает все координаты G53 положительными, что, вероятно, легче всего запомнить. Сделайте это, установив MIN_LIMIT = 0 и убедитесь, что MAX_LIMIT положителен.

### 4.5.4 Пример схемы общего концевика предела/исходного положения

В этом примере показан концевик максимального предела и комбинированный концевик минимальног предела/исходного положения.

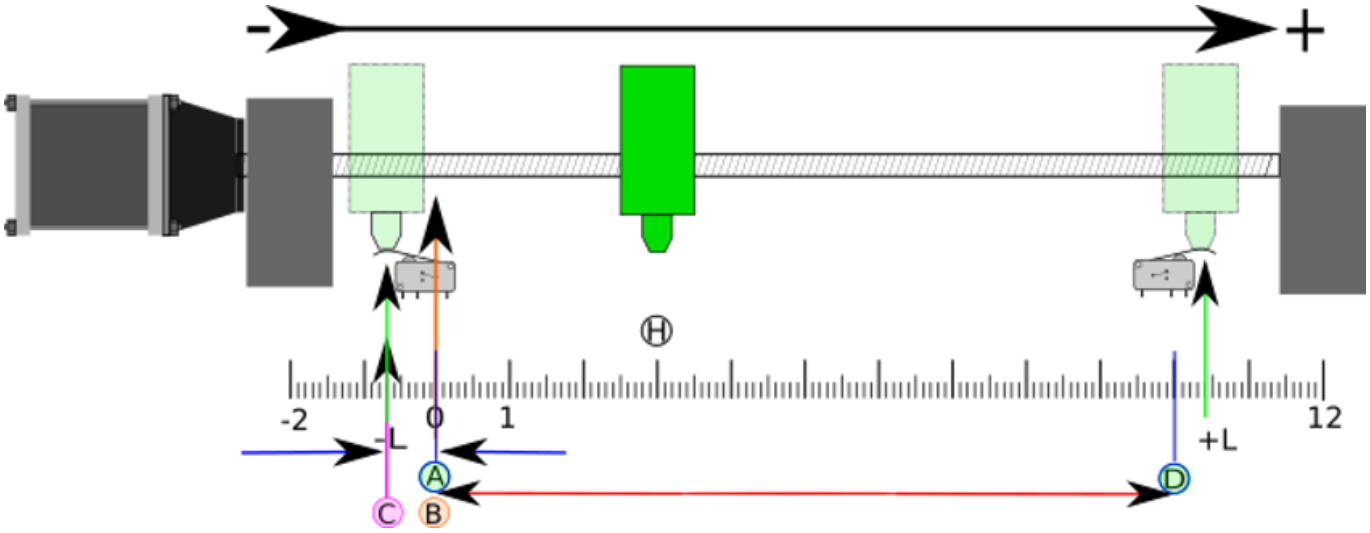



Figure 4.7: Демонстрационная схема общего концевика

- A — отрицательный программный предел.
- B — это G53 начало координат станка.
- C — точка срабатывания концевика исходного положения, общий с (-L) срабатыванием минимальног предела.
- D — положительный программный предел.
- H — конечное исходное положение (HOME) = 3 единицы.

- -L и +L — это точки срабатывания концевика предела.
- A<->B — отрицательные программные пределы (MIN_LIMITS) = 0 единиц.
- B<->C — это home_offset (HOME_OFFSET) = -0,7 единицы.
- B<->D — положительные программные пределы (MAX_LIMITS) 10 единиц.
- A<->D — общий ход = 10 единиц.
- Расстояние между концевиками пределов и программными пределами (-L<->A и D<->+L) в этом примере увеличено.
- Обратите внимание, что между концевиками и реальным физическим жестким контактом существует расстояние для выбега после отключения усилителя.

### 4.5.5 Последовательность приведения к исходному положению

Существует четыре возможных последовательности возврата в исходное положение, определяемые знаком HOME_SEARCH_VEL и HOME_LATCH_VEL, а также соответствующими параметрами конфигур как показано в следующей таблице. Существуют два основных условия: HOME_SEARCH_VEL и HOME_LATCH_VEL — это один и тот же знак или противоположные знаки. Более подробное описание того, что делает каждый параметр конфигурации, см. в следующем разделе.

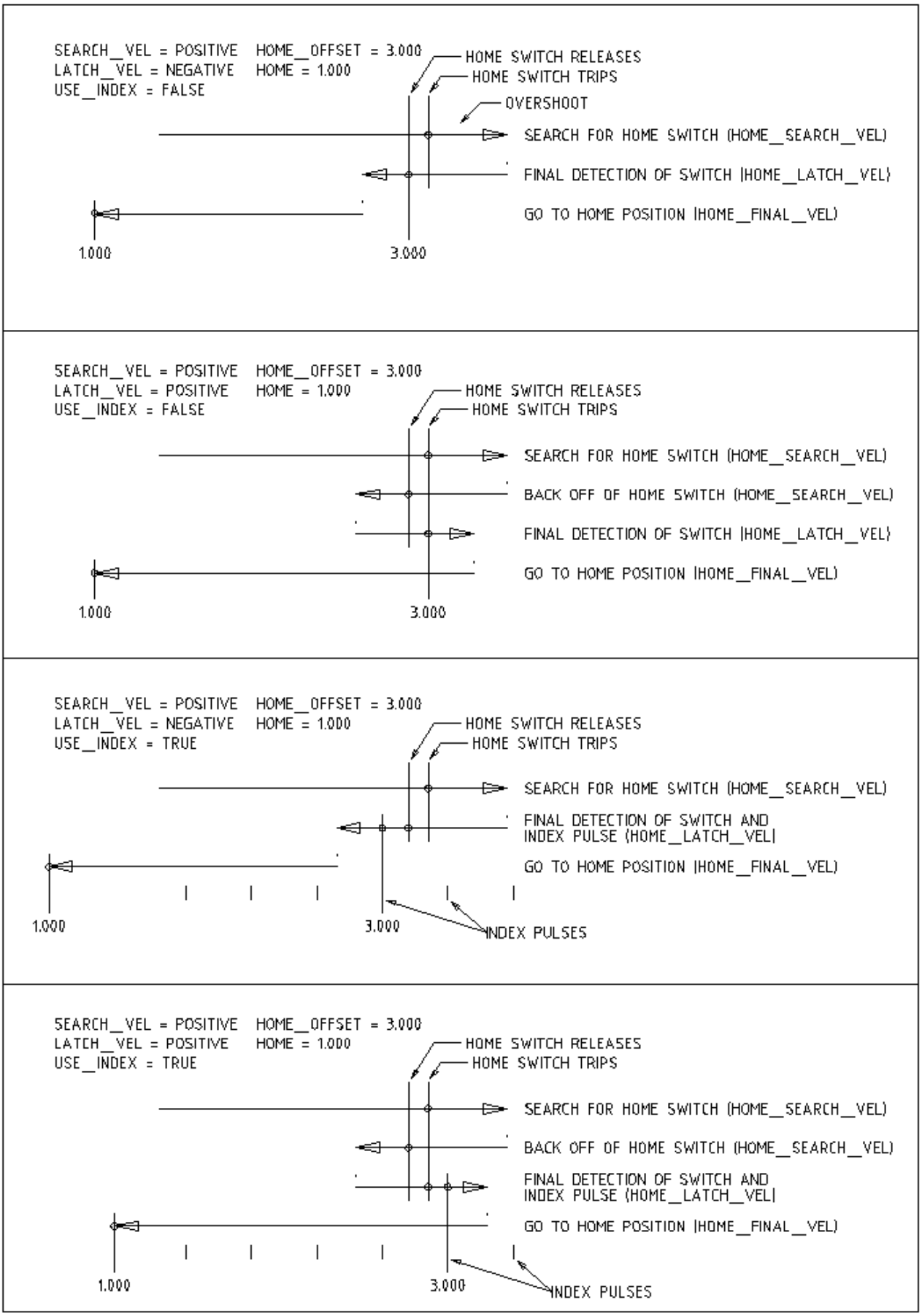



Figure 4.8: Последовательности приведения в исходное положение

## 4.5.6 Конфигурация

Следующее точно определяет, как ведет себя последовательность приведения в исходное положение. Они определены в разделе [JOINT_n] INI-файла.

| Тип последовательности | HOME_SEARCH_VEL | HOME_LATCH_VEL | HOME_USE_INDEX |
|---|---|---|---|
| Немедленно | 0 | 0 | НЕТ |
| Только метка | 0 | nonzero | ДА |
| Только концевик | nonzero | nonzero | НЕТ |
| Концевик и Метка | nonzero | nonzero | ДА |

**Note**
Любые другие комбинации могут привести к ошибке.

### 4.5.6.1 HOME_SEARCH_VEL

Эта переменная имеет единицы измерения — единицы-станка в секунду.

Значение по умолчанию равно нулю. Значение 0 заставляет LinuxCNC предполагать, что концевика исходного положения нет; этап поиска исходного положения пропускается.

Если HOME_SEARCH_VEL не равно нулю, LinuxCNC предполагает, что существует концевик исходного положения. Он начинает с проверки, не сработал ли уже концевик исходного положения. В случае срабатывания он откатывает на HOME_SEARCH_VEL. Направление отката противоположно знаку HOME_SEARCH_VEL. Затем он ищет концевик исходного положения, перемещаясь в направлени указанном знаком HOME_SEARCH_VEL, со скоростью, определяемой его абсолютным значением. При обнаружении концевика исходного положения сочленение остановится как можно быстрее, но всегда будет некоторый выкат. Величина выката зависит от скорости. Если он слишком высок, сочленение может проскочить настолько, что ударится о концевой выключатель или врежется в конец хода. С другой стороны, если HOME_SEARCH_VEL слишком низкий, приведение к исходному положению может занять много времени.

### 4.5.6.2 HOME_LATCH_VEL

Эта переменная имеет единицы измерения — единицы-станка в секунду.

Указывает скорость и направление, которые LinuxCNC использует при окончательном точном определении концевика исходного положения (если он присутствует) и местоположения индексного импульса (если он присутствует). Обычно она будет медленнее, чем скорость поиска, чтобы максимизировать точность. Если HOME_SEARCH_VEL и HOME_LATCH_VEL имеют одинаковый знак, то фаза фиксации выполняется при движении в том же направлении, что и фаза поиска. (В этом случае LinuxCNC сначала откатывает от концевика, а затем снова движется к нему со скоростью фиксации.) Если HOME_SEARCH_VEL и HOME_LATCH_VEL имеют противоположные знаки, фаза фиксации выполняется при движении в направлении, противоположном фазе поиска. Это означает, что LinuxCNC зафиксирует первый импульс после того, как он уйдет с концевика. Если HOME_SEARCH_VEL равен нулю (это означает, что концевик исходной позиции отсутствует), и этот параметр не равен нулю, LinuxCNC переходит к поиску индексного импульса. Если HOME_SEARCH_VEL не равно нулю и этот параметр равен нулю, это ошибка и операция возврата в исходное положение завершится неудачей. Значение по умолчанию равно нулю.

#### 4.5.6.3 HOME_FINAL_VEL

Эта переменная имеет единицы измерения — единицы-станка в секунду.

Этот параметр определяет скорость, которую LinuxCNC использует при переходе из позиции HOME_OFFSET в позицию HOME. Если HOME_FINAL_VEL отсутствует в INI-файле, то для выполнения этого движения используется максимальная скорость сочленения. Значение должно быть положитель числом.

#### 4.5.6.4 HOME_IGNORE_LIMITS

Этот параметр может иметь значения YES / NO. Значение по умолчанию для этого параметра — NO. Этот флаг определяет, будет ли LinuxCNC игнорировать вход концевика предела для этого сочленения во время возврата в исходное положение. Эта настройка не будет игнорировать входы пределов для других сочленений. Если у вас нет отдельного концевика исходного положения, установите этот параметр в YES и подключите сигнал концевика предела к входу концевика исходного положения сочленения в HAL. LinuxCNC будет игнорировать вход концевика для этого сочленения во время возврата в исходное положение. Чтобы использовать только один вход для всех исходных положений и пределов, вам придется заблокировать сигналы пределов сочленений, не возвращающихся в исходное положение в HAL, и приводить в исходное положение одно сочленение за раз.

#### 4.5.6.5 HOME_USE_INDEX

Указывает, есть ли индексный импульс. Если флаг равен true (HOME_USE_INDEX = YES), Linux-CNC будет фиксировать фронт индексного импульса. Если значение равно false, LinuxCNC будет фиксировать либо фронт, либо срез концевика исходного положения (в зависимости от знаков HOME_SEARCH_VEL и HOME_LATCH_VEL). Значение по умолчанию — NO.

---

**Note**
HOME_USE_INDEX требует соединения в вашем файле HAL к «joint.n.index-enable» из «encoder.n.index-enable».

---

#### 4.5.6.6 HOME_INDEX_NO_ENCODER_RESET

По умолчанию NO. Используйте YES, если энкодер, используемый для этого сочленения, не сбрасывает свой счетчик при обнаружении индексного импульса после активации контакта in-dex_enable сочленения HAL. Применимо только для HOME_USE_INDEX = YES.

#### 4.5.6.7 HOME_OFFSET

Этот параметр определяет положение начальной нулевой точки системы координат станка G53. Это расстояние (смещение) в единицах сочленения от начала координат станка до точки срабатывания концевика исходного положения или индексного импульса. После обнаружения точки срабатывания концевика/индексного импульса LinuxCNC устанавливает положение координат сочленения на HOME_OFFSET, тем самым определяя начало координат, от которого происходит программный предел. Значение по умолчанию равно нулю.

---

**Note**
Местоположение концевика исходного положения, указанное переменной HOME_OFFSET, может находиться внутри или за пределами программных пределов. Они будут использоваться совместно или внутри аппаратных концевиков пределов.

---

#### 4.5.6.8 HOME

Положение, в которое перейдет сочленение по завершению последовательности приведения в исходное положение. После обнаружения срабатывания концевика исходного положения или концевика исходного положения, а затем индексного импульса (в зависимости от конфигурации) и установки координаты этой точки на HOME_OFFSET, LinuxCNC выполняет переход на HOME в качестве последнего шага процесса возврата в исходное положение. Значение по умолчанию равно нулю. Обратите внимание, что даже если этот параметр такой же, как HOME_OFFSET, при остановке сочленение немного выйдет за пределы зафиксированного положения. Поэтому в это время всегда будет небольшое перемещение (если только HOME_SEARCH_VEL не равен нулю и весь этап поиска/фиксации не был пропущен). Это последнее движение будет выполнено с максимальной скоростью сочленения, если не установлено значение HOME_FINAL_VEL.

---

**Note**
Разница между *HOME_OFFSET* и *HOME* заключается в том, что *HOME_OFFSET* сначала устанавливает исходное местоположение и шкалу на станке, применяя значение *HOME_OFFSET* к месту, где был найдено исходное положение, а затем *HOME* указывает, куда должно переместиться сочленение на этой шкале.

---

#### 4.5.6.9 HOME_IS_SHARED

Если для этого сочленения нет отдельного входа концевика исходного положения, но к одному контакту подключено несколько концевиков мгновенного действия, установите это значение равным 1, чтобы предотвратить запуск возврата в исходное состояние, если один из общих переключателей уже замкнут. Установите это значение на 0, чтобы разрешить возврат в исходное положение, даже если переключатель уже замкнут.

#### 4.5.6.10 HOME_ABSOLUTE_ENCODER

Используется для абсолютных энкодеров. Когда делается запрос на приведение в исходное положение сочленения, текущая позиция сочленения устанавливается на значение «[JOINT_n]HOME_

Окончательный переход в позицию «[JOINT_n]HOME» не является обязательным в соответствии с настройкой «HOME_ABSOLUTE_ENCODER»:

```
HOME_ABSOLUTE_ENCODER = 0 (b''Пb''b''ob'' b''yb''b''мb''b''ob''b''лb''b''чb''b''ab''b''нb'' ↩
    b''иb''b''юb'') b''cb''b''ob''b''чb''b''лb''b''eb''b''нb''b''eb''b''нb''b''иb''b''eb'' b ↩
    ''нb''b''eb'' b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''yb''b''eb''b''тb'' b' ↩
    'ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ыb''b''йb'' b''эb''b''нb''b' ↩
    'кb''b''ob''b''дb''b''eb''b''pb''
HOME_ABSOLUTE_ENCODER = 1 b''Ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ыb''b' ↩
    'йb'' b''эb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb'', b''ob''b''кb''b''ob''b''нb''b' ↩
    'чb''b''ab''b''тb''b''eb''b''лb''b''ьb''b''нb''b''ыb''b''йb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''xb''b''ob''b''дb'' b''вb'' [JOINT_n]HOME
HOME_ABSOLUTE_ENCODER = 2 b''Ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ыb''b' ↩
    'йb'' b''эb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb'', NO (b''нb''b''eb''b''тb'') b' ↩
    'ob''b''кb''b''ob''b''нb''b''чb''b''ab''b''тb''b''eb''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'гb''b''ob'' b''пb''b''eb''b''pb''b''eb''b''xb''b''ob''b''дb''b''ab'' b''вb'' [JOINT_n] ↩
    HOME
```

---

**Note**
Параметр HOME_IS_SHARED молча игнорируется.

---

**Note**
Запрос приведения сочленения в исходное положение молча игнорируется.

#### 4.5.6.11 HOME_SEQUENCE

Используется для определения последовательности приведения множества сочленений в исходное положение **HOME ALL** и обеспечения порядка возвращения в исходное положение (например, Z не может быть возвращен в исходное положение, если X еще не установлен в исходное положение). Сочленение может быть приведено в исходное положение после того, как все сочленения с меньшим (абсолютным значением) HOME_SEQUENCE уже приведены и находятся в положении HOME_OFFSET. Если два сочленения имеют одинаковую HOME_SEQUENCE, они могут быть приведены одновременно.

**Note**
Если HOME_SEQUENCE не указан, сочленение не будет приведено в исходное положение последовательностью **HOME ALL** (но может быть приведено отдельными командами приведения конкретного сочленения).

Начальное число HOME_SEQUENCE может быть 0, 1 (или -1). Абсолютное значение порядковых номеров должно увеличиваться на единицу — пропуск порядковых номеров не поддерживается. Если порядковый номер опущен, возврат в исходное положение **HOME ALL** прекратится после завершения последнего допустимого порядкового номера.

**Отрицательные** значения HOME_SEQUENCE указывают, что соединения в последовательности должны **синхронизировать окончательное перемещение** с [JOINT_n]HOME, ожидая, пока все сочленения в последовательности не будут готовы. Если какое-либо сочленение имеет **отрицател** значение HOME_SEQUENCE, то все сочленения с одинаковым абсолютным значением (положительны или отрицательным) значения элемента HOME_SEQUENCE будут синхронизировать окончательное перемещение.

**Отрицательное** значение HOME_SEQUENCE также применяется к командам приведения одного сочленения в исходное положение. Если значение HOME_SEQUENCE **отрицательное**, все сочленени имеющие одинаковое абсолютное значение этого HOME_SEQUENCE, будут **приведены в исходное положение вместе с синхронизированным окончательным перемещением**. Если значение HOME_SEQUENCE равно нулю или положительному значению, команда для исходного сочленения будет приводить в исходное положение только указанное соединение.

Режим медленной подачи сочленений с отрицательным значением HOME_SEQUENCE запрещен. В обычных портальных приложениях такое перемещение может привести к перекосу (сдвигу). Обратите внимание, что обычное перемещение по глобальным координатам всегда доступно по приведению станка в исходное положение.

Примеры для системы с тремя сочленениями

Две последовательности (0,1), без синхронизации

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 1
[JOINT_2]HOME_SEQUENCE = 1
```

Две последовательности, сочленения 1 и 2 синхронизированы

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

При смешанных положительных и отрицательных значениях сочленения 1 и 2 синхронизированы

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE =  1
```

Одна последовательность, без синхронизации

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE =  0
[JOINT_2]HOME_SEQUENCE =  0
```

Одна последовательность, все сочленения синхронизированы

```
[JOINT_0]HOME_SEQUENCE = -1
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

#### 4.5.6.12 VOLATILE_HOME

Если настройка в true, это сочленение становится не в исходном положении всякий раз, когда станок переходит в состояние ВЫКЛ. Это подходит для любого сочленения, которое не сохраняет положение при выключенном приводе сочленения. Это может потребоваться некоторым шаговым приводам, особенно микрошаговым.

#### 4.5.6.13 LOCKING_INDEXER

Если это сочленение представляет собой делительную головку с блокировкой, оно разблокируется перед возвратом в исходное положение и зафиксируется после этого.

#### 4.5.6.14 Немедленно исходная позиция

Если сочленение не имеет концевиков исходного положения или не имеет логического исходного положения, как, например, поворотное сочленение, и вы хотите, чтобы это сочленение возвращалось в исходное положение в текущем положении при нажатии кнопки "Home All" в графическом интерфейсе AXIS, то необходимы следующие записи INI для этого сочленения.

```
HOME_SEARCH_VEL = 0
HOME_LATCH_VEL = 0
HOME_USE_INDEX = NO
HOME_OFFSET = 0 (b''Иb''b''лb''b''иb'' b''cb''b''мb''b''eb''b''щb''b''eb''b''нb''b''иb''b' ↩
    'eb'' b''иb''b''cb''b''хb''b''ob''b''дb''b''нb''b''ob''b''йb'' b''пb''b''ob''b''зb''b' ↩
    'иb''b''цb''b''иb''b''иb'' (HOME))
HOME_SEQUENCE = 0 (b''иb''b''лb''b''иb'' b''дb''b''pb''b''уb''b''гb''b''ob''b''йb'' b''дb'' ↩
    b''eb''b''йb''b''cb''b''тb''b''вb''b''иb''b''тb''b''eb''b''лb''b''ьb''b''нb''b''ыb''b' ↩
    'йb'' b''пb''b''ob''b''pb''b''яb''b''дb''b''кb''b''ob''b''вb''b''ыb''b''йb'' b''нb''b' ↩
    'ob''b''мb''b''eb''b''pb'')
```

---

**Note**
Значения по умолчанию для неуказанных HOME_SEARCH_VEL, HOME_LATCH_VEL, HOME_USE_INDEX, HOME и HOME_OFFSET равны **нулю**, поэтому их можно опустить при запросе немедленного возврата в исходное положение. Обычно следует включать действительный номер HOME_SEQUENCE, поскольку пропуск HOME_SEQUENCE исключает сочленение из поведения **HOME ALL**, как отмечено выше.

---

#### 4.5.6.15 Inhibiting Homing

Контакт HAL (motion.homing-inhibit) предназначен для запрета инициирования возврата в исходное положение как для ”Home All”, так и для отдельного сочленения.

Некоторые системы используют преимущества синхронизации окончательных перемещений сочленен в исходное положение, контролируемых отрицательными [JOINT_N]HOME_SEQUENCE= элементами INI файла. По умолчанию условия синхронизации запрещают медленную подачу **сочленения** перед перемещением в исходное положение, чтобы предотвратить перемещение **сочленения**, которое может привести к смещению станка (например, портальная стойка).

Системный разработчике может разрешить медленную подачу **сочленения** перед возвратом в исходное положение с помощью логики HAL, которая переключает элементы [JOINT_N]HOME_SEQUE Эта логика также должна активировать вывод **motion.homing-inhibit**, чтобы гарантировать, что возврат в исходное положение не будет случайно инициирован, когда разрешена медленная подача **сочленения**.

Пример: Синхронизированные сочленения 0,1 с использованием отрицательной последовательности (-1) для синхронного возврата в исходное положение с концевиком (allow_jjog), который выбирает положительную последовательность (1) для индивидуального перемещения **сочленения** перед возвратом в исходное положение (частичный код HAL):

```
loadrt mux2           names=home_sequence_mux
loadrt conv_float_s32 names=home_sequence_s32
setp home_sequence_mux.in0 -1
setp home_sequence_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: pin created by a virtual panel or hardware switch
net hsequence_select <= allow_jjog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

---

**Note**
Контакты INI HAL (подобные ini.N.home_sequence) недоступны до тех пор, пока не запустится milltask, поэтому выполнение вышеуказанных команд HAL следует отложить с помощью файла HAL postgui или отложенного сценария [APPLICATION]APP=.

---

---

**Note**
Синхронизация медленной подачи сочленения для нескольких сочленений в реальном времени требует дополнительных соединений HAL для контактов медленной подачи типа РГИ (MPG) (`joint.N.enable`, `joint.N.scale`, `joint.N.counts`). .

---

Пример конфигурации моделирования (gantry_jjog.ini), демонстрирующий медленную подачу сочленения при использовании отрицательных последовательностей приведения в исходное положени находится в каталоге: configs/sim/axis/gantry/.

# 4.6 Конфигурация токарного станка

## 4.6.1 Default Plane

When LinuxCNC's interpreter was first written, it was designed for mills. That is why the default plane is XY (G17). A normal lathe only uses the XZ plane (G18). To change the default plane place the following line in the INI file in the RS274NGC section.

```
RS274NGC_STARTUP_CODE = G18
```

The above can be overwritten in a G-code program so always set important things in the preamble of the G-code file.

## 4.6.2 INI Settings

The following INI settings are needed for lathe mode in Axis in addition to or replacing normal settings in the INI file. These historical settings use identity kinematics (trivkins) and *three* joints (0,1,2) corresponding to coordinates x, y, z. The joint 1 for the unused y axis is required but not used in these historical configurations. Simulated lathe configs may use these historical settings. GMOCCAPY also uses the mentioned settings, but does offer additional settings, check the GMOCCAPY section for details.

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...

[KINS]
KINEMATICS = trivkins
JOINTS = 3

[TRAJ]
COORDINATES = X Z
...

[JOINT_0]
...
[JOINT_2]
...
[AXIS_X]
...
[AXIS_Z]
...
```

With joints_axes incorporation, a simpler configuration can be made with just the two required joints by specifying trivkins with the *coordinates*= parameter:

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...

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

[TRAJ]
COORDINATES = X Z
```

```
...

[JOINT_0]
...
[JOINT_1]
...
[AXIS_X]
...
[AXIS_Z]
...
```

## 4.7 Stepper Quickstart

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.

### 4.7.1 Latency Test

The Latency Test determines how late your computer processor is in responding to a request. Some hardware can interrupt the processing which could cause missed steps when running a CNC machine. This is the first thing you need to do. Follow the instructions here to run the latency test.

### 4.7.2 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.

### 4.7.3 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.

### 4.7.4 Machine Information

Gather the information about each axis of your machine.

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.

| Axis | Drive Type | Step Time (ns) | Step Space (ns) | Dir. Hold (ns) | Dir. Setup (ns) |
|---|---|---|---|---|---|
| X | | | | | |
| Y | | | | | |
| Z | | | | | |
| | | | | | |

### 4.7.5 Pinout Information

Gather the information about the connections from your machine to the PC parallel port.

| Output Pin | Typ. Function | If Different | Input Pin | Typ. Function | If Different |
|---|---|---|---|---|---|
| 1 | E-Stop Out | | 10 | X Limit/Home | |
| 2 | X Step | | 11 | Y Limit/Home | |
| 3 | X Direction | | 12 | Z Limit/Home | |
| 4 | Y Step | | 13 | A Limit/Home | |
| 5 | Y Direction | | 15 | Probe In | |
| 6 | Z Step | | | | |
| 7 | Z Direction | | | | |
| 8 | A Step | | | | |
| 9 | A Direction | | | | |
| 14 | Spindle CW | | | | |
| 16 | Spindle PWM | | | | |
| 17 | Amplifier Enable | | | | |

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.

### 4.7.6 Mechanical Information

Gather information on steps and gearing. The result of this is steps per user unit which is used for SCALE in the INI file.

| Axis | Steps/Rev. | Micro Steps | Motor Teeth | Leadscrew Teeth | Leadscrew Pitch |
|---|---|---|---|---|---|
| X | | | | | |
| Y | | | | | |
| Z | | | | | |
| | | | | | |

- *Steps per revolution* - is how many stepper-motor-steps it takes to turn the stepper motor one revolution. Typical is 200.
- *Micro Steps* - is how many steps the drive needs to move the stepper motor one full step. If microstepping is not used, this number will be 1. If microstepping is used the value will depend on the stepper drive hardware.
- *Motor Teeth and Leadscrew Teeth* - is if you have some reduction (gears, chain, timing belt, etc.) between the motor and the leadscrew. If not, then set these both to 1.
- *Leadscrew Pitch* - is how much movement occurs (in user units) in one leadscrew turn. If you're setting up in inches then it is inches per turn. If you're setting up in millimeters then it is millimeters per turn.

The net result you're looking for is how many CNC-output-steps it takes to move one user unit (inches or mm).

**Example 4.1** Units inches

```
Stepper         = 200 steps per revolution
Drive           =  10 micro steps per step
Motor Teeth     =  20
Leadscrew Teeth =  40
Leadscrew Pitch =   0.2000 inches per turn
```

From the above information, the leadscrew moves 0.200 inches per turn. - The motor turns 2.000 times per 1 leadscrew turn. - The drive takes 10 microstep inputs to make the stepper step once. - The drive needs 2000 steps to turn the stepper one revolution.

So the scale needed is:

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

**Example 4.2** Units mm

```
    Stepper         = 200 steps per revolution
    Drive           =   8 micro steps per step
    Motor Teeth     =  30
    Leadscrew Teeth =  90
    Leadscrew Pitch =   5.00 mm per turn
```

From the above information: - The leadscrew moves 5.00 mm per turn. - The motor turns 3.000 times per 1 leadscrew turn. - The drive takes 8 microstep inputs to make the stepper step once. - The drive needs 1600 steps to turn the stepper one revolution.

So the scale needed is:

$$\frac{200\,full\,steps}{1\,rev} x \frac{8\,microsteps}{1\,step} x \frac{3\,revs}{1\,leadscrew\,rev} x \frac{1\,leadscrew\,rev}{5.00\text{mm}} = \frac{960\,steps}{1\,mm}$$

## 4.8 Конфигурация шагового двигателя

### 4.8.1 Введение

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.

It is one of the simpler setups, because the motors run open-loop (no feedback comes back from the motors), yet the system needs to be configured properly so the motors don’t stall or lose steps.

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.

### 4.8.2 Maximum step rate

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.

If your stepper driver can accept quadrature input, use this mode. With a quadrature signal, one step is possible for each BASE_PERIOD, doubling the maximum step rate.

The other remedies are to decrease one or more of: the BASE_PERIOD (setting this too low will cause the machine to become unresponsive or even lock up), the INPUT_SCALE (if you can select different step sizes on your stepper driver, change pulley ratios, or leadscrew pitch), or the MAX_VELOCITY and STEPGEN_MAXVEL.

If no valid combination of BASE_PERIOD, INPUT_SCALE, and MAX_VELOCITY is acceptable, then consider using hardware step generation (such as with the LinuxCNC-supported Universal Stepper Controller, Mesa cards, and others).

### 4.8.3 Pinout

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.

Как описано во введении и руководстве по HAL, внутри HAL у нас есть сигналы, контакты и параметры.

---

**Note**
We are presenting one axis to keep it short, all others are similar.

---

The ones relevant for our pinout are:

```
signals: 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.

#### 4.8.3.1 Standard Pinout HAL

This file contains several HAL commands, and usually looks like this:

```
# standard pinout config file for 3-axis steppers
# using a parport for I/O
#
# first load the parport driver
loadrt hal_parport cfg="0x0378"
#
# next connect the parport functions to threads
# read inputs first
addf parport.0.read base-thread 1
# write outputs last
addf parport.0.write base-thread -1
#
```

```
# finally connect physical pins to the signals
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

# create a signal for the estop loopback
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

# create signals for tool loading loopback
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed

# connect "spindle on" motion controller pin to a physical pin
net spindle-on spindle.0.on => parport.0.pin-09-out

###
### You might use something like this to enable chopper drives when machine ON
### the Xen signal is defined in core_stepper.hal
###

# net Xen => parport.0.pin-01-out

###
### If you want active low for this pin, invert it like this:
###

# setp parport.0.pin-01-out-invert 1

###
### A sample home switch on the X axis (axis 0).  make a signal,
### link the incoming parport pin to the signal, then link the signal
### to LinuxCNC's axis 0 home switch input pin.
###

# net Xhome parport.0.pin-10-in => joint.0.home-sw-in

###
### Shared home switches all on one parallel port pin?
### that's ok, hook the same signal to all the axes, but be sure to
### set HOME_IS_SHARED and HOME_SEQUENCE in the INI file.
###

# 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

###
### Sample separate limit switches on the X axis (axis 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

###
### Just like the shared home switches example, you can wire together
### limit switches.  Beware if you hit one, LinuxCNC will stop but can't tell
### you which switch/axis has faulted.  Use caution when recovering from this
### extreme position to avoid a hard stop.
```

```
###

# net Xlimits parport.0.pin-13-in => joint.0.neg-lim-sw-in joint.0.pos-lim-sw-in
```

The lines starting with # are comments, and their only purpose is to guide the reader through the file.

#### 4.8.3.2 Обзор

There are a couple of operations that get executed when the standard_pinout.hal gets executed/interpreted:

- The Parport driver gets loaded (see the Parport Chapter for details).
- The read & write functions of the parport driver get assigned to the base thread [2].
- The step & direction signals for axes X, Y, Z get linked to pins on the parport.
- Further I/O signals get connected (estop loopback, toolchanger loopback).
- A spindle-on signal gets defined and linked to a parport pin.

#### 4.8.3.3 Changing the 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.

If you want for example to change the pin for the X-axis Step & Directions signals, all you need to do is to change the number in the *parport.0.pin-XX-out* name:

```
net Xstep parport.0.pin-03-out
net Xdir  parport.0.pin-02-out
```

can be changed to:

```
net Xstep parport.0.pin-02-out
net Xdir  parport.0.pin-03-out
```

or basically any other *out* pin you like.

Hint: make sure you don't have more than one signal connected to the same pin.

#### 4.8.3.4 Changing polarity of a signal

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-out-invert TRUE
```

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

#### 4.8.3.5 Adding PWM Spindle Speed Control

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
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.

#### 4.8.3.6 Adding an enable signal

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*.

To connect them use the following example:

```
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.

#### 4.8.3.7 External ESTOP button

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

## 4.9 Диагностика шагового двигателя

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.

### 4.9.1 Common Problems

#### 4.9.1.1 Stepper Moves One Step

The most common reason in a new installation for a stepper motor not to move is that the step and direction signals are exchanged. If you press the jog forward and jog backward keys, alternately , and the stepper moves one step each time, and in the same direction, there is your clue.

#### 4.9.1.2 No Steppers Move

Many drives have an enable pin or need a charge pump to enable the output.

#### 4.9.1.3 Distance Not Correct

If you command the axis to move a specific distance and it does not move that distance, then your scale setting is wrong.

### 4.9.2 Error Messages

#### 4.9.2.1 Following Error

The concept of a following error is strange when talking about stepper motors. Since they are an open loop system, there is no position feedback to let you know if you actually are out of range. LinuxCNC calculates if it can keep up with the motion called for, and if not, then it gives a following error. Following errors usually are the result of one of the following on stepper systems.

- FERROR too small
- MIN_FERROR too small
- MAX_VELOCITY too fast
- MAX_ACCELERATION too fast
- BASE_PERIOD set too long
- 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.

If you added backlash you need to increase the STEPGEN_MAXACCEL up to double the MAX_ACCELERATIO in the AXIS section of the INI file for each axis you added backlash to. LinuxCNC uses "extra acceleration" at a reversal to take up the backlash. Without backlash correction, step generator acceleration can be just a few percent above the motion planner acceleration.

#### 4.9.2.2 RTAPI Error

When you get this error:

```
RTAPI: ERROR: 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 tracks the number of CPU cycles between invocations of the real-time thread. If some element of your hardware is causing delays or your realtime threads are set too fast you will get this error.

---

**Note**
This error is only displayed once per session. If you had your BASE_PERIOD too low you could get hundreds of thousands of error messages per second if more than one was displayed.

---

### 4.9.3 Testing

#### 4.9.3.1 Параметры шагового импульса

If you are seeing an axis ending up in the wrong location over multiple moves, it is likely that you do not have the correct direction hold times or step timing for your stepper drivers. Each direction change may be losing a step or more. If the motors are stalling, it is also possible you have either the MAX_ACCELERATION or MAX_VELOCITY set too high for that axis.

The following program will test the Z axis configuration for proper setup. Copy the program to your \~/emc2/nc_files directory and name it TestZ.ngc or similar. Zero your machine with Z = 0.000 at the table top. Load and run the program. It will make 200 moves back and forth from 0.5 to 1”. If you have a configuration issue, you will find that the final position will not end up 0.500” that the axis window is showing. To test another axis just replace the Z with your axis in the G0 lines.

```
( test program to see if Z axis loses position )
( msg, test 1 of Z axis configuration )
G20 #1000=100 ( loop 100 times )
( this loop has delays after moves )
( tests acc and velocity settings )
o100 while [#1000]
   G0 Z1.000
   G4 P0.250
   G0 Z0.500
   G4 P0.250
   #1000 = [#1000 - 1]
o100 endwhile
( msg, test 2 of Z axis configuration S to continue)
M1 (stop here)
#1000=100 ( loop 100 times )
( the next loop has no delays after moves )
( tests direction hold times on driver config and also max accel setting )
o101 while [#1000]
   G0 Z1.000
   G0 Z0.500
   #1000 = [#1000 - 1]
o101 endwhile
( msg, Done...Z should be exactly .5” above table )
M2
```

## 4.10 Фильтрующие программы

### 4.10.1 Введение

Большинство экранов LinuxCNC имеют возможность отправлять загруженные файлы через *filter program* или использовать программу-фильтр для создания G-кода. Такой фильтр может выполнять любую желаемую задачу: что-то простое, например проверка того, что файл заканчивается на «M2», или что-то более сложное, например создание G-кода из изображения.

### 4.10.2 Настройка INI для программных фильтров

Раздел *[FILTER]* INI-файла управляет работой фильтров. Сначала для каждого типа файла напишите строку *PROGRAM_EXTENSION*. Затем укажите программу, которая будет выполняться для каждого типа файла. Этой программе присваивается имя входного файла в качестве первого аргумента, и она должна записать код rs274ngc в стандартный вывод. Этот вывод будет отображаться

в текстовой области, просматриваться в области отображения и выполняться LinuxCNC при нажатии кнопки *Run*. Следующие строки добавляют поддержку конвертера *image-to-gcode*, включенно в LinuxCNC:

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

Также возможно указать интерпретатор:

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

Таким образом, любой скрипт Python может быть открыт, а его выходные данные обрабатываются как G-код. Один из таких примеров скрипта доступен по адресу *nc_files/holecircle.py*. Этот скрипт создает G-код для сверления серии отверстий по окружности.

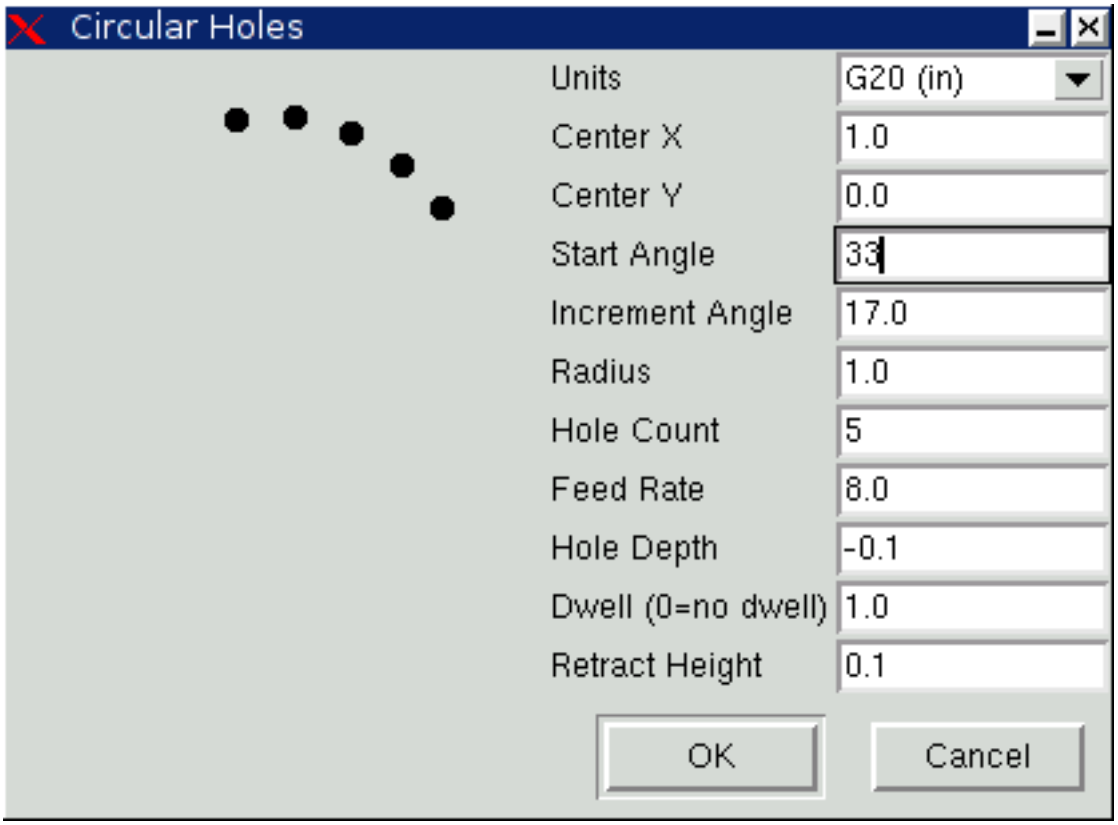



Figure 4.9: Отверстия по кругу

Если программа фильтра отправляет в stderr строки вида:

```
FILTER_PROGRESS=10
```

Он установит индикатор выполнения экрана на заданный (в данном случае 10) процент. Эту функцию следует использовать любому фильтру, работающему в течение длительного времени.

### 4.10.3 Создание программ фильтров на основе Python

Вот очень простой пример механики фильтрации: при запуске экрана LinuxCNC, предлагающего программную фильтрацию, он будет генерировать и записывать строку G-кода каждую 100ю

долю секунды в стандартный вывод. Он также отправляет сообщение о ходе выполнения в стандартный поток ошибок UNIX. Если произошла ошибка, он выведет сообщение об ошибке и завершит работу с кодом выхода 1.

```
import time
import sys

for i in range(0,100):
    try:
        # simulate calculation time
        time.sleep(.1)

        # output a line of G-code
        print('G0 X1', file=sys.stdout)

        # update progress
        print('FILTER_PROGRESS={}'.format(i), file=sys.stderr)
    except:
        # This causes an error message
        print('Error; But this was only a test', file=sys.stderr)
        raise SystemExit(1)
```

Вот похожая программа, но она действительно умеет фильтровать. Она открывает диалоговое окно PyQt5 с кнопкой отмены. Затем она считывает программу построчно и передает ее на стандартный вывод. По ходу дела она обновляет любой процесс, прослушивающий стандартный вывод ошибок.

```
#!/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: '{}' doesn't exist:".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:
            # get next line of code
            codeLine = self.temp[self.line]

            # process the line somehow

            # push out processed code
            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 end with no error/error message
            if self._percentDone >= 99:
                print('FILTER_PROGRESS=-1', file=sys.stderr)
                self.infile.close()
                raise SystemExit(0)

        except Exception as e:
            # This provides an error message
            print(('Something bad happened:',e), file=sys.stderr)
            # this signals the error message should be shown
            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_() )
```

# Chapter 5

# HAL (Уровень аппаратной абстракции

## 5.1 HAL Введение

LinuxCNC — это взаимодействие с оборудованием. Но немногие пользователи имеют одинаковые характеристики оборудования — похожие, но не одинаковые. И даже для одного и того же оборудования могут существовать разные способы его использования, скажем, для разных материалов или с разными фрезерами, что потребует адаптации к управлению уже работающей системой. Абстракция была необходима, чтобы упростить настройку LinuxCNC для широкого спектра аппаратны устройств. На самом высоком уровне это может быть просто способ загрузить и соединить ряд *строительных блоков* для сборки сложной системы.

В этой главе рассматривается уровень аппаратной абстракции. Вы увидите, что многие из строительных блоков действительно являются драйверами для аппаратных устройств. Однако HAL может делать больше, чем просто настраивать драйверы оборудования.

### 5.1.1 HAL Обзор

Уровень аппаратной абстракции (или со ссылкой на фильм Космическая одиссея 2001 просто ”HAL”) — это программное обеспечение, позволяющее

- обеспечить инфраструктуру для связи с и между многими программными и аппаратными компонентами системы.
- при необходимости обрабатывать и/или переопределять эту информацию при ее передаче от компонента к компоненту.

Сама по себе это link: Связующее программное обеспечение не содержит информации о ее применении на ЧПУ. Поиск в Интернете, например, нашел астрономическое приложение для управления телескопами с помощью LinuxCNC. Двигатели перемещают телескоп в правильное положение, и он должен знать, как сопоставить двигательную активность с эффектом этого позиционирования в реальном мире. Такая синхронизация положений двигателя с реальными положениями напоминает то, что должны делать станки с ЧПУ или космические корабли.

Любой контролер станка должен знать:

- о его внутреннем состоянии и о том, как оно соотносится с окружающей средой (координаты станка, состояние переключателей/регуляторов),
- как исполнительные механизмы должны изменить это состояние,
- как разрешить обновление внутреннего состояния датчиков (энкодеров, зондов).

Уровень HAL состоит из частей (называемых «компонентами»), которые

- связаны друг с другом, например, для обновления данных о местоположении или для того, чтобы алгоритм планирования сообщал двигателям о следующем шаге.
- может знать, как взаимодействовать с оборудованием,
- может просто обрабатывать входящие данные и предоставлять выходные данные другим компонента
- всегда периодически выполняются либо
  - с очень высокой частотой в несколько микросекунд (мкс) времени выполнения, называемой базовым потоком, например, для
    1. дать шаговому двигателю триггер, чтобы сделать шаг вперед или
    2. считать позицию, представленную энкодером.
  - с более низкой частотой каждую миллисекунду (мс), например, чтобы
    1. скорректировать планирование следующих ходов, чтобы завершить инструкцию G-кода.
  - как компоненты «user-space», которые запускают «основной цикл», так же как и любое другое программное обеспечение, и могут прерываться или задерживаться, когда остальная часть системы занята или перегружена.

В совокупности HAL позволяет

1. программировать для станка, который программист не знает напрямую, но может полагаться на программный интерфейс с четко определенным воздействием на станок. Этот интерфейс может быть использован для
   - сказать станку, что делать
   - послушать, что станок хочет сказать о своем состоянии.
2. Вертикальные абстракции: системный интегратор такого станка использует HAL
   - чтобы описать, как выглядит станок и как какой кабель управляет каким двигателем, который приводит в движение определенную ось.
   - Описание станка, интерфейсы программиста и интерфейс пользователя каким-то образом «встречаются» на этом абстрактном уровне.
3. Горизонтальные абстракции:
   - Не все станки имеют все виды функций
   - фрезеры, токарные станки и роботы имеют много общего
     - особенностей (двигатели, сочленения, …),
     - алгоритмы планирования своих движений.

HAL не имеет прямого взаимодействия с пользователем. Но было предоставлено несколько интерфейсов, позволяющих манипулировать HAL

- из командной строки с помощью команды ”halcmd”.
- из скриптов Python и
- из программ C/C++,

но ни один из этих интерфейсов не является ”самим HAL”.

HAL сам по себе не является программой, он состоит из одного или нескольких списков загруженных программ (компонентов), которые периодически выполняются (в строгой последовательности), и области общей памяти, которую эти компоненты используют для обмена данными. Основной скрипт HAL запускается только один раз при запуске станка, настраивая потоки реального времени и места в общей памяти, загружая компоненты и настраивая каналы передачи данных между ними («сигналы» и «контакты»).

В принципе, несколько станков могут использовать общий HAL, чтобы обеспечить их взаимодействие, однако текущая реализация LinuxCNC ограничена одним интерпретатором и одним модулем задач. В настоящее время это почти всегда интерпретатор G-кода и ”задача фрезерования” (которая также хорошо работает для токарных станков и адекватно для роботов), но эти модули можно выбрать во время загрузки. Учитывая растущий интерес к управлению несколькими взаимодействующими станками, преодоление этого ограничения, вероятно, станет одним из основных шагов для решения проблемы будущего развития LinuxCNC. Однако это немного сложно, и сообщество все еще формирует свои мысли по этому поводу.

HAL лежит в основе LinuxCNC и используется и/или расширяется всеми частями LinuxCNC, включая ГИП. Интерпретатор G-кода (или альтернативного языка) знает, как интерпретировать G-код, и переводит его в операции станка, запуская сигналы в HAL. Пользователь может запрашивать HAL различными способами, чтобы получить информацию о его состоянии, которая затем также представляет состояние станка. Во время разработки версии 2.9, ГИПы по-прежнему являются исключением из этого правила и могут знать что-то, чего HAL не знает (но должен).

### 5.1.2 Коммуникация

HAL уникален тем, что может коммуницировать очень быстро

- с другими программами, но в частности
- с его компонентами, которые обычно выполняются в одном из потоков реального времени.

И при общении та часть LinuxCNC, с которой происходит обмен данными, не нуждается в подготовке к обмену: все эти действия выполняются асинхронно, т. е. ни один компонент не прерывает свое обычное выполнение для получения сигнала, и сигналы могут быть отправлены сразу, т. е. приложение может ждать, пока не поступит конкретное сообщение (например, сигнал разрешения), но ему не нужно готовиться к получению этого сообщения.

Система обмена данными

- представляет и контролирует все оборудование, подключенное к системе,
- запускает и останавливает другие коммуникационные программы.

Связь с аппаратным обеспечением самого станка осуществляется соответствующими выделенными компонентами HAL.

Уровень HAL — это общее пространство, в котором все части, составляющие LinuxCNC, обмениваются информацией. В этом пространстве есть контакты, которые идентифицируются по имени, хотя инженер LinuxCNC может предпочесть ассоциацию с контактом электронной схемы. Эти контакты могут нести числовые и логические значения, булевы значения, числа с плавающей запятой, а также целые числа со знаком и без знака. Существует также (относительно новый) тип вывода с именем hal_port, предназначенный для потоков байтов, и платформа для обмена более сложными данными, называемая hal_stream (которая использует частную общую область памяти, а не контакт HAL). Последние два типа используются сравнительно редко.

С помощью HAL вы можете отправить сигнал на этот названный контакт. Каждая часть HAL может прочитать этот контакт, который содержит значение сигнала. То есть до тех пор, пока

новый сигнал не отправляется для замены предыдущего значения на контакт с тем же именем. Базовая система обмена сообщениями HAL не зависит от ЧПУ, но HAL поставляется с большим количеством компонентов, которые многое знают о ЧПУ и представляют эту информацию через контакты. Есть контакты, обозначающие

- статическую информация о станке
- текущее состояние станка
  - концевиков
  - позиции, подсчитанные шаговыми двигателями или измеренные энкодерами
- получатели инструкций
  - ручное упрваление положением станка (”медленная подача”)
  - положения, которые шаговые двигатели должны занять в следующий раз

По аналогии с электронными кабелями контакты могут быть соединены ”проводами”, поэтому значение, изменяющееся на одном контакте, служит входом для другого контакта. Компоненты HAL подготавливают такие входные и выходные контакты и, таким образом, автоматически запускаются для работы.

**HAL Components** Многие «экспертные» программные части LinuxCNC обычно реализуются как *компоненты* HAL, концептуально также называемые *модулями*. Эти компьютеризированные эксперты постоянно читают из HAL о состоянии, к которому должен стремиться станок, и сравнивают это желаемое состояние с состоянием, в котором станок находится в текущий момент. Когда существует разница между тем, что должно быть, и тем, каким является текущее состояние, выполняются некоторые действия, чтобы уменьшить эту разницу, при этом постоянно записывая обновления текущих состояний обратно в пространство данных HAL.

Есть компоненты, специализирующиеся на взаимодействии с шаговыми двигателями, а другие компоненты умеют управлять сервоприводами. На более высоком уровне некоторые компоненты знают, как расположены оси станка в 3D, а другие знают, как плавно перемещаться из одной точки пространства в другую. Токарные станки, фрезерные станки и роботы будут различаться активным компонентом LinuxCNC, т.е. который загружается файлом конфигурации HAL для этого станка. Тем не менее, два станка могут выглядеть очень по-разному, поскольку они созданы для совершенно разных целей, но когда они оба используют серводвигатели, они все равно могут использовать один и тот же серво-компонент HAL.

**Происхождение стимула к перемещению** На самом нижнем (ближайшем к аппаратному) уровне, например, для шаговых двигателей, описание состояния этого двигателя весьма понятно интуитивно: это количество шагов в определенном направлении. Разница между желаемым положением и фактическим положением преобразуется в движение. Скорость, ускорение и другие параметры могут быть ограничены внутри самого компонента или могут быть ограничены вышестоящими компонентами. (Например, в большинстве случаев значения положения оси по каждому моменту, отправляемые компонентам генератора шаговых импульсов , уже ограничены и сформированы в соответствии с настроенными ограничениями станка или текущей скоростью подачи.)

Любая строка G-кода интерпретируется и запускает набор процедур, которые, в свою очередь, знают, как взаимодействовать с компонентами, расположенными на среднем уровне, например, для создания круга.

**Контакты и Сигналы** HAL занимает особое место в сердцах своих программистов из-за способа представления потока данных между модулями. Когда традиционные программисты думают о переменных, адресах или портах ввода-вывода, HAL относится к ”контактам”. И эти контакты подключены или им присвоены значения через сигналы. Подобно тому, как инженер-электрик соединяет провода между контактами компонентов фрезера, инженер HAL устанавливает поток данных между контактами экземпляров модуля.

ГИПы LinuxCNC (AXIS, GMOCCAPY, Touchy и т. д.) будут отображать состояния некоторых контактов (например, концевых выключателей), но существуют и другие графические инструменты для устранения неполадок и настройки: Halshow, Halmeter, Halscope и Halreport.

Оставшаяся часть этого введения представляет

- синтаксис того, как контакты различных компонентов соединяются в файлах конфигурации HAL, и
- программное обеспечение для проверки значений контактов
  - в любой данный момент,
  - развивающийся с течением времени.

### 5.1.3 HAL System Design

.HAL основан на традиционных методах проектирования систем.

HAL основан на тех же принципах, которые используются для проектирования аппаратных схем и систем, поэтому полезно сначала изучить эти принципы. Любая система, в том числе и станок с ЧПУ состоит из взаимосвязанных компонентов. Для станка с ЧПУ такими компонентами могут быть главный контроллер, сервоусилители или шаговые приводы, двигатели, энкодеры, концевые выключатели, выносные пульты, возможно, ЧРП для привода шпинделя, ПЛК для управления устройством смены инструмента и т. д. Производитель станка должен выбрать, смонтировать и соединить эти части вместе, чтобы создать целостную систему.

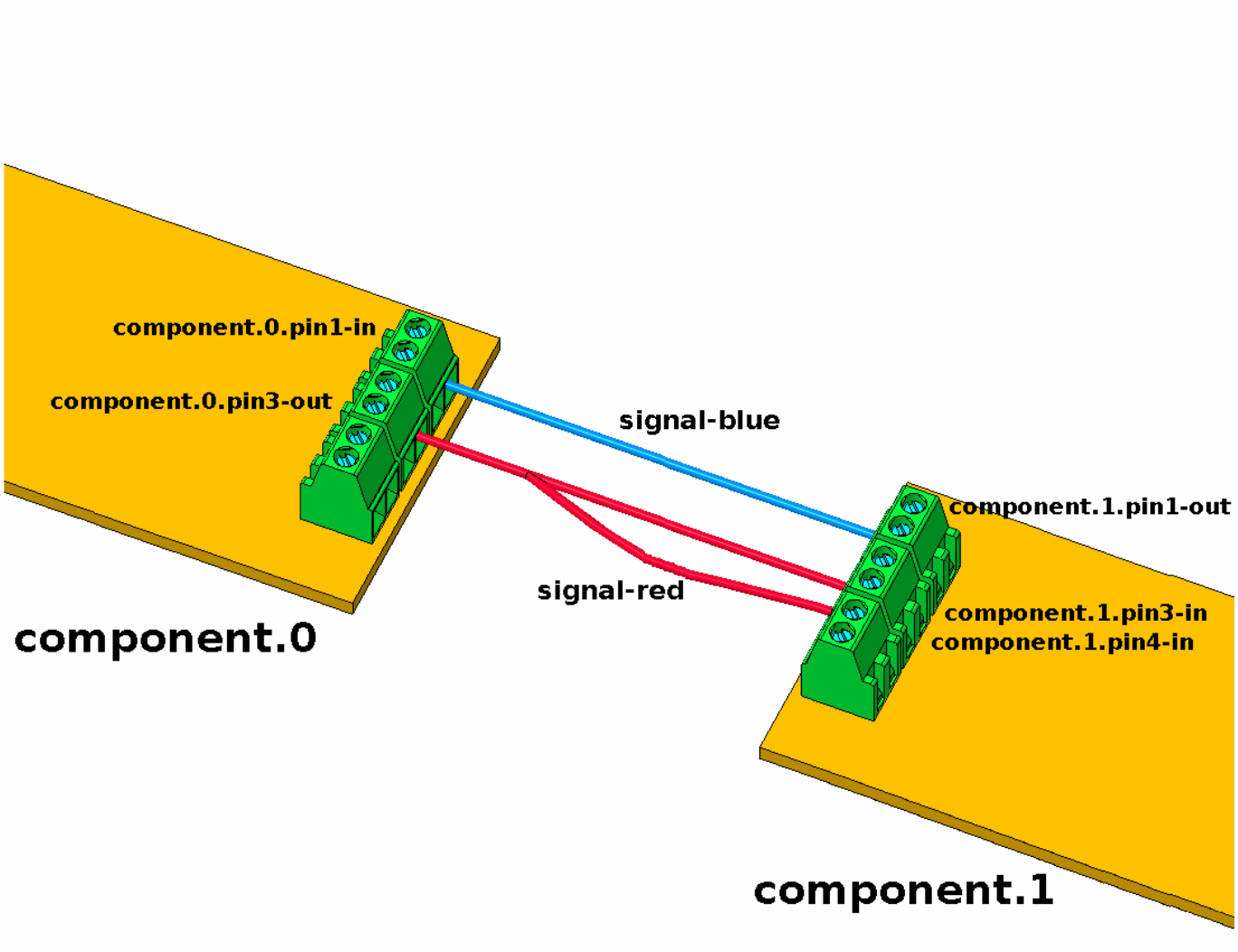



Figure 5.1: Концепция HAL – Соединение как электрические цепи.

Рисунок 1 будет записан в коде HAL следующим образом:

```
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 Выбор детали

Станкостроителю не нужно беспокоиться о том, как работает каждая отдельная деталь. Он обращается с ними как с черными ящиками. На этапе проектирования он решает, какие детали он собирается использовать — шаговые двигатели или сервоприводы, сервоусилитель какой марки, какие концевые выключатели и в каком количестве и т. д. Решение интегратора какие конкретно компоненты использовать, основывается на том, что именно делает компонент и характеристиках, предоставленным производителем устройства. Размер двигателя и нагрузка, которую он должен развивать, будут влиять на выбор усилителя, необходимого для его работы. Выбор усилителя может повлиять на тип обратной связи, необходимой усилителю, а также на сигналы скорости или положения, которые должны быть отправлены на усилитель от элемента управления.

В мире HAL интегратор должен решить, какие компоненты HAL необходимы. Обычно для каждой интерфейсной карты требуется драйвер. Дополнительные компоненты могут потребоваться для программного генерирования шаговых импульсов, функциональности ПЛК и множества других задач.

#### 5.1.3.2 Проектирование межсоединений

Разработчик аппаратной системы не только выбирает детали, но и решает, как эти части будут связаны между собой. Каждый черный ящик имеет клеммы, возможно, всего две для простого переключателя или десятки для сервопривода или ПЛК. Их необходимо соединить вместе. Двигатели подключаются к сервоусилителям, концевые выключатели подключаются к контроллеру и так далее. Работая над проектом, машиностроитель создает большую схему соединений, на которой показано, как все детали должны быть соединены между собой.

При использовании HAL компоненты соединяются между собой сигналами. Проектировщик должен решить, какие сигналы необходимы и что они должны соединять.

#### 5.1.3.3 Реализация

Когда схема подключения готова, пришло время собрать станок. Детали нужно приобрести и смонтировать, а затем соединить их между собой согласно схеме подключения. В физической системе каждое межсоединение представляет собой кусок провода, который необходимо отрезать и подключить к соответствующим клеммам.

HAL предоставляет ряд инструментов, помогающих *построить* систему HAL. Некоторые инструменты позволяют *подключить* (или отключить) один *провод*. Другие инструменты позволяют сохранить полный список всех деталей, проводов и другую информацию о системе, чтобы ее можно было *перестроить* с помощью одной команды.

#### 5.1.3.4 Testing

Очень немногие станки работают правильно с первого раза. Во время тестирования создатель может использовать измеритель, чтобы проверить, работает ли концевой выключатель, или измерить напряжение постоянного тока, поступающее на серводвигатель. Он может подключить осциллограф, чтобы проверить настройку привода или поискать электрические помехи. Он может обнаружить проблему, требующую изменения схемы подключения; возможно, какую-то деталь нужно подключить по-другому или заменить на что-то совсем другое.

HAL предоставляет программные эквиваленты вольтметра, осциллографа, генератора сигналов и других инструментов, необходимых для тестирования и настройки системы. Те же команды, которые использовались для построения системы, можно использовать для внесения изменений по мере необходимости.

#### 5.1.3.5 Краткое содержание

Этот документ предназначен для людей, которые уже знают, как выполнять подобную интеграцию системы оборудования, но не знают, как подключить оборудование к LinuxCNC. См. раздел Пример удаленного запуска в документации HAL UI Examples.

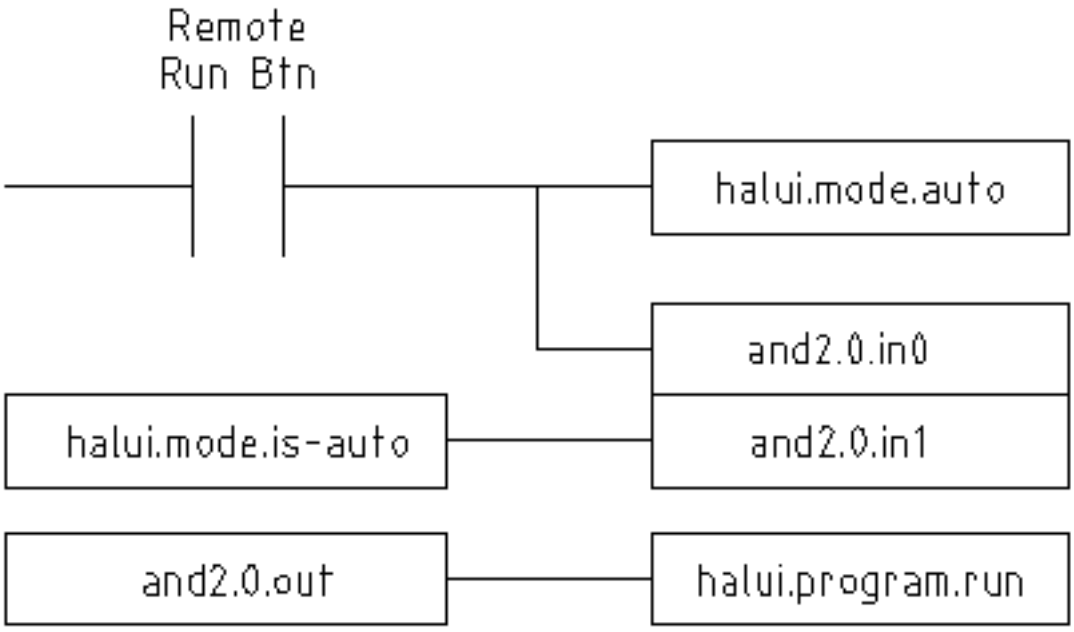



Figure 5.2: Пример удаленного запуска (схема)

Традиционная аппаратная конструкция, описанная выше, заканчивается на краю основного элемента управления. За пределами элемента управления находится куча относительно простых коробок, соединенных вместе, чтобы делать все необходимое. Внутри управление представляет собой большую загадку: один огромный черный ящик, который, как мы надеемся, работает.

HAL распространяет этот традиционный метод проектирования аппаратного обеспечения на внутреннюю часть большого черного ящика. Он превращает драйверы устройств и даже некоторые внутренние части контроллера в меньшие черные ящики, которые можно соединять между собой и даже заменять, как и внешнее оборудование. Это позволяет на *схеме подключения системы* отображать часть внутреннего контроллера, а не просто большой черный ящик. И самое главное, это позволяет интегратору тестировать и модифицировать контроллер, используя те же методы, которые он использовал бы для остального оборудования.

Такие термины, как двигатели, усилители и энкодеры, знакомы большинству разработчиков станков. Когда мы говорим об использовании сверхгибкого восьмижильного экранированного кабеля для подключения энкодера к плате ввода сервопривода в компьютере, читатель сразу понимает, что это такое, и возникает вопрос: *Какие разъемы мне понадобятся, чтобы его оконцевать* Тот же тип мышления необходим и для HAL, но конкретный ход мыслей может занять некоторое время, чтобы войти в нужное русло. Использование слов HAL на первый взгляд может показаться немного странным, но концепция работы от одного соединения к другому одна и та же.

Идея расширения схемы подключения внутрь контроллера и есть суть HAL. Если вас устраивает идея соединения аппаратных черных ящиков, у вас, вероятно, не возникнет проблем с использованием HAL для соединения программных черных ящиков.

### 5.1.4 HAL Концепции

Этот раздел представляет собой глоссарий, в котором определены ключевые термины HAL, но он немного отличается от традиционного глоссария, поскольку эти термины расположены не в алфавитном порядке. Они упорядочены по своим отношениям или последовательности в порядке вещей HAL.

Component (Компонент ):: Когда мы говорили о проектировании аппаратного обеспечения, мы называли отдельные кусочки *частями*, *строительными блоками*, *черными ящиками* и т. д. Эквивалентом HAL является *компонент* или *компонент HAL*. В этом документе используется *компонент HAL*, когда существует вероятность путаницы с другими типами компонентов, но обычно используется просто *компонент*. Компонент HAL — это часть программного обеспечения с четко определенными входами, выходами и поведением, которую можно установить и подключить по мере необходимости. + + Многие компоненты HAL моделируют поведение осязаемой части станка, и **контакт** действительно может предназначаться для подключения к **физическому контакту** на устройстве для связи с ним, отсюда и названия. Но чаще всего это не так. Представьте себе модернизацию ручного токарно-фрезерного станка. LinuxCNC реализует то, как станок представляет себя внешнему миру, и это вторично, если реализация рисования круга уже реализована на машине или предоставлена LinuxCNC. И к воображаемой модернизации часто добавляют кнопки, которые **сигнализируют** о действии, например, об аварийной остановке. LinuxCNC и станок становятся одним целым. И это происходит через HAL.

Parameter (Параметр):: Многие аппаратные компоненты имеют настройки, которые не связаны ни с какими другими компонентами, но к которым все же требуется доступ. Например, сервоусилители часто имеют подстроечные потенциометры, позволяющие выполнять настройку, и контрольные точки, к которым можно подключить измеритель или осциллограф для просмотра результатов настройки. Компоненты HAL также могут иметь такие элементы, которые называются *параметрами*. Существует два типа параметров: Входные параметры эквивалентны регуляторам потенциометра — это значения, которые может регулировать пользователь и они остаются фиксированными после установки. Выходные параметры не могут быть настроены пользователем — они эквивалентны контрольным точкам, позволяющим контролировать внутренние сигналы.

Контакт:: Аппаратные компоненты имеют клеммы, которые используются для их соединения. Эквивалентом HAL является *контакт* или *контакт HAL*. *Контакт HAL* используется, когда необходимо, чтобы избежать путаницы. Все контакты HAL имеют имена, и имена контактов используются при их соединении. Контакты HAL — это программные объекты, существующие только внутри компьютера.

Physical_Pin (Физический контакт):: Многие устройства ввода-вывода имеют реальные физические контакты или клеммы, которые подключаются к внешнему оборудованию, например контакты разъема параллельного порта. Во избежание путаницы их называют *физическими контактами*. Это вещи, которые *смотрят* в реальный мир.

---

**Note**
Вам может быть интересно, какая связь существует между HAL_pins, Physical_pins и внешними элементами, такими как энкодеры или карта STG: здесь мы имеем дело с интерфейсами типа трансляции/преобразования данных.

---

Signal (Сигнал):: В физическом станке выводы реальных аппаратных компонентов соединены между собой проводами. Эквивалентом провода HAL является *сигнал* или *сигнал HAL*. Сигналы HAL соединяют контакты HAL вместе, как того требует производитель оборудования. Сигналы HAL можно отключать и повторно подключать по желанию (даже во время работы станка).

Type:: При использовании реального оборудования вы не будете подключать релейный выход 24 В к аналоговому входу +/-10 В сервоусилителя. Контакты HAL имеют те же ограничения, которые зависят от их типа. И контакты, и сигналы имеют типы, и сигналы могут быть подключены только к контактам одного и того же типа. На данный момент существует 4 типа:

+ - bit - единичное значение TRUE/FALSE или ON/OFF - float — 64-битное значение с плавающей запятой, разрешением примерно 53 бита и динамическим диапазоном более 1000 бит. - u32 — 32-битное целое число без знака, допустимые значения от 0 до 4,294,967,295 - s32 — 32-битное целое число со знаком, допустимые значения: от -2,147,483,648 до +2,147,483,647 - u64 — 64-битное целое число без знака, допустимые значения от 0 до 18,446,744,073,709,551,615 - s64— 64-битное целое число со знаком, допустимые значения: от -9,223,372,036,854,775,808 до +9,223,372,036,854,775,807

Function:: Реальные аппаратные компоненты имеют тенденцию немедленно реагировать на свои входы. Например, если входное напряжение сервоусилителя изменится, выходное напряжение также изменится автоматически. Однако программные компоненты не могут действовать *автоматич* Каждый компонент имеет определенный код, который должен быть выполнен, чтобы сделать все, что этот компонент должен делать. В некоторых случаях этот код просто выполняется как часть компонента. Однако в большинстве случаев, особенно в компонентах реального времени, код должен выполняться в определенной последовательности и через определенные промежутки времени. Например, входные данные следует считывать до выполнения вычислений над входными данными, а выходные данные не следует записывать до тех пор, пока вычисления не будут выполнены. В этих случаях код предоставляется системе в виде одной или нескольких *функций*. Каждая функция представляет собой блок кода, выполняющий определенное действие. Системный интегратор может использовать *потоки* для планирования ряда функций, которые будут выполняться в определенном порядке и через определенные промежутки времени.

Thread (Поток):: *поток* — это список функций, которые выполняются через определенные промежутки времени как часть задачи реального времени. Когда поток создается впервые, он имеет определенный временной интервал (период), но не имеет функций. Функции можно добавлять в поток, и они будут выполняться по порядку при каждом запуске потока.

В качестве примера предположим, что у нас есть компонент parport с именем hal_parport. Этот компонент определяет один или несколько контактов HAL для каждого физического контакта. Контакты описаны в разделе документации этого компонента: их имена, как каждый контакт связан с физическим выводом, инвертированы ли они, можно ли изменить полярность и т. д. Но это само по себе не передает данные с контактов HAL на физические выводы. Для этого требуется код, и именно здесь на сцену выходят функции. Компоненту parport необходимы как минимум две функции: одна для чтения физических входных контактов и обновления контактов HAL, другая для получения данных с контактов HAL и записи их на физические выходные контакты. Обе эти функции являются частью драйвера parport.

### 5.1.5 HAL компоненты

Каждый компонент HAL представляет собой часть программного обеспечения с четко определенными входами, выходами и поведением, которую можно установить и подключить по мере необходимости. В разделе HAL Components List перечислены все доступные компоненты и краткое описание того, что каждый из них делает.

### 5.1.6 Проблемы с синхронизацией в HAL

В отличие от моделей физического соединения между черными ящиками, на которых, как мы говорили, основан HAL, простое соединение двух контактов с сигналом HAL далеко не соответствует действию физического случая.

Настоящая релейная логика состоит из реле, соединенных вместе, и когда контакт размыкается или замыкается, ток течет (или прекращается) немедленно. Другие катушки могут изменить состояние и т. д., и все это просто *происходит*. Но в языке релейных схем ПЛК это не работает. Обычно за один проход по цепям каждая цепь оценивается в том порядке, в котором она появляется, и только один раз за проход. Прекрасным примером является цепь с одним НЗ контактом, включенным последовательно с катушкой. Контакт и катушка принадлежат одному реле.

Если бы это было обычное реле, то как только на катушку подается напряжение, контакты начинают размыкаться и обесточивать ее. Это означает, что контакты снова замыкаются и т. д. и т. п. Реле начинает издавать звуковой сигнал.

В ПЛК, если катушка выключена и контакт замкнут, когда ПЛК начинает оценивать цепь, то по завершении этого прохода катушка включается. Тот факт, что включение катушки размыкает контакт, питающий ее, игнорируется до следующего прохода. На следующем проходе ПЛК видит, что контакт разомкнут, и обесточивает катушку. Таким образом, реле по-прежнему

быстро переключается между включением и выключением, но со скоростью, определяемой тем, как часто ПЛК оценивает цепь.

В HAL функция — это код, который оценивает цепь(и). Фактически, версия ClassicLadder реального времени с поддержкой HAL экспортирует функцию, которая делает именно это. Между тем, поток — это то, что запускает функцию через определенные промежутки времени. Точно так же, как вы можете выбрать, чтобы ПЛК оценивал все свои цепочки каждые 10 мс или каждую секунду, вы можете определять потоки HAL с разными периодами.

Что отличает один поток от другого, так это *не* то, что поток делает — это определяется тем, какие функции к нему подключены. Настоящее различие заключается лишь в том, как часто запускается поток.

В LinuxCNC у вас может быть поток 50 мкс и поток 1 мс. Они будут созданы на основе BASE_PERIOD и SERVO_PERIOD, фактическое время зависит от значений в вашем INI-файле.

Следующий шаг — решить, что должен делать каждый поток. Некоторые из этих решений одинаковы (почти) в любой системе LinuxCNC. Например, обработчик команды движения всегда добавляется в сервопоток.

Другие соединения будут выполнены интегратором. Они могут включать в себя подключение функций чтения и записи ЦАП драйвера STG к сервопотоку или подключение функции StepGen к базовому потоку вместе с функциями parport для записи импульсов шага в порт.

## 5.2 HAL Основы

Этот документ дает основы HAL.

### 5.2.1 HAL команды

Более подробную информацию можно найти на странице руководства по halcmd: запустите *man halcmd* в окне терминала.

Чтобы просмотреть конфигурацию HAL и проверить состояние контактов и параметров, используйте окно «Конфигурация HAL» в меню «Станок» в AXIS. Чтобы просмотреть статус контакта, откройте вкладку Watch и щелкните каждый контакт, который вы хотите просмотреть, и он будет добавлен в окно просмотра.

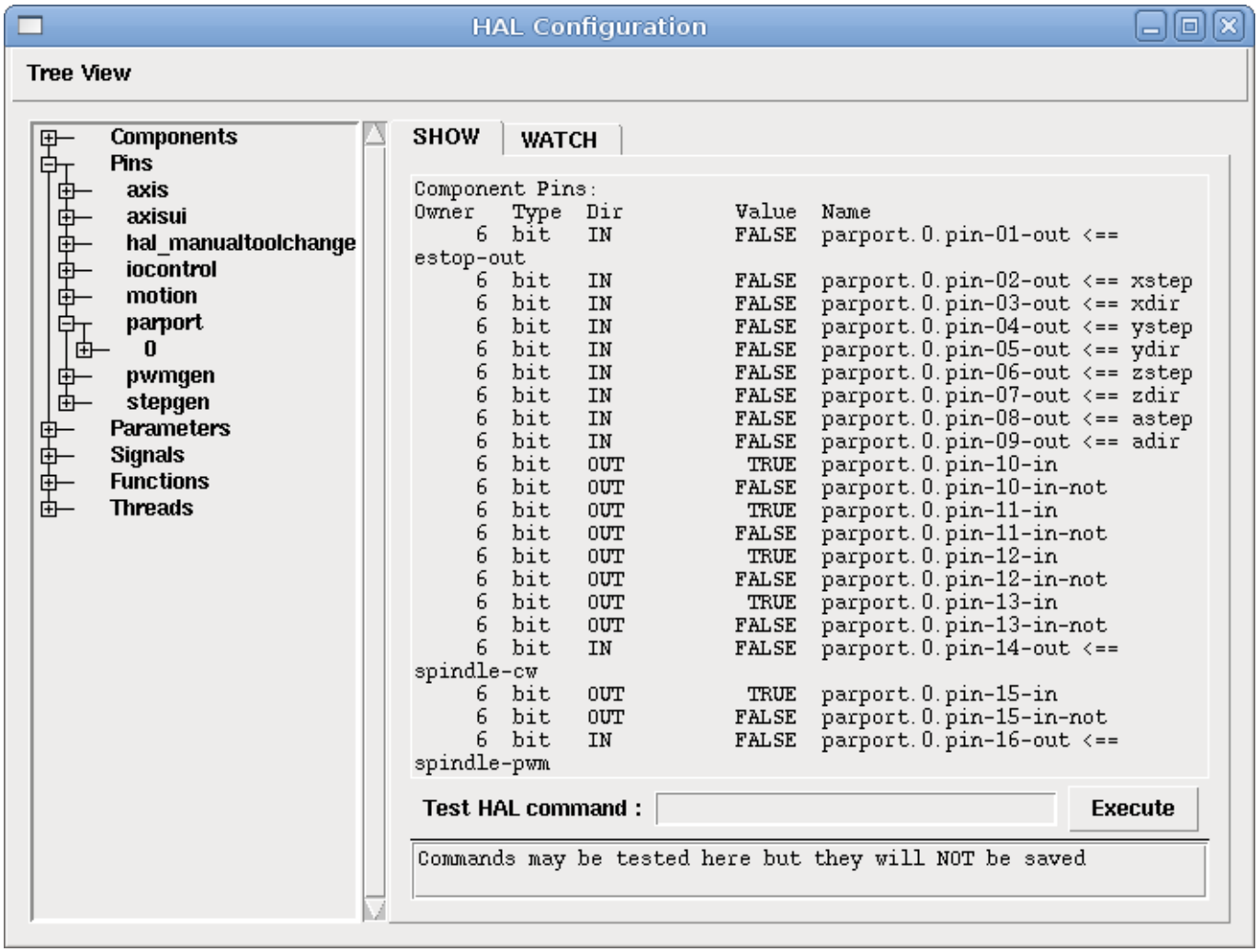



Figure 5.3: Окно конфигурации HAL

#### 5.2.1.1 loadrt

Команда `loadrt` загружает компонент HAL в реальном времени. Функции компонентов реального времени необходимо добавлять в поток, чтобы они обновлялись со скоростью потока. Вы не можете загрузить компонент, не работающий в реальном времени, в пространство реального времени.

**loadrt Синтаксис и Пример**

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

#### 5.2.1.2 addf

Команда `addf` добавляет функцию в поток реального времени. Если для создания конфигурации использовался мастер StepConf, были созданы два потока (`base-thread` и `servo-thread`).

`addf` добавляет функцию *functname* в поток *threadname*. По умолчанию функции добавляются в том порядке, в котором они находятся в файле. Если *position* указано, функция добавляется в

это место в потоке. Отрицательное значение *position* указывает положение относительно конца потока. Например, *1* — начало потока, *-1* — конец потока, *-3* — третий от конца.

Для некоторых функций важно загружать их в определенном порядке, например, функции чтения и записи parport. Имя функции обычно представляет собой имя компонента плюс номер. В следующем примере загружается компонент *or2*, а `show function` показывает имя функции or2.

```
$ 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.

---

**Note**
The FP column and the `uses_fp` parameter are deprecated. All threads now unconditionally save and restore floating point state. The `fp`/`nofp` distinction will be removed in a future version.

---

```
$ halrun
halcmd: loadrt motmod base_period_nsec=55555 servo_period_nsec=1000000 num_joints=3
halcmd: show thread
Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     995976  YES          servo-thread (        0,        0 )
      55332  YES           base-thread (        0,        0 )
```

- base-thread (the high-speed thread): This thread handles items that need a fast response, like making step pulses, and reading and writing the parallel port.
- servo-thread (the slow-speed thread): This thread handles items that can tolerate a slower response, like the motion controller, ClassicLadder, and the motion command handler.

#### addf Синтаксис и Пример

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

### 5.2.1.3 loadusr

Команда loadusr загружает компонент HAL не в реальном времени. Программы, не работающие в режиме реального времени, представляют собой отдельные процессы, которые при необходимости взаимодействуют с другими компонентами HAL через контакты и параметры. Вы не можете загружать компоненты реального времени в пространство не в реальном времени.

Флаги могут быть одним или несколькими из следующих:

| | |
|---|---|
| -W | дождаться готовности компонента. Предполагается, что компонент имеет то же имя, что и первый аргумент команды. |
| -Wn <name> | дождаться компонента, который будет иметь заданное <name>. Это применимо только в том случае, если у компонента есть опция имени. |
| -w | дождаться выхода программы |
| -i | игнорировать возвращаемое значение программы (с -w) |

-n назвать компонент, если это допустимая опция для этого компонента.

**Синтаксис и Примеры loadusr**

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

По-английски это означает *loadusr ждет имя шпинделя компонента gs2_vfd имя шпинделя*.

#### 5.2.1.4 net

Команда `net` создает `соединение` между сигналом и одним или несколькими контактами. Если сигнал не существует, net создает новый сигнал. Это заменяет необходимость использования команды newsig. Необязательные стрелки направления `<=`, `=>` и `<=>` облегчают отслеживание логики при чтении командной строки `net` и не используются командой net. Стрелки направления должны быть отделены пробелом от названий контактов.

**Синтаксис и Примеры 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
```

В приведенном выше примере `home-x` — это имя сигнала, `joint.0.home-sw-in` — это контакт `Direction IN`, `<=` — необязательная стрелка направления, а `parport.0.pin.-11-in` — это контакт «Direction OUT». Это может показаться запутанным, но метки входа и выхода для контакта параллельного порта указывают на физический способ работы контакта, а не на то, как он обрабатывается в HAL.

Контакт может быть подключен к сигналу, если он подчиняется следующим правилам:

- Контакт IN всегда можно подключить к сигналу.
- Конакт IO можно подключить, если на сигнале нет контакта OUT.
- Контакт OUT можно подключить только в том случае, если в сигнале нет других контактов OUT или IO.

Одно и то же *signal-name* можно использовать в нескольких сетевых командах для подключения дополнительных контактов, если соблюдаются приведенные выше правила.

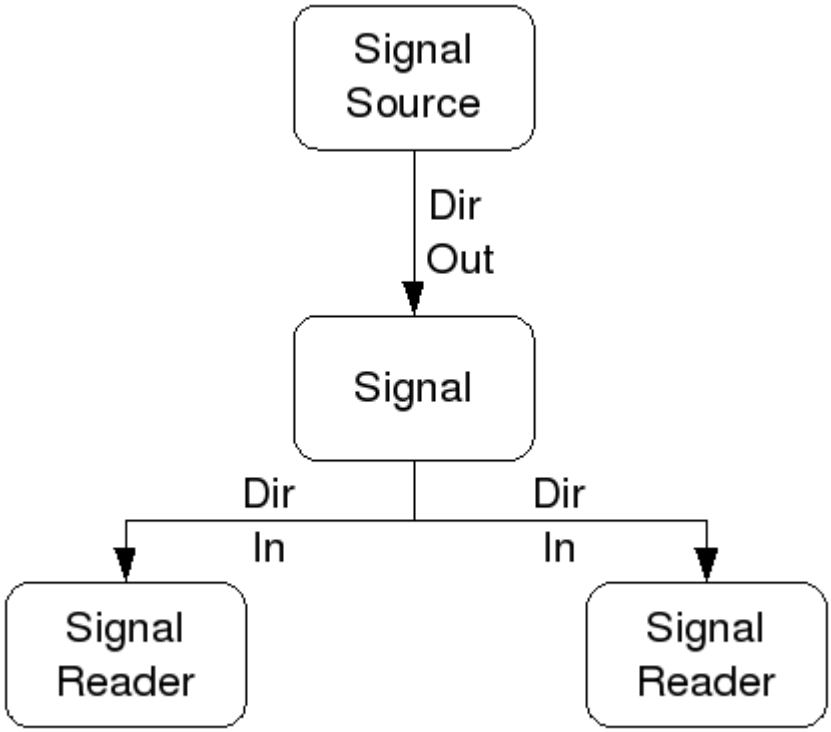



Figure 5.4: Направление сигнала

В этом примере показан сигнал xStep с источником `stepgen.0.out` и двумя считывателями: `parport.0.pin-02-out` и `parport.0.pin-08-out`. По сути, значение `stepgen.0.out` отправляется в сигнал xStep, а затем это значение отправляется в `parport.0.pin-02-out` и `parport.0.pin-08-out`.

```
#   b''cb''b''ив''b''гb''b''нb''b''ab''b''лb''    b''ив''b''cb''b''тb''b''ob''b''чb''b' ↩
    'нb''b''ив''b''кb''           b''пb''b''ob''b''лb''b''yb''b''чb''b''ab''b''тb''b''eb''b ↩
    ''лb''b''ьb''          b''пb''b''ob''b''лb''b''yb''b''чb''b''ab''b''тb''b''eb''b''лb''b' ↩
    'ьb''
net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out
```

Поскольку сигнал xStep содержит значение `stepgen.0.out` (источник), вы можете использовать тот же сигнал снова, чтобы отправить значение другому получателю. Для этого достаточно использовать сигнал с получателями на другой линии.

```
#   b''cb''b''ив''b''гb''b''нb''b''ab''b''лb''       b''пb''b''ob''b''лb''b''yb''b''чb''b' ↩
    'ab''b''тb''b''eb''b''лb''b''ьb''2
net xStep => parport.0.pin-06-out
```

**I/O контакты** Контакт ввода-вывода, такой как encoder.*N*.index-enable, может быть считан или установлен в соответствии с разрешением компонента.

#### 5.2.1.5 setp

Команда `setp` устанавливает значение контакта или параметра. Допустимые значения будут зависеть от типа контакта или параметра. Если типы данных не совпадают, это ошибка.

Некоторые компоненты имеют параметры, которые необходимо настроить перед использованием. При необходимости параметры можно установить перед использованием или во время работы. Вы не можете использовать setp на контакте, подключенном к сигналу.

**Синтаксис и примеры setp**

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

#### 5.2.1.6 sets

Команда `sets` устанавливает значение сигнала.

**Синтаксис и примеры sets**

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

It is an error if:

- Имя сигнала не существует
- Если у сигнала уже есть источник
- Если значение не является правильным типом для сигнала

#### 5.2.1.7 unlinkp

Команда `unlinkp` отключает контакт от подключенного сигнала. Если до выполнения команды к контакту не был подключен сигнал, ничего не произойдет. Команда `unlinkp` полезна при устранении неполадок.

**Синтаксис и примеры unlinkp**

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

#### 5.2.1.8 Устаревшие команды

Следующие команды устарели и могут быть удалены из будущих версий. Любая новая конфигурация должна использовать команду net. Эти команды включены, поэтому старые конфигурации по-прежнему будут работать.

Команда `linksp` создает `соединение` между сигналом и одним контактом.

**Синтаксис и примеры linksp**

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

**Команда `linksp` заменена командой `net`.**

Команда *linkps* создает *соединение* между одним контактом и одним сигналом. Это то же самое, что и `linksp`, но аргументы обратные.

**Синтаксис и примеры linkps**

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

**Команда `linkps` заменена командой `net`.**

команда `newsig` создает новый сигнал HAL с именем *<signname>* и типом данных *<type>*. Тип должен быть *bum*, *s32*, *u32*, *s64*, *u64* или *float*. Ошибка, если *<signame>* уже существует.

**Синтаксис и примеры newsig**

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

Дополнительную информацию можно найти в руководстве HAL или на страницах руководства `halrun`.

## 5.2.2 HAL Data

### 5.2.2.1 Bit

Значение бита ВКЛ или ВЫКЛ.

- значение bit = true или 1 и false илиг 0 (True, TRUE, true все действительны)

### 5.2.2.2 Float

«Float» — это число с плавающей запятой. Другими словами, десятичная точка может перемещаться по мере необходимости.

- Значения float = 64-битное значение с плавающей запятой, с разрешением примерно 53 бита и более $2^{10}$ (~ 1000) бит динамического диапазона.

Дополнительную информацию о числах с плавающей запятой см.:

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

### 5.2.2.3 s32

Число «s32» — это целое число, которое может иметь отрицательное или положительное значение.

- Значения s32 = целые числа от -2147483648 до 2147483647

### 5.2.2.4 u32

Число «u32» — это целое число, которое может быть только положительным.

- Значения u32 = целые числа от 0 до 4294967295

### 5.2.2.5 s64

Число *s64* — это целое число, которое может иметь отрицательное или положительное значение.

- Значения s64 = целые числа от -9,223,372,036,854,775,808 до +9,223,372,036,854,775,807

#### 5.2.2.6 u64

Число *u64* — это целое число, которое может быть только положительным.

- Значения u64 = целые числа от 0 до 18,446,744,073,709,551,615

### 5.2.3 HAL файлы

Если вы использовали Stepper Config Wizard для создания конфигурации, в вашем каталоге конфигурации будет до трех файлов HAL.

- «my-mill.hal» (если ваша конфигурация называется «my-mill»). Этот файл загружается первым, и его не следует изменять, если вы использовали Stepper Config Wizard.
- *custom.hal* Этот файл загружается следующим, до загрузки ГИП. Здесь вы помещаете свои собственные команды HAL, которые хотите загрузить до загрузки ГИП.
- *custom_postgui.hal* Этот файл загружается после загрузки ГИП. Здесь вы помещаете свои собственны команды HAL, которые хотите загрузить после загрузки ГИП. В нем необходимо разместить любые команды HAL, использующие виджеты PyVCP.

### 5.2.4 HAL Параметр

Два параметра автоматически добавляются к каждому компоненту HAL при его создании. Эти параметры позволяют вам ограничить время выполнения компонента.

| | |
|---|---|
| `.time` | Время — это количество циклов ЦП, необходимое для выполнения функции. |
| `.tmax` | Tmax — максимальное количество циклов ЦП, необходимое для выполнения функции. |

`tmax` — это параметр чтения/записи, поэтому пользователь может установить его равным 0, чтобы избавиться от первой инициализации во время выполнения функции.

### 5.2.5 Основные логические компоненты

HAL содержит несколько компонентов логики реального времени. Логические компоненты следуют *Truth Table*, в которой указывается, какой выходной сигнал соответствует любому заданному входному сигналу. Обычно это битовые манипуляторы, которые следуют таблицам истинности электрических логических элементов.

Дополнительные компоненты см. в <<sec:hal-comComponents,HAL Components List> или на страницах руководства.

#### 5.2.5.1 and2

Компонент «and2» представляет собой логический элемент ”И” с двумя входами. В приведенной ниже таблице истинности показаны выходные данные, основанные на каждой комбинации входных данных.

**Syntax**

```
and2 [count=N] | [names=name1[,name2...]]
```

**Функции**

```
and2.n
```

**Контакты**

```
and2.N.in0 (bit, in)
and2.N.in1 (bit, in)
and2.N.out (bit, out)
```

Table 5.3: Таблица истинности для and2

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

#### 5.2.5.2 not

Компонент not является битовым инвертером.

**Syntax**

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

**Функции**

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

**Контакты**

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

Table 5.4: Таблица истинности для not

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

#### 5.2.5.3 or2

Компонент or2 является логическим элементом ”ИЛИ” с двумя входами.

**Syntax**

```
or2[count=n] | [names=name1[,name2...]]
```

**Функции**

```
or2.n
```

**Контакты**

```
or2.n.in0 (bit, in)
or2.n.in1 (bit, in)
or2.n.out (bit, out)
```

Table 5.5: or2 Таблица истинности

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

#### 5.2.5.4 xor2

Компонент `xor2` является логическим элементом ”Исключающее ИЛИ” с двумя входами.

**Syntax**

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

**Функции**

```
xor2.n
```

**Контакты**

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

Table 5.6: xor2 Таблица истинности

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

### 5.2.6 Примеры логики

.Пример использования 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
```

В приведенном выше примере одна копия and2 загружается в пространство реального времени и добавляется в сервопоток. Следующий контакт pin-11 параллельного порта подключен к биту in0 вентиля”И”. Следующий контакт pin-12 подключен к биту in1 вентиля ”И”. Наконец, мы подключаем выходной бит and2 к параллельному порту pin-14. Итак, следуя таблице истинности для and2, если контакт 11 и контакт 12 включены, то выходной контакт 14 будет включен.

### 5.2.7 Компоненты преобразования

#### 5.2.7.1 weighted_sum

Взвешенная сумма преобразует группу битов в целое число. Преобразование представляет собой сумму *весов* присутствующих битов плюс любое смещение. Это похоже на *BCD*, но с бОльшим количеством опций. Бит *hold* прерывает обработку ввода, так что значение *sum* больше не изменяется.

**Синтаксис загрузки компонента weighted_sum**

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

Создает группы ``weighted_sum``, каждая с заданным количеством входных битов (размером).

Чтобы обновить weighted_sum, process_wsums должен быть прикреплен к потоку.

**Добавьте process_wsums в поток сервопривода**

```
addf process_wsums servo-thread
```

При этом обновляется компонент weighted_sum.

В следующем примере, копия окна конфигурации AXIS HAL, биты *0* и *2* имеют значение TRUE, они не имеют смещения. Вес (*weight*) бита 0 равен 1, бита 2 — 4, поэтому сумма равна 5.

Table 5.7: Контакты компонента weighted_sum

| Владелец | Type | Dir | Value | Имя |
|---|---|---|---|---|
| 10 | бит | In | TRUE | wsum.0.bit.0.in |
| 10 | s32 | I/O | 1 | wsum.0.bit.0.weight |
| 10 | бит | In | FALSE | wsum.0.bit.1.in |
| 10 | s32 | I/O | 2 | wsum.0.bit.1.weight |
| 10 | бит | In | TRUE | wsum.0.bit.2.in |
| 10 | s32 | I/O | 4 | wsum.0.bit.2.weight |
| 10 | бит | In | FALSE | wsum.0.bit.3.in |
| 10 | s32 | I/O | 8 | wsum.0.bit.3.weight |
| 10 | бит | In | FALSE | wsum.0.hold |
| 10 | s32 | I/O | 0 | wsum.0.offset |
| 10 | s32 | Out | 5 | wsum.0.sum |

# 5.3 HAL Двойной проход

## 5.3.1 Двойной проход

В этом разделе описывается возможность использования нескольких команд загрузки для нескольких экземпляров одного и того же компонента в разных позициях файла или среди разных файлов. Внутренне для этого требуется дважды прочитать файл HAL, отсюда и название TWOPASS. Поддержив начиная с версии LinuxCNC 2.5 обработка TWOPASS файлов конфигурации LinuxCNC помогает обеспечить их модульность и удобочитаемость. Напомним, что файлы конфигурации LinuxCNC указываются в INI-файле LinuxCNC как `[HAL]HALFILE=filename`.

Обычно набор из одного или нескольких файлов конфигурации LinuxCNC должен использовать одну уникальную строку `loadrt` для загрузки компонента реального времени, что может создать несколько экземпляров компонента. Например, если вы используете компонент вентиля И с двумя входами (and2) в трех разных местах вашей установки, вам нужно будет где-то указать одну строку:

**Пример, в результате которого создаются компоненты реального времени с именами по умолчанию and2.0, and2.1, and2.2.**

```
loadrt and2 count=3
```

Конфигурации станут более читабельными, если вы укажете с помощью опции `names=` те компоненты, где она поддерживается, например:

**Пример команды загрузки, в результате которой были явно названы компоненты aa, ab, ac.**

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

Отслеживание компонентов и их имен может стать проблемой при обслуживании, поскольку при добавлении (или удалении) компонента необходимо найти и обновить единственную директиву loadrt, применимую к этому компоненту.

**Обработка TWOPASS включается путем включения параметра INI-файла в раздел [HAL], где ”anystring” может быть любой строкой, отличной от нуля.**

```
[HAL]

TWOPASS = anystring
```

При включенном TWOPASS вы можете использовать несколько спецификаций, например:

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

Эти команды могут появляться в разных файлах HAL. Файлы HAL обрабатываются в порядке их появления в файле INI, в нескольких назначениях HALFILE.

Опцию TWOPASS можно указать с опциями для добавления вывода для отладки (verbose) и для предотвращения удаления временных файлов (nodelete). Опции разделяются запятыми.

**Пример**

```
[HAL]
TWOPASS = on,verbose,nodelete
```

При обработке TWOPASS сначала считываются все [HAL]HALFILES и накапливаются многочисленные появления директив loadrt для каждого модуля. Компоненты, не работающие в реальном времени (loadusr), загружаются по порядку, но на начальном этапе никакие другие команды LinuxCNC не выполняются.

---

**Note**
Компоненты, не работающие в режиме реального времени, должны использовать параметр ожидания (-W), чтобы гарантировать готовность компонента до выполнения других команд.

---

После первого прохода модули реального времени загружаются (loadrt) автоматически

- с числом, равным общему числу, при использовании опции *count*= или
- со всеми отдельными именами, указанными при использовании опции *names*=.

Затем выполняется второй проход для выполнения всех остальных инструкций LinuxCNC, указанных в HALFILES. Команды addf, которые связывают функции компонента с выполнением потока, выполняются в порядке появления других команд во время этого второго прохода.

Хотя вы можете использовать опции *count*= или *names*=, они являются взаимоисключающими — для данного модуля можно указать только один тип.

Обработка TWOPASS наиболее эффективна при использовании опции *names*=. Эта опция позволяет вам предоставлять уникальные имена, которые являются мнемоническими или иным образом связаны с конфигурацией. Например, если вы используете производный компонент для оценки скоростей и ускорений по каждой координате (x,y,z), использование метода *count*= даст загадочные имена компонентов, такие как ddt.0, ddt.1, ddt.2. , и т. д.

В качестве альтернативы можно использовать опцию *names*=, например:

```
loadrt ddt names=xvel,yvel,zvel
...
loadrt ddt names=xaccel,yaccel,zaccel
```

в результате получаются компоненты с разумными названиями xvel, yvel, zvel, xaccel, yaccel, zaccel.

Многие компоненты, поставляемые с дистрибутивом, созданы с помощью утилиты halcompile и поддерживают опцию *names*=. К ним относятся общие логические компоненты, которые являются связующим звеном многих конфигураций LinuxCNC.

Созданные пользователем компоненты, которые используют утилиту halcompile, также автоматически поддерживают опцию names=. Помимо композиций, созданных с помощью утилиты halcompile, многие другие композиции поддерживают опцию name=. Компиляции, поддерживающие опцию name=, включают: at_pid, encoder, encoder_ratio, pid, siggen и sim_encoder.

Двухэтапная обработка происходит до загрузки графического интерфейса. При использовании [HAL]POSTGUI_HALFILE удобно поместить все объявления loadrt [HAL]POSTGUI_HALFILE для необходимых компонентов в предварительно загруженный файл HAL.

**Пример раздела HAL при использовании 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 lines for components in postgui.hal

POSTGUI_HALFILE = postgui.hal
HALUI = halui
```

### 5.3.2 Post GUI

Некоторые ГИПы поддерживают файлы HAL, которые обрабатываются после запуска графического интерфейса для подключения контактов LinuxCNC, созданных ГИП. При использовании файла HAL postgui с обработкой TWOPASS включите все элементы loadrt для компонентов, добавленных файлами HAL postgui, в отдельный файл HAL, который обрабатывается перед графическим интерфейсо Команды `addf` также могут быть включены в файл.

**Пример**

```
[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 Исключение файлов .hal

Обработка TWOPASS преобразует файлы *.hal* в эквивалентные файлы *.tcl* и использует *haltcl* для поиска команд loadrt и addf с целью накопления и консолидации их использования. Ожидаются параметры loadrt, соответствующие простым параметрам `names=` (или `count=`), принимаемым генератором компонентов HAL (`halcompile`). Более сложные элементы параметров, включенные в специализированные компоненты LinuxCNC, могут обрабатываться неправильно.

Файл *.hal* можно исключить из обработки TWOPASS, включив строку магического комментария в любом месте файла *.hal*. Строка магического комментария должна начинаться со строки: `#NOTWOPASS`. Файлы, указанные с этим магическим комментарием, используется в качестве источника halcmd с использованием опций `-k` (продолжать работу в случае сбоя) и `-v` (подробный вариант).

Это положение об исключении можно использовать для изоляции проблем или для загрузки любого специального компонента LinuxCNC, который не требует обработки TWOPASS или не получает преимуществ от нее.

Обычно порядок loadrt компонентов реального времени не является критическим, но порядок loadrt для специальных компонентов можно обеспечить, поместив такие директивы loadrt в файл исключений.

---

**Note**
Хотя порядок директив loadrt обычно не имеет решающего значения, порядок директив addf часто очень важен для правильной работы компонентов сервоконтура.

---

**Пример файла исключений HAL**

```
$ cat twopass_excluded.hal
# The following magic comment causes this file to
# be excluded from twopass processing:
# NOTWOPASS

# debugging component with complex options:
loadrt mycomponent parm1=”abc def” parm2=ghi
show pin mycomponent

# ordering special components
loadrt component_1
loadrt component_2
```

**Note**
Регистр и пробелы в магическом комментарии игнорируются. Загрузка компонентов, использующих параметры names= или count= (обычно создаваемые с помощью halcompile), не должна использоваться в файлах исключений, так как это лишит преимуществ обработки TWOPASS. Команды LinuxCNC, создающие сигналы (net), и команды, устанавливающие порядок выполнения (addf), не следует помещать в файлы исключений. Это особенно актуально для команд addf, поскольку их порядок может иметь важное значение.

### 5.3.4 Examples

Примеры использования TWOPASS для симулятора включены в каталоги:

```
configs/sim/axis/twopass/
configs/sim/axis/simtcl/
```

## 5.4 HAL Учебное пособие

### 5.4.1 Введение

Конфигурация переходит от теории к устройству — то есть к устройству HAL. Для тех, кто хоть немного разбирался в компьютерном программировании, этот раздел называется *Hello World* HAL.

halrun можно использовать для создания работающей системы. Это инструмент командной строки или текстового файла для настройки и настройки. Следующие примеры иллюстрируют его настройку и работу.

### 5.4.2 Halcmd

«halcmd» — это инструмент командной строки для управления HAL. Для halcmd существует более полная справочная страница, которая устанавливается вместе с LinuxCNC из исходного кода или из пакета. Если LinuxCNC был скомпилирован как *run-in-place*, страница руководства не устанавливается, но доступна в главном каталоге LinuxCNC с помощью следующей команды:

```
$ man -M docs/man halcmd
```

#### 5.4.2.1 Обозначения

В этом руководстве команды для операционной системы обычно отображаются без приглашения оболочки UNIX, т. е. обычно со знаком доллара ($) или решеткой/двойным крестом (#). При прямом общении с HAL через halcmd or halrun подсказки показаны в примерах. Окно терминала находится в разделе *Applications/Accessories* в главной строке меню Ubuntu.

**Пример команды в терминале - приглашения командной строки**

```
me@computer:~linuxcnc$ halrun
(will be shown like the following line)
halrun

(the halcmd: prompt will be shown when running HAL)
halcmd: loadrt counter
halcmd: show pin
```

#### 5.4.2.2 Табуляция-завершение

Ваша версия `halcmd` может включать в себя завершение табуляции. Вместо завершения имен файлов, как это делает оболочка, она дополняет команды идентификаторами HAL. Вам нужно будет ввести достаточно букв, чтобы получить уникальное совпадение. Попробуйте нажать Tab после запуска команды HAL:

**Табуляция-завершение**

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

#### 5.4.2.3 Среда RTAPI

RTAPI означает Real Time Application Programming Interface. Многие компоненты HAL работают в реальном времени, и все компоненты HAL хранят данные в общей памяти, поэтому компоненты реального времени могут получить к ним доступ. Обычный Linux не поддерживает программирование в реальном времени или тип общей памяти, необходимый HAL. К счастью, существуют операционные системы реального времени (RTOS), которые предоставляют необходимые расширения Linux. К сожалению, каждая RTOS действует по-своему.

Чтобы устранить эти различия, команда LinuxCNC разработала RTAPI, который обеспечивает согласованный способ взаимодействия программ с RTOS. Если вы программист, который хочет работать над внутренними компонентами LinuxCNC, возможно, вам захочется изучить *linuxcnc/src/rtapi/rtapi.h*, чтобы понять API. Но если вы обычный человек, все, что вам нужно знать о RTAPI, это то, что его (и RTOS) необходимо загрузить в память вашего компьютера, прежде чем вы что-либо сделаете с HAL.

### 5.4.3 Простой пример

#### 5.4.3.1 Загрузка компонента

В этом руководстве мы будем предполагать, что вы успешно установили Live CD и, если используете RIP footnote: [Run In Place, когда исходные файлы загружены в пользовательский каталог, компилируются и выполняются непосредственно оттуда.], вызовите сценарий *rip-environment*, чтобы подготовить shell. В этом случае все, что вам нужно сделать, это загрузить в память необходимые модули RTOS и RTAPI. Просто запустите следующую команду из окна терминала:

**Загрузка HAL**

```
cd linuxcnc
halrun
halcmd:
```

Загрузив ОС реального времени и RTAPI, мы можем перейти к первому примеру. Обратите внимание, что приглашение теперь отображается как *halcmd:*. Это связано с тем, что последующие команды будут интерпретироваться как команды HAL, а не команды оболочки.

В первом примере мы будем использовать компонент HAL под названием *siggen*, который представляет собой простой генератор сигналов. Полное описание компонента *siggen* можно найти в разделе SigGen данного руководства. Это компонент реального времени. Чтобы загрузить компонент "`siggen`", используйте команду HAL *loadrt*.

**Загрузка siggen**

```
halcmd: loadrt siggen
```

#### 5.4.3.2 Изучение HAL

Теперь, когда модуль загружен, пришло время представить `halcmd` — инструмент командной строки, используемый для настройки HAL. В этом руководстве будут представлены только некоторые функции halcmd. Для более полного описания попробуйте `man halcmd` или посмотрите ссылку в разделе HAL Commands этого документа. Первая функция halcmd — это команда `show`. Эта команда отображает информацию о текущем состоянии HAL. Чтобы отобразить все установленные компоненты:

**Показать компоненты с помощью halrun/halcmd**

```
halcmd: show comp

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

Поскольку *halcmd* сам по себе также является компонентом HAL, он всегда будет отображаться в списке. Число после "`halcmd`" в списке компонентов — это ID UNIX процесса. Можно запустить более одной копии halcmd одновременно (например, в разных окнах терминала), поэтому PID добавляется в конец имени, чтобы сделать его уникальным. В списке также показан компонент *siggen*, который мы установили на предыдущем шаге. *RT* в разделе *Type* указывает, что *siggen* — это компонент реального времени. *User* в разделе *Type* указывает, что это компонент, не работающий в реальном времени.

Далее, давайте посмотрим, какие контакты делает доступными `siggen`:

**Показать контакты**

```
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
```

Эта команда отображает все контакты текущего HAL. Сложная система может иметь десятки или сотни контактов. Но сейчас контактов всего девять. Из этих контактов восемь являются плавающей запятой, а один — битовым (логическим). Шесть переносят данные из компонента *siggen*, а три используются для передачи настроек в компонент. Поскольку мы еще не выполнили код, содержащийся в компоненте, некоторые контакты имеют нулевое значение.

Следующий шаг — просмотр параметров:

**Показать параметры**

```
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
```

Команда *show param* показывает все параметры в HAL. Сейчас каждый параметр имеет значение по умолчанию, которое ему было присвоено при загрузке компонента. Обратите внимание на

столбец с надписью *Dir*. Параметры с пометкой *-W* доступны для записи и никогда не изменяются самим компонентом, вместо этого они предназначены для изменения пользователем, чтобы управлять компонентом. Мы увидим, как это сделать позже. Параметры, помеченные как *R-*, доступны только для чтения. Их можно изменить только компонентом. Наконец, параметр с надписью *RW* предназначен для чтения и записи. Это означает, что они изменяются компонентом, но также могут быть изменены пользователем. Примечание: параметры siggen.0.update.time и siggen.0.update.tmax предназначены для целей отладки и не будут рассматриваться в этом разделе.

Большинство компонентов реального времени экспортируют одну или несколько функций для фактического запуска содержащегося в них кода реального времени. Давайте посмотрим, какие функции экспортирует *siggen*:

**Показать функции с помощью halcmd`**

```
halcmd: show funct

Exported Functions:
Owner   CodeAddr  Arg       FP   Users  Name
00003   f801b000  fae820b8  YES      0  siggen.0.update
```

The siggen component exported a single function. It is not currently linked to any threads, so *users* is zero [1].

#### 5.4.3.3 Запуск кода в реальном времени

Чтобы фактически запустить код, содержащийся в функции `siggen.0.update`, нам нужен поток реального времени. Компонент под названием *threads*, который используется для создания нового потока. Давайте создадим поток под названием ”test-thread” с периодом 1 мс (1,000 мкс или 1,000,000 нс):

```
halcmd: loadrt threads name1=test-thread period1=1000000
```

Посмотрим, сработало ли это:

**Показать потоки**

```
halcmd: show thread

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

It did. The period is not exactly 1,000,000 ns because of hardware limitations, but we have a thread that runs at approximately the correct rate. The next step is to connect the function to the thread:

**Добавить функцию**

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

До сих пор мы использовали `halcmd` только для просмотра HAL. Однако на этот раз мы использовали команду `addf` (добавить функцию), чтобы фактически изменить что-то в HAL. Мы сказали `halcmd` добавить функцию `siggen.0.update` в поток `test-thread`, и если мы еще раз посмотрим на список потоков, то увидим, что это удалось:

```
halcmd: show thread

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

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

Прежде чем компонент *siggen* начнет генерировать сигналы, необходим еще один шаг. При первом запуске HAL поток(и) фактически не выполняются. Это позволит вам полностью настроить систему до запуска кода реального времени. Как только вы будете довольны конфигурацией, вы можете запустить код реального времени следующим образом:

```
halcmd: start
```

Теперь генератор сигналов работает. Давайте посмотрим на его выходные контакты:

```
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    -0.1640929  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
```

И давайте посмотрим еще раз:

```
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     0.0507619  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
```

Мы быстро выполнили две команды `show pin`, и вы можете видеть, что выходные данные больше не равны нулю. Выходные сигналы синуса, косинуса, пилы и треугольника постоянно изменяются. Прямоугольный выход также работает, однако он просто переключается с +1,0 на -1,0 каждый цикл.

#### 5.4.3.4 Изменение параметров

Настоящая сила HAL в том, что вы можете изменить ситуацию. Например, мы можем использовать команду `setp` для установки значения параметра. Давайте изменим амплитуду генератора сигналов с 1,0 на 5,0:

**Настроить контакт**

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

**Проверьте параметры и контакты еще раз**

```
halcmd: show param

Parameters:
Owner   Type  Dir         Value  Name
```

```
     3  s32   RO           1754  siggen.0.update.time
     3  s32   RW          16997  siggen.0.update.tmax

halcmd: show pin

Component 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
```

Обратите внимание, что значение параметра `siggen.0.amplitude` изменилось на 5, и что выводы теперь имеют большие значения.

#### 5.4.3.5 Сохранение конфигурации HAL

Большую часть того, что мы до сих пор делали с `halcmd`, было просто просмотром чего-либо с помощью команды show. Однако две команды фактически изменили ситуацию. По мере того, как мы проектируем более сложные системы с помощью HAL, мы будем использовать множество команд для настройки так, как мы хотим. HAL имеет память слона и сохранит эту конфигурацию, пока мы его не выключим. Но что насчет следующего раза? Мы не хотим вручную вводить кучу команд каждый раз, когда хотим использовать систему.

**Сохранение конфигурации всего HAL одной командой.**

```
halcmd: save

# components
loadrt threads name1=test-thread period1=1000000
loadrt siggen
# pin aliases
# signals
# nets
# parameter values
setp siggen.0.update.tmax 14687
# realtime thread/function links
addf siggen.0.update test-thread
```

Вывод команды `save` представляет собой последовательность команд HAL. Если вы начнете с *empty* HAL и запустите все эти команды, вы получите конфигурацию, которая существовала на момент выполнения команды *save* . Чтобы сохранить эти команды для дальнейшего использования, мы просто перенаправляем вывод в файл:

**Сохраните конфигурацию в файл с помощью halcmd**

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

#### 5.4.3.6 Выход из halrun

Когда вы закончите сеанс HAL, введите `exit` в командной строке `halcmd:`. Это вернет вас к системному приглашению и закроет сеанс HAL. Не закрывайте просто окно терминала, не завершив сеанс HAL.

**Выход из HAL**

```
halcmd: exit
```

#### 5.4.3.7 Восстановление конфигурации HAL

Чтобы восстановить конфигурацию HAL, хранящуюся в файле ”saved.hal”, нам нужно выполнить все эти команды HAL. Для этого мы используем ключи ”`-f _<file name>_`”, который считывает команды из файла, и «`-I`» (заглавная буква i), который показывает приглашение halcmd после выполнения команд:

**Запуск сохраненного файла**

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

Обратите внимание, что в файле saved.hal нет команды ”start”. Необходимо ввести ее снова (или отредактировать файл saved.hal, чтобы добавить ее туда).

#### 5.4.3.8 Удаление HAL из памяти

Если произойдет неожиданное завершение сеанса HAL, возможно, вам придется выгрузить HAL перед началом следующего сеанса. Для этого введите следующую команду в окне терминала.

**Удаление HAL**

```
halrun -U
```

### 5.4.4 Halmeter

Вы можете создавать очень сложные системы HAL, даже не используя графический интерфейс. Однако есть что-то приятное в том, чтобы видеть результат своей работы. Первый и самый простой инструмент с ГИП для HAL — это halmeter. Это очень простая программа, которая является HAL-эквивалентом удобного мультиметра (или аналогового измерителя для олдов).

Это позволяет наблюдать за контактами, сигналами или параметрами, отображая текущие значения этих объектов. Это очень простое в использовании приложение для графических сред. В консоли введите:

```
halmeter
```

Появятся два окна. Окно выбора самое большое и включает в себя три вкладки:

- В одном перечислены все контакты, определенные в настоящее время в HAL,
- в другом перечислены все сигналы,
- в третьем перечислены все параметры.

Нажмите на вкладку, затем нажмите на один из элементов, чтобы выбрать его. В небольшом окне будет показано имя и значение выбранного элемента. Дисплей обновляется примерно 10 раз в секунду. Чтобы освободить место на экране, окно выбора можно закрыть кнопкой *Close*. В маленьком окне, скрытом под окном выбора при запуске программы, кнопка *Select* повторно открывает окно выбора, а кнопка *Exit* останавливает программу и закрывает оба окна.

Возможен запуск нескольких halmeter’ов одновременно, что дает возможность визуализировать несколько элементов одновременно. Чтобы открыть halmeter и высвободить консоль, запустив его в фоновом режиме, выполните следующую команду:

```
halmeter &
```

Можно запустить halmeter и заставить его сразу отображать элемент. Для этого добавьте аргументы *pin|sig|par[am] name* в командную строку. Он отобразит сигнал, контакт или параметр *name*, как только запустится. Если указанный элемент не существует, то он запустится нормально.

Наконец, если элемент указан для отображения, можно добавить *-s* перед pin|sig|param, чтобы указать halmeter использовать еще меньшее окно. Имя элемента будет отображаться в строке заголовка, а не под значением, и кнопки не будет. Это полезно для отображения большого количества halmeter'ов в небольшом пространстве.

Мы снова будем использовать компонент siggen, чтобы проверить halmeter. Если вы только что закончили предыдущий пример, то вы можете загрузить siggen, используя сохраненный файл. Если нет, мы можем загрузить его так же, как делали это раньше:

```
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
```

На данный момент у нас загружен и запущен компонент siggen. Пришло время запустить halmeter.

**Запуск Halmeter**

```
halcmd: loadusr halmeter
```

Первое окно, которое вы увидите, — это окно "Select Item to Probe".

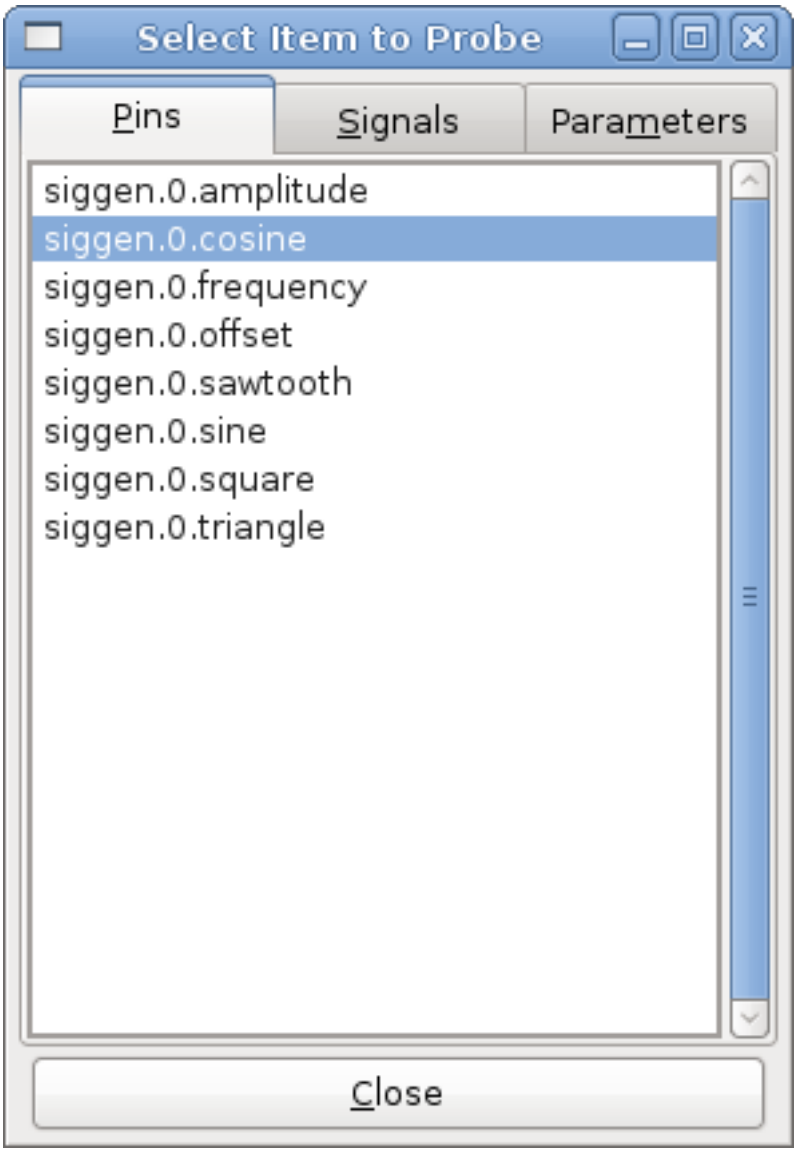



Figure 5.5: Halmeter Окно выбора

Этот диалог имеет три вкладки. На первой вкладке отображаются все контакты HAL в системе. На втором отображаются все сигналы, а на третьем — все параметры. Мы хотели бы сначала посмотреть на контакт `siggen.0.cosine`, поэтому щелкните его, а затем нажмите кнопку "Close". Диалоговое окно выбора объекта для измерения закроется, и окно измерения будет выглядеть примерно так, как показано на следующем рисунке.

Figure 5.6: Halmeter Окно

Чтобы изменить то, что отображает измеритель, нажмите кнопку ”Select”, которая возвращает окно ”Select Item to Probe”.

Вы должны увидеть, как значение меняется по мере того, как siggen генерирует косинусоидальную волну. Halmeter обновляет дисплей примерно 5 раз в секунду.

Чтобы выключить Halmeter, просто нажмите кнопку выхода.

Если вы хотите просмотреть более одного контакта, сигнала или параметра одновременно, вы можете просто запустить больше halmeter’ов. Окно halmeter было намеренно сделано очень маленьким, чтобы на экране могло быть одновременно много изображений.

### 5.4.5 Stepgen Пример

До сих пор мы загрузили только один компонент HAL. Но вся идея HAL заключается в том, чтобы позволить вам загружать и соединять ряд простых компонентов для создания сложной системы. В следующем примере будут использоваться два компонента.

Прежде чем мы сможем приступить к созданию этого нового примера, мы хотим начать с чистого листа. Если вы только что закончили один из предыдущих примеров, нам нужно удалить все компоненты и перезагрузить библиотеки RTAPI и HAL.

```
halcmd: exit
```

#### 5.4.5.1 Установка компонентов

Теперь мы собираемся загрузить компонент генератора шаговых импульсов. Подробное описание этого компонента см. в разделе «Stepgen» Руководства интегратора. В этом примере мы будем использовать тип управления StepGen *velocity*. На данный момент мы можем пропустить детали и просто выполнить следующие команды.

В этом примере мы будем использовать тип управления *velocity* из компонента `stepgen`.

```
halrun
halcmd: loadrt stepgen step_type=0,0 ctrl_type=v,v
halcmd: loadrt siggen
halcmd: loadrt threads name1=fast period1=50000 name2=slow period2=1000000
```

The first command loads two step generators, both configured to generate stepping type 0. The second command loads our old friend siggen, and the third one creates two threads, a fast one with a period of 50 microseconds (µs) and a slow one with a period of 1 millisecond (ms).

---

**Note**
The `fp1=` parameter is deprecated and ignored. All threads now unconditionally support floating point.

---

Как и прежде, мы можем использовать `halcmd show`, чтобы просмотреть HAL. На этот раз у нас гораздо больше контактов и параметров, чем раньше:

```
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 Соединение контактов с сигналами

У нас есть два генератора шаговых импульсов и генератор сигналов. Теперь пришло время создать несколько сигналов HAL для соединения двух компонентов. Мы собираемся представить, что два генератора шаговых импульсов приводят в движение оси X и Y станка. Мы хотим перемещать стол по кругу. Для этого мы отправим косинусный сигнал на ось X, а синусоидальный сигнал на ось Y. Модуль siggen создает синус и косинус, но нам нужны «провода» для соединения модулей вместе. В HAL «провода» называются сигналами. Нам нужно создать два из них. Мы можем называть их как угодно, в этом примере это будут «X-vel» и «Y-vel». Сигнал «X-vel» предназначен для передачи от косинусного выхода генератора сигналов на вход скорости первого генератора шаговых импульсов. Первым шагом является подключение сигнала к выходу генератора сигналов. Чтобы подключить сигнал к выводу, мы используем команду net.

**net command**

```
halcmd: net X-vel <= siggen.0.cosine
```

Чтобы увидеть эффект от команды `net`, мы снова покажем сигналы.

```
halcmd: show sig

Signals:
Type          Value  Name     (linked to)
float             0  X-vel <== siggen.0.cosine
```

Когда сигнал подключен к одному или нескольким контактам, команда show выводит список контактов сразу после имени сигнала. *Стрелка* показывает направление потока данных — в данном случае данные передаются от контакта «siggen.0.cosine» к сигналу «X-vel». Теперь давайте подключим X-vel ко входу скорости генератора шаговых импульсов.

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

Мы также можем подключить сигнал оси Y `Y-vel`. Он предназначен для работы от синусоидального выхода генератора сигналов до входа второго генератора шаговых импульсов. Следующая команда в одной строке выполняет то, что две команды `net` выполнили для `X-vel`.

```
halcmd: net Y-vel siggen.0.sine => stepgen.1.velocity-cmd
```

Теперь давайте еще раз взглянем на сигналы и подключенные к ним контакты.

```
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
```

Команда *show sig* поясняет, как именно данные проходят через HAL. Например, сигнал *X-vel* поступает с контакта *siggen.0.cosine* и идет на контакт *stepgen.0.velocity-cmd*.

#### 5.4.5.3 Настройка выполнения в реальном времени — потоки и функции

Представление данных, проходящих по ”проводам”, позволяет достаточно легко понять контакты и сигналы . С потоками и функциями немного сложнее. Функции содержат компьютерные инструкции, которые действительно позволяют добиться цели. Поток — это метод, используемый для запуска этих инструкций, когда они необходимы. Для начала давайте посмотрим на доступные нам функции.

```
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
```

В общем, вам придется обратиться к документации каждого компонента, чтобы узнать, какие функции он выполняет. В этом случае функция `siggen.0.update` используется для обновления выходов генератора сигналов. Каждый раз, когда он выполняется, он вычисляет значения выходных сигналов синуса, косинуса, треугольника и прямоугольника. Чтобы сигналы были плавными, его необходимо запускать через определенные промежутки времени.

Остальные три функции относятся к генераторам шаговых импульсов.

Первый, `stepgen.capture_position`, используется для обратной связи по положению. Он фиксирует значение внутреннего счетчика, который считает шаговые импульсы по мере их генерации. При отсутствии пропущенных импульсов этот счетчик показывает положение двигателя.

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.

Последняя функция, `stepgen.update-freq`, отвечает за масштабирование и некоторые другие вычисления, которые необходимо выполнять только при изменении команды частоты.

В нашем примере это означает, что мы хотим запускать `siggen.0.update` с умеренной скоростью для вычисления значений синуса и косинуса. Сразу после запуска `siggen.0.update` мы хотим запустить `stepgen.update_freq`, чтобы загрузить новые значения в генератор шаговых импульсов. Наконец, нам нужно запускать `stepgen.make_pulses` как можно быстрее для получения плавных импульсов. Поскольку мы не используем обратную связь по положению, нам вообще не нужно запускать `stepgen.capture_position`.

Мы запускаем функции, добавляя их в потоки. Каждый поток работает с определенной скоростью. Давайте посмотрим, какие потоки у нас есть.

```
halcmd: show thread

Realtime Threads:
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                slow (        0,        0 )
      49849  YES                fast (        0,        0 )
```

The two threads were created when we loaded `threads`. The first one, *slow*, runs every millisecond. We will use it for `siggen.0.update` and `stepgen.update_freq`. The second thread is *fast*, which runs every 50 microseconds (µs). We will use it for `stepgen.make_pulses`. To connect the functions to the proper thread, we use the `addf` command. We specify the function first, followed by the thread.

```
halcmd: addf siggen.0.update slow
halcmd: addf stepgen.update-freq slow
halcmd: addf stepgen.make-pulses fast
```

После того, как мы дадим эти команды, мы можем снова запустить команду `show thread`, чтобы увидеть, что произошло.

```
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
```

Теперь за каждым потоком следуют имена функций в том порядке, в котором они будут выполняться.

#### 5.4.5.4 Настройка параметров

Мы почти готовы запустить нашу систему HAL. Однако нам все еще нужно настроить несколько параметров. По умолчанию компонент siggen генерирует сигналы, которые колеблются от +1 до -1. Для нашего примера это нормально: мы хотим, чтобы скорость стола менялась от +1 до -1 дюйма в секунду. Однако масштабирование генератора шаговых импульсов не совсем правильное. По умолчанию он генерирует выходную частоту 1 импульс в секунду при входном значении 1,0. Вряд ли один импульс в секунду даст нам движение стола на один дюйм в секунду. Вместо этого предположим, что у нас есть ШВП со скоростью 5 оборотов на дюйм, соединенная с шаговым двигателем со скоростью 200 шагов на оборот и 10-кратным микрошагом. Таким образом, для одного оборота винта требуется 2000 шагов, а для перемещения на один дюйм — 5 оборотов. Это означает, что общее масштабирование составляет 10 000 шагов на дюйм. Нам нужно умножить входную скорость, подаваемую на генератор шаговых импульсов, на 10000, чтобы получить правильный выходной сигнал. Именно для этого нужен параметр `stepgen.n.position` В этом случае оси X и Y имеют одинаковое масштабирование, поэтому мы устанавливаем параметры масштабирования для обеих на 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
```

Это масштабирование скорости означает, что когда контакт `stepgen.0.velocity-cmd` равен 1,0, генератор шагов будет генерировать 10000 импульсов в секунду (10 кГц). При использовании двигателя и ШВП, описанных выше, ось будет перемещаться со скоростью ровно 1,0 дюйм в секунду. Это иллюстрирует ключевую концепцию HAL — такие операции, как масштабирование, выполняются на минимально возможном уровне, в данном случае в генераторе шаговых импульсов. Внутренний сигнал `X-vel` — это скорость стола в дюймах в секунду, а другие компоненты, такие как `siggen`, вообще не знают (или заботятся) о масштабировании. Если бы мы поменяли ШВП или двигатель, мы бы изменили только параметр масштабирования генератора шаговых импульсов.

#### 5.4.5.5 Запустить его!

Теперь у нас все настроено и мы готовы к запуску. Как и в первом примере, мы используем команду `start`.

```
halcmd: start
```

Хотя кажется, что ничего не происходит, внутри компьютера генератор шаговых импульсов каждую секунду выдает шаговые импульсы с частотой от 10 кГц вперед до 10 кГц назад и обратно. Позже в этом уроке мы увидим, как использовать эти внутренние сигналы для запуска двигателей в реальном мире, но сначала мы хотим взглянуть на них и посмотреть, что происходит.

### 5.4.6 Halscope

Предыдущий пример генерирует несколько очень интересных сигналов. Но многое из того, что происходит, происходит слишком быстро, чтобы увидеть с помощью halmeter. Чтобы поближе

взглянуть на то, что происходит внутри HAL, нам понадобится осциллограф. К счастью, у HAL есть такой инструмент, который называется halscope.

Halscope состоит из двух частей: части реального времени, которая считывает сигналы HAL, и части не реального времени, которая обеспечивает ГИП и отображение. Однако вам не нужно об этом беспокоиться, поскольку часть, не работающая в реальном времени, автоматически загрузит часть реального времени, когда это необходимо.

Когда LinuxCNC работает в терминале, вы можете запустить halscope с помощью следующей команды.

**Запуск Halscope**

```
halcmd loadusr halscope
```

Если LinuxCNC не запущен или файл autosave.halscope не соответствует контактам, доступным в текущем работающем LinuxCNC, откроется окно ГИП осциллографа, за которым сразу последует диалоговое окно *Realtime function not linked*, которое выглядит как показано на следующем рисунке. Чтобы изменить частоту дискретизации, щелкните левой кнопкой мыши на поле семплов.

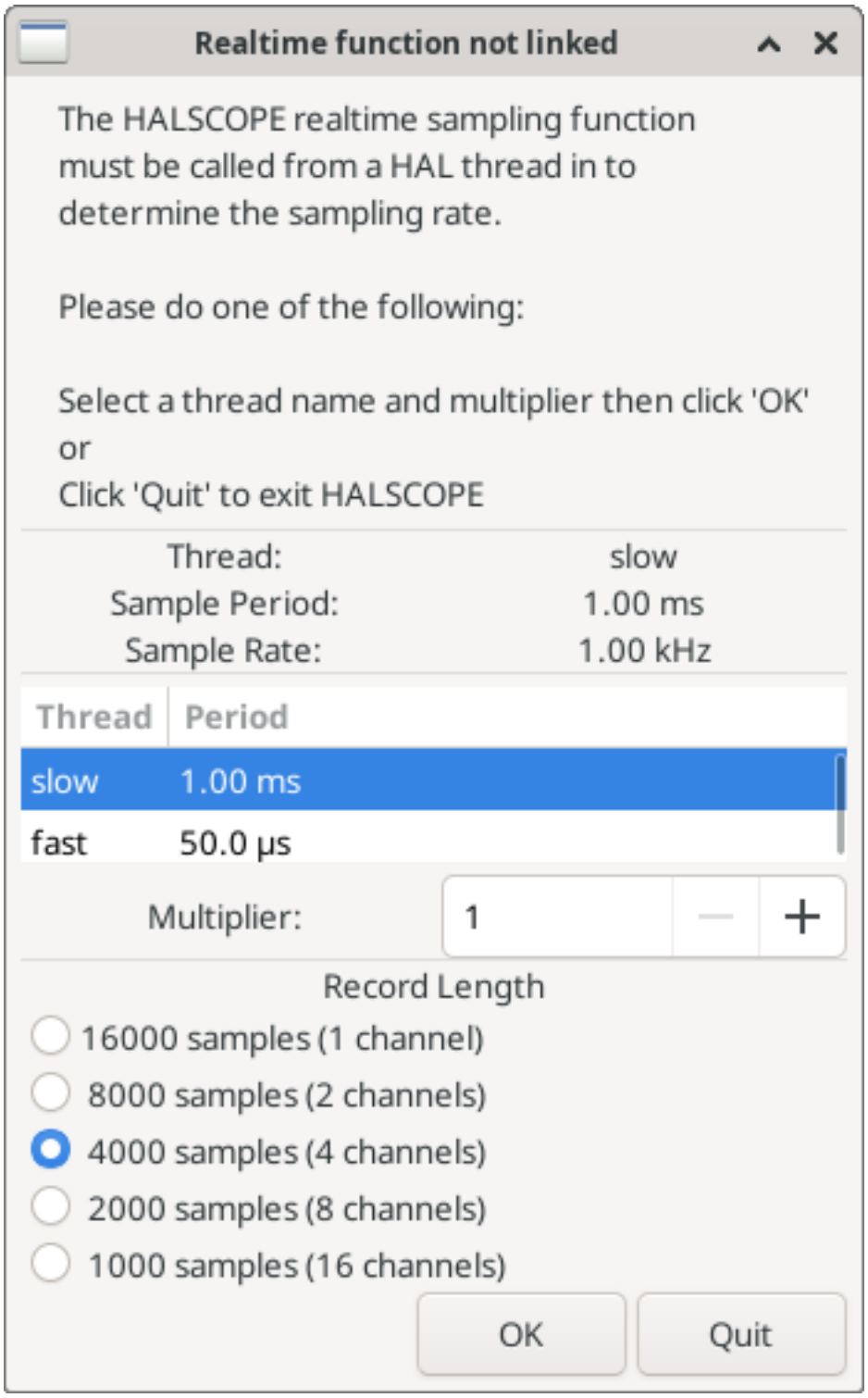



Figure 5.7: Диалоговое окно не связанное с функцией реального времени

В этом диалоговом окне вы устанавливаете частоту дискретизации осциллографа. На данный момент мы хотим производить выборку один раз в миллисекунду, поэтому нажмите на *медленный* поток 989 мкс и оставьте множитель равным 1. Мы также оставим длину записи равной 4000 выборок, чтобы мы могли использовать до четырех каналов одновременно. Когда вы выбираете поток и нажимаете *OK*, диалоговое окно исчезает, а окно области действия выглядит примерно

так, как показано на следующем рисунке.

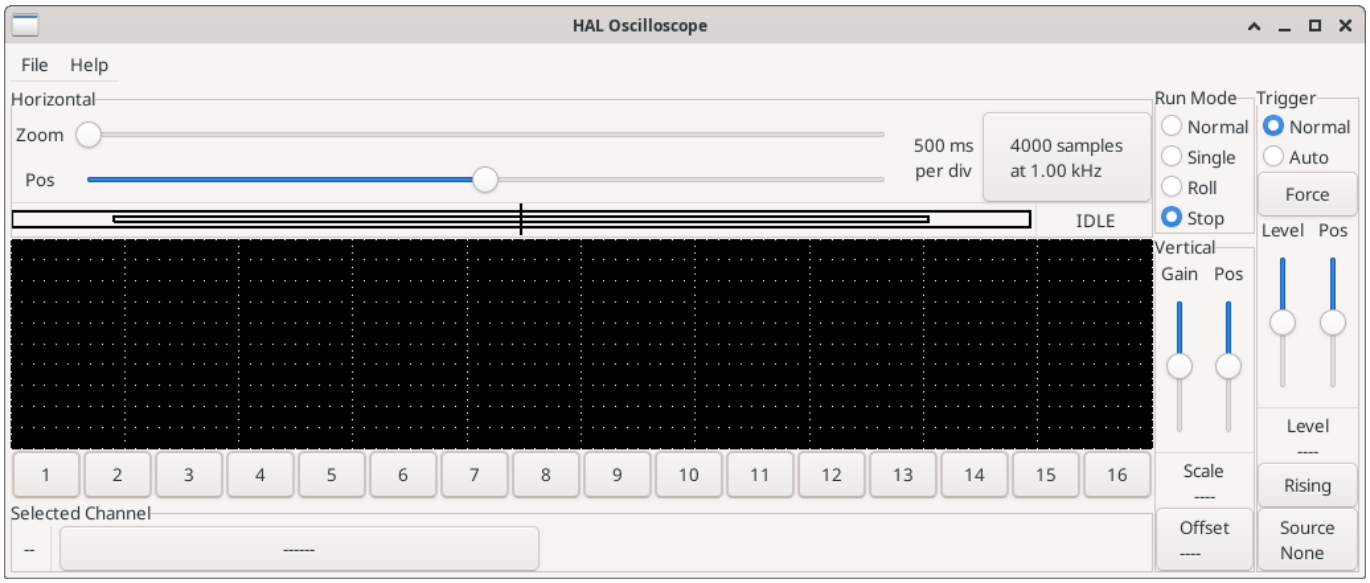



Figure 5.8: Начальное окно осциллографа

#### 5.4.6.1 Подключение щупов

На этом этапе Halscope готов к использованию. Мы уже выбрали частоту дискретизации и длину записи, поэтому следующий шаг — решить, на что обратить внимание. Это эквивалентно подключению *виртуальных щупов осциллографа* к HAL. Halscope имеет 16 каналов, но количество, которое вы можете использовать одновременно, зависит от длины записи: больше каналов означает более короткие записи, поскольку доступная для записи память фиксирована и составляет примерно 16 000 выборок.

Кнопки каналов расположены в нижней части экрана галоскопа. Нажмите кнопку *1*, и вы увидите диалоговое окно *Select Channel Source*, как показано на следующем рисунке. Этот диалог очень похож на тот, который использует Halmeter. Нам хотелось бы просмотреть сигналы, которые мы определили ранее, поэтому мы нажимаем вкладку *Signals*, и в диалоговом окне отображаются все сигналы в HAL (только два для этого примера).

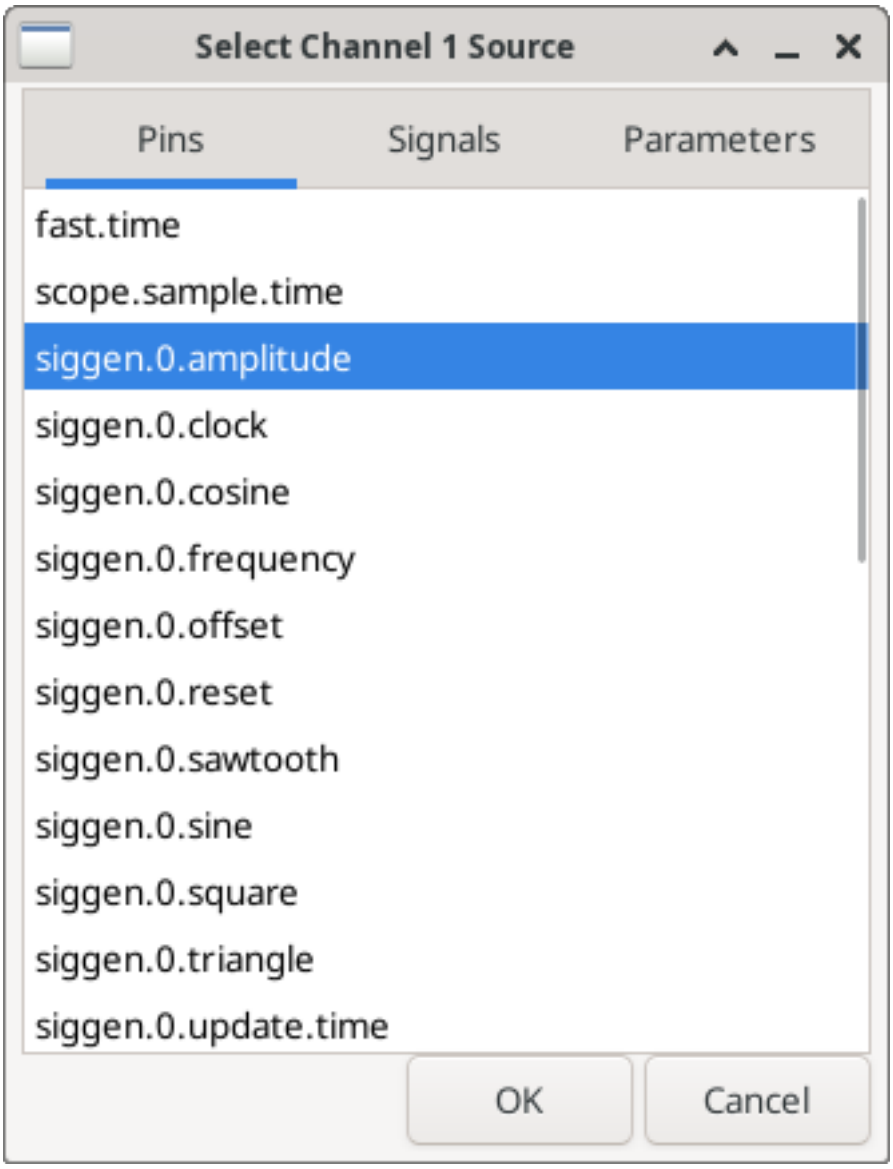



Figure 5.9: Выбор канала источника

Чтобы выбрать сигнал, просто нажмите на него. В данном случае мы хотим, чтобы канал 1 отображал сигнал *X-vel*. Нажмите вкладку «Сигналы», затем нажмите *X-vel*, диалоговое окно закроется, и канал будет выбран.

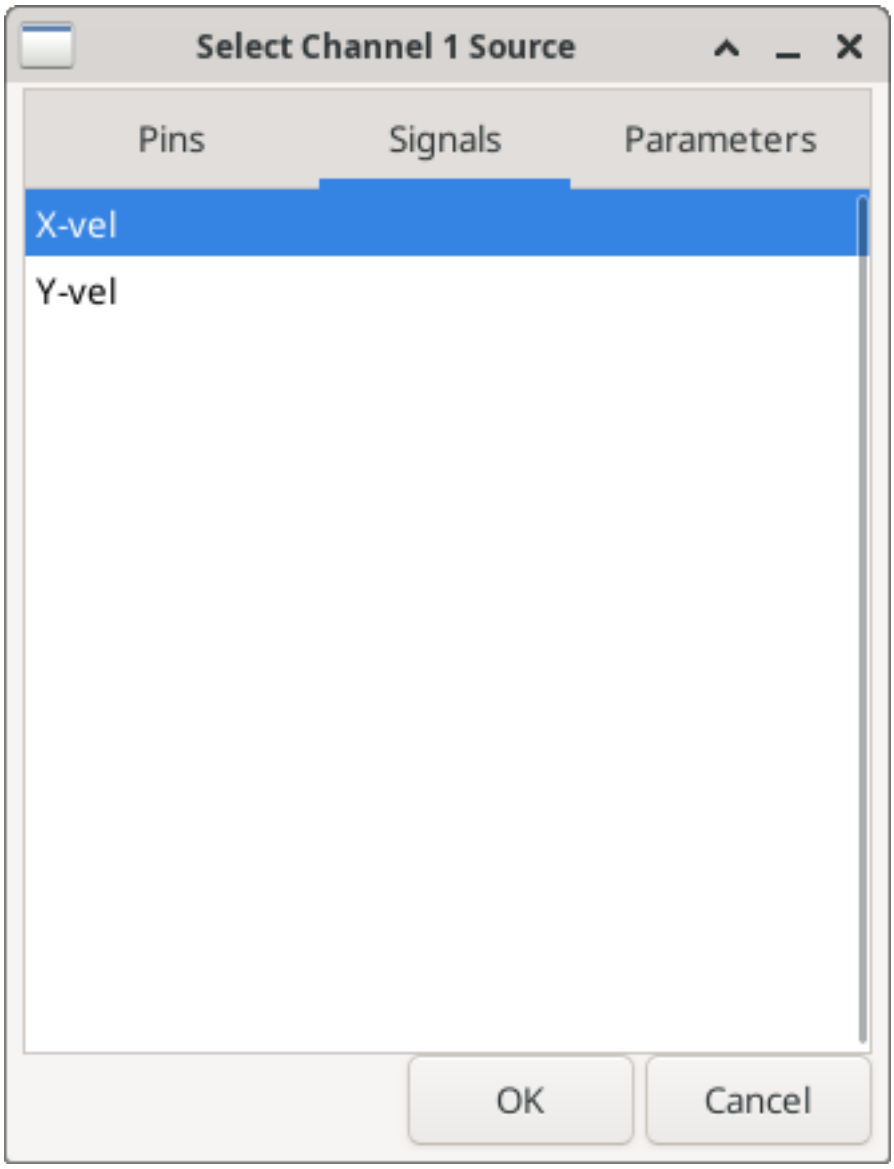



Figure 5.10: Выбрать сигнал

Кнопка канала 1 нажата, и под рядом кнопок появляются номер канала 1 и название *X-vel*. На этом дисплее всегда отображается выбранный канал — на экране может быть много каналов, но выбранный будет выделен, а различные элементы управления, такие как вертикальное положение и масштаб, всегда будут работать с выбранным каналом.

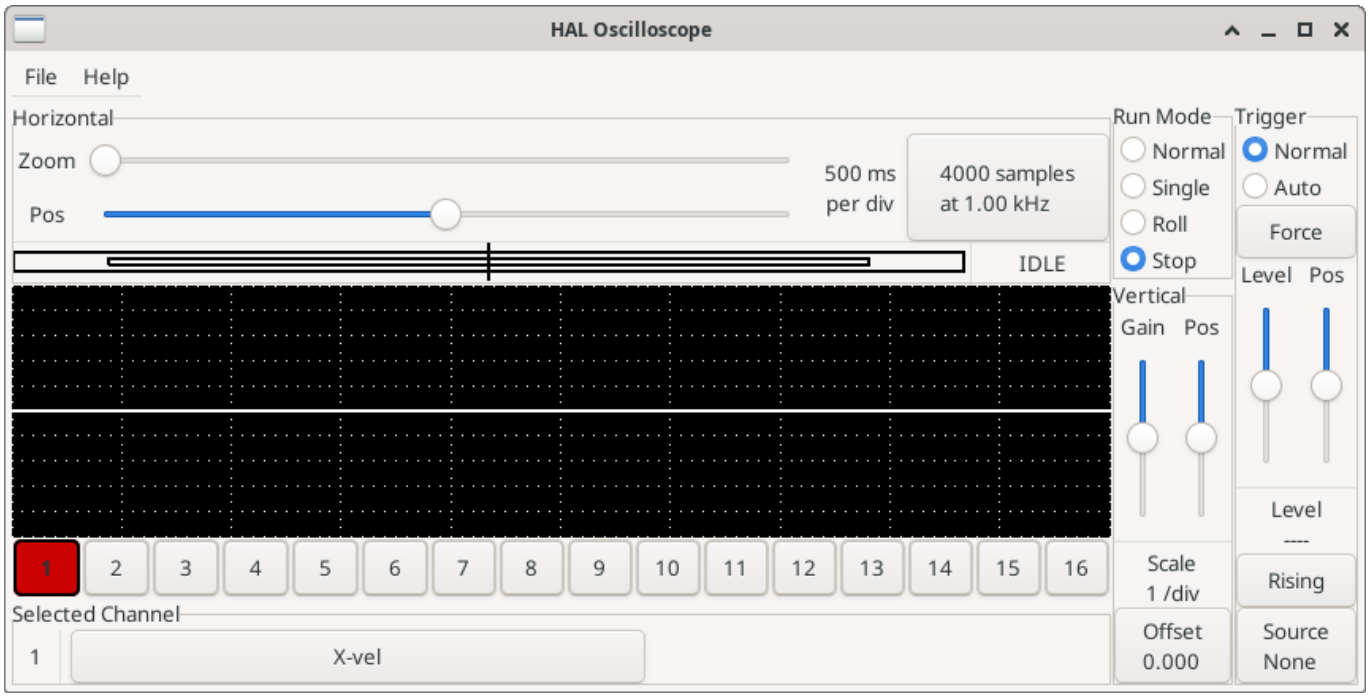



Figure 5.11: Halscope

Чтобы добавить сигнал на канал 2, нажмите кнопку *2*. Когда появится диалоговое окно, перейдите на вкладку *Signals*, затем нажмите *Y-vel*. Мы также хотим посмотреть на выходные сигналы прямоугольных и треугольных волн. К этим контактам не подключены никакие сигналы, поэтому вместо этого мы используем вкладку *Pins*. Для канала 3 выберите *siggen.0.triangle*, а для канала 4 — *siggen.0.square*.

#### 5.4.6.2 Захват наших первых сигналов

Теперь, когда к HAL подключено несколько щупов, пришло время посмотреть несколько сигналов. Чтобы запустить halscope, нажмите кнопку *Normal* в разделе экрана *Run Mode* (вверху справа). Поскольку у нас есть длина записи 4000 выборок и мы получаем 1000 выборок в секунду, halscope потребуется около 2 секунд, чтобы заполнить половину буфера. В это время индикатор выполнения над главным экраном будет показывать заполнение буфера. Как только буфер заполнится наполовину, halscope ожидает триггера. Поскольку мы его еще не настроили, он будет ждать вечно. Чтобы запустить его вручную, нажмите кнопку *Force* в разделе *Trigger* в правом верхнем углу. Вы должны увидеть заполнение оставшейся части буфера, после чего на экране отобразятся захваченные сигналы. Результат будет выглядеть примерно так, как показано на следующем рисунке.

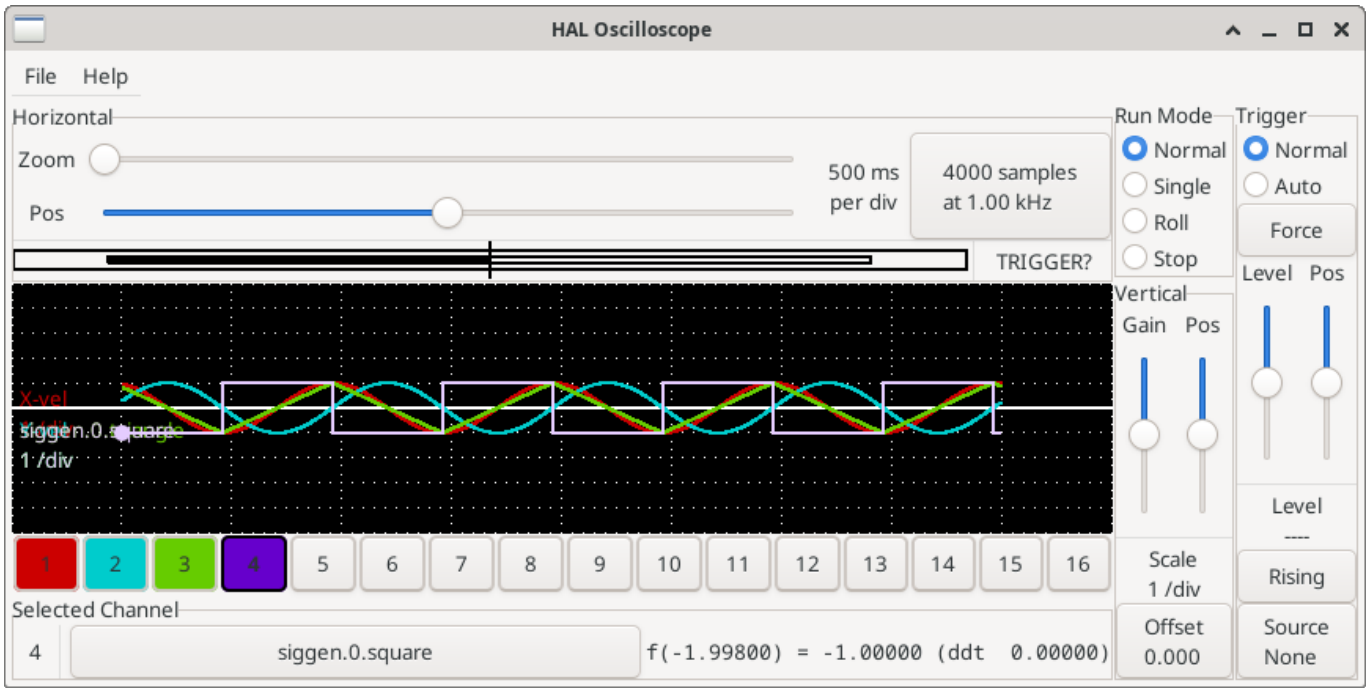



Figure 5.12: Захваченные сигналы

В поле *Selected Channel* внизу указано, что фиолетовая кривая — это выбранный в данный момент канал 4, который отображает значение вывода `siggen.0.square`. Попробуйте нажать кнопки каналов с 1 по 3, чтобы выделить три другие кривые.

#### 5.4.6.3 Вертикальная регулировки

Сигналы довольно сложно различить, поскольку все четыре находятся друг на друге. Чтобы это исправить, мы используем элементы управления *Vertical* в поле справа от экрана. Эти элементы управления действуют на текущий выбранный канал. При настройке усиления обратите внимание, что он охватывает огромный диапазон — в отличие от настоящего осциллографа, он может отображать сигналы в диапазоне от очень маленьких (пико-единицы) до очень больших (тера-единицы). Элемент управления положением перемещает отображаемый сигнал вверх и вниз только по высоте экрана. Для более крупных регулировок следует использовать кнопку смещения.

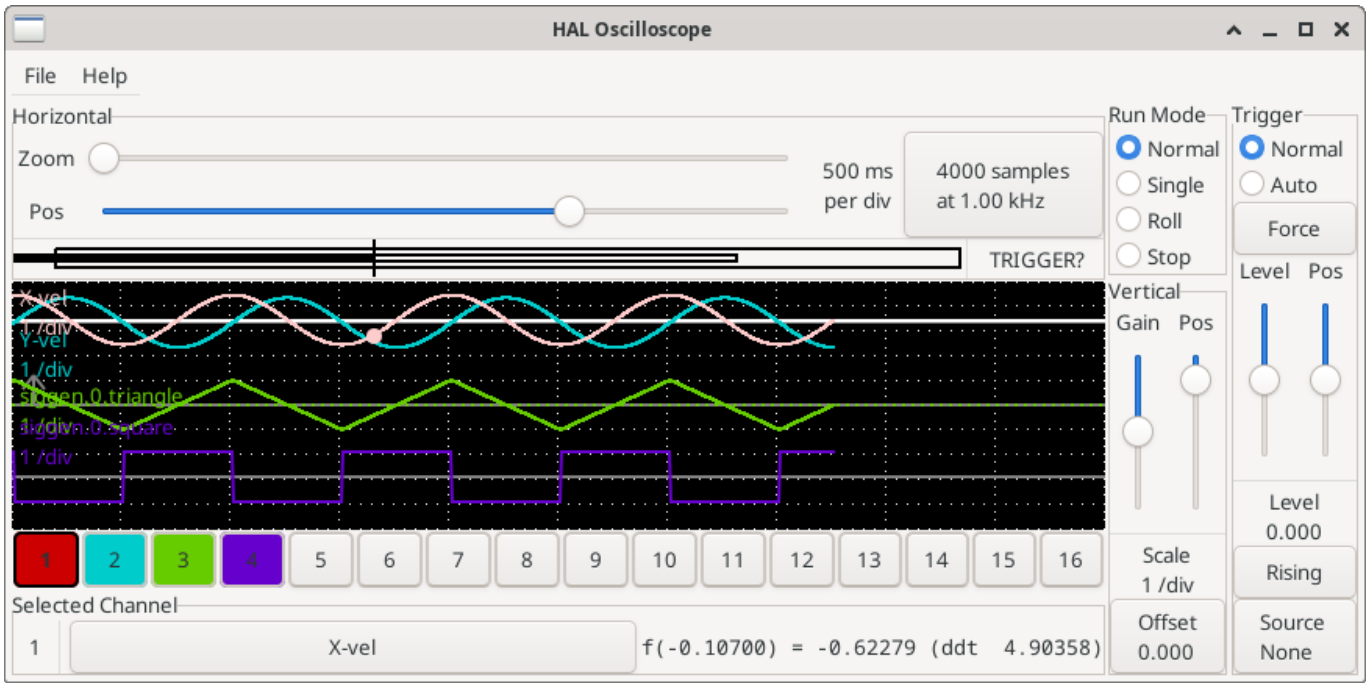



Figure 5.13: Вертикальная регулировка

Большая кнопка *Selected Channel* внизу указывает, что канал 1 в данный момент выбран и соответствует сигналу *X-vel*. Попробуйте нажать на другие каналы, чтобы увидеть их сигналы и иметь возможность перемещать их с помощью курсора *Pos*.

#### 5.4.6.4 Запуск развертки

Использование кнопки *Force* — довольно неудовлетворительный способ активировать осциллограф. Чтобы настроить реальный триггер, нажмите кнопку *Source* справа внизу. Появится диалоговое окно *Trigger Source*, которое представляет собой просто список всех щупов, которые в данный момент подключены. Выберите щуп, который будет использоваться для запуска, щелкнув по нему. В этом примере мы будем использовать канал 3, треугольную волну, как показано на следующем рисунке.

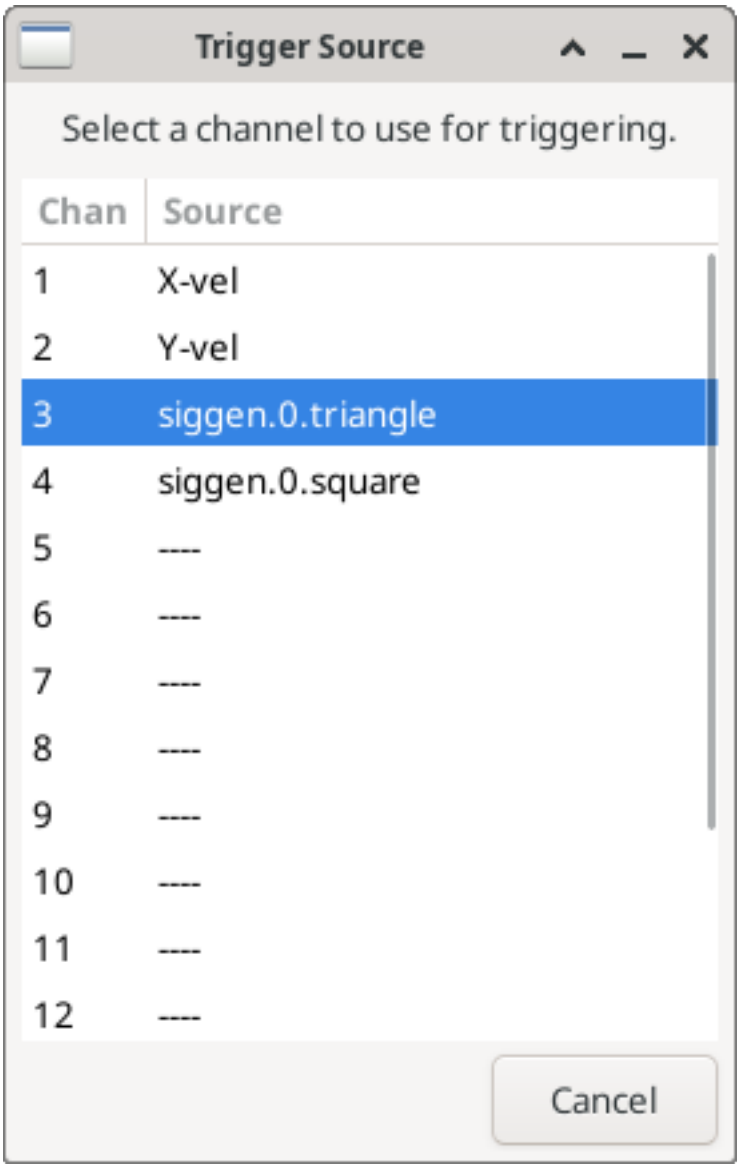



Figure 5.14: Диалог источника запуска

После установки источника триггера вы можете настроить уровень и положение триггера с помощью ползунков в поле *Trigger* вдоль правого края. Уровень можно регулировать сверху вниз по экрану, он отображается под ползунками. Позиция — это расположение триггерной точки в общей записи. Если ползунок полностью опущен, точка запуска находится в конце записи, а halscope отображает то, что произошло до точки запуска. Когда ползунок полностью поднят, точка запуска находится в начале записи, отображая то, что произошло после ее запуска. Точка запуска отображается в виде вертикальной линии в окне выполнения над экраном. Полярность триггера можно изменить, нажав кнопку чуть ниже дисплея уровня триггера. Тогда он станет *descendant*. Обратите внимание, что изменение положения триггера останавливает осциллограф, как только положение было отрегулировано. Вы перезапускаете осциллограф, нажав кнопку *Normal* в группе *Run mode* .

Теперь, когда мы настроили вертикальные элементы управления и триггер, отображение осциллограф выглядит примерно так, как показано на следующем рисунке.

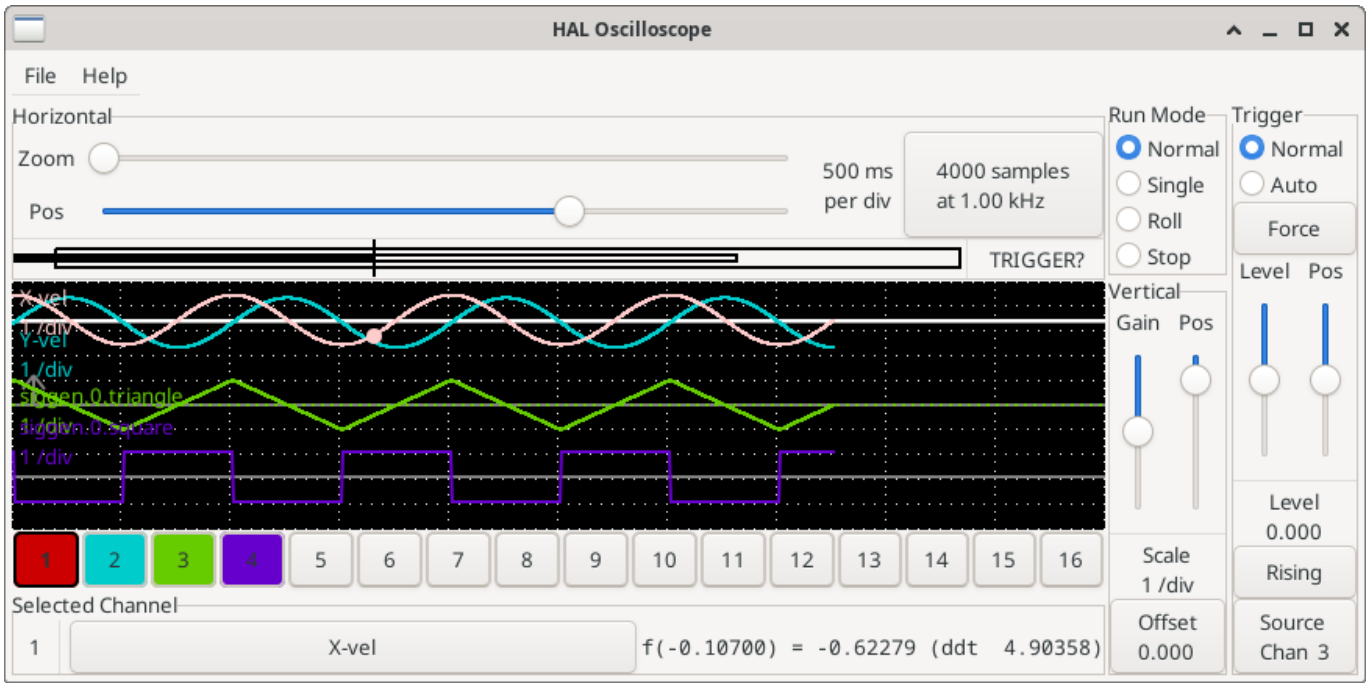



Figure 5.15: Сигналы с запуском

#### 5.4.6.5 Горизонтальные регулировки

Чтобы внимательно рассмотреть часть сигнала, вы можете использовать ползунок масштабирования в верхней части экрана, чтобы развернуть сигналы по горизонтали, а также ползунок положения, чтобы определить, какая часть увеличенного сигнала видна. Однако, иногда простого расширения формы сигнала недостаточно, и необходимо увеличить частоту дискретизации. Например, мы хотели бы взглянуть на фактические шаговые импульсы, которые генерируются в нашем примере. Поскольку длительность шаговых импульсов может составлять всего 50 мкс, выборка на частоте 1 кГц недостаточна. Для изменения частоты дискретизации, нажмите кнопку, которая отображает количество выборок и частоту дискретизации, чтобы открыть диалоговое окно *Select Sample Rate*. В этом примере мы выберем поток 50 мкс, *быстрый*, что даст нам частоту дискретизации около 20 кГц. Теперь вместо отображения данных за 4 секунды одна запись содержит 4000 выборок с частотой 20 кГц, или около 0,20 секунды.

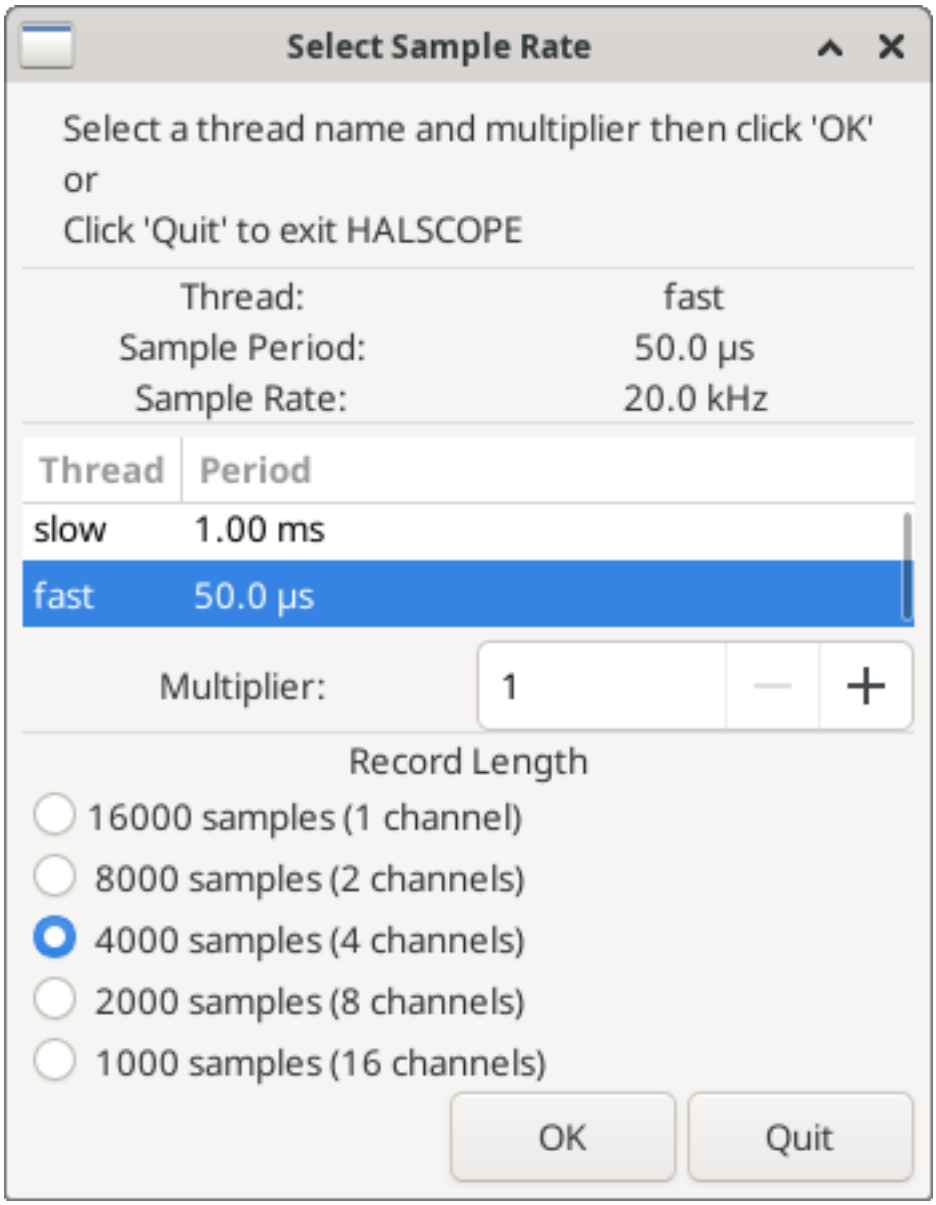



Figure 5.16: Диалог частоты выборки

#### 5.4.6.6 Больше каналов

Теперь давайте посмотрим на импульсы шага. У Halscope 16 каналов, но в этом примере мы используем только 4 за раз. Прежде чем выбирать другие каналы, нам нужно отключить несколько из них. Нажатие на кнопку выбранного канала (черная рамка) выключит канал. Поэтому нажмите кнопку канала 2, затем снова нажмите на эту кнопку, и канал выключится. Затем дважды щелкните на канале 3 и сделайте то же самое для канала 4. Несмотря на то, что каналы выключены, они все еще помнят, к чему они подключены, и фактически мы продолжим использовать канал 3 в качестве источника запуска. Чтобы добавить новые каналы, выберите канал 5 и выберите вывод `stepgen.0.dir`, затем канал 6 и выберите `stepgen.0.step`. Затем выберите режим работы `Normal`, чтобы запустить осциллограф, и настройте горизонтальное масштабирование до 5 мс на деление. Вы должны увидеть, как импульсы шага замедляются по мере того, как команда скорости (канал 1) приближается к нулю, затем вывод направления меняет состояние, и импульсы шага снова ускоряются. Возможно, вам стоит увеличить усиление на канале 1

примерно до 20 миллисекунд на деление, чтобы лучше увидеть изменение команды скорости. Результат должен выглядеть, как показано на следующем рисунке.

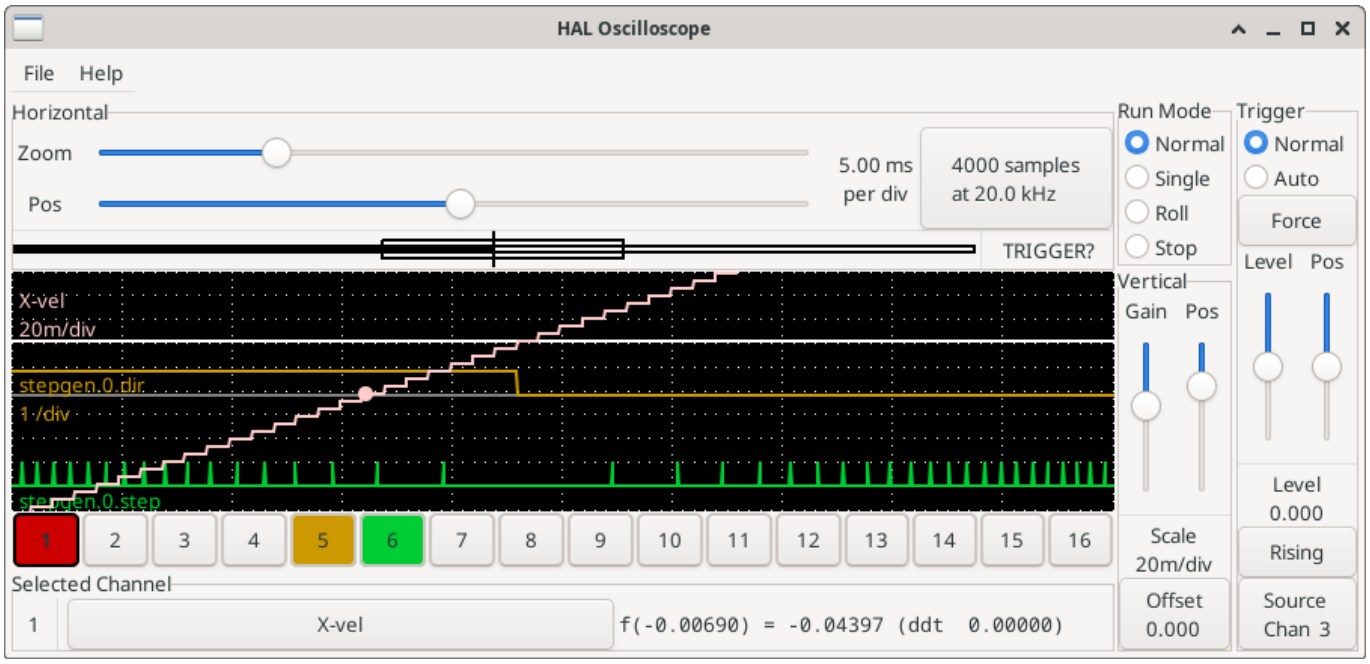



Figure 5.17: Шаговые импульсы

#### 5.4.6.7 Больше выборок

Если вы хотите записать больше выборок одновременно, перезапустите режим реального времени и загрузите halscope с числовым аргументом, который указывает количество выборок, которые вы хотите захватить.

```
halcmd loadusr halscope 80000
```

Если компонент *scope_rt* еще не был загружен, halscope загрузит его и запросит 80 000 общих выборок, так что при выборке 4 каналов одновременно будет 20 000 выборок на канал. (Если *scope_rt* уже загружен, числовой аргумент halscope не будет иметь никакого эффекта).

## 5.5 HAL Примеры

Во всех этих примерах предполагается, что вы начинаете с конфигурации на основе StepConf и имеете два потока *base-thread* и *servo-thread*. Мастер StepConf создаст пустые файлы custom.hal и custom_postgui.hal. Файл custom.hal будет загружен после файла конфигурации HAL, а файл custom_postgui.hal загружается после загрузки ГИП.

### 5.5.1 Подключение двух выходов

Чтобы соединить два выхода со входом, вы можете использовать компонент or2. or2 работает следующим образом: если какой-либо вход or2 включен, то и выход or2 включен. Если ни один из входов or2 не включен, выход or2 выключен.

Например, чтобы две кнопки PyVCP были подключены к одному светодиоду.

**Файл .xml, дающий указание PyVCP подготовить графический интерфейс с двумя кнопками (с именами ”button-1” и ”button-2”) и светодиодом (с именем ”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>
```

**Файл postgui.hal, считывается после настройки ГИП и готовности портов принять логику, описанную в 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
```

Когда вы запустите этот пример в симуляторе оси, созданном с помощью мастера StepConf, вы можете открыть терминал и просмотреть выводы, созданные с помощью `loadrt or2`, набрав в терминале `halcmd show pin or2`.

**Запуск halcmd в командной строке UNIX, чтобы отобразить контакты, созданные с помощью модуля 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
```

Из команды HAL `show pin or2` вы можете видеть, что вывод `button-1` подключен к выводу `or2.0.in0`. По стрелке направления вы можете видеть, что кнопка является выходом, а `or2.0.in0` — входом. Выход с or2 поступает на вход светодиода.

### 5.5.2 Ручная смена инструмента

В этом примере предполагается, что вы *развертываете свою собственную* конфигурацию и хотите добавить окно ручной смены инструментов HAL. Ручная смена инструмента HAL в первую очередь полезна, если у вас есть предустановленные инструменты и вы сохраняете смещения в таблице инструментов. Если вам нужно приступить к каждой смене инструмента, лучше всего просто разделить G-код. Чтобы использовать окно HAL Manual Toolchange, вам, по сути, необходимо

1. загрузить компонент hal_manualtoolchange,

2. затем отправить *tool change* iocontrol на hal_manualtoolchange *change* и
3. отправить *changed* hal_manualtoolchange обратно в *tool changed* iocontrol.

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

Это пример ручной смены инструмента *с* помощью компонента 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
```

Это пример ручной смены инструмента *без* компонента 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 Вычислить скорость

В этом примере используются *ddt*, *mult2* и *abs* для вычисления скорости одной оси. Дополнительную информацию о компонентах реального времени см. на страницах руководства или в списке компонентов HAL (Section 5.1.5).

Прежде всего, проверьте свою конфигурацию, чтобы убедиться, что вы не используете готовые компоненты реального времени. Вы можете сделать это, открыв окно конфигурации HAL и найдя компоненты в разделе контактов. Если да, то найдите файл HAL, в который они загружаются, увеличьте количество и настройте экземпляр на правильное значение. Добавьте следующее в файл custom.hal.

Загрузите компоненты реального времени.

```
loadrt ddt count=1
loadrt mult2 count=1
loadrt abs count=1
```

Добавьте функции в поток, чтобы он обновлялся.

```
addf ddt.0 servo-thread
addf mult2.0 servo-thread
addf abs.0 servo-thread
```

Установите связи.

```
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
```

В этом последнем разделе мы устанавливаем для параметра `mult2.0.in1` значение 60, чтобы преобразовать дюймы в секунду в дюймы в минуту (IPM), которые мы получаем из `ddt.0.out`.

`xpos-cmd` отправляет заданную позицию в `ddt.0.in`. ddt вычисляет производную изменения входных данных.

`ddt2.0.out` умножается на 60, чтобы получить IPM.

`mult2.0.out` отправляется в `abs` для получения абсолютного значения.

На следующем рисунке показан результат, когда ось X перемещается со скоростью 15 дюймов в минуту в отрицательном направлении. Обратите внимание, что мы можем получить абсолютное значение либо от контакта `abs.0.out`, либо от сигнала `X-IPM`.

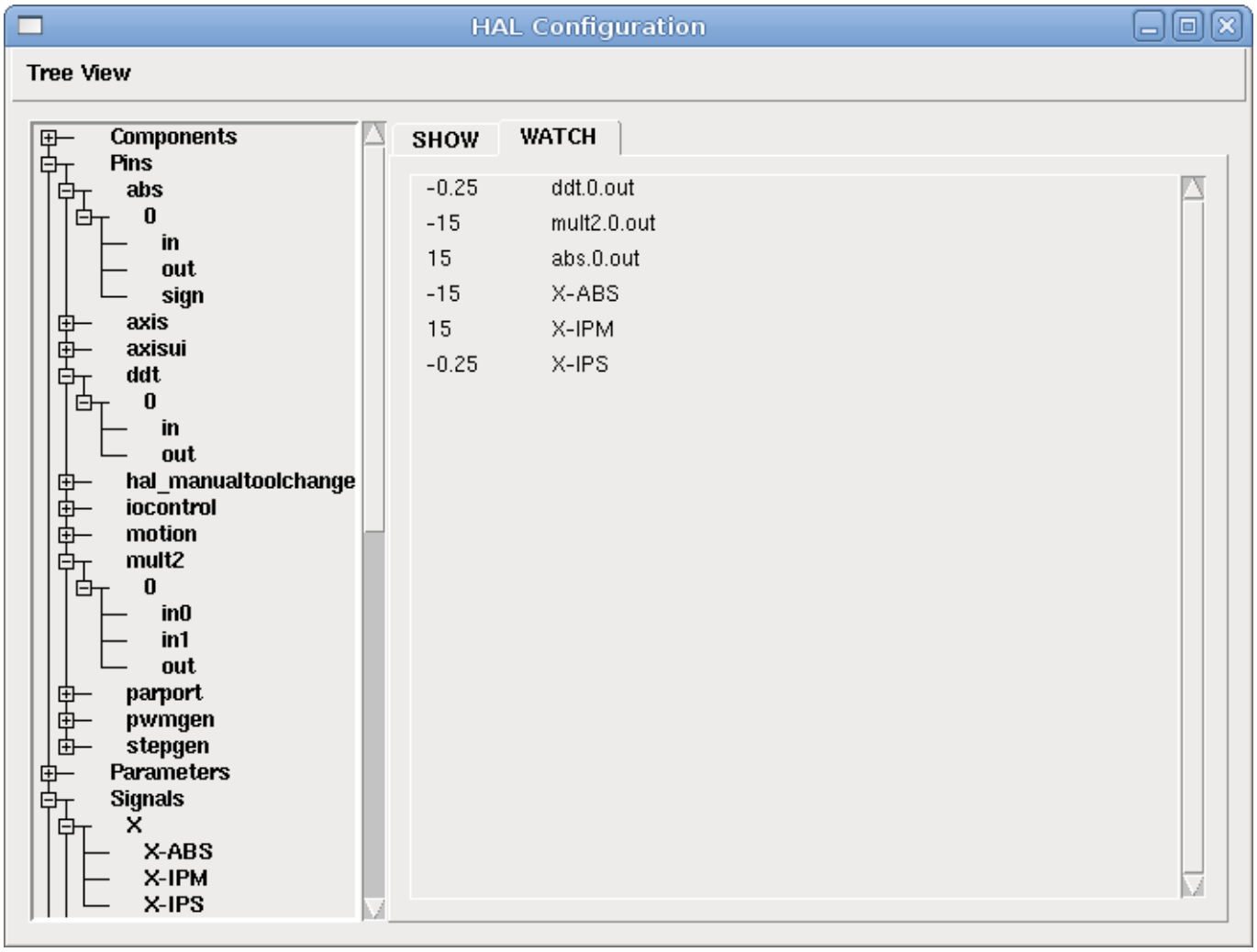



Figure 5.18: HAL: пример скорости

### 5.5.4 Подробности плавного запуска

В этом примере показано, как компоненты HAL *lowpass*, *limit2* или *limit3* могут использоваться для ограничения скорости изменения сигнала.

В этом примере у нас есть серводвигатель, приводящий в движение шпиндель токарного станка. Если мы просто использовали заданные скорости шпинделя сервопривода, он попытается перейти от текущей скорости к заданной скорости как можно быстрее. Это может вызвать проблемы или повредить привод. Чтобы замедлить скорость изменения, мы можем отправить `spindle.N.speed-out` через ограничитель перед PID, чтобы значение команды PID изменялось на новые настройки медленнее.

Три встроенных компонента, ограничивающих сигнал:

- *limit2* ограничивает диапазон и первую производную сигнала.

- *limit3* ограничивает диапазон, первую и вторую производные сигнала.
- moving average to track an input signal.

Чтобы найти дополнительную информацию об этих компонентах HAL, обратитесь к справочным страницам.

Поместите следующее в текстовый файл с именем softstart.hal. Если вы не знакомы с Linux, поместите файл в свой домашний каталог.

```
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
```

Откройте окно терминала и запустите файл с помощью следующей команды.

```
halrun -I softstart.hal
```

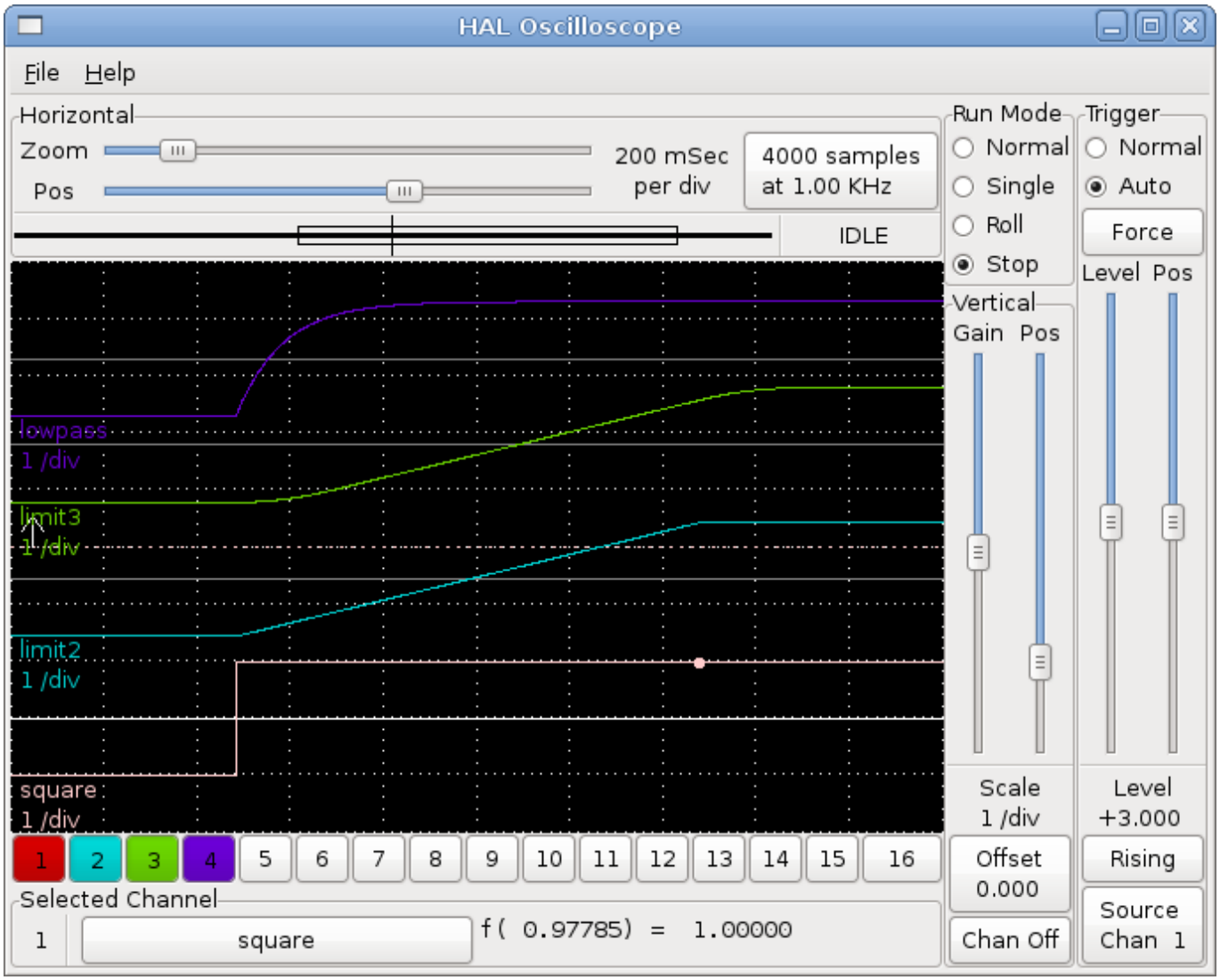
При первом запуске осциллографа HAL нажмите *OK*, чтобы принять поток по умолчанию.

Далее вам нужно добавить сигналы в каналы. Нажмите на channel 1, затем выберите *square* на вкладке Signals. Повторите то же самое для channels 2–4 и добавьте lowpass, limit2 и limit3.

Далее, чтобы настроить триггерный сигнал, нажмите кнопку Source None и выберите square. Кнопка изменится на Source Chan 1.

Затем нажмите Single в поле переключателей Run Mode. Это запустит прогон, и когда он закончится, вы увидите свои сигналы.

Чтобы разделить сигналы и лучше их видеть, щелкните канал, а затем используйте ползунок Pos в поле Vertical, чтобы установить позиции.




Чтобы увидеть эффект от изменения заданных значений любого из компонентов, вы можете изменить их в окне терминала. Чтобы увидеть, что делают различные настройки усиления для нижних частот, просто введите следующее в окне терминала и попробуйте разные настройки.

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

После изменения настройки снова запустите осциллограф, чтобы увидеть изменение.

Когда вы закончите, введите *exit* в окне терминала, чтобы завершить работу halrun и закрыть halscope. Не закрывайте окно терминала при работающем halrun, так как в памяти могут остаться некоторые вещи, которые могут помешать загрузке LinuxCNC.

Для получения дополнительной информации о Halscope см. руководство HAL и учебное пособие.

### 5.5.5 Автономный HAL

В некоторых случаях вам может потребоваться запустить экран GladeVCP только с HAL. Например, предположим, что у вас есть устройство с шаговым приводом, и все, что вам нужно, это запустить шаговый двигатель. Простой интерфейс *Пуск/Стоп* — это все, что вам нужно для вашего приложения, поэтому нет необходимости загружать и настраивать полнофункциональное приложение ЧПУ.

В следующем примере мы создадим простую панель GladeVCP с одним шаговым приводом.

**Основной синтаксис**

```
# load the winder.glade GUI and name it winder
loadusr -Wn winder gladevcp -c winder -u handler.py winder.glade

# load realtime components
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"

# add functions to threads
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

# make HAL connections
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

# b''зb''b''ab''b''пb''b''уb''b''cb''b''кb'' b''пb''b''ob''b''тb''b''ob''b''кb''b''ob''b' ↩
    'вb''
start

# b''зb''b''ab''b''кb''b''ob''b''мb''b''мb''b''eb''b''нb''b''тb''b''иb''b''pb''b''уb''b' ↩
    'йb''b''тb''b''eb'' b''cb''b''лb''b''eb''b''дb''b''уb''b''юb''b''щb''b''иb''b''eb'' b' ↩
    'cb''b''тb''b''pb''b''ob''b''кb''b''иb'' b''вb''b''ob'' b''вb''b''pb''b''eb''b''мb''b' ↩
    'яb'' b''тb''b''eb''b''cb''b''тb''b''иb''b''pb''b''ob''b''вb''b''ab''b''нb''b''иb''b' ↩
    'яb'' b''иb'' b''иb''b''cb''b''пb''b''ob''b''лb''b''ьb''b''зb''b''уb''b''йb''b''тb''b' ↩
    'eb'' b''иb''b''нb''b''тb''b''eb''b''pb''b''ab''b''кb''b''тb''b''иb''b''вb''b''нb''b' ↩
    'уb''b''юb''
# b''ob''b''пb''b''цb''b''иb''b''юb'' halrun -I -f start.hal, b''чb''b''тb''b''ob''b''бb''b ↩
    ''ыb'' b''иb''b''мb''b''eb''b''тb''b''ьb'' b''вb''b''ob''b''зb''b''мb''b''ob''b''жb''b' ↩
    'нb''b''ob''b''cb''b''тb''b''ьb'' b''ob''b''тb''b''ob''b''бb''b''pb''b''ab''b''жb''b' ↩
    'ab''b''тb''b''ьb'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''ыb'' b''иb'' b' ↩
    'тb''. b''дb''..

# b''дb''b''ob''b''жb''b''дb''b''иb''b''тb''b''eb''b''cb''b''ьb'' b''зb''b''ab''b''вb''b' ↩
    'eb''b''pb''b''шb''b''eb''b''нb''b''иb''b''яb'' b''pb''b''ab''b''бb''b''ob''b''тb''b' ↩
    'ыb'' b''Гb''b''Иb''b''Пb'' GladeVCP b''cb'' b''иb''b''мb''b''eb''b''нb''b''eb''b''мb''  ↩
    winder
waitusr winder

# b''ob''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b''пb''b''ob''b' ↩
    'тb''b''ob''b''кb''b''иb'' HAL
stop

# b''вb''b''ыb''b''гb''b''pb''b''уb''b''зb''b''иb''b''тb''b''ьb'' b''вb''b''cb''b''eb'' b' ↩
    'кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ыb'' HAL b''пb''b''eb''b ↩
    ''pb''b''eb''b''дb'' b''вb''b''ыb''b''хb''b''ob''b''дb''b''ob''b''мb''
unloadrt all
```

## 5.6 Основные компоненты

См. также справочные страницы *motion(9)*.

### 5.6.1 Motion

Эти контакты и параметры создаются модулем реального времени *motmod*.

Этот модуль предоставляет интерфейс HAL для планировщика движения LinuxCNC.

По сути, motmod принимает список путевых точек и генерирует хорошо смешанный и ограниченный поток положений сочленений для подачи на приводы двигателей.

Дополнительно количество цифровых входов/выходов устанавливается с помощью *num_dio*. Количеств аналоговых входов/выходов устанавливается с помощью *num_aio*, по умолчанию 4 для каждого. Количество шпинделей задается с помощью *num_spindles*, по умолчанию 1.

Имена контактов и параметров, начинающиеся с *axis.*L и *joint.*N, считываются и обновляются функцией контроллера движения.

Движение загружается командой motmod. Кинематики должны быть загружены перед движением.

```
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* - период потока *Base* в наносекундах. Это самый быстрый поток в станке.

---

**Note**
В системах на основе сервоприводов обычно нет причин для того, чтобы «base_period_nsec» был меньше, чем «servo_period_nsec». На станках с программной генерацией импульсов «base_period_nsec» определяет максимальное количество импульсов в секунду. При отсутствии требований к длине длительности импульса и длительности паузы абсолютная максимальная скорость импульса составляет один импульс за *base_period_nsec*. Таким образом, «base_period_nsec», показанный выше, дает абсолютную максимальную скорость импульсов 20 000 импульсов в секунду. 50 000 нс (50 мкс) — довольно консервативное значение. Наименьшее используемое значение связано с результатом Latency Test, необходимой длиной импульса и скоростью процессора. Выбор слишком малого значения «base_period_nsec» может привести к появлению сообщения «Неожиданная задержка в реальном времени», зависаниям или самопроизвольным перезагрузкам.

---

- *servo_period_nsec = 1000000* — это период задачи *Servo* в наносекундах. Это значение будет округлено до целого числа, кратного base_period_nsec. Этот период используется даже в системах на основе шаговых двигателей.

  Это скорость, с которой вычисляются новые положения двигателя, проверяется ошибка рассогласов обновляются выходные значения ПИД-регулятора и т. д. В большинстве систем нет необходимости изменять это значение. Это частота обновления низкоуровневого планировщика движения.

- *traj_period_nsec = 100000* - это период задачи *Планировщик траектории* в наносекундах. Это значение будет округлено до целого числа, кратного *servo_period_nsec*. За исключением станков с необычной кинематикой (например, гексаподы), нет причин делать это значение больше, чем *servo_period_nsec*.

#### 5.6.1.1 Варианты

Если количество необходимых цифровых входов/выходов превышает установленное по умолчанию 4, вы можете добавить до 64 цифровых входов/выходов, используя опцию *num_dio* при загрузке *motmod*.

Если количество необходимых аналоговых входов/выходов больше, чем 4 по умолчанию, вы можете добавить до 16 аналоговых входов/выходов, используя опцию *num_aio* при загрузке *motmod*.

Параметр unlock_joints_mask используется для создания контактов для сочленения, используемого в качестве индексатора блокировки (обычно поворотного). Биты маски выбирают сочленение(я). Младший бит маски выбирает сочленение 0. Пример:

```
unlock_joints_mask=0x38 b''вb''b''ыb''b''бb''b''иb''b''рb''b''аb''b''еb''b''тb'' b''cb''b' ↩
    'оb''b''чb''b''лb''b''еb''b''нb''b''еb''b''нb''b''иb''b''яb'' 3,4,5
```

#### 5.6.1.2 Контакты

Эти контакты, параметры и функции создаются модулем реального времени *motmod* .

- *motion.adaptive-feed* - (float, in) Когда адаптивная подача включена с помощью *M52 P1*, заданная скорость умножается на это значение. Этот эффект мультипликативен со значением переопределен подачи на уровне NML и *motion.feed-hold*. Начиная с версии 2.9 LinuxCNC, можно использовать отрицательное значение адаптивной подачи для запуска пути G-кода в обратном направлении.
- *motion.analog-in-00* - (float, in) Эти контакты (00, 01, 02, 03 или более, если настроено) управляются M66.
- *motion.analog-out-00* — (float, out) Эти контакты (00, 01, 02, 03 или более, если настроено) управляются M67 или M68.
- *motion.coord-error* - (bit, out) TRUE, когда движение столкнулось с ошибкой, такой как превышение программного предела
- *motion.coord-mode* - (bit, out) TRUE, когда движение в *координированном режиме*, а не в *режиме телеоперации*
- *motion.current-vel* - (float, out) Текущая скорость инструмента в пользовательских единицах в секунду.
- *motion.digital-in-00* - (bit, in) Эти контакты (00, 01, 02, 03 или более, если настроены) управляются M62-65.
- *motion.digital-out-00* - (bit, out) Эти контакты (00, 01, 02, 03 или более, если настроены) управляются *M62-65*.
- *motion.distance-to-go* - (float,out) Расстояние, оставшееся в текущем перемещении.
- *motion.enable* - (bit, in) Если этот бит установлен в FALSE, движение останавливается, станок переходит в состояние *machine off*, и для оператора отображается сообщение. Для нормального движения установите этот бит в значение TRUE.
- *motion.feed-hold* - (bit, in) Когда управление остановкой подачи включено с помощью *M53 P1*, и этот бит в TRUE, скорость подачи устанавливается на 0.
- *motion.feed-inhibit* - (bit, in) Когда этот бит имеет значение TRUE, скорость подачи устанавливается на 0. Это будет отложено во время синхронизации шпинделя до конца перемещения.
- *motion.in-position* - (bit, out) TRUE, если станок находится в положении.
- *motion.motion-enabled* - (bit, out) TRUE, когда в состянии *станок включен*.
- *motion.motion-type* - (s32, out) Эти значения взяты из src/emc/nml_intf/motion_types.h
  - 0: холостой ход (нет движения)
  - 1: Traverse
  - 2: Linear feed

  - 3: Arc feed
  - 4: Tool change
  - 5: Probing
  - 6: Rotary axis indexing

- *motion.on-soft-limit* - (bit, out) TRUE, когда станок находится на программном пределе.
- *motion.probe-input* - (bit, in) *G38.n* использует значение на этом контакте, чтобы определить, когда щуп установил контакт. TRUE для замкнутого контакта щупа (касается), FALSE для разомкнутого контакта щупа.
- *motion.program-line* - (s32, out) Текущая строка программы во время выполнения. Нуль, если не работает или между строками при одиночных шагах.
- *motion.requested-vel* - (float, out) Текущая запрошенная скорость в пользовательских единицах в секунду. Это значение представляет собой настройку F-слова из файла G-кода, возможно, уменьшенную для соответствия пределам скорости и ускорения станка. Значение на этом контакте не отражает переопределение подачи или какие-либо другие настройки.
- *motion.teleop-mode* - (bit, out) TRUE when motion is in *teleop mode*, as opposed to *coordinated mode*
- *motion.tooloffset.x ... motion.tooloffset.w* - (float, out, one per axis) shows the tool offset in effect; it could come from the tool table (*G43* active), or it could come from the G-code (*G43.1* active)
- *motion.on-soft-limit* - (bit, out) TRUE, когда станок находится на программном пределе.
- *motion.probe-input* - (bit, in) *G38.n* uses the value on this pin to determine when the probe has made contact. TRUE for probe contact closed (touching), FALSE for probe contact open.
- *motion.program-line* - (s32, out) Текущая строка программы во время выполнения. Нуль, если не работает или между строками при одиночных шагах.
- *motion.requested-vel* - (float, out) Текущая запрошенная скорость в пользовательских единицах в секунду. Это значение представляет собой настройку F-слова из файла G-кода, возможно, уменьшенную для соответствия пределам скорости и ускорения станка. Значение на этом контакте не отражает переопределение подачи или какие-либо другие настройки.
- *motion.teleop-mode* - (bit, out) TRUE when motion is in *teleop mode*, as opposed to *coordinated mode*
- *motion.tooloffset.x ... motion.tooloffset.w* - (float, out, one per axis) shows the tool offset in effect; it could come from the tool table (*G43* active), or it could come from the G-code (*G43.1* active)

#### 5.6.1.3 Параметры

Многие из этих параметров служат в качестве вспомогательных средств отладки и подлежат изменению или удалению в любое время.

- *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) Это используется в целях отладки.
- *motion.debug-bit-1* - (bit, RO) Это используется в целях отладки.

- *motion.debug-float-0* - (float, RO) Это используется в целях отладки.
- *motion.debug-float-1* - (float, RO) Это используется в целях отладки.
- *motion.debug-float-2* - (float, RO) Это используется в целях отладки.
- *motion.debug-float-3* - (float, RO) Это используется в целях отладки.
- *motion.debug-s32-0* - (s32, RO) Это используется в целях отладки.
- *motion.debug-s32-1* - (s32, RO) Это используется в целях отладки.
- *motion.servo.last-period* - (u32, RO) Количество циклов ЦП между вызовами сервопотока. Обычно это число, разделенное на скорость процессора, дает время в секундах и может использоваться для определения того, соответствует ли контроллер движений в реальном времени своим временным ограничениям
- *motion.servo.last-period-ns* - (float, RO)

#### 5.6.1.4 Функции

Обычно обе эти функции добавляются в сервопоток в указанном порядке.

- *motion-command-handler* - Получает и обрабатывает команды движения
- *motion-controller* - Запускает контроллер движения LinuxCNC

### 5.6.2 Spindle

LinuxCNC может осуществлять управление до восьми шпинделей. Модуль движения создаст следующие контакты: *N* (целое число от 0 до 7) заменяет номер шпинделя.

#### 5.6.2.1 Контакты

* *spindle*.N.*at-speed* - (bit, in) Движение будет приостановлено до тех пор, пока этот контакт не станет TRUE, при следующих условиях: **перед первым ходом подачи после каждого запуска шпинделя или изменения скорости;** перед началом каждой цепочки синхронизированны со шпинделем ходов; ** и если в режиме CSS, то при каждом быстром переходе на подачу. Этот ввод можно использовать, чтобы убедиться, что шпиндель набрал нужную скорость перед началом резки или что шпиндель токарного станка в режиме CSS замедлился после прохода снятия от большого к малому перед началом следующего прохода на большем диаметре. Многие VFD имеют выход *at speed*. В противном случае этот сигнал легко сгенерировать с помощью компонента *HAL near*, сравнивая запрашиваемую и фактическую скорость шпинделя. * *spindle*.N.*brake* - (bit, out) TRUE, когда следует задействовать тормоз шпинделя. * *spindle*.N.*forward* - (bit, out) TRUE, когда шпиндель должен вращаться вперед. * *spindle*.N.*index-enable* - (bit, I/O) Для правильной работы синхронизированных перемещений шпинделя этот контакт должен быть подключен к контакту index-enable энкодера шпинделя. * *spindle*.N.*inhibit* - (bit, in) Когда этот бит равен TRUE, скорость шпинделя установлена на 0. * *spindle*.N.*on* - (bit, out) TRUE, когда шпиндель должен вращаться. * *spindle*.N.*reverse* - (bit, out) TRUE, когда шпиндель должен вращаться назад * *spindle*.N.*revs* - (float, in) Для правильной работы синхронизированных перемещений шпинделя этот сигнал должен быть подключен к контакту положения энкодера шпинделя. Положение энкодера шпинделя должно быть масштабировано таким образом, чтобы число оборотов шпинделя увеличивалось на 1,0 при каждом обороте шпинделя по часовой стрелке (*M3*). * *spindle*.N.*speed-in* - (float, in) Обратная связь о фактической скорости шпинделя в оборотах в секунду. Это используется при движении с подачей на оборот (*G95*). Если ваш драйвер энкодера шпинделя не имеет выхода скорости, вы можете сгенерировать подходящий выход, отправив положение

шпинделя через компонент *ddt* . Если у вас нет энкодера шпинделя, вы можете сделать обратную связь с *spindle*.N.*speed-out-rps*. * *spindle*.N.*speed-out* - (float, out) Заданная скорость шпинделя в оборотах в минуту. Положительный для вращения шпинделя вперед (*M3*), отрицательный для вращения шпинделя назад (*M4*). * *spindle*.N.*speed-out-abs* - (float, out) Заданная скорость шпинделя в оборотах в минуту. Это всегда будет положительное число. * *spindle*.N.*speed-out-rps* - (float, out) Заданная скорость шпинделя в оборотах в секунду. Положительная для вращения шпинделя вперед (*M3*), отрицательная для вращения шпинделя назад (*M4*). * *spindle*.N.*speed-out-rps-abs* - (float, out) Заданная скорость шпинделя в оборотах в секунду. Это всегда будет положительное число. * *spindle*.N.*orient-angle* - (float,out) Желаемая ориентация шпинделя для M19. Значение параметра M19 R word плюс значение параметра [RS274NGC]ORIENT_OFFSET INI. * *spindle*.N.*orient-mode* - (s32,out) Желаемый режим вращения шпинделя M19. По умолчанию 0. * *spindle*.N.*orient* - (out,bit) Указывает на начало цикла ориентации шпинделя. Устанавливается M19. Очищается любым из M3, M4 или M5. Если spindle-orient-fault не равна нулю при spindle-orient true, команда M19 завершается сбоем с сообщением об ошибке. * *spindle*.N.*is-oriented* - (in, bit) Контакт подтверждения для spindle-orient. Завершает цикл ориентации. Если spindle-orient было true, когда стал активным spindle-is-oriented, контакт spindle-orient сбрасывается и устанавливается в активный уровень контакт spindle-locked. Также устанавливается контакт spindle-brake. * *spindle*.N.*orient-fault* - (s32, in) Ввод кода неисправности для цикла ориентации. Любое значение, отличное от нуля, приведет к прерыванию цикла ориентации. * *spindle*.N.*lock* - (bit, out) Контакт ориентация шпинделя выполнена. Очищается любым из M3, M4 или M5.

**Использование контакта HAL для ориентации шпинделя M19** Концептуально шпиндель находится в одном из следующих режимов:

- режим вращения (по умолчанию)
- поиск нужного режима ориентации
- режим завершения ориентации.

Когда выполняется M19, шпиндель переключается на *поиск желаемой ориентации*, и подключается контакт HAL `spindle.__N__.orient`. Желаемая целевая позиция задается контактами `spindle.__N__.` и `spindle.__N__.orient-fwd` и управляется параметрами M19 R и P.

Ожидается, что логика поддержки HAL будет реагировать на `spindle.__N__.orient`, перемещая шпиндель в желаемое положение. Когда это будет завершено, ожидается, что логика HAL подтвердит это, установив активным вывод `spindle.__N__.is-ориентированный`.

Затем Motion подтверждает это, делая неактивным контакт `spindle.__N__.orient` и делая активным контакт `spindle.__N__.locked`, чтобы показать режим *orientation complete*. Он также делает активным контакт `spindle.__N__.brake`. Шпиндель теперь находится в режиме *orientation complete*.

Если, пока `spindle.__N__.orient` является true, а `spindle.__N__.is-oriented` еще не установлен активным, контакт `spindle.__N__.orient-fault` имеет значение, отличное от нуля, команда M19 прерывается, отображается сообщение, включающее код неисправности, и очередь движений очищается. Шпиндель возвращается в режим вращения.

Кроме того, любая из команд M3, M4 или M5 отменяет режим *поиска нужной ориентации* или *ориентация завершена*. На это указывает отключение контактов `spindle-orient` и `spindle-locked`.

Контакт `spindle-orient-mode` отражает слово M19 P и интерпретируется следующим образом:

- 0: вращение по часовой стрелке или против часовой стрелки для наименьшего углового перемещени
- 1: всегда вращать по часовой стрелке
- 2: всегда вращать против часовой стрелки

Его можно использовать с компонентом HAL `orient`, который предоставляет значение команды PID на основе положения энкодера шпинделя, угла ориентации шпинделя и режима ориентации шпинделя.

### 5.6.3 Контакты осей и сочленений и параметры

Эти контакты и параметры создаются модулем реального времени motmod. [В станках с *trivial kinematics* между сочленениями и осями существует соответствие один-к-одному.] Они считываются и обновляются функцией *motion-controller*.

Подробную информацию о контактах и параметрах см. на странице руководства по движению «motion(9)».

### 5.6.4 iocontrol

iocontrol - принимает команды ввода-вывода не в реальном времени через NML, взаимодействует с HAL.

Контакты HAL iocontrol включаются и выключаются не в режиме реального времени. Если у вас строгие требования к синхронизации или вам просто нужно больше операций ввода-вывода, рассмотрите возможность использования вместо этого синхронизированного ввода-вывода в реальном времени, предоставляемого motion.

#### 5.6.4.1 Контакты

* *iocontrol.0.coolant-flood* (bit, out) TRUE когда запрашивается охлаждающая жидкость. * *iocontrol.0.coo mist* (bit, out) TRUE когда запрашивается охлаждающая жидкость в виде тумана. * *iocontrol.0.emc-enable-in* (bit, in) Должно быть установлено значение FALSE, когда существует внешнее условие аварийного останова. * *iocontrol.0.tool-change* (bit, out) TRUE когда требуется смена инструмента. * *iocontrol.0.tool-changed* (bit, in) Должен быть установлен в TRUE, когда смена инструмента завершена. * *iocontrol.0.tool-number* (s32, out) Текущий номер инструмента. * *iocontrol.0.tool-prep-number* (s32, out) Номер следующего инструмента из T-слова RS274NGC. * *iocontrol.0.tool-prepare* (bit, out) TRUE, когда запрашивается подготовка инструмента. * *iocontrol.0.tool-prepared* (bit, in) Должен быть установлен в TRUE, когда подготовка инструмента завершена. * *iocontrol.0.user-enable-out* (bit, out) FALSE , когда существует внутреннее условие аварийного останова. * *iocontrol.0.use request-enable* (bit, out) TRUE, когда пользователь запросил очистку аварийного останова.

### 5.6.5 настройки INI

Ряд настроек INI доступен как входные контакты HAL.

#### 5.6.5.1 Контакты

*N* обозначает номер сочленения, *L* обозначает букву оси.

- *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

---

**Note**
Контакты min_limit и max_limit для каждой оси учитываются постоянно после возврата в исходное положение. Выводы ferror и min_ferror по каждой оси учитываются, когда станок включен и не находится в нужном положении. Контакты max_velocity и max_acceleration для каждой оси сэмплируются, когда станок включен и motion_state свободен (возврат в исходное положение или медленная подача), но не выбираются во время работы программы (автоматический режим) или в режиме MDI. Следовательно, изменение значений контактов во время работы программы не будет иметь эффекта до тех пор, пока программа не будет остановлена и motion_state снова не станет свободным.

---

- *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

---

**Note**
Выборка контактов traj_arc_blend происходит непрерывно, но изменение значений контактов во время работы программы может не иметь немедленного эффекта из-за очереди команд.

---

- *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

S-curve trajectory planning pins (sampled continuously, can be changed at runtime):

- *ini.traj_planner_type* - (s32, in) [TRAJ]PLANNER_TYPE
- *ini.traj_max_jerk* - (float, in) [TRAJ]MAX_LINEAR_JERK

Per-axis jerk limit pins (where *L* is x, y, z, a, b, c, u, v, or w):

- *ini*.L.*max_jerk* - (float, in) [AXIS__L_]MAX_JERK

Per-joint jerk limit pins (where *N* is the joint number 0-8):

- *ini*.N.*max_jerk* - (float, in) [JOINT__N_]MAX_JERK

# 5.7 HAL Список компонентов

## 5.7.1 Компонентов

Большинство команд в следующем списке имеют свои собственные справочные страницы. У некоторых будут расширенные описания, у некоторых — ограниченные. Из этого списка вы узнаете, какие компоненты существуют, и можете использовать `man` *name* в командной строке UNIX для получения дополнительной информации. Чтобы просмотреть информацию на странице руководства, в окне терминала введите:

```
man axis
```

Та или иная установка системы UNIX может потребовать явного указания раздела man-страницы. Если вы не нашли справочную страницу или имя справочной страницы уже занято другим инструмент UNIX, а справочная страница LinuxCNC находится в другом разделе, попробуйте явно указать раздел, как в `man _section-no_ axis`, с помощью *section-no* = 1 для компонентов не в реальном времени и 9 для компонентов реального времени.

**Note**
См. также раздел «Страницы руководства» по docs main page или directory listing. Для поиска по страницам руководства используйте инструмент UNIX `apropos`.

### 5.7.1.1 Пользовательские интерфейсы (не в реальном времени)

| | |
|---|---|
| **axis** | ГИП AXIS LinuxCNC (расширенный контроллер станка) |
| **axis-remote** | Удаленный интерфейс AXIS |
| **gmoccapy** | Touchy LinuxCNC ГИП |
| **gscreen** | Touchy LinuxCNC ГИП |
| **halui** | Наблюдайте за контактами HAL и управляйте LinuxCNC через NML |
| **mdro** | только ручной Универсальный Цифровой Индикатор (DRO) |
| **ngcgui** | Платформа для генерации диалогового G-кода на контроллере |
| **panelui** | |
| **pyngcgui** | Реализация NGCGUI на Python |
| **touchy** | AXIS — TOUCHY ГИП LinuxCNC |

| | |
|---|---|
| **gladevcp** | Виртуальная панель управления для LinuxCNC на базе виджетов Glade, Gtk и HAL |
| **gladevcp_demo** | GladeVCP — используется в примерах конфигураций для демонстрации Glade Virtual_demo |
| **gremlin_view** | Графический предварительный просмотр G-кода |
| **moveoff_gui** | ГИП для компонента moveoff |
| **pyui** | Утилита для panelui |
| **pyvcp** | Виртуальная панель управления для LinuxCNC |
| **pyvcp_demo** | Демонстрационный компонент виртуальной панели управления Python |
| **qtvcp** | Виртуальная панель управления на базе Qt |

| | |
|---|---|
| **5axisgui** | Vismach ГИП Виртуального станка |
| **hbmgui** | Vismach ГИП Виртуального станка |
| **hexagui** | Vismach ГИП Виртуального станка |
| **lineardelta** | Vismach ГИП Виртуального станка |
| **maho600gui** | hexagui - Vismach ГИП Виртуального станка |

| | |
|---|---|
| **max5gui** | hexagui - Vismach ГИП Виртуального станка |
| **melfagui** | Vismach ГИП Виртуального станка |
| **puma560gui** | puma560agui - Vismach ГИП Виртуального станка |
| **pumagui** | Vismach ГИП Виртуального станка |
| **rotarydelta** | Vismach ГИП Виртуального станка |
| **scaragui** | Vismach ГИП Виртуального станка |
| **xyzac-trt-gui** | Vismach ГИП Виртуального станка |
| **xyzbc-trt-gui** | Vismach ГИП Виртуального станка |
| **xyzab-tdr-gui** | Vismach ГИП Виртуального станка |

#### 5.7.1.2 Движение (не в реальном времени)

| | |
|---|---|
| **io** | iocontrol — взаимодействует с HAL или G-кодом не в реальном времени |
| **iocontrol** | Взаимодействует с HAL или G-кодом не в реальном времени |
| **mdi** | Отправляйте команды G-кода с терминала на работающий экземпляр LinuxCNC |
| **milltask** | Контроллер задач не в реальном времени для LinuxCNC |

#### 5.7.1.3 Драйверы оборудования

| | |
|---|---|
| **elbpcom** | Коммуникация с Mesa Ethernet картами |
| **gs2_vfd** | Компонент HAL, работающий не в реальном времени, для VFD Automation Direct GS2 |
| **hy_gt_vfd** | Компонент HAL, работающий не в реальном времени, для частотно-регулируемых приводов Huanyang серии GT |
| **hy_vfd** | Компонент HAL не в реальном времени для VFD Huanyang |
| **mb2hal** | MB2HAL — это универсальный компонент HAL, работающий не в реальном времени, для связи с одним или несколькими устройствами Modbus. Поддерживаются Modbus RTU и Modbus TCP. |
| **mitsub_vfd** | Компонент HAL не в реальном времени для VFD Mitsubishi A500 F500 E500 A500 D700 E700 F700 (другие также могут работать) |
| **monitor-xhc-hb04** | Контролирует пульт XHC-HB04 и предупреждает об отключении |
| **pi500_vfd** | Powtran PI500 modbus дравейр |
| **pmx485** | Связь Modbus с плазменным резаком Powermax |
| **pmx485-test** | Тестирование связи Modbus с помощью плазменного резака Powermax |
| **shuttle** | управление контактами HAL с помощью устройств ShuttleXpress, ShuttlePRO и ShuttlePRO2, сделанных Contour Design |
| **svd-ps_vfd** | Компонент HAL, работающий не в режиме реального времени, для частотно-регулируемых приводов SVD-P(S) |
| **vfdb_vfd** | Компонент HAL, работающий не в режиме реального времени, для преобразователей частоты Delta VFD-B |
| **vfs11_vfd** | Компонент HAL не в реальном времени для преобразователей частоты Toshiba-Schneider VF-S11 |
| **wj200_vfd** | Hitachi wj200 Modbus driver |
| **xhc-hb04** | Компонент HAL не в реальном времени для пульта xhc-hb04 |
| **xhc-hb04-accels** | Устаревший скрипт для маховичка медленной подачи |
| **xhc-whb04b-6** | Компонент HAL не в реальном времени для беспроводного пульта USB-устройства XHC WHB04B-6 |

#### 5.7.1.4 Mesa и другие карты ввода-вывода (в реальном времени)

| | |
|---|---|
| **hal_ppmc** | Pico Systems driver для аналогового сервопривода, ШИМ и шагового контроллера |
| **hal_bb_gpio** | Драйвер для контактов BeagleBone GPIO |
| **hal_parport** | Компонент HAL реального времени для связи с одним или несколькими параллельными портами ПК |
| **hm2_7i43** | Драйвер Mesa Electronics для платы Anything IO 7I43 EPP с HostMot2. (Для получения дополнительной информации см. справочную страницу) |
| **hm2_7i90** | Драйвер LinuxCNC HAL для платы Mesa Electronics 7I90 EPP Anything IO с прошивкой HostMot2 |
| **hm2_eth** | LinuxCNC HAL драйвер для плат Mesa Electronics Ethernet Anything IO с прошивкой HostMot2 |
| **hm2_pci** | Mesa Electronics драйвер для плат 5I20, 5I22, 5I23, 4I65 и 4I68 Anything I/O, с прошивкой HostMot2. (Смотрите страницу man для дополнительной информации) |
| **hm2_rpspi** | LinuxCNC HAL драйвер для плат Mesa Electronics SPI Anything IO с прошивкой HostMot2 |
| **hm2_spi** | LinuxCNC HAL драйвер для плат Mesa Electronics SPI Anything IO с прошивкой HostMot2 |
| **hostmot2** | Mesa Electronics driver для прошивки HostMot2. |
| **max31855** | Поддержка преобразователя термопары в цифру MAX31855 с использованием побитового SPI |
| **mesa_7i65** | Драйвер Mesa Electronics для восьмиосной сервокарты 7I65. (Для получения дополнительной информации см. справочную страницу) |
| **mesa_pktgyro_test** | Простой тест PktUART с гироскопом Microstrain 3DM-GX3-15 |
| **mesa_uart** | An example component demonstrating how to access the Hostmot2 UART |
| **opto_ac5** | Драйвер реального времени для карт opto22 PCI-AC5 |
| **pluto_servo** | Pluto-P driver и прошивка для параллельного порта ПЛИС, для сервоприводов |
| **pluto_step** | Pluto-P driver для параллельного порта FPGA, для шаговых двигателей |
| **serport** | Аппаратный драйвер для битов цифрового ввода-вывода последовательного порта 8250 и 16550 |
| **setsserial** | An utility for setting Smart Serial NVRAM parameters |
| **sserial** | hostmot2 — драйвер Smart Serial LinuxCNC HAL для удаленных карт Mesa Electronics HostMot2 Smart-Serial |

#### 5.7.1.5 Утилиты (не в реальном времени)

| | |
|---|---|
| **hal-histogram** | Отображает значение контакта HAL в виде гистограммы |
| **halcompile** | Сборка, компиляция и установка компонентов LinuxCNC HAL |
| **halmeter** | Наблюдайте за контактами, сигналами и параметрами HAL |
| **halcmd** | Управляйте LinuxCNC HAL из командной строки |
| **halcmd_twopass** | Utility script used when parsing HAL files. It allows to have multiple load-commands for multiple instances of the same component. |
| **halreport** | Создает отчет о состоянии HAL |
| **halrmt** | Интерфейс удаленного управления для LinuxCNC |
| **halrun** | Управляйте LinuxCNC HAL из командной строки |
| **halsampler** | Примеры данных из HAL в реальном времени |
| **halscope** | Программный осциллограф для просмотра форм сигналов и контактов HAL в реальном времени |
| **halshow** | Показать параметры, контакты и сигналы HAL |

| | |
|---|---|
| **halstreamer** | Потоковая передача данных файла в HAL в режиме реального времени |
| **haltcl** | Управляет LinuxCNC HAL из командной строки с помощью Tcl |
| **image-to-gcode** | Преобразует растровые изображения в G-код |
| **inivar** | Запросить INI-файле |
| **latency-histogram** | Построение гистограммы задержки станка |
| **latency-plot** | Другой способ просмотреть цифры задержки |
| **latency-test** | Тестирует задержку системы в реальном времени |
| **linuxcncmkdesktop** | Создайте значок LinuxCNC на рабочем столе |
| **modcompile** | Utility for compiling Modbus drivers |
| **motion-logger** | Log motion commands sent from LinuxCNC |
| **pncconf** | Мастер настройки карт Mesa |
| **sim_pin** | ГИП для отображения и настройки одного или нескольких входов HAL |
| **stepconf** | Мастер настройки для станков с интерфейсом через параллельный порт |
| **update_ini** | Конвертирует INI-файлы формата 2.7 в формат 2.8 |
| **debuglevel** | Sets the debug level for the non-realtime part of LinuxCNC |
| **emccalib** | Adjust ini tuning variables on the fly with save option |
| **hal_input** | Control HAL pins with any Linux input device, including USB HID devices |
| **linuxcnc_info** | Collects information about the LinuxCNC version and the host |
| **linuxcnc_module_helper** | Controls root access for system hardware |
| **linuxcnc_var** | Retrieves LinuxCNC variables |
| **linuxcnc** | LinuxCNC (Улучшенный контроллер станка) |
| **linuxcnclcd** | Графический пользовательский интерфейс LinuxCNC для ЖК-дисплея |
| **linuxcncrsh** | Text-mode interface for commanding LinuxCNC over the network |
| **linuxcncsvr** | Allows network access to LinuxCNC internals via NML |
| **linuxcnctop** | Описание статуса Live LinuxCNC |
| **rs274** | Автономный интерпретатор G-кода |
| **schedrmt** | Планировщик на базе Telnet для LinuxCNC |
| **setup_designer** | A script to configure the system for use of Qt Designer |
| **teach-in** | Jog the machine to a position, and record the state |
| **tool_mmap_read** | A component of the tool database system (an alternative to the classic tooltable) |
| **tool_watch** | A component of the tool database system (an alternative to the classic tooltable) |
| **tooledit** | Редактор таблиц инструментов |

#### 5.7.1.6 Обработка сигналов (в реальном времени)

| | |
|---|---|
| **and2** | Двухвходовой вентиль И. Чтобы вывод был истинным, оба входных параметра должны быть истинными. (and2) |
| **bitwise** | Вычисляет различные побитовые операции над двумя входными значениями |
| **dbounce** | Ссылка на фильтр зашумленных цифровых Details |
| **debounce** | Фильтрация шумных цифровых входов Details description |
| **demux** | Выберите один из нескольких выходных контактов целым числом и/или отдельными битами |
| **edge** | Детектор импульса |
| **estop_latch** | Защелка аварийного останова |
| **flipflop** | триггер типа D |
| **logic** | Компонент общей логической функции |
| **lut5** | Логическая функция с 5 входами на основе справочной таблицы description |

| | |
|---|---|
| **match8** | 8-битный детектор двоичных совпадений |
| **multiclick** | Детектор одиночного, двойного, тройного и четверного клика |
| **multiswitch** | Переключает между указанным количеством выходных битов |
| **not** | Инвертер |
| **oneshot** | Одновибратор |
| **or2** | Двухвходовой вентиль ИЛИ |
| **reset** | Сбрасывает сигнал ввода-вывода |
| **select8** | 8-битный двоичный детектор совпадений. |
| **tof** | Таймер IEC TOF — задержка среза сигнала |
| **toggle** | Включение нажатием и выключение отпусканием с помощью кнопок мгновенного действия |
| **toggle2nist** | Кнопка переключения на nist логику |
| **ton** | Таймер IEC TON - задержка фронта сигнала |
| **timedelay** | Эквивалент реле задержки времени. |
| **tp** | Таймер IEC TP — генерирует импульс высокого уровня определенной длительности по фронту |
| **tristate_bit** | Подает сигнал на контакт ввода-вывода только когда разрешено, подобно буферу с тремя состояниями в электронике |
| **tristate_float** | Подает сигнал на контакт ввода-вывода только когда разрешено, подобно буферу с тремя состояниями в электронике |
| **xor2** | Двухвходовой вентиль XOR (исключающее ИЛИ) |

| | |
|---|---|
| **abs_s32** | Вычисляет абсолютное значение и знак целочисленного входного сигнала |
| **abs_s64** | Вычисляет абсолютное значение и знак 64-битного целочисленного входного сигнала. |
| **abs** | Вычисляет абсолютное значение и знак входного сигнала с плавающей точкой |
| **biquad** | Биквадратный БИХ-фильтр |
| **blend** | Выполняет линейную интерполяцию между двумя значениями |
| **comp** | Двухвходовой компаратор с гистерезисом |
| **counter** | Подсчитывает входные импульсы (устарело). Используйте компонент encoder. |
| **ddt** | Вычисляет производную входной функции. |
| **deadzone** | Возвращает центр, если находится в пределах порога. |
| **demux_generic** | Routes a single input signal to one of multiple outputs. |
| **div2** | Quotient of two floating point inputs. |
| **hypot** | Калькулятор гипотенузы (евклидова расстояния) с тремя входами. |
| **ilowpass** | Фильтр нижних частот с целочисленными входами и выходами |
| **integ** | Интегратор |
| **invert** | Вычисляет обратную величину входного сигнала. |
| **filter_kalman** | Одномерный фильтр Калмана, также известный как линейно-квадратическая оценка (LQE) |
| **knob2float** | Преобразует значения (вероятно, полученные от энкодера) в число с плавающей запятой. |
| **led_dim** | Компонент HAL для диммирования светодиодов |
| **lowpass** | Фильтр нижних частот |
| **limit1** | Ограничивает выходной сигнал между минимальным и максимальным значениями. footnote: [Когда входные данные представляют собой позицию, это означает, что *позиция* ограничена.] |
| **limit2** | Ограничивает выходной сигнал между минимальным и максимальным значениями. Ограничьте скорость нарастания ниже maxv в секунду. footnote: [Когда входными данными является положение, это означает, что *положение* и *скорость* ограничены.] |

| | |
|---|---|
| **limit3** | Ограничить выходной сигнал между минимальным и максимальным значениями. Ограничить скорость нарастания ниже maxv в секунду. Ограничить его вторую производную значением менее MaxA на секунду в квадрате, footnote: [Когда входные данные — это позиция, это означает, что *положение*, *скорость* и *ускорение* ограничены.]. |
| **lincurve** | Одномерная справочная таблица |
| **maj3** | Вычислить наибольшее из 3 входов |
| **minmax** | Отслеживает минимальные и максимальные значения входа и выхода. |
| **mult2** | Продукт двух входов. |
| **mux16** | Выбирает одно из 16 входных значений (мультиплексор). |
| **mux2** | Выбирает одно из двух входных значений (мультиплексор). |
| **mux4** | Выбирает одно из четырех входных значений (мультиплексор). |
| **mux8** | Выбирает одно из восьми входных значений (мультиплексор). |
| **mux_generic** | Выбирает одно из нескольких входных значений (мультиплексор). |
| **near** | Определяет, равны ли примерно два значения. |
| **offset** | Добавляет смещение к входным данным и вычитает его из значения обратной связи. |
| **safety_latch** | latch for error signals |
| **sample_hold** | Выборка и хранение. |
| **scaled_s32_sums** | Сумма четырех входов (каждый с масштабированием) |
| **scale** | Применяет масштаб и смещение к входным данным. |
| **sum2** | Сумма двух входов (каждый с усилением) и смещения. |
| **time** | Накопленный таймер времени выполнения отсчитывает ЧЧ:ММ:СС входа *active*. |
| **timedelta** | Компонент, который измеряет поведение времени планирования потоков. |
| **updown** | Считает в большую или меньшую сторону, с дополнительными ограничениями и циклическим поведением. |
| **wcomp** | Оконный компаратор. |
| **watchdog** | Контролируйте от одного до тридцати двух входов на предмет *heartbeat* (пульса). |
| **weighted_sum** | Преобразуйте группу битов в целое число. |
| **xhc_hb04_util** | Удобная утилита xhc-hb04 |

#### 5.7.1.7 Генерация сигнала (в реальном времени)

| | |
|---|---|
| **charge_pump** | Creates a square-wave for the *charge pump* input of some controller boards. |
| **pwmgen** | Программная генерация PWM/PDM, см. description. |
| **siggen** | Генератор сигналов, см. description. |
| **sim_encoder** | Имитированный квадратурный энкодер, см. description. |
| **stepgen** | Программная генерация шаговых импульсов, см. description. |

| | |
|---|---|
| **bin2gray** | Преобразует число в представление кода Грея |
| **bitmerge** | Converts individual input bits into an unsigned-32 |
| **bitslice** | Преобразует входной сигнал без знака-32 в отдельные биты |
| **conv_bit_float** | Преобразует бит в число с плавающей запятой |
| **conv_bit_s32** | Конвертирует из бита в s32 |
| **conv_bit_u32** | Конвертирует из бита в u32 |
| **conv_float_s32** | Конвертирует из float в s32 |
| **conv_float_u32** | Преобразует из float в u32 |
| **conv_s32_bit** | Конвертирует из s32 в бит |
| **conv_s32_float** | Конвертирует из s32 в float |
| **conv_s32_u32** | Конвертирует из s32 в u32 |

| | |
|---|---|
| **conv_u32_bit** | Конвертирует из u32 в бит |
| **conv_u32_float** | Преобразует из u32 в float |
| **conv_u32_s32** | Конвертирует из u32 в s32 |
| **conv_bit_s64** | Конвертирует из bit в s64 |
| **conv_bit_u64** | Преобразовать значение из bit в u64 |
| **conv_float_s64** | Преобразовать значение из float в s64 |
| **conv_float_u64** | Преобразовать значение из float в u64 |
| **conv_s32_s64** | Преобразовать значение из s32 в s64 |
| **conv_s32_u64** | Преобразовать значение из s32 в u64 |
| **conv_s64_bit** | Преобразовать значение из s64 в бит |
| **conv_s64_float** | Преобразовать значение из s64 в float |
| **conv_s64_s32** | Преобразовать значение из s64 в s32 |
| **conv_s64_u32** | Преобразовать значение из s64 в u32 |
| **conv_s64_u64** | Преобразовать значение из s64 в u64 |
| **conv_u32_s64** | Преобразовать значение из u32 в s64 |
| **conv_u32_u64** | Преобразовать значение из u32 в u64 |
| **conv_u64_bit** | Преобразовать значение из u64 в бит |
| **conv_u64_float** | Преобразовать значение из u64 в float |
| **conv_u64_s32** | Преобразовать значение из u64 в s32 |
| **conv_u64_s64** | Преобразовать значение из u64 в s64 |
| **conv_u64_u32** | Преобразовать значение из u64 в u32 |
| **gray2bin** | Преобразует ввод кода Грея в двоичный |

#### 5.7.1.8 Кинематика (в реальном времени)

| | |
|---|---|
| **corexy_by_hal** | Кинематика CoreXY |
| **differential** | Кинематика для дифференциальной передачи |
| **gantry** | Компонент LinuxCNC HAL для управления несколькими сочленениями с одной оси |
| **gantrykins** | Модуль кинематики, который отображает одну ось на несколько сочленений. |
| **genhexkins** | Дает шесть степеней свободы в положении и ориентации (XYZABC). Расположение двигателей определяется во время компиляции. |
| **genserkins** | Кинематика, позволяющая моделировать обычный манипулятор с последовательным соединением, имеющий до 6 угловых сочленений. |
| **gentrivkins** | Соответствие 1:1 между сочленениями и осями. Большинство стандартных фрезерных и токарных станков используют тривиальный модуль кинематики. |
| **kins** | Определения кинематики для LinuxCNC. |
| **lineardeltakins** | Кинематика линейного дельта-робота |
| **matrixkins** | Калиброванная кинематика для 3-осевого станкаs |
| **maxkins** | Кинематика настольного 5-осевого фрезерного станка названного *max* с наклонной головкой (ось B) и горизонтально-поворотным узлом, смонтированным на стол (ось C). Обеспечивает движение UVW во вращающейся системе координат. |
| **millturn** | Переключаемая кинематика токарно-фрезерного станка |
| **pentakins** | |
| **pumakins** | Кинематика роботов типа PUMA. |
| **rosekins** | Кинематика двигателя rose |
| **rotatekins** | Оси X и Y повернуты на 45 градусов по сравнению с сочленениями 0 и 1. |
| **scarakins** | Кинематика роботов типа SCARA. |
| **tripodkins** | Сочленения представляют собой расстояние от контролируемой точки до трех заранее заданных мест (двигателей), что дает три степени свободы положения (XYZ). |

| | |
|---|---|
| **userkins** | Шаблон для пользовательской кинематики |
| **xyzab_tdr_kins** | Переключаемая кинематика для 5-осевого станка с поворотным столом A и B |
| **xyzacb_trsrn** | Switchable kinematics for 6 axis machine with a rotary table C, rotary spindle B and nutating spindle A |
| **xyzbca_trsrn** | Switchable kinematics for 6 axis machine with a rotary table B, rotary spindle C and nutating spindle A |

#### 5.7.1.9 Управление движением (в реальном времени)

| | |
|---|---|
| **feedcomp** | Умножает входное значение на отношение текущей скорости к скорости подачи. |
| **homecomp** | Шаблон модуля исходной позиции |
| **limit_axis** | Dynamic range based axis limits |
| **motion** | Принимает команды движения NML, взаимодействует с HAL в реальном времени |
| **simple_tp** | This component is a single axis simple trajectory planner, same as used for jogging in LinuxCNC. |
| **tpcomp** | Trajectory Planning (tp) module skeleton |

#### 5.7.1.10 Управление двигателем (в реальном времени)

| | |
|---|---|
| **at_pid** | Пропорциональный/интегральный/производный регулятор с автоматической настройкой. |
| **bldc** | Компонент управления BLDC и AC-сервоприводом |
| **clarke2** | Версия преобразования Кларка с двумя входами |
| **clarke3** | Преобразование Кларка (3-фазное в Декартово) |
| **clarkeinv** | Обратное преобразование Кларка |
| **encoder** | Программный подсчет сигналов квадратурного энкодера, см. вescription. |
| **pid** | Пропорциональный/интегральный/производный регулятор, description. |
| **pwmgen** | Программная генерация PWM/PDM, см. description. |
| **stepgen** | Программная генерация шаговых импульсов, см. description. |

#### 5.7.1.11 Моделирование/Тестирование

| | |
|---|---|
| **axistest** | Used to allow testing of an axis. Used In PnCConf. |
| **rtapi_app** | creates a simulated real time environment |
| **sim-torch** | A simulated plasma torch |
| **sim_axis_hardware** | A component to simulate home and limit switches |
| **sim_home_switch** | Home switch simulator |
| **sim_matrix_kb** | convert HAL pin inputs to key codes |
| **sim_parport** | Компонент для имитации контактов компонента hal_parport |
| **sim_spindle** | Simulated spindle with index pulse |
| **simulate_probe** | имитировать входной сигнал зонда |

#### 5.7.1.12 Другое (в реальном времени)

| | |
|---|---|
| **anglejog** | Jog two axes (or joints) at an angle |

| | |
|---|---|
| **classicladder** | Программный ПЛК реального времени на основе релейной логики. Дополнительную информацию смотрите в главе ClassicLadder. |
| **charge_pump** | Creates a square-wave for the *charge pump* input of some controller boards. |
| **encoder_ratio** | Электронный редуктор для синхронизации двух осей. |
| **enum** | Enumerate integer values into bits |
| **eoffset_per_angle** | Compute External Offset Per Angle |
| **gladevcp (Realtime)** | отображает виртуальные панели управления, созданные с помощью GTK/GLADE |
| **histobins** | Histogram bins utility for scripts/hal-histogram |
| **joyhandle** | Устанавливает нелинейные движения джойстика, зоны нечувствительности и масштабы. |
| **latencybins** | Comp utility for scripts/latency-histogram |
| **message** | Display a message |
| **moveoff** | Component for HAL-only offsets |
| **raster** | Outputs laser power based upon pre programmed rastering data |
| **sampler** | Выборка данных из HAL в реальном времени. |
| **siggen** | Генератор сигналов, см. description. |
| **sphereprobe** | Исследовать воображаемое полушарие. |
| **threads** | Создает потоки HAL жесткого реального времени. |
| **threadtest** | Компонент для тестирования поведения потока. |
| **steptest** | Используется StepConf для проверки значений ускорения и скорости оси. |
| **streamer** | Потоковая передача данных файла в HAL в режиме реального времени. |
| **supply** | Установить выходные контакты со значениями из параметров (устарело). |

| | |
|---|---|
| **laserpower** | Scales laser power output based upon velocity input power and distance to go |
| **lcd** | Передача данных HAL на ЖК-экран |
| **matrix_kb** | Convert integers to HAL pins. Optionally scan a matrix of I/O ports to create those integers. |

| | |
|---|---|
| **gearchange** | Выберите один из двух диапазонов скорости. |
| **orient** | Provide a PID command input for orientation mode based on current spindle position, target angle and orient mode |
| **spindle** | Control a spindle with different acceleration and deceleration and optional gear change scaling |
| **spindle_monitor** | Spindle at-speed and underspeed detection |

| | |
|---|---|
| **carousel** | Orient a toolchanger carousel using various encoding schemes |
| **hal_manualtoolchange** | Компонент HAL, работающий не в реальном времени, позволяющий вручную менять инструменты&. |

| | |
|---|---|
| **thc** | Управление высотой факела с помощью карты Mesa THC или любого входа аналог-в-скорость |
| **thcud** | Torch Height Control Up/Down Input |
| **ohmic** | LinuxCNC HAL component that uses a Mesa THCAD (A/D card) for ohmic sensing |
| **plasmac** | A plasma cutter controller |

### 5.7.2 Вызовы HAL API

hal_add_funct_to_thread.3
hal_bit_t.3
hal_create_thread.3
hal_del_funct_from_thread.3
hal_exit.3
hal_export_funct.3
hal_export_functf.3
hal_float_t.3
hal_get_lock.3
hal_init.3
hal_link.3
hal_malloc.3
hal_param_bit_new.3
hal_param_bit_newf.3
hal_param_float_new.3
hal_param_float_newf.3
hal_param_new.3
hal_param_s32_new.3
hal_param_s32_newf.3
hal_param_u32_new.3
hal_param_u32_newf.3
hal_parport.3
hal_pin_bit_new.3
hal_pin_bit_newf.3
hal_pin_float_new.3
hal_pin_float_newf.3
hal_pin_new.3
hal_pin_s32_new.3
hal_pin_s32_newf.3
hal_pin_u32_new.3
hal_pin_u32_newf.3
hal_ready.3
hal_s32_t.3
hal_set_constructor.3
hal_set_lock.3
hal_signal_delete.3
hal_signal_new.3
hal_start_threads.3
hal_type_t.3
hal_u32_t.3
hal_unlink.3
hal.3

### 5.7.3 RTAPI-вызовы

EXPORT_FUNCTION.3
MODULE_AUTHOR.3
MODULE_DESCRIPTION.3
MODULE_LICENSE.3
RTAPI_MP_ARRAY_INT.3
RTAPI_MP_ARRAY_LONG.3
RTAPI_MP_ARRAY_STRING.3
RTAPI_MP_INT.3

RTAPI_MP_LONG.3
RTAPI_MP_STRING.3
rtapi.3
rtapi_app_exit.3
rtapi_app_main.3
rtapi_clock_set_period.3
rtapi_delay.3
rtapi_delay_max.3
rtapi_exit.3
rtapi_get_clocks.3
rtapi_get_msg_level.3
rtapi_get_time.3
rtapi_inb.3
rtapi_init.3
rtapi_module_param.3
RTAPI_MP_ARRAY_INT.3
RTAPI_MP_ARRAY_LONG.3
RTAPI_MP_ARRAY_STRING.3
RTAPI_MP_INT.3
RTAPI_MP_LONG.3
RTAPI_MP_STRING.3
rtapi_mutex.3
rtapi_outb.3
rtapi_print.3
rtapi_prio.3
rtapi_prio_highest.3
rtapi_prio_lowest.3
rtapi_prio_next_higher.3
rtapi_prio_next_lower.3
rtapi_region.3
rtapi_release_region.3
rtapi_request_region.3
rtapi_set_msg_level.3
rtapi_shmem.3
rtapi_shmem_delete.3
rtapi_shmem_getptr.3
rtapi_shmem_new.3
rtapi_snprintf.3
rtapi_task_delete.3
rtapi_task_new.3
rtapi_task_pause.3
rtapi_task_resume.3
rtapi_task_start.3
rtapi_task_wait.3

## 5.8 HAL Component Descriptions

В этой главе представлена подробная информация об основных функциях LinuxCNC, которые требуют точного времени для

- генерации сигналов, которые интерпретируются оборудованием (например, двигателями) или
- для интерпретации сигналов, отправляемых оборудованием (например, энкодерами).

## 5.8.1 StepGen

Этот компонент обеспечивает программную генерацию шаговых импульсов в ответ на команды положения или скорости. В режиме позиционирования он имеет встроенный предварительно настроенный контур позиционирования, поэтому настройка ПИД-регулятора не требуется. В режиме скорости он приводит двигатель в движение с заданной скоростью, соблюдая при этом ограничения скорости и ускорения. Это компонент, работающий только в режиме реального времени, и в зависимости от скорости процессора и т. д. он способен работать с максимальной частотой шагов от 10 кГц до, возможно, 50 кГц. Блок-схема генератора шаговых импульсов показывает три блок-схемы, каждая из которых представляет собой один генератор шаговых импульсов. Первая диаграмма предназначена для типа шага *0* (step и direction). Второй предназначен для типа шага *1* (up/down или pseudo-PWM), а третий — для типов шагов со 2 по 14 (различные шаблоны шагов). Первые две диаграммы показывают управление режимом положения, а третья – режим скорости. Режим управления и тип шага задаются независимо, можно выбрать любую комбинацию.

**Структурная схема генератора шаговых импульсов в режиме положения**

:images/stepgen-block-diag.png

**Загрузка компонента stepgen**

```
halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

**<type-array>**
: представляет собой серию десятичных целых чисел, разделенных запятыми. Каждое число вызывает загрузку одиночного генератора шаговых импульсов, значение числа определяет тип шага.

**<ctrl_array>**
: представляет собой серию символов *p* или *v*, разделенных запятыми, для указания режима положения или скорости.

**<ctrl_type>**
: является необязательным, если его опустить, все генераторы шагов будут работать в режиме положения.

Например:

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

Установит три генератора шаговых импульсов. Первые два используют тип шага *0* (step и direction) и выполняются в режиме позиционирования. Последний использует тип шага *2* (квадратура) и работает в режиме скорости. Значением по умолчанию для *<config-array>* является *0,0,0*, при котором будут установлены три генератора типа *0* (step/dir). Максимальное количество генераторов шаговых импульсов — 8 (как определено в MAX_CHAN в файле Stepgen.c). Каждый генератор независим, но все они обновляются с помощью одной и той же функции одновременно. В следующих описаниях *<chan>* — это номер конкретного генератора. Первый генератор имеет номер 0.

**Выгрузка компонента stepgen**

```
halcmd: unloadrt stepgen
```

### 5.8.1.1 Контакты

От выбранных типа шаговых импульсов и типа управления.

- (float) stepgen.__<chan>__.position-cmd` - Требуемое положение двигателя, в единицах положения (только режим положения).
- (float) stepgen.__<chan>__.velocity-cmd` - Требуемая скорость двигателя, в единицах положения в секунду (только режим скорости).
- (s32) stepgen.__<chan>__.counts` - Позиция обратной связи в отсчетах, обновленная с помощью *capture_position()*.
- (float) stepgen.__<chan>__.position-fb` - Положение обратной связи в единицах положения, обновленное с помощью *capture_position()*.
- (bit) stepgen.__<chan>__.enable` - Включает выходные шаговые импульсы — когда false, импульсы не генерируются..
- (bit) stepgen.__<chan>__.step` - Шаговый импульсный выход (только импульс типа 0).
- (bit) stepgen.__<chan>__.dir` - Выход direction (только тип импульса 0).
- (bit) stepgen.__<chan>__.up` - Псевдо-ШИМ-выход UP (только импульс типа 1).
- (bit) stepgen.__<chan>__.down` - Выход псевдо-ШИМ DOWN (только импульс типа 1).
- (bit) stepgen.__<chan>__.phase-A` - Выход фазы A (только типы импульсов 2–14).
- (bit) stepgen.__<chan>__.phase-B` - Выход фазы B (только типы импульсов 2–14).
- (bit) stepgen.__<chan>__.phase-C` - Выход фазы C (только типы импульсов 3–14).
- (bit) stepgen.__<chan>__.phase-D` - Выход фазы D (только типы импульсов 5–14).
- (bit) stepgen.__<chan>__.phase-E` - Выход фазы E (только типы импульсов 11–14).

#### 5.8.1.2 Параметры

* (float) stepgen.__<chan>__.position-scale` - Шагов на единицу позиции. Этот параметр используется как для выхода, так и для обратной связи. * (float) stepgen.__<chan>__.maxvel` - Максимальная скорость в единицах положения в секунду. Если 0.0, не имеет эффекта. * (float) stepgen.__<chan>__.m - Максимальная скорость ускорения/замедления, в единицах позиции в секунду в квадрате. Если 0.0, не имеет эффекта. * (float) stepgen.__<chan>__.frequency` - Текущая частота шагов в шагах в секунду. * (float) stepgen.__<chan>__.steplen` - Длина шагового импульса (типы импульсов 0 и 1) или минимальное время в данном состоянии (типы импульсов 2–14) в наносекундах. * (float) stepgen.__<chan>__.stepspace` - Минимальный интервал между двумя шаговыми импульсами (только типы импульсов 0 и 1), в наносекундах. Установите значение 0, чтобы включить функцию *doublefreq* генератора шаговых импульсов. Чтобы использовать *doublefreq*, необходимо включить parport reset function. * (float) stepgen.__<chan>__.dirsetup` - Минимальное время от смены направления до начала следующего импульса шага (только тип импульсов 0), в наносекундах. * (float) stepgen.__<chan>__.dirhold` - Минимальное время от окончания шагового импульса до изменения направления (только тип импульсов 0), в наносекундах. * (float) stepgen.__<chan>__.dirde - Минимальное время от любого импульса до импульса в противоположном направлении (только типы импульсов 1–14), в наносекундах. * (s32) stepgen.__<chan>__.rawcounts` - Необработанные отсчеты обратной связи, обновленные с помощью *make_pulses()*.

В режиме позиционирования значения maxvel и maxaccel используются внутренним контуром положения, чтобы избежать генерации последовательностей шаговых импульсов, которым двигатель не может следовать. При установке значений, соответствующих двигателю, даже большое мгновенное изменение заданного положения приведет к плавному трапецеидальному перемещению в новое положение. Алгоритм работает путем измерения как ошибки положения, так и ошибки скорости, а также расчета ускорения, которое пытается одновременно свести их к нулю. Для получения более подробной информации, включая содержимое поля *уравнение управления*, обратитесь к коду.

В режиме скорости, maxvel — это простой предел, который применяется к заданной скорости, а maxaccel используется для линейного изменения фактической частоты, если заданная скорость резко меняется. Как и в режиме позиционирования, правильные значения этих параметров гарантируют, что двигатель сможет следовать сгенерированной последовательности импульсов.

#### 5.8.1.3 Типы шаговых импульсов

Генератор шаговых импульсов поддерживает 15 различных *последовательностей шаговых импульсо*

**Шаговые импульсы Тип 0** Тип шага 0 — это стандартный тип импульса шага и направления. При настройке для импульса шага типа 0 есть четыре дополнительных параметра, которые определяют точное время импульса шага и сигналов направления. На следующем рисунке показано значение этих параметров. Параметры указаны в наносекундах, но будут округлены до целого числа кратного периоду потока для потока, вызывающего *make_pulses()*. Например, если *make_pulses()* вызывается каждые 16 мкс, а значение Steplen равно 20000, то длительность шаговых импульсов будет 2 x 16 = 32 мкс. Значение по умолчанию для всех четырех параметров — 1 нс, но автоматическое округление вступает в силу при первом запуске кода. Поскольку для одного импульса шага требуется высокий уровень *steplen* ns и низкий уроввень *stepspace* ns , максимальная частота равна 1,000,000,000, разделенному на (*steplen* + *stepspace*)'. Если *maxfreq* установлено выше этого предела, оно будет понижено автоматически. Если *maxfreq* равно нулю, оно останется нулевым, но выходная частота по-прежнему будет ограничена.

При использовании драйвера параллельного порта частоту шаговых импульсов можно удвоить с помощью функции parport reset вместе с настройкой *doublefreq* StepGen.

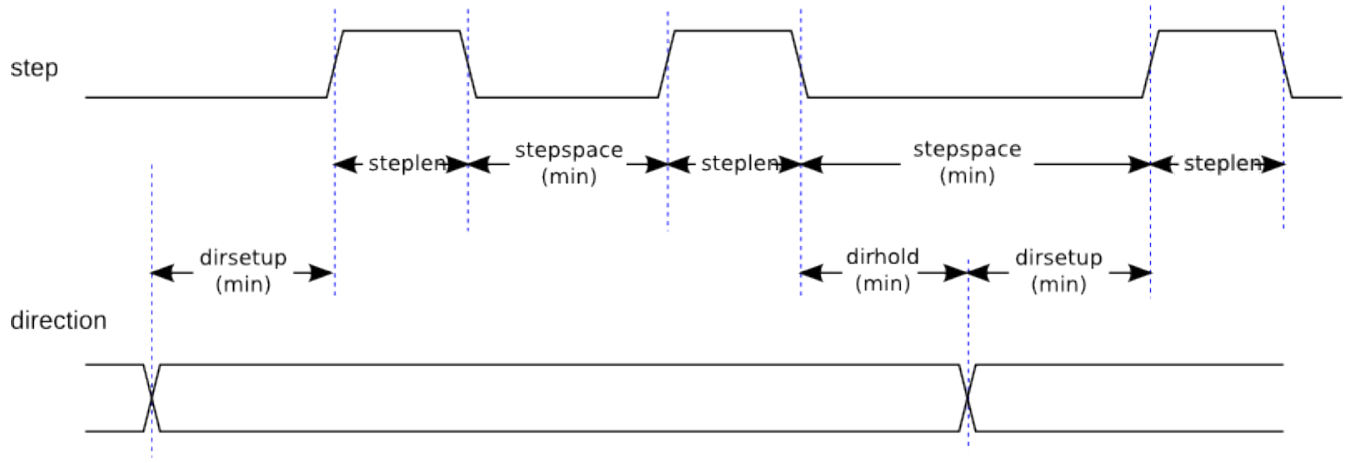



Figure 5.19: Временная диаграмма импульсов шага и направления

**Шаговые импульсы Тип 1** Шаговый импульс типа 1 имеет два выхода: вверх и вниз. Импульсы появляются то на одном, то на другом, в зависимости от направления движения. Каждый импульс имеет длину *steplen* нс, и импульсы разделены по крайней мере *stepspace* нс. Максимальная частота такая же, как и для шагового импульса типа 0. Если *maxfreq* установлено выше предельного значения, она будет понижена. Если *maxfreq* равно нулю, оно останется нулевым, но выходная частота по-прежнему будет ограничена.

**Warning**
Не используйте функцию parport reset с типами шаговых импульсов 2–14. Могут возникнуть неожиданные результаты.

**Шаговые импульсы Тип 2 - 14** Типы шаговых импульсов с 2 по 14 основаны на состоянии и имеют от двух до пяти выходов. На каждом шаге счетчик состояния увеличивается или уменьшается. Двух- и трехфазный, четырехфазный и пятифазный режимы показывают выходные шаблоны в зависимости от счетчика состояний. Максимальная частота равна 1,000,000,000, разделенному на *steplen*, и, как и в других режимах, *maxfreq* будет понижена, если она превышает предел.

**Типы двух- и трехфазных шаговых импульсов** Типы шаговых импульсов: двух- и трехфазные

**Типы четырехфазных шаговых импульсов** Типы шаговых импульсов: Четырехфазные

**Типы пятифазных шаговых импульсов** Типы шаговых импульсов: Пятифазные

#### 5.8.1.4 Функции

Компонент экспортирует три функции. Каждая функция действует на все генераторы шаговых импульсов — запуск разных генераторов в разных потоках не поддерживается.

- (funct) `stepgen.make-pulses` - High speed function to generate and count pulses.
- (funct) `stepgen.update-freq` - Функция низкой скорости выполняет преобразование положения в скорость, масштабирование и ограничение.
- (funct) `stepgen.capture-position` - Функция низкой скорости для обратной связи, обновления защелок и масштабирования положения.

Высокоскоростная функция *stepgen.make-pulses* должна выполняться в очень быстром потоке, от 10 до 50 мкс в зависимости от возможностей компьютера. Период этого потока определяет максимальную частоту шагов, поскольку *steplen*, *stepspace*, *dirsetup*, *dirhold* и *dirdelay* округляются до целого числа, кратного периоду потока в наносекундах. Две другие функции могут вызываться с гораздо меньшей скоростью.

### 5.8.2 PWMgen

Этот компонент обеспечивает программную генерацию сигналов ШИМ (широтно-импульсной модуляции) и ПИМ (плотно-импульсной модуляции). Это компонент, работающий только в режиме реального времени, и в зависимости от скорости процессора и т. д. он способен работать с частотой ШИМ от нескольких сотен герц при довольно хорошем разрешении до, возможно, 10 кГц при ограниченном разрешении.

**Загрузка PWMgen**

```
loadrt pwmgen output_type=<config-array>
```

*<config-array>* представляет собой серию десятичных целых чисел, разделенных запятыми. Каждое число вызывает загрузку одного генератора ШИМ, значение числа определяет тип выхода. В следующем примере будут установлены три генератора ШИМ. Значения по умолчанию нет. Если не указан *<config-array>*, генераторы ШИМ не будут установлены. Максимальное количество генераторов частоты — 8 (как определено параметром MAX_CHAN в pwmgen.c). Каждый генератор независим, но все они обновляются с помощью одной и той же функции одновременно. В следующих описаниях <chan> — это номер конкретного генератора. Первый генератор имеет номер 0.

**Загрузка примера PWMgen**

```
loadrt pwmgen output_type=0,1,2
```

Установит три ШИМ-генератора. Первый будет использовать выход типа *0* (только PWM ), следующий будет использовать выход типа 1 (PWM и направление), а третий будет использовать выход типа 2 (ВВЕРХ и ВНИЗ). Значения по умолчанию нет, если *<config-array>* не указан, ШИМ-генератор

установлен не будет. Максимальное количество генераторов частоты — 8 (как определено параметром MAX_CHAN в pwmgen.c). Каждый генератор независим, но все они обновляются с помощью одной и той же функции одновременно. В последующих описаниях <*chan*> — это количество конкретных генераторов. Нумерация ШИМ-генераторов начинается с 0.

**Выгрузка PWMgen**

```
unloadrt pwmgen
```

#### 5.8.2.1 Типы выходов

Генератор ШИМ поддерживает три различных *output types*.

- *Output type 0* - только выходной контакт ШИМ. Принимаются только положительные команды, отрицательные значения рассматриваются как ноль (на них влияет параметр *min-dc*, если он не равен нулю).
- *Output type 1* - PWM/PDM и контакты направления. Положительные и отрицательные входы будут выводиться как положительные и отрицательные ШИМ. Контакт направления является ложным для положительных команд и истинным для отрицательных команд. Если вашему элементу управления требуется положительный ШИМ как для CW, так и для CCW, используйте компонент abs для преобразования вашего сигнала ШИМ в положительное значение, когда на вход подается отрицательный вход.
- *Output type 2* - контакты UP и DOWN. Для положительных команд сигнал ШИМ появляется на выходе UP, а выход DOWN остается false. Для отрицательных команд сигнал ШИМ появляется на выходе DOWN, а на выход UP остается false. Выход типа 2 подходит для управления большинством H-мостов.

#### 5.8.2.2 Контакты

Каждый генератор ШИМ будет иметь следующие контакты:

- (float) pwmgen.__<chan>__.value - Значение команды в произвольных единицах. Будет масштабиров параметром *scale* (см. ниже).
- (bit) pwmgen.__<chan>__.enable - Включает или отключает выходы генератора ШИМ.

Каждый генератор ШИМ также будет иметь некоторые из этих контактов, в зависимости от выбранного типа выхода:

- (bit) pwmgen.__<chan>__.pwm - Выход ШИМ (или ПИМ) (только типы выходов 0 и 1).
- (bit) pwmgen.__<chan>__.dir - Выход направления (только тип выхода 1).
- (bit) pwmgen.__<chan>__.up - Выход ШИМ/ПИМ для положительного входного значения (только тип выхода 2).
- (bit) pwmgen.__<chan>__.down - Выход ШИМ/ПИМ для отрицательного входного значения (только тип выхода 2).

#### 5.8.2.3 Параметры

- (float) pwmgen.`__<chan>__.scale` - Коэффициент масштабирования для преобразования value из произвольных единиц в рабочий цикл. Например, если масштаб установлен на 4000, а входное значение, переданное в pwmgen.`__<chan>__.value`, равно 4000, то это будет 100% сважность (всегда включен). Если значение равно 2000, то это будет меандр с частотой 50% и частотой 25 Гц.
- (float) pwmgen.`__<chan>__.pwm-freq` - желаемая частота ШИМ, в Гц. Если 0,0, генерирует PDM вместо PWM. Если установлено значение выше внутреннего предела, следующий вызов update_freq() установит внутренний предел. Если ненулевое значение и значение параметра *dither* равно false, следующий вызов *update_freq()* установит его в ближайшее целое число, кратное периоду функции *make_pulses()*.
- (bit) pwmgen.`__<chan>__.dither-pwm` - Если это правда, включает дизеринг для достижения средних частот ШИМ или скважности, которые невозможно получить с помощью чистого ШИМ. Если значение равно false, и частота ШИМ, и скважность будут округлены до значений, которые могут быть точно достигнуты.
- (float) pwmgen.`__<chan>__.min-dc` - Минимальная скважность от 0,0 до 1,0 (при отключении скважность будет равна нулю, независимо от этой настройки).
- (float) pwmgen.`__<chan>__.max-dc` - Максимальная скважность от 0,0 до 1,0.
- (float) pwmgen.`__<chan>__.curr-dc` - Текущая скважность – после всех ограничений и округлений (только чтение).

#### 5.8.2.4 Функции

Компонент экспортирует две функции. Каждая функция действует на все генераторы ШИМ — запуск разных генераторов в разных потоках не поддерживается.

- (funct) pwmgen.make-pulses - High speed function to generate PWM waveforms. The high speed function pwmgen.make-pulses should be run in the base (fastest) thread, from 10 to 50 µs depending on the capabilities of the computer. That thread's period determines the maximum PWM carrier frequency, as well as the resolution of the PWM or PDM signals. If the base thread is 50,000 ns then every 50 µs the module decides if it is time to change the state of the output. At 50% duty cycle and 25 Hz PWM frequency this means that the output changes state every (1/25) s / 50 µs * 50% = 400 iterations. This also means that you have a 800 possible duty cycle values (without dithering).
- (funct) pwmgen.update - Функция низкой скорости для масштабирования и ограничения значения, а также обработки других параметров. Это функция модуля, который выполняет более сложную математическую работу, чтобы определить, для скольких базисных периодов выходные данные должны быть высокими, а для скольких — низкими.

### 5.8.3 Энкодер

Этот компонент обеспечивает программный подсчет сигналов от квадратурных (или одноимпульсных) энкодеров. Это компонент, работающий только в режиме реального времени, и в зависимости от скорости процессора, задержки и т. д. он способен поддерживать максимальную скорость счета от 10 кГц до, возможно, до 50 кГц.

Базовый поток должен быть 1/2 скорости счета , чтобы учитывать шум и изменения времени. Например, если у вас есть энкодер со 100 импульсами на оборот на шпинделе и максимальная частота вращения составляет 3000, максимальный период базового потока должен составлять 25 мкс. Энкодер с 100 импульсами на оборот будет иметь 400 отсчетов. Скорость шпинделя

3000 RPM = 50 RPS (оборотов в секунду). 400 * 50 = 20,000 отсчетов в секунду или 50 мкс между отсчетами.

Блок-схема счетчика энкодера представляет собой блок-схему одного канала счетчика энкодера.

**Блок-схема счетчика энкодера**

:images/encoder-block-diag.png

**Загрузка энкодера**

```
halcmd: loadrt encoder [num_chan=<counters>]
```

*<counters>* — количество счетчиков энкодера, которое вы хотите установить. Если *num_chan* не указан, будут установлены три счетчика. Максимальное количество счетчиков — 8 (как определено в MAX_CHAN в encoder.c). Каждый счетчик независим, но все они обновляются одной и той же функцией одновременно. В следующих описаниях *<chan>* — это номер конкретного счетчика. Первый счетчик имеет номер 0.

**Выгрузка энкодера**

```
halcmd: unloadrt encoder
```

#### 5.8.3.1 Контакты

- `encoder._<chan>_.counter-mode` (bit, I/O) (по умолчанию: FALSE) — включает режим счетчика. Если значение истинно, счетчик подсчитывает каждый фронт входного сигнала фазы A, игнорируя значение на фазе B. Это полезно для подсчета выходного сигнала одноканального (неквадратурного) датчика. Если значение false, оно считается в квадратурном режиме.
- `encoder._<chan>_.missing-teeth` (s32, In) (по умолчанию: 0) — включает использование индекса отсутствующих зубов. Это позволяет одному контакту ввода-вывода предоставлять информацию как о положении, так и об индексе. Если колесо энкодера имеет 58 зубцов, из которых два отсутствуют, и расположены так, как если бы их было 60 (обычно для автомобильных датчиков коленвала), тогда шкала положения должна быть установлена на 60, а отсутствующие зубья - на 2. Для использования этого режима counter-mode должен быть установлен в true. Этот режим будет работать для нарезания резьбы на токарном станке, но не для жесткого нарезания резьбы.
- `encoder._<chan>_.counts` (s32, Out) - Позиция в отсчетах энкодера.
- `encoder._<chan>_.counts-latched` (s32, Out) - В настоящее время не используется.
- `encoder._<chan>_.index-enable` (bit, I/O) - Когда True, `counts` и `position` сбрасываются в ноль при следующем фронте фазы Z.
  В то же время `index-enable` сбрасывается в ноль, указывая на то, что произошел фронт. Вывод `index-enable` является двунаправленным. Если параметр `index-enable`- False, канал фазы Z энкодера будет игнорироваться, и счетчик будет считать нормально. Драйвер энкодера никогда не установит параметр `index-enable` в значение True. Однако это может сделать какой-то другой компонент.
- `encoder._<chan>_.latch-falling` (bit, In) (default: TRUE) - В настоящее время не используется.
- `encoder._<chan>_.latch-input` (bit, In) (default: TRUE) - В настоящее время не используется.
- `encoder._<chan>_.latch-rising` (bit, In) - В настоящее время не используется.
- `encoder._<chan>_.min-speed-estimate` (float, in) - Определите минимальную истинную величину скорости, при которой скорость будет оценена как ненулевая, а position-interpolated будет интерполирована. Единицы `min-speed-estimate` такие же, как и единицы `velocity`. Масштабный коэффициент, в отсчетах на единицу длины. Установка слишком низкого значения этого параметра приведет к тому, что скорость достигнет 0 после прекращения поступления импульсов энкодера.

- encoder._<chan>_.phase-A (bit, In) - Фаза A сигнала квадратурного энкодера.
- encoder._<chan>_.phase-B (bit, In) - Фаза B сигнала квадратурного энкодера.
- encoder._<chan>_.phase-Z (bit, In) - Фаза Z (индексный импульс) сигнала квадратурного энкодера.
- encoder._<chan>_.position (float, Out) - Позиция в масштабированных единицах (см. position-scal
- encoder._<chan>_.position-interpolated (float, Out) -Позиция в масштабированных единицах, интерполированная между отсчетами энкодера.
  position-interpolated пытается интерполировать между отсчетами энкодера на основе последней измеренной скорости. Действительно только тогда, когда скорость приблизительно постоянна и превышает min-speed-estimate. Не используйте для управления положением, поскольку его значение неверно на низких скоростях, при реверсе направления и при изменении скорости. Тем не менее, он позволяет использовать энкодер с низким ppr (включая *encoder* с одним импульсом на оборот) для нарезания резьбы на токарном станке, а также может иметь и другие применения.
- encoder._<chan>_.position-latched (float, Out) - В настоящее время не используется.
- encoder._<chan>_.position-scale (float, I/O) - Масштабный коэффициент, в отсчетах на единицу длины. Например, если масштаб позиции равен 500, то 1000 отсчетов энкодера будут выданы как позиция в 2.0 единицы.
- encoder._<chan>_.rawcounts (s32, In) - Необработанные отсчеты, определенные update-counters. Это значение обновляется чаще, чем отсчеты и позиция. На него также не влияет сброс или индексный импульс.
- encoder._<chan>_.reset (bit, In) - Когда True, принудительно немедленно обнуляются *counts* и ’position.
- encoder._<chan>_.velocity (float, Out) - Скорость в масштабированных единицах в секунду. encoder использует алгоритм, который значительно снижает шум квантования по сравнению с простым дифференцированием выходного сигнала *position*. Когда величина истинной скорости ниже min-speed-estimate, выход скорости равен 0.
- encoder._<chan>_.x4-mode (bit, I/O) (default: TRUE) - Включает режим X4. Когда в true, счетчик считает каждый фронт квадратурного сигнала (четыре отсчета за полный цикл). Если значение false, оно учитывается только один раз за полный цикл. В режиме счетчика этот параметр игнорируется. Режим 1x полезен для некоторых джойстиков.

#### 5.8.3.2 Параметры

- 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 Функции

Компонент экспортирует две функции. Каждая функция действует на все счетчики энкодера — запуск разных счетчиков в разных потоках не поддерживается.

- (funct) encoder.update-counters - High speed function to count pulses.
- (funct) encoder.capture-position - Функция низкой скорости для обновления защелок и масштабиро позиции.

## 5.8.4 ПИД

Этот компонент обеспечивает контуры пропорционального/интегрального/производного управления. Это компонент только реального времени. Для простоты, в этом обсуждении предполагается, что мы говорим о контурах положения, однако этот компонент можно использовать для реализации других контуров обратной связи, таких как скорость, высота горелки, температура и т. д. Блок-схема контура ПИД-регулятора представляет собой блок-схему одного контура ПИД-регулятора.
.

**Блок-схема ПИД-контура**

:images/pid-block-diag.png

**Загрузка ПИД**

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

*<loops>* — количество контуров ПИД, которое вы хотите установить. Если *num_chan* не указан, будет установлен один контур. Максимальное количество контуров — 16 (согласно значению MAX_CHAN в pid.c). Каждый цикл полностью независим. В следующих описаниях *<loopnum>* представляет собой номер конкретного цикла. Первый цикл имеет номер 0.

Если `debug=1`, компонент экспортирует несколько дополнительных контактов, которые могут быть полезны во время отладки и настройки. По умолчанию дополнительные контакты не экспортируют чтобы сэкономить место в общей памяти и не загромождать список контактов.

**Выгрузка ПИД**

```
halcmd: unloadrt pid
```

### 5.8.4.1 Контакты

Три наиболее важных контакта:

- (float) `pid.`__<loopnum>__.command` - Желаемое положение, заданное другим компонентом системы.
- (float) `pid.`__<loopnum>__.feedback` - Текущее положение, измеренное устройством обратной связи, например энкодером.
- (float) `pid.`__<loopnum>__.output` - Команда скорости, которая пытается переместиться из текущего положения в желаемое.

Для контура положения `.command` и `.feedback` указаны в единицах измерения положения. Для линейной оси это могут быть дюймы, мм, метры или что-то еще. Аналогично, для угловой оси это могут быть градусы, радианы и т. д. Единицы измерения вывода `.output` представляют собой изменение, необходимое для того, чтобы обратная связь соответствовала команде. Таким образом, для цикла положения `.output` — это скорость в дюймах/с, мм/с, градусах/с и т. д. Единицами времени всегда являются секунды, а единицы скорости соответствуют единицам положения. Если команда и обратная связь указаны в метрах, то выходной сигнал измеряется в метрах в секунду.

Каждый контур имеет два контакта, которые используются для мониторинга или управления общей работой компонента.

- *(float) pid.*<loopnum>*.error* - Равно *.command* минус *.feedback*.
- *(bit) pid.*<loopnum>*.enable* - Бит, который включает контур. Если *.enable* - false, все интеграторы сбрасываются, а выходной сигнал обнуляется. Если *.enable* - true, контур работает нормально.

Контакты, используемые для сообщения о насыщении. Насыщение происходит, когда выход блока ПИД достигает максимального или минимального предела.

- *(bit) pid.*<loopnum>*.saturated* - True , когда выход насыщен.
- *(float) pid.*<loopnum>*.saturated_s* - Время насыщения выхода.
- *(s32) pid.*<loopnum>*.saturated_count* - Время насыщения выхода.

Коэффициенты усиления, пределы ПИД-регулятора и другие *настраиваемые* функции контура доступны в виде контактов, поэтому их можно динамически регулировать для более расширенных возможностей настройки.

- *(float) pid.*<loopnum>*.Pgain* - Пропорциональное усиление
- *(float) pid.*<loopnum>*.Igain* - Интегральный коэффициент усиления
- *(float) pid.*<loopnum>*.Dgain* - Производное усиление
- *(float) pid.*<loopnum>*.bias* - Постоянное смещение на выходе
- *(float) pid.*<loopnum>*.FF0* - Упреждение нулевого порядка – выходной сигнал пропорционален команде (положению).
- *(float) pid.*<loopnum>*.FF1* - Упреждающая связь первого порядка – выходной сигнал пропорционален производной команды (скорости).
- *(float) pid.*<loopnum>*.FF2* - Прямая связь второго порядка - выходной сигнал пропорционален 2й производной команды (ускорения).
- *(float) pid.*<loopnum>*.deadband* - Количество ошибок, которые будут игнорироваться
- *(float) pid.*<loopnum>*.maxerror* - Ограничение на ошибку
- *(float) pid.*<loopnum>*.maxerrorI* - Ограничение на интегратор ошибок
- *(float) pid.*<loopnum>*.maxerrorD* - Ограничение на производную ошибки
- *(float) pid.*<loopnum>*.maxcmdD* - Ограничение на производную команды
- *(float) pid.*<loopnum>*.maxcmdDD* - Ограничение на 2ю производную команды
- *(float) pid.*<loopnum>*.maxoutput* - Ограничение выходного значения

Все *max** ограничения реализованы таким образом, что если значение этого параметра равно нулю, ограничения нет.

Если при установке компонента было указано *debug=1*, будут экспортированы четыре дополнительных контакта:

- *(float) pid.*<loopnum>*.errorI* - Интеграл ошибки.
- *(float) pid.*<loopnum>*.errorD* - Производная ошибки.
- *(float) pid.*<loopnum>*.commandD* - Производная команды.
- *(float) pid.*<loopnum>*.commandDD* - 2я производная команды.

#### 5.8.4.2 Функции

Компонент экспортирует одну функцию для каждого контура ПИД. Эта функция выполняет все вычисления, необходимые для контура. Поскольку каждый контур имеет свою собственную функцию, отдельные контуры могут быть включены в разные потоки и выполняться с разной скоростью.

- *(funct) pid.*<loopnum>*.do_pid_calcs* - Выполняет все вычисления для одного контура ПИД.

Если вы хотите понять точный алгоритм, используемый для вычисления выхода контура ПИД-регулятора, см.

- figure PID Loop Block Diagram,
- комментарии в начале *emc2/src/hal/comComponents/pid.c* и, конечно же,
- сам код.

Вычисления цикла находятся в функции C *calc_pid()*.

### 5.8.5 Имитированный энкодер

Имитируемый кодер именно такой. Он производит квадратурные импульсы с индексным импульсом со скоростью, контролируемой выводом HAL. В основном полезно для тестирования.

**Загрузка sim-encoder**

```
halcmd: loadrt sim-encoder num_chan=<number>
```

*<number>* — количество энкодеров, которые вы хотите имитировать. Если не указано, будет установлен один энкодер. Максимальное число — 8 (как определено MAX_CHAN в sim_encoder.c).

**Выгрузка sim-encoder**

```
halcmd: unloadrt sim-encoder
```

#### 5.8.5.1 Контакты

- (float) `sim-encoder.` `__<chan-num>__.speed` - Команда скорости для моделируемого вала.
- (bit) `sim-encoder.` `__<chan-num>__.phase-A` - Квадратурный выход.
- (bit) `sim-encoder.` `__<chan-num>__.phase-B` - Квадратурный выход.
- (bit) `sim-encoder.` `__<chan-num>__.phase-Z` - Выход индексного импульса.

Когда `.speed` положительна, `.phase-A` опережает `.phase-B`.

#### 5.8.5.2 Параметры

- (u32) `sim-encoder.` `__<chan-num>__.ppr` - Импульсов на оборот.
- (float) `sim-encoder.` `__<chan-num>__.scale` - Масштабный коэффициент для `.speed`. По умолчанию установлено значение 1.0, что означает, что `.speed` измеряется в оборотах в секунду. Измените значение на 60 для числа оборотов в минуту, на 360 для градусов в секунду, на 6,283185 (= 2*п) для радиан в секунду и т.д.

Обратите внимание, что количество импульсов на оборот — это не то же самое, что количество отсчетов на оборот. Импульс представляет собой полный квадратурный цикл. Большинство счетчиков энкодеров считают четыре раза за один полный цикл.

#### 5.8.5.3 Функции

Компонент экспортирует две функции. Каждая функция влияет на все моделируемые энкодеры.

- (funct) `sim-encoder.make-pulses` - High speed function to generate quadrature pulses.
- (funct) `sim-encoder.update-speed` - Функция низкой скорости для чтения `.speed`, масштабирования и настройки `.make-pulses`.

### 5.8.6 Debounce

Debounce — это компонент реального времени, который может фильтровать помехи, создаваемые контактами механических переключателей. Это также может быть полезно в других приложениях, где необходимо подавлять короткие импульсы.

**Загрузка debounce**

```
halcmd: loadrt debounce cfg=<config-string>
```

***<config-string>***
: Представляет собой серию десятичных целых чисел, разделенных запятыми. Каждое число устанавливает группу одинаковых фильтров устранения дребезга, число определяет количество фильтров в группе.

**Загрузка примера debounce**

```
halcmd: loadrt debounce cfg=1,4,2
```

установит три группы фильтров. Группа 0 содержит один фильтр, группа 1 — четыре, а группа 2 — два фильтра. Значение по умолчанию для *<config-string>* — ”1”, при этом будет установлена одна группа, содержащая один фильтр. Максимальное количество групп 8 (как определено в MAX_GROUPS в debounce.c). Максимальное количество фильтров в группе ограничено только объемом общей памяти. Каждая группа полностью независима. Все фильтры в одной группе идентичны, и все они обновляются одной и той же функцией одновременно. В следующих описаниях *<G>* представляет собой номер группы, а *<F>* представляет собой номер фильтра внутри группы. Первый фильтр — группа 0, фильтр 0.

**Выгрузка debounce**

```
halcmd: unloadrt debounce
```

#### 5.8.6.1 Контакты

Каждый отдельный фильтр имеет два контакта.

- (bit) `debounce.` `__<G>__.__<F>__.in` - Ввод фильтра *<F>* в группе *<G>*.
- (bit) `debounce.` `__<G>__.__<F>__.out` - Вывод фильтра *<F>* в группе *<G>*.

#### 5.8.6.2 Параметры

Каждая группа фильтров имеет один параметр, footnote: [Каждый отдельный фильтр также имеет внутреннюю переменную состояния. Существует переключатель времени компиляции, который может экспортировать эту переменную в качестве параметра. Это предназначено для тестирования и при обычных обстоятельствах просто бесполезно расходует разделяемую память.].

• (s32) `debounce.`__<G>__`.delay` - Задержка фильтра для всех фильтров в группе <*G*>.

Задержка фильтра выражается в единицах периодов потока. Минимальная задержка равна нулю. Выходной сигнал фильтра с нулевой задержкой точно соответствует его входному сигналу — он ничего не фильтрует. По мере увеличения `.delay` все более и более длинные сбои отклоняются. Если `.delay` равен 4, все сбои, меньшие или равные четырем периодам потока, будут отклонены.

#### 5.8.6.3 Функции

Каждая группа фильтров имеет одну функцию, которая обновляет все фильтры в этой группе *одновременно*. Различные группы фильтров могут обновляться из разных потоков в разные периоды времени.

• (funct) `debounce.`<*G*> - Обновляет все фильтры в группе <*G*>.

### 5.8.7 SigGen

SigGen — это компонент реального времени, который генерирует прямоугольные, треугольные и синусоидальные сигналы. В основном он используется для тестирования.

**Загрузка siggen**

```
halcmd: loadrt siggen [num_chan=<chans>]
```

**<chans>**
— это количество генераторов сигналов, которые вы хотите установить. Если *numchan* не указан, будет установлен один генератор сигналов. Максимальное количество генераторов — 16 (как определено в MAX_CHAN в siggen.c). Каждый генератор полностью независим. В следующих описаниях есть

***<chan>***
номер конкретного генератора сигналов (нумерация начинается с 0).

**Выгрузка siggen**

```
halcmd: unloadrt siggen
```

#### 5.8.7.1 Контакты

Каждый генератор имеет пять выходных контактов.

• (float) `siggen.`__<chan>__`.sine` - Выход синусоидального сигнала.

• (float) `siggen.`__<chan>__`.cosine` - Выход косинусоидального сигнала.

• (float) `siggen.`__<chan>__`.sawtooth` - Выход пилообразного сигнала.

- (float) `siggen.__<chan>__.triangle` - Выход треугольного сигнала.
- (float) `siggen.__<chan>__.square` - Выход прямоугольного сигнала.

Все пять выходов имеют одинаковую частоту, амплитуду и смещение.

Помимо выходных контактов, имеется три управляющих контакта:

- (float) `siggen.__<chan>__.frequency` - Устанавливает частоту в герцах, значение по умолчанию — 1 Гц.
- (float) `siggen.__<chan>__.amplitude` - Устанавливает пиковую амплитуду выходных сигналов, по умолчанию — 1.
- (float) `siggen.__<chan>__.offset` - Устанавливает смещение по постоянному току выходных сигналов, по умолчанию — 0.

Например, если `siggen.0.amplitude` равен 1,0, а `siggen.0.offset` равен 0,0, выходные данные будут колебаться от -1,0 до +1,0. Если `siggen.0.amplitude` равен 2,5, а `siggen.0.offset` равен 10,0, то выходные значения будут колебаться от 7,5 до 12,5.

#### 5.8.7.2 Параметры

Ничего. footnote: [До версии 2.1 частота, амплитуда и смещение были параметрами. Они были заменены на контакты, чтобы обеспечить возможность управления другими компонентами.]

#### 5.8.7.3 Функции

- (funct) `siggen.__<chan>__.update` - Вычисляет новые значения для всех пяти выходов.

### 5.8.8 `lut5`

Компонент `lut5` представляет собой логический компонент с пятью входами, основанный на справочной таблице.

- `lut5` does not use floating point math.

**Загрузка lut5**

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

**lut5 Вычислительная функция** Чтобы вычислить шестнадцатеричное число для функции, начиная сверху, поставьте 1 или 0, чтобы указать, будет ли эта строка true или false. Затем запишите все числа в выходном столбце, начиная сверху и записывая их справа налево. Это будет двоичное число. Используя калькулятор с программным видом как, например, в Ubuntu, введите двоичное число, а затем преобразуйте его в шестнадцатеричное, и это будет значение функции.

Table 5.28: `lut5` Справочная таблица

| Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 | Выход |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 1 | |
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 0 | 1 | 1 | 1 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 1 | |
| 0 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 1 | |
| 0 | 1 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 0 | 1 | 1 | |
| 1 | 0 | 1 | 0 | 0 | |
| 1 | 0 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 1 | 0 | |
| 1 | 0 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | 1 | |
| 1 | 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | 1 | |

**lut5 Пример двух входов** В следующей таблице мы выбрали выходное состояние для каждой строки, которое мы хотим, чтобы было true.

Table 5.29: `lut5` Пример таблицы поиска с двумя входами

| Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 | Выход |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

Глядя на выходной столбец нашего примера, мы хотим, чтобы вывод был включен, когда бит 0 или бит 0 и бит 1 включены, и ничего больше. Двоичное число — *b1010* (поворот выхода на 90 градусов по часовой стрелке). Введите это число в калькулятор, затем измените отображение на шестнадцатеричное, и число, необходимое для функции, будет *0xa*. Шестнадцатеричный префикс — *0x*.

## 5.9 HAL Генератор компонентов

### 5.9.1 Введение

Этот раздел знакомит с компиляцией компонентов HAL, т. е. добавляет некоторые знания станочников о том, как обращаться с станком. Следует отметить, что такие компоненты не обязательно связаны с аппаратным обеспечением напрямую. Часто так и происходит, но не обязательно, например, может быть компонент для преобразования между британскими и метрически шкалами, поэтому в этом разделе не требуется углубляться во взаимодействие с оборудованием.

Написание компонента HAL может оказаться утомительным процессом, большая часть которого связана с вызовами функций *rtapi_* и *hal_* и связанной с ними проверкой ошибок. *halcompile* автоматически напишет за вас весь этот код. Компилировать компонент HAL также намного проще при использовании *halcompile*, независимо от того, является ли компонент частью исходного дерева LinuxCNC или вне его.

Например, при кодировании на C простой компонент, такой как ”ddt”, занимает около 80 строк кода. Эквивалентный компонент очень короткий, если написан с использованием препроцессора halcompile:

**Пример простого компонента**

```
component ddt ”Compute the derivative of the input function”;
pin in float in;
pin out float out;
variable double old;
option period no;
function _;
license ”GPL”; // indicates GPL v2 or later
;;
float tmp = in;
out = (tmp - old) / fperiod;
old = tmp;
```

### 5.9.2 Installing

Чтобы скомпилировать компонент, если используется упакованная версия LinuxCNC, пакеты разработки необходимо установить либо с помощью Synaptic из главного меню *System -> Administration -> Synaptic package manager*, либо выполнив одну из следующих команд в окне терминала. :

**Установка пакетов разработки для LinuxCNC**

```
sudo apt install linuxcnc-dev
# or
sudo apt install linuxcnc-uspace-dev
```

Другой метод — использовать менеджер пакетов Synaptic из меню «Приложения» для установки пакетов `linuxcnc-dev` или `linuxcnc-uspace-dev`.

## 5.9.3 Компиляция

### 5.9.3.1 Внутри дерева исходников

Поместите файл `.comp` в исходный каталог `linuxcnc/src/hal/components` и повторно запустите `make`. Файлы *Comp* автоматически обнаруживаются системой сборки.

Если файл `.comp` является драйвером для оборудования, его можно поместить в `linuxcnc/src/hal/dri` и он будет создан, если только LinuxCNC не настроен как симулятор не в реальном времени.

### 5.9.3.2 Компоненты реального времени вне дерева исходного кода

*halcompile* может обрабатывать, компилировать и устанавливать компонент реального времени за один шаг, помещая `rtexample.ko` в каталог модулей реального времени LinuxCNC:

```
[sudo] halcompile --install rtexample.comp
```

---

**Note**
sudo (для прав root) необходим при использовании LinuxCNC из установки пакета deb. При использовании сборки Run-In-Place (RIP) права root не требуются.

---

Или он может обработать и скомпилировать за один шаг, оставив `example.ko` (или `example.so` для симулятора) в текущем каталоге:

```
halcompile --compile rtexample.comp
```

Или он может просто обработать, оставив `example.c` в текущем каталоге:

```
halcompile rtexample.comp
```

*halcompile* также может скомпилировать и установить компонент, написанный на C, используя параметры `--install` и `--compile`, показанные выше:

```
[sudo] halcompile --install rtexample2.c
```

Документация в формате man также может быть создана на основе информации в разделе объявлений:

```
halcompile --document -o example.9 rtexample.comp
```

Полученную справочную страницу *example.9* можно просмотреть с помощью

```
man ./example.9
```

или скопировать в стандартное место для страниц руководства.

### 5.9.3.3 Компоненты не реального времени вне дерева исходного кода

*halcompile* может обрабатывать, компилировать, устанавливать и документировать компоненты, не работающие в реальном времени:

```
halcompile non-rt-example.comp
halcompile --compile non-rt-example.comp
[sudo] halcompile --install non-rt-example.comp
halcompile --document non-rt-example.comp
```

Для некоторых библиотек (например, modbus) может потребоваться добавить дополнительные аргументы компилятора и компоновщика, чтобы компилятор мог найти и связать библиотеки. В случае файлов .comp это можно сделать с помощью операторов ”option” в файле `.comp`. Для файлов `.c` это невозможно, поэтому вместо этого можно использовать параметры `--extra-compile-ar` и `--extra-link-args`. Например, эту командную строку можно использовать для компиляции компонента `vfdb_vfd.c` вне дерева.

```
halcompile --userspace --install --extra-compile-args=”-I/usr/include/modbus” --extra-link- ↩
    args=”-lm -lmodbus -llinuxcncini” vfdb_vfd.c
```

**Note**
Эффект от использования дополнительных аргументов как в командной строке, так и в файле не определен.

### 5.9.4 Использование компонента

Компоненты необходимо загрузить и добавить в поток, прежде чем его можно будет использовать. Предоставленная функциональность может затем напрямую и неоднократно вызываться одним из потоков или другими компонентами, имеющими свои собственные триггеры.

**Пример сценария HAL для установки компонента (ddt) и его выполнения каждую миллисеку**

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

Дополнительную информацию о `loadrt` и `addf` можно найти в HAL Основы.

Чтобы протестировать свой компонент, вы можете следовать примерам в HAL Учебник.

### 5.9.5 Определения

- ***component*** — компонент представляет собой один модуль реального времени, который загружается с помощью `halcmd loadrt`. Один файл `.comp` определяет один компонент. Имя компонента и имя файла должны совпадать.
- **instance** — компонент может иметь ноль или более экземпляров. Каждый экземпляр компонента создается равным (все они имеют одинаковые выводы, параметры, функции и данные), но ведут себя независимо, когда их контакты, параметры и данные имеют разные значения.
- **singleton** - компонент может быть ”singleton”, и в этом случае создается ровно один экземпляр. Редко имеет смысл писать *singleton* компонент, если только в системе не может быть буквально только один объект такого типа (например, компонент, цель которого — предоставить контакт с текущим временем UNIX или драйвер оборудования для внутреннего динамика ПК).

### 5.9.6 Создание экземпляра

Для singleton один экземпляр создается при загрузке компонента.

Для не-singleton параметр модуля *count* определяет, сколько пронумерованных экземпляров будет создано. Если *count* не указано, параметр модуля *names* определяет, сколько именованных экземпляров будет создано. Если не указано ни *количество*, ни *имена*, создается один пронумерованн экземпляр.

### 5.9.7 Неявные параметры

Функциям неявно передается параметр *period*, который представляет собой время в наносекундах последнего периода выполнения компонента. Функции, использующие числа с плавающей запятой, также могут ссылаться на *f period*, который представляет собой время с плавающей запятой в секундах, или (период*1e-9). Это может быть полезно в компонентах, которым требуется информация о времени. См. также «период опции» ниже.

### 5.9.8 Syntax

Файл `.comp` состоит из некоего количества деклараций, за которыми следует `;;` в отдельной строке, за которой следует код C, реализующий функции модуля.

Декларации включают в себя:

- *component HALNAME (DOC);*
- *pin PINDIRECTION TYPE HALNAME ([SIZE]|[MAXSIZE: CONDSIZE]) (if CONDITION) (= START-VALUE) (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;*

Круглые скобки указывают на необязательные элементы. Вертикальная черта обозначает альтернати
Слова, написанные *ЗАГЛАВНЫМИ* буквами, обозначают изменяемый текст следующим образом:

- *NAME* - Стандартный идентификатор C
- *STARREDNAME* - Идентификатор C с нулем или более * перед ним. Этот синтаксис можно использовать для объявления переменных экземпляра, которые являются указателями. Обратите внимание, что из-за грамматики между * и именем переменной может отсутствовать пробел.
- *HALNAME* - Расширенный идентификатор. При использовании для создания идентификатора HAL все подчеркивания заменяются дефисами, а любые конечные дефисы или точки удаляются, так что ”this_name_” будет преобразовано в ”this-name”, а если имя ”_”, то завершающая точка также удаляется, так что ”function _” дает имя функции HAL, например ”component.<num>” вместо ”component.<num>.”

  Если он присутствует, префикс *hal_* удаляется из начала имени компонента при создании контактов, параметров и функций.

В идентификаторе HAL для контакта или параметра # обозначает элемент массива и должен использоваться вместе с объявлением *[SIZE]*. Хэш-метки заменяются числом, дополненным 0, той же длины, что и количество символов #.

При использовании для создания идентификатора C к HALNAME применяются следующие изменения

1. Любые символы «#», а также любые символы ”.”, ”_” или ”-” непосредственно перед ними удаляются.
2. Все оставшиеся ”.” и символы ”-” заменяются на ”_”.
3. Повторяющиеся символы ”_” заменяются одним символом ”_”.

Завершающий символ ”_” сохраняется, поэтому можно использовать идентификаторы HAL, которые в противном случае могли бы конфликтовать с зарезервированными именами или ключевыми словами (например, *min*).

| HALNAME | C идентификатор | HAL Идентификатор |
|---|---|---|
| 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* — выражение, включающее переменную *personality*, которая не равна нулю, когда необходимо создать вывод или параметр.
- *SIZE* - Число, указывающее размер массива. Элементы массива пронумерованы от 0 до *SIZE*-1.
- *MAXSIZE : CONDSIZE* - Число, которое дает максимальный размер массива, за которым следует выражение, включающее переменную *personality*, значение которого всегда меньше *MAX-SIZE*. Когда массив будет создан, его размер будет *CONDSIZE*.
- *DOC* - Строка, документирующая элемент. Строка может быть строкой в стиле C с ”двойными кавычками”, например:

```
”b''Bb''b''ыb''b''бb''b''иb''b''pb''b''ab''b''eb''b''тb'' b''жb''b''eb''b''лb''b''ab''b' ↩
    'eb''b''мb''b''ыb''b''eb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b' ↩
    'ыb'' b''иb''b''мb''b''пb''b''уb''b''лb''b''ьb''b''cb''b''ab'': TRUE b''ob''b''зb''b' ↩
    'нb''b''ab''b''чb''b''ab''b''eb''b''тb'' b''cb''b''pb''b''eb''b''зb'', FALSE b''ob''b' ↩
    'зb''b''нb''b''ab''b''чb''b''ab''b''eb''b''тb'' b''фb''b''pb''b''ob''b''нb''b''тb''”
```

или строку в ”тройных кавычках” в стиле Python, которая может включать встроенные символы новой строки и кавычки, такие как:

```
”””The effect of this parameter, also known as ”the orb of zot”,
will require at least two paragraphs to explain.

Hopefully these paragraphs have allowed you to understand ”zot”
better.”””
```

Или строке может предшествовать буквальный символ *r*, и в этом случае строка интерпретируется как необработанная строка Python.

Строка документации имеет формат ”groff -man”. Дополнительную информацию об этом формате разметки см. в *groff_man(7)*. Помните, что *halcompile* интерпретирует символы обратной косой черты в строках, поэтому, например, чтобы установить курсив для слова *example*, напишите:

```
”\\fIexample\\fB”
```

В этом случае особенно полезны r-строки, поскольку обратные косые черты в r-строке не нужно удваивать:

```
r”\fIexample\fB”
```

- *TYPE* - Один из типов HAL: *bit*, *s32*, *u32*, *s64*, *u64*, или *float*. Имена *signed* и *unsigned* также могут использоваться для *s32* and *u32*, но *s32* и *u32* являются предпочтительными.
- *PINDIRECTION* - Одно из следующих: *in*, *out*, или *io*. Компонент устанавливает значение для контакта *out*, он считывает значение с контакта *in* и может считывать или устанавливать значение контакта *io*.
- *PARAMDIRECTION* - Одно из следующих: *r* или *rw*. Компонент устанавливает значение для параметра *r* и может читать или устанавливать значение параметра *rw*.
- *STARTVALUE* - Указывает начальное значение контакта или параметра. Если он не указан, то по умолчанию используется значение *0* или *FALSE*, в зависимости от типа элемента..
- *HEADERFILE* - Имя заголовочного файла в двойных кавычках (`include ”myfile.h”;`) или в угловых скобках (`include <systemfile.h>;`). Файл заголовка будет включен (с использованием #include языка C) в начало файла, перед объявлениями контактов и параметров.

#### 5.9.8.1 HAL функции

- *fp* — указывает, что функция выполняет вычисления с плавающей запятой.
- «nofp» — указывает, что он выполняет только целочисленные вычисления. Если ни один из них не указан, предполагается *fp*. Ни *halcompile*, ни gcc не могут обнаружить использование вычислений с плавающей запятой в функциях, помеченных тегом *nofp*, но использование таких операций приводит к неопределенному поведению.

#### 5.9.8.2 Варианты

В настоящее время определены следующие параметры:

- *option singleton yes* - (по умолчанию: нет)
  Не создавайте параметр модуля *count* и всегда создавайте один экземпляр. При использовании *singleton* элементы называются *component-name.item-name*, а без *singleton* элементы для пронумеров экземпляров называются *component-name.<num>.item-name*.
- *option default_count number* - (по умолчанию: 1)
  Обычно параметр модуля *count* по умолчанию равен 1. Если он указан, тогда *count* по умолчанию принимает указанное значение.
- *option count_function yes* - (по умолчанию: no)
  Обычно количество создаваемых экземпляров указывается в параметре модуля *count*; если указана *count_function*, вместо нее используется значение, возвращаемое функцией *int get_count(voi* а параметр модуля *count* не определен.
- *option rtapi_app no* - (по умолчанию: yes)
  Обычно функции `rtapi_app_main()` и `rtapi_app_exit()` определяются автоматически. При *option rtapi_app no* они отсутствуют и должны быть указаны в коде C. Используйте следующие прототипы:

```
‘int rtapi_app_main(void);‘

‘void rtapi_app_exit(void);‘
```

  При реализации собственного `rtapi_app_main()` вызовите функцию `int export(char *prefix, long extra_arg)`, чтобы зарегистрировать контакты, параметры и функции для `prefix`.

- *option data TYPE* - (по умолчанию: none) **устарело**
  Если указано, каждый экземпляр компонента будет иметь связанный блок данных типа *TYPE* (который может быть простым типом, например *float*, или именем типа, созданным с помощью *typedef*). В новых компонентах вместо этого следует использовать *variable*.

- *option extra_setup yes* - (по умолчанию: no)
  Если указано, вызовите функцию, определенную *EXTRA_SETUP*, для каждого экземпляра. При использовании автоматически определенного *rtapi_app_main extra_arg* — это номер этого экземпляр

- *option extra_cleanup yes* - (по умолчанию: no)
  Если указано, вызовите функцию, определенную *EXTRA_CLEANUP*, из автоматически определенног *rtapi_app_exit* или, в случае обнаружения ошибки, из автоматически определенного *rtapi_app_main*.

- *option userspace yes* - (по умолчанию: no)
  Если указано, этот файл описывает компонент не реального времени (ранее известный как ”userspace”), а не обычный (т. е. realtime). Компонент, не работающий в режиме реального времени, может не иметь функций, определенных директивой *function*. Вместо этого, после создания всех экземпляров, вызывается функция C `void user_mainloop(void);`. Когда эта функция возвращает значение, компонент завершает работу. Обычно *user_mainloop()* использует *FOR_ALL_INSTS()* для выполнения действия обновления для каждого экземпляра, а затем на короткое время переходит в режим сна. Другим распространенным действием в *user_mainloop()* может быть вызов цикла обработчика событий набора инструментов ГИП.

- *option userinit yes* - (по умолчанию: no)
  Эта опция игнорируется, если для опции *пространство пользователя* (см. выше) установлено значение *нет*. Если указан *userinit*, функция *userinit(argc,argv)* вызывается перед *rtapi_app_main()* (и, следовательно, перед вызовом *hal_init()*). Эта функция может обрабатывать аргументы командной строки или выполнять другие действия. Тип возвращаемого значения — *void*; он может вызвать *exit()*, если хочет завершить работу, а не создать компонент HAL (например, из-за недопустимых аргументов командной строки).

- *option extra_link_args ”...”* - (по умолчанию: ””)
  Эта опция игнорируется, если для опции *userspace* (см. выше) установлено *no*. При линковании компонента, не работающего в реальном времени, указанные аргументы вставляются в строку ссылки. Обратите внимание: поскольку компиляция происходит во временном каталоге ”-L”. относится к временному каталогу, а не к каталогу, в котором находится исходный файл .comp. Эту опцию можно установить в командной строке halcompile с помощью -extra-link-args=”-L.....”. Эта альтернатива позволяет установить дополнительные флаги в тех случаях, когда входной файл представляет собой файл .c, а не файл .comp.

- *option extra_compile_args ”...”* - (по умолчанию: ””)
  Эта опция игнорируется, если для опции *userspace* (см. выше) установлено значение *no*. При компиляции компонента, не работающего в реальном времени, указанные аргументы вставляются в командную строку компилятора. Если входной файл представляет собой файл .c, этот параметр можно установить в командной строке halcompile с помощью --extra-compile-args=”-I.....”. Эта альтернатива позволяет установить дополнительные флаги в тех случаях, когда входной файл представляет собой файл .c, а не файл .comp.

- *option homemod yes* - (по умолчанию: no)
  Модуль — это пользовательский модуль Homing, загружаемый с помощью `[EMCMOT]HOMEMOD=`*modulen*
  .

- *option tpmod yes* - (по умолчанию: no)
  Модуль — это пользовательский модуль Trajectory Planning (tp), загружаемый с помощью `[TRAJ]TPMO`
  .

- *option period no* - (default: yes)
  Control the implicit *period* parameter of the function(s) defined in the component. A standard function has an implicit parameter *period*. Many components do no use the *period* parameter and would cause a ”unused parameter” compiler warning. Setting *option period no* creates a function

declaration omitting the *period* parameter preventing the warning. Setting this option will also prevent *fperiod* from being defined, as it depends on *period*.

Если VALUE опции не указано, это эквивалентно указанию *option ... yes*.
Результат присвоения опции неподходящего значения не определен.
Результат использования любой другой опции не определен.

#### 5.9.8.3 Лицензия и авторство

- `LICENSE` — укажите лицензию модуля для документации и для объявления модуля MODULE_LICENSE
  Например, чтобы указать, что лицензия модуля — GPL v2 или новее:

  ```
  ‘license ”GPL”; // indicates GPL v2 or later‘
  ```

  Дополнительную информацию о значении MODULE_LICENSE() и дополнительных идентификаторах лицензий см. в разделе «<linux/module.h>» или на странице руководства «rtapi_module_param(3)».

  Это заявление **обязательно**.

- `AUTHOR` - Укажите автора модуля для документации.

#### 5.9.8.4 Поэкземплярное хранилище данных

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARRED = DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Объявите переменную *STARREDNAME* для каждого экземпляра типа *CTYPE*, при необходимости как массив элементов *SIZE* и при необходимости со значением по умолчанию *DEFAULT*. Элементы без *DEFAULT* инициализируются нулевыми битами. *CTYPE* — это простой тип C, состоящий из одного слова, такой как `float`, `u32`, `s32`, `int` и т. д. Для доступа к переменным массива используются квадратные скобки.

Если переменная должна иметь тип указателя, между знаком ”*” и именем переменной не может быть пробела. Поэтому допустимо следующее:

```
variable int *example;
```

А это не допустимо:

```
variable int* badexample;
variable int * badexample;
```

#### 5.9.8.5 Comments

В разделе объявлений поддерживаются однострочные комментарии в стиле C++ (`//...`) и многострочные комментарии в стиле C (`/* ... */`).

### 5.9.9 Ограничения

Хотя HAL позволяет контакту, параметру и функции иметь одно и то же имя, «halcompile» этого не делает.

Имена переменных и функций, которые нельзя использовать или которые могут вызвать проблемы, включают:

- Все, что начинается с _comp.
- *comp_id*
- *fperiod*
- *rtapi_app_main*
- *rtapi_app_exit*
- *extra_setup*
- *extra_cleanup*

### 5.9.10 Удобные макросы

На основе элементов раздела объявлений halcompile создает структуру C с именем `struct __comp_sta`
Однако вместо обращения к членам этой структуры (например, `*(inst->name)`), к ним обычно будут обращаться с помощью приведенных ниже макросов. Детали `struct __comp_state` и этих макросов могут меняться от одной версии 'halcompile'к другой.

- `FUNCTION( `__name__)`` — используйте этот макрос, чтобы начать определение функции реального времени, которая ранее была объявлена с помощью *function NAME*. Функция включает параметр *period*, который представляет собой целое число наносекунд между вызовами функции. См. также «период опции» выше.
- `EXTRA_SETUP()` — используйте этот макрос, чтобы начать определение функции, вызываемой для выполнения дополнительной настройки этого экземпляра. Возвращает отрицательное значение UNIX *errno*, чтобы указать на ошибку (например, *return -EBUSY*, если не удалось зарезервировать порт ввода-вывода), или 0, чтобы указать на успех.
- `EXTRA_CLEANUP()` — используйте этот макрос, чтобы начать определение функции, вызываемой для дополнительной очистки компонента. Обратите внимание, что эта функция должна очищать все экземпляры компонента, а не только один. Макросы «pin_name», «parameter_name» и «data» здесь использовать нельзя.
- *pin_name* или *parameter_name* — для каждого контакта *имя_контакта* или параметра *имя_парамет* существует макрос, который позволяет использовать имя отдельно для ссылки на контакт или параметр. Если *pin_name* или *parameter_name* представляет собой массив, макрос имеет форму *pin_name(idx)* или *param_name(idx)*, где *idx* — это индекс в массиве контактов. Когда массив представляет собой массив переменного размера, разрешено ссылаться только на элементы до его *condsize*.

  Если элемент является условным элементом, ссылаться на него можно только в том случае, если его *условие* оценено как ненулевое значение.
- *variable_name* — для каждой переменной *variable_name* существует макрос, который позволяет использовать имя отдельно для ссылки на переменную. Когда *variable_name* представляет собой массив, используется обычный индекс в стиле C: *variable_name[idx]*.
- *data* — если указаны "option data", этот макрос разрешает доступ к данным экземпляра.
- *fperiod* — число секунд с плавающей запятой между вызовами этой функции реального времени. См. также «период опции» выше.
- `FOR_ALL_INSTS() {...}` — для компонентов, не работающих в режиме реального времени. Этот макрос перебирает все определенные экземпляры. Внутри тела цикла макросы *pin_name*, *parameter_name* и *data* работают так же, как и в функциях реального времени.

### 5.9.11 Компоненты с одной функцией

Если компонент имеет только одну функцию и строка "FUNCTION" не появляется нигде после ;;, то часть после ;; все это считается телом единственной функции компонента. Пример этого см. в Simple Comp.

### 5.9.12 Индивидуальность компонента

Если компонент имеет какие-либо контакты или параметры с условием "if" или "[maxsize : cond-size]", он называется компонентом с "индивидуальностью". *Индивидуальность* каждого экземпляра определяется при загрузке модуля. *Индивидуальность* можно использовать для создания контактов только при необходимости. Например, в *логическом* компоненте используется индивидуальность, чтобы обеспечить переменное количество входных контактов для каждого логического элемента и обеспечить выбор любой из основных логических функций *и*, *или* и *хог*.

Число разрешенных элементов *индивидуальности* по умолчанию устанавливается во время компиляц (64). Значение по умолчанию применяется к многочисленным компонентам, включенным в дистрибутив, созданным с использованием halcompile.

Чтобы изменить разрешенное количество элементов индивидуальности для пользовательских компонентов, используйте опцию --personality с halcompile. Например, чтобы разрешить до 128 индивидуальных раз:

```
  [sudo] halcompile --personalities=128 --install ...
```

При использовании компонентов с индивидуальностью обычно указывается элемент индивидуальност для **каждого** указанного экземпляра компонента. Пример для 3 экземпляров логического компонент

```
loadrt logic names=and4,or3,nand5, personality=0x104,0x203,0x805
```

---

**Note**
Если в строке loadrt указано больше экземпляров, чем индивидуальностей, экземплярам с неуказанными индивидуальностями присваивается индивидуальность 0. Если запрошенное количество экземпляров превышает количество разрешенных индивидуальностей, индивидуальность назначаются путем индексации по модулю количества разрешенных индивидуальностей. Печатается сообщение, обозначающее такие назначения.

---

### 5.9.13 Examples

#### 5.9.13.1 constant

Обратите внимание, что объявление "function _" создает функции с именем "constant.0" и т. д. Имя файла должно совпадать с именем компонента.

```
component constant;
pin out float out;
param r float value = 1.0;
option period no;
function _;
license "GPL"; // indicates GPL v2 or later
;;
FUNCTION(_) { out = value; }
```

### 5.9.13.2 sincos

Этот компонент вычисляет синус и косинус входного угла в радианах. Он имеет другие возможности, чем выходы ”sine” и ”cosine” siggen, потому что вход представляет собой угол, а не работает свободно на основе параметра ”frequency”.

В исходном коде выводы объявлены с именами *sin_* и *cos_*, чтобы они не мешали функциям *sin()* и *cos()*. Контакты HAL по-прежнему называются *sincos.<num>.sin*.

```
component sincos;
pin out float sin_;
pin out float cos_;
pin in float theta;
option period no;
function _;
license ”GPL”; // indicates GPL v2 or later
;;
#include <rtapi_math.h>
FUNCTION(_) { sin_ = sin(theta); cos_ = cos(theta); }
```

### 5.9.13.3 out8

Этот компонент является драйвером *вымышленной* карты под названием *out8*, которая имеет 8 контактов цифрового выхода, которые обрабатываются как одно 8-битное значение. Таких карт в системе может быть разное количество и они могут находиться по разным адресам. Вывод называется *out_*, поскольку *out* — это идентификатор, используемый в *<asm/io.h>*. Он иллюстрирует использование *EXTRA_SETUP* и *EXTRA_CLEANUP* для запроса области ввода-вывода и последующего ее освобождения в случае ошибки или при выгрузке модуля.

```
component out8;
pin out unsigned out_ ”Output value; only low 8 bits are used”;
param r unsigned ioaddr;

function _;

option period no;
option count_function;
option extra_setup;
option extra_cleanup;
option constructable no;

license ”GPL”; // indicates GPL v2 or later
;;
#include <asm/io.h>

#define MAX 8
int io[MAX] = {0,};
RTAPI_MP_ARRAY_INT(io, MAX, ”I/O addresses of out8 boards”);

int get_count(void) {
    int i = 0;
    for(i=0; i<MAX && io[i]; i++) { /* Nothing */ }
    return i;
}

EXTRA_SETUP() {
    if(!rtapi_request_region(io[extra_arg], 1, ”out8”)) {
        // set this I/O port to 0 so that EXTRA_CLEANUP does not release the IO
        // ports that were never requested.
        io[extra_arg] = 0;
```

```
        return -EBUSY;
    }
    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); }
```

#### 5.9.13.4 hal_loop

```
component hal_loop;
pin out float example;
```

Этот фрагмент компонента иллюстрирует использование префикса *hal_* в имени компонента.

`loop` — это общее имя, а префикс `hal_` позволяет избежать потенциальных конфликтов имен с другим несвязанным программным обеспечением. Например, в системах реального времени RTAI код реального времени выполняется в ядре, поэтому, если бы компонент назывался просто `loop`, он мог бы легко конфликтовать со стандартным модулем ядра `loop`.

При загрузке *halcmd show comp* покажет компонент под названием *hal_loop*. Однако вывод, отображаемый *halcmd show pin*, будет *loop.0.example*, а не *hal-loop.0.example*.

#### 5.9.13.5 arraydemo

Этот компонент реального времени иллюстрирует использование массивов фиксированного размера:

```
component arraydemo "4-bit Shift register";
pin in bit in;
pin out bit out-# [4];
option period no;
function _ nofp;
license "GPL"; // indicates GPL v2 or later
;;
int i;
for(i=3; i>0; i--) out(i) = out(i-1);
out(0) = in;
```

#### 5.9.13.6 rand

Этот компонент, работающий не в реальном времени, меняет значение на своем выходном контакте на новое случайное значение в диапазоне (0,1) примерно раз в 1 мс.

```
component rand;
option userspace;

pin out float out;
license "GPL"; // indicates GPL v2 or later
;;
#include <unistd.h>
```

```
void user_mainloop(void) {
    while(1) {
        usleep(1000);
        FOR_ALL_INSTS() out = drand48();
    }
}
```

### 5.9.13.7 logic (using personality)

Этот компонент реального времени показывает, как использовать ”индивидуальность” для создания массивов переменного размера и дополнительных выводов.

```
component logic ”LinuxCNC HAL component providing experimental logic functions”;
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;
option period no;
function _ nofp;
description """
Experimental general 'logic function' component.  Can perform 'and', 'or'
and 'xor' of up to 16 inputs.  Determine the proper value for 'personality'
by adding:
.IP \\(bu 4
The number of input pins, usually from 2 to 16
.IP \\(bu
256 (0x100)  if the 'and' output is desired
.IP \\(bu
512 (0x200)  if the 'or' output is desired
.IP \\(bu
1024 (0x400)  if the 'xor' (exclusive or) output is desired""";
license ”GPL”; // indicates GPL v2 or later
;;
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;
}
```

Типичная линия нагрузки для этого компонента может быть такой

```
loadrt logic count=3 personality=0x102,0x305,0x503
```

который создает следующие контакты:

- Элемент *И* с двумя входами: `logic.0.and`, `logic.0.in-00`, `logic.0.in-01`
- Элемент *И* с 5-ю входами и элементы *ИЛИ*: `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`,
- Элементы *И* с тремя входами и элементы *Исключающее ИЛИ*: `logic.2.and`, `logic.2.xor`, `logic.2.in` `logic.2.in-01`, `logic.2.in-02`

#### 5.9.13.8 Общие функции

В этом примере показано, как вызывать функции из основной функции. Он также показывает, как передать ссылку на контакты HAL этим функциям.

```
component example;
pin in s32 in;
pin out bit out1;
pin out bit out2;

option period no;
function _;
license "GPL";
;;

// general pin set true function
void set(hal_bit_t *p){
    *p = 1;
}

// general pin set false function
void unset(hal_bit_t *p){
    *p = 0;
}

//main function
FUNCTION(_) {
    if (in < 0){
        set(&out1);
        unset(&out2);
    }else if (in >0){
        unset(&out2);
        set(&out2);
    }else{
        unset(&out1);
        unset(&out2);
    }
}
```

Этот компонент использует две общие функции для управления связанным с ним битом контакта HAL.

### 5.9.14 Использование командной строки

Страница руководства halcompile содержит подробную информацию о вызове `halcompile`.

```
$ man halcompile
```

Краткое описание использования halcompile представлено следующим образом:

```
$ halcompile --help
```

## 5.10 HALTCL Files

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.

The haltcl facility provides a means to use Tcl scripting and its features for computation, looping, branching, procedures, etc. in INI files. To use this functionality, you use the Tcl language and the extension .tcl for HAL files.

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).

**Пример**

```
[HAL]
HALFILE = conventional_file.hal
HALFILE = tcl_based_file.tcl
```

With appropriate care, HAL and Tcl files can be intermixed.

### 5.10.1 Compatibility

The halcmd language used in HAL files has a simple syntax that is actually a subset of the more powerful general-purpose Tcl scripting language.

### 5.10.2 Haltcl Commands

Haltcl files use the Tcl scripting language augmented with the specific commands of the LinuxCNC hardware abstraction layer (HAL). The HAL-specific commands are:

```
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
```

Two special cases occur for the *gets* and *list* commands due to conflicts with Tcl builtin commands. For haltcl, these commands must be preceded with the keyword *hal*:

```
halcmd   haltcl
------   ------
gets     hal gets
list     hal list
```

### 5.10.3 Haltcl INI-file variables

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]
...
```

The INI file values are accessible by text substitution in HAL files using the form:

```
[SECTION]ITEM
```

The same INI file values are accessible in Tcl files using the form of a Tcl global array variable:

```
$::SECTION(ITEM)
```

For example, an INI file item like:

```
[JOINT_0]
MAX_VELOCITY = 4
```

is expressed as `[JOINT_0]MAX_VELOCITY` in HAL files for halcmd
and as `$::JOINT_0(MAX_VELOCITY)` in Tcl files for haltcl.

Because INI files can repeat the same ITEM in the same SECTION multiple times, `$::SECTION(ITEM)` is actually a Tcl list of each individual value.

When there is just one value and it is a simple value (all values that are just letters and numbers without whitespace are in this group), then it is possible to treat `$::SECTION(ITEM)` as though it is not a list.

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, this is written `[lindex $::SECTION(ITEM) 0]`.

For example: given the following INI values

```
[HOSTMOT2]
DRIVER=hm2_eth
IPADDR=”10.10.10.10”
BOARD=7i92
CONFIG=”num_encoders=0 num_pwmgens=0 num_stepgens=6”
```

And this loadrt command:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=$::HOSTMOT2(IPADDR) config=$::HOSTMOT2(CONFIG)
```

Here is the actual command that is run:

```
loadrt hm2_eth board_ip={”10.10.10.10”} config={”num_encoders=0 num_pwmgens=0 num_stepgens ↩
    =6”}
```

This fails because loadrt does not recognize the braces.

So to get the values just as entered in the INI file, re-write the loadrt line like this:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=[lindex $::HOSTMOT2(IPADDR) 0] config=[lindex  ↩
    $::HOSTMOT2(CONFIG) 0]
```

### 5.10.4 Converting HAL files to Tcl files

Existing HAL files can be converted to Tcl files by hand editing to adapt to the differences mentioned above. The process can be automated with scripts that convert using these substitutions.

```
[SECTION]ITEM ---> $::SECTION(ITEM)
gets          ---> hal gets
list          ---> hal list
```

### 5.10.5 Haltcl Notes

In haltcl, the value argument for the *sets* and *setp* commands is implicitly treated as an expression in the Tcl language.

**Пример**

```
# set gain to convert deg/sec to units/min for JOINT_0 radius
setp scale.0.gain 6.28/360.0*$::JOINT_0(radius)*60.0
```

Whitespace in the bare expression is not allowed, use quotes for that:

```
setp scale.0.gain "6.28 / 360.0 * $::JOINT_0(radius) * 60.0"
```

In other contexts, such as *loadrt*, you must explicitly use the Tcl expr command ([expr {}]) for computational expressions.

**Пример**

```
loadrt motion base_period=[expr {500000000/$::TRAJ(MAX_PULSE_RATE)}]
```

### 5.10.6 Haltcl Examples

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.

```
# Create position, velocity and acceleration signals for each axis
set ddt 0
for {set jnum 0} {$jnum < $::KINS(JOINTS)} {incr jnum} {
  # 'list pin' returns an empty list if the pin doesn't exist
  if {[hal list pin joint.${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 Interactive

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.

**Пример**

```
$ halrun -T haltclfile.tcl
```

### 5.10.8 Haltcl Distribution Examples (sim)

The configs/sim/axis/simtcl directory includes an INI file that uses a .tcl file to demonstrate a haltcl configuration in conjunction with the usage of twopass processing. The example shows the use of Tcl procedures, looping, the use of comments and output to the terminal.

## 5.11 HAL User Interface

### 5.11.1 Введение

Halui — это пользовательский интерфейс для LinuxCNC на основе HAL, он соединяет контакты HAL с командами NML. Большая часть функциональности (кнопки, индикаторы и т. д.), предоставляем традиционным ГИП (AXIS, GMOCCAPY, QtDragon и т. д.), обеспечивается контактами HAL в Halui.

Самый простой способ добавить halui — добавить следующее в раздел [HAL] INI-файла:

```
[HAL]
HALUI = halui
```

Альтернативный способ его вызова (особенно если вы создаете конфигурацию с помощью Step-Conf) — включить следующее в ваш файл `custom.hal`.
Убедитесь, что вы используете правильный путь к вашему INI-файлу.

```
loadusr halui -ini /path/to/inifile.ini
```

### 5.11.2 MDI

Иногда пользователь хочет добавить более сложные задачи, которые будут выполняться путем активации контакта HAL. Это возможно путем добавления команд MDI в INI-файл в разделе [HALUI]. Пример:

```
[HALUI]
MDI_COMMAND = G0 X0
MDI_COMMAND = G0 G53 Z0
MDI_COMMAND = G28
MDI_COMMAND = o<mysub>call
...
```

Когда halui запускается, он считывает поля MDI_COMMAND в INI и экспортирует контакты от 00 до количества MDI_COMMAND, найденного в INI, и до максимального количества 64 команд. Эти контакты можно подключать как любые контакты HAL. Распространенный метод — использовать кнопки, предоставляемые виртуальными панелями управления, как показано в примере Пример для соединений MDI_COMMAND.

**Example 5.1** Пример соединений MDI_COMMAND

**HAL-файл**

```
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
```

**Сети, соединяющие контакты halui.mdi-command-NN, предоставленные halui.**

```
$ 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
    ...
```

Когда на выводе halui MDI установлено (импульсное) значение true, halui отправит команду MDI, определенную в INI. Это не всегда будет успешным в зависимости от текущего режима работы (например, в режиме AUTO halui не может успешно отправлять команды MDI).

### 5.11.3 Пример конфигурации

Пример конфигурации sim (`configs/sim/axis/halui_pyvcp/halui.ini`) включен в дистрибутив.

### 5.11.4 Справочник по контактам Halui

Все контакты halui также описаны на странице руководства halui:

```
$ man halui
```

Or see http://linuxcnc.org/docs/devel/html/man/man1/halui.1.html

#### 5.11.4.1 Прерывание

- *halui.abort* (bit, in) - контакт для отправки сообщения об прерывании (очищает большинство ошибок)

#### 5.11.4.2 E-Stop

- *halui.estop.activate* (bit, in) - контакт для запроса аварийного останова
- *halui.estop.is-activated* (bit, out) - отображает сброс аварийного останова
- *halui.estop.reset* (bit, in) - контакт для запроса сброса аварийного останова

#### 5.11.4.3 Переопределение подачи

- *halui.feed-override.count-enable* (bit, in) - должно быть true, чтобы работали *counts* или *direct-value*.
- *halui.feed-override.counts* (s32, in) - counts * scale = процент FO. Может использоваться с энкодером или ’direct-value.

- *halui.feed-override.decrease* (bit, in) - контакт для уменьшения FO (-=scale)
- *halui.feed-override.increase* (bit, in) - контакт для увеличения ФО (+=scale)
- *halui.feed-override.reset* (bit, in) - вывод для сброса FO (scale=1,0)
- *halui.feed-override.direct-value* (bit, in) - false при использовании энкодера для изменения отсчетов, true для непосредственного изменения отсчетов.
- *halui.feed-override.scale* (float, in) - контакт для установки масштабирования увеличения и уменьшен *feed-override*.
- *halui.feed-override.value* (float, out) - текущее значение FO

#### 5.11.4.4 Туман

- *halui.mist.is-on* (bit, out) - показывает, что туман включен
- *halui.mist.off* (bit, in) - контакт для запроса отключения тумана
- *halui.mist.on* (bit, in) - контакт для запроса включения тумана

#### 5.11.4.5 СОЖ

- *halui.flood.is-on* (bit, out) - показывает, что СОЖ включена
- *halui.flood.off* (bit, in) - контакт для запроса отключения СОЖ
- *halui.flood.on* (bit, in) - контакт для запроса отключения СОЖ

#### 5.11.4.6 Homing

- *halui.home-all* (bit, in) - контакт для запроса возврата в исходное положение всех осей. Этот контакт будет доступен только в том случае, если в INI-файле установлено значение HOME_SEQUEN

#### 5.11.4.7 Станок

- *halui.machine.units-per-mm* (float out) - контакт для единиц станка на мм (дюйм:1/25,4, мм:1) в соответствии с настройкой inifile: [TRAJ]LINEAR_UNITS
- *halui.machine.is-on* (bit, out) - показывает, что станок включен
- *halui.machine.off* (bit, in) - контакт для запроса выключения станка
- *halui.machine.on* (bit, in) - контакт для запроса включения станка

#### 5.11.4.8 Макс скорость

Максимальную линейную скорость можно регулировать от 0 до MAX_VELOCITY, установленной в разделе [TRAJ] INI-файла.

- *halui.max-velocity.count-enable* (bit, in) - должно быть true , чтобы *counts* или *direct-value* работали.
- *halui.max-velocity.counts* (s32, in) - counts * scale = процент MV. Может использоваться с энкодером или *direct-value*.

- *halui.max-velocity.direct-value* (bit, in) - false при использовании энкодера для изменения отсчетов, true для непосредственного изменения отсчетов.
- *halui.max-velocity.decrease* (bit, in) - контакт для уменьшения максимальной скорости
- *halui.max-velocity.increase* (bit, in) - контакт для увеличения максимальной скорости
- *halui.max-velocity.scale* (float, in) - величина, применяемая к текущей максимальной скорости при каждом переходе от выключения к включению контакта увеличения или уменьшения в единицах станках в секунду.
- *halui.max-velocity.value* (float, out) - максимальная линейная скорость в единицах станка в секунду.

#### 5.11.4.9 MDI

- *halui.mdi-command*-<nn> (bit, in) - halui попытается отправить команду MDI, определенную в INI. <*nn*> — двузначное число, начинающееся с 00.
  Если команда выполнена успешно, LinuxCNC перейдет в режим MDI, а затем вернется в ручной режим.
  Если в ini-файле не заданы переменные [HALUI]MDI_COMMAND, halui не будет экспортировать контакты halui.mdi-command-<nn>.
- *halui.halui-mdi-is-running* (bit, out) - execution status of MDI commands sent by halui. The status is active even during mode switching. If no [HALUI]MDI_COMMAND variables are set in the ini file, this pins will not be exported by halui.

#### 5.11.4.10 Сочленение

*N* = joint number (0 … num_joints-1)
Пример:

- *halui.joint.*N.*select* (bit in) - контакт для выбора сочленения *N*
- *halui.joint.*N.*is-selected* (bit out) - контакт состояния, что выбрано сочленение *N*
- *halui.joint.*N.*has-fault* (bit out) - контакт состояния, сообщающий о неисправности сочленения *N*
- *halui.joint.*N.*home* (bit in) - контакт для приведения в исходное положение сочленения *N*
- *halui.joint.*N.*is-homed* (bit out) - контакт состояния, сообщающий, что сочленение *N* возвращено в исходное положение
- *halui.joint.*N.*on-hard-max-limit* (bit out) - контакт состояния, сообщающий, что сочленение *N* находится на положительном аппаратном пределе
- *halui.joint.*N.*on-hard-min-limit* (bit out) - контакт состояния, сообщающий, что сочленение *N* находится на отрицательном аппаратном пределе
- *halui.joint.*N.*on-soft-max-limit* (bit out) - контакт состояния, сообщающий, что сочленение *N* находится на положительном программном пределе
- *halui.joint.*N.*on-soft-min-limit* (bit out) - контакт состояния, сообщающий, что сочленение *N* находится на отрицательном программном пределе
- *halui.joint.*N.*override-limits* (bit out) - контакт состояния, сообщающий, что ограничения сочленения *N* временно переназначены
- *halui.joint.*N.*unhome* (bit in) - контакт для вывода из исходного состояния сочленения *N*

- *halui.joint.selected* (u32 out) - номер выбранного сочленения (0 … num_joints-1)
- *halui.joint.selected.has-fault* (bit out) - контакт состояния, что выбранное сочленение неисправно
- *halui.joint.selected.home* (bit in) - контакт для возврата в исходное положение выбранного сочленения
- *halui.joint.selected.is-homed* (bit out) - контакт состояния, сообщающий, что выбранное сочленение в исходном положении
- *halui.joint.selected.on-hard-max-limit* (bit out) - контакт состояния, сообщающий, что выбранное сочленение находится на положительном аппаратном пределе
- *halui.joint.selected.on-hard-min-limit* (bit out) - контакт состояния, сообщающий, что выбранное сочленение находится на отрицательном аппаратном пределе
- *halui.joint.selected.on-soft-max-limit* (bit out) - контакт состояния, сообщающий, что выбранное сочленение находится на положительном программном пределе
- *halui.joint.selected.on-soft-min-limit* (bit out) - контакт состояния, сообщающий, что выбранное сочленение находится на отрицательном программном пределе
- *halui.joint.selected.override-limits* (bit out) - контакт состояния, сообщающий, что ограничения выбранного сочленения временно переназначены
- *halui.joint.selected.unhome* (bit in) - контакт для вывода из исходного положения выбранного сочленения

#### 5.11.4.11 Медленная подача сочленения

*N* = joint number (0 … num_joints-1)

- *halui.joint.jog-deadband* (float in) - конаткт для установки зоны аналоговой нечувствительности медленной подачи (аналоговые значения медленной подачи, меньше/медленнее этого - в абсолютно значении - игнорируются)
- *halui.joint.jog-speed* (float in) - контакт для настройки скорости медленной подачи для медленной подачи плюс/минус.
- *halui.joint.*N.*analog* (float in) - контакт для медленной подачи сочленения *N* с использованием значения с плавающей запятой (например, джойстика). Значение, обычно устанавливаемое в диапазоне от 0,0 до ±1,0, используется в качестве множителя скорости медленной подачи.
- *halui.joint.*N.*increment* (float in) - контакт для установки инкремента медленной подачи для сочленения *N* при использовании приращения-плюс/минус
- *halui.joint.*N.*increment-minus* (bit in) - фронт заставит сочленение *N* двигаться в отрицательном направлении на величину приращения
- *halui.joint.*N.*increment-plus* (bit in) - фронт заставит сочленение *N* медленно перемещаться в положительном направлении на величину приращения
- *halui.joint.*N.*minus* (bit in) - контакт для медленной подачи сочленения *N* в отрицательном направлении со скоростью halui.joint.jog-speed
- *halui.joint.*N.*plus* (bit in) - контакт для медленной подачи сочленения *N* в положительном направлени со скоростью halui.joint.jog-speed
- *halui.joint.selected.increment* (float in) - контакт для установки инкремента медленной подачи для выбранного сочленения при использовании приращения плюс/минус
- *halui.joint.selected.increment-minus* (bit in) - фронт приведет к медленной подаче выбранного сочленения в отрицательном направлении на величину приращения

- *halui.joint.selected.increment-plus* (bit in) - фронт приведет к медленной подаче выбранного сочленения в положительном направлении на величину приращения
- *halui.joint.selected.minus* (bit in) - контакт для медленной подачи выбранного сочленения в отрицательном направлении со скоростью halui.joint.jog-speed
- *halui.joint.selected.plus* (bit in) - контакт для медленной подачи выбранного сочленения в положитель направлении со скоростью halui.joint.jog-speed

#### 5.11.4.12 Axis

*L* = буква оси (xyzabcuvw)

- *halui.axis*.L.*select* (bit) - контакт для выбора оси по букве
- *halui.axis*.L.*is-selected* (bit out) - контакт состояния, что выбрана ось *L*
- *halui.axis*.L.*pos-commanded* (float out) - Заданное положение оси в координатах станка
- *halui.axis*.L.*pos-feedback* float out) - Обратная связь положения оси в координатах станка
- *halui.axis*.L.*pos-relative* (float out) - Обратная связь положения оси в относительных координатах

#### 5.11.4.13 Медленная подача оси

*L* = буква оси (xyzabcuvw)

- *halui.axis.jog-deadband* (float in) - контакт для установки зоны аналоговой нечувствительности медленной подачи (аналоговые входы медленной подачи, меньшие/медленнее этого значения (в абсолютном значении), игнорируются)
- *halui.axis.jog-speed* (float in) - контакт для настройки скорости медленной подачи для медленной подачи плюс/минус.
- *halui.axis*.L.*analog* (float in) - контакт для медленной подачи оси *L* с использованием плавающего значения (например, джойстика). Значение, обычно устанавливаемое в диапазоне от 0,0 до ±1,0, используется в качестве множителя скорости медленной подачи.
- *halui.axis*.L.*increment* (float in) - контакт для установки инкремента медленной подачи для оси *L* при использовании приращения плюс/минус
- *halui.axis*.L.*increment-minus* (bit in) - по фронту ось *L* будет медленно подаваться в отрицательном направлении на величину приращения
- *halui.axis*.L.*increment-plus* (bit in) - по фронту будет осуществляться медленная подача оси *L* в положительном направлении на величину приращения
- *halui.axis*.L.*minus* (bit in) - контакт для медленной подачи оси *L* в отрицательном направлении со скоростью halui.axis.jog-speed
- *halui.axis*.L.*plus* (bit in) - контакт для медленной подачи оси *L* в положительном направлении со скоростью halui.axis.jog-speed
- *halui.axis.selected* (u32 out) - выбранная ось (по индексу: 0:x 1:y 2:z 3:a 4:b 5:cr 6:u 7:v 8:w)
- *halui.axis.selected.increment* (float in) - контакт для установки инкремента медленной подачи для выбранной оси при использовании приращения - плюс/минус
- *halui.axis.selected.increment-minus* (bit in) - по фронту выбранная ось будет медленно подаваться в отрицательном направлении на величину приращения

- *halui.axis.selected.increment-plus* (bit in) - по фронту выбранная ось будет медленно подаваться в положительном направлении на величину приращения
- *halui.axis.selected.minus* (bit in) - контакт для медленной подачи выбранной оси в отрицательном направлении со скоростью halui.axis.jog-speed
- *halui.axis.selected.plus* (pin in) - для медленной подачи выбранной оси в положительном направлении со скоростью halui.axis.jog-speed

#### 5.11.4.14 Mode

- *halui.mode.auto* (bit, in) - контакт для запроса автоматического режима
- *halui.mode.is-auto* (bit, out) - указывает на то, что автоматический режим включен
- *halui.mode.is-joint* (bit, out) - указывает сочленение по включенному режиму медленной подачи сочленения
- *halui.mode.is-manual* (bit, out) - указывает на то, что ручной режим включен
- *halui.mode.is-mdi* (bit, out) - указывает на то, что режим MDI включен
- *halui.mode.is-teleop* (bit, out) - указывает на то, что включен режим медленной подачи
- *halui.mode.joint* (bit, in) - контакт для запроса режима совместной работы сочленения
- *halui.mode.manual* (bit, in) - контакт для запроса ручного режима
- *halui.mode.mdi* (bit, in) - контакт для запроса режима MDI
- *halui.mode.teleop* (bit, in) - контакт для запроса режима координированной медленной подачи

#### 5.11.4.15 Программа

- *halui.program.block-delete.is-on* (bit, out) - контакт состояния, сообщающий, что удаление блока включено
- *halui.program.block-delete.off* (bit, in) - контакт для запроса удаления блока отключен
- *halui.program.block-delete.on* (bit, in) - контакт для запроса удаления блока включен
- *halui.program.is-idle* (bit, out) - контакт состояния, сообщающий, что ни одна программа не запущена
- *halui.program.is-paused* (bit, out) - контакт состояния, сообщающий о том, что программа приостанов
- *halui.program.is-running* (bit, out) - контакт состояния, сообщающий о том, что программа запущена
- *halui.program.optional-stop.is-on* (bit, out) - контакт состояния, сообщающий о том, что опциональная остановка включена
- *halui.program.optional-stop.off* (bit, in) - контакт, запрашивающий отключение опциональной остановки
- *halui.program.optional-stop.on* (bit, in) - контакт запроса на включение опциональнаой остановки
- *halui.program.pause* (bit, in) - контакт для приостановки программы
- *halui.program.resume* (bit, in) - контакт для возобновления приостановленной программы
- *halui.program.run* (bit, in) - контакт для запуска программы
- *halui.program.step* (bit, in) - контакт для шага в программе
- *halui.program.stop* (bit, in) - контакт для остановки программы

#### 5.11.4.16 Быстрое переназначение

- *halui.rapid-override.count-enable* (bit in (default: TRUE)) - Когда TRUE, изменить Rapid Override при изменении счетчика.
- *halui.rapid-override.counts* (s32 in) - отсчитывает X коэффициент масштаба = процент Rapid Override. Может использоваться с энкодером или *direct-value*.
- *halui.rapid-override.decrease* (bit in) - контакт для уменьшения быстрого переопределения (-=масштаб)
- *halui.rapid-override.direct-value* (bit in) - контакт для разрешения ввода напрямую значения Rapid Override
- *halui.rapid-override.increase* (bit in) - контакт для увеличения быстрого переопределения (+=масштаб
- *halui.rapid-override.scale* (float in) - контакт для установки масштаба при изменении быстрого переопределения
- *halui.rapid-override.value* (float out) - текущее значение быстрого переопределения
- *halui.rapid-override.reset* (bit, in) - контакт для сброса значения быстрого переопределения (шкала=1,0)

#### 5.11.4.17 Переопределение шпинделя

- *halui.spindle*.N.*override.count-enable* (bit, in) - должно быть true, чтобы *counts* или *direct-value* работали.
- *halui.spindle*.N.*override.counts* (s32, in) - counts * scale = процент SO. Может использоваться с энкодером или *direct-value*.
- *halui.spindle*.N.*override.decrease* (bit, in) - контакт для уменьшения SO (-=шкала)
- *halui.spindle*.N.*override.direct-value* (bit, in) - false при использовании энкодера для изменения отсчетов, true при непосредственной установке отсчетов.
- *halui.spindle*.N.*override.increase* (bit, in) - контакт для увеличения SO (+=scale)
- *halui.spindle*.N.*override.scale* (float, in) - контакт для настройки шкалы при изменении SO
- *halui.spindle*.N.*override.value* (float, out) - текущее значение SO
- *halui.spindle*.N.*override.reset* (bit, in) - контакт для сброса значения SO (масштаб=1,0)

#### 5.11.4.18 Spindle

- *halui.spindle*.N.*brake-is-on* (bit, out) - указывает на то, что тормоз включен
- *halui.spindle*.N.*brake-off* (bit, in) - контакт для отключения шпинделя/тормоза
- *halui.spindle*.N.*brake-on* (bit, in) - контакт для активации тормоза шпинделя
- *halui.spindle*.N.*decrease* (bit, in) - уменьшает скорость шпинделя
- *halui.spindle*.N.*forward* (bit, in) - запускает шпиндель по часовой стрелке
- *halui.spindle*.N.*increase* (bit, in)- увеличивает скорость шпинделя
- *halui.spindle*.N.*is-on* (bit, out) - указывает на то, что шпиндель включен (в любом направлении)
- *halui.spindle*.N.*reverse* (bit, in)- запускает движение шпинделя против часовой стрелки

- *halui.spindle*.N.*runs-backward* (bit, out) - указывает на то, что шпиндель включен и в реверсе
- *halui.spindle*.N.*runs-forward* (bit, out) - указывает, что шпиндель включен и вращается в прямом направлении
- *halui.spindle*.N.*start* (bit, in) - запускает шпиндель
- *halui.spindle*.N.*stop* (bit, in) - останавливает шпиндель

#### 5.11.4.19 Tool

- *halui.tool.length-offset.a* (float out) - текущая примененная коррекция длины инструмента для оси A
- *halui.tool.length-offset.b* (float out) - текущая примененная коррекция длины инструмента для оси B
- *halui.tool.length-offset.c* (float out) - текущая примененная коррекция длины инструмента для оси C
- *halui.tool.length-offset.u* (float out) - текущее применяемое смещение длины инструмента для оси U
- *halui.tool.length-offset.v* (float out) - текущее применяемое смещение длины инструмента для оси V
- *halui.tool.length-offset.w* (float out) - текущее применяемое смещение длины инструмента для оси W
- *halui.tool.length-offset.x* (float out) - текущее примененное смещение длины инструмента для оси X
- *halui.tool.length-offset.y* (float out) - текущее примененное смещение длины инструмента для оси Y
- *halui.tool.length-offset.z* (float out) - текущая примененная коррекция длины инструмента для оси Z
- *halui.tool.diameter* (float out) - Текущий диаметр инструмента или 0, если инструмент не загружен.
- *halui.tool.number* (u32, out) - указывает текущий выбранный инструмент

## 5.12 Примеры Halui

Для работы примеров Halui необходимо добавить следующую строку в раздел [HAL] файла INI.

```
HALUI = halui
```

### 5.12.1 Remote 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.

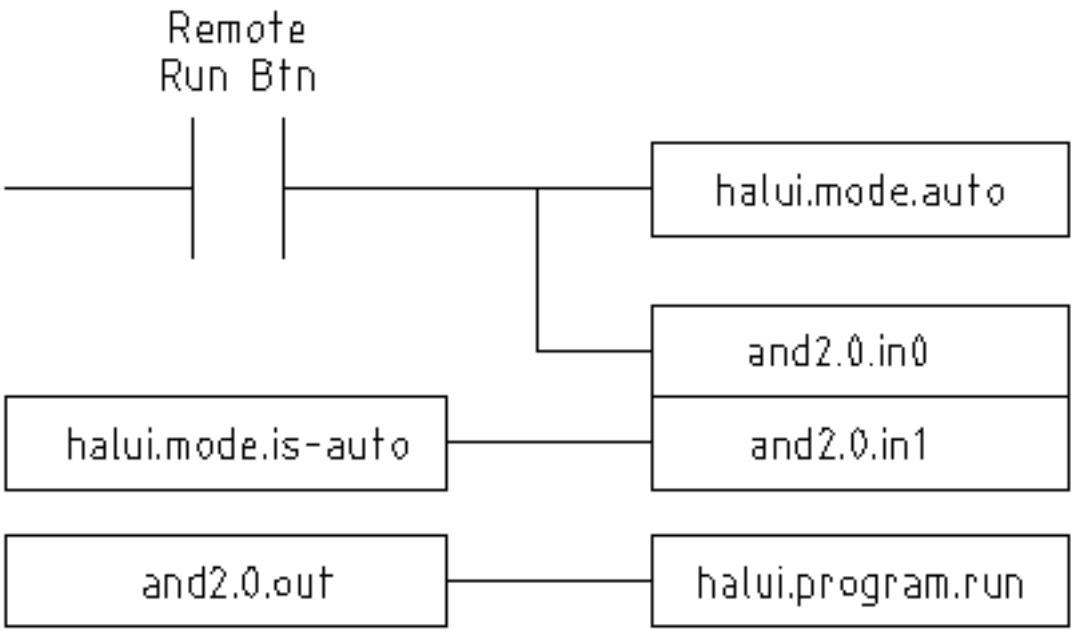



Figure 5.20: Remote Start Example

The hal commands needed to accomplish the above are:

```
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
```

Notice on line one that there are two reader pins, this can also be split up to two lines like this:

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

### 5.12.2 Pause & Resume

This example was developed to allow LinuxCNC to move a rotary axis on a signal from an external machine. The coordination between the two systems will be provided by two Halui components:

- `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 => ”your output pin”
net resume halui.program.resume <= ”your input pin”
```

Your input and output pins are connected to the pins wired to the other controller. They may be parallel port pins or any other I/O pins that you have access to.

This system works in the following way. When an M0 is encountered in your G-code, the `halui.program.is-p` signal goes true. This turns on your output pin so that the external controller knows that LinuxCNC is paused.

To resume the LinuxCNC G-code program, when the external controller is ready it will make its output true. This will signal LinuxCNC that it should resume executing G-code.

Difficulties in timing

- The ”resume” input return signal should not be longer than the time required to get the G-code running again.
- The ”is-paused” output should no longer be active by the time the ”resume” signal ends.

These timing problems could be avoided by using ClassicLadder to activate the ”is-paused” output via a monostable timer to deliver one narrow output pulse. The ”resume” pulse could also be received via a monostable timer.

## 5.13 Создание Python компонентов не реального времени

В этом разделе объясняются принципы реализации компонентов HAL с помощью языка программиров Python.

### 5.13.1 Базовый пример использования

Компонент, не работающий в режиме реального времени, начинает с создания своих контактов и параметров, затем входит в цикл, который периодически управляет всеми выходами со входов. Следующий компонент копирует значение, отображаемое на его входном контакте (*passthrough.in*), на выходной контакт (*passthrough.out*) примерно раз в секунду.

```
#!/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
```

Скопируйте приведенный выше листинг в файл с именем ”passthrough”, сделайте его исполняемым (*chmod +x*) и поместите в свой *$PATH*. Затем попробуйте его:

**Копия экрана с подробной информацией о выполнении вновь созданного passthrough модуля HAL.**

```
$ halrun

halcmd: loadusr passthrough

halcmd: show pin

    Component Pins:
    Owner Type  Dir     Value  Name
     03   float IN          0  passthrough.in
     03   float OUT         0  passthrough.out

halcmd: setp passthrough.in 3.14
```

```
halcmd: show pin

    Component Pins:
    Owner Type  Dir     Value  Name
     03   float IN       3.14  passthrough.in
     03   float OUT      3.14  passthrough.out
```

### 5.13.2 Компоненты и задержки, не относящиеся к реальному времени

Если вы быстро набрали ”show pin”, вы можете увидеть, что *passthrough.out* по-прежнему имеет старое значение 0. Это связано с вызовом *time.sleep(1)*, который делает присвоение выходному контакту не чаще одного раза в секунду. Поскольку это компонент, не работающий в режиме реального времени, фактическая задержка между назначениями может быть намного дольше, если память, используемая транзитным компонентом, выгружается на диск, поскольку назначение может быть отложено до тех пор, пока эта память не будет выгружена обратно.

Таким образом, компоненты, не работающие в режиме реального времени, подходят для интерактивны с пользователем элементов, таких как панели управления (задержки в диапазоне миллисекунд не заметны и более длинные задержки допустимы), но не для отправки шаговых импульсов на плату шагового драйвера (задержки всегда должны быть в диапазоне микросекунд, несмотря ни на что).

### 5.13.3 Создание контактов и параметров

```
h = hal.component(”passthrough”)
```

Сам компонент создается вызовом конструктора *hal.component*. Аргументами являются имя компонента HAL и (опционально) префикс, используемый для имен выводов и параметров. Если префикс не указан, используется имя компонента.

```
h.newpin(”in”, hal.HAL_FLOAT, hal.HAL_IN)
```

Затем выводы создаются путем вызова методов объекта компонента. Аргументами являются: суффикс имени контакта, тип контакта и направление контакта. Для параметров аргументами являются: суффикс имени параметра, тип параметра и направление параметра.

Table 5.31: Имена опций HAL

| **Типы контактов и параметров:** | HAL_BIT | HAL_FLOAT | HAL_S32 | HAL_U32 |
|---|---|---|---|---|
| **HAL_S64** | HAL_U64 | Направления контактов: | HAL_IN | HAL_OUT |
| **HAL_IO** | | | | Направления параметров: |
| **HAL_RO** | HAL_RW | | | |

Полное имя контакта или параметра формируется путем объединения префикса и суффикса с помощью ”.”, поэтому в примере созданный вывод называется *passthrough.in*.

```
h.ready()
```

После создания всех выводов и параметров вызовите метод *.ready()*.

#### 5.13.3.1 Изменение префикса

Префикс можно изменить, вызвав метод *.setprefix()*. Текущий префикс можно получить, вызвав метод *.getprefix()*.

### 5.13.4 Чтение и запись контактов и параметров

Для контактов и параметров, которые также являются правильными идентификаторами Python, значение можно получить или установить с помощью синтаксиса атрибута:

```
h.out = h.in
```

Для всех контактов, независимо от того, являются ли они правильными идентификаторами Python, значение можно получить или установить с помощью синтаксиса индексов:

```
h['out'] = h['in']
```

Чтобы увидеть все контакты с их значениями, getpins возвращает все значения в словаре этого компонента.

```
h.getpins()
>>>{'in': 0.0, 'out': 0.0}
```

#### 5.13.4.1 Управляющие выходные контакты (HAL_OUT)

Периодически, обычно в ответ на таймер, все выводы HAL_OUT должны "управляться", присваивая им новое значение. Это следует делать независимо от того, отличается ли значение от последнего присвоенного значения. Когда контакт подключен к сигналу, его старое выходное значение не копируется в сигнал, поэтому правильное значение появится в сигнале только после того, как компонент присвоит новое значение.

#### 5.13.4.2 Управление двунаправленными (HAL_IO) контактами

Вышеупомянутое правило не распространяется на двунаправленные контакты. Вместо этого двунаправленный контакт должен управляться компонентом только тогда, когда компонент желает изменить значение. Например, в интерфейсе стандартного энкодера, компонент энкодера только устанавливает для вывода *index-enable* значение **FALSE** (когда виден индексный импульс и старое значение **TRUE**), но никогда не устанавливает для него значение **TRUE**. Повторяющееся управление контактом **FALSE** может привести к тому, что другой подключенный компонент будет действовать так, как будто был замечен еще один индексный импульс.

### 5.13.5 Выход

Запрос *halcmd unload* для компонента доставляется как исключение *KeyboardInterrupt*. При поступлении запроса на выгрузку процесс должен либо завершиться через короткое время, либо вызвать метод *.exit()* компонента, если для завершения процесса завершения работы необходимо выполнить существенную работу (например, чтение или запись файлов).

### 5.13.6 Полезные функции

См. Интерфейс Python HAL для обзора доступных функций.

### 5.13.7 Constants

Используйте их для указания деталей, а не значения, которое они содержат.

- HAL_BIT
- HAL_FLOAT
- HAL_S32
- HAL_U32
- HAL_S64
- HAL_U64
- HAL_IN
- HAL_OUT
- HAL_RO
- HAL_RW
- MSG_NONE
- MSG_ALL
- MSG_DBG
- MSG_ERR
- MSG_INFO
- MSG_WARN

### 5.13.8 Системная информация

Прочтите их, чтобы получить информацию о системе реального времени.

- is_kernelspace
- is_rt
- is_sim
- is_userspace

## 5.14 Canonical Device Interfaces

### 5.14.1 Введение

В следующих разделах показаны контакты, параметры и функции, предоставляемые ”стандартными устройствами”. Все драйверы устройств HAL должны предоставлять одни и те же контакты и параметры и реализовывать одинаковое поведение.

Обратите внимание, что для канонического устройства определены только поля _<io-type>_ и _<specific-name>_. Поля _<device-name>, _<device-num>_, и _<chan-num>_ устанавливаются на основе характеристик реального устройства.

## 5.14.2 Цифровой вход

Стандартный цифровой ввод (поле типа ввода-вывода: `digin`) довольно прост.

### 5.14.2.1 Контакты

(bit) **in**:: Состояние аппаратного входа. (bit) **in-not**:: Инвертированное состояние входа.

### 5.14.2.2 Параметры

Нет

### 5.14.2.3 Функции

(funct) **read**:: Считайте оборудование и установите контакты HAL `in` и `in-not`.

## 5.14.3 Цифровой вывод

Стандартный цифровой выход (поле типа ввода-вывода: `digout`) также очень прост.

### 5.14.3.1 Контакты

(bit) **out**:: Значение, которое будет записано (возможно, инвертировано) на аппаратный выход.

### 5.14.3.2 Параметры

(bit) **invert**:: Если TRUE, **out** инвертируется перед записью в оборудование.

### 5.14.3.3 Функции

(funct) **write**:: Считайте **out** и **invert** и соответствующим образом настройте аппаратный выход.

## 5.14.4 Аналоговый вход

Стандартный аналоговый вход (тип ввода-вывода: `adcin`). Ожидается, что это будет использоваться для аналого-цифровых преобразователей, которые преобразуют, например, напряжение в непрерывно диапазоне значений.

### 5.14.4.1 Контакты

(float) **value**:: Аппаратные показания, масштабированные в соответствии с параметрами **scale** и **offset**.
**value** = ((входное значение, в аппаратно-зависимых единицах) * **scale**) - **offset**

#### 5.14.4.2 Параметры

(float) **scale**:: Входное напряжение (или ток) будет умножено на **scale** перед выводом в **value**. (float) **offset**:: Оно будет вычтено из входного напряжения (или тока) оборудования после применения масштабного множителя. (float) **bit_weight**:: Значение одного младшего бита (LSB). Фактически это степень детализации входных данных. (float) **hw_offset**:: Значение, присутствующее на входе, когда на входные контакт(ы) подается напряжение 0 В.

#### 5.14.4.3 Функции

(funct) **read**:: Считайте значения этого аналогового входного канала. Это может использоваться для считывания отдельных каналов или может привести к считыванию всех каналов.

### 5.14.5 Аналоговый вывод

Стандартный аналоговый выход (Тип ввода-вывода: **adcout**). Это предназначено для любого типа оборудования, которое может выводить более или менее непрерывный диапазон значений. Примерами являются цифро-аналоговые преобразователи или генераторы ШИМ.

#### 5.14.5.1 Контакты

(float) **value**:: Значение, которое необходимо записать. Фактическое значение, выводимое на оборудование, будет зависеть от параметров масштаба и смещения. (bit) **enable**:: Если false, то выводится 0 на оборудование, независимо от вывода **value**.

#### 5.14.5.2 Параметры

(float) **offset**:: Это будет добавлено к **value** перед обновлением оборудования. (float) **scale**:: Это значение должно быть установлено таким образом, чтобы вход 1 на контакте **value** приводил к тому, что на выводе аналогового контакта появлялся 1 вольт. (float) **high_limit** (optional):: Если при вычислении значения для вывода на оборудование **value** + **offset** больше, чем **high_limit**, то вместо него будет использоваться **high_limit**. (float) **low_limit** (опционально):: Если при вычислении значения для вывода на оборудование **value** + **offset** меньше **low_limit**, то вместо него будет использоваться **low_limit**. (float) **bit_weight** (опционально):: Значение одного младшего бита (LSB) в вольтах (или мА для токовых выходов). (float) **hw_offset** (опционально):: Фактическое напряжение (или ток), которое будет выводиться, если в аппаратуру записано 0.

#### 5.14.5.3 Функции

(funct) **write**:: Это приводит к выводу вычисленного значения на оборудование. Если enable имеет значение false, то на выходе будет 0, независимо от **value**, **scale** и **offset**. Значение ”0” зависит от аппаратуры. Например, биполярному 12-битному АЦП может потребоваться записать 0x1FF (средняя шкала) в ЦАП, чтобы получить 0 Вольт на контакте оборудования. Если enable в true, считываются масштаб, смещение и значение и выводит их на АЦП (**scale** * **value**) + **offset**. Если enable в false, тогда выводится 0.

# 5.15 HAL Инструменты

## 5.15.1 Halcmd

`halcmd` — это инструмент командной строки для управления HAL. Существует довольно полная справочная страница по `halcmd`, которая будет установлена, если вы установили LinuxCNC из исходного кода или пакета. На странице руководства представлена информация об использовании:

```
man halcmd
```

Если вы скомпилировали LinuxCNC для ”run-in-place”, вам необходимо запустить сценарий rip-environment, чтобы сделать справочную страницу доступной:

```
cd toplevel_directory_for_rip_build
. scripts/rip-environment
man halcmd
```

HAL Tutorial содержит ряд примеров использования halcmd и является хорошим учебным пособием по `halcmd`.

## 5.15.2 Halmeter

Halmeter — это *вольтметр* для HAL. Он позволяет вам просмотреть вывод, сигнал или параметр и отображает текущее значение этого элемента. Его довольно просто использовать. Запустите его, набрав *halmeter* в оболочке X Windows. Halmeter — это приложение с графическим интерфейсом. Появится небольшое окно с двумя кнопками ”Select” и ”Exit”. Выход легкий - он закрывает программу. Select открывает окно большего размера с тремя вкладками. На одной вкладке перечислены все контакты, определенные в настоящее время в HAL. На следующей вкладке перечислены все сигналы, а на последней вкладке перечислены все параметры. Щелкните вкладку, затем щелкните контакт/сигнал/параметр. Затем нажмите ”OK”. Списки исчезнут, а в небольшом окне отобразятся имя и значение выбранного элемента. Дисплей обновляется примерно 10 раз в секунду. Если вы нажмете ”Accept” вместо ”OK”, в маленьком окне отобразятся имя и значение выбранного элемента, но большое окно останется на экране. Это удобно, если вы хотите быстро просмотреть несколько разных элементов.

Вы можете запустить несколько halmeters одновременно, если хотите отслеживать несколько объектов. Если вы хотите запустить halmeter, не закрывая окно оболочки, введите `halmeter &`, чтобы запустить его в фоновом режиме. Вы также можете заставить halmeter немедленно начать отображать определенный элемент, добавив `pin|sig|par[am] _<name>_` в командную строку. Он отобразит контакт, сигнал или параметр *<name>* сразу после запуска, если такого элемента нет, он просто запустится нормально. И, наконец, если вы указываете элемент для отображения, вы можете добавить `-s` перед `pin|sig|param`, чтобы указать halmeter использовать маленькое окно. Имя элемента будет отображаться в строке заголовка, а не под значением, и кнопок не будет. Полезно, когда вам нужно много метров на небольшом пространстве экрана.

Дополнительную информацию см. в разделе Halmeter Tutorial.

`halmeter` можно загрузить с терминала или из AXIS. `halmeter` отображает значения быстрее, чем `halshow`. «halmeter» имеет два окна: одно для выбора контакта, сигнала или параметра для мониторинга, а другое отображает значение. Одновременно могут быть открыты несколько ``halmeter``ов. Если вы используете скрипт для открытия нескольких ``halmeter``, вы можете установить положение каждого из них с помощью `-g X Y` относительно верхнего левого угла экрана. Например:

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

Дополнительные параметры см. на странице руководства и в разделе Halmeter.

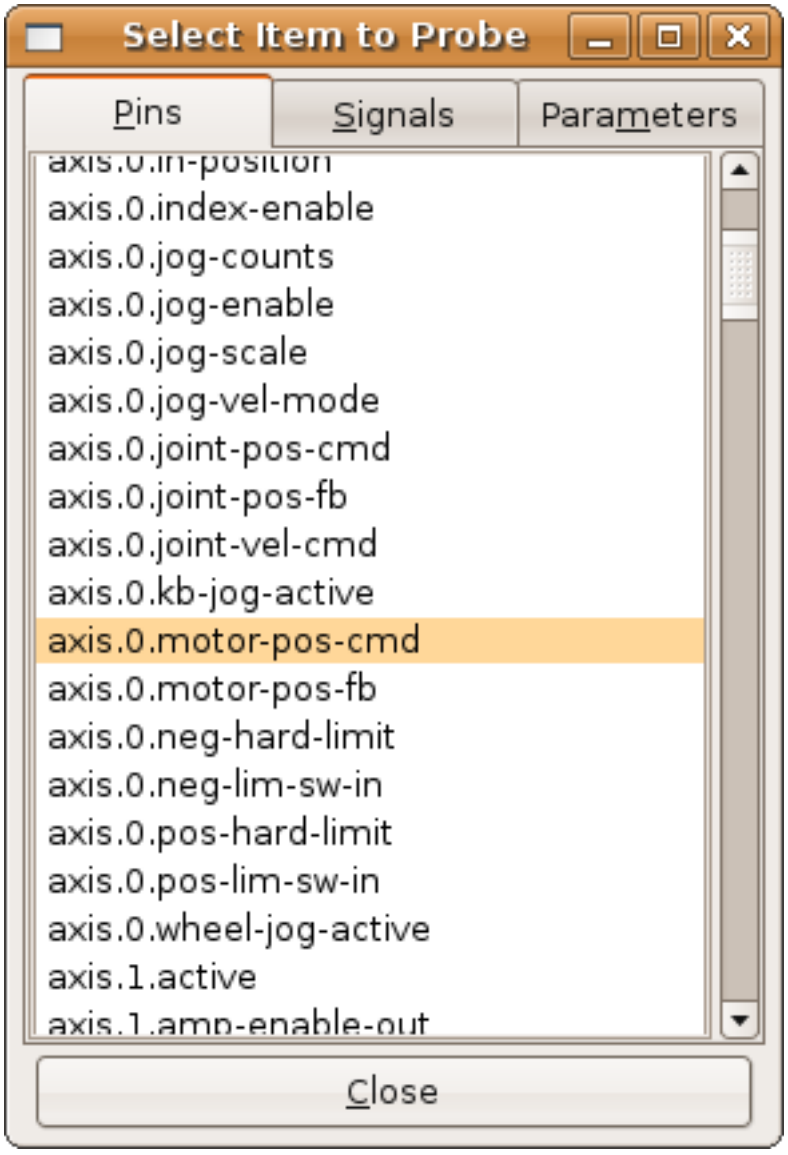



Figure 5.21: Окно выбора Halmeter

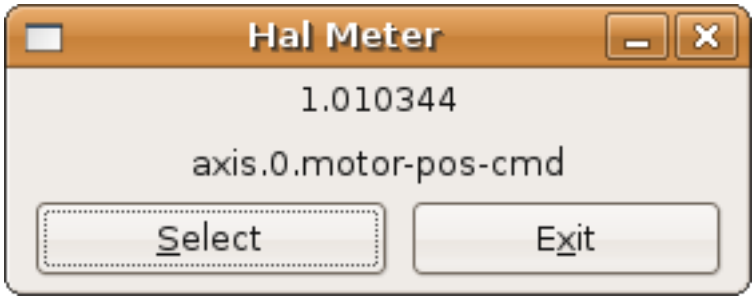



Figure 5.22: Окно наблюдения Halmeter

### 5.15.3 Halshow

`halshow` (полное описание использования) можно запустить из командной строки, чтобы показать подробную информацию о выбранных компонентах, контактах, параметрах, сигналах, функциях и потоках работающего HAL. Вкладка WATCH обеспечивает непрерывное отображение выбранных контактов, параметров и элементов сигналов. В меню File имеются кнопки для сохранения элементов наблюдения в списке наблюдения и для загрузки существующего списка наблюдения. Элементы списка наблюдения также могут загружаться автоматически при запуске. Для использован командной строки:

```
halshow --help
Usage:
  halshow [Options] [watchfile]
  Options:
           --help    (this help)
           --fformat format_string_for_float
           --iformat format_string_for_int

Notes:
  Create watchfile in halshow using: 'File/Save Watch List'.
  LinuxCNC must be running for standalone usage.
```

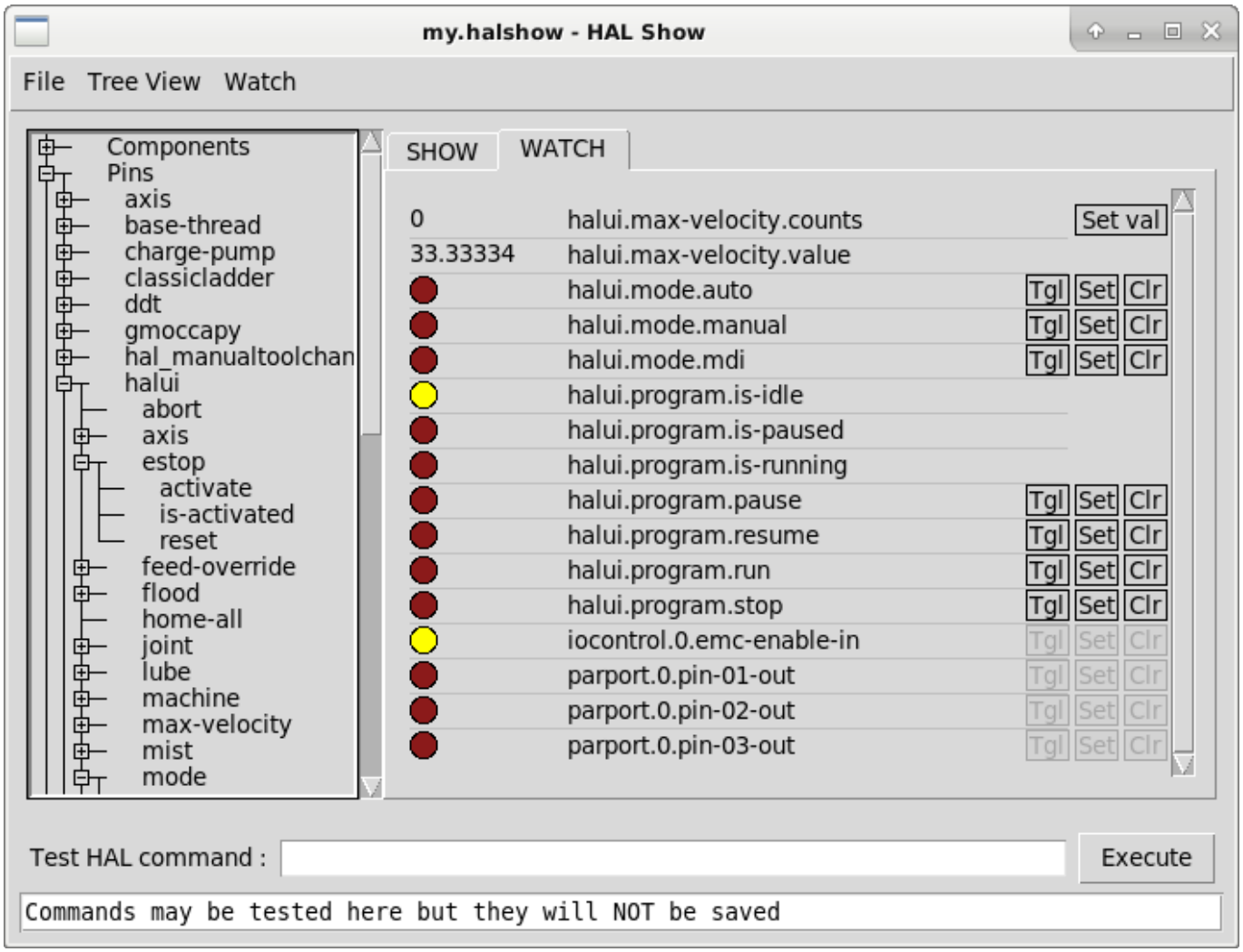



Figure 5.23: Halshow Watch Tab

Файл наблюдения, созданный с помощью пункта меню *File/Save Watch List*, форматируется как одна строка с токенами ”pin+”, ”param+”, ”sig=+”, за которыми следуют соответствующий вывод, параметр или имя сигнала. Пары токен-имя разделяются пробелом.

**Пример однострочного файла наблюдения**

```
pin+joint.0.pos-hard-limit pin+joint.1.pos-hard-limit sig+estop-loop
```

Файл наблюдения, созданный с помощью пункта меню *File/Save Watch List (multiline)*, форматируется отдельными строками для каждого элемента, идентифицируемого парами имени токена, как описано выше.

**Пример файла наблюдения с разделенными строками**

```
pin+joint.0.pos-hard-limit
pin+joint.1.pos-hard-limit
sig+estop-loop
```

При загрузке файла наблюдения с помощью пункта меню *File/Load Watch List* пары «токен-имя» могут отображаться в виде одной или нескольких строк. Пустые строки и строки, начинающиеся с символа #, игнорируются.

### 5.15.4 Halscope

*Halscope* — это *осциллограф* для HAL. Он позволяет фиксировать значение выводов, сигналов и параметров в зависимости от времени. Полные инструкции по эксплуатации со временем должны быть размещены здесь. А пока обратитесь к разделу Halscope в главе руководства, где объясняются основы.

Селектор меню ”File” на halscope содержит кнопки для сохранения конфигурации или открытия ранее сохраненной конфигурации. При завершении работы halscope последняя конфигурация сохраняется в файле с именем autosave.halscope.

Файлы конфигурации также можно указать при запуске halscope из командной строки. Использование справки по командной строке (`-h`):

```
halscope -h
Usage:
  halscope [-h] [-i infile] [-o outfile] [num_samples]
```

### 5.15.5 Sim Pin

sim_pin — это утилита командной строки для отображения и обновления любого количества записываемых контактов, параметров или сигналов.

**sim_pin Использование**

```
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

        b''Пb''b''ob''b''дb''b''дb''b''eb''b''pb''b''жb''b''иb''b''вb''b''аb''b''юb''b' ↩
            'тb''b''cb''b''яb'' b''тb''b''иb''b''пb''b''ыb'' b''эb''b''лb''b''eb''b''мb''b' ↩
            'eb''b''нb''b''тb''b''ob''b''вb'' HAL bit,s32,u32,float.

        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
```

Полную информацию смотрите на странице руководства:

```
man sim_pin
```

**sim_pin Пример (при работающем LinuxCNC)**

```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0
sim_pin example.sel example.in1 sig_example &
```

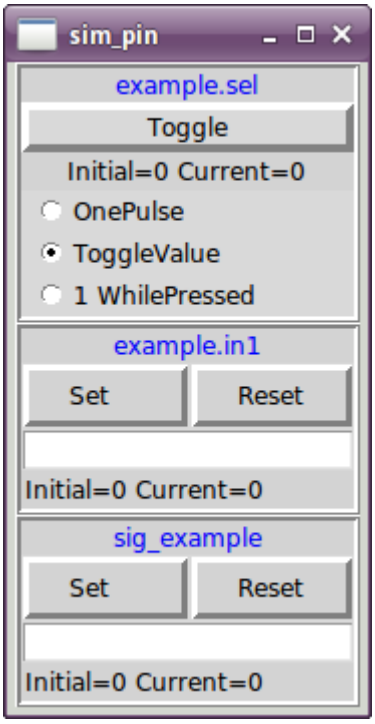



Figure 5.24: sim_pin Окно

## 5.15.6 Simulate Probe

`simulate_probe` — это простой ГИП для имитации активации вывода motion.probe-input. Использовани

```
simulate_probe &
```

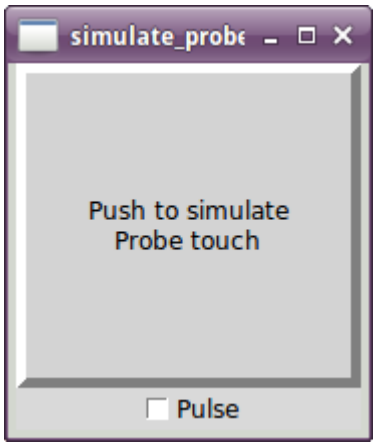



Figure 5.25: simulate_probe Окно

### 5.15.7 HAL Histogram

`hal-histogram` — это утилита командной строки для отображения гистограмм для контактов HAL.

**Использование:**

```
   hal-histogram --help | -?
or
   hal-histogram [Options] [pinname]
```

Table 5.32: Options:

| Вариант | Value | Description |
|---|---|---|
| --minvalue | minvalue | минимальное количество столбцов, default: 0 |
| --binsize | binsize | binsize, размер столбца, по умолчанию: 100 |
| --nbins | nbins | количество столбцов, по умолчанию: 50 |
| | | |
| --logscale | 0/1 | Логарифмический масштаб оси Y, по умолчанию: 1 |
| --text | note | отображаемый текст, по умолчанию: ”” |
| --show | | показывает количество неотображаемых nbins, по умолчанию отключено |
| --verbose | | прогресс и отладка, по умолчанию выключено |

Примечания:

1. LinuxCNC (или другое приложение HAL) должен быть запущен.
2. Если имя контакта не указано, по умолчанию используется: `motion-command-handler.time`.
3. Это приложение может быть открыто для 5 контактов.
4. Поддерживаются типы контактов: float, s32, u32, bit.
5. The pin must be associated with a realtime thread.

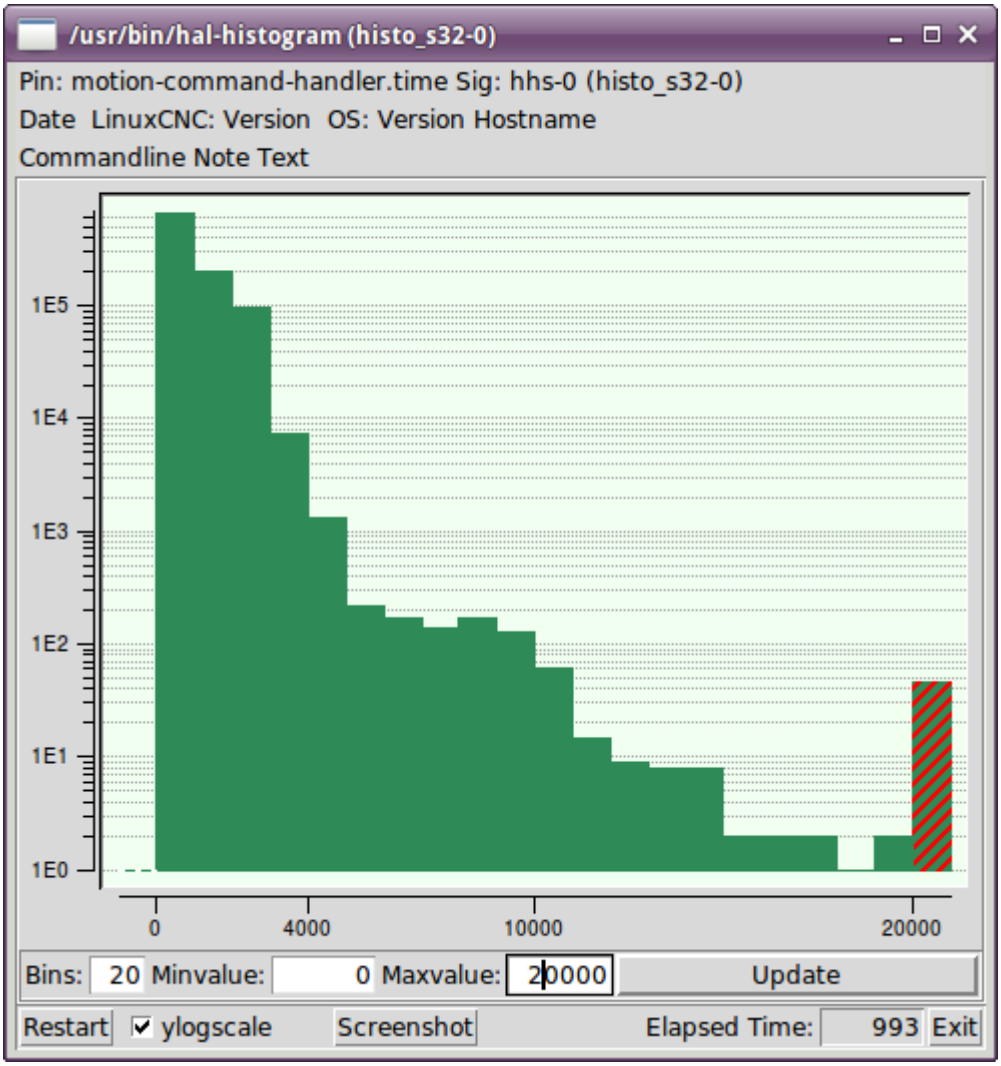



Figure 5.26: hal-histogram Окно

### 5.15.8 Halreport

`halreport` — это утилита командной строки, которая генерирует отчет о соединениях HAL для работающего приложения LinuxCNC (или другого HAL). В отчете показаны все соединения сигналов и отмечены потенциальные проблемы. Информация включает:

1. Описание системы и версия ядра.
2. Сигналы и все подключенные выходные, ввода/вывода и входные контакты.

3. Component_function, thread и addf-order каждого контакта.
4. Выводы компонентов не в реальном времени, имеющие неупорядоченные функции.
5. Идентификация неизвестных функций для необработанных компонентов.
6. Сигналы без выхода.
7. Сигналы без входов.
8. Функции без addf.
9. Теги предупреждений для компонентов, помеченных в документации как устаревшие.
10. Настоящие имена для контактов, которые используют псевдонимы.

Отчет можно создать из командной строки и направить в выходной файл (или на стандартный вывод, если не указано имя выходного файла):

**halreport Использование**

```
Usage:
  halreport -h | --help (this help)
or
  halreport [outfilename]
```

Чтобы генерировать отчет для каждого запуска LinuxCNC, включите halreport и имя выходного файла в качестве записи [APPLICATIONS]APP в INI-файле.

**halreport Пример**

```
[APPLICATIONS]
APP = halreport /tmp/halreport.txt
```

Функция addf-ordering может быть важна для контуров сервопривода, где важна последовательность функций, вычисляемых в каждом периоде сервопривода. Обычно порядок такой:

1. Считать входные контакты,
2. выполнять функции обработчика команд движения и контроллера движения,
3. выполнить расчеты ПИД и, наконец
4. записать выходные контакты.

Для каждого сигнала на критическом пути порядок addf выходного контакта должен быть численно ниже, чем порядок addf критических входных контактов, к которым он подключен.

Для рутинных путей прохождения сигналов, которые обрабатывают входы переключателей, контакты не в реальном времени и т. д., порядок addf часто не имеет решающего значения. Более того, время изменения значений контактов не в реальном времени не может контролироваться или гарантироваться с интервалами, обычно используемыми для потоков HAL.

Пример выдержек из файла отчета, показывающий pid цикл для hostmot2 stepgen, работающего в режиме скорости на станке с trivkins и `joint.0`, соответствующим координате оси X:

```
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)
```

В приведенном выше примере HALFILE использует псевдонимы halcmd для упрощения имен контактов платы FPGA hostmot2 с помощью таких команд:

```
alias pin hm2_7i92.0.stepgen.00.position-fb h.00.position-fb
```

---

**Note**
Обнаружение сомнительной функции компонента может произойти для

1. неподдерживаемых (устаревших) компонентов,
2. созданных пользователем компонентов, которые используют несколько функций или нетрадиционные имена функций, или
3. Компоненты, созданные не в реальном времени с помощью ГИП, у которых отсутствуют отличительные характеристики, такие как префикс, основанный на имени ГИП программы.

Сомнительные функции помечаются знаком вопроса ”?”.

---

---

**Note**
Контакты компонента, которые не могут быть связаны с известной функцией потока, сообщают о функции как ”Unknown”.

---

`halreport` генерирует отчет о соединениях (без типов контактов и текущих значений) для работающего приложения HAL, чтобы помочь в проектировании и проверке соединений. Это помогает понять, что является источником значения контакта. Используйте эту информацию с такими приложениями как «`halshow`, `halmeter`, `halscope` или командой `halcmd show` в терминале.

# Chapter 6

# Драйверы оборудования

## 6.1 Драйвер параллельного порта

Компонент hal_parport — это драйвер традиционного параллельного порта ПК. Порт имеет в общей сложности 17 физических контактов. Исходный параллельный порт делил эти контакты на три группы: данные, управление и состояние. Группа данных состоит из 8 выходных контактов, группа управления состоит из 4 контактов, а группа состояния состоит из 5 входных контактов.

В начале 1990-х годов был представлен двунаправленный параллельный порт, который позволяет использовать группу данных для вывода или ввода. Драйвер HAL поддерживает двунаправленный порт и позволяет пользователю устанавливать группу данных как входную или выходную. Если порт настроен как *выход*, порт обеспечивает в общей сложности 12 выходов и 5 входов. Если он настроен как *вход*, он обеспечивает 4 выхода и 13 входов.

В некоторых параллельных портах контакты группы управления представляют собой открытые коллекторы, которые также могут быть переведены на низкий уровень с помощью внешнего затвора. На плате с управляющими контактами с открытым коллектором. Если настроено как *x*, он обеспечивает 8 выходов и 9 входов.

В некоторых параллельных портах группа управления имеет двухтактные драйверы и не может использоваться в качестве входа.

---

**HAL и открытые коллекторы**
HAL не может автоматически определить, являются ли двунаправленные контакты режима *x* на самом деле открытыми коллекторами (OK). В противном случае их нельзя использовать в качестве входов, и попытка перевести их на НИЗКИЙ уровень от внешнего источника может привести к повреждению оборудования.
Чтобы определить, имеет ли ваш порт контакты «открытый коллектор», загрузите hal_parport в режиме *x*. Если устройство не подключено, HAL должен прочитать вывод как TRUE. Затем вставьте резистор сопротивлением 470 Ом от одного из управляющих контактов к GND. Если результирующее напряжение на выводе управления близко к 0 В, и HAL теперь считывает вывод как ЛОЖЬ, то у вас есть порт OK. Если результирующее напряжение далеко от 0,0 В или HAL не считывает вывод как FALSE, то ваш порт нельзя использовать в режиме *x*.
Внешнее оборудование, которое управляет выводами управления, также должно использовать затворы с открытым коллектором, например 74LS05 (555ЛН2).
На некоторых компьютерах настройки BIOS могут влиять на возможность использования режима *x*. Режим *SPP*, скорее всего, сработает.

---

Никакие другие комбинации не поддерживаются, и порт нельзя изменить с входа на выход после установки драйвера.

Драйвер parport может управлять до 8 портов (определенными параметром MAX_PORTS в hal_parport.c
Порты нумеруются начиная с нуля.

### 6.1.1 Загрузка

Драйвер hal_parport — это компонент реального времени, поэтому его необходимо загрузить в поток реального времени с помощью *loadrt*. Строка конфигурации описывает используемые параллельные порты и (необязательно) их типы. Если строка конфигурации не описывает хотя бы один порт, это ошибка.

```
loadrt hal_parport cfg="port [type] [port [type] ...]"
```

**Указание порта** Числа ниже 16 относятся к параллельным портам, обнаруженным системой. Это самый простой способ настройки драйвера `hal_parport`, который взаимодействует с драйвером parport_pc в Linux, если он загружен. Порт 0 — это первый параллельный порт, обнаруженный в системе, порт 1 — следующий и так далее.

**Базовая конфигурация** При этом будет использоваться первый параллельный порт, обнаруженный Linux:

```
loadrt hal_parport cfg="0"
```

**Использование адреса порта** Вместо этого адрес порта можно указать в шестнадцатеричном формате с префиксом *0x*.

Строка конфигурации представляет собой шестнадцатеричный адрес порта, за которым может следовать направление, повторяющееся для каждого порта. Направления — *in*, *out* или *x* и определяют направление физических контактов 2–9 разъема D-Sub 25. Если направление не указано, группа данных по умолчанию будет настроена как выходные данные. Например:

**Команда для загрузки модуля реального времени hal_parport с дополнительным <config-string>, чтобы указать порт, на котором ожидается карта параллельного порта.**

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

В этом примере устанавливаются драйверы для порта 0x0278 с контактами 2–9 в качестве выходов (по умолчанию, поскольку не указаны ни *in*, ни *out*), порта 0x0378 с контактами 2–9 в качестве входов и порта 0x20A0 с контактами 2–9. 9 явно указаными как выходы. Обратите внимание: для правильной настройки драйверов необходимо знать базовый адрес параллельных портов. Для портов шины ISA это обычно не является проблемой, поскольку порты почти всегда имеют хорошо известный адрес, например 0x278 или 0x378, которые обычно настраиваются в BIOS. Адреса карт шины PCI обычно находятся с помощью `lspci -v` в строке *I/Oports* или в сообщении ядра после запуска `sudo modprobe -a parport_pc`. Адреса по умолчанию нет, поэтому, если `<config-string>` не содержит хотя бы одного адреса, это ошибка.

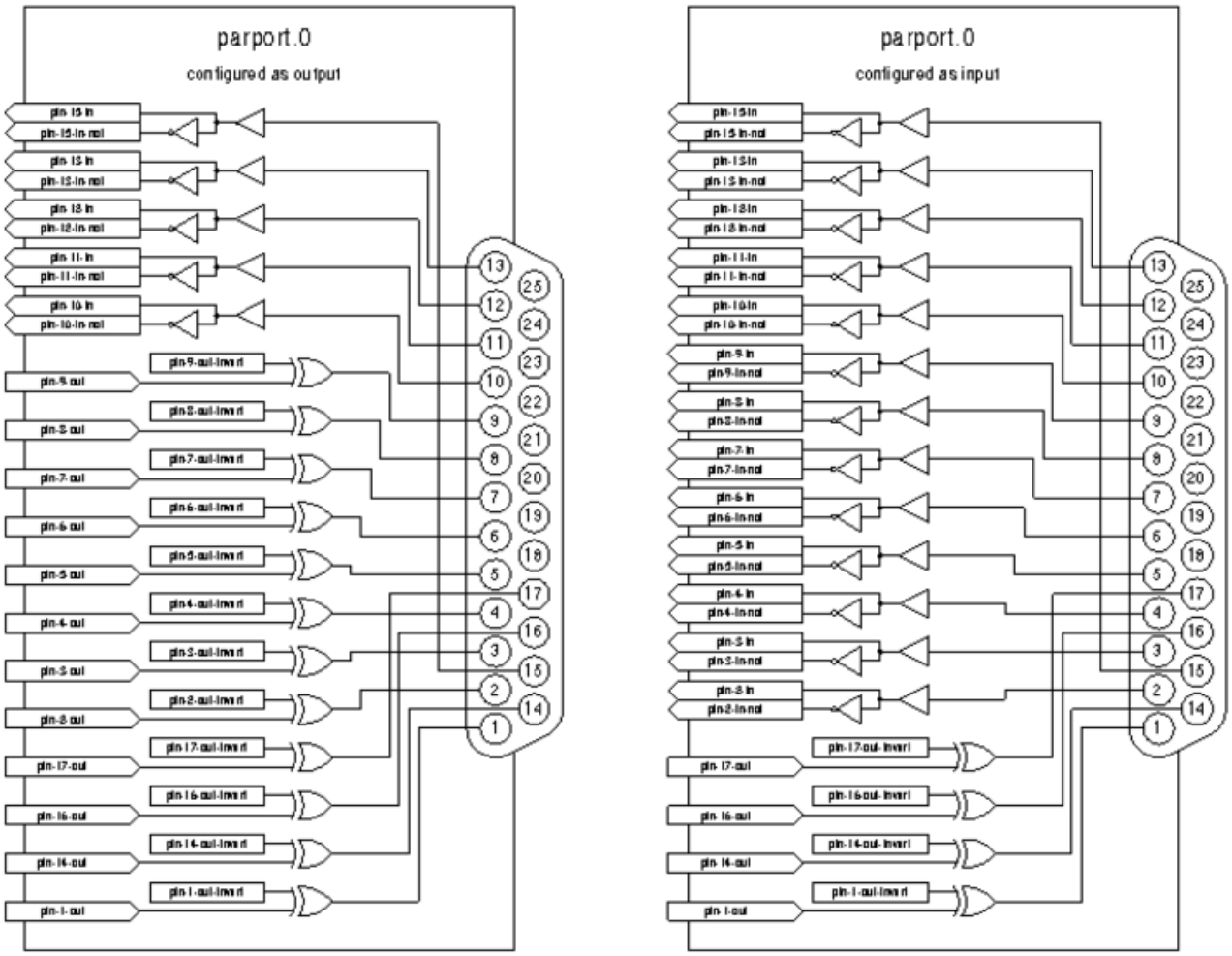



Figure 6.1: Блок-схема Parport

**Type** Для каждого параллельного порта, обслуживаемого драйвером hal_parport, опционально можно указать *тип*. Типом может быть *вход*, *выход*, *epp* или *x*.

Table 6.1: Направление параллельного порта

| Контакт | 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 |

Table 6.1: (continued)

| Контакт | in | out/epp | x |
|---|---|---|---|
| 14 | out | out | in |
| 15 | in | in | in |
| 16 | out | out | in |
| 17 | out | out | in |

Если тип не указан, по умолчанию используется значение *out*.

Тип *epp* аналогичен *out*, но драйвер hal_parport запрашивает переключение порта в режим EPP. Драйвер hal_parport **не** использует протокол шины EPP, но в некоторых системах режим EPP изменяет электрические характеристики порта таким образом, что некоторые аппаратные средства могут работать лучше. Известно, что генератор подкачки Gecko G540 требует этого на некоторых параллельных портах.

См. примечание выше о режиме *x*.

**Пример с двумя параллельными портами** Это активирует два обнаруженных системой параллельн порта: первый в режиме вывода, а второй в режиме ввода:

```
loadrt hal_parport cfg="0 out 1 in"
```

**Parport функции Чтения/Записи (R/W)** Вы также должны указать LinuxCNC выполнить функции *чтения* и *записи*.

```
addf parport.0.read base-thread
addf parport.0.write base-thread
```

### 6.1.2 Адрес порта PCI

Одна хорошая карта порта PCI сделана на чипсете Netmos 9815. Он имеет хорошие сигналы +5 В и может подключаться к одному или двум портам.

Чтобы найти адреса ввода-вывода для карт PCI, откройте окно терминала и используйте команду list pci:

```
lspci -v
```

Найдите запись с надписью «Netmos». Пример 2-портовой карты:

```
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]
```

В ходе экспериментов я обнаружил, что первый порт (порт на карте) использует третий указанный адрес (c000), а второй порт (тот, который подключается с помощью ленточного кабеля) использует первый указанный адрес (b800). В следующем примере показаны встроенный параллельный порт и параллельный порт PCI с использованием направления вывода по умолчанию.

```
loadrt hal_parport cfg="0x378 0xc000"
```

Обратите внимание, что ваши значения будут отличаться. Карты Netmos поддерживают технологию Plug-N-Play и могут менять свои настройки в зависимости от того, в какой слот вы их вставили, поэтому, если вы хотите *залезть под капот* и переставить вещи, обязательно проверьте эти значения перед началом работы. LinuxCNC.

### 6.1.3 Контакты

- parport.<p>.pin-__<n>__-out (bit) Управляет физическим выходным контактом.
- parport.<p>.pin-__<n>__-in (bit) Отслеживает физический входной контакт.
- parport.<p>.pin-__<n>__-in-not (bit) Отслеживает физический входной контакт, но инвертирован.

Для каждого контакта <*p*> — это номер порта, а <*n*> — это физический номер контакта в 25-контактном разъеме D-Sub.

Для каждого физического выходного контакта драйвер создает один контакт HAL, например: parport.0.pin-14-out.

Для каждого физического входного контакта драйвер создает два контакта HAL, например: parport.0.pin-12-in и parport.0.pin-12-in-not.

Контакт -in HAL имеет значение TRUE, если физический контакт имеет высокий уровень, и FALSE, если физический контакт имеет низкий уровень. Контакт -in-not HAL инвертирован и имеет значение FALSE, если на физическом контакте высокий уровень.

### 6.1.4 Параметры

- parport.__<p>__.pin-__<n>__-out-invert (bit) Инвертирует выходной контакт.
- parport.__<p>__.pin-__<n>__-out-reset (bit)(только для контактов -out) TRUE, если этот вывод должен быть сброшен при выполнении функции -reset.
- parport.__<p>__.reset-time (U32) Время (в наносекундах) между тем, когда контакт устанавливает функцией -write и сбрасывается функцией -reset, если она включена.

Параметр -invert определяет, является ли выходной контакт активным высоким или активным низким. Если -invert имеет значение FALSE, установка вывода HAL -out TRUE переводит физический вывод в высокий уровень, а FALSE переводит его в низкий уровень. Если -invert имеет значение TRUE, то установка вывода HAL -out в TRUE приведет к низкому физическому выводу.

### 6.1.5 Функции

- parport.__<p>__.read (funct) Считывает физические входные контакты порта с номером <*p*> и обновляет контакты HAL -in и -in-not..
- parport.read-all (funct) Считывает физические входные контакты всех портов и обновляет контакты HAL -in и -in-not.
- parport.__<p>__.write (funct) Считывает контакты HAL -out порта номер <*p*> и обновляет физические выходные контакты этого порта.

- `parport.write-all` (funct) Считывает контакты HAL `-out` всех портов и обновляет все физические выходные контакты.
- `parport. `__<p>__.reset` (funct) Ожидает, пока не истечет `reset-time` с момента соответствующей `write`, затем сбрасывает контакты на значения, указанные настройками `-out-invert` и `-out-invert`. `reset` должен быть позже в том же потоке, что и `write`. Если `-reset` в TRUE, то функция `reset` установит для контакта значение `-out-invert`. Это можно использовать в сочетании с Stepgen *doublefreq* для создания одного импульса шага за период. Для включения двойной частоты параметр stepgen Stepspace для этого конаткта должен быть установлен в 0.

Отдельные функции предусмотрены для ситуаций, когда один порт необходимо обновить в очень быстром потоке, а другие порты можно обновить в более медленном потоке для экономии времени процессора. Вероятно, не очень хорошая идея использовать одновременно функцию `-all` и отдельную функцию.

### 6.1.6 Распространенные проблемы

Если при загрузке модуль сообщает

```
insmod: error inserting '/home/jepler/emc2/rtlib/hal_parport.ko':
-1 Device or resource busy
```

затем убедитесь, что стандартный модуль ядра «parport_pc» не загружен, footnote: [В пакетах LinuxCNC для Ubuntu файл /etc/modprobe.d/emc2 обычно предотвращает автоматическую загрузку *parport_pc*.] и что другое устройство в системе не затребовало порты ввода-вывода.

Если модуль загружается, но не работает, значит, адрес порта неверен.

### 6.1.7 Использование DoubleStep

Чтобы настроить DoubleStep на параллельном порту, вы должны добавить функцию parport.n.reset после parport.n.write и настроить Stepspace на 0 и желаемое время сброса. Таким образом, этот шаг может быть подтвержден в каждом периоде в HAL, а затем отключен parport после удержания на время, указанное в `parport. `__n__.reset-time`.

Например:

```
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
```

Дополнительную информацию о DoubleStep можно найти на странице wiki.

### 6.1.8 probe_parport

В современных ПК для использования параллельных портов может потребоваться настройка Plug and Play (PNP). Модуль ядра *probe_parport* настраивает все имеющиеся порты PNP. Он должен быть загружен до *hal_parport*. На машинах без порта PNP его можно загрузить, но это не окажет никакого эффекта.

#### 6.1.8.1 Установка probe_parport

Если, когда модуля ядра parport_pc загружен командой:

```
sudo modprobe -a parport_pc; sudo rmmod parport_pc
```

Ядро Linux выводит сообщение, подобное:

```
parport: PnPBIOS parport detected.
```

Вероятно, использование этого модуля будет необходимым.

Наконец, должны быть загружены компоненты HAL parport:

```
loadrt probe_parport
loadrt hal_parport ...
```

## 6.2 AX5214H Driver

The Axiom Measurement & Control AX5214H is a 48 channel digital I/O board. It plugs into an ISA bus, and resembles a pair of 8255 chips. In fact it may be a pair of 8255 chips, but I'm not sure. If/when someone starts a driver for an 8255 they should look at the ax5214 code, much of the work is already done.

### 6.2.1 Installing

```
loadrt hal_ax5214h cfg="<config-string>"
```

The config string consists of a hex port address, followed by an 8 character string of "I" and "O" which sets groups of pins as inputs and outputs. The first two character set the direction of the first two 8 bit blocks of pins (0-7 and 8-15). The next two set blocks of 4 pins (16-19 and 20-23). The pattern then repeats, two more blocks of 8 bits (24-31 and 32-39) and two blocks of 4 bits (40-43 and 44-47). If more than one board is installed, the data for the second board follows the first. As an example, the string *"0x220 IIIOIIOO 0x300 OIOOIOIO"* installs drivers for two boards. The first board is at address 0x220, and has 36 inputs (0-19 and 24-39) and 12 outputs (20-23 and 40-47). The second board is at address 0x300, and has 20 inputs (8-15, 24-31, and 40-43) and 28 outputs (0-7. 16-23, 32-39, and 44-47). Up to 8 boards may be used in one system.

### 6.2.2 Контакты

- *(bit) ax5214.<boardnum>.out-<pinnum>* — Drives a physical output pin.
- *(bit) ax5214.<boardnum>.in-<pinnum>* — Tracks a physical input pin.
- *(bit) ax5214.<boardnum>.in-<pinnum>-not* — Tracks a physical input pin, inverted.

For each pin, <boardnum> is the board number (starts at zero), and <pinnum> is the I/O channel number (0 to 47).

Note that the driver assumes active LOW signals. This is so that modules such as OPTO-22 will work correctly (TRUE means output ON, or input energized). If the signals are being used directly without buffering or isolation the inversion needs to be accounted for. The in- HAL pin is TRUE if the physical pin is low (OPTO-22 module energized), and FALSE if the physical pin is high (OPTO-22 module off). The in-<pinnum>-not HAL pin is inverted — it is FALSE if the physical pin is low (OPTO-22 module energized). By connecting a signal to one or the other, the user can determine the state of the input.

### 6.2.3 Параметры

- *(bit) ax5214.<boardnum>.out-<pinnum>-invert* — Inverts an output pin.

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 low, turning ON an attached OPTO-22 module, and FALSE drives it high, turning OFF the OPTO-22 module. If -invert is TRUE, then setting the HAL out- pin TRUE will drive the physical pin high and turn the module OFF.

### 6.2.4 Функции

- *(funct) ax5214.<boardnum>.read* — Reads all digital inputs on one board.
- *(funct) ax5214.<boardnum>.write* — Writes all digital outputs on one board.

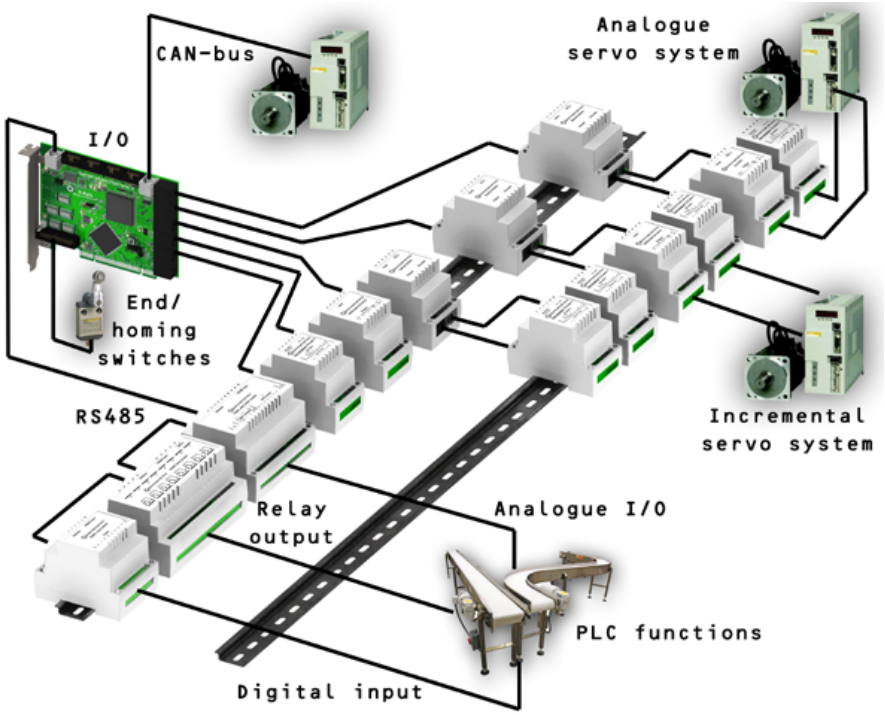
## 6.3 General Mechatronics Driver

General Mechatronics GM6-PCI card based motion control system

For detailed description, please refer to the System integration manual.

The GM6-PCI motion control card is based on an FPGA and a PCI bridge interface ASIC. A small automated manufacturing cell can be controlled, with a short time system integration procedure. The following figure demonstrating the typical connection of devices related to the control system:

- It can control up to six axis, each can be stepper or CAN bus interface or analogue servo.
- GPIO: Four time eight I/O pins are placed on standard flat cable headers.
- RS485 I/O expander modules: RS485 bus was designed for interfacing with compact DIN-rail mounted expander modules. An 8-channel digital input, an 8-channel relay output and an analogue I/O (4x +/-10 Volts output and 8x +/-5 Volts input) modules are available now. Up to 16 modules can be connected to the bus altogether.
- 20 optically isolated input pins: Six times three for the direct connection of two end switch and one homing sensor for each joint. And additionally, two optically isolated E-stop inputs.




Installing:

```
loadrt hal_gm
```

Во время загрузки (или попытки загрузки) драйвер печатает некоторые полезные отладочные сообщения в журнал ядра, которые можно просмотреть с помощью dmesg.

Up to 3 boards may be used in one system.

The following connectors can be found on the GM6-PCI card:

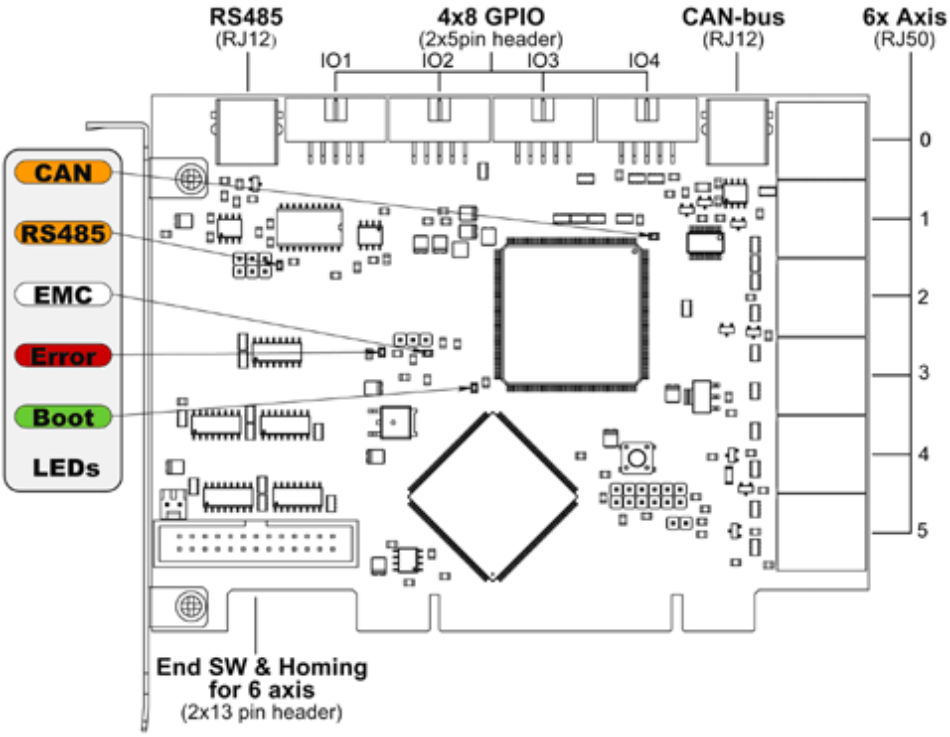



Figure 6.2: GM6-PCI card connectors and LEDs

### 6.3.1 I/O connectors

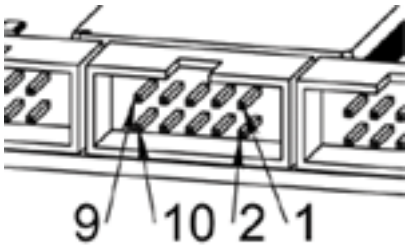



Figure 6.3: Pin numbering of GPIO connectors

Table 6.2: Pinout of GPIO connectors

| 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 |

Each pin can be configured as digital input or output. GM6-PCI motion control card has 4 general purpose I/O (GPIO) connectors, with eight configurable I/O on each. Every GPIO pin and parameter name begins as follows:

```
gm.<card_no>.gpio.<gpio_con_no>
```

where <*gpio_con_no*> is from 0 to 3.

**State of the first pin of the first GPIO connector on the GM6-PCI card.**

```
gm.0.gpio.0.in-0
```

HAL pins are updated by function

```
gm.<card_no>.read
```

#### 6.3.1.1 Контакты

Table 6.4: GPIO pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| `.in-<0-7>` | (bit, Out) | Input pin |
| `.in-not-<0-7>` | (bit, Out) | Negated input pin |
| `.out-<0-7>` | (bit, In) | Output pin. Used only when GPIO is set to output. |

#### 6.3.1.2 Параметры

Table 6.5: GPIO parameters

| Контакты | Type and direction | Parameter description |
|---|---|---|
| `.is-out-<0-7>` | (bit, R/W) | When True, the corresponding GPIO is set to totem-pole output, other wise set to high impedance input. |
| `.invert-out-<0-7>` | (bit, R/W) | When True, pin value will be inverted. Used when pin is configured as output. |

### 6.3.2 Axis connectors

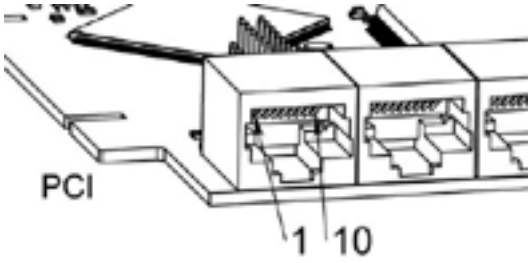



Figure 6.4: Pin numbering of axis connectors

Table 6.6: Pinout of axis connectors

| | |
|---|---|
| 1 | Encoder A |
| 2 | +5 Volt (PC) |
| 3 | Encoder B |
| 4 | Encoder Index |
| 5 | Fault |
| 6 | Power Enabled |
| 7 | Step/CCW/B |
| 8 | Direction/CW/A |
| 9 | Ground (PC) |
| 10 | DAC serial line |

#### 6.3.2.1 Axis interface modules

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.

For evaluating the appropriate servo-drive structure the modules have to be connected as the following block diagram shows:

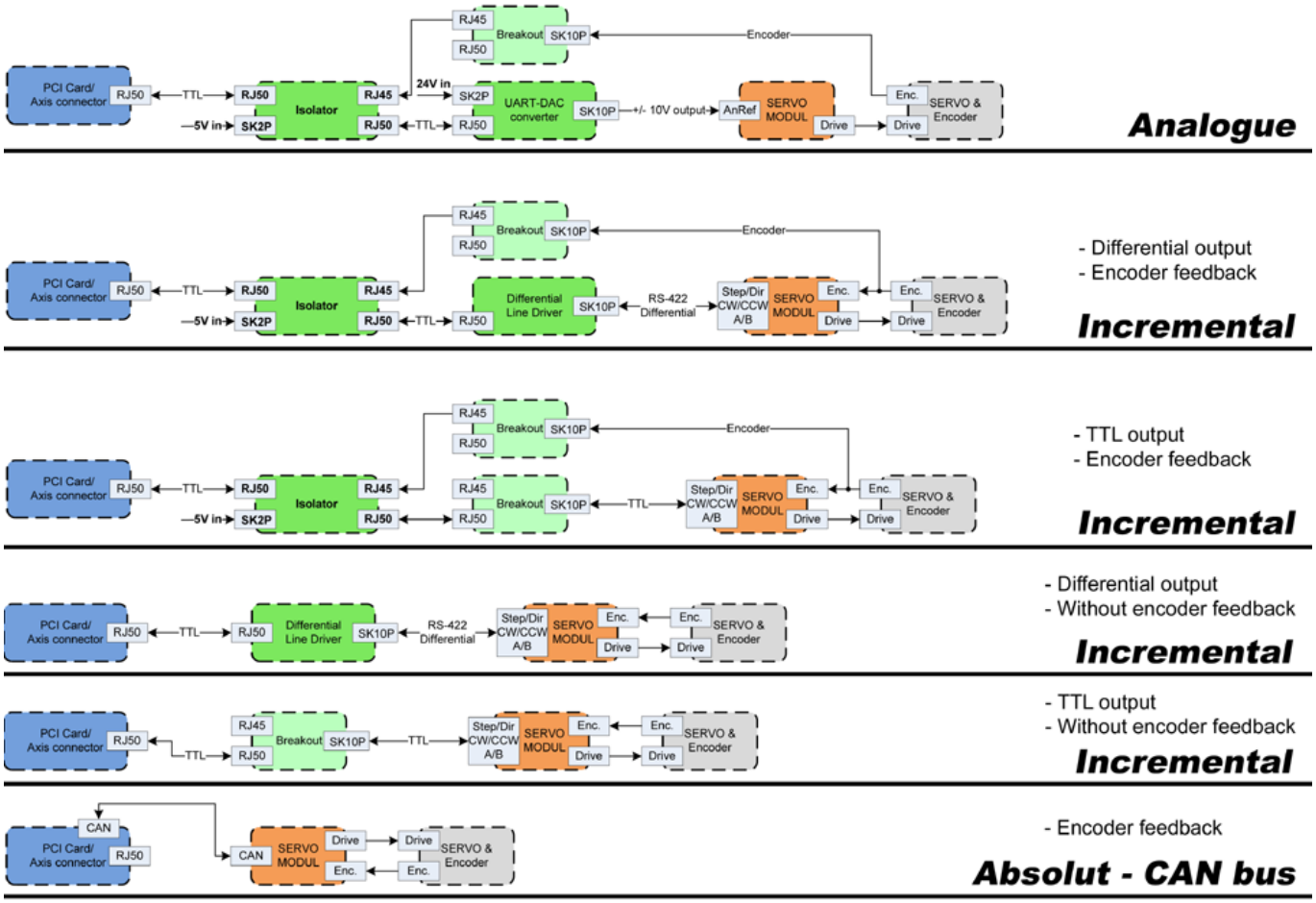



Figure 6.5: Servo axis interfaces

#### 6.3.2.2 Энкодер

The GM6-PCI motion control card has six encoder modules. Each encoder module has three channels:

- Channel-A
- Channel-B
- Channel-I (index)

It is able to count quadrature encoder signals or step/dir signals. Each encoder module is connected to the inputs of the corresponding RJ50 axis connector.

Every encoder pin and parameter name begins as follows:

```
gm.<card_no>.encoder.<axis_no>
```

where *<axis_no>* is from 0 to 5. For example, `gm.0.encoder.0.position` refers to the position of encoder module of axis 0.

The GM6-PCI card counts the encoder signal independently from LinuxCNC. HAL pins are updated by function:

```
gm.<card_no>.read
```

Table 6.7: Encoder pins

| **Контакты** | **Type and direction** | **Pin description** |
|---|---|---|
| `.reset` | (bit, In) | When True, resets counts and position to zero. |
| `.rawcounts` | (s32, Out) | The raw count is the counts, but unaffected by reset or the index pulse. |
| `.counts` | (s32, Out) | Position in encoder counts. |
| `.position` | (float, Out) | Position in scaled units (=.counts/.position-scale). |
| `.index-enabled` | (bit, IO) | When True, counts and position are rounded or reset (depends on index-mode) on next rising edge of channel-I. Every time position is reset because of Index, the `index-enabled` pin is set to 0 and remains 0 until connected HAL pin does not set it. |
| `.velocity` | (float, Out) | Velocity in scaled units per second. GM encoder uses high frequency hardware timer to measure time between encoder pulses in order to calculate velocity. It greatly reduces quantization noise as compared to simply differentiating the position output. When the measured velocity is below min-speed-estimate, the velocity output is 0. |

Table 6.8: Encoder parameters

| **Параметры** | **Type and Read/Write** | **Parameter description** |
|---|---|---|
| `.counter-mode` | (bit, R/W) | When True, the counter counts each rising edge of the channel-A input to the direction determined by channel-B. This is useful for counting the output of a single channel (non-quadrature) or step/dir signal sensor. When false, it counts in quadrature mode. |
| `.index-mode` | (bit, R/W) | When True and .index-enabled is also true, .counts and .position are rounded (based on .counts-per-rev) at rising edge of channel-I. This is useful to correct few pulses error caused by noise. In round mode, it is essential to set .counts-per-rev parameter correctly. When .index-mode is False and .index-enabled is true, .counts and .position are reset at channel-I pulse. |
| `.counts-per-rev` | (s32, R/V) | Determine how many counts are between two index pulses. It is used only in round mode, so when both .index-enabled and .index-mode parameters are True. GM encoder process encoder signal in 4x mode, so for example in case of a 500 CPR encoder it should be set to 2000. This parameter can be easily measured by setting .index-enabled True and .index-mode False (so that .counts resets at channel-I pulse), than move axis by hand and see the maximum magnitude of .counts pin in halmeter. |
| `.index-invert` | (bit, R/W) | When True, channel-I event (reset or round) occur on falling edge of channel-I signal, otherwise on rising edge. |
| `.min-speed-estimate` | (float, R/W) | Determine the minimum measured velocity magnitude at which .velocity will be set as nonzero. Setting this parameter too low will cause it to take a long time for velocity to go to zero after encoder pulses have stopped arriving. |
| `.position-scale` | (float, R/W) | Scale in counts per length unit. .position=.counts/.position-scale. For example, if position-scale is 2000, then 1000 counts of the encoder will produce a position of 0.5 units. |

**Setting encoder module of axis 0 to receive 500 CPR quadrature encoder signal and use reset to round position.**

```
setp gm.0.encoder.0.counter-mode 0         # 0: quad, 1: stepDir
setp gm.0.encoder.0.index-mode 1           # 0: reset pos at index, 1:round pos at index
setp gm.0.encoder.0.counts-per-rev 2000    # GM process encoder in 4x mode, 4x500=2000
setp gm.0.encoder.0.index-invert 0         #
setp gm.0.encoder.0.min-speed-estimate 0.1 # in position unit/s
setp gm.0.encoder.0.position-scale 20000   # 10 encoder rev cause the machine to move one  ↩
    position unit (10x2000)
```

**Connect encoder position to LinuxCNC joint position feedback**

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

#### 6.3.2.3 StepGen module

The GM6-PCI motion control card has six StepGen modules, one for each joint. Each module has two output signals. It can produce Step/Direction, Up/Down or Quadrature (A/B) pulses. Each StepGen module is connected to the pins of the corresponding RJ50 axis connector.

Every StepGen pin and parameter name begins as follows:

```
gm.<card_no>.stepgen.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.stepgen.0.position-cmd` refers to the position command of StepGen module of axis 0 on card 0.

The GM6-PCI card generates step pulses independently from LinuxCNC. HAL pins are updated by function

```
gm.<card_no>.write
```

Table 6.9: StepGen module pins

| **Контакты** | **Type and direction** | **Pin description** |
|---|---|---|
| `.enable` | (bit, In) | StepGen produces pulses only when this pin is true. |
| `.count-fb` | (s32, Out) | Position feedback in counts unit. |
| `.position-fb` | (float, Out) | Position feedback in position unit. |
| `.position-cmd` | (float, In) | Commanded position in position units. Used in position mode only. |
| `.velocity-cmd` | (float, In) | Commanded velocity in position units per second. Used in velocity mode only. |

Table 6.10: StepGen module parameters

| **Параметры** | **Type and Read/Write** | **Parameter description** |
|---|---|---|
| `.step-type` | (u32, R/W) | When 0, module produces Step/Dir signal. When 1, it produces Up/Down step signals. And when it is 2, it produces quadrature output signals. |

Table 6.10: (continued)

| **Параметры** | **Type and Read/Write** | **Parameter description** |
|---|---|---|
| `.control-type` | (bit, R/W) | When True, .velocity-cmd is used as reference and velocityvcontrol calculate pulse rate output. When False, .position-cmd is used as reference and position control calculate pulse rate output. |
| `.invert-step1` | (bit, R/W) | Invert the output of channel 1 (Step signal in StepDir mode) |
| `.invert-step2` | (bit, R/W) | Invert the output of channel 2 (Dir signal in StepDir mode) |
| `.maxvel` | (float, R/W) | Maximum velocity in position units per second. If it is set to 0.0, .maxvel parameter is ignored. |
| `.maxaccel` | (float, R/W) | Maximum acceleration in position units per second squared. mf it is set to 0.0, .maxaccel parameter is ignored. |
| `.position-scale` | (float, R/W) | Scale in steps per length unit. |
| `.steplen` | (u32, R/W) | Length of step pulse in nano-seconds. |
| `.stepspace` | (u32, R/W) | Minimum time between two step pulses in nano-seconds. |
| `.dirdelay` | (u32, R/W) | Minimum time between step pulse and direction change in nanoseconds. |

For evaluating the appropriate values see the timing diagrams below:

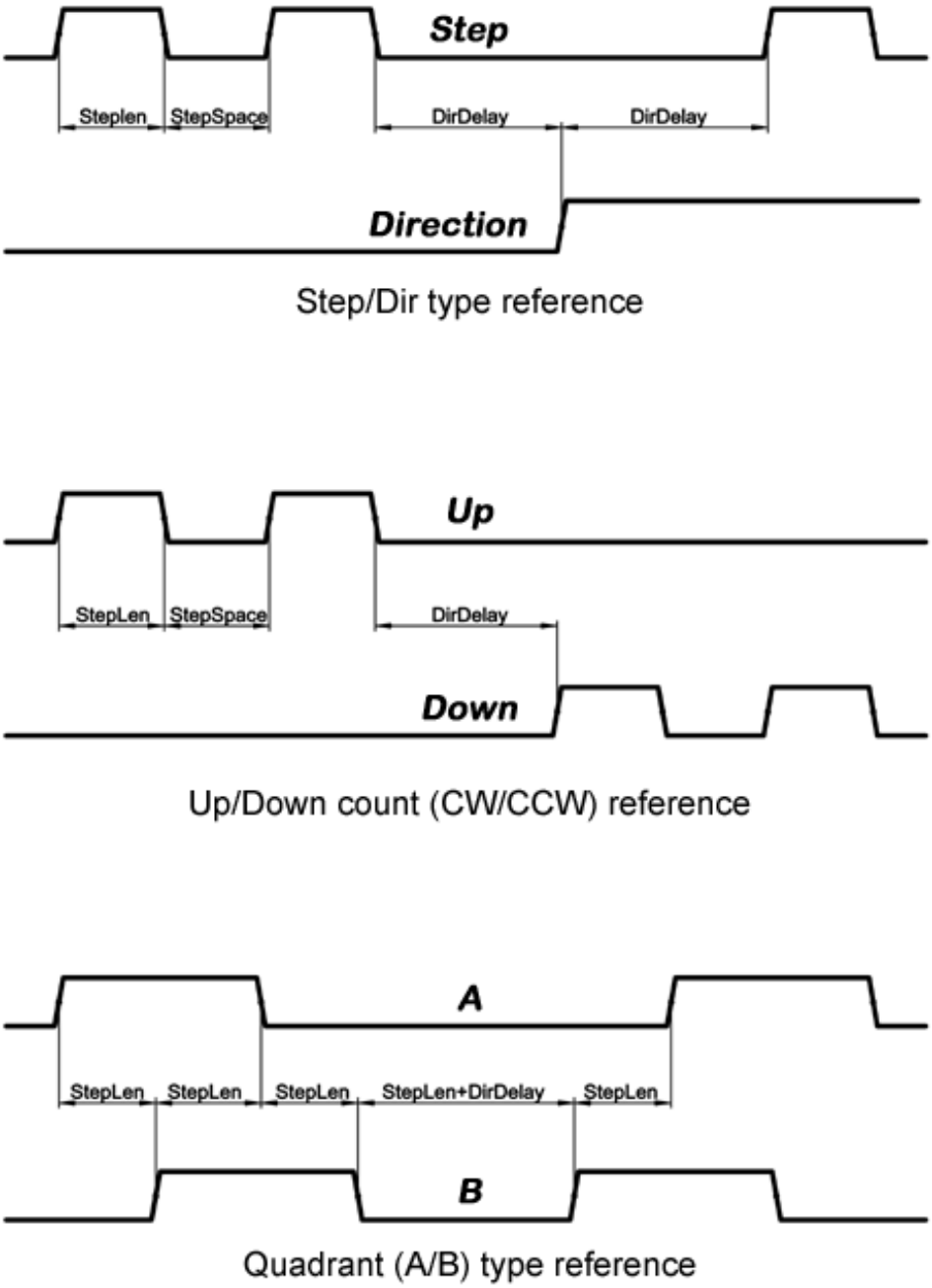



Figure 6.6: Reference signal timing diagrams

**Setting StepGen module of axis 0 to generate 1000 step pulse per position unit**

```
setp gm.0.stepgen.0.step-type 0         # 0:stepDir, 1:UpDown, 2:Quad
setp gm.0.stepgen.0.control-type 0      # 0:Pos. control, 1:Vel. Control
setp gm.0.stepgen.0.invert-step1 0
setp gm.0.stepgen.0.invert-step2 0
setp gm.0.stepgen.0.maxvel 0            # do not set maxvel for step
                                        # generator, let interpolator control it.
setp gm.0.stepgen.0.maxaccel 0          # do not set max acceleration for
                                        # step generator, let interpolator control it.
setp gm.0.stepgen.0.position-scale 1000 # 1000 step/position unit
setp gm.0.stepgen.0.steplen 1000        # 1000 ns = 1 µs
setp gm.0.stepgen.0.stepspace1000       # 1000 ns = 1 µs
```

```
setp gm.0.stepgen.0.dirdelay 2000       # 2000 ns = 2 µs
```

**Connect StepGen to axis 0 position reference and enable pins**

```
net Xpos-cmd joint.0.motor-pos-cmd => gm.0.stepgen.0.position-cmd
net Xen joint.0.amp-enable-out => gm.0.stepgen.0.enable
```

#### 6.3.2.4 Enable and Fault signals

The GM6-PCI motion control card has one enable output and one fault input HAL pins, both are connected to each RJ50 axis connector and to the CAN connector.

HAL pins are updated by function:

```
gm.<card_no>.read
```

Table 6.11: Enable and Fault signal pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| gm.<card_no>.power-enable | (bit, In) | If this pin is True,<br>* and Watch Dog Timer is not expired<br>* and there is no power fault<br>then power enable pins of axis- and CAN connectors are set to high, otherwise set to low. |
| gm.<card_no>.power-fault | (bit, Out) | Power fault input. |

#### 6.3.2.5 Axis DAC

The GM6-PCI motion control card has six serial axis DAC driver modules, one for each joint. Each module is connected to the pin of the corresponding RJ50 axis connector. Every axis DAC pin and parameter name begins as follows:

```
gm.<card_no>.dac.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.dac.0.value` refers to the output voltage of DAC module of axis 0.

HAL pins are updated by function:

```
gm.<card_no>.write
```

Table 6.12: Axis DAC pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| .enable | (bit, In) | Enable DAC output. When enable is false, DAC output is 0.0 V. |

Table 6.12: (continued)

| Контакты | Type and direction | Pin description |
|---|---|---|
| .value | (float, In) | Value of DAC output in Volts. |

Table 6.13: Axis DAC parameters

| Параметры | Type and direction | Parameter description |
|---|---|---|
| .offset | (float, R/W) | Offset is added to the value before the hardware is updated. |
| .high-limit | (float, R/W) | Maximum output voltage of the hardware in Volts. |
| .low-limit | (float, R/W) | Minimum output voltage of the hardware in Volts. |
| .invert-serial | (float, R/W) | GM6-PCI card is communicating with DAC hardware via fast serial communication to highly reduce time delay compared to PWM. DAC module is recommended to be isolated which is negating serial communication line. In case of isolation, leave this parameter to default (0), while in case of none-isolation, set this parameter to 1. |

### 6.3.3 CAN-bus servo amplifiers

The GM6-PCI motion control card has CAN module to drive CAN servo amplifiers. Implementation of higher level protocols like CANopen is further development. Currently GM produced power amplifiers has upper level driver which export pins and parameters to HAL. They receive position reference and provide encoder feedback via CAN bus.

The frames are standard (11 bit) ID frames, with 4 byte data length. The baud rate is 1 Mbit/s. The position command IDs for axis 0..5 are 0x10..0x15. The position feedback IDs for axis 0..5 are 0x20..0x25.

These configuration can be changed with the modification of hal_gm.c and recompiling LinuxCNC.

Every CAN pin and parameter name begins as follows:

```
gm.<card_no>.can-gm.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.can-gm.0.position` refers to the output position of axis 0 in position units.

HAL pins are updated by function:

```
gm.<card_no>.write
```

#### 6.3.3.1 Контакты

Table 6.14: CAN module pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| .enable | (bit, In) | Enable sending position references. |
| .position-cmd | (float, In) | Commanded position in position units. |
| .position-fb | (float, In) | Feed back position in position units. |

#### 6.3.3.2 Параметры

Table 6.15: CAN module parameters

| Параметры | Type and direction | Parameter description |
|---|---|---|
| .position-scale | (float, R/W) | Scale in per length unit. |

### 6.3.4 Watchdog timer

Watchdog timer resets at function:

```
gm.<card_no>.read
```

#### 6.3.4.1 Контакты

Table 6.16: Watchdog pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| gm.<*card_no*>.watchdog-expired | (bit, Out) | Indicates that watchdog timer is expired. |

Watchdog timer overrun causes the set of power-enable to low in hardware.

#### 6.3.4.2 Параметры

Table 6.17: Watchdog parameters

| Параметры | Type and direction | Parameter description |
|---|---|---|
| gm.<*card_no*>.watchdog-enable | (bit, R/W) | Enables watchdog timer.<br>It is strongly recommended to enable the watchdog timer, because it can disable all the servo amplifiers by pulling down all enable signals in case of a PC error. |
| gm.<*card_no*>.watchdog-timeout | (float, R/W) | Time interval in within the gm.<*card_no*>.read function must be executed. The gm.<*card_no*>.read is typically added to servo-thread, so watch timeout is typically set to 3 times of the servo period. |

### 6.3.5 End-, homing- and E-stop switches

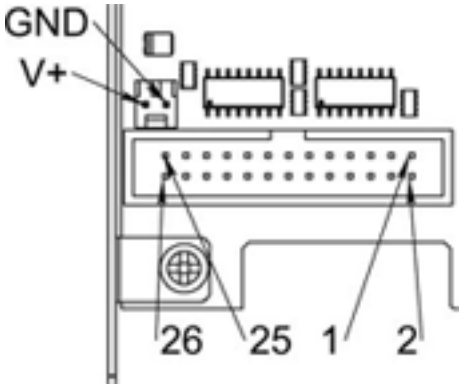



Figure 6.7: Pin numbering of homing & end switch connector

Table 6.18: End- and homing switch connector pinout

| 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 | E-Stop 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- | E-Stop 1 | V+ (Ext.) |

The GM6-PCI motion control card has two limit- and one homing switch input for each joint. All the names of these pins begin as follows:

```
gm.<card_no>.joint.<axis_no>
```

where <*axis_no*> is from 0 to 5. For example, `gm.0.joint.0.home-sw-in` indicates the state of the axis 0 home switch.

HAL pins are updated by function:

```
gm.<card_no>.read
```

#### 6.3.5.1 Контакты

Table 6.20: End- and homing switch pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| `.home-sw-in` | (bit, Out) | Home switch input |
| `.home-sw-in-not` | (bit, Out) | Negated home switch input |
| `.neg-lim-sw-in` | (bit, Out) | Negative limit switch input |
| `.neg-lim-sw-in-not` | (bit, Out) | Negated negative limit switch input |
| `.pos-lim-sw-in` | (bit, Out) | Positive limit switch input |
| `.pos-lim-sw-in-not` | (bit, Out) | Negated positive limit switch input |

#### 6.3.5.2 Параметры

Table 6.21: E-stop switch parameters

| Параметры | Type and direction | Parameter description |
|---|---|---|
| `gm.0.estop.0.in` | (bit, Out) | Estop 0 input |
| `gm.0.estop.0.in-not` | (bit, Out) | Negated Estop 0 input |
| `gm.0.estop.1.in` | (bit, Out) | Estop 1 input |
| `gm.0.estop.1.in-not` | (bit, Out) | Negated Estop 1 input |

### 6.3.6 Status LEDs

#### 6.3.6.1 CAN

Color: Orange

- Blink, during data communication.
- On, when any of the buffers are full - communication error.
- Off, when no data communication.

#### 6.3.6.2 RS485

Color: Orange

- Blink, during initialization of modules on the bus
- On, when the data communication is up between all initialized modules.
- Off, when any of the initialized modules dropped off because of an error.

#### 6.3.6.3 EMC

Color: White

- Blink, when LinuxCNC is running.
- Otherwise off.

#### 6.3.6.4 Boot

Color: Green

- On, when system booted successfully.
- Otherwise off.

#### 6.3.6.5 Error

Color: Red

- Off, when there is no fault in the system.
- Blink, when PCI communication error.
- On, when watchdog timer overflowed.

### 6.3.7 RS485 I/O expander modules

These modules were developed for expanding the I/O and function capability along an RS485 line of the GM6-PCI motion control card.

Available module types:

- 8-channel relay output module - gives eight NO-NC relay output on a three pole terminal connector for each channel.
- 8-channel digital input module - gives eight optical isolated digital input pins.
- 8 channel ADC and 4-channel DAC module - gives four digital-to-analogue converter outputs and eight analogue-to-digital inputs. This module is also optically isolated from the GM6-PCI card.

**Automatic node recognizing** Each node connected to the bus was recognized by the GM6-PCI card automatically. During starting LinuxCNC, the driver export pins and parameters of all available modules automatically.

**Fault handling** If a module does not answer regularly the GM6-PCI card drops down the module. If a module with output do not gets data with correct CRC regularly, the module switch to error state (green LED blinking), and turns all outputs to error state.

**Connecting the nodes** The modules on the bus have to be connected in serial topology, with termination resistors on the end. The start of the topology is the PCI card, and the end is the last module.

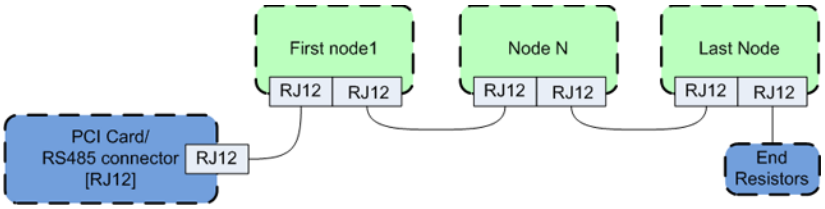



Figure 6.8: Connecting the RS485 nodes to the GM6-PCI card

**Addressing** Each node on the bus has a 4 bit unique address that can be set with a red DIP switch.

**Status LED** A green LED indicates the status of the module:

- Blink, when the module is only powered, but not jet identified, or when module is dropped down.
- Off, during identification (computer is on, but LinuxCNC not started)
- On, when it communicates continuously.

#### 6.3.7.1 Relay output module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

All the pins and parameters are updated by the following function:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Table 6.22: Relay output module pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| `.relay-<0-7>` | (bit, Out) | Output pin for relay |

Table 6.23: Relay output module parameters

| Параметры | Type and direction | Parameter description |
|---|---|---|
| .invert-relay-<0-7> | (bit, R/W) | Negate relay output pin |

**HAL example**

```
  gm.0.rs485.0.relay-0   # First relay of the node.
# gm.0                   # Identifies the first GM6-PCI motion control card (PCI card  ↩
    address = 0)
#     .rs485.0           # Selects node with address 0 on the RS485 bus
#             .relay-0   # Selects the first relay
```

#### 6.3.7.2 Digital input module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

All the pins and parameters are updated by the following function:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Table 6.24: Digital input output module pins

| Контакты | Type and direction | Pin description |
|---|---|---|
| .in-<*0-7*> | (bit, Out) | Input pin |
| .in-not-<*0-7*> | (bit, Out) | Negated input pin |

**HAL example**

```
  gm.0.rs485.0.in-0   # First input of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card address  ↩
    = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .in-0   # Selects the first digital input module
```

#### 6.3.7.3 DAC & ADC module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

All the pins and parameters are updated by the following function:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where <*module ID*> is from 00 to 15.

Table 6.25: DAC & ADC module pins

| **Контакты** | **Type and direction** | **Pin description** |
|---|---|---|
| `.adc-<0-7>` | (float, Out) | Value of ADC input in Volts. |
| `.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) | Value of DAC output in Volts. |

Table 6.26: DAC & ADC module parameters

| **Параметры** | **Type and direction** | **Parameter description** |
|---|---|---|
| `.adc-scale-<0-7>` | (float, R/W) | The input voltage will be multiplied by scale before being output to .adc- pin. |
| `.adc-offset-<0-7>` | (float, R/W) | Offset is subtracted from the hardware input voltage after the scale multiplier has been applied. |
| `.dac-offset-<0-3>` | (float, R/W) | Offset is added to the value before the hardware is updated. |
| `.dac-high-limit-<0-3>` | (float, R/W) | Maximum output voltage of the hardware in Volts. |
| `.dac-low-limit-<0-3>` | (float, R/W) | Minimum output voltage of the hardware in Volts. |

**HAL example**

```
  gm.0.rs485.0.adc-0   # First analogue channel of the node.
# gm.0                 # Identifies the first GM6-PCI motion control card (PCI card address ↩
    = 0)
#     .rs485.0         # Selects node with address 0 on the RS485 bus
#             .adc-0   # Selects the first analogue input of the module
```

#### 6.3.7.4 Teach Pendant module

For pinout, connection and electrical charasteristics of the module, please refer to the System integration manual.

All the pins and parameters are updated by the following function:

```
gm.<card_no>.rs485
```

It should be added to servo thread or other thread with larger period to avoid CPU overload. Every RS485 module pin and parameter name begins as follows:

```
gm.<card_no>.rs485.<module ID>
```

where *<module ID>* is from 00 to 15. Note that on the Teach Pendant module it cannot be changed, and pre-programmed as zero. Upon request it can be delivered with firmware pre-programmed different ID.

Table 6.27: Teach Pendant module pins

| **Контакты** | **Type and direction** | **Pin description** |
|---|---|---|
| `.adc-<0-5>` | (float, Out) | Value of ADC input in Volts. |
| `.enc-reset` | (bit, In) | When True, resets counts and position to zero. |
| `.enc-counts` | (s32, Out) | Position in encoder counts. |
| `.enc-rawcounts` | (s32, Out) | The raw count is the counts, but unaffected by reset. |
| `.enc-position` | (float, Out) | Position in scaled units (=.enc-counts/.enc-position-scale). |
| `.in-<0-7>` | (bit, Out) | Input pin |
| `.in-not-<0-7>` | (bit, Out) | Negated input pin |

Table 6.28: Teach Pendant module parameters

| **Параметры** | **Type and direction** | **Parameter description** |
|---|---|---|
| `.adc-scale-<0-5>` | (float, R/W) | The input voltage will be multiplied by scale before being output to .adc- pin. |
| `.adc-offset-<0-5>` | (float, R/W) | Offset is subtracted from the hardware input voltage after the scale multiplier has been applied. |
| `.enc-position-scale` | (float, R/W) | Scale in per length unit. |

**HAL example**

```
  gm.0.rs485.0.adc-0  # First analogue channel of the node.
# gm.0                # Identifies the first GM6-PCI motion control card (PCI card address
    = 0)
#     .rs485.0        # Selects node with address 0 on the RS485 bus
#             .adc-0  # Selects the first analogue input of the module
```

### 6.3.8 Errata

#### 6.3.8.1 GM6-PCI card Errata

The revision number in this section refers to the revision of the GM6-PCI card device.

Rev. 1.2

- Error: The PCI card do not boot, when Axis 1. END B switch is active (low). Found on November 16, 2013.
- Reason: This switch is connected to a boot setting pin of FPGA
- Problem fix/workaround: Use other switch pin, or connect only normally open switch to this switch input pin.

## 6.4 GS2 VFD Driver

This is a non-realtime HAL program for the GS2 series of VFDs at Automation Direct. [1]

This component is loaded using the halcmd "loadusr" command:

```
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd
```

The above command says: loadusr, wait for named to load, component gs2_vfd, named spindle-vfd. The HAL `loadusr` command is described in the loadusr chapter.

### 6.4.1 Command Line Options

- *-b* or *--bits* <n> (default: 8) Set number of data bits to *n*, where *n* must be from 5 to 8 inclusive.
- *-d* or *--device <path>* (default: /dev/ttyS0) Set the file path to the serial device node to use.
- *-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* or *--parity {even,odd,none}* (default: odd) Set serial parity to even, odd, or 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.
- *-s* or *--stopbits {1,2}* (default: 1) Set serial stop bits to 1 or 2
- *-t* or *--target* <n> (default: 1) Set MODBUS target (slave) number. This must match the device number you set on the GS2.
- *-v* or *--verbose* Turn on debug messages.
- *-A* or *--accel-seconds <n>* (default: 10.0) Seconds to accelerate the spindle from 0 to max. RPM.
- *-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.

[1] In Europe the equivalent can be found under the brand name Omron.

- *-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.

---

**Note**
That if there are serial configuration errors, turning on verbose may result in a flood of timeout errors.

---

### 6.4.2 Контакты

With <*name*> being ”gs2_vfd” or the name given during loading with the `-n` option:

- <name>.*DC-bus-volts* (float, out) DC bus voltage of the VFD
- <name>.*at-speed* (bit, out) when drive is at commanded speed
- <name>.*err-reset* (bit, in) reset errors sent to VFD
- <name>.*firmware-revision* (s32, out) from the VFD
- <name>.*frequency-command* (float, out) from the VFD
- <name>.*frequency-out* (float, out) from the VFD
- <name>.*is-stopped* (bit, out) when the VFD reports 0 Hz output
- <name>.*load-percentage* (float, out) from the VFD
- <name>.*motor-RPM* (float, out) from the VFD
- <name>.*output-current* (float, out) from the VFD
- <name>.*output-voltage* (float, out) from the VFD
- <name>.*power-factor* (float, out) from the VFD
- <name>.*scale-frequency* (float, out) from the VFD
- <name>.*speed-command* (float, in) speed sent to VFD in RPM It is an error to send a speed faster than the Motor Max RPM as set in the VFD.
- <name>.*spindle-fwd* (bit, in) 1 for FWD and 0 for REV sent to VFD
- <name>.*spindle-rev* (bit, in) 1 for REV and 0 if off
- <name>.*spindle-on* (bit, in) 1 for ON and 0 for OFF sent to VFD
- <name>.*status-1* (s32, out) Drive Status of the VFD (see the GS2 manual)
- <name>.*status-2* (s32, out) Drive Status of the VFD (see the GS2 manual)

---

**Note**
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 Параметры

With <*name*> being gs2_vfd or the name given during loading with the `-n` option:

- <name>.*error-count* (s32, RW)
- <name>.*loop-time* (float, RW) how often the modbus is polled (default: 0.1)
- <name>.*nameplate-HZ* (float, RW) Nameplate Hz of motor (default: 60)
- <name>.*nameplate-RPM* (float, RW) Nameplate RPM of motor (default: 1730)
- <name>.*retval* (s32, RW) the return value of an error in HAL
- <name>.*tolerance* (s32, RW) speed tolerance (default: 0.01)
- <name>.*ack-delay* (s32, RW) number of read/write cycles before checking at-speed (default 2)

For an example of using this component to drive a spindle see the GS2 Spindle example.

## 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 analogous to the parallel port driver on x86 PCs. It can use the same step generators, encoder counters and similar components.

### 6.5.2 Использование

```
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.

Table 6.29: 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 |

Table 6.29: (continued)

| GPIO Num | Decimal | Hex | Pin Num |
|---|---|---|---|
| 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 Контакты

- hal_pi_gpio.pin-*NN*-out
- hal_pi_gpio.pin-*NN*-in

Depending on the dir and exclude masks.

### 6.5.4 Параметры

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 Функции

- `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 Numbering

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 Универсальный драйвер для любого GPIO, поддерживаемог 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 Использование

```
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 Контакты

- 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 Параметры

- 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 Функции

- *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 Идентификация контактов

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 Автор

Andy Pugh

### 6.6.9 Known Bugs

На данный момент нет.

## 6.7 Mesa HostMot2 драйвер

### 6.7.1 Введение

HostMot2 - это конфигурация FPGA, разработанная Mesa Electronics для линейки их плат управления движением *Anything I/O* . Прошивка с открытым исходным кодом, портативная и гибкая. Она может быть сконфигурирована (во время сборки) с нулевым или более экземплярами (объект, создаваемый во время выполнения) каждого из нескольких модулей: энкодеры (квадратурные счетчики), ШИМ-генераторы и генераторы Step/Dir. Прошивка может быть настроена (во время выполнения) для подключения каждого из этих экземпляров к контактам ввода-вывода (I/O). Контакты ввода-вывода, не управляемые экземпляром модуля, возвращаются к цифровому двунаправл вводу-выводу общего назначения.

### 6.7.2 Бинарные файлы прошивки

**Карты FPGA с 50-контактным разъемом** Несколько предкомпилированных HostMot2 бинарных файлов прошивки доступны для различных плат Anything I/O. Этот список является неполным, проверьте источник hostmot2-firmware программ на предмет актуальных списков прошивок.

- 3x20 (144 I/O контактов): используя модуль hm2_pci
  - 24 серво каналов
  - 16 серво-каналов плюс 24 генератора step/dir
- 5I22 (96 I/O контактов): используя модуль hm2_pci
  - 16 серво-каналов
  - 8 серво-каналов плюс 24 генератора step/dir
- 5I20, 5I23, 4I65, 4I68 (72 I/O контакта): используя модуль hm2_pci
  - 12 серво-каналов
  - 8 серво-каналов плюс 4 генератора step/dir
  - 4 серво-канала плюс 8 генераторов step/dir
- 7I43 (48 I/O контактов): используя модуль hm2_7i43
  - 8 серво-каналов (8 ШИМ генераторов и 8 энкодеров)
  - 4 серво-канала плюс 4 генератора step/dir

**DB25 FPGA карты** Карта 5I25 Superport FPGA предварительно запрограммирована при покупке и не требует бинарной прошивки.

### 6.7.3 Установка прошивки

В зависимости от того, как вы установили LinuxCNC, вам, возможно, придется открыть диспетчер пакетов Synaptic из меню «Система» и установить пакет для вашей карты Mesa. Самый быстрый способ их найти — выполнить поиск «hostmot2» в диспетчере пакетов Synaptic. Отметьте прошивку для установки, затем примените.

### 6.7.4 Загрузка HostMot2

Поддержка LinuxCNC прошивки HostMot2 разделена на общий драйвер под названием *hostmot2* и два низкоуровневых драйвера ввода-вывода для плат Anything I/O. Драйверы ввода-вывода низкого уровня — *hm2_7i43* и *hm2_pci* (для всех плат Anything I/O на базе PCI и PC-104/Plus). Сначала необходимо загрузить драйвер hostmot2 с помощью такой команды HAL:

```
loadrt hostmot2
```

Смотрите hostmot2(9) man страницу на предмет подробностей.

Драйвер hostmot2 сам по себе ничего не делает, ему нужен доступ к реальным платам, на которых установлена прошивка HostMot2. Драйверы ввода-вывода низкого уровня обеспечивают такой доступ. Драйверы ввода-вывода низкого уровня загружаются такими командами:

```
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT
       num_encoders=3 num_pwmgens=3 num_stepgens=1"
```

Параметры конфигурации описаны на странице man hostmot2.

### 6.7.5 Сторожевой таймер

Прошивка HostMot2 может включать модуль сторожевого таймера; если это так, то драйвер hostmot2 будет использовать его.

LinuxCNC необходимо периодически ”гладить сторожевого пса”, иначе он ”укусит”. Функция записи hm2 (см. ниже) заботится о сторожевом таймере.

Когда сторожевой таймер срабатывает, все контакты ввода/вывода платы отключаются от своих экземпляров модуля и становятся входами с высоким импедансом (поднимаются до высокого уровня). Состояние модулей прошивки HostMot2 не нарушается (за исключением конфигурации контактов ввода/вывода). Экземпляры энкодеров продолжают считать квадратурные импульсы, а ШИМ- и шаговые генераторы продолжают генерировать сигналы (которые не передаются на двигатели, поскольку контакты ввода-вывода стали входами).

Сброс сторожевого таймера сбрасывает контакты ввода-вывода в конфигурацию, выбранную во время загрузки.

Если прошивка включает сторожевой таймер, будут экспортированы следующие объекты HAL:

#### 6.7.5.1 Контакты

- *has_bit* - (bit i/o) True, если сторожевой таймер сработал, False, если сторожевой таймер не срабатывал. Если сторожевой таймер сработал и бит has_bit имеет значение True, пользователь может сбросить его на False, чтобы возобновить работу.

#### 6.7.5.2 Параметры

- *timeout_ns* - (u32 read/write) Тайм-аут сторожевого таймера в наносекундах. Оно инициализируется значением ”5000000” (5 миллисекунд) во время загрузки модуля. Если между вызовами функции записи hm2 пройдет больше этого времени, сторожевой таймер сработает.

### 6.7.6 HostMot2 Функции

- *hm2_<BoardType>.<BoardNum>.read* — прочитать все входы, обновить входные контакты HAL.
- *hm2_<BoardType>.<BoardNum>.write* — записать все данные выходов .
- *hm2_<BoardType>.<BoardNum>.read_gpio* — чтение только входных контактов GPIO. (Эта функция недоступна на 7I43 из-за ограничений шины EPP.)
- *hm2_<BoardType>.<BoardNum>.write_gpio* — записывает только регистры управления GPIO и выходные контакты. (Эта функция недоступна на 7I43 из-за ограничений шины EPP.)

---

**Note**
Вышеупомянутые функции *read_gpio* и *write_gpio* обычно не нужны, поскольку биты GPIO считываются и записываются вместе со всем остальным в стандартных функциях *чтения* и *записи*, указанных выше, которые обычно выполняются в servo thread.
Функции *read_gpio* и *write_gpio* были предусмотрены на случай, если потребуется очень быстрый (часто обновляемый) ввод-вывод. Эти функции должны выполняться в base thread. Если у вас есть в этом необходимость, отправьте электронное письмо и сообщите нам об этом и о том, какое у вас приложение.

---

### 6.7.7 Распиновка

Драйвер hostmot2 не имеет определенной распиновки. Распиновка берется из прошивки, которую драйвер hostmot2 отправляет на плату Anything I/O. Каждая прошивка имеет разную распиновку, причем распиновка зависит от того, сколько доступных encoders, pwmgen и stepgens используется. Чтобы получить список контактов для вашей конфигурации после загрузки LinuxCNC, введите в окне терминала:

```
dmesg > hm2.txt
```

Полученный текстовый файл будет содержать большое количество информации, а также распиновку HostMot2 и любые сообщения об ошибках и предупреждения.

Чтобы уменьшить беспорядок, очистив буфер сообщений перед загрузкой LinuxCNC, введите в окне терминала следующее:

```
sudo dmesg -c
```

Теперь, когда вы запускаете LinuxCNC, а затем выполняете команду *dmesg > hm2.txt* в терминале, в вашем файле будет только информация с момента загрузки LinuxCNC вместе с распиновкой. Файл будет находиться в текущем каталоге окна терминала. Каждая строка будет содержать имя карты, номер карты, номер контакта ввода-вывода, разъем и контакт, а также использование. Из этой распечатки вы узнаете физические подключения к вашей карте в зависимости от вашей конфигурации.

Пример конфигурации 5I20 :

```
[HOSTMOT2]
DRIVER=hm2_pci
BOARD=5i20
CONFIG="firmware=hm2/5i20/SVST8_4.BIT num_encoders=1 num_pwmgens=1 num_stepgens=3"
```

Приведенная выше конфигурация создала эту распечатку.

```
[ 1141.053386] hm2/hm2_5i20.0: 72 I/O Pins used:
[ 1141.053394] hm2/hm2_5i20.0: IO Pin 000 (P2-01): IOPort
[ 1141.053397] hm2/hm2_5i20.0: IO Pin 001 (P2-03): IOPort
[ 1141.053401] hm2/hm2_5i20.0: IO Pin 002 (P2-05): Encoder #0, pin B (Input)
[ 1141.053405] hm2/hm2_5i20.0: IO Pin 003 (P2-07): Encoder #0, pin A (Input)
[ 1141.053408] hm2/hm2_5i20.0: IO Pin 004 (P2-09): IOPort
[ 1141.053411] hm2/hm2_5i20.0: IO Pin 005 (P2-11): Encoder #0, pin Index (Input)
[ 1141.053415] hm2/hm2_5i20.0: IO Pin 006 (P2-13): IOPort
[ 1141.053418] hm2/hm2_5i20.0: IO Pin 007 (P2-15): PWMGen #0, pin Out0 (PWM or Up) (Output)
[ 1141.053422] hm2/hm2_5i20.0: IO Pin 008 (P2-17): IOPort
[ 1141.053425] hm2/hm2_5i20.0: IO Pin 009 (P2-19): PWMGen #0, pin Out1 (Dir or Down) ( ↩
    Output)
[ 1141.053429] hm2/hm2_5i20.0: IO Pin 010 (P2-21): IOPort
[ 1141.053432] hm2/hm2_5i20.0: IO Pin 011 (P2-23): PWMGen #0, pin Not-Enable (Output)
<snip>...
[ 1141.053589] hm2/hm2_5i20.0: IO Pin 060 (P4-25): StepGen #2, pin Step (Output)
[ 1141.053593] hm2/hm2_5i20.0: IO Pin 061 (P4-27): StepGen #2, pin Direction (Output)
[ 1141.053597] hm2/hm2_5i20.0: IO Pin 062 (P4-29): StepGen #2, pin (unused) (Output)
[ 1141.053601] hm2/hm2_5i20.0: IO Pin 063 (P4-31): StepGen #2, pin (unused) (Output)
[ 1141.053605] hm2/hm2_5i20.0: IO Pin 064 (P4-33): StepGen #2, pin (unused) (Output)
[ 1141.053609] hm2/hm2_5i20.0: IO Pin 065 (P4-35): StepGen #2, pin (unused) (Output)
[ 1141.053613] hm2/hm2_5i20.0: IO Pin 066 (P4-37): IOPort
[ 1141.053616] hm2/hm2_5i20.0: IO Pin 067 (P4-39): IOPort
[ 1141.053619] hm2/hm2_5i20.0: IO Pin 068 (P4-41): IOPort
[ 1141.053621] hm2/hm2_5i20.0: IO Pin 069 (P4-43): IOPort
[ 1141.053624] hm2/hm2_5i20.0: IO Pin 070 (P4-45): IOPort
[ 1141.053627] hm2/hm2_5i20.0: IO Pin 071 (P4-47): IOPort
[ 1141.053811] hm2/hm2_5i20.0: registered
[ 1141.053815] hm2_5i20.0: initialized AnyIO board at 0000:02:02.0
```

**Note**
Что контакт ввода-вывода nnn будет соответствовать номеру контакта, показанному на экране конфигурации HAL для GPIO. Некоторые из StepGen, Encoder и PWMGen также будут отображаться как GPIO на экране конфигурации HAL.

### 6.7.8 PIN Файлы

Распиновка по умолчанию описана в файле .PIN (читабельный текст). При установке пакета прошивки файлы .PIN устанавливаются в

```
/usr/share/doc/hostmot2-firmware-<board>/
```

### 6.7.9 Прошивка

Выбранная прошивка (файл .BIT) и конфигурация загружаются с материнской платы ПК на материнскую карту Mesa при запуске LinuxCNC. Если вы используете Run In Place, вам все равно необходимо установить пакет hostmot2-firmware-<board>. Более подробная информация о прошивке и настройке находится в разделе *Конфигурации*.

### 6.7.10 HAL Контакты

Контакты HAL для каждой конфигурации можно увидеть, открыв *Show HAL Configuration* в меню *Machine*. Там можно найти все контакты и параметры HAL. На следующем рисунке показана конфигурация 5I20, использованная выше.

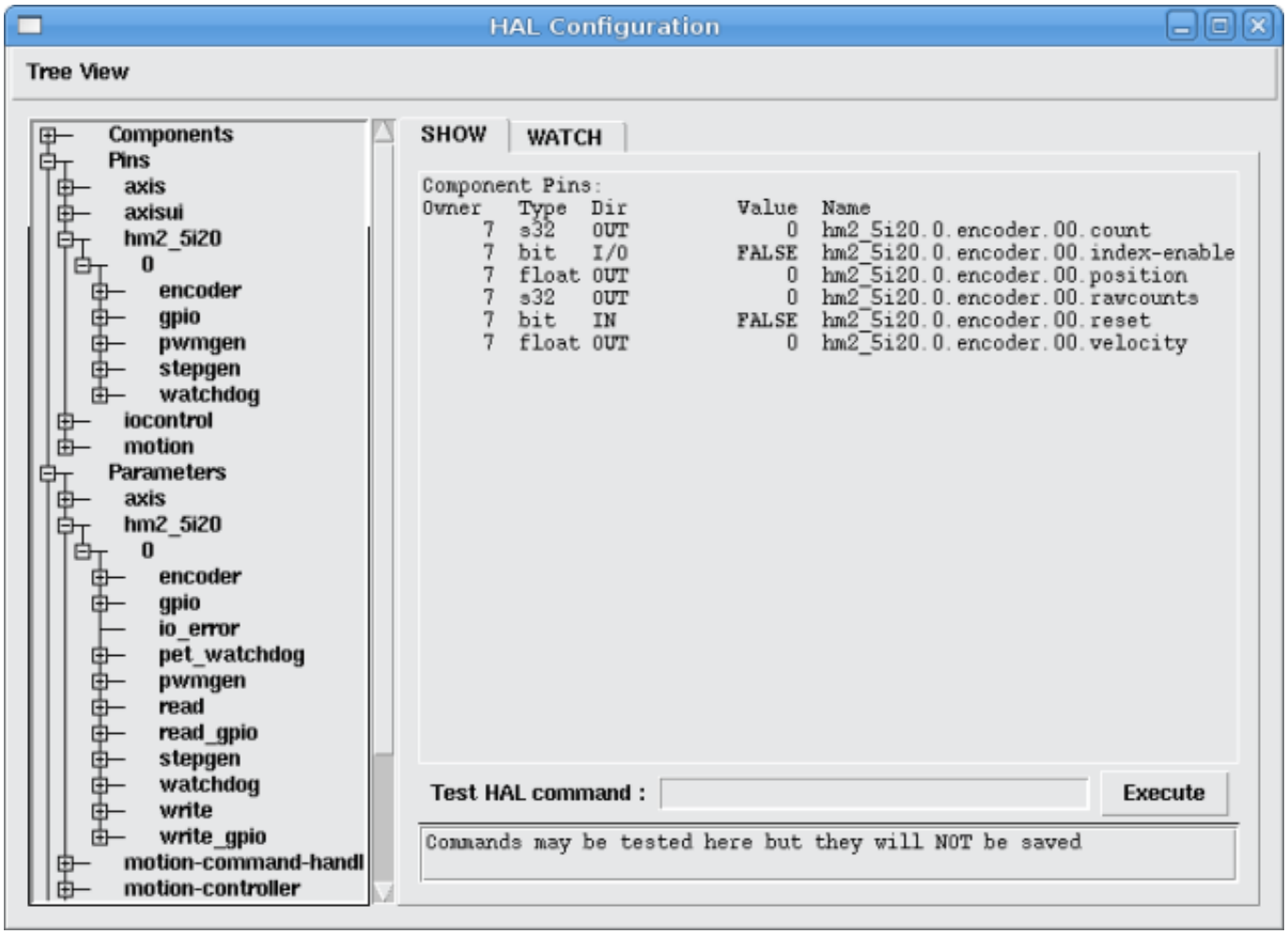



Figure 6.9: 5i20 HAL Контакты

### 6.7.11 Конфигурации

Прошивка Hostmot2 доступна в нескольких версиях, в зависимости от того, чего вы пытаетесь достичь. Вы можете получить напоминание о том, для чего нужна конкретная прошивка, посмотрев на название. Давайте посмотрим на пару примеров.

В 7I43 (два порта) SV8 (*Servo 8*) будет рассчитан на 8 или меньше сервоприводов с использованием *классической* 4-осевой (на порт) сервоплаты 7I33. Таким образом, 8 сервоприводов будут использовать все 48 сигналов в двух портах. Но если вам нужно только 3 сервопривода, вы можете сказать *num_encoders=3* и *num_pwmgens=3* и восстановить 5 сервоприводов по 6 сигналов каждый, получив таким образом 30 бит GPIO.

Или, в 5I22 (четыре порта), SVST8_24 (*Servo 8, Stepper 24*) будет для 8 сервоприводов или меньше (снова 7I33 x2) и 24 шаговых двигателей или меньше (7I47 x2). Это приведет к использованию всех четырех портов. Если вам нужно только 4 сервопривода, вы можете сказать *num_encoders=4* и *num_pwmgens=4* и восстановить 1 порт (и сохранить 7I33). А если вам нужно всего 12 степперов, вы можете сказать *num_stepgens=12* и освободить один порт (и сохранить 7I47). Таким образом, мы можем сохранить два порта (48 бит) для GPIO.

Вот таблицы прошивок, доступных в официальных пакетах. На веб-сайте Mesanet.com могут быть доступны дополнительные прошивки, которые еще не вошли в официальные пакеты прошивок LinuxCNC, поэтому проверьте и их.

3x20 (различные 6 портов) Конфигурации по умолчанию (3x20 поставляется в версиях вентилей 1M, 1,5M и 2M. На данный момент вся прошивка доступна для всех размеров вентилей.)

| Прошивка | Энкодер | PWMGen | StepGen | GPIO |
|---|---|---|---|---|
| SV24 | 24 | 24 | 0 | 0 |
| SVST16_24 | 16 | 16 | 24 | 0 |

5I22 (4-порта PCI) Конфигурации по умолчанию (5I22 поставляется в версиях с воротами 1M и 1,5M. На данный момент вся прошивка доступна для всех размеров вентилей.)

| Прошивка | Энкодер | 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-порта PCI) Конфигурации по умолчанию (5I23 имеет 400 тыс. вентилей.)

| Прошивка | Энкодер | 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-порта PCI) Конфигурации по умолчанию (5I20 имеет 200 тыс. вентилей.)

| Прошивка | Энкодер | 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-порта PC/104) Конфигурации по умолчанию (4I68 имеет 400 тыс. вентилей.)

| Прошивка | Энкодер | 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-порта PC/104) Конфигурации по умолчанию (4I65 имеет 200 тыс. вентилей.)

| Прошивка | Энкодер | PWM | StepGen | GPIO |
|---|---|---|---|---|
| SV12 | 12 | 12 | 0 | 0 |
| SVST8_4 | 8 | 8 | 4 | 0 |
| SVST8_4IM2 | 8 (+IM) | 8 | 4 | 8 |

7I43 (2 параллельных порта) версии с 400k вентилей, Конфигурации по умолчанию

| Прошивка | Энкодер | 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 порта, параллельный порт) версии 200k вентилей, конфигурации по умолчанию

| Прошивка | Энкодер | 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 |

Даже если несколько карт могут иметь одинаково именованные .BIT файлы, вы не можете использовать .BIT файл от другой карты. Разные карты имеют разные тактирующие частоты, так что будьте уверены, что вы используете корректный .BIT файл для вашей карты. Кастомные прошивки hm2 могут быть созданы для специальных приложений и вы можете увидеть некоторые кастомные прошивки в директориях с прошивками по умолчанию.

Когда вы загружаете драйвер карты (hm2_pci или hm2_7i43), вы можете указать запретить экземпляры трех основных модулей (pwmgen, stepgen и encoder) уменьшением количества. Любые контакты ввода/вывода принадлежавших к запрещенным модулям экземпляров становятся вводом/выводом общего назначения (GPIO).

## 6.7.12 GPIO

Контакты ввода-вывода общего назначения на плате, которые не используются экземпляром модуля, экспортируются в HAL как *полные* контакты GPIO. Контакты с полным набором GPIO могут быть настроены во время выполнения как входы, выходы или открытые стоки и иметь интерфейс HAL, обеспечивающий такую гибкость. Контакты ввода-вывода, принадлежащие экземпляру активного модуля, ограничены требованиями модуля-владельца и имеют ограниченный интерфейс HAL.

Контакты GPIO имеют имена типа *hm2_<BoardType>.<BoardNum>.gpio.<IONum>*. *IONum* — трехзначное число. Сопоставление IONum с разъемом и контактом на этом разъеме записывается в системный журнал при загрузке драйвера и описано в руководстве Mesa для плат Anything I/O.

Представление hm2 GPIO смоделировано по образцу цифровых входов и цифровых выходов, описанных в интерфейсе стандартных устройств (часть документа HAL Общий справочник).

Контакты GPIO по умолчанию являются входами.

### 6.7.12.1 Контакты

- *in* - (Bit, Out) Нормальное состояние аппаратного входного контакта. Этот контакт есть как у полных GPIO контактов , так и у контактов ввода/вывода, используемых в качестве входов активными экземплярами модулей.

- *in_not* - (Bit, Out) Инвертированное состояние аппаратного входного контакта. Этот контакт есть как у полных GPIO контактов, так и у контактов ввода/вывода, используемых в качестве входов активными экземплярами модулей.
- *out* - (Bit, In) Значение, которое будет записано (возможно, инвертировано) на выходной контакт оборудования. Этот контакт есть только у полных GPIO контактов .

#### 6.7.12.2 Параметры

- *invert_output* — (Bit, RW) Этот параметр имеет эффект только в том случае, если параметр *is_output* имеет значение true. Если этот параметр имеет значение true, выходное значение GPIO будет обратным значению на контакте HAL *out*. Этот параметр имеют только полные GPIO контакты и контакты ввода-вывода, используемые в качестве выходов активными экземплярам модуля. Чтобы инвертировать контакт активного модуля, вам необходимо инвертировать контакт GPIO, а не контакт модуля.
- *is_opendrain* - (Bit, RW) Этот параметр имеет эффект только в том случае, если параметр *is_output* имеет значение true. Если этот параметр имеет значение false, GPIO ведет себя как обычный выходной контакт: контакт ввода-вывода на разъеме переключается на значение, указанное выходным контактом HAL (возможно, инвертированное), а значения *in* и *in_not* контактов HAL не определены. Если этот параметр имеет значение true, GPIO ведет себя как контакт с открытым стоком. Запись 0 на контакт HAL *out* переводит контакт ввода-вывода в низкий уровень, запись 1 на контакт HAL *out* переводит контакт ввода-вывода в состояние с высоким импедансом. В этом состоянии с высоким импедансом контакт ввода-вывода является *плавающим* (слабо подтянут к высокому уровню), и другие устройства могут управлять значением; результирующ значение на контакте ввода-вывода доступно на контактах *in* и *in_not*. Этот параметр имеют только полные GPIO контакты и контакты ввода-вывода, используемые в качестве выходов активными экземплярами модуля.
- *is_output* — (Bit, RW). Если установлено значение 0, GPIO является входом. Контакт ввода/вывода переводится в состояние с высоким импедансом (слабо подтянутый вверх), чтобы управляться другими устройствами. Логическое значение на контакте ввода-вывода доступно на выводах HAL «in» и «in_not». Запись на контакт HAL *out* не имеет никакого эффекта. Если для этого параметра установлено значение 1, GPIO является выходом; тогда его поведение зависит от параметра *is_opendrain*. Этот параметр есть только у полных GPIO контактов.

### 6.7.13 StepGen

StepGen'ы имеют имена типа *hm2_<BoardType>.<BoardNum>.stepgen.<Instance>*. *Instance* — это двузначное число, соответствующее номеру экземпляра Stepgen HostMot2. Существуют экземпляры *num_stepgens*, начиная с 00.

Каждый степгенератор выделяет 2–6 контактов ввода-вывода (выбирается во время компиляции прошивки), но в настоящее время использует только два: выходы Step и Direction. footnote: [В настоящее время прошивка поддерживает выходы многофазных шаговых двигателей, но драйвер этого не делает. Приглашаются заинтересованные добровольцы.]

Представление StepGen смоделировано на основе программного компонента stepgen. По умолчанию StepGen имеет активным высоким выход step (высокий во время длительности импульса, низкий во время паузы). Чтобы инвертировать выходной контакт StepGen, вы инвертируете соответствующий контакт GPIO, который используется StepGen. Чтобы найти контакт GPIO, используемый для вывода StepGen, запустите *dmesg*, как показано выше.

Каждый экземпляр StepGen имеет следующие контакты и параметры:

#### 6.7.13.1 Контакты

- *control-type* - (Bit, In) Переключает между режимом управления положением (0) и режимом управления скоростью (1). По умолчанию управление положением (0).
- *counts* - (s32, Out) Положение обратной связи в отсчетах (количество шагов).
- *enable* - (Bit, In) Разрешает вывод импульсов шага. Если установлено значение false, импульсы не генерируются.
- *position-cmd* - (Float, In) Целевая позиция пошагового движения в единицах, определяемых пользователем.
- *position-fb* - (Float, Out) Положение обратной связи в определяемых пользователем единицах измерения положения (counts/position_scale).
- *velocity-cmd* - (Float, In) Целевая скорость пошагового движения, в определяемых пользователем единицах положения в секунду. Этот контакт используется только тогда, когда степ-генератор находится в режиме управления скоростью (control-type=1).
- *velocity-fb* - (Float, Out) Скорость обратной связи в определяемых пользователем единицах положения в секунду.

#### 6.7.13.2 Параметры

- *dirhold* - (u32, RW) Минимальная продолжительность стабильного сигнала Direction после окончания импульса шага, в наносекундах.
- *dirsetup* - (u32, RW) Минимальная длительность стабильного сигнала Direction перед началом импульса шага, в наносекундах.
- *maxaccel* - (Float, RW) Максимальное ускорение, в единицах положения в секунду в секунду. Если установлено значение 0, драйвер не будет ограничивать его ускорение.
- *maxvel* - (Float, RW) Максимальная скорость, в единицах положения в секунду. Если установлено значение 0, драйвер будет выбирать максимальную скорость на основе значений Steplen и Stepspace (в то время, когда maxvel был установлен в 0).
- *position-scale* - (Float, RW) Преобразует число в единицы положения. position = counts / position_scale
- *step_type* - (u32, RW) Формат вывода, аналогичный step_type modparam для программного компонента stepgen(9). 0 = Step/Dir, 1 = Up/Down, 2 = Quadrature. В квадратурном режиме (step_type=2) stepgen выводит один полный цикл Грея (00 -> 01 -> 11 -> 10 -> 00) для каждого принимаемого *импульса шага*.
- *steplen* - (u32, RW) Длительность импульса шага в наносекундах.
- *stepspace* - (u32, RW) Минимальный интервал между импульсами шага в наносекундах.

#### 6.7.13.3 Выходные параметры

Выводы Step и Direction каждого StepGen имеют два дополнительных параметра. Чтобы определить, какой контакт ввода-вывода принадлежит какому выводу импульсов шага и направления, запустите команду «dmesg», как описано выше.

- *invert_output* — (Bit, RW) Этот параметр имеет эффект только в том случае, если параметр *is_output* имеет значение true. Если этот параметр имеет значение true, выходное значение GPIO будет обратным значению на контакте HAL *out*.

- *is_opendrain* - (Bit, RW) Если этот параметр имеет значение false, GPIO ведет себя как обычный выходной контакт: контакт ввода-вывода на разъеме переключается на значение, указанное контактом HAL *out* (возможно, инвертированое). Если этот параметр имеет значение true, GPIO ведет себя как контакт с открытым стоком. Запись 0 на контакт HAL *out* переводит контакт ввода-вывода в низкий уровень, запись 1 на контакт HAL *out* переводит контакт ввода-вывода в состояние с высоким импедансом. В этом состоянии с высоким импедансом контакт ввода-вывода плавает (слабо подтянут к высокому уровню), и другие устройства могут управлять значением; результирующее значение на контакте ввода-вывода доступно на выводах *in* и *in_not*. Этот параметр имеют только полные GPIO контакты и контакты ввода-вывода, используемые в качестве выходов активными экземплярами модуля.

## 6.7.14 PWMGen

PWMgens имеют имена типа *hm2_<BoardType>.<BoardNum>.pwmgen.<Instance>*. *Instance* — это двузначное число, соответствующее номеру экземпляра pwmgen HostMot2. Существуют экземпляры *num_pwmgens*, начиная с 00.

В HM2 каждый генератор импульсов использует три контакта ввода/вывода: Not-Enable, Out0 и Out1. Чтобы инвертировать выходной контакт PWMGen, вы инвертируете соответствующий контакт GPIO, который используется PWMGen. Чтобы найти контакт GPIO, используемый для контакта PWMGen, запустите *dmesg*, как показано выше.

Функция контактов ввода/вывода Out0 и Out1 зависит от параметра типа выхода (см. ниже).

Представление hm2 pwmgen аналогично программному компоненту pwmgen. Каждый экземпляр pwmgen имеет следующие контакты и параметры:

### 6.7.14.1 Контакты

- *enable* - (Bit, In) Если true, pwmgen установит на своем контакте Not-Enable значение false и выдаст свои импульсы. Если *enable* имеет значение false, pwmgen установит для своего контакта Not-Enable значение true и не будет выводить никаких сигналов.
- *value* - (Float, In) Текущее значение команды pwmgen в произвольных единицах измерения.

### 6.7.14.2 Параметры

- *output-type* - (s32, RW) Эмулирует аргумент output_type во время загрузки для программного компонента pwmgen. Этот параметр можно изменить во время выполнения, но в большинстве случаев вам, вероятно, захочется установить его при запуске, а затем оставить в покое. Допустимые значения: 1 (PWM на Out0 и Direction на Out1), 2 (Up на Out0 и Down на Out1), 3 (режим PDM, PDM на Out0 и Dir на Out1) и 4 (Direction на Out0 и PWM на Out1, *для заблокированной противофазы*).
- *scale* — (Float, RW) Коэффициент масштабирования для преобразования *значения* из произвольных единиц в рабочий цикл: dc = value / scale. Рабочий цикл имеет эффективный диапазон от -1,0 до +1,0 включительно, все, что выходит за пределы этого диапазона, обрезается.
- *pdm_Frequency* - (u32, RW) Определяет частоту PDM (в Гц) всех экземпляров pwmgen, работающих в режиме PDM (режим 3). Это «частота импульсного интервала»; частота, на которой генератор PDM на плате Anything I/O выбирает, выдавать ли импульс или пробел. Каждый импульс (и пауза) в последовательности импульсов PDM имеет длительность 1/pdm_frequency секунд. Например, установка pdm_frequency на $2*10^6$ Гц (2 МГц) и рабочего цикла на 50% приводит к прямоугольному сигналу частотой 1 МГц, идентичному сигналу ШИМ с частотой 1 МГц и рабочим циклом 50%. Эффективный диапазон этого параметра составляет примерно от 1525 Гц до чуть менее 100 МГц. Обратите внимание, что максимальная частота определяется

частотой ClockHigh платы Anything I/O; 5I20 и 7I43 обе имеют тактовую частоту 100 МГц, что обеспечивает максимальную частоту PDM 100 МГц. Другие платы могут иметь другие тактовые частоты, что приводит к разным максимальным частотам PDM. Если пользователь попытается установить слишком высокую частоту, она будет ограничена максимальной частотой, поддерживаемой платой.

- *pwm_frequency* - (u32, RW) Определяет частоту ШИМ в Гц всех экземпляров pwmgen, работающих в режимах ШИМ (режимы 1 и 2). Это частота с переменной скважностью импульсов. Ее эффективный диапазон составляет от 1 Гц до 193 кГц. Обратите внимание, что максимальная частота определяется частотой ClockHigh платы Anything I/O; 5i20 и 7i43 обе имеют тактовую частоту 100 МГц, что обеспечивает максимальную частоту ШИМ 193 кГц. Другие платы могут иметь другие тактовые частоты, что приводит к разным максимальным частотам ШИМ. Если пользователь попытается установить слишком высокую частоту, она будет ограничена максимальной частотой, поддерживаемой платой. Частоты около 5 Гц не очень точны, но выше 5 Гц они довольно близки.

#### 6.7.14.3 Выходные параметры

Выходные контакты каждого PWMGen имеют два дополнительных параметра. Чтобы определить, какой контакт ввода-вывода принадлежит какому выходу, запустите *dmesg*, как описано выше.

- `invert_output` — (Bit, RW) Этот параметр имеет эффект только в том случае, если параметр `is_output` имеет значение true. Если этот параметр имеет значение true, выходное значение GPIO будет обратным значению на контакте HAL `out`.
- `is_opendrain` - (Bit, RW) Если этот параметр имеет значение false, GPIO ведет себя как обычный выходной контакт: контакт ввода-вывода на разъеме переключается на значение, указанное выводом HAL `out` (возможно, инвертированное). Если этот параметр имеет значение true, GPIO ведет себя как контакт с открытым стоком. Запись 0 на вывод HAL «out» переводит контакт ввода-вывода в низкий уровень, запись 1 на вывод HAL «out» переводит контакт ввода-вывода в состояние с высоким импедансом. В этом состоянии с высоким импедансом контакт ввода-вывода является плавающим (слабо подтянут к высокому уровню), и другие устройства могут управлять значением; результирующее значение на контакте ввода-вывода доступно на контактах `in` и `in_not`. Этот параметр имеют только полные GPIO контакты и контакты ввода-вывода, используемые в качестве выходов активными экземплярами модуля.

### 6.7.15 Энкодер

Энкодеры имеют имена типа `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.`. *Instance* — это двузначное число, соответствующее номеру экземпляра энкодера HostMot2. Существуют экземпляры *num_encoders*, начиная с 00.

Каждый энкодер использует три или четыре контакта ввода-вывода, в зависимости от того, как была скомпилирована прошивка. Трехконтактные энкодеры используют A, B и Index (иногда также известный как Z). Четырехконтактные энкодеры используют A, B, Index и Index-mask.

Представление энкодера hm2 аналогично описанному в Canonical Device Interface (в документе HAL Общий справочник) и компоненту программного кодера. Каждый экземпляр энкодера имеет следующие контакты и параметры:

#### 6.7.15.1 Контакты

- `count` - (s32, Out) Количество отсчетов энкодера с момента предыдущего сброса.
- `index-enable` - (Bit, I/O). Когда этот контакт в True, счетчик (и, следовательно, также позиция) сбрасываются в ноль при следующем индексном импульсе (Phase-Z). В то же время index-enable сбрасывается в ноль, указывая на то, что импульс был.

- `position` - (Float, Out) Положение энкодера в единицах положения (count / scale).
- `rawcounts` — (s32, Out) Общее количество отсчетов энкодера с момента запуска, без поправки на индекс или сброс.
- `reset` - (Bit, In) Когда этот вывод имеет значение TRUE, контакты счета и положения устанавливаются в 0. На значение контакта скорости это не влияет. Драйвер не сбрасывает этот вывод на FALSE после сброса счетчика на 0, это задача пользователя.
- `velocity` - (Float, Out) Расчетная скорость энкодера в единицах положения в секунду.

#### 6.7.15.2 Параметры

- `counter-mode` - (Bit, RW) Установите значение False (по умолчанию) для Quadrature. Установите значение True для Up/Down или для одного входа на Phase A. Может использоваться для преобразователя частоты в скорость с одним входом на Phase A, если установлен в значение true.
- `filter` - (Bit, RW) Если установлено значение True (по умолчанию), квадратурному счетчику требуется 15 тактов для регистрации изменения на любой из трех входных линий (любой импульс короче этого значения отвергается как шум). Если установлено значение False, квадратурному счетчику потребуется всего 3 такта для регистрации изменения. Тактовая частота выборки энкодера работает на частоте 33 МГц на картах ввода-вывода PCI Anything I/O и 50 МГц на 7I43.
- `index-invert` - (Bit, RW) Если установлено значение True, нарастающий фронт входного контакта индекса запускает событие Index (если index-enable равно true). Если установлено значение False, запускает задний фронт.
- `index-mask` - (Bit, RW) Если установлено значение True, входной контакт Index имеет эффект только в том случае, если входной контакт Index-Mask имеет значение True (или False, в зависимости от контакта `index-mask-invert` ниже) .
- `index-mask-invert` - (Bit, RW) Если установлено значение True, Index-Mask должен быть False, чтобы индекс имел эффект. Если установлено значение False, контакт *Index-Mask* должен быть True.
- `scale` - (Float, RW) Преобразует из «счетных» единиц в единицы «позиции». Квадратурный энкодер обычно имеет 4 отсчета на импульс, поэтому энкодер со скоростью 100 PPR будет иметь 400 отсчетов на оборот. В `.counter-mode` энкодер со скоростью 100 PPR будет иметь 100 отсчетов на оборот, поскольку он использует только нарастающий фронт сигнала A и направление B.
- `vel-timeout` - (Float, RW) Когда энкодер движется медленнее, чем один импульс каждый раз, когда драйвер считывает счетчик из FPGA (в функции hm2_read()), скорость оценить труднее. Драйвер может ждать поступления следующего импульса несколько итераций, при этом сообщая о верхней границе скорости энкодера, которую можно точно угадать. Этот параметр определяет, как долго ждать следующего импульса, прежде чем сообщить об остановке энкодера. Этот параметр указывается в секундах.

### 6.7.16 5I25 Конфигурация

#### 6.7.16.1 Прошивка

Прошивка 5I25 поставляется предустановленной для дочерней карты, с которой она приобретена. Таким образом, `firmware=xxx.BIT` не является частью строки конфигурации hm2_pci при использовани 5I25.

#### 6.7.16.2 Конфигурация

Примеры конфигураций плат 5I25/7I76 и 5I25/7I77 включены в раздел Configuration Selector.

Если вы хотите создать собственную конфигурацию, в следующих примерах показано, как загрузить драйверы в файл HAL.

**5I25 + 7I76 карта**

```
#b''зb''b''ab''b''гb''b''pb''b''yb''b''зb''b''иb''b''тb''b''ьb'' b''yb''b''нb''b''иb''b' ↩
    'вb''b''eb''b''pb''b''cb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b''йb'' b''дb''b''pb''b' ↩
    'ab''b''йb''b''вb''b''eb''b''pb''
loadrt hostmot2

# b''зb''b''ab''b''гb''b''pb''b''yb''b''зb''b''иb''b''тb''b''eb'' b''иb'' b''нb''b''ab''b' ↩
    'cb''b''тb''b''pb''b''ob''b''йb''b''тb''b''eb'' b''дb''b''pb''b''ab''b''йb''b''вb''b' ↩
    'eb''b''pb'' PCI
loadrt hm2_pci config="num_encoders=1 num_stepgens=5 sserial_port_0=0XXX"
```

**5I25 + 7I77 карта**

```
#b''зb''b''ab''b''гb''b''pb''b''yb''b''зb''b''иb''b''тb''b''ьb'' b''yb''b''нb''b''иb''b' ↩
    'вb''b''eb''b''pb''b''cb''b''ab''b''лb''b''ьb''b''нb''b''ыb''b''йb'' b''дb''b''pb''b' ↩
    'ab''b''йb''b''вb''b''eb''b''pb''
loadrt hostmot2

# b''зb''b''ab''b''гb''b''pb''b''yb''b''зb''b''иb''b''тb''b''eb'' b''иb'' b''нb''b''ab''b' ↩
    'cb''b''тb''b''pb''b''ob''b''йb''b''тb''b''eb'' b''дb''b''pb''b''ab''b''йb''b''вb''b' ↩
    'eb''b''pb'' PCI
loadrt hm2_pci config="num_encoders=6 num_pwmgens=6 sserial_port_0=0XXX"
```

#### 6.7.16.3 SSERIAL Конфигурация

Строка конфигурации `sserial_port_0=0XXX` устанавливает некоторые параметры для дочерней smart serial карты. Эти параметры индивидуальны для каждой дочерней карты. Дополнительную информацию о точном использовании см. в руководстве Mesa (обычно в разделе SOFTWARE PROCESS DATA MODES) или на странице руководства SSERIAL(9).

#### 6.7.16.4 7I77 Пределы

Minlimit и maxlimit являются границами значения контакта (в данном случае значение аналогового выхода), а Fullscalemax — масштабный коэффициент.

По умолчанию они установлены на аналоговый вход или аналоговый диапазон (скорее всего, в вольтах).

Так, например, для аналоговых выходов 7I77 +-10 В значения по умолчанию следующие:

minlimit: -10
maxlimit: +10
maxfullscale: 10

Если вы хотите масштабировать аналоговый выход канала до IPS для сервопривода в режиме скорости (скажем, максимум 24 IPS), вы можете установить такие ограничения:

minlimit: -24
maxlimit: +24
maxfullscale: 24

Если вы хотите масштабировать аналоговый сигнал канала до числа оборотов в минуту для шпинделя от 0 до 6000 об/мин с управлением 0–10 В, вы можете установить следующие ограничения:

minlimit: 0
maxlimit: 6000
maxfullscale: 6000
(это предотвратит установку нежелательных отрицательных выходных напряжений.)

### 6.7.17 Примеры конфигураций

Несколько примеров конфигураций оборудования Mesa включены в состав LinuxCNC. Конфигурации расположены в разделах hm2-servo и hm2-stepper раздела Configuration Selector. Как правило, вам потребуется плата, установленная для конфигурации, которую вы выбираете для загрузки. Примеры — хорошая отправная точка, они сэкономят вам время. Просто выберите подходящий пример из Configuration Selector LinuxCNC и сохраните копию на своем компьютере, чтобы ее можно было редактировать. Чтобы увидеть точные контакты и параметры, которые предоставила вам ваша конфигурация, откройте окно ”Show HAL Configuration” из меню Machine или выполните команду *dmesg*, как описано выше.

## 6.8 MB2HAL

### 6.8.1 Введение

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. Now there is a third option that consists of a ”generic” driver configured by text file, this is called MB2HAL.

Why MB2HAL? Consider using MB2HAL if:

- You have to write a new driver and you don’t know anything about programming.
- You need to use Classic Ladder ”only” to manage the Modbus connections.
- 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 Использование

a. Create a config file from the example below

   1. Set component name (optional)
      Set `HAL_MODULE_NAME=mymodule` (default `HAL_MODULE_NAME=mb2hal`)

2. Load the modbus HAL non-realtime component

b. Default component name: `loadusr -W mb2hal config=config_file.ini`

c. Custom component name: `loadusr -Wn mymodule mb2hal config=config_file.ini`

## 6.8.3 Варианты

### 6.8.3.1 Init Section

`[MB2HAL_INIT]`

| Value | Type | Required | Description |
|---|---|---|---|
| `INIT_DEBUG` | Integer | No | Debug level of init and INI file parsing.<br>0 = silent<br>1 = error messages (default)<br>2 = OK confirmation messages<br>3 = debugging messages<br>4 = maximum debugging messages (only in transactions) |
| `VERSION` | String | No | Version number in the format N.N[NN]. Defaults to 1.0. |
| `HAL_MODULE_NAME` | String | No | HAL module (component) name. Defaults to ”mb2hal”. |
| `SLOWDOWN` | Float | No | 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. |
| `TOTAL_TRANSACTIONS` | Integer | Yes | The number of total Modbus transactions. There is no maximum. |

### 6.8.3.2 Transaction Sections

One transaction section is required per transaction, starting at `[TRANSACTION_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.

| Value | Type | Required | Description |
|---|---|---|---|
| `LINK_TYPE` | String | Yes | 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 address | If `LINK_TYPE=tcp` | The Modbus slave device IP address. Ignored if `LINK_TYPE=serial`. |
| `TCP_PORT` | Integer | No | The Modbus slave device TCP port. Defaults to 502. Ignored if `LINK_TYPE=serial`. |
| `SERIAL_PORT` | 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` | 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`. |

| Value | Type | Required | Description |
|---|---|---|---|
| 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 | Yes | Modbus slave number. |
| FIRST_ELEMENT | Integer | Yes | The first element address. |
| NELEMENTS | Integer | 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. |
| PIN_NAMES | List | 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. Example: PIN_NAMES=cycle_start,stop,feed_hold |
| MB_TX_CODE | String | Yes | Код функции транзакции Modbus (смотрите ссылку:specifications):<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 | No | 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 | No | 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 | String | No | 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 | Float | No | 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 | String | No | Debug level for this transaction only. See INIT_DEBUG parameter above. |

#### 6.8.3.3 Error codes

While debugging transactions, note the returned "ret[]" value correspond to:

Modbus protocol exceptions:

- 0x01 - ILLEGAL_FUNCTION - the FUNCTION code received in the query is not allowed or invalid.
- 0x02 - ILLEGAL_DATA_ADDRESS - the DATA ADDRESS received in the query is not an allowable address for the slave or is invalid.
- 0x03 - ILLEGAL_DATA_VALUE - a VALUE contained in the data query field is not an allowable value or is invalid.

- 0x04 - SLAVE_DEVICE_FAILURE - SLAVE (or MASTER) device unrecoverable FAILURE while attempting to perform the requested action.
- 0x04 - SERVER_FAILURE - (see above).
- 0x05 - ACKNOWLEDGE - 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.
- 0x06 - SLAVE_DEVICE_BUSY - The slave (or server) is BUSY. Retransmit the request later.
- 0x06 - SERVER_BUSY - (see above).
- 0x07 - NEGATIVE_ACKNOWLEDGE - Unsuccessful programming request using function code 13 or 14.
- 0x08 - MEMORY_PARITY_ERROR - SLAVE parity error in MEMORY.
- 0x0A (-10) - GATEWAY_PROBLEM_PATH - Gateway path(s) not available.
- 0x0B (-11) - GATEWAY_PROBLEM_TARGET - The target device failed to respond (generated by master, not slave).

Program or connection:

- 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 Example config file

Click here to download.

```
#This .INI file is also the HELP, MANUAL and HOW-TO file for mb2hal.

#Load the Modbus HAL userspace module as the examples below,
#change to match your own HAL_MODULE_NAME and INI file name
#Using HAL_MODULE_NAME=mb2hal or nothing (default): loadusr -W mb2hal config=config_file. ↩
    ini
#Using HAL_MODULE_NAME=mymodule: loadusr -Wn mymodule mb2hal config=config_file.ini

# ++++++++++++++++++++++++
# Common section
# ++++++++++++++++++++++++
[MB2HAL_INIT]

#OPTIONAL: Debug level of init and INI file parsing.
# 0 = silent.
# 1 = error messages (default).
# 2 = OK confirmation messages.
# 3 = debugging messages.
# 4 = maximum debugging messages (only in transactions).
INIT_DEBUG=3

#OPTIONAL: Set to 1.1 to enable the new functions:
```

```
# - fnct_01_read_coils
# - fnct_05_write_single_coil
# - changed pin names (see https://linuxcnc.org/docs/2.9/html/drivers/mb2hal.html#_pins).
VERSION=1.1

#OPTIONAL: HAL module (component) name. Defaults to ”mb2hal”.
HAL_MODULE_NAME=mb2hal

#OPTIONAL: Insert a delay of ”FLOAT seconds” between transactions in order
#to not to have a lot of logging and facilitate the debugging.
#Useful when using DEBUG=3 (NOT INIT_DEBUG=3)
#It affects ALL transactions.
#Use ”0.0” for normal activity.
SLOWDOWN=0.0

#REQUIRED: The number of total Modbus transactions. There is no maximum.
TOTAL_TRANSACTIONS=9

# ++++++++++++++++++++++++
# Transactions
# ++++++++++++++++++++++++
#One transaction section is required per transaction, starting at 00 and counting up  ↩
    sequentially.
#If there is a new link (not transaction), you must provide the REQUIRED parameters 1st  ↩
    time.
#Warning: Any OPTIONAL parameter not specified are copied from the previous transaction.
[TRANSACTION_00]

#REQUIRED: You must specify either a ”serial” or ”tcp” link for the first transaction.
#Later transaction will use the previous transaction link if not specified.
LINK_TYPE=tcp

#if LINK_TYPE=tcp then REQUIRED (only 1st time): The Modbus slave device ip address.
#if LINK_TYPE=serial then IGNORED
TCP_IP=192.168.2.10

#if LINK_TYPE=tcp then OPTIONAL.
#if LINK_TYPE=serial then IGNORED
#The Modbus slave device tcp port. Defaults to 502.
TCP_PORT=502

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The serial port.
SERIAL_PORT=/dev/ttyS0

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#The baud rate.
SERIAL_BAUD=115200

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data bits. One of 5,6,7,8.
SERIAL_BITS=8

#if LINK_TYPE=serial then REQUIRED (only 1st time).
#if LINK_TYPE=tcp then IGNORED
#Data parity. One of: even, odd, none.
SERIAL_PARITY=none

#if LINK_TYPE=serial then REQUIRED (only 1st time).
```

```
#if LINK_TYPE=tcp then IGNORED
#Stop bits. One of 1, 2.
SERIAL_STOP=2

#if LINK_TYPE=serial then OPTIONAL:
#if LINK_TYPE=tcp then IGNORED
#Serial port delay between for this transaction only.
#In ms. Defaults to 0.
SERIAL_DELAY_MS=10

#REQUIRED (only 1st time).
#Modbus slave number.
MB_SLAVE_ID=1

#REQUIRED: The first element address (decimal integer).
FIRST_ELEMENT=0

#REQUIRED unless PIN_NAMES is specified: The number of elements.
#It is an error to specify both NELEMENTS and PIN_NAMES
#The pin names will be sequential numbers e.g mb2hal.plcin.01
#NELEMENTS=4

#REQUIRED unless NELEMENTS is specified: A list of element names.
#these names will be used for the pin names, e.g mb2hal.plcin.cycle_start
#NOTE: there must be no white space characters in the list
PIN_NAMES=cycle_start,stop,feed_hold

#REQUIRED: Modbus transaction function code (see www.modbus.org specifications).
#    fnct_01_read_coils               (01 = 0x01) (new in 1.1)
#    fnct_02_read_discrete_inputs     (02 = 0x02)
#    fnct_03_read_holding_registers   (03 = 0x03)
#    fnct_04_read_input_registers     (04 = 0x04)
#    fnct_05_write_single_coil        (05 = 0x05) (new in 1.1)
#    fnct_06_write_single_register    (06 = 0x06)
#    fnct_15_write_multiple_coils     (15 = 0x0F)
#    fnct_16_write_multiple_registers (16 = 0x10)
#
# Created pins:
# fnct_01_read_coils:
# fnct_02_read_discrete_inputs:
#     mb2hal.m.n.bit     (output)
#     mb2hal.m.n.bit-inv (output)
# fnct_03_read_holding_registers:
# fnct_04_read_input_registers:
#     mb2hal.m.n.float   (output)
#     mb2hal.m.n.int     (output)
# fnct_05_write_single_coil:
#     mb2hal.m.n.bit     (input)
#         NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
# fnct_06_write_single_register:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     NELEMENTS needs to be 1 or PIN_NAMES must contain just one name.
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
# fnct_15_write_multiple_coils:
#     mb2hal.m.n.bit     (input)
# fnct_16_write_multiple_registers:
#     mb2hal.m.n.float   (input)
#     mb2hal.m.n.int     (input)
#     Both pin values are added and limited to 65535 (UINT16_MAX). Normally use one and let ↩
    the other open (read as 0).
```

```
#
# m = HAL_TX_NAME or transaction number if not set, n = element number (NELEMENTS) or name  ↩
    from PIN_NAMES
# Example: mb2hal.00.01.<type> (transaction=00, second register=01 (00 is the first one))
#          mb2hal.TxName.01.<type> (HAL_TX_NAME=TxName, second register=01 (00 is the first ↩
     one))
MB_TX_CODE=fnct_03_read_holding_registers

#OPTIONAL: Response timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait for 1st byte before raise an error.
MB_RESPONSE_TIMEOUT_MS=500

#OPTIONAL: Byte timeout for this transaction. In INTEGER ms. Defaults to 500 ms.
#This is how much to wait from byte to byte before raise an error.
MB_BYTE_TIMEOUT_MS=500

#OPTIONAL: Instead of giving the transaction number, use a name.
#Example: mb2hal.00.01 could become mb2hal.plcin.01
#The name must not exceed 28 characters.
#NOTE: when using names be careful that you dont end up with two transactions
#using the same name.
HAL_TX_NAME=remoteIOcfg

#OPTIONAL: Maximum update rate in HZ. Defaults to 0.0 (0.0 = as soon as available =  ↩
    infinite).
#NOTE: This is a maximum rate and the actual rate may be lower.
#If you want to calculate it in ms use (1000 / required_ms).
#Example: 100 ms = MAX_UPDATE_RATE=10.0, because 1000.0 ms / 100.0 ms = 10.0 Hz
MAX_UPDATE_RATE=0.0

#OPTIONAL: Debug level for this transaction only.
#See INIT_DEBUG parameter above.
DEBUG=2

#While DEBUGGING transactions note the returned ”ret[]” value correspond to:
#/* Modbus protocol exceptions */
#ILLEGAL_FUNCTION        -0x01 the FUNCTION code received in the query is not allowed or  ↩
    invalid.
#ILLEGAL_DATA_ADDRESS    -0x02 the DATA ADDRESS received in the query is not an allowable  ↩
    address for the slave or is invalid.
#ILLEGAL_DATA_VALUE      -0x03 a VALUE contained in the data query field is not an  ↩
    allowable value or is invalid.
#SLAVE_DEVICE_FAILURE    -0x04 SLAVE (or MASTER) device unrecoverable FAILURE while  ↩
    attempting to perform the requested action.
#SERVER_FAILURE          -0x04 (see above).
#ACKNOWLEDGE             -0x05 This response is returned to PREVENT A TIMEOUT in the master ↩
    .
#                             A long duration of time is required to process the request  ↩
    in the slave.
#SLAVE_DEVICE_BUSY       -0x06 The slave (or server) is BUSY. Retrasmit the request later.
#SERVER_BUSY             -0x06 (see above).
#NEGATIVE_ACKNOWLEDGE    -0x07 Unsuccessful programming request using function code 13 or  ↩
    14.
#MEMORY_PARITY_ERROR     -0x08 SLAVE parity error in MEMORY.
#GATEWAY_PROBLEM_PATH    -0x0A (-10) Gateway path(s) not available.
#GATEWAY_PROBLEM_TARGET  -0x0B (-11) The target device failed to respond (generated by  ↩
    master, not slave).
#/* Program or connection */
#COMM_TIME_OUT           -0x0C (-12)
#PORT_SOCKET_FAILURE     -0x0D (-13)
#SELECT_FAILURE          -0x0E (-14)
#TOO_MANY_DATAS          -0x0F (-15)
```

```
#INVALID_CRC              -0x10 (-16)
#INVALID_EXCEPTION_CODE   -0x11 (-17)

[TRANSACTION_01]
MB_TX_CODE=fnct_01_read_coils
FIRST_ELEMENT=1024
NELEMENTS=24
HAL_TX_NAME=remoteIOin
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_02]
MB_TX_CODE=fnct_02_read_discrete_inputs
FIRST_ELEMENT=1280
NELEMENTS=8
HAL_TX_NAME=readStatus
MAX_UPDATE_RATE=0.0

[TRANSACTION_03]
MB_TX_CODE=fnct_05_write_single_coil
FIRST_ELEMENT=100
NELEMENTS=1
HAL_TX_NAME=setEnableout
MAX_UPDATE_RATE=0.0

[TRANSACTION_04]
MB_TX_CODE=fnct_15_write_multiple_coils
FIRST_ELEMENT=150
NELEMENTS=10
HAL_TX_NAME=remoteIOout
MAX_UPDATE_RATE=0.0

[TRANSACTION_05]
LINK_TYPE=serial
SERIAL_PORT=/dev/ttyS0
SERIAL_BAUD=115200
SERIAL_BITS=8
SERIAL_PARITY=none
SERIAL_STOP=2
SERIAL_DELAY_MS=50
MB_SLAVE_ID=1
MB_TX_CODE=fnct_03_read_holding_registers
FIRST_ELEMENT=1
NELEMENTS=2
HAL_TX_NAME=XDrive01
MAX_UPDATE_RATE=0.0
DEBUG=1

[TRANSACTION_06]
MB_TX_CODE=fnct_04_read_input_registers
FIRST_ELEMENT=12
NELEMENTS=3
HAL_TX_NAME=XDrive02
MAX_UPDATE_RATE=10.0
DEBUG=1

[TRANSACTION_07]
MB_TX_CODE=fnct_06_write_single_register
FIRST_ELEMENT=20
NELEMENTS=1
HAL_TX_NAME=XDrive03
MAX_UPDATE_RATE=0.0
```

```
DEBUG=1

[TRANSACTION_08]
MB_TX_CODE=fnct_16_write_multiple_registers
FIRST_ELEMENT=55
NELEMENTS=8
HAL_TX_NAME=XDrive04
MAX_UPDATE_RATE=10.0
DEBUG=1
```

### 6.8.5 Контакты

---

**Note**
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`
Example:

- `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` needs to be 1 or `PIN_NAMES` must contain just one name.

#### 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 Driver

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.

---

**Note**
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.

---

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01
```

The above command says:
loadusr, wait for coolant pins to be ready, component mitsub_vfd, with 2 slaves named spindle (slave #2) and coolant (slave #1)

### 6.9.1 Command Line Options

The command line options are:

- *-b or --baud <rate>* : set the baud rate - all networked VFDs must be the same

- *-p or --port <device path>* : sets the port to use such as /dev/ttyUSB0
- *<name>=<slave#>* : sets the HAL component/pin name and slave number.

Debugging can be toggled by setting the debug pin true.

**Note**
Turning on debugging will result in a flood of text in the terminal.

### 6.9.2 Контакты

Where <n> is `mitsub_vfd` or the name given during loading.

- *<n>.fwd* (bit, in) True sets motion forward, False sets reverse.
- *<n>.run* (bit, in) True sets the VFD in motion based on the .fwd pin.
- *<n>.debug* (bit, in) Prints debug info to the terminal.
- *<n>.alarm* (bit, out) signals an alarm state of VFD.
- *<n>.up-to-speed* (bit, out) when drive is at commanded speed (speed-tolerance is set on vfd)
- *<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) commanded speed to the VFD (scaled to hertz by default).
- *<n>.motor-fb* (float, out) feedback speed from the VFD (scaled to hertz by default).
- *<n>.motor-amps* (float, out) Current amperage output of motor.
- *<n>.motor-power* (float, out) Current power output of motor.
- *<n>.scale-cmd* (float, in) Scales the motor-cmd pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-fb* (float, in) Scales the motor-fb pin to arbitrary units. default 1 = Hertz.
- *<n>.scale-amps* (float, in) Scales the motor-amps pin to arbitrary units. default 1 = amps.
- *<n>.scale-power* (float, in) Scales the motor-power pin to arbitrary units. default 1 = .
- *<n>.estop* (bit, in) puts the VFD into emergency-stopped 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 example

```
#
# example usage of the Mitsubishi VFD driver
#
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01

# *************** Spindle VFD setup slave 2 *********************
net spindle-vel-cmd               spindle.motor-cmd
net spindle-cw                    spindle.fwd
net spindle-on                    spindle.run
net spindle-at-speed              spindle.up-to-speed
```

```
net estop-out                     spindle.estop
#       cmd scaled to RPM
setp spindle.scale-cmd .135
#       feedback is in rpm
setp spindle.scale-fb 7.411
#       allows us to see status
setp spindle.monitor 1

net spindle-speed-indicator spindle.motor-fb        gladevcp.spindle-speed

# *************** Coolant 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 and feedback scaled to hertz
setp coolant.scale-cmd 1
setp coolant.scale-fb 1
#       command full speed
setp coolant.motor-cmd 60
#       allows us to see status
setp coolant.monitor 1
```

## 6.9.4 Configuring the Mitsubishi VFD for serial usage

### 6.9.4.1 Connecting the Serial Port

The Mitsubishi VFDs have an RJ-45 jack for serial communication.
Since they use RS485 protocol, they can be networked together point to point.
This driver was tested using the Opto22 AC7A to convert from RS232 to RS485.

### 6.9.4.2 Modbus setup

Referenced manuals:

*communication option reference manual* and *A500 technical manual* for 500 series.
*Fr-A700 F700 E700 D700 technical manual* for the 700 series

The VFD must have PR settings adjusted manually for serial communication.
One must power cycle the VFD for some of these to register eg PR 79

- *PR 77* set to 1 *-to unlock other PR modification.*
- *PR 79* set to 1 or 0 *-for communication thru serial.*
- *PR 117* set to 0-31 *-slave number, driver must reference same number.*
- *PR 118* tested with 96 *-baud rate (can be set to 48,96,192) if driver is also set.*
- *PR 119* set to 0 *-stop bit/data length (8 bits, two stop)*
- *PR 120* set to 0 *-no parity*
- *PR 121* set to 1-10 *-if 10 (maximum) COM errors then VFD faults.*
- *PR 122* tested with 9999 *-if communication is lost VFD will not error.*
- *PR 123* set to 9999 *-no wait time is added to the serial data frame.*
- *PR 124* set to 0 *-no carriage return at end of line.*

# 6.10 Motenc Driver

Vital Systems Motenc-100 and Motenc-LITE

The Vital Systems Motenc-100 and Motenc-LITE are 8- and 4-channel servo control boards. The Motenc-100 provides 8 quadrature encoder counters, 8 analog inputs, 8 analog outputs, 64 (68?) digital inputs, and 32 digital outputs. The Motenc-LITE has only 4 encoder counters, 32 digital inputs and 16 digital outputs, but it still has 8 analog inputs and 8 analog outputs. The driver automatically identifies the installed board and exports the appropriate HAL objects.

Installing:

```
loadrt hal_motenc
```

Во время загрузки (или попытки загрузки) драйвер печатает некоторые полезные отладочные сообщения в журнал ядра, которые можно просмотреть с помощью dmesg.

Up to 4 boards may be used in one system.

## 6.10.1 Контакты

In the following pins, parameters, and functions, <board> is the board ID. According to the naming conventions the first board should always have an ID of zero. However this driver sets the ID based on a pair of jumpers on the board, so it may be non-zero even if there is only one board.

- *(s32) motenc.<board>.enc-<channel>-count* - Encoder position, in counts.
- *(float) motenc.<board>.enc-<channel>-position* - Encoder position, in user units.
- *(bit) motenc.<board>.enc-<channel>-index* - Current status of index pulse input.
- *(bit) motenc.<board>.enc-<channel>-idx-latch* - Driver sets this pin true when it latches an index pulse (enabled by latch-index). Cleared by clearing latch-index.
- *(bit) motenc.<board>.enc-<channel>-latch-index* - If this pin is true, the driver will reset the counter on the next index pulse.
- *(bit) motenc.<board>.enc-<channel>-reset-count* - If this pin is true, the counter will immediately be reset to zero, and the pin will be cleared.
- *(float) motenc.<board>.dac-<channel>-value* - Analog output value for DAC (in user units, see -gain and -offset)
- *(float) motenc.<board>.adc-<channel>-value* - Analog input value read by ADC (in user units, see -gain and -offset)
- *(bit) motenc.<board>.in-<channel>* - State of digital input pin, see canonical digital input.
- *(bit) motenc.<board>.in-<channel>-not* - Inverted state of digital input pin, see canonical digital input.
- *(bit) motenc.<board>.out-<channel>* - Value to be written to digital output, seen canonical digital output.
- *(bit) motenc.<board>.estop-in* - Dedicated estop input, more details needed.
- *(bit) motenc.<board>.estop-in-not* - Inverted state of dedicated estop input.
- *(bit) motenc.<board>.watchdog-reset* - Bidirectional, - Set TRUE to reset watchdog once, is automatically cleared.

### 6.10.2 Параметры

- *(float) motenc.<board>.enc-<channel>-scale* - The number of counts / user unit (to convert from counts to units).
- *(float) motenc.<board>.dac-<channel>-offset* - Sets the DAC offset.
- *(float) motenc.<board>.dac-<channel>-gain* - Sets the DAC gain (scaling).
- *(float) motenc.<board>.adc-<channel>-offset* - Sets the ADC offset.
- *(float) motenc.<board>.adc-<channel>-gain* - Sets the ADC gain (scaling).
- *(bit) motenc.<board>.out-<channel>-invert* - Inverts a digital output, see canonical digital output.
- *(u32) motenc.<board>.watchdog-control* - Configures the watchdog.
  The value may be a bitwise OR of the following values:

| Bit # | Value | Meaning |
|---|---|---|
| 0 | 1 | Timeout is 16ms if set, 8ms if unset |
| 1 | 2 | |
| 2 | 4 | Watchdog is enabled |
| 3 | 8 | |
| 4 | 16 | Watchdog is automatically reset by DAC writes (the HAL dac-write function) |

Typically, the useful values are 0 (watchdog disabled) or 20 (8ms watchdog enabled, cleared by dac-write).

- *(u32) motenc.<board>.led-view* - Maps some of the I/O to onboard LEDs.

### 6.10.3 Функции

- *(funct) motenc.<board>.encoder-read* - Reads all encoder counters.
- *(funct) motenc.<board>.adc-read* - Reads the analog-to-digital converters.
- *(funct) motenc.<board>.digital-in-read* - Reads digital inputs.
- *(funct) motenc.<board>.dac-write* - Writes the voltages to the DACs.
- *(funct) motenc.<board>.digital-out-write* - Writes digital outputs.
- *(funct) motenc.<board>.misc-update* - Updates misc stuff.

## 6.11 Opto22 Driver

**PCI AC5 ADAPTER CARD / HAL DRIVER**

### 6.11.1 The Adapter Card

This is a card made by Opto22 for adapting the PCI port to solid state relay racks such as their standard or G4 series. It has 2 ports that can control up to 24 points each and has 4 on board LEDs. The ports use 50 pin connectors the same as Mesa boards. Any relay racks/breakout boards that work with Mesa Cards should work with this card with the understanding any encoder counters, PWM, etc., would have to be done in software. The AC5 does not have any *smart* logic on board, it is just an adapter.

See the manufacturer's website for more info:

https://www.opto22.com/site/pr_details.aspx?cid=4&item=PCI-AC5

I would like to thank Opto22 for releasing info in their manual, easing the writing of this driver!

### 6.11.2 The Driver

This driver is for the PCI AC5 card and will not work with the ISA AC5 card. The HAL driver is a realtime module. It will support 4 cards as is (more cards are possible with a change in the source code). Load the basic driver like so:

```
loadrt opto_ac5
```

This will load the driver which will search for max 4 boards. It will set I/O of each board's 2 ports to a default setting. The default configuration is for 12 inputs then 12 outputs. The pin name numbers correspond to the position on the relay rack. For example the pin names for the default I/O setting of port 0 would be:

- *opto_ac5.0.port0.in-00* - They would be numbered from 00 to 11
- *opto_ac5.0.port0.out-12* - They would be numbered 12 to 23 port 1 would be the same.

### 6.11.3 Контакты

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER] OUT bit* -
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER]-not OUT bit* - Connect a HAL bit signal to this pin to read an I/O point from the card. The PINNUMBER represents the position in the relay rack. Eg. PINNUMBER 0 is position 0 in a Opto22 relay rack and would be pin 47 on the 50 pin header connector. The -not pin is inverted so that LOW gives TRUE and HIGH gives FALSE.
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER] IN bit* - Connect a HAL bit signal to this pin to write to an I/O point of the card. The PINNUMBER represents the position in the relay rack.Eg. PINNUMBER 23 is position 23 in a Opto22 relay rack and would be pin 1 on the 50 pin header connector.
- *opto_ac5.[BOARDNUMBER].led[NUMBER] OUT bit* - Turns one of the 4 onboard LEDs on/off. LEDs are numbered 0 to 3.

BOARDNUMBER can be 0-3 PORTNUMBER can be 0 or 1. Port 0 is closest to the card bracket.

### 6.11.4 Параметры

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER]-invert W bit* - When TRUE, invert the meaning of the corresponding -out pin so that TRUE gives LOW and FALSE gives HIGH.

### 6.11.5 FUNCTIONS

- *opto_ac5.0.digital-read* - Add this to a thread to read all the input points.
- *opto_ac5.0.digital-write* - Add this to a thread to write all the output points and LEDs.

For example the pin names for the default I/O setting of port 0 would be:

```
opto_ac5.0.port0.in-00
```

They would be numbered from 00 to 11

```
opto_ac5.0.port0.out-12
```

They would be numbered 12 to 23 port 1 would be the same.

### 6.11.6 Configuring I/O Ports

To change the default setting load the driver something like so:

```
loadrt opto_ac5 portconfig0=0xffff portconfig1=0xff0000
```

Of course changing the numbers to match the I/O you would like. Each port can be set up different.

Here's how to figure out the number: The configuration number represents a 32 bit long code to tell the card which I/O points are output vrs input. The lower 24 bits are the I/O points of one port. The 2 highest bits are for 2 of the on board LEDs. A one in any bit position makes the I/O point an output. The two highest bits must be output for the LEDs to work. The driver will automatically set the two highest bits for you, we won't talk about them.

The easiest way to do this is to fire up the calculator under APPLICATIONS/ACCESSORIES. Set it to scientific (click view). Set it BINARY (radio button Bin). Press 1 for every output you want and/or zero for every input. Remember that HAL pin 00 corresponds to the rightmost bit. 24 numbers represent the 24 I/O points of one port. So for the default setting (12 inputs then 12 outputs) you would push 1 twelve times (that's the outputs) then 0 twelve times (that's the inputs). Notice the first I/O point is the lowest (rightmost) bit. (that bit corresponds to HAL pin 00 .looks backwards) You should have 24 digits on the screen. Now push the Hex radio button. The displayed number (fff000) is the configport number ( put a *0x* in front of it designating it as a HEX number).

Another example: To set the port for 8 outputs and 16 inputs (the same as a Mesa card). Here is the 24 bits represented in a BINARY number. Bit 1 is the rightmost number:

**16 zeros for the 16 inputs and 8 ones for the 8 outputs**

```
000000000000000011111111
```

This converts to FF on the calculator, so 0xff is the number to use for portconfig0 and/or portconfig1 when loading the driver.

### 6.11.7 Pin Numbering

HAL pin 00 corresponds to bit 1 (the rightmost) which represents position 0 on an Opto22 relay rack. HAL pin 01 corresponds to bit 2 (one spot to the left of the rightmost) which represents position 1 on an Opto22 relay rack. HAL pin 23 corresponds to bit 24 (the leftmost) which represents position 23 on an Opto22 relay rack.

HAL pin 00 connects to pin 47 on the 50 pin connector of each port. HAL pin 01 connects to pin 45 on the 50 pin connector of each port. HAL pin 23 connects to pin 1 on the 50 pin connector of each port.

Note that Opto22 and Mesa use opposite numbering systems: Opto22 position 23 = connector pin 1, and the position goes down as the connector pin number goes up. Mesa Hostmot2 position 1 = connector pin 1, and the position number goes up as the connector pin number goes up.

## 6.12 Pico Drivers

Pico Systems has a family of boards for doing analog servo, stepper, and PWM (digital) servo control. The boards connect to the PC through a parallel port working in EPP mode. Although most users connect one board to a parallel port, in theory any mix of up to 8 or 16 boards can be used on a single parport. One driver serves all types of boards. The final mix of I/O depends on the connected board(s). The driver doesn't distinguish between boards, it simply numbers I/O channels (encoders, etc) starting from 0 on the first board. The driver is named hal_ppmc.ko The analog servo interface is also called the PPMC for Parallel Port Motion Control. There is also the Universal Stepper Controller, abbreviated the USC. And the Universal PWM Controller, or UPC.

Installing:

```
loadrt hal_ppmc port_addr=<addr1>[,<addr2>[,<addr3>...]]
```

The *port_addr* parameter tells the driver what parallel port(s) to check. By default, <*addr1*> is 0x0378, and <*addr2*> and following are not used. The driver searches the entire address space of the enhanced parallel port(s) at *port_addr*, looking for any board(s) in the PPMC family. It then exports HAL pins for whatever it finds. During loading (or attempted loading) the driver prints some useful debugging messages to the kernel log, which can be viewed with *dmesg*.

Up to 3 parport buses may be used, and each bus may have up to 8 (or possibly 16 PPMC) devices on it.

### 6.12.1 Command Line Options

There are several options that can be specified on the loadrt command line. First, the USC and UPC can have an 8-bit DAC added for spindle speed control and similar functions. This can be specified with the extradac=*0xnn[,0xmm]* parameter. The part enclosed in [ ] allows you to specify this option on more than one board of the system. The first hex digit tells which EPP bus is being referred to, it corresponds to the order of the port addresses in the port_addr parameter, where <addr1> would be zero here. So, for the first EPP bus, the first USC or UPC board would be described as *0x00*, the second USC or UPC on the same bus would be *0x02*. (Note that each USC or UPC takes up two addresses, so if one is at 00, the next would have to be 02.)

Alternatively, the 8 digital output pins can be used as additional digital outputs, it works the same way as above with the syntax : extradout=0xnn'. The extradac and extradout options are mutually exclusive on each board, you can only specify one.

The UPC and PPMC encoder boards can timestamp the arrival of encoder counts to refine the derivation of axis velocity. This derived velocity can be fed to the PID hal component to produce smoother D term response. The syntax is : timestamp=*0xnn[,0xmm]*, this works the same way as above to select which board is being configured. Default is to not enable the timestamp option. If you put this option on the command line, it enables the option. The first *n* selects the EPP bus, the second one matches the address of the board having the option enabled. The driver checks the revision level of the board to make sure it has firmware supporting the feature, and produces an error message if the board does not support it.

The PPMC encoder board has an option to select the encoder digital filter frequeency. (The UPC has the same ability via DIP switches on the board.) Since the PPMC encoder board doesn't have these extra DIP switches, it needs to be selected via a command-line option. By default, the filter runs at 1 MHz, allowing encoders to be counted up to about 900 kHz (depending on noise and quadrature accuracy of the encoder.) The options are 1, 2.5, 5 and 10 MHz. These are set with a parameter of 1,2,5 and 10 (decimal) which is specified as the hex digit "A". These are specified in a manner similar to the above options, but with the frequency setting to the left of the bus/address digits. So, to set 5 MHz on the encoder board at address 3 on the first EPP bus, you would write: `enc_clock='0x503'`.

It was recently discovered that some parallel port chips would not work with the ppmc driver. Especially, the Oxford OXPCIe952 chip on the SIIG PCIe parallel port cards had this trouble. The ppmc

driver in all LinuxCNC versions starting from 2.7.8 have been corrected for this problem by default. However, this possibly could cause problems with really old EPP parallel port hardware, so there is a command line option to go back to the previous behavior. The new behavior is set by default, or by adding the parameter epp_dir=0 on the command line. To get the old behavior, add epp_dir=1 to the command line. All parallel ports I have here work with the new default behavior. As on the other parameters, it is possible to give a list, like epp_dir=1,0,1 to set different settings for each of up to 3 parallel ports.

### 6.12.2 Контакты

In the following pins, parameters, and functions, <port> is the parallel port ID. According to the naming conventions the first port should always have an ID of zero. All the boards have some method of setting the address on the EPP bus. USC and UPC have simple provisions for only two addresses, but jumper foil cuts allow up to 4 boards to be addressed. The PPMC boards have 16 possible addresses. In all cases, the driver enumerates the boards by type and exports the appropriate HAL pins. For instance, the encoders will be enumerated from zero up, in the same order as the address switches on the board specify. So, the first board will have encoders 0 — 3, the second board would have encoders 4 — 7. The first column after the bullet tells which boards will have this HAL pin or parameter associated with it. All means that this pin is available on all three board types. Option means that this pin will only be exported when that option is enabled by an optional parameter in the loadrt HAL command. These options require the board to have a sufficient revision level to support the feature.

- *(All s32 output) ppmc.<port>.encoder.<channel>.count* - Encoder position, in counts.
- *(All s32 output) ppmc.<port>.encoder.<channel>.delta* - Change in counts since last read, in raw encoder count units.
- *(All float output) 'ppmc.<port>.encoder.<channel>.velocity* - Velocity scaled in user units per second. On PPMC and USC this is derived from raw encoder counts per servo period, and hence is affected by encoder granularity. On UPC boards with the 8/21/09 and later firmware, velocity estimation by timestamping encoder counts can be used to improve the smoothness of this velocity output. This can be fed to the PID HAL component to produce a more stable servo response. This function has to be enabled in the HAL command line that starts the PPMC driver, with the timestamp=0x00 option.
- *(All float output) ppmc.<port>.encoder.<channel>.position* - Encoder position, in user units.
- *(All bit bidir) ppmc.<port>.encoder.<channel>.index-enable* - Connect to joint.#.index-enable for home-to-index. This is a bidirectional HAL signal. Setting it to true causes the encoder hardware to reset the count to zero on the next encoder index pulse. The driver will detect this and set the signal back to false.
- *(PPMC float output) ppmc.<port>.DAC.<channel>.value* - sends a signed value to the 16-bit Digital to Analog Converter on the PPMC DAC16 board commanding the analog output voltage of that DAC channel.
- *(UPC bit input) ppmc.<port>.pwm.<channel>.enable* - Enables a PWM generator.
- *(UPC float input) ppmc.<port>.pwm.<channel>.value* - Value which determines the duty cycle of the PWM waveforms. The value is divided by *pwm.<channel>.scale*, and if the result is 0.6 the duty cycle will be 60%, and so on. Negative values result in the duty cycle being based on the absolute value, and the direction pin is set to indicate negative.
- *(USC bit input) ppmc.<port>.stepgen.<channel>.enable* - Enables a step pulse generator.
- *(USC float input) ppmc.<port>.stepgen.<channel>.velocity* - Value which determines the step frequency. The value is multiplied by *stepgen.<channel>.scale* , and the result is the frequency in steps per second. Negative values result in the frequency being based on the absolute value, and the direction pin is set to indicate negative.

- *(All bit output) ppmc.<port>.din.<channel>.in* - State of digital input pin, see canonical digital input.
- *(All bit output) ppmc.<port>.din.<channel>.in-not* - Inverted state of digital input pin, see canonical digital input.
- *(All bit input) ppmc.<port>.dout.<channel>.out* - Value to be written to digital output, see canonical digital output.
- *(Option float input) ppmc.<port>.DAC8-<channel>.value* - Value to be written to analog output, range from 0 to 255. This sends 8 output bits to J8, which should have a Spindle DAC board connected to it. 0 corresponds to zero Volts, 255 corresponds to 10 Volts. The polarity of the output can be set for always minus, always plus, or can be controlled by the state of SSR1 (plus when on) and SSR2 (minus when on). You must specify extradac = 0x00 on the HAL command line that loads the PPMC driver to enable this function on the first USC ur UPC board.
- *(Option bit input) ppmc.<port>.dout.<channel>.out* - Value to be written to one of the 8 extra digital output pins on J8. You must specify extradout = 0x00 on the HAL command line that loads the ppmc driver to enable this function on the first USC or UPC board. extradac and extradout are mutually exclusive features as they use the same signal lines for different purposes. These output pins will be enumerated after the standard digital outputs of the board.

### 6.12.3 Параметры

- *(All float) ppmc.<port>.encoder.<channel>.scale* - The number of counts / user unit (to convert from counts to units).
- *(UPC float) ppmc.<port>.pwm.<channel-range>.freq* - The PWM carrier frequency, in Hz. Applies to a group of four consecutive PWM generators, as indicated by <*channel-range*>. Minimum is 610 Hz, maximum is 500 kHz.
- *(PPMC float) ppmc.<port>.DAC.<channel>.scale* - Sets scale of DAC16 output channel such that an output value equal to the 1/scale value will produce an output of + or - value Volts. So, if the scale parameter is 0.1 and you send a value of 0.5, the output will be 5.0 Volts.
- *(UPC float) ppmc.<port>.pwm.<channel>.scale* - Scaling for PWM generator. If *scale* is X, then the duty cycle will be 100% when the *value* pin is X (or -X).
- *(UPC float) ppmc.<port>.pwm.<channel>.max-dc* - Maximum duty cycle, from 0.0 to 1.0.
- *(UPC float) ppmc.<port>.pwm.<channel>.min-dc* - Minimum duty cycle, from 0.0 to 1.0.
- *(UPC float) ppmc.<port>.pwm.<channel>.duty-cycle* - Actual duty cycle (used mostly for troubleshooting.)
- *(UPC bit) ppmc.<port>.pwm.<channel>.bootstrap* - If true, the PWM generator will generate a short sequence of pulses of both polarities when E-stop goes false, to reset the shutdown latches on some PWM servo drives.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.setup-time* - Sets minimum time between direction change and step pulse, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-width* - Sets width of step pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs may be specified.
- *(USC u32) ppmc.<port>.stepgen.<channel-range>.pulse-space-min* - Sets minimum time between pulses, in units of 100 ns. Applies to a group of four consecutive step generators, as indicated by <*channel-range*>. Values between 200 ns and 25.5 µs can be specified. The maximum step rate is:

$$\frac{1}{100\text{ns} * (pulsewidth + pulsespacemin)}$$

- *(USC float) ppmc.<port>.stepgen.<channel>.scale* - Scaling for step pulse generator. The step frequency in Hz is the absolute value of *velocity* * *scale*.
- *(USC float) ppmc.<port>.stepgen.<channel>.max-vel* - The maximum value for *velocity*. Commands greater than *max-vel* will be clamped. Also applies to negative values. (The absolute value is clamped.)
- *(USC float) ppmc.<port>.stepgen.<channel>.frequency* - Actual step pulse frequency in Hz (used mostly for troubleshooting.)
- *(Option float) ppmc.<port>.DAC8.<channel>.scale* - Sets scale of extra DAC output such that an output value equal to scale gives a magnitude of 10.0 V output. (The sign of the output is set by jumpers and/or other digital outputs.)
- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output, see canonical digital output.
- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Inverts a digital output pin of J8, see canonical digital output.

### 6.12.4 Функции

- *(All funct) ppmc.<port>.read* - Reads all inputs (digital inputs and encoder counters) on one port. These reads are organized into blocks of contiguous registers to be read in a block to minimize CPU overhead.
- *(All funct) ppmc.<port>.write* - Writes all outputs (digital outputs, stepgens, PWMs) on one port. These writes are organized into blocks of contiguous registers to be written in a block to minimize CPU overhead.

## 6.13 Pluto P Driver

### 6.13.1 General Info

The Pluto-P is a FPGA board featuring the ACEX1K chip from Altera.

#### 6.13.1.1 Requirements

1. A Pluto-P board
2. An EPP-compatible parallel port, configured for EPP mode in the system BIOS or a PCI EPP compatible parallel port card.

---

**Note**
The Pluto P board requires EPP mode. Netmos98xx chips do not work in EPP mode. The Pluto P board will work on some computers and not on others. There is no known pattern to which computers work and which don't work.

---

For more information on PCI EPP compatible parallel port cards see the LinuxCNC Supported Hardware page on the wiki.

#### 6.13.1.2 Connectors

- The Pluto-P board is shipped with the left connector presoldered, with the key in the indicated position. The other connectors are unpopulated. There does not seem to be a standard 12-pin IDC connector, but some of the pins of a 16P connector can hang off the board next to QA3/QZ3.
- The bottom and right connectors are on the same .1" grid, but the left connector is not. If OUT2…OUT9 are not required, a single IDC connector can span the bottom connector and the bottom two rows of the right connector.

#### 6.13.1.3 Physical Pins

- Read the ACEX1K datasheet for information about input and output voltage thresholds. The pins are all configured in *LVTTL/LVCMOS* mode and are generally compatible with 5V TTL logic.
- Before configuration and after properly exiting LinuxCNC, all Pluto-P pins are tristated with weak pull-ups (20 kΩ min, 50 kΩ max). If the watchdog timer is enabled (the default), these pins are also tristated after an interruption of communication between LinuxCNC and the board. The watchdog timer takes approximately 6.5 ms to activate. However, software bugs in the pluto_servo firmware or LinuxCNC can leave the Pluto-P pins in an undefined state.
- In pwm+dir mode, by default dir is HIGH for negative values and LOW for positive values. To select HIGH for positive values and LOW for negative values, set the corresponding dout-NN-invert parameter TRUE to invert the signal.
- The index input is triggered on the rising edge. Initial testing has shown that the QZx inputs are particularly noise sensitive, due to being polled every 25 ns. Digital filtering has been added to filter pulses shorter than 175 ns (seven polling times). Additional external filtering on all input pins, such as a Schmitt buffer or inverter, RC filter, or differential receiver (if applicable) is recommended.
- The IN1…IN7 pins have 22 Ω series resistors to their associated FPGA pins. No other pins have any sort of protection for out-of-spec voltages or currents. It is up to the integrator to add appropriate isolation and protection. Traditional parallel port optoisolator boards do not work with pluto_servo due to the bidirectional nature of the EPP protocol.

#### 6.13.1.4 LED

- When the device is unprogrammed, the LED glows faintly. When the device is programmed, the LED glows according to the duty cycle of PWM0 (*LED = UP0 xor DOWN0*) or STEPGEN0 (*LED = STEP0 xor DIR0*).

#### 6.13.1.5 Power

- A small amount of current may be drawn from VCC. The available current depends on the unregulated DC input to the board. Alternately, regulated +3.3VDC may be supplied to the FPGA through these VCC pins. The required current is not yet known, but is probably around 50mA plus I/O current.
- The regulator on the Pluto-P board is a low-dropout type. Supplying 5V at the power jack will allow the regulator to work properly.

#### 6.13.1.6 PC interface

- Only a single pluto_servo or pluto_step board is supported.

#### 6.13.1.7 Rebuilding the FPGA firmware

The *src/hal/drivers/pluto_servo_firmware/* and *src/hal/drivers/pluto_step_firmware/* subdirectories contain the Verilog source code plus additional files used by Quartus for the FPGA firmwares. Altera's Quartus II software is required to rebuild the FPGA firmware. To rebuild the firmware from the .hdl and other source files, open the *.qpf* file and press CTRL-L. Then, recompile LinuxCNC.

Like the HAL hardware driver, the FPGA firmware is licensed under the terms of the GNU General Public License.

The gratis version of Quartus II runs only on Microsoft Windows, although there is apparently a paid version that runs on Linux.

#### 6.13.1.8 For more information

Some additional information about it is available from KNJC LLC and from the developer's blog.

### 6.13.2 Pluto Servo

The pluto_servo system is suitable for control of a 4-axis CNC mill with servo motors, a 3-axis mill with PWM spindle control, a lathe with spindle encoder, etc. The large number of inputs allows a full set of limit switches.

This driver features:

- 4 quadrature channels with 40 MHz sample rate. The counters operate in *4x* mode. The maximum useful quadrature rate is 8191 counts per LinuxCNC servo cycle, or about 8 MHz for LinuxCNC's default 1 ms servo rate.
- 4 PWM channels, *up/down* or *pwm+dir* style. 4095 duty cycles from -100% to +100%, including 0%. The PWM period is approximately 19.5 kHz (40 MHz / 2047). A PDM-like mode is also available.
- 18 digital outputs: 10 dedicated, 8 shared with PWM functions. (Example: A lathe with unidirectional PWM spindle control may use 13 total digital outputs)
- 20 digital inputs: 8 dedicated, 12 shared with Quadrature functions. (Example: A lathe with index pulse only on the spindle may use 13 total digital inputs.)
- EPP communication with the PC. The EPP communication typically takes around 100 µs on machines tested so far, enabling servo rates above 1 kHz.

#### 6.13.2.1 Pinout

- *UPx* - The *up* (up/down mode) or *pwm* (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is always negative. The corresponding digital output invert may be set to TRUE to make UPx active low rather than active high.
- *DNx* - The *down* (up/down mode) or *direction* (pwm+direction mode) signal from PWM generator X. May be used as a digital output if the corresponding PWM channel is unused, or the output on the channel is never negative. The corresponding digital output invert may be set to TRUE to make DNx active low rather than active high.
- *QAx, QBx* - The A and B signals for Quadrature counter X. May be used as a digital input if the corresponding quadrature channel is unused.
- *QZx* - The Z (index) signal for quadrature counter X. May be used as a digital input if the index feature of the corresponding quadrature channel is unused.

- *INx* - Dedicated digital input #x
- *OUTx* - Dedicated digital output #x
- *GND* - Ground
- *VCC* - +3.3V regulated DC

.Pluto-Servo Pinout Pluto-Servo Pinout

Table 6.41: Pluto-Servo Alternate Pin Functions

| **Primary function** | **Alternate Function** | **Behavior if both functions used** |
|---|---|---|
| **UP0** | PWM0 | When pwm-0-pwmdir is TRUE, this pin is the PWM output |
| | OUT10 | XOR'd with UP0 or PWM0 |
| **UP1** | PWM1 | When pwm-1-pwmdir is TRUE, this pin is the PWM output |
| | OUT12 | XOR'd with UP1 or PWM1 |
| **UP2** | PWM2 | When pwm-2-pwmdir is TRUE, this pin is the PWM output |
| | OUT14 | XOR'd with UP2 or PWM2 |
| **UP3** | PWM3 | When pwm-3-pwmdir is TRUE, this pin is the PWM output |
| | OUT16 | XOR'd with UP3 or PWM3 |
| **DN0** | DIR0 | When pwm-0-pwmdir is TRUE, this pin is the DIR output |
| | OUT11 | XOR'd with DN0 or DIR0 |
| **DN1** | DIR1 | When pwm-1-pwmdir is TRUE, this pin is the DIR output |
| | OUT13 | XOR'd with DN1 or DIR1 |
| **DN2** | DIR2 | When pwm-2-pwmdir is TRUE, this pin is the DIR output |
| | OUT15 | XOR'd with DN2 or DIR2 |
| **DN3** | DIR3 | When pwm-3-pwmdir is TRUE, this pin is the DIR output |
| | OUT17 | XOR'd with DN3 or DIR3 |
| **QZ0** | IN8 | Read same value |
| **QZ1** | IN9 | Read same value |
| **QZ2** | IN10 | Read same value |
| **QZ3** | IN11 | Read same value |
| **QA0** | IN12 | Read same value |
| **QA1** | IN13 | Read same value |
| **QA2** | IN14 | Read same value |
| **QA3** | IN15 | Read same value |
| **QB0** | IN16 | Read same value |
| **QB1** | IN17 | Read same value |
| **QB2** | IN18 | Read same value |

Table 6.41: (continued)

| Primary function | Alternate Function | Behavior if both functions used |
|---|---|---|
| **QB3** | IN19 | Read same value |

#### 6.13.2.2 Input latching and output updating

- PWM duty cycles for each channel are updated at different times.
- Digital outputs OUT0 through OUT9 are all updated at the same time. Digital outputs OUT10 through OUT17 are updated at the same time as the pwm function they are shared with.
- Digital inputs IN0 through IN19 are all latched at the same time.
- Quadrature positions for each channel are latched at different times.

#### 6.13.2.3 HAL Functions, Pins and Parameters

A list of all *loadrt* arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_servo.9*.

#### 6.13.2.4 Compatible driver hardware

A schematic for a 2A, 2-axis PWM servo amplifier board is available from the (the software developer). The L298 H-Bridge can be used for motors up to 4A (one motor per L298) or up to 2A (two motors per L298) with the supply voltage up to 46V. However, the L298 does not have built-in current limiting, a problem for motors with high stall currents. For higher currents and voltages, some users have reported success with International Rectifier's integrated high-side/low-side drivers.

### 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.

The board features:

- 4 *step+direction* channels with 312.5 kHz maximum step rate, programmable step length, space, and direction change times
- 14 dedicated digital outputs
- 16 dedicated digital inputs
- EPP communication with the PC

#### 6.13.3.1 Pinout

- *STEPx* - The *step* (clock) output of stepgen channel *x*
- *DIRx* - The *direction* output of stepgen channel *x*
- *INx* - Dedicated digital input #x
- *OUTx* - Dedicated digital output #x
- *GND* - Ground
- *VCC* - +3.3V regulated DC

While the *extended main connector* has a superset of signals usually found on a Step & Direction DB25 connector—4 step generators, 9 inputs, and 6 general-purpose outputs—the layout on this header is different than the layout of a standard 26-pin ribbon cable to DB25 connector.

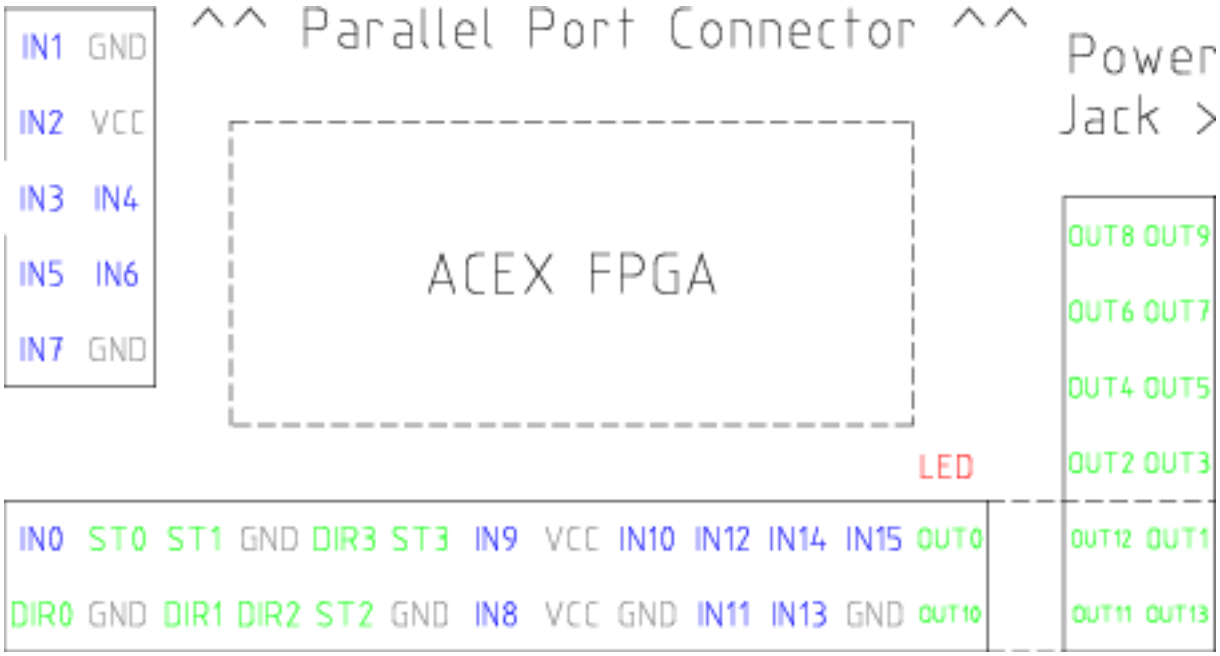



Figure 6.10: Pluto-Step Pinout

#### 6.13.3.2 Input latching and output updating

- Step frequencies for each channel are updated at different times.
- Digital outputs are all updated at the same time.
- Digital inputs are all latched at the same time.
- Feedback positions for each channel are latched at different times.

#### 6.13.3.3 Step Waveform Timings

The firmware and driver enforce step length, space, and direction change times. Timings are rounded up to the next multiple of 1.6µs, with a maximum of 49.6µs. The timings are the same as for the software stepgen component, except that *dirhold* and *dirsetup* have been merged into a single parameter

*dirtime* which should be the maximum of the two, and that the same step timings are always applied to all channels.

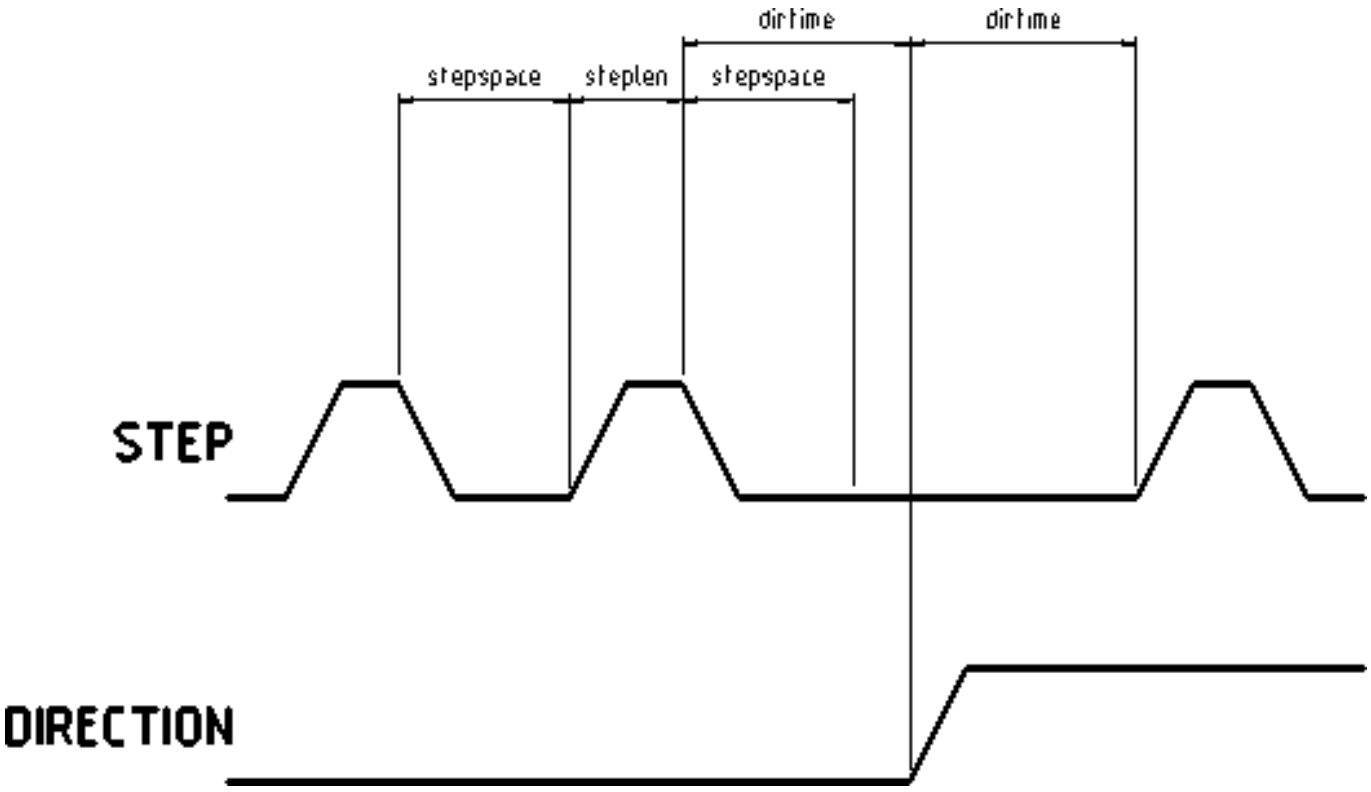



Figure 6.11: Pluto-Step Timings

#### 6.13.3.4 HAL Functions, Pins and Parameters

A list of all *loadrt* arguments, HAL function names, pin names and parameter names is in the manual page, *pluto_step.9*.

## 6.14 Powermax Modbus Driver

This is a non-realtime HAL program, written in python, to control Hypetherm Powermax plasma cutters using the Modbus ASCII protocol over RS485.

**Note**
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.

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn pmx485 pmx485 /dev/ttyUSB0
```

This will load the pmx485 component using the /dev/ttyUSB0 port and wait for it to become ready.

It is necessary to name the port to use for communications.

### 6.14.1 Контакты

- **pmx485.mode-set** (bit, in) # set cutting mode
- **pmx485.current-set** (bit, in) # set cutting current
- **pmx485.pressure-set** (bit, in) # set gas pressure
- **pmx485.enable** (bit, in) # enable the component
- **pmx485.mode** (bit, out) # cut mode feedback
- **pmx485.current** (bit, out) # cutting current feedback
- **pmx485.pressure** (bit, out) # gas pressure feedback
- **pmx485.fault** (bit, out) # powermax fault code
- **pmx485.status** (bit, out) # connection status
- **pmx485.current-min** (bit, out) # minimum allowed current
- **pmx485.current-max** (bit, out) # maximum allowed current
- **pmx485.pressure-min** (bit, out) # minimum allowed gas pressure
- **pmx485.pressure-max** (bit, out) # maximum allowed gas pressure

### 6.14.2 Description

To communicate with a Powermax, the component must first be enabled via the **enable** pin and it may then initiate a request to the Powermax by writing a valid string to the following pins:

- **mode-set**
- **current-set**
- **pressure-set**

---

**Note**
A **pressure-set** value of zero is valid, the Powermax will then calculate the required pressure internally.

---

Communications may be validated from the Powermax display or the **status** pin. While in remote mode, the mode, current and pressure may be changed as needed.

To terminate the communications, do one of the following:

- Set all set pins to zero: **mode-set**, **current-set**, and **pressure-set**.
- Disconnect the Powermax power supply from its power source for approximately 30 seconds. When you power the system back ON, it will no longer be in remote mode.

### 6.14.3 Reference:

- Hypertherm Application Note #807220
  ”Powermax45 XP/65/85/105/125® Serial Communication Protocol”

## 6.15 Servo To Go Driver

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.

---

**Note**
We have had reports that the opamps on the Servo To Go card do not work with newer ATX power supplies that use modern switch mode DC-DC converters. The failure mode is that STG card outputs a constant voltage regardless of what the driver is commanding it to do. Older ATX power supplies with linear voltage regulators do not have this problem, and work fine with the STG cards.

---

### 6.15.1 Installing

```
loadrt hal_stg [base=<address>] [num_chan=<nr>] [dio="<dio-string>"] \
                  [model=<model>]
```

The base address field is optional; if it's not provided the driver attempts to autodetect the board. The num_chan field is used to specify the number of channels available on the card, if not used the 8 axis version is assumed. The digital inputs/outputs configuration is determined by a config string passed to insmod when loading the module. The format consists of a four character string that sets the direction of each group of pins. Each character of the direction string is either "I" or "O". The first character sets the direction of port A (Port A - DIO.0-7), the next sets port B (Port B - DIO.8-15), the next sets port C (Port C - DIO.16-23), and the fourth sets port D (Port D - DIO.24-31). The model field can be used in case the driver doesn't autodetect the right card version.

HINT: after starting up the driver, *dmesg* can be consulted for messages relevant to the driver (e.g. autodetected version number and base address). For example:

```
loadrt hal_stg base=0x300 num_chan=4 dio="IOIO"
```

This example installs the STG driver for a card found at the base address of 0x300, 4 channels of encoder feedback, DACs and ADCs, along with 32 bits of I/O configured like this: the first 8 (Port A) configured as Input, the next 8 (Port B) configured as Output, the next 8 (Port C) configured as Input, and the last 8 (Port D) configured as Output

```
loadrt hal_stg
```

This example installs the driver and attempts to autodetect the board address and board model, it installs 8 axes by default along with a standard I/O setup: Port A & B configured as Input, Port C & D configured as Output.

### 6.15.2 Контакты

- *stg.<channel>.counts* - (s32) Tracks the counted encoder ticks.
- *stg.<channel>.position* - (float) Outputs a converted position.
- *stg.<channel>.dac-value* - (float) Drives the voltage for the corresponding DAC.
- *stg.<channel>.adc-value* - (float) Tracks the measured voltage from the corresponding ADC.
- *stg.in-<pinnum>* - (bit) Tracks a physical input pin.

- *stg.in-<pinnum>-not* - (bit) Tracks a physical input pin, but inverted.
- *stg.out-<pinnum>* - (bit) Drives a physical output pin

For each pin, <channel> is the axis number, and <pinnum> is the logic pin number of the STG if `II00` is defined, there are 16 input pins (in-00 .. in-15) and 16 output pins (out-00 .. out-15), and they correspond to PORTs ABCD (in-00 is PORTA.0, out-15 is PORTD.7).

The in-<pinnum> HAL pin is TRUE if the physical pin is high, and FALSE if the physical pin is low. The in-<pinnum>-not HAL pin is inverted — it is FALSE if the physical pin is high. By connecting a signal to one or the other, the user can determine the state of the input.

### 6.15.3 Параметры

- *stg.<channel>.position-scale* - (float) The number of counts / user unit (to convert from counts to units).
- *stg.<channel>.dac-offset* - (float) Sets the offset for the corresponding DAC.
- *stg.<channel>.dac-gain* - (float) Sets the gain of the corresponding DAC.
- *stg.<channel>.adc-offset* - (float) Sets the offset of the corresponding ADC.
- *stg.<channel>.adc-gain* - (float) Sets the gain of the corresponding ADC.
- *stg.out-<pinnum>-invert* - (bit) Inverts an output pin.

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.

### 6.15.4 Функции

- *stg.capture-position* - Reads the encoder counters from the axis <channel>.
- *stg.write-dacs* - Writes the voltages to the DACs.
- *stg.read-adcs* - Reads the voltages from the ADCs.
- *stg.di-read* - Reads physical in- pins of all ports and updates all HAL in-<pinnum> and in-<pinnum>-not pins.
- *stg.do-write* - Reads all HAL out-<pinnum> pins and updates all physical output pins.

## 6.16 Shuttle

### 6.16.1 Description

Shuttle is a non-realtime HAL component that interfaces Contour Design's ShuttleXpress, ShuttlePRO, and ShuttlePRO2 devices with LinuxCNC's HAL.

If the driver is started without command-line arguments, it will probe all /dev/hidraw* device files for Shuttle devices, and use all devices found. If it is started with command-line arguments, it will only probe the devices specified.

The ShuttleXpress has five momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

The ShuttlePRO has 13 momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

The ShuttlePRO2 has 15 momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

**Warning**
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.
This means that if the first event is generated by a jog-wheel move, that first move will be lost. 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 Setup

The shuttle driver needs read permission to the /dev/hidraw* device files. This can be accomplished by adding a file /etc/udev/rules.d/99-shuttle.rules, with the following contents:

```
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”
```

The LinuxCNC Debian package installs an appropriate udev file automatically, but if you are building LinuxCNC from source and are not using the Debian packaging you'll need to install this file by hand. If you install the file by hand you'll need to tell udev to reload its rules files by running `udevadm control --reload-rules`.

### 6.16.3 Контакты

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)**
These pins are True (1) when the button is pressed.

**<*Prefix*>.button-<ButtonNumber>-not (bit out)**
These pins have the inverse of the button state, so they're True (1) when the button is not pressed.

**<*Prefix*>.counts (s32 out)**
Accumulated counts from the jog wheel (the inner wheel).

**<*Prefix*>.spring-wheel-s32 (s32 out)**
The current deflection of the spring-wheel (the outer wheel). It's 0 at rest, and ranges from -7 at the counter-clockwise extreme to +7 at the clockwise extreme.

**<*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 Driver

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.

Regardless of the connection type, vfs11_vfd operates as a Modbus master.

This component is loaded using the halcmd ”loadusr” command:

```
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
```

The above command says: loadusr, wait for named to load, component vfs11_vfd, named spindle-vfd

## 6.17.1 Command Line Options

*vfs11_vfd* is mostly configured through INI file options. The command line options are:

- *-n or --name <halname>* : set the HAL component name
- *-I or --ini <inifilename>* : take configuration from this INI file. Defaults to environment variable INI_FILE_NAME.
- *-S or --section <section name>* : take configuration from this section in the INI file. Defaults to *VFS11*.
- *-d or --debug* enable debug messages on console output.
- *-m or --modbus-debug* enable modbus messages on console output
- *-r or --report-device* report device properties on console at startup

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` ).

**Note**
That if there are serial configuration errors, turning on verbose may result in a flood of timeout errors.

## 6.17.2 Контакты

Where <n> is `vfs11_vfd` or the name given during loading with the -n option.

- *<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) when drive is at commanded speed (see speed-tolerance below)
- *<n>.current-load-percentage* (float, out) reported from the VFD

- *<n>.dc-brake* (bit, in) engage the DC brake. Also turns off spindle-on.
- *<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) current target frequency in Hz as set through speed-command (which is in RPM), from the VFD
- *<n>.frequency-out* (float, out) current output frequency of the VFD
- *<n>.inverter-load-percentage* (float, out) current load report from 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 when the VFD reports 0 Hz output
- *<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 when the Modbus session is successfully established and the last 10 transactions returned without error.
- *<n>.motor-RPM* (float, out) estimated current RPM value, from the VFD
- *<n>.output-current-percentage* (float, out) from the VFD
- *<n>.output-voltage-percentage* (float, out) from the VFD
- *<n>.output-voltage* (float, out) from the VFD
- *<n>.speed-command* (float, in) speed sent to VFD in RPM. It is an error to send a speed faster than the Motor Max RPM as set in the VFD
- *<n>.spindle-fwd* (bit, in) 1 for FWD and 0 for REV, sent to VFD
- *<n>.spindle-on* (bit, in) 1 for ON and 0 for OFF sent to VFD, only on when running
- *<n>.spindle-rev* (bit, in) 1 for ON and 0 for OFF, only on when running
- *<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) trip code if VF-S11 is in tripped state.
- *<n>.error-count* (s32, out) number of Modbus transactions which returned an error
- *<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 Параметры

Where <n> is `vfs11_vfd` or the name given during loading with the -n option.

- *<n>.frequency-limit* (float, RO) upper limit read from VFD setup.
- *<n>.loop-time* (float, RW) how often the Modbus is polled (default interval 0.1 seconds)
- *<n>.nameplate-HZ* (float, RW) Nameplate Hz of motor (default 50). Used to calculate target frequency (together with nameplate-RPM ) for a target RPM value as given by speed-command.
- *<n>.nameplate-RPM* (float, RW) Nameplate RPM of motor (default 1410)
- *<n>.rpm-limit* (float, RW) do-not-exceed soft limit for motor RPM (defaults to nameplate-RPM ).
- *<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 file configuration

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]
# serial connection
TYPE=rtu

# serial port
DEVICE=/dev/ttyS0

# TCP server - wait for incoming connection
TYPE=tcpserver

# tcp portnumber for TYPE=tcpserver or tcpclient
PORT=1502

# TCP client - active outgoing connection
TYPE=tcpclient

# destination to connect to if TYPE=tcpclient
TCPDEST=192.168.1.1

#---------- meaningful only if TYPE=rtu -------
# serial device detail
# 5 6 7 8
BITS= 5

# even odd none
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
# this feature might not work with a stock Ubuntu
```

```
# libmodbus5/libmodbus-dev package, and generate a warning
# execution will continue as if RTS_MODE=up were given.
RTS_MODE=up
#---------------------

# modbus timers in seconds
# inter-character timer
BYTE_TIMEOUT=0.5
# packet timer
RESPONSE_TIMEOUT=0.5

# target modbus ID
TARGET=1

# on I/O failure, try to reconnect after sleeping
# for RECONNECT_DELAY seconds
RECONNECT_DELAY=1

# misc. parameters
DEBUG=10
MODBUS_DEBUG=0
POLLCYCLES=10
```

### 6.17.5 HAL example

```
#
# example usage of the VF-S11 VFD driver
#
#
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd

# connect the spindle direction pins to the VFD
net vfs11-fwd spindle-vfd.spindle-fwd <= spindle.0.forward
net vfs11-rev spindle-vfd.spindle-rev <= spindle.0.reverse

# connect the spindle on pin to the VF-S11
net vfs11-run spindle-vfd.spindle-on <= spindle.0.on

# connect the VF-S11 at speed to the motion at speed
net vfs11-at-speed spindle.0.at-speed <= spindle-vfd.at-speed

# connect the spindle RPM to the VF-S11
net vfs11-RPM spindle-vfd.speed-command <= spindle.0.speed-out

# connect the VF-S11 DC brake
# since this draws power during spindle off, the dc-brake pin would
# better be driven by a monoflop which triggers on spindle-on falling edge
#net vfs11-spindle-brake spindle.N.brake => spindle-vfd.dc-brake

# to use the VFS11 jog mode for spindle orient
# see orient.9 and motion.9
net spindle-orient spindle.0.orient spindle-vfd.max-speed spindle-vfd.jog-mode

# take precedence over control panel
setp spindle-vfd.enable 1
```

### 6.17.6 Panel operation

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 Error Recovery

`vfs11_vfd` recovers from I/O errors as follows: First, all HAL pins are set to default values, and the driver will sleep for `RECONNECT_DELAY` seconds (default 1 second).

- Serial (`TYPE=rtu`) mode: on error, close and reopen the serial port.
- TCP server (`TYPE=tcpserver`) mode: on losing the TCP connection, the driver will go back to listen for incoming connections.
- TCP client (`TYPE=tcpclient`) mode: on losing the TCP connection, the driver will reconnect to *TCPDEST:PORTNO*.

### 6.17.8 Configuring the VFS11 VFD for Modbus usage

#### 6.17.8.1 Connecting the Serial Port

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).

Note: the 24V output from the VFD has no short-circuit protection.

Serial port factory defaults are 9600/8/1/even, the protocol defaults to the proprietary ”Toshiba Inverter Protocol”.

#### 6.17.8.2 Modbus setup

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 Programming Note

The vfs11_vfd driver uses the libmodbus version 3 library which is more recent than the version 2 code used in `gs2_vfd`.

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.

To use the full functionality on lucid and precise:

- remove the libmodbus packages: `sudo apt-get remove libmodbus5 libmodbus-dev`
- build and install libmodbus version 3 from source as outlined here.

Libmodbus does not build on Ubuntu Hardy, hence vfs11_vfd is not available on Hardy.

# Chapter 7

# Примеры оборудования

## 7.1 PCI Parallel Port

When you add a second parallel port to your PCI bus you have to find out the address before you can use it with LinuxCNC.

To find the address of your parallel port card open a terminal window and type

```
lspci -v
```

You will see something similar to this as well as info on everything else on the 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 my case the address was the first one so I changed my .hal file from

```
loadrt hal_parport cfg=0x378
```

to

```
loadrt hal_parport cfg=”0x378 0xa800 in”
```

(Note the double quotes surrounding the addresses.)

and then added the following lines so the parport will be read and written:

```
addf parport.1.read base-thread
addf parport.1.write base-thread
```

After doing the above then run your config and verify that the parallel port got loaded in Machine/Show HAL Configuration window.

## 7.2 Управление шпинделем

LinuxCNC может управлять макс. 8-ю шпинделями. Их количество задаётся в INI-файле. Все приведённые ниже примеры относятся к конфигурации с одним шпинделем и контактами управления шпинделями с именами типа *spindle.0...* В случае многошпиндельного станка всё, что меняется, — это наличие дополнительных контактов с именами типа *spindle.6...*

### 7.2.1 Скорость вращения шпинделя 0-10 В

Если скорость вашего шпинделя управляется аналоговым сигналом (например, VFD с сигналом от 0 В до 10 В) и вы используете карту ЦАП, такую как m5i20, для вывода управляющего сигнала:

Сначала необходимо определить масштаб скорости вращения шпинделя для управляющего сигнала, то есть напряжение. В этом примере максимальная скорость вращения шпинделя 5000 об/мин соответствует 10 вольтам.

$$\frac{10\,Volts}{5000\,RPM} = \frac{0.002\,Volts}{1\,RPM}$$

Нам необходимо добавить компонент масштабирования в файл HAL, чтобы масштабировать *spindle.N.s out* до диапазона от 0 до 10, необходимого VFD, если ваша карта ЦАП не поддерживает масштабирован

```
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 => <b''ив''b''мb''b''яв'' b''кb''b''оb''b''нв''b''тb''b' ↩
    'ab''b''кb''b''тb''b''ab'' b''вb''b''ab''b''шb''b''eb''b''гb''b''ob'' b''Цb''b''Ab''b' ↩
    'Пb''>
```

### 7.2.2 Скорость шпинделя ШИМ

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
```

При этом предполагается, что реакция контроллера шпинделя на ШИМ проста: 0% ШИМ дает 0 об/мин, 10% ШИМ дает 180 об/мин и т. д. Если для вращения шпинделя требуется минимальная ШИМ, следуйте примеру в образце конфигурации nist-lathe, чтобы использовать компонент шкалы.

### 7.2.3 Разрешить шпиндель

Если вам нужен сигнал включения шпинделя, подключите выходной контакт к *spindle.0.on*. Чтобы связать эти контакты с контактом параллельного порта, добавьте в файл .hal примерно следующее, убедившись, что выбранный вами контакт подключен к устройству управления.

```
net spindle-enable spindle.0.on => parport.0.pin-14-out
```

### 7.2.4 Направление вращения шпинделя

Если у вас есть управление направлением вращения шпинделя, то выводы HAL *spindle*.N.*forward* и *spindle*.N.*reverse* управляются G-кодами M3 и M4. Для включения движения шпинделя с помощью команд M3/M4 скорость вращения шпинделя S*n* должна быть установлена на положительное ненулевое значение.

Чтобы связать эти контакты с контактом параллельного порта, поместите в файл .hal что-то вроде следующего, убедившись, что вы выбрали контакт, подключенный к вашему устройству управления.

```
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 Плавный пуск шпинделя

Если вам нужно увеличить скорость вращения шпинделя, а ваша система управления не поддерживает такую функцию, это можно сделать в HAL. По сути, вам нужно перехватить выходной сигнал *spindle*.N.*speed-out* и пропустить его через компонент *limit2* с заданным масштабом, чтобы он увеличивал скорость вращения от *spindle*.N.*speed-out* до вашего устройства, принимающего эти данные. Вторая часть — сообщить LinuxCNC, когда шпиндель достигнет нужной скорости, чтобы можно было начать движение.

В примере 0-10 Вольт линия

```
net spindle-speed-scale spindle.0.speed-out => scale.0.in
```

изменяется, как показано в следующем примере:

**Введение в компоненты HAL limit2 и около** Если вы раньше с ними не сталкивались, вот краткое введение в два компонента HAL, используемые в следующем примере.

- *limit2* — это компонент HAL (с плавающей точкой), который принимает входное значение и предоставляет выходное значение, ограниченное максимальным значением.
- *Near* — это компонент HAL (с плавающей точкой) с двоичным выходом, который определяет, приблизительно ли равны два входных сигнала.

Более подробную информацию можно найти в документации по компонентам HAL или на страницах руководства, просто введите *man limit2* или *man near* в терминале.

```
# load the real time modules limit2 and near with names so it is easier to follow their  ↩
    connections
loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed

# add the functions to a thread
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread

# set the parameter for max rate-of-change
# (max spindle accel/decel in units per second)
setp spindle-ramp.maxv 60

# hijack the spindle speed out and send it to spindle ramp in
net spindle-cmd <= spindle.0.speed-out => spindle-ramp.in

# the output of spindle ramp is sent to the scale in
net spindle-ramped <= spindle-ramp.out => scale.0.in
```

```
# to know when to start the motion we send the near component
# (named spindle-at-speed) to the spindle commanded speed from
# the signal spindle-cmd and the actual spindle speed
# provided your spindle can accelerate at the maxv setting.
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2

# the output from spindle-at-speed is sent to spindle.0.at-speed
# and when this is true motion will start
net spindle-ready <= spindle-at-speed.out => spindle.0.at-speed
```

## 7.2.6 Обратная связь шпинделя

### 7.2.6.1 Синхронизированное движение шпинделя

Обратная связь шпинделя необходима LinuxCNC для выполнения любых координируемых шпинделем движений, таких как нарезание резьбы и поддержание постоянной скорости резания. LinuxCNC может выполнять синхронизированное движение и CSS с любым из восьми шпинделей. Выбор используемого шпинделя определяется G-кодом. CSS возможна одновременно с несколькими шпинделями.

Мастер StepConf может выполнить для вас подключения для конфигурации с одним шпинделем, если вы выберете в качестве входов фазу A энкодера и Z индекс энкодера.

Предположения относительно оборудования для этого примера:

- К шпинделю подключен энкодер, который выдает 100 импульсов за один оборот в фазе A.
- Фаза энкодера A подключена к контакту 10 параллельного порта.
- Индексный импульс энкодера подключается к контакту 11 параллельного порта.

Основные шаги по добавлению компонентов и их настройке: сноска: [В этом примере мы предположим что некоторые энкодеры уже назначены осям/сочленениям 0, 1 и 2. Поэтому следующий энкодер, который мы сможем присоединить к шпинделю, будет иметь номер 3. Ваша ситуация может отличаться.] footnote: [Включение индекса энкодера HAL является исключением из правила, поскольку он ведет себя и как вход, и как выход; подробности см. в разделе Encoder Section] footnote: [Так как выше мы выбрали *неквадратурный простой подсчет...*, мы можем обойтись *квадратурным* подсчетом без использования квадратурного входа B.]

```
# Add the encoder to HAL and attach it to threads.
loadrt encoder num_chan=4
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread

# Set the HAL encoder to 100 pulses per revolution.
setp encoder.3.position-scale 100

# Set the HAL encoder to non-quadrature simple counting using A only.
setp encoder.3.counter-mode true

# Connect the HAL encoder outputs to 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

# Connect the HAL encoder inputs to the real encoder.
net spindle-phase-a encoder.3.phase-A <= parport.0.pin-10-in
```

```
net spindle-phase-b encoder.3.phase-B
net spindle-index encoder.3.phase-Z <= parport.0.pin-11-in
```

#### 7.2.6.2 Шпиндель достиг скорости (Spindle At Speed)

Чтобы LinuxCNC мог ожидать набора шпинделем заданной скорости перед выполнением серии перемещений, значение spindle.*N*.at-speed должно быть равно true в тот момент, когда шпиндель достигает заданной скорости. Для этого необходима обратная связь от энкодера. Поскольку обратная связь и заданная скорость обычно не совпадают *точно*, следует использовать компонент *near* для определения того, насколько близки эти два значения.

Необходимые соединения: сигнал команды скорости шпинделя – к near.n.in1, а сигнал скорости шпинделя – к near.n.in2. Затем сигнал near.n.out подключается к spindle.*N*.at-speed. Параметр near.n.scale необходимо настроить так, чтобы он указывал, насколько близко должны быть два числа друг к другу, прежде чем будет включен выход. В зависимости от конфигурации может потребоваться настройка масштаба для работы с вашим оборудованием.

Ниже приведены типичные дополнения, которые необходимо добавить в файл HAL для включения функции Spindle At Speed. Если в файле HAL уже есть сигнал Near, увеличьте его количество и скорректируйте код соответствующим образом. Убедитесь, что названия сигналов в файле HAL совпадают.

```
# load a near component and attach it to a thread
loadrt near
addf near.0 servo-thread

# connect one input to the commanded spindle speed
net spindle-cmd => near.0.in1

# connect one input to the encoder-measured spindle speed
net spindle-velocity => near.0.in2

# connect the output to the spindle-at-speed input
net spindle-at-speed spindle.0.at-speed <= near.0.out

# set the spindle speed inputs to agree if within 1%
setp near.0.scale 1.01
```

## 7.3 MPG Pendant

This example is to explain how to hook up the common MPG pendants found on the market today. This example uses an MPG3 pendant and a C22 pendant interface card from CNC4PC connected to a second parallel port plugged into the PCI slot. This example gives you 3 axes with 3 step increments of 0.1, 0.01, 0.001

In your custom.hal file or jog.hal file add the following, making sure you don't have mux4 or an encoder already in use. If you do just increase the counts and change the reference numbers. More information about mux4 and encoder can be found in the HAL manual or the man page.

See the INI HAL Section of the documentation for more information on adding a HAL file. Jog management pins are provided for each joint and all coordinate letters. This example uses the axis jog pins for jogging in world mode. Machines with non-identity kinematics may need use additional connections for jogging in joint mode.

**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

# If your MPG outputs a quadrature signal per click set x4 to 1
# If your MPG puts out 1 pulse per click set x4 to 0
setp encoder.0.x4-mode 0

# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
# For position mode (the default), set to 0
# In position mode the axis will move exactly jog-scale
# units for each count, regardless of how long that might take,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# This sets the scale that will be used based on the input to the mux4
setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001

# The inputs to the mux4 component
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# The output from the mux4 is sent to each axis jog scale
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

# The MPG inputs
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 Axis select inputs
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

# The encoder output counts to the axis. Only the selected axis will move.
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
```

If the machine is capable of high acceleration to smooth out the moves for each click of the MPG use the ilowpass component to limit the acceleration.

**jog.hal with 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

# If your MPG outputs a quadrature signal per click set x4 to 1
# If your MPG puts out 1 pulse per click set x4 to 0
setp encoder.0.x4-mode 0

# For velocity mode, set to 1
# In velocity mode the axis stops when the dial is stopped
# even if that means the commanded motion is not completed,
# For position mode (the default), set to 0
# In position mode the axis will move exactly jog-scale
# units for each count, regardless of how long that might take,
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# This sets the scale that will be used based on the input to the mux4
# The scale used here has to be multiplied by the ilowpass scale
setp mux4.0.in0 0.0001
setp mux4.0.in1 0.00001
setp mux4.0.in2 0.000001

# The inputs to the mux4 component
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# The output from encoder counts is sent to ilowpass
net mpg-out ilowpass.0.in <= encoder.0.counts

# The output from the mux4 is sent to each axis jog scale
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

# The MPG inputs
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 Axis select inputs
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

# The output from the ilowpass is sent to each axis jog count
# Only the selected axis will move.
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 Spindle

## 7.4.1 Пример

This example shows the connections needed to use an Automation Direct GS2 VFD to drive a spindle. The spindle speed and direction is controlled by LinuxCNC.

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 the custom.hal file we place the following to connect LinuxCNC to the GS2 and have LinuxCNC control the drive.

**GS2 Example**

```
# 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
```

**Note**
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.

- The communications switches must be set to RS-232C.
- The motor parameters must be set to match the motor.
- P3.00 (Source of Operation Command) must be set to Operation determined by RS-485 interface, 03 or 04.
- P4.00 (Source of Frequency Command) must be set to Frequency determined by RS232C/RS485 communication interface, 05.
- P9.01 (Transmission Speed) must be set to 9600 baud, 01.
- P9.02 (Communication Protocol) must be set to ”Modbus RTU mode, 8 data bits, no parity, 2 stop bits”, 03.

A PyVCP panel based on this example is here.

# Chapter 8

# Язык релейных схем

## 8.1 ClassicLadder введение

### 8.1.1 История

ClassicLadder — это бесплатная реализация интерпретатора языка релейных схем, выпущенная под лицензией LGPL. Он был написан Марком Ле Дуарен.

Он описывает начало проекта на своем сайте:

> Я решил программировать язык релейных схем только в целях тестирования на старте, в феврале 2001 года. Планировалось, что мне придется заняться новым продуктом после ухода с предприятия, на котором я тогда работал. И я подумал, что неплохо было бы рассмотреть возможность использования в этих продуктах языка релейных схем. Итак, я начал писать первые строки кода для расчета ступени с минимальным количеством элементов и ее динамического отображения под Gtk, чтобы проверить, работает ли моя первая идея реализовать все это.
>
> И как только я обнаружил, что продвинулся достаточно хорошо, я продолжил работу с более сложными элементами: таймером, множественными ступенями и т.д.
>
> Вуаля, вот эта работа... и еще: с тех пор я продолжаю добавлять новые функции.
>
> — Марк Ле Дуарен из ”Genesis” на сайте ClassicLadder

ClassicLadder адаптирован для работы с HAL LinuxCNC и в настоящее время распространяется вместе с LinuxCNC. Если есть проблемы/проблемы/ошибки, сообщите о них в проект LinuxCNC.

### 8.1.2 Введение

Лестничная логика или язык программирования Ladder — это метод рисования электрических логических схем. Сейчас это графический язык, очень популярный для программирования программируемых логических контроллеров (ПЛК). Первоначально он был изобретен для описания логики, состоящей из реле. Название основано на наблюдении, что программы на этом языке напоминают лестницы с двумя вертикальными *рельсами* и рядом горизонтальных *ступенек* между ними. В Германии и других странах Европы принято рисовать направляющие горизонтально вверху и внизу страницы, а перекладины — вертикально слева направо.

Программа в релейной логике, также называемая лестничной схемой, аналогична схеме набора релейных схем. Лестничная логика полезна, поскольку из-за сходства ее могут понять и использовать самые разные инженеры и техники без особого дополнительного обучения.

Релейная логика широко используется для программирования ПЛК, где требуется последовательное управление процессом или производственной операцией. Лестничная логика полезна для простых, но важных систем управления или для переделки старых проводных релейных схем. Поскольку программируемые логические контроллеры стали более сложными, они также стали использоваться в очень сложных системах автоматизации.

Лестничную логику можно рассматривать как язык, основанный на правилах, а не как процедурный язык. *Ступенька* лестницы представляет собой правило. При реализации с помощью реле и других электромеханических устройств различные правила *исполняются* одновременно и немедленно. При реализации в программируемом логическом контроллере правила обычно выполняются программным обеспечением последовательно, в цикле. При достаточно быстром выполнении цикла, обычно много раз в секунду, достигается эффект одновременного и немедленного выполнения.

Лестничная логика следует этим общим шагам работы.

- Чтение входов
- Решить логику
- Обновление выходов

### 8.1.3 Пример

Наиболее распространенными компонентами лестничной схемы являются контакты (входы), обычно это NC (нормально замкнутый) или NO (нормально разомкнутый) и coils (катушки) (выходы).

- NO контакт
- NC контакт
- coil (катушка) (выход)

Конечно, в полном языке релейной логики есть еще много компонентов, но понимание их поможет вам понять общую концепцию.

Лестница состоит из одной или нескольких ступеней. Эти ступеньки представляют собой горизонталь дорожки (представляющие провода) с расположенными на них компонентами (входы, выходы и т. д.), которые оцениваются слева направо.

Этот пример представляет собой простейшую ступеньку:

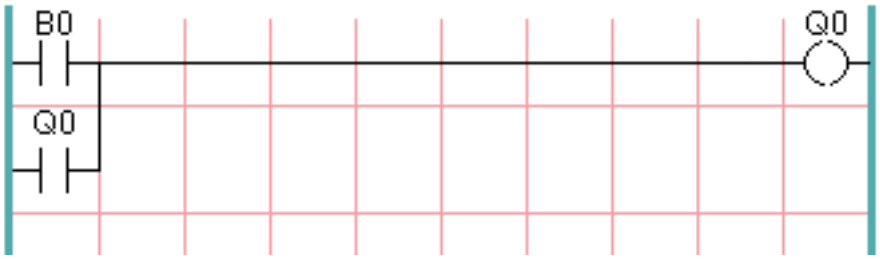
Вход слева, B0, нормально разомкнутый контакт, подключен к катушке (выходу) справа, Q0. Теперь представьте, что напряжение подается на самый левый конец, потому что вход B0 становится true (например, вход активирован или пользователь нажал размыкающий контакт). Напряжение имеет прямой путь к катушке (выходу) справа, Q0. Как следствие, катушка Q0 (выход) изменится с 0/выкл/false на 1/вкл/true. Если пользователь отпускает B0, выход Q0 быстро возвращается в состояние 0/выкл/false.

### 8.1.4 Базовая схема включения-выключения с фиксацией

Опираясь на приведенный выше пример, предположим, что мы добавим переключатель, который замыкается всякий раз, когда катушка Q0 активна. Это будет иметь место в реле, где катушка может активировать контакты переключателя; или в контакторе, где помимо больших трехфазных контактов, которые являются основной особенностью контактора, часто имеется несколько небольших вспомогательных контактов.

Поскольку в нашем предыдущем примере этот вспомогательный переключатель приводится в действие катушкой Q0, мы присвоим ему тот же номер, что и катушке, которая им управляет. Это стандартная практика, которой следуют при любом лестничном программировании, хотя поначалу может показаться странным видеть переключатель, помеченный так же, как катушка. Итак, давайте назовем этот вспомогательный контакт Q0 и подключим его к контакту *pushbutton* B0 из нашего предыдущего примера.

Давайте посмотрим на это:

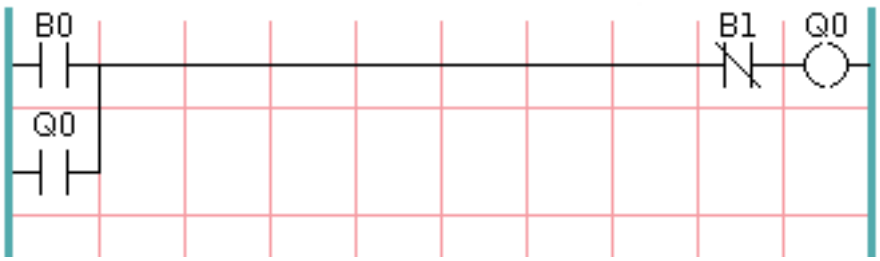



Как и раньше, когда пользователь нажимает кнопку B0, включается катушка Q0. И когда включается катушка Q0, включается переключатель Q0. Теперь происходит самое интересное. Когда пользовател отпускает кнопку B0, катушка Q0 не останавливается, как это было раньше. Это связано с тем, что переключатель Q0 этой схемы фактически удерживает кнопку пользователя нажатой. Итак, мы видим, что переключатель Q0 все еще удерживает катушку Q0 включенной после того, как кнопка *start* была отпущена.

Этот тип контакта на катушке или реле, используемый таким образом, часто называют *holding contact*, поскольку он *holds on* катушку, с которой связан. Его также иногда называют *seal* контактом, и когда он активен, говорят, что цепь *sealed*.

К сожалению, наша схема пока имеет мало практического применения, поскольку, хотя у нас есть кнопка *on* или *start* в виде кнопки B0, у нас нет возможности отключить эту схему после ее запуска. Но это легко исправить. Все, что нам нужно, это способ отключить питание катушки Q0. Итак, давайте добавим нормально замкнутую (NC) кнопку непосредственно перед катушкой Q0.

Вот как это будет выглядеть:




Теперь мы добавили кнопку *off* или *stop* B1. Если пользователь нажимает на нее, контакт между перекладиной и катушкой нарушается. Когда катушка Q0 теряет питание, оно падает до 0/выкл/-false. Когда катушка Q0 отключается, отключается и переключатель Q0, поэтому *holding contact* разрывается или цепь *unsealed*. Когда пользователь отпускает кнопку *stop*, контакт между

ступенькой и катушкой Q0 восстанавливается, но ступенька отключается, поэтому катушка не включается снова.

Эта схема десятилетиями использовалась практически на каждой машине, имеющей трехфазный двигатель, управляемый контактором, поэтому было неизбежно, что она будет принята на вооружение программистами лестничных цепей и ПЛК. Это также очень безопасная схема: если кнопки *start* и *stop* нажаты одновременно, функция *stop* всегда выигрывает.

Это основной строительный блок большей части лестничного программирования, поэтому, если вы новичок в этом, вам следует убедиться, что вы понимаете, как работает эта схема.

## 8.2 Программирование ClassicLadder

### 8.2.1 Концепции языка релейных схем

ClassicLadder — это тип языка программирования, первоначально реализованный в промышленных ПЛК (он называется языком релейных схем). Он основан на концепции контактов и катушек реле и может использоваться для построения логических проверок и функций способом, знакомым многим системным интеграторам. Лестница состоит из ступенек, которые могут иметь ответвления и напоминают электрическую цепь. Важно знать, как оцениваются лестничные программы при запуске.

Кажется естественным, что каждая строка будет оцениваться слева направо, затем следующая строка вниз и т. д., но в лестничной логике это не работает. Релейная логика *scans* ступеньки 3 раза, чтобы изменить состояние выходов.

- входные данные считываются и обновляются
- выясняется логика
- выходы настроены

Поначалу это может сбить с толку, если выходные данные одной строки считываются входными данными другой цепочки. Будет одно сканирование, прежде чем второй вход станет true после установки выхода.

Еще одна ошибка при программировании лестниц — правило ”Last One Wins”. Если у вас есть один и тот же выход в разных местах вашей лестницы, состояние последнего будет таким, как установлено для выхода.

### 8.2.2 Языки

Наиболее распространенным языком, используемым при работе с ClassicLadder, является *ladder*. ClassicLadder также поддерживает Sequential Function Chart (Grafcet).

### 8.2.3 Компонентов

В ClassicLadder есть два компонента.

- Модуль реального времени classicladder_rt
- Модуль не в реальном времени (включая ГИП) classicladder

#### 8.2.3.1 Файлы

Обычно компоненты ClassicLadder помещаются в файл custom.hal, если вы работаете с конфигурацией созданной StepConf. Их нельзя размещать в файле custom_postgui.hal, иначе меню Ladder Editor будет неактивно.

**Note**
Файлы релейных диаграмм (.clp) не должны содержать пробелов в имени.

#### 8.2.3.2 Модуль реального времени

Загрузка модуля реального времени ClassicLadder (classicladder_rt) возможна из файла HAL или напрямую с помощью инструкции halcmd. Первая строка загружает в реальном времени модуль ClassicLadder. Вторая строка добавляет функцию classicladder.0.refresh в servo thread. Эта строка заставляет ClassicLadder обновляться со скоростью servo thread.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Скорость потока, в котором работает ClassicLadder, напрямую влияет на скорость реагирования на входные и выходные данные. Если вы можете включать и выключать переключатель быстрее, чем ClassicLadder это заметит, возможно, вам придется ускорить поток. Максимальное время, с которым ClassicLadder может обновлять ступени, составляет одну миллисекунду. Вы можете поместить его в более быстрый поток, но он не будет обновляться быстрее. Если вы поместите его в поток медленнее одной миллисекунды, ClassicLadder будет обновлять ступени медленнее. На дисплее секции будет отображаться текущее время сканирования, оно округляется до микросекунд Если время сканирования превышает одну миллисекунду, вы можете сократить лестницу или поместить ее в более медленный поток.

#### 8.2.3.3 Переменные

При загрузке модуля реального времени ClassicLadder можно настроить количество объектов релейной логики каждого типа. Если вы не настроите количество объектов релейной логики, ClassicLadder будет использовать значения по умолчанию.

Table 8.1: Количество переменных по умолчанию

| **Имя объекта** | **Имя переменной** | **Значение по умолчанию** |
|---|---|---|
| Количество ступенек | (numRungs) | 100 |
| Количество бит | (numBits) | 20 |
| Количество word переменных | (numWords) | 20 |
| Количество таймеров | (numTimers) | 10 |
| Количество таймеров IEC | (numTimersIec) | 10 |
| Количество monostables | (numMonostables) | 10 |
| Количество счетчиков | (numCounters) | 10 |
| Количество контактов битовых входов HAL | (numPhysInputs) | 15 |
| Количество контактов выходных битов HAL | (numPhysOutputs) | 15 |
| Количество арифметических выражений | (numArithmExpr) | 50 |
| Количество секций | (numSections) | 10 |
| Количество символов | (numSymbols) | Автоматический |
| Количество входов S32 | (numS32in) | 10 |

Table 8.1: (continued)

| **Имя объекта** | **Имя переменной** | **Значение по умолчанию** |
|---|---|---|
| Количество выходов S32 | (numS32out) | 10 |
| Количество входов с плавающей запятой | (numFloatIn) | 10 |
| Число выходов с плавающей запятой | (numFloatOut) | 10 |

Наибольший интерес представляют объекты numPhysInputs, numPhysOutputs, numS32in и numS32out.

Изменение этих чисел приведет к изменению количества доступных битовых контактов HAL. numPhysInputs и numPhysOutputs управляют количеством доступных битовых контактов HAL (вкл./выкл.). numS32in и numS32out управляют количеством доступных контактов HAL целых чисел со знаком (диапазон целых чисел +-).

Например (вам не нужно все это, чтобы изменить лишь немного):

```
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
```

Чтобы загрузить количество объектов по умолчанию:

```
loadrt classicladder_rt
```

### 8.2.4 Загрузка ClassicLadder модуля не в реальном времени

Команды ClassicLadder HAL должны выполняться до загрузки ГИП, иначе пункт меню Ladder Editor не будет работать. Если вы использовали мастер настройки Stepper, поместите все команды ClassicLadder HAL в файл custom.hal.

Чтобы загрузить модуль не в реальном времени:

```
loadusr classicladder
```

**Note**
Можно загрузить только один файл .clp. Если вам нужно разделить лестницу, используйте секции.

Чтобы загрузить файл релейной логики:

```
loadusr classicladder myladder.clp
```

Параметры загрузки ClassicLadder

- *--nogui* - (загружается без ladder editor), обычно используется после завершения отладки.
- *--modbus_port=port* - (загружает номер порта Modbus)
- *--modmaster* - (инициализирует ведущее устройство MODBUS) должен одновременно загрузить программу релейной логики, иначе TCP является портом по умолчанию.

- *--modslave* - (инициализирует ведомое устройство MODBUS) только TCP

Чтобы использовать ClassicLadder с HAL без EMC:

```
loadusr -w classicladder
```

Параметр -w сообщает HAL не закрывать среду HAL до завершения работы ClassicLadder.

Если вы сначала загрузите лестничную программу с опцией *--nogui*, а затем снова загрузите ClassicLadder без опций, ГИП отобразит последнюю загруженную лестничную программу.

В AXIS вы можете загрузить ГИП из File/Ladder Editor...

## 8.2.5 ClassicLadder ГИП

Если вы загрузите ClassicLadder с графическим интерфейсом, он отобразит два окна: отображение разделов и менеджер разделов.

### 8.2.5.1 Менеджер разделов

При первом запуске ClassicLadder вы получаете пустое окно Менеджера разделов.

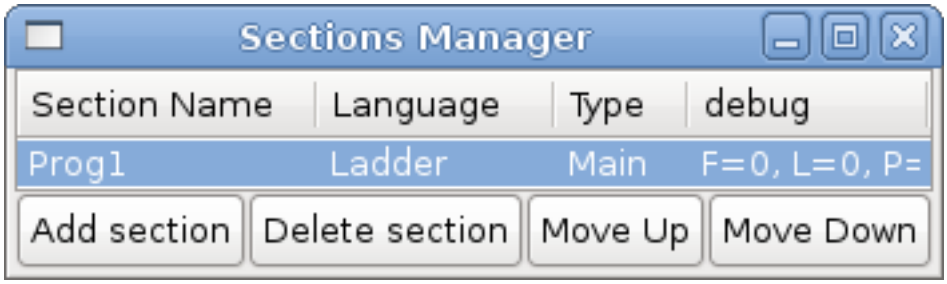



Figure 8.1: Окно Менеджера разделов по умолчанию

Это окно позволяет вам называть, создавать или удалять разделы и выбирать, какой язык будет использовать этот раздел. Так же можно назвать подпрограмму для обработчиков вызовов.

### 8.2.5.2 Отображение раздела

При первом запуске ClassicLadder вы получаете пустое окно отображения разделов. Отображается одна пустая ступень.

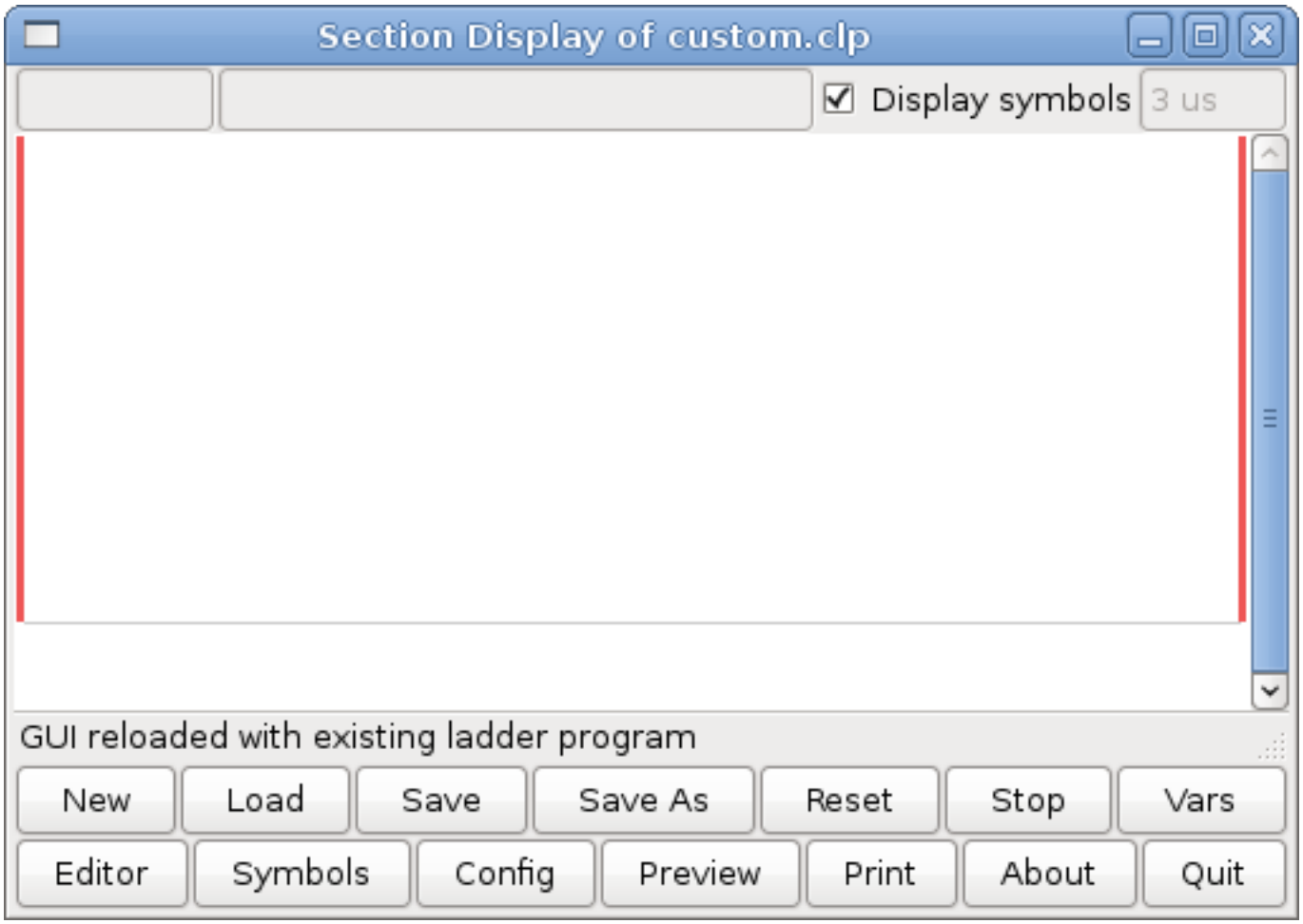



Figure 8.2: Окно отображения раздела по умолчанию

Большинство кнопок говорят сами за себя:

Кнопка Vars предназначена для просмотра переменных, переключайте ее для отображения одного, другого, обоих, а затем ни одного окна.

Кнопка Config используется для Modbus и показывает максимальное количество элементов релейной схемы, загруженных с помощью модуля реального времени.

Кнопка Symbols отобразит редактируемый список символов для переменных (подсказка как вы можете назвать входы, выходы, катушки и т. д.).

Кнопка Quit закроет программу, работающую не в режиме реального времени, т. е. Modbus и дисплей. Программа релейной логики в реальном времени по-прежнему будет работать в фоновом режиме.

Флажок в правом верхнем углу позволяет выбрать, будут ли отображаться имена переменных или имена символов

Вы могли заметить, что под отображением программы релейной логики есть строка с надписью ”Project failed to load...”. Это строка состояния, которая дает вам информацию об элементах релейной схемы, на которые вы нажимаете в окне отображения. В этой строке состояния теперь будут отображаться имена сигналов HAL для переменных %I, %Q и первого %W (в уравнении). На ступенях вы можете увидеть забавные метки, например (103). Это отображается (специально) из-за старой ошибки - при стирании элементов старые версии иногда не удаляли объект с правильным кодом. Вы могли заметить, что длинная горизонтальная кнопка подключения иногда не работала

в старых версиях. Это произошло потому, что он искал *free* код, но нашел что-то еще. Число в скобках — это нераспознанный код. Программа релейной логики по-прежнему будет работать корректно, для исправления сотрите коды редактором и сохраните программу.

#### 8.2.5.3 Окна переменных

Это два окна переменных: Bit Status Window (окно состояния бита) (логическое значение) и Watch Window (окно наблюдения) (целое число со знаком). Кнопка Vars находится в окне Section Display Window. Переключите кнопку Vars, чтобы отобразить одно, другое, оба окна, а затем ни одного окна переменных.

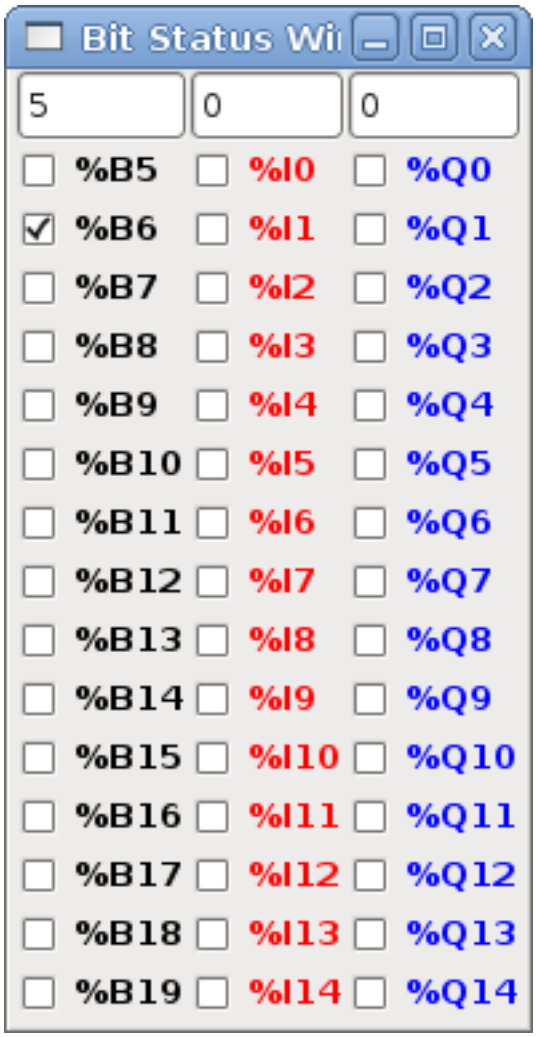



Figure 8.3: Окно статуса бита

Окно состояния бита отображает некоторые логические переменные (вкл/выкл). Обратите внимание, что все переменные начинаются со знака %. Переменные %I представляют контакты входных битов HAL. %Q представляет катушки реле и контакты выходных битов HAL. %B представляет собой внутреннюю катушку реле или внутренний контакт. Три области редактирования вверху позволяют вам выбрать, какие 15 переменных будут отображаться в каждом столбце. Например, если в столбце переменная %B было 15 записей, а вы ввели 5 в верхней части столбца, будут отображены переменные от %B5 до %B19. Флажки позволяют вам устанавливать и отключать переменные %B вручную, если программа релейной логики не устанавливает их в качестве

выходных данных. Любые биты, которые установлены программой как выходные данные во время работы ClassicLadder, не могут быть изменены и будут отображаться как отмеченные, если они включены, и не отмеченные, если выключены.

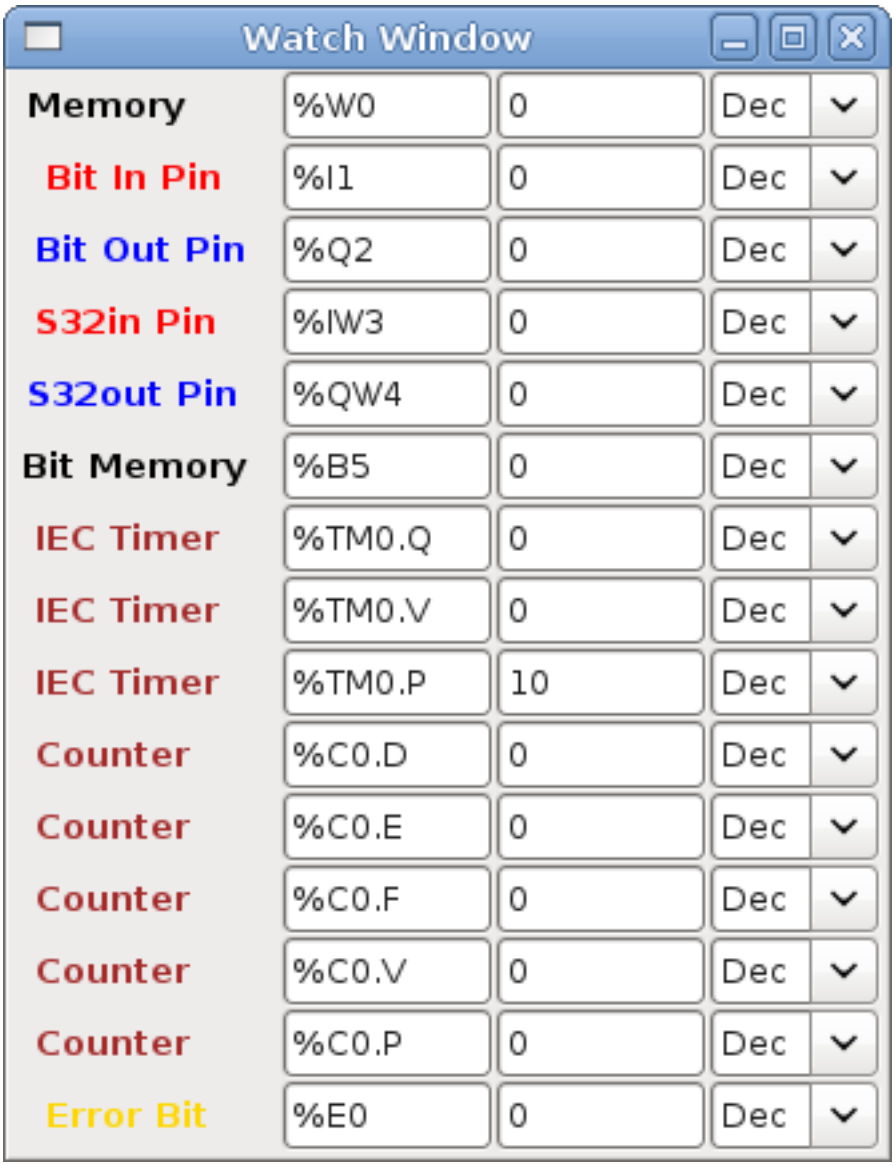



Figure 8.4: Окно просмотра

Окно просмотра (Watch Window) отображает статус переменной. Поле редактирования рядом с ним — это число, хранящееся в переменной, и выпадающий список рядом с ним, который позволяет вам выбрать, будет ли число отображаться в шестнадцатеричном, десятичном или двоичном формате. Если в окне символов определены имена символов для отображения словесных переменных и в окне отображения раздела установлен флажок *display symbols*, имена символов будут отображаться. Чтобы изменить отображаемую переменную, введите номер переменной, например %W2 (если флажок display symbols не установлен) или введите имя символа (если флажок display symbols установлен) поверх существующего номера/имени переменной и нажмите клавишу Enter.

#### 8.2.5.4 Окно символов

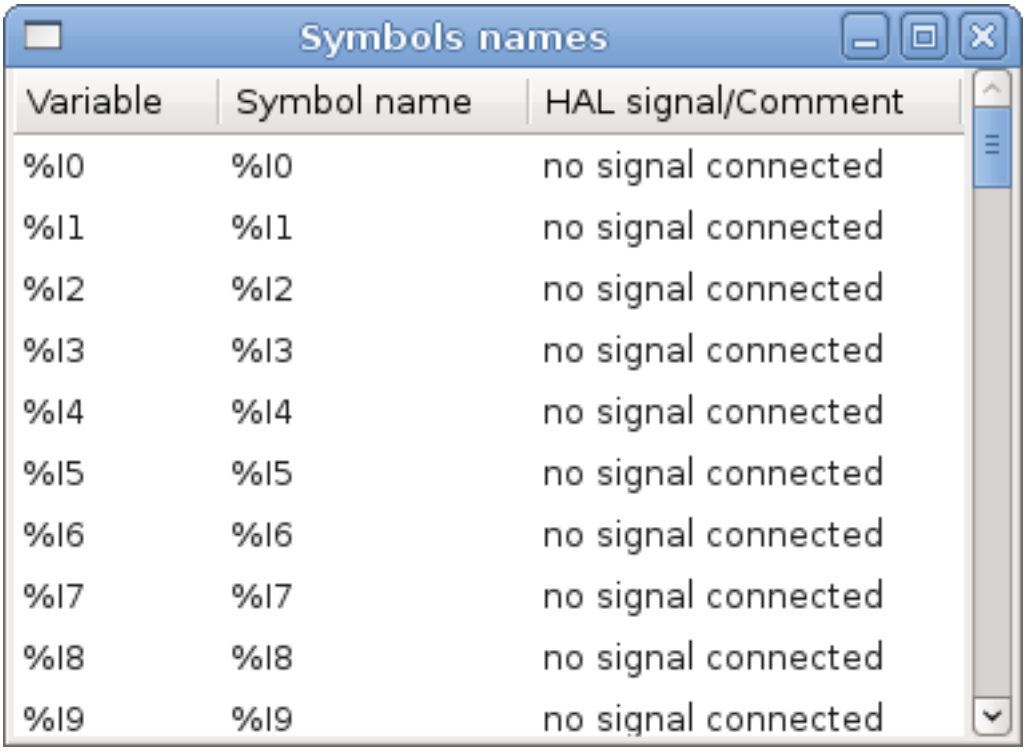



Figure 8.5: Окно символьных имен

Это список имен *символов*, которые можно использовать вместо имен переменных, которые будут отображаться в окне раздела, когда установлен флажок *display symbols*. Вы добавляете имя переменной (помните символ % и заглавные буквы), имя символа. Если к переменной может быть подключен сигнал HAL (%I, %Q и %W — если вы загрузили контакт s32 с модулем реального времени), то в разделе комментариев будет показано текущее имя сигнала HAL или его отсутствие. Имена символов должны быть короткими, чтобы они лучше отображались. Имейте в виду, что вы можете отобразить более длинные имена сигналов HAL для переменных %I, %Q и %W, кликнув на них в окне раздела. Между тем, нужно иметь возможность отслеживать, с чем связана программа релейной логики!

#### 8.2.5.5 Окно редактора

Figure 8.6: Окно редактора

- *Add* - добавляет ступень после выбранной цепочки
- *Insert* - вставляет цепочку перед выбранной цепочкой
- *Delete* - удаляет выбранную цепочку
- *Modify* - открывает выбранную цепочку для редактирования

Начиная с верхнего левого изображения:

- Селектор объекта, Стиратель
- Н.Р. вход, Н.З. вход, вход по фронту, вход по срезу

- Горизонтальное соединение Вертикальное соденинение, Длинное горизонтальное соединение
- Блок таймера IEC, Блок счетчика, Сравнить переменную
- Блок старого таймера, Блок старый моностабильный (эти блоки были заменены Таймером IEC)
- КАТУШКИ - Н.Р. выход, Н.З. выход, установить выход, сбросить выход
- Катушка перехода, катушка вызова, назначение переменной

Краткое описание каждой из кнопок:

- *Селектор* - позволяет выбирать существующие объекты и изменять информацию.
- *Стиратель* - стирает объект.
- *Н.Р. контакт* - создает нормально разомкнутый контакт. Это может быть входной контакт внешнего контакта HAL (%I), контакт внутреннего бита катушки (%B) или контакт внешней катушки (%Q). Входной контакт HAL-контакта замкнут, когда HAL-контакт находится в состоянии true. Контакты катушки замыкаются, когда соответствующая катушка активна (контакт %Q2 замыкается, когда катушка %Q2 активна).
- *Н.З. Контакт* - создает нормально замкнутый контакт. Это то же самое, что и Н.Р. контакт, за исключением того, что контакт разомкнут, когда контакт HAL true или катушка активна.
- *Контакт по фронту* - создает контакт, который замыкается, когда контакт HAL переходит из состояния False в состояние true или когда катушка из неактивного состояния в активное.
- *Контакт по срезу* - создает контакт, который замыкается, когда контакт HAL переходит из состояния true в состояние false или катушка из состояния активно в состояние не активно.
- *Горизонтальное соединение* - создает горизонтальную связь с объектами.
- *Вертикальное соединение* - создает вертикальное соединение с горизонтальными линиями.
- *Горизонтальное рабочее соединение* - создает горизонтальное соединение между двумя объектами и является быстрым способом соединить объекты, находящиеся на расстоянии более одного блока друг от друга.
- *IEC Таймер* - создает таймер и заменяет *Timer*.
- *Timer* - создает модуль таймера (вместо него используется таймер IEC).
- *Monostable* - создает одноразовый моностабильный модуль
- *Счетчик* - создает модуль счетчика.
- *Сравнение* - создает блок сравнения для сравнения переменной со значениями или другими переменными, например. `%W1<=5` или `%W1=%W2`. Сравнение не может быть размещено в самой правой части экрана раздела.
- *Назначение переменной* - создает блок присвоения, позволяющий присваивать значения переменны например. `%W2=7` или `%W1=%W2`. Функции НАЗНАЧЕНИЯ можно разместить только в самой правой части дисплея раздела.

#### 8.2.5.6 Окно конфигурации

Окно конфигурации показывает текущий статус проекта и содержит вкладки настройки Modbus.

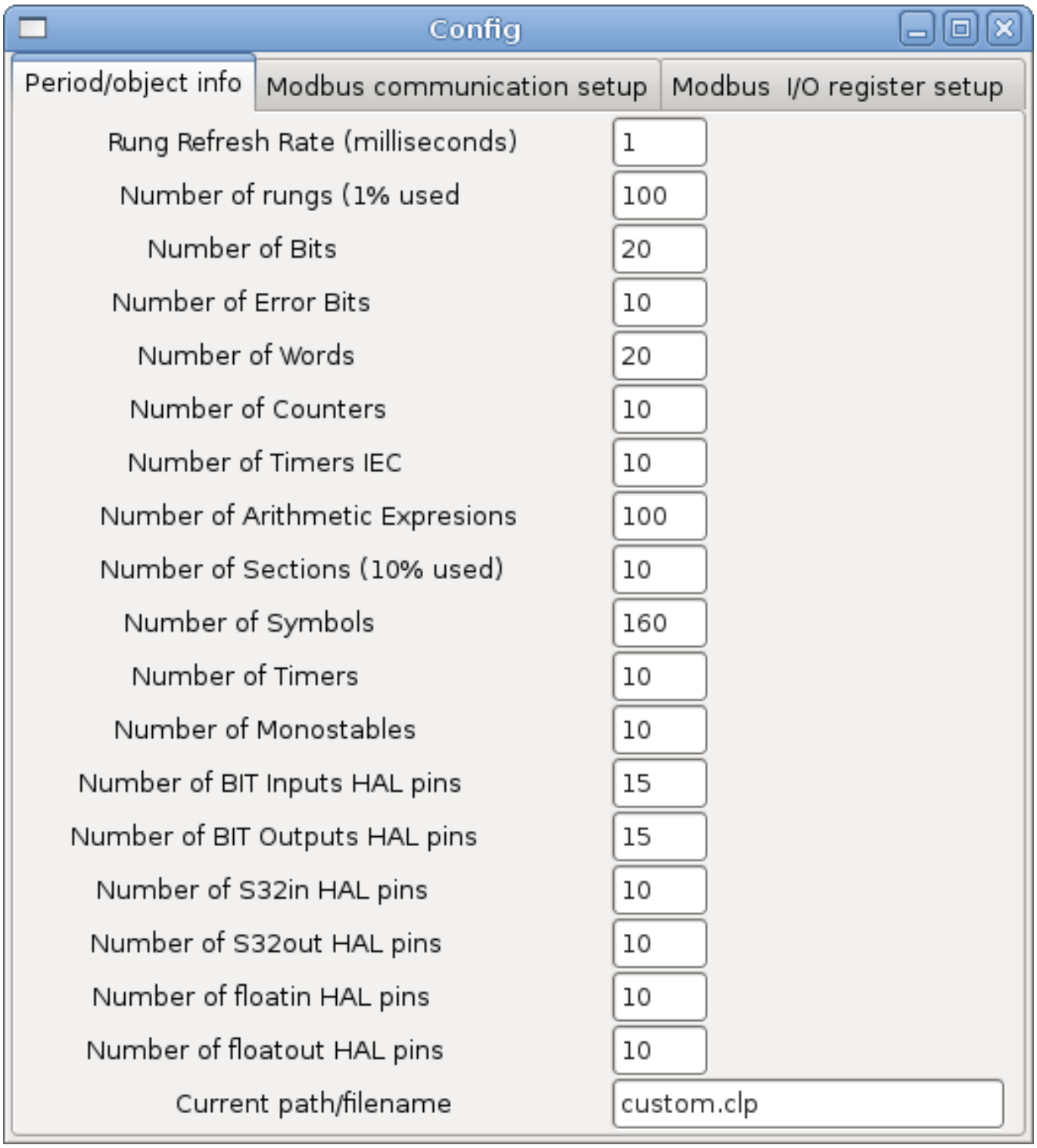



Figure 8.7: Окно конфигурации

## 8.2.6 Объекты релейной логики

### 8.2.6.1 КОНТАКТЫ

Представляют переключатели или контакты реле. Они управляются присвоенной им буквой и номером переменной.

Переменная буква может быть B, I или Q, а число может состоять из трех цифр, например `%I2`, `%Q3` или `%B123`. Переменная I управляется входным контактом HAL с соответствующим номером. Переменная B предназначена для внутренних контактов, управляемых катушкой B с соответствующим номером. Переменная Q управляется катушкой Q с соответствующим номером (как в реле с несколькими контактами). Например, если вывод HAL `classicladder.0.in-00` имеет значение true, то %I0 Н.Р. контакт будет включен (замкнут, true, как бы вы его ни называли). Если катушка %B7 находится под напряжением (включена, true и т. д.), то %B7 Н.Р. контакт будет включен. Если катушка %Q1 находится под напряжением, то %Q1 Н.Р. контакт будет включен (и контакт HAL `classicladder.0.out-01` будет true).

- *Н.Р. Контакт* - (Normally Open) Когда переменная имеет значение false, переключатель выключен.
- *Н.З. Контакт* - (Normally Closed) Когда переменная имеет значение false, переключатель включен.
- *Контакт по фронту* - Когда значение переменной меняется с false на true, переключатель включается ИМПУЛЬСНО.
- *Контакт по срезу* - Когда переменная меняется с true на false, переключатель включается ИМПУЛЬСНО.

### 8.2.6.2 IEC ТАЙМЕРЫ

Представляют новые таймеры обратного отсчета. Таймеры IEC заменяют таймеры и моностабильные устройства.

Таймеры IEC имеют 2 контакта.

- *I* - входной контакт
- *Q* - выходной контакт

Есть три режима - TON, TOF, TP.

- *TON* - Когда входной сигнал таймера true, начинается обратный отсчет, который продолжается до тех пор, пока вход остается true . После завершения обратного отсчета и до тех пор, пока входной сигнал таймера остается true, выходной сигнал будет true.
- *TOF* - Когда входной сигнал таймера равен true, устанавливает выходной сигнал true. Когда входной сигнал false, таймер ведет обратный отсчет, а затем устанавливает выходной сигнал false.
- *TP* - Когда на входе таймера подается импульс true или удерживается true, таймер устанавливает выход в true до тех пор, пока таймер не начнет обратный отсчет. (одно срабатывание)

Временные интервалы можно задавать кратными 100&8239; мс, секундам или минутам.

Существуют также переменные для таймеров IEC, которые можно читать и/или записывать в блоках сравнения или управления.

- *%TMxxx.Q* - таймер завершен (логический, чтение и запись)
- *%TMxxx.P* - предустановка таймера (чтение запись)
- *%TMxxx.V* - значение таймера (чтение и запись)

#### 8.2.6.3 ТАЙМЕРЫ

Представляют таймеры обратного отсчета. Это устарело и заменено таймерами IEC.

Таймеры имеют 4 контакта.

- *E* - разрешить (вход) запускает таймер когда true, сбрасывает когда переходит в false
- *C* - управление (вход) должно быть включено, чтобы таймер запустился (обычно подключается к E)
- *D* - завершено (вывод) true когда время таймера истекает и пока E остается true
- *R* - работает (вывод) true, когда таймер работает

База таймера может быть кратна миллисекундам, секундам или минутам.

Существуют также переменные для таймеров, которые можно читать и/или записывать в блоках сравнения или управления.

- *%Txx.R* - Таймер xx работает (логическое значение, только чтение)
- *%Txx.D* - Таймер xx выполнен (логическое значение, только чтение)
- *%Txx.V* - Текущее значение таймера xx (целое число, только чтение)
- *%Txx.P* - Предустановка таймера xx (целое число, чтение или запись)

#### 8.2.6.4 МОНОСТАБИЛЬНЫЕ

Представляют оригинальные одноразовые таймеры. Сейчас они устарели и заменены таймерами IEC.

Моностабильники имеют 2 контакта I и R.

- *I* - вход (вход) запустит работу моно-таймера.
- *R* - работа (выход) будет true, пока таймер работает.

Контакт I чувствителен к фронту, что означает, что он запускает таймер только при изменении состояния с false на true (или с выкл на вкл). Во время работы таймера контакт I может изменяться, не влияя на работающий таймер. R будет true и останется true до тех пор, пока таймер не завершит отсчет до нуля. База таймера может быть кратна миллисекундам, секундам или минутам.

Существуют также переменные для моностабильных таймеров, которые можно читать и/или записывать в блоках сравнения или управления.

- *%Mxx.R* - Моностабильный xx работает (логическое значение, только чтение)
- *%Mxx.V* - Моностабильный текущее значение xx (целое число, только чтение)
- *%Mxx.P* - Предустановка xx моностабильного таймера (целое число, чтение или запись)

#### 8.2.6.5 СЧЕТЧИКИ

Представляют счетчики прямого/обратного отсчета.

Есть 7 контактов:

- *R* - сброс (вход) сбросит счетчик на 0.
- *P* - предустановка (вход) установит счетчик на число предустановки, назначенное в меню редактирования.
- *U* - прямой отсчет (вход) добавит единицу к отсчету.
- *D* - обратный счет (вход) вычитает единицу из отсчета.
- *E* - начало отсчета (выход) будет истинным, когда счетчик возобновится от 0 до 9999.
- *D* - завершен (выход) будет true, когда отсчет будет равен предустановке.
- *F* - переполнение (выход) будет true когда остчет перезапускается от 9999 к 0.

Контакты прямого и обратного счета чувствительны к фронту, что означает, что они учитываются только тогда, когда контакт меняется с false на true (или с выключенного на включенный, если вы предпочитаете).

Диапазон от 0 до 9999.

Существуют также переменные для счетчиков, которые можно читать и/или записывать в блоках сравнения или управления.

- '%C'*xx*`.D` - Счетчик *xx* завершен (*Двоичный*, *только чтение*)
- '%C'*xx*`.E` - Счетчик *xx* пустой переполнение (*логическое значение*, *только чтение*)
- '%C'*xx*`.F` - Счетчик *xx* полный переполнение (*логическое значение*, *только чтение*)
- '%C'*xx*`.V` - Счетчик *xx* текущее значение (*целое число*, *чтение* или *запись*)
- '%C'*xx*`.P` - Предустановка счетчика *xx* (*целое*, *чтение* или *запись*)

#### 8.2.6.6 СРАВНЕНИЕ

Для арифметического сравнения. Является переменной %XXX = этому числу (или оцененному числу)

Блок сравнения будет истинным, если сравнение истинно. Вы можете использовать большинство математических символов:

- +, -, *, /, = (стандартные математические символы)
- < (меньше чем), > (больше чем), <= (меньше или равно), >= (больше или равно), <> (не равно)
- (, ) пример разделения на группы `%IF1=2,&%IF2<5` в псевдокоде означает, что если %IF1 равен 2, а %IF2 меньше 5, то сравнение true. Обратите внимание на запятую, разделяющую две группы сравнений.
- ^ (экспонента), % (модуль), & (и), | (или),. -
- `ABS` (абсолютное), `MOY` (французское усреднение), `AVG` (усреднение)

Например ABS(%W2)=1, MOY(%W1,%W2)<3.

В уравнении сравнения не допускаются пробелы. Например, `%C0.V>%C0.P` является допустимым выражением сравнения, а `%C0.V > %CO.P` не является допустимым выражением.

Ниже на странице приведен список переменных, которые можно использовать для чтения и записи в объекты релейной логики. При открытии нового блока сравнения обязательно удалите символ # при вводе сравнения.

Чтобы понять, меньше ли в два раза word переменная #1 текущего значения счетчика #0, синтаксис будет:

```
%W1<2*%C0.V
```

Чтобы узнать, равен ли 2-й бит S32in 10-и, синтаксис будет следующим:

```
%IW2=10
```

Примечание. В функции сравнения используются арифметические равенства, а не двойные равенства, к которым привыкли программисты.

#### 8.2.6.7 НАЗНАЧЕНИЕ ПЕРЕМЕННОЙ

Для присвоения переменных, например. присвойте это число (или оцененное число) этой переменной %xxx, есть две математические функции MINI и MAXI, которые проверяют переменную на максимальн (0x80000000) и минимальное значения (0x07FFFFFF) (подумайте о знаковых значениях) и не позволяют им выйти за пределы.

При открытии нового блока назначения переменных обязательно удалите символ # при вводе назначения.

Чтобы присвоить значение 10 предустановке таймера IEC Timer 0, синтаксис будет следующим:

```
%TM0.P=10
```

Чтобы присвоить значение 12 биту 3 s32out, синтаксис будет следующим:

```
%QW3=12
```

**Note**
Когда вы присваиваете значение переменной с помощью блока присвоения переменных, это значение сохраняется до тех пор, пока вы не присвоите новое значение с помощью блока присвоения переменных. Последнее назначенное значение будет восстановлено при запуске LinuxCNC.

На следующем рисунке показан пример назначения и сравнения. %QW0 — это бит S32out, а %IW0 — это бит S32in. В этом случае для контакта HAL `classicladder.0.s32out-00` будет установлено значение 5, а когда вывод HAL `classicladder.0.s32in-00` равен 0, вывод HAL `classiclad` будет установлено значение True.

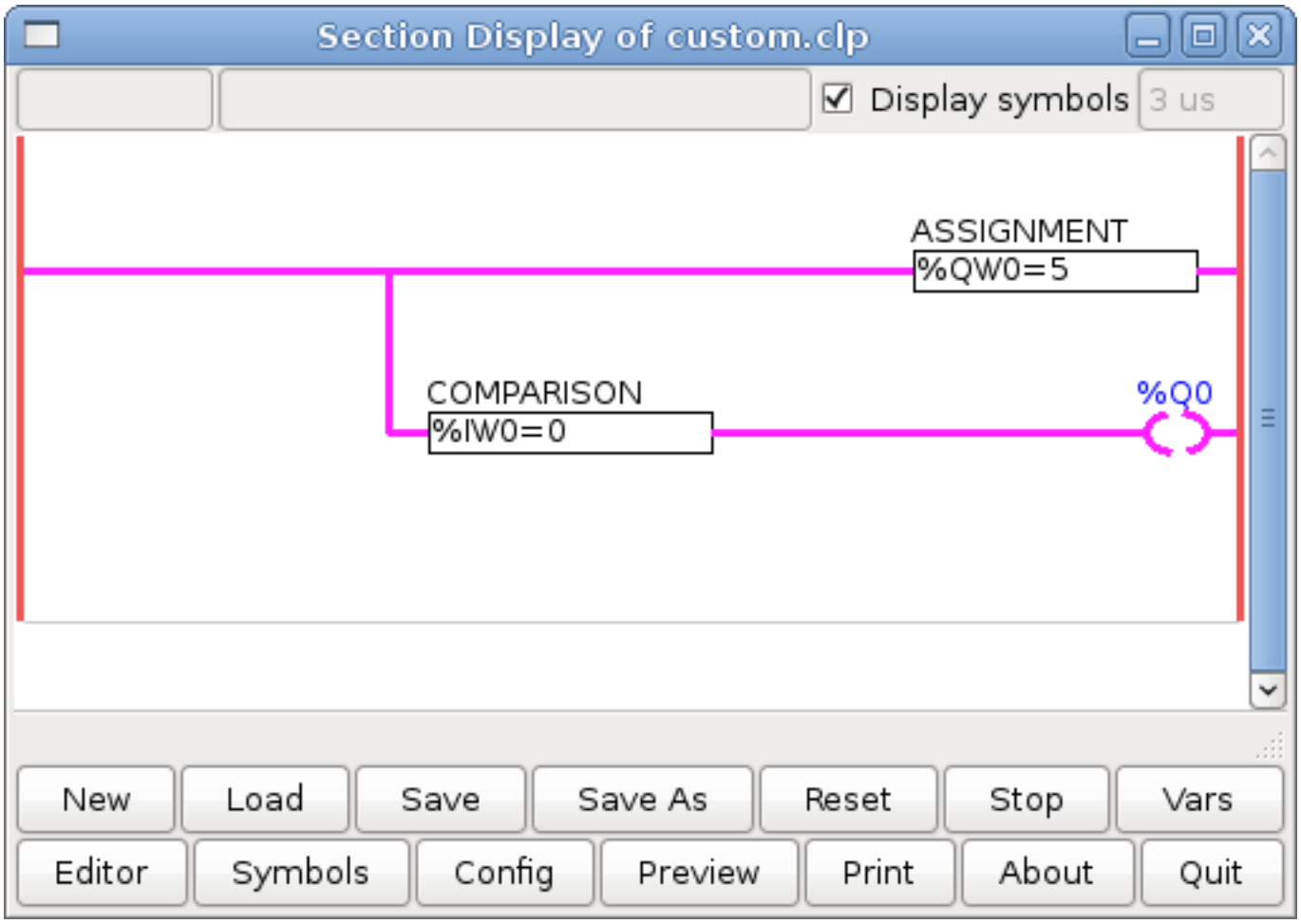



Figure 8.8: Пример релейной логики назначения/сравнения

Figure 8.9: Пример выражения присваивания

Figure 8.10: Пример сравнительного выражения

#### 8.2.6.8 КАТУШКИ

Катушки представляют собой катушки реле. Они управляются присвоенной им буквой и номером переменной.

Буква переменной может быть B или Q, а число может состоять из трех цифр, например, %Q3 или %B123. Катушки Q управляют выходными контактами HAL, например. если %Q15 находится под напряжением, то контакт HAL classicladder.0.out-15 будет true. Катушки B — это внутренние катушки, используемые для управления потоком программы.

- *N.O. COIL* -Катушка реле: когда катушка находится под напряжением, ее контакт, который обычно нормально разомкнутый (сокращенно: H.P.), будет замкнут (включен, true и т. д.), и ток может пройти.
- *N.C. COIL* - Катушка реле которое инвертирует свои контакты: когда на катушку подается напряжение, ее нормально замкнутый контакт (сокращенно: H3) размыкается (выключается, false и т. д.), и ток прерывается.
- *SET COIL* - Катушка реле с фиксирующими контактами: когда катушка находится под напряжением, ее H.P контакт будет зафиксирован замкнутым.
- *RESET COIL* - Катушка реле с фиксирующими контактами: когда катушка находится под напряжени ее H.P. контакт будет зафиксирован разомкнутым.
- *JUMP COIL* - Катушка *goto*: когда катушка находится под напряжением, программа релейной логики переходит на ступень (в разделе CURRENT) - точки перехода обозначаются меткой ступени. (Добавьте метки ступеней в поле меток в верхнем левом углу отображения раздела.)
- *CALL COIL* - Катушка *gosub*: когда на катушку подается напряжение, программа переходит к разделу подпрограммы, обозначенному номером подпрограммы - подпрограммы обозначаются от SR0 до SR9 (назначьте их в диспетчере разделов).

**Warning**
Если вы используете H3 контакт с H3 катушкой, логика будет работать (когда катушка находится под напряжением, контакт будет замкнут), но за этим действительно трудно уследить!

JUMP COIL используется для *JUMP* в другой раздел, как переход в языке программирования BASIC.

Если вы посмотрите в верхний левый угол окна отображения разделов, вы увидите небольшое поле с меткой и более длинное поле для комментариев рядом с ним. Теперь перейдите в Editor→Modify, затем вернитесь в маленькое поле и введите имя.

Продолжайте и добавьте комментарий в разделе комментариев. Это имя метки является именем только этой ступени и используется JUMP COIL для определения того, куда идти.

Размещая JUMP COIL, добавьте ее в крайнее правое положение и измените метку на ступеньку, на которую вы хотите JUMP.

CALL COIL используется для перехода к разделу подпрограммы и последующего возврата, как gosub в языке программирования BASIC.

Если вы перейдете в окно менеджера разделов, нажмите кнопку add section. Вы можете назвать этот раздел, выбрать, какой язык он будет использовать (лестничный или последовательный), и выбрать тип (основной или подпрограмма).

Выберите номер подпрограммы (например, SR0). Появится пустой раздел, и вы сможете создать свою подпрограмму.

Когда вы это сделаете, вернитесь к менеджеру разделов и щелкните свой основной раздел (имя по умолчанию prog1).

Теперь вы можете добавить CALL COIL в свою программу. CALL COIL должны быть размещены в крайнем правом положении ступеньки.

Не забудьте изменить метку на номер подпрограммы, который вы выбрали ранее.

### 8.2.7 Переменные ClassicLadder

Эти переменные используются в COMPARE или OPERATE для получения информации или изменения характеристик объектов релейной логики, например, для изменения предустановки счетчика или проверки завершения работы таймера.

Список переменных:

- `%B`*xxx* - Битовая память *xxx* (Boolean)
- `%W`*xxx* - Память слов (word) *xxx* (32 бита целое число со знаком)
- `%IW`*xxx* - Память слов (word) *xxx* (S32 входной контакт)
- `%QW`*xxx* - Память слов (word) *xxx* (S32 выходной контакт)
- `%IF`*xx* - Память слов(word)*xx* (Вещественный входной контакт) (**преобразован в S32 в ClassicLadder**)
- `%QF`*xx* - Память слов (word) *xx* (Вещественный выходной контакт) (**преобразован в S32 в ClassicLadder**)
- `%T __xx__.R` - Таймер *xx* запущен (Boolean, только для чтения пользователем)
- `%T __xx__.D` - Таймер *xx* завершен (Boolean, только для чтения пользователем)
- `%T __xx__.V` - Таймер_xx_ текущее значение (целое число, только для чтения пользователем)
- `%T __xx__.P` - Таймер_xx_ предустановка (целое число)
- `%TM __xxx__.Q` - Таймер_xxx_ готов(Boolean, чтение запись)
- `%TM __xxx__.P` - Таймер_xxx_ предустановка (целое число, чтение запись)

- %TM `__xxx__.V` - Таймер_xxx_ значение (целое число, чтение запись)
- %M `__xx__.R` - Моностабильный_xx_ запущен (Boolean)
- %M `__xx__.V` - Моностабильный_xx_ текущее значение (целое число, только для чтения пользовател
- %M `__xx__.P` - Моностабильный_xx_ предустановка (целое число)
- %C `__xx__.D` - Счетчик_xx_ готов (Boolean, только для чтения пользователем)
- %C `__xx__.E` - Счетчик_xx_ пустое переполнение(Boolean, только для чтения пользователем)
- %C `__xx__.F` - Счетчик_xx_ полное переполнение (Boolean, только для чтения пользователем)
- %C `__xx__.V` - Счетчик_xx_ текущее значение (целое число)
- %C `__xx__.P` - Счетчик_xx_ предустановка (целое число)
- %I*xxx* - Физический ввод_xxx_ (Boolean) (HAL входной бит)
- %Q*xxx* - Физический вывод_xxx_ (Boolean) (HAL выходной бит)
- %X*xxx* - Активность шага_xxx_ (последовательный язык)
- %X `__xxx__.V` - Время активности в секундах шага *xxx* (последовательный язык)
- %E*xx* - Ошибки (Boolean, чтение запись (будет перезаписан))
- *Indexed or vectored variables* - Это переменные, индексированные другой переменной. Некоторые могут назвать это векторными переменными. Пример: %W0[%W4] => если %W4 равно 23, это соответствует %W23

### 8.2.8 Программирование GRAFCET (конечный автомат)

**Warning**
Вероятно, это наименее используемая и наиболее плохо понятая функция ClassicLadder. Последовательное программирование используется для обеспечения того, чтобы серия событий релейной логики всегда происходила в заданном порядке. Последовательные программы не работают в одиночку. Всегда существует также лестничная программа, которая управляет переменными. Вот основные правила, регулирующие последовательные программы:

- Правило 1: Исходная ситуация. Исходная ситуация характеризуется начальными шагами, которые по определению находятся в активном состоянии в начале операции. Должен быть хотя бы один начальный шаг.
- Правило 2: R2, Очистка перехода. Переход либо разрешен, либо отключен. Говорят, что он включен, когда все непосредственно предшествующие шаги, связанные с соответствующим символом перехода, активны, в противном случае он отключен. Переход не может быть очищен, если он не включен и связанное с ним условие перехода не true.
- Правило 3: R3, Эволюция активных шагов. Очистка перехода одновременно приводит к активному состоянию следующего шага(ов) и к неактивному состоянию непосредственно предшествующего шага(ов).
- Правило 4: R4, Одновременная очистка переходов. Все одновременно очищаемые переходы очищаются одновременно.
- Правило 5: R5, Одновременная активация и деактивация шага. Если во время работы шаг одновременно активируется и деактивируется, приоритет отдается активации.

Это окно редактора ПОСЛЕДОВАТЕЛЬНОСТИ. (Начиная с верхнего левого угла):
Selector arrow, Eraser (Курсор выделения, Стиратель)
Ordinary step, Initial (Starting) step (Обычный шаг, Начальный (стартовый) шаг)
Transition, Step and Transition (Переход, Шаг и переход)
Transition Link-Downside, Transition Link-Upside (Переходная ссылка-нижняя часть, Переходная ссылка-верхняя часть)
Pass-through Link-Downside, Pass-through Link-Upside Jump (Сквозная ссылка–нижняя сторона, Сквозная ссылка-переход верхняя часть)
Link, Comment Box + (Ссылка, Поле для комментариев)

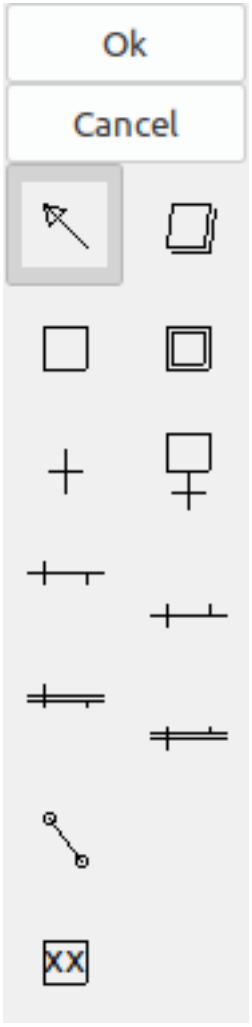



Figure 8.11: Окно редактора последовательности

- *ORDINARY STEP* - имеет уникальный номер для каждого
- *STARTING STEP* - в последовательной программе он должен быть. Здесь программа запустится.
- *TRANSITION* - показывает переменную, которая должна иметь значение true, чтобы управление перешло к следующему шагу.
- *STEP AND TRANSITION* - объединены для удобства
- *TRANSITION LINK-DOWNSIDE* - разделяет логический поток на одну из двух возможных строк в зависимости от того, какой из следующих шагов выполняется первым (Подумайте о логике ИЛИ)

- *TRANSITION LINK=UPSIDE* - объединяет две логические линии (ИЛИ) обратно в одну
- *PASS-THROUGH LINK-DOWNSIDE* - разделяет поток логики на две строки, для продолжения которых ОБЕ должны быть истинными (Подумайте об логике И)
- *PASS-THROUGH LINK-UPSIDE* - объединяет две параллельные логические линии (логика И) вместе
- *JUMP LINK* - соединяет шаги, которые не находятся друг под другом, например, соединение последнего шага с первым
- *COMMENT BOX* - используется для добавления комментариев

Чтобы использовать ссылки, у вас уже должны быть размещены шаги. Выберите тип ссылки, затем выберите два шага или транзакции по одному. Это требует практики!

При последовательном программировании: переменная `%X `__xxx__` (например, `%X5`) используется, чтобы проверить, активен ли шаг. Переменная `%X `__xxx__.V` (например, `%X5.V`) используется, чтобы узнать, как долго шаг был активен. Переменные %X и %X.v используются в лестничной логике. Переменные, назначенные переходам (например, %B), контролируют, перейдет ли логика к следующему шагу. После того, как шаг стал активным, переменная перехода, вызвавшая его активацию, больше не имеет над ним контроля. Последний шаг должен JUMP LINK обратно только к начальному шагу.

### 8.2.9 Modbus

Что следует учитывать:

- Modbus — это программа, работающая не в режиме реального времени, поэтому на сильно загруженном компьютере у нее могут возникнуть проблемы с задержкой.
- Modbus не совсем подходит для событий в реальном времени, таких как управление положением двигателей или управление аварийным остановом.
- Для работы Modbus должен быть запущен ГИП ClassicLadder.
- Modbus не полностью завершен, поэтому он не выполняет все функции Modbus.

Чтобы инициализировать MODBUS, вы должны указать это при загрузке программы ClassicLadder, работающей не в реальном времени.

**Загрузка Modbus**

```
loadusr -w classicladder --modmaster myprogram.clp
```

Параметр `-w` заставляет HAL ждать, пока вы не закроете ClassicLadder, прежде чем закрыть сеанс реального времени. ClassicLadder также загружает подчиненное устройство TCP Modbus, если вы добавите `--modserver` в командной строке.

Функции Modbus

- *1* - чтение катушек
- *2* - чтение входов
- *3* - читать регистры временного хранения
- *4* - прочитать входные регистры
- *5* - писать одиночные катушки

- *6* - записать один регистр
- *8* - тест эхо
- *15* - записать несколько катушек
- *16* - записать несколько регистров

Если вы не укажете *--modmaster* при загрузке программы ClassicLadder, работающей не в реальном времени, эта страница не будет отображаться.

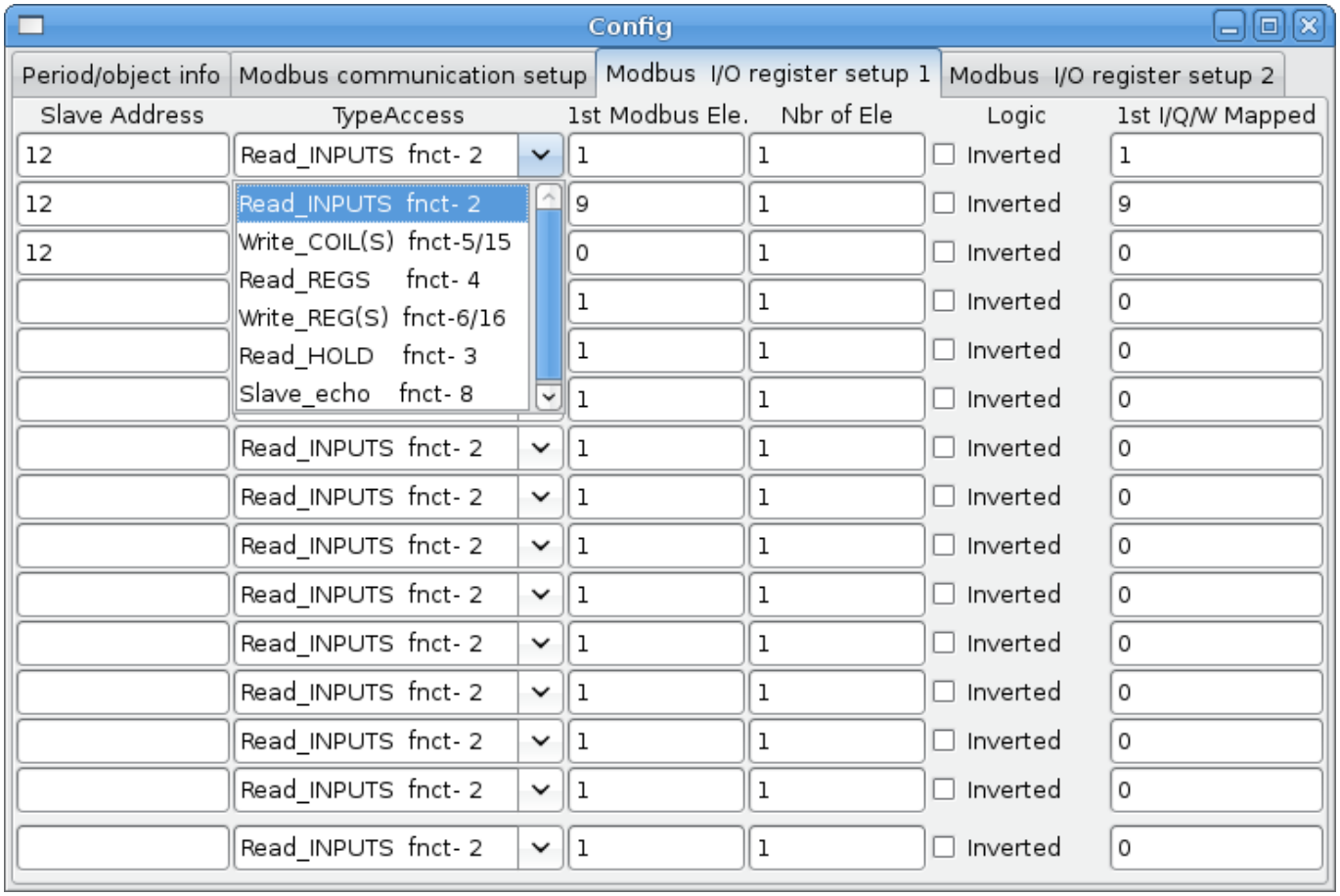



Figure 8.12: Modbus I/O Конфиг

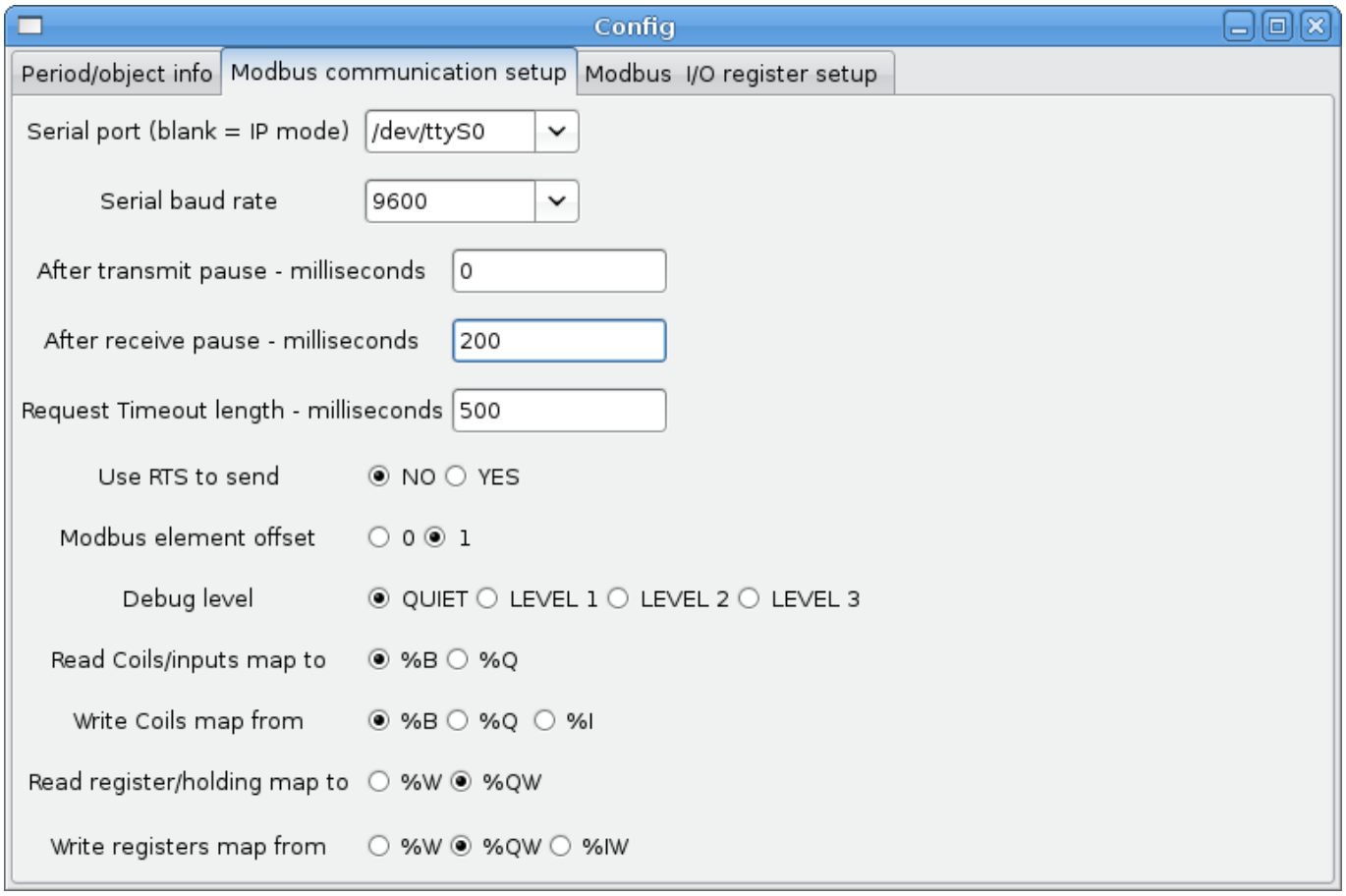



Figure 8.13: Конфигурация связи Modbus

- *SERIAL PORT* - Для IP пусто. Для последовательного порта — расположение/имя последовательного драйвера, например, /dev/ttyS0 (или /dev/ttyUSB0 для преобразователя USB-последовательный порт).
- *SERIAL SPEED* - Должна быть установлена скорость, на которую настроено ведомое устройство — поддерживаются 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- *PAUSE AFTER TRANSMIT* - Пауза (миллисекунды) после передачи и перед получением ответа. Некоторым устройствам требуется больше времени (например, преобразователям USB-последовате порт).
- *PAUSE INTER-FRAME* - Пауза (в миллисекундах) после получения ответа от ведомого устройства. Это задает рабочий цикл запросов (это пауза для КАЖДОГО запроса).
- *REQUEST TIMEOUT LENGTH* - Продолжительность (миллисекунды) времени, прежде чем мы решим, что ведомое устройство не ответило.
- *MODBUS ELEMENT OFFSET* - используется для смещения номеров элементов на 1 (для различий в нумерации производителей).
- *DEBUG LEVEL* - Установите значение 0–3 (0, чтобы прекратить печать отладочной информации, кроме ошибок отсутствия ответа).
- *READ COILS/INPUTS MAP TO* - Выберите, какие переменные, которые читают катушки/входы, будут обновляться. (B или Q).
- *WRITE COILS MAP TO* - Выберите, какие переменные, которые записывают катушки, будут обновляться (B, Q или I).

- *READ REGISTERS/HOLDING* - Выберите, какие переменные, которые будут обновляться в регистрах чтения (W или QW).
- *WRITE REGISTERS MAP TO* - Выберите, из каких переменных будут обновляться регистры чтения (W, QW или IW).
- *SLAVE ADDRESS* - Для последовательного порта идентификационный номер ведомого устройства обычно устанавливается на ведомом устройстве (обычно 1–256). Для IP — IP-адрес ведомого устройства плюс, при необходимости, номер порта.
- *TYPE ACCESS* - Здесь выбирается код функции MODBUS для отправки ведомому устройству (например, тип запроса).
- *COILS / INPUTS* - Входы и катушки (биты) считываются/записываются в переменные I, B или Q (выбирает пользователь).
- *REGISTERS (WORDS)* - Регистры (слова/цифры) сопоставляются с переменными IW, W или QW (выбирает пользователь).
- *1st MODBUS ELEMENT* - Адрес (или номер регистра) первого элемента в группе (не забудьте правильно установить MODBUS ELEMENT OFFSET).
- *NUMBER OF ELEMENTS* - Количество элементов в этой группе.
- *LOGIC* - Здесь можно инвертировать логику.
- *1st%I%Q IQ WQ MAPPED* - Это начальное количество переменных %B, %I, %Q, %W, %IW или %QW, которые отображаются в группу элементов modbus или из нее (начиная с первого номера элемента modbus).

В приведенном выше примере: Номер порта — для моего компьютера /dev/ttyS0 был моим последовате портом.

Последовательная скорость установлена на 9600 бод.

Адрес подчиненного устройства установлен на 12 (на моем VFD я могу установить его от 1 до 31, что означает, что я могу общаться максимум с 31 VFD в одной системе).

Первая строка настроена на 8 входных битов, начиная с первого номера регистра (регистр 1). Таким образом, номера регистров 1–8 сопоставляются с переменными %B ClassicLadder, начиная с %B1 и заканчивая %B8.

Вторая строка настроена на 2 выходных бита, начиная с номера девятого регистра (регистр 9), поэтому номера регистров 9–10 отображаются в переменные ClassicLadder %Q , начиная с %Q9 и заканчивая %Q10.

Третья строка настроена на запись в 2 регистра (по 16 бит каждый), начиная с номера регистра $0^{th}$ (регистр 0), поэтому номера регистров 0-1 отображаются в переменные ClassicLadder %W, начиная с %W0 и заканчивая %W1.

Легко допустить ошибку отклонения на единицу, поскольку иногда ссылки на элементы modbus начинаются с единицы, а не с 0 (на самом деле по стандарту так и должно быть!). Чтобы помочь в этом, вы можете использовать переключатель смещения элемента Modbus.

В документации к вашему ведомому устройству Modbus будет указано, как настраиваются регистры — стандартного способа не существует.

SERIAL PORT, PORT SPEED, PAUSE, и уровень DEBUG доступны для редактирования (значения применяются когда вы закрываете окно конфигурации, хотя Radio-кнопки применяются немедленно).

Чтобы использовать функцию эха, выберите функцию эха и добавьте номер ведомого устройства, которое вы хотите протестировать. Вам не нужно указывать какие-либо переменные.

Число 257 будет отправлено на указанный вами номер подчиненного устройства, и ведомое устройство должно отправить его обратно. Чтобы увидеть сообщение, вам понадобится запустить ClassicLadder в терминале.

## 8.2.10 Настройки MODBUS

Serial:

- ClassicLadder использует протокол RTU (не ASCII).
- 8 бит данных, четность не используется, 1 стоповый бит это так же известно как 8-N-1.
- Скорость передачи данных должна быть одинаковой для ведомого и ведущего устройства. ClassicLadder может иметь только одну скорость передачи данных, поэтому для всех ведомых устройств должна быть установлена одинаковая скорость.
- Пауза между кадрами — это время паузы после получения ответа.
- MODBUS_TIME_AFTER_TRANSMIT — это продолжительность паузы после отправки запроса и до получения ответа (видимо, это помогает с медленными USB-конвертерами).

### 8.2.10.1 Информация о MODBUS

- ClassicLadder может использовать распределенные входы/выходы на модулях, использующих протокол Modbus (”master”: опрос ведомых устройств).
- Ведомые устройства и их ввод/вывод можно настроить в окне конфигурации.
- Доступны 2 эксклюзивных режима: Ethernet с использованием Modbus/TCP и последовательный режим с использованием Modbus/RTU.
- Никакая четность не используется.
- Если имя порта для последовательного порта не задано, будет использоваться режим TCP/IP…
- Адрес ведомого устройства — это адрес ведомого устройства (Modbus/RTU) или IP-адрес.
- IP-адрес может быть дополнен требуемым номером порта (xx.xx.xx.xx:pppp), иначе по умолчанию будет использоваться порт 9502.
- Для испытаний использовались 2 продукта: Modbus/TCP (Adam-6051, https://www.advantech.com) и последовательный Modbus/RTU (https://www.ipac.ws).
- См. примеры: adam-6051 и modbus_rtu_serial.
- Веб-ссылки: https://www.modbus.org и вот этот интересный: https://www.iatips.com/modbus.html
- MODBUS TCP SERVER INCLUDED
- ClassicLadder имеет встроенный сервер Modbus/TCP. Порт по умолчанию — 9502. (Предыдущий стандарт 502 требует, чтобы приложение запускалось с правами root).
- Список поддерживаемых кодов функций Modbus: 1, 2, 3, 4, 5, 6, 15 и 16.
- Таблица соответствия битов и слов Modbus на самом деле не является параметрической и напрямую соответствует переменным %B и %W.

Более подробную информацию о протоколе Modbus можно найти в Интернете.

https://www.modbus.org/

### 8.2.10.2 Ошибки связи

Если возникнет ошибка связи, появится всплывающее окно с предупреждением (если запущен ГИП), и значение %E0 будет true. Modbus продолжит попытки установить связь. %E0 можно использовать для принятия решения на основе ошибки. Таймер можно использовать для остановки станка, если истекло время и т. д.

### 8.2.11 Отладка проблем с Modbus

Хорошая ссылка на протокол: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Если вы запустите linuxcnc/classicladder с терминала, он распечатает команды Modbus и ответы ведомых устройств.

Здесь мы настраиваем ClassicLadder на запрос ведомого устройства 1 для чтения регистров временного хранения (код функции 3), начиная с адреса 8448 (0x2100). Мы запрашиваем возврат 1 элемента данных (шириной 2 байта). Мы сопоставляем его с переменной ClassicLadder, начиная с 2.

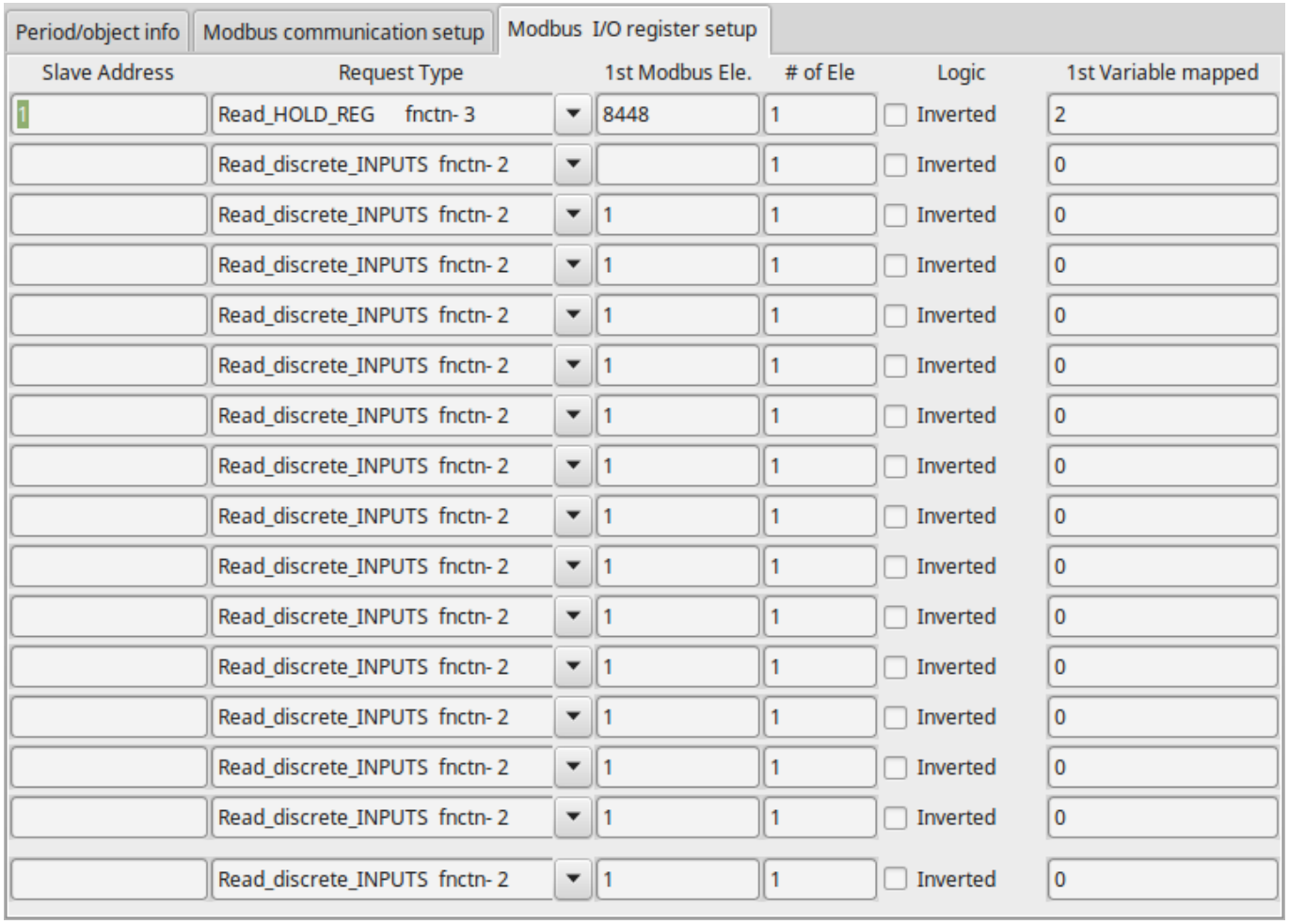



Figure 8.14: Настройка регистра ввода/вывода Modbus

Обратите внимание, что на этом изображении мы установили уровень отладки на 1, чтобы сообщения Modbus выводились на терминал. Мы сопоставили наши регистры хранения для чтения и записи с переменными %W ClassicLadder, поэтому возвращаемые данные будут в %W2, как и в другом изображении, где мы сопоставили данные, начиная со второго элемента.

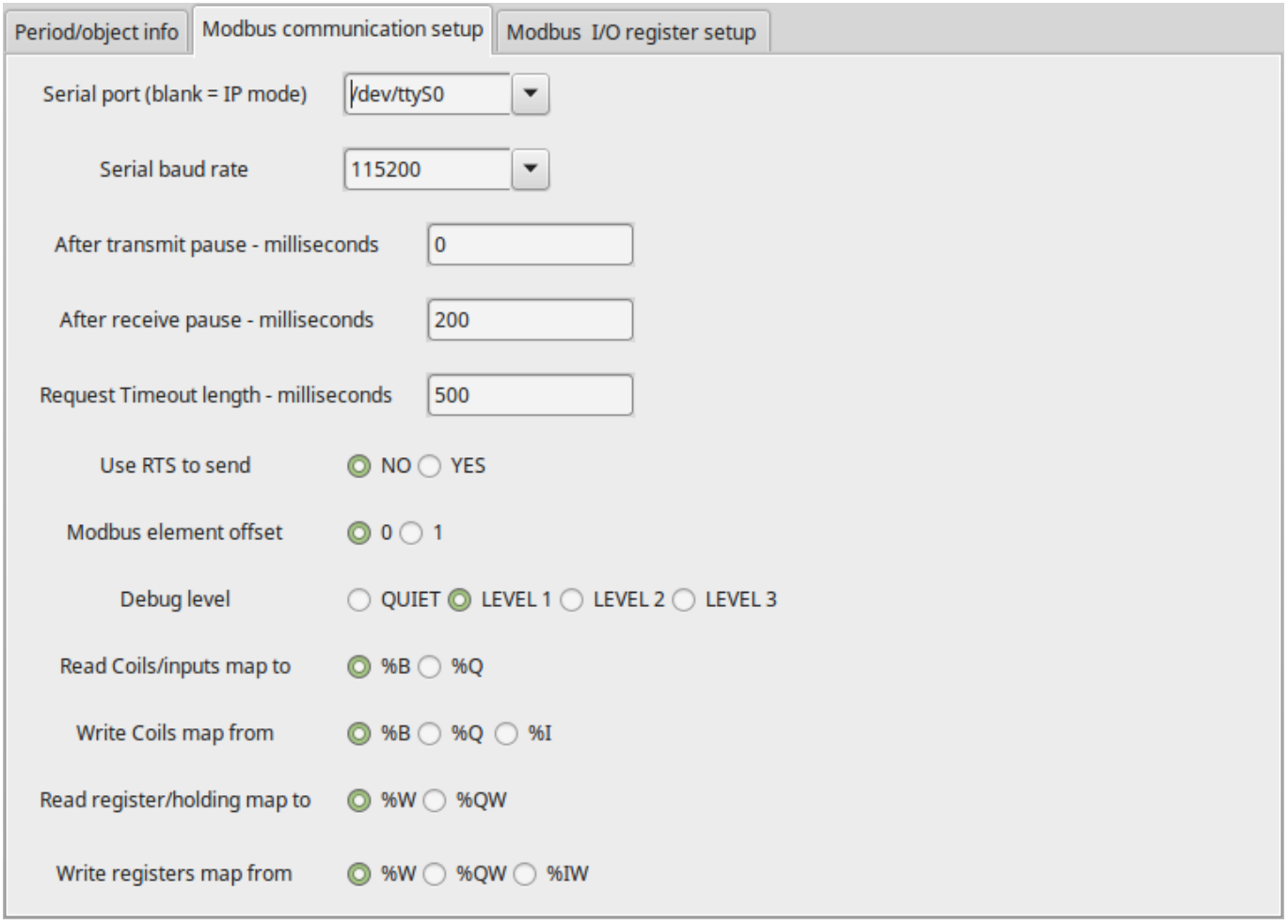



Figure 8.15: Настройка связи Modbus

#### 8.2.11.1 Запрос

Давайте рассмотрим пример чтения одного регистра хранения по адресу 8448 Decimal (0x2100 Hex).

Глядя в ссылку на протокол Modbus:

Table 8.2: Запрос чтения регистра временного хранения

| **Имя** | **количество байтов** | **Значение (hex)** |
|---|---|---|
| Код функции | (1 Byte) | 3 (0x03) |
| Начальный адрес | (2 Bytes) | 0 - 65535 (0x0000 to 0xFFFF) |
| Количество регистров | (2 Bytes) | 1 to 125 (0x7D) |
| Контрольная сумма | (2 bytes) | Рассчитывается автоматически |

Вот пример отправленной команды, напечатанной в терминале (все Hex):

```
INFO CLASSICLADDER-   Modbus I/O module to send: Lgt=8 <-  Slave address-1  Function code-3 ↩
      Data-21 0 0 1 8E 36
```

Значение (Hex):

- Lgt = 8 = сообщение имеет размер 8 байт, включая номер ведомого и число контрольной суммы
- Номер ведомого = 1 (0x1) = Slave address 1
- Function code = 3 (0x3) = прочитать регистр временного хранения
- Начать с адреса = старший байт 33 (0x21) младший байте 0 (0x00) = комбинированный адрес = 8448 (0x2100)
- Число регистров = 1 (0x1) = вернуть 1 2-байтовый регистр (регистры хранения и чтения всегда имеют ширину 2 байта)
- Контрольная сумма = старший байт 0x8E младший байт 0x36 = (0x8E36)

#### 8.2.11.2 Ответ об ошибке

Если есть ответ об ошибке, он отправляет код функции плюс 0x80, код ошибки и контрольную сумму. Получение ответа об ошибке означает, что ведомое устройство видит команду запроса, но не может предоставить действительные данные. Глядя в ссылку на протокол Modbus:

Table 8.3: Возвращена ошибка для кода функции 3 (чтение регистра временного хранения)

| **Имя** | **Количество байтов** | **Значение (hex)** |
|---|---|---|
| Код ошибки | 1 Byte | 131 (0x83) |
| Код исключения | 1 Byte | 1-4 (0x01 to 0x04) |
| Контрольная сумма | (2 bytes) | Рассчитывается автоматически |

Значение кода исключения:

- 1 - недопустимая функция
- 2 - недопустимый адрес данных
- 3 - Недопустимое значение данных
- 4 - сбой ведомого устройства

Вот пример полученной команды, напечатанной в терминале (все Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=5 ->   (Slave address-1  Function  ↩
    code-83 ) 2 C0 F1
```

Значение (Hex):

- Номер ведомого = 1 (0x1) = Slave address 1
- Код функции = 131 (0x83) = ошибка при чтении регистра временного хранения
- Код ошибки = 2 (0x2) = запрошен недопустимый адрес данных
- Контрольная сумма = (0x8E36)

#### 8.2.11.3 Ответ данных

Просмотрите ссылку на протокол Modbus для ответа:

Table 8.4: Ответ данных для кода функции 3 (чтение регистра временного хранения)

| **Имя** | **количество байтов** | **Значение (hex)** |
|---|---|---|
| Код функции | 1 Byte | 3 (0x03) |
| Число байтов | 1 Byte | 2 x N* |
| Значение регистра | N* x 2 Bytes | возвращаемое значение запрошенного адреса |
| Контрольная сумма | (2 bytes) | рассчитывается автоматически |

*N = Количество регистров

Вот пример полученной команды, напечатанной в терминале (все Hex):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=7 ->   (Slave address-1  Function  ↩
    code-3 2 0 0 B8 44)
```

значение (Hex):

- Номер ведомого = 1 (0x1) = Slave address 1
- Запрошенный код функции = 3 (0x3) = запрос на чтение регистра временного хранения
- количество байтовых регистров = 2 (0x1) = вернуть 2 байта (каждое значение регистра имеет ширину 2 байта)
- значение старшего байта = 0 (0x0) = значение старшего байта адреса 8448 (0x2100)
- значение младшего байта = 0 (0x0) = значение старшего байта адреса 8448 (0x2100)
- Контрольная сумма = (0xB844)

(старшие и младшие байты объединяются для создания 16-битного значения, а затем передаются в переменную ClassicLadder.) Регистры чтения могут быть сопоставлены с %W или %QW (внутренняя память или выходные контакты HAL). Регистры записи могут быть сопоставлены с %W, %QW или %IW (внутренняя память, выходные контакты HAL или входные контакты HAL). Номер переменной будет начинаться с номера, введенного в столбце страницы настройки реестра ввода-вывода Modbus: *First variable mapped*. Если за одно чтение/запись запрашивается несколько регистров, то номера переменных идут последовательно после первого.

#### 8.2.11.4 Ошибки MODBUS

- В блоках сравнения функция %W=ABS(%W1-%W2) принимается, но не вычисляется должным образом. в настоящее время разрешен только %W0=ABS(%W1).
- При загрузке программы релейной логики она загружает информацию Modbus, но не говорит ClassicLadder инициализировать Modbus. Вы должны инициализировать Modbus при первой загрузке ГИП, добавив --modmaster.
- Если менеджер разделов расположен в верхней части экрана раздела, поперек полосы прокрутки и при нажатии кнопки exit, программа, работающая не в режиме реального времени, аварийно завершает работу.
- При использовании *--modmaster* вы должны одновременно загрузить программу релейной логики, иначе будет работать только TCP.
- чтение/запись нескольких регистров в Modbus приводит к ошибкам контрольной суммы.

### 8.2.12 Настройка ClassicLadder

В этом разделе мы рассмотрим шаги, необходимые для добавления ClassicLadder в конфигурацию, созданную мастером StepConf. На странице дополнительных параметров конфигурации мастера StepConf установите флажок ”Include ClassicLadder PLC”.

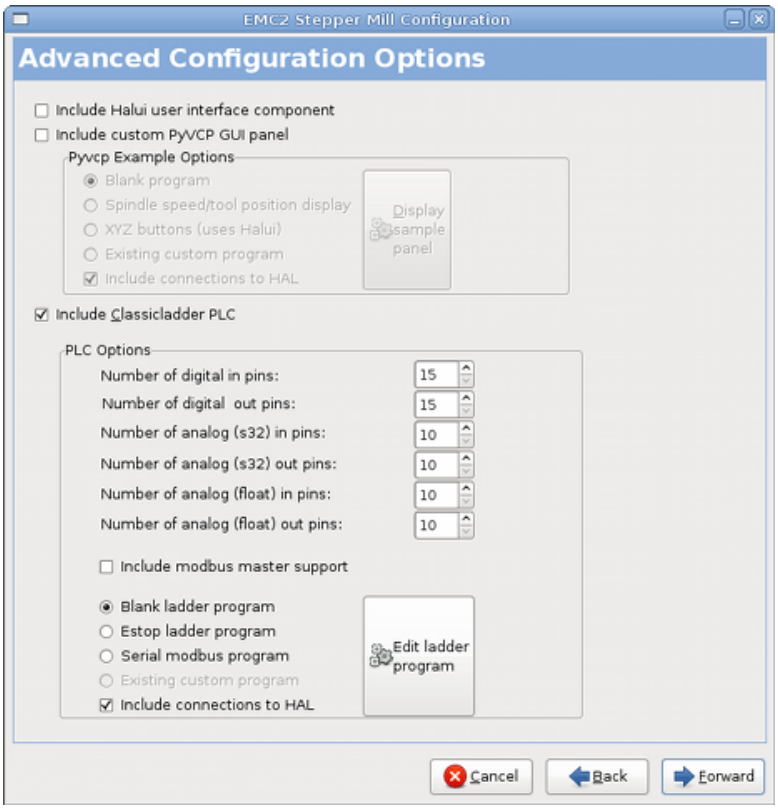



Figure 8.16: StepConf ClassicLadder

#### 8.2.12.1 Добавление модулей

Если вы использовали мастер StepConf для добавления ClassicLadder, вы можете пропустить этот шаг.

Чтобы добавить ClassicLadder вручную, вам необходимо сначала добавить модули. Это делается добавлением пары строк в файл custom.hal.

Эта строка загружает модуль реального времени:

```
loadrt classicladder_rt
```

Эта строка добавляет функцию ClassicLadder в поток сервопривода:

```
addf classicladder.0.refresh servo-thread
```

#### 8.2.12.2 Добавление лестничной логики

Теперь запустите свою конфигурацию и выберите ”File/Ladder Editor”, чтобы открыть графический интерфейс ClassicLadder. Вы должны увидеть пустое окно Section Display и Sections Manager, как показано выше. В окне Section Display откройте Редактор. В окне редактора выберите Modify. Теперь появляется окно Properties , и в Section Display отображается сетка. Сетка — это одна ступенька лестницы. Ступенька может содержать ветви. Простая ступенька имеет один вход, соединительную линию и один выход. Ступенька может иметь до шести горизонтальных ветвей. Несмотря на то, что в одной ступеньке можно использовать более одной трассы, результаты непредсказуемы.

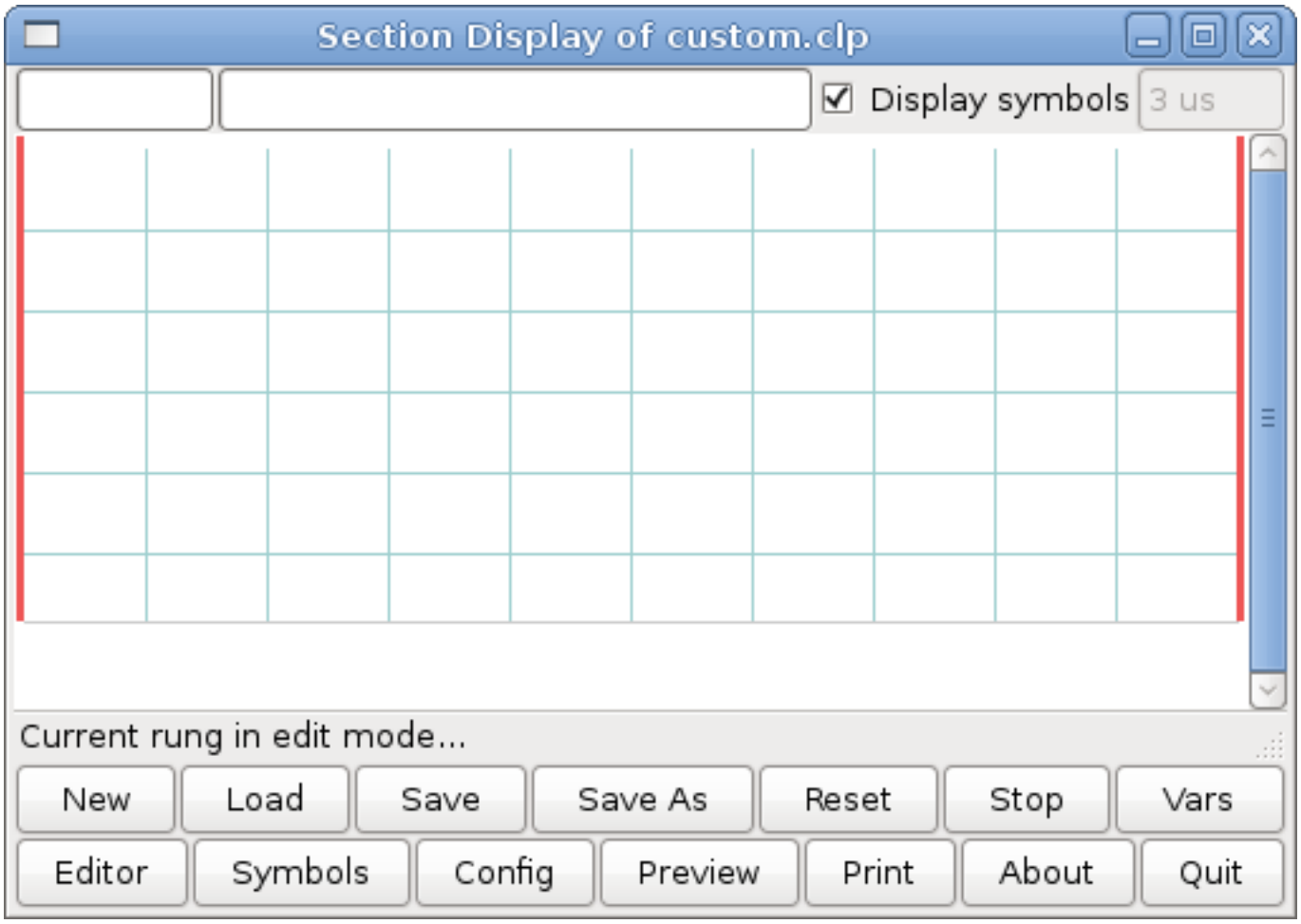



Figure 8.17: Section Display с сеткой

Теперь нажмите на H.P. вход в окне редактора.

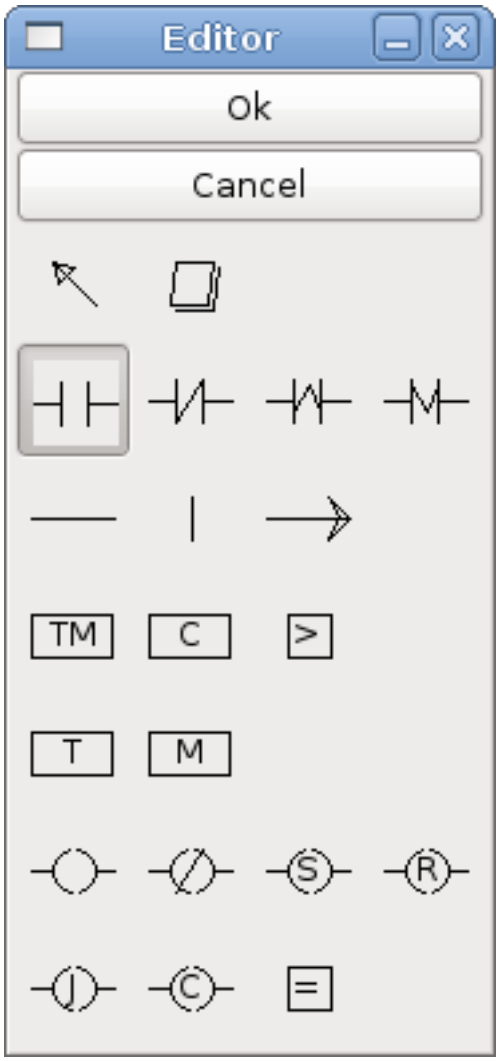



Figure 8.18: Окно редактора

Теперь щелкните в верхней левой сетке, чтобы разместить H.P. Войдите в лестницу.

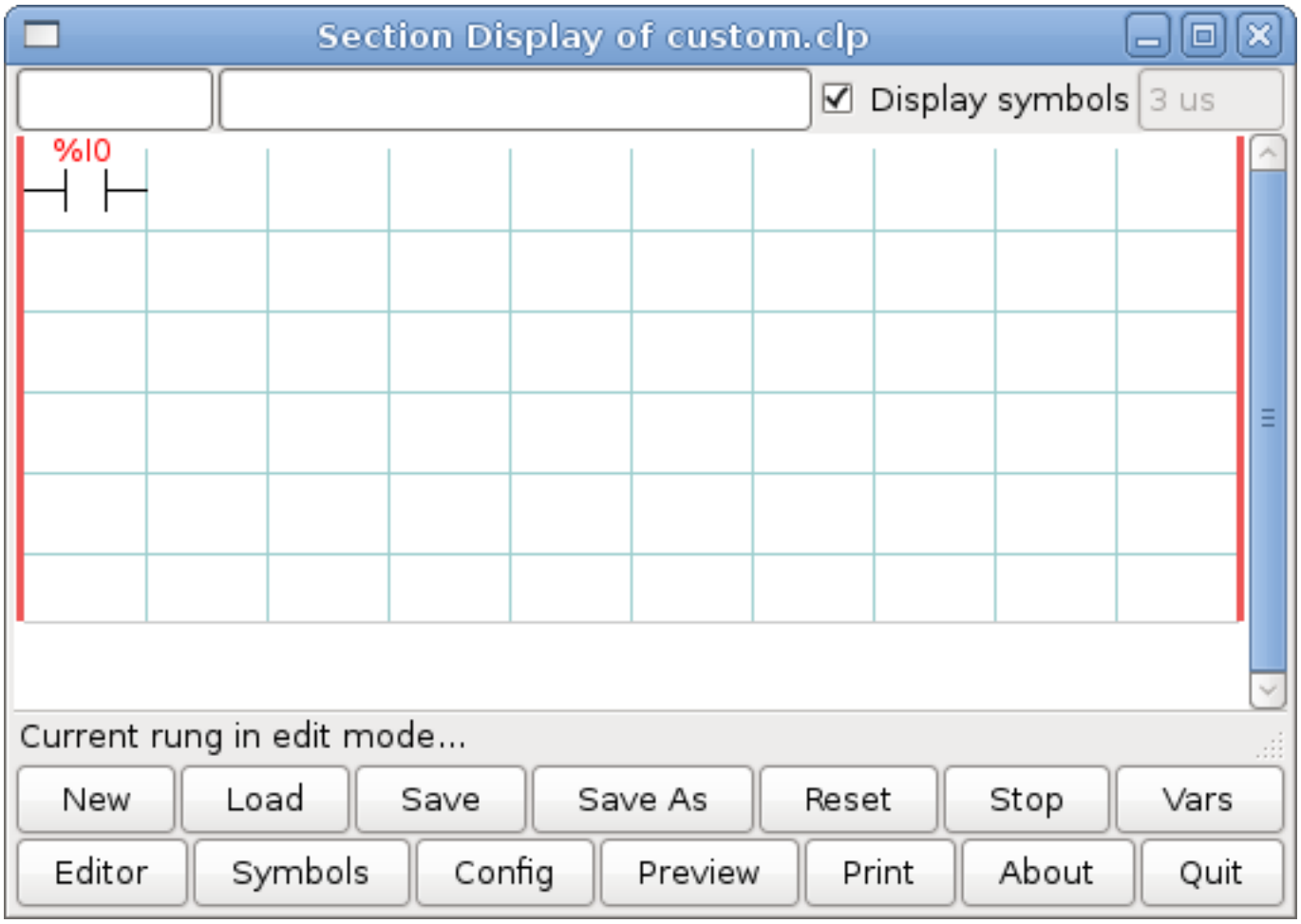



Figure 8.19: Section Display с вводом

Повторите вышеуказанные шаги, чтобы добавить H.P. выход в верхнюю правую сетку и используйте горизонтальное соединение, чтобы соединить их. Это должно выглядеть следующим образом. Если нет, используйте Eraser, чтобы удалить ненужные участки.

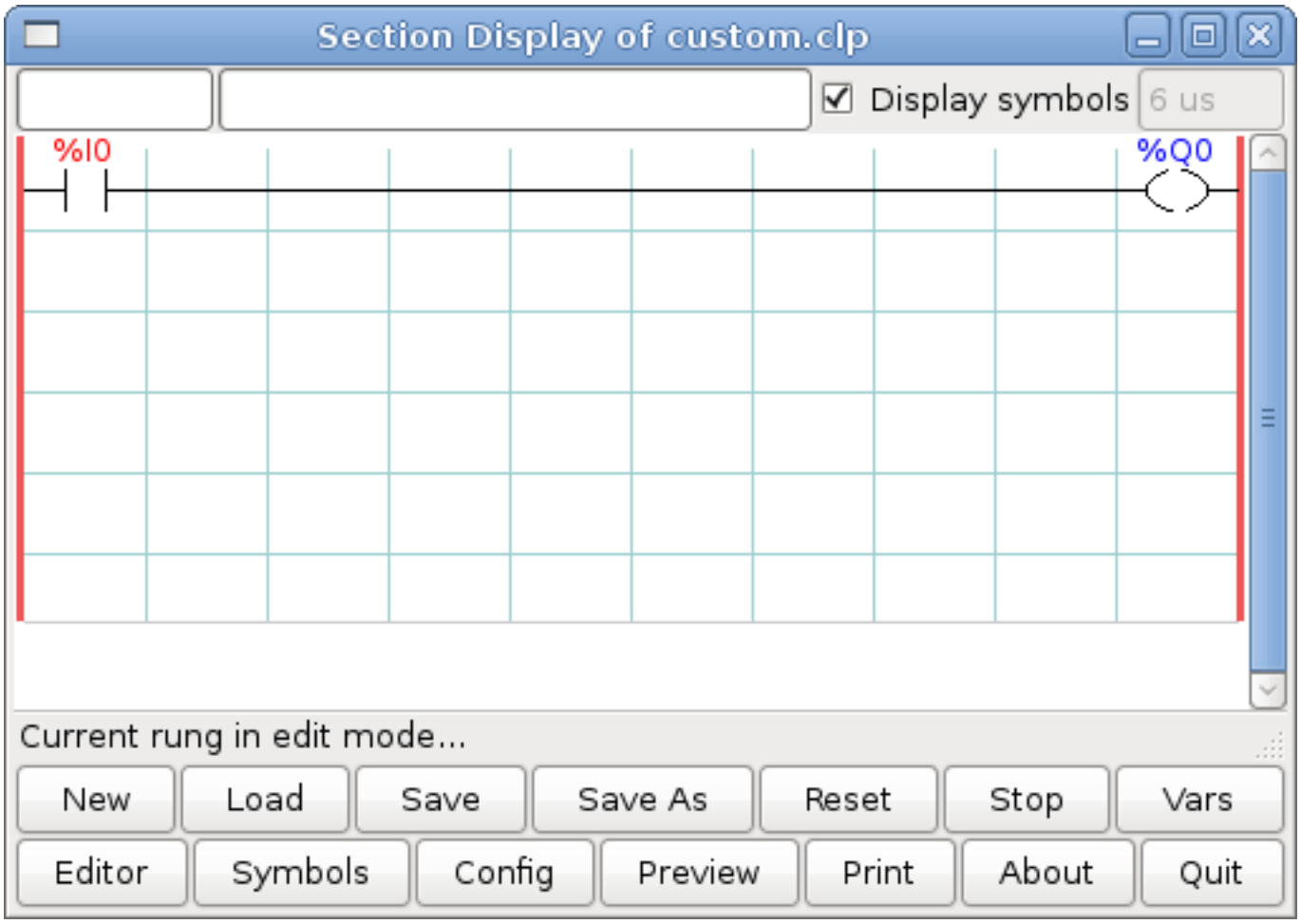



Figure 8.20: Section Display со ступенькой

Теперь нажмите кнопку OK в окне Editor. Теперь ваш Section Display должен выглядеть так:

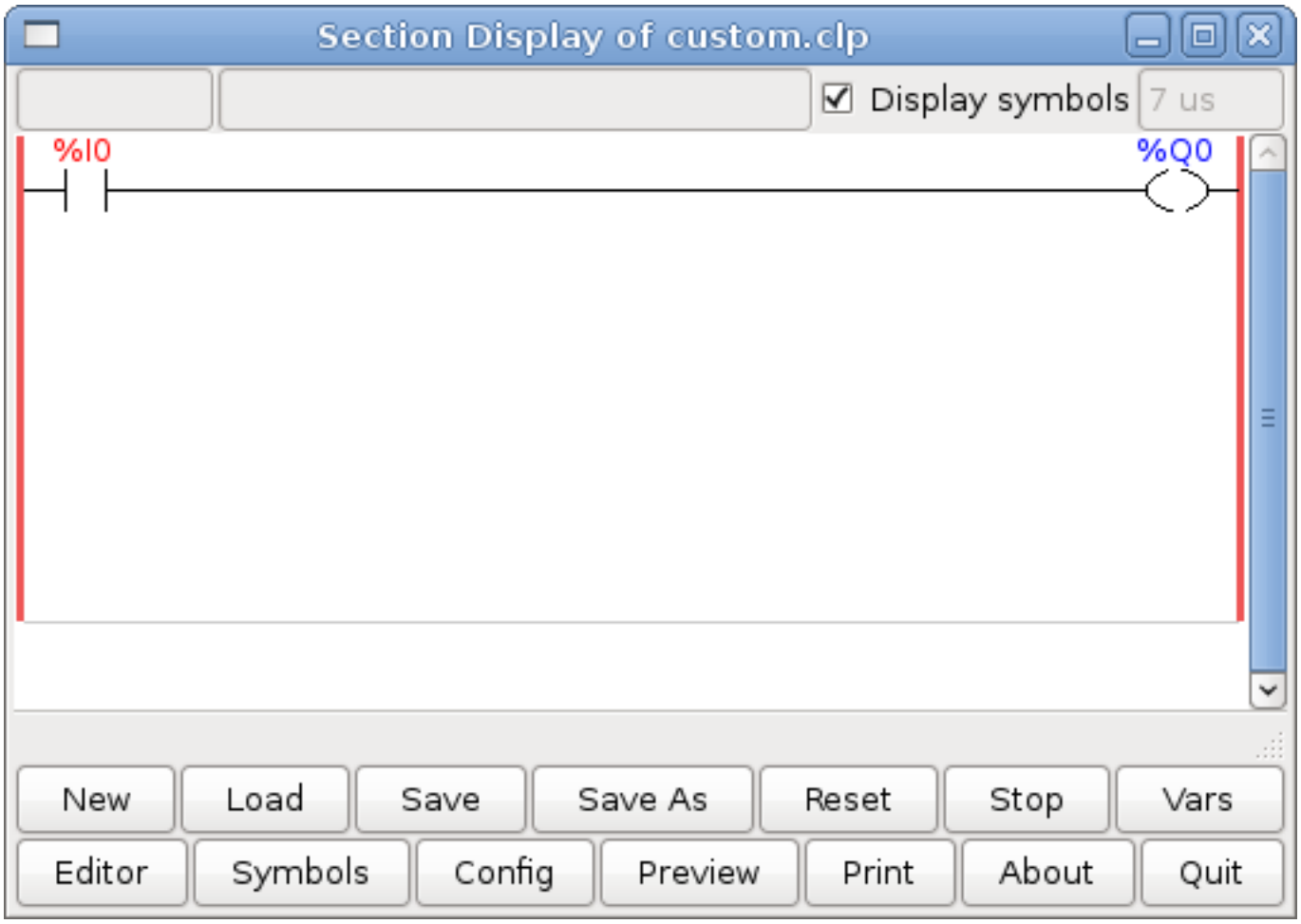



Figure 8.21: Section Display завершен

Чтобы сохранить новый файл, выберите *Save As* и дайте ему имя. Расширение .clp будет добавлено автоматически. По умолчанию в качестве места для его сохранения должен использоваться текущий каталог конфигурации.

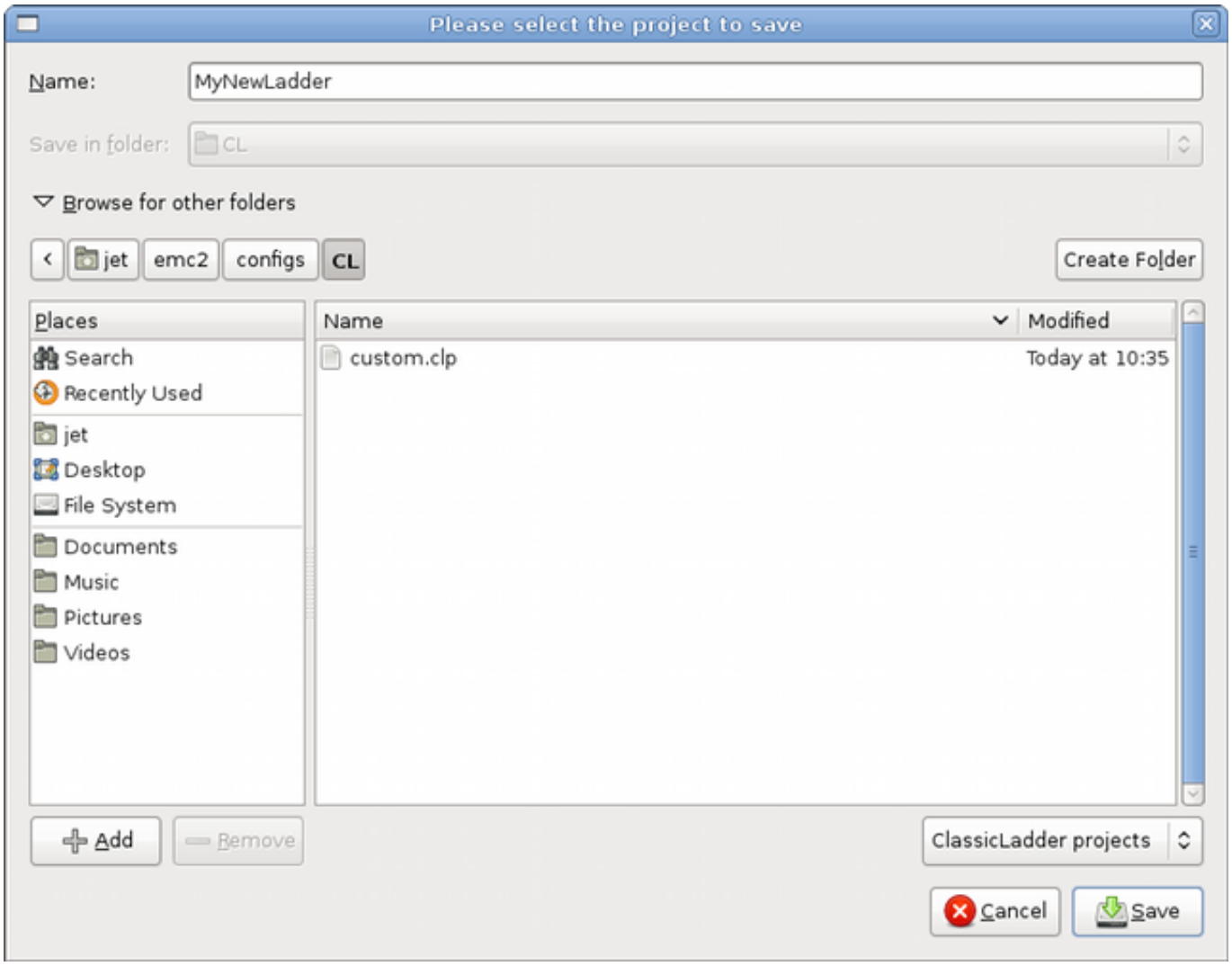



Figure 8.22: Диалог Save As

Опять же, если вы использовали мастер StepConf для добавления ClassicLadder, вы можете пропустить этот шаг.

Чтобы вручную добавить лестницу, вам нужно добавить в файл custom.hal строку, которая будет загружать ваш файл лестницы. Закройте сеанс LinuxCNC и добавьте эту строку в файл custom.hal.

```
loadusr -w classicladder --nogui MyLadder.clp
```

Теперь, если вы запустите конфигурацию LinuxCNC, ваша лестничная программа также будет работать. Если вы выберете ”File/Ladder Editor”, созданная вами программа появится в окне Section Display.

## 8.3 ClassicLadder примеры

### 8.3.1 Счетчик с предустановкой

Чтобы иметь счетчик, который *зацикливается*, вам необходимо использовать вывод предустановки и вывод сброса. Когда вы создаете счетчик, установите предустановку на число, которого вы хотите достичь, прежде чем обнулить ее до 0. Логика заключается в том, что если значение счетчика превышает предустановленное значение, то счетчик сбрасывается, и если антипереполнени счетчик устанавливается на предустановленное значение. Как вы можете видеть в примере, когда значение счетчика больше, чем заданное значение счетчика, срабатывает сброс счетчика, и значение теперь равно 0. Выход нижнего уровня %Q2 установит значение счетчика на заданное значение при обратном счете.

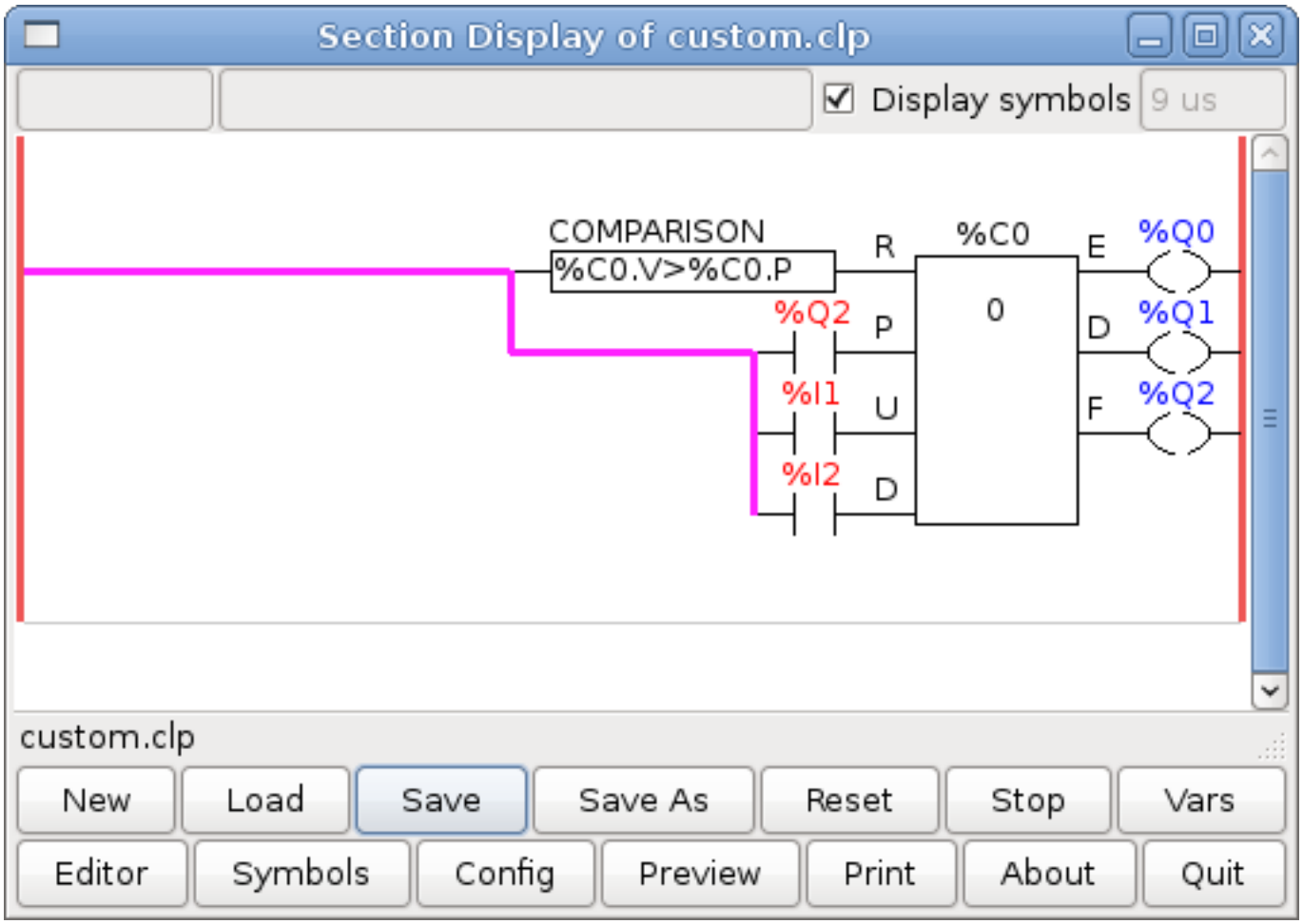



Figure 8.23: Счетчик с предустановкой

### 8.3.2 Отклонить дополнительные импульсы

В этом примере показано, как отклонить лишние импульсы на входе. Предположим, что входной импульс %I0 имеет раздражающую привычку давать дополнительный импульс, который портит нашу логику. TOF (задержка выключения таймера) предотвращает попадание дополнительного импульса на наш очищенный выход %Q0. Это работает так: когда таймер получает входной сигнал, выход таймера включается на время, установленное в настройках времени. Используя

нормально закрытый контакт %TM0.Q, выход таймера блокирует любые дальнейшие входы от достижения нашего выхода до тех пор, пока не истечет время ожидания.

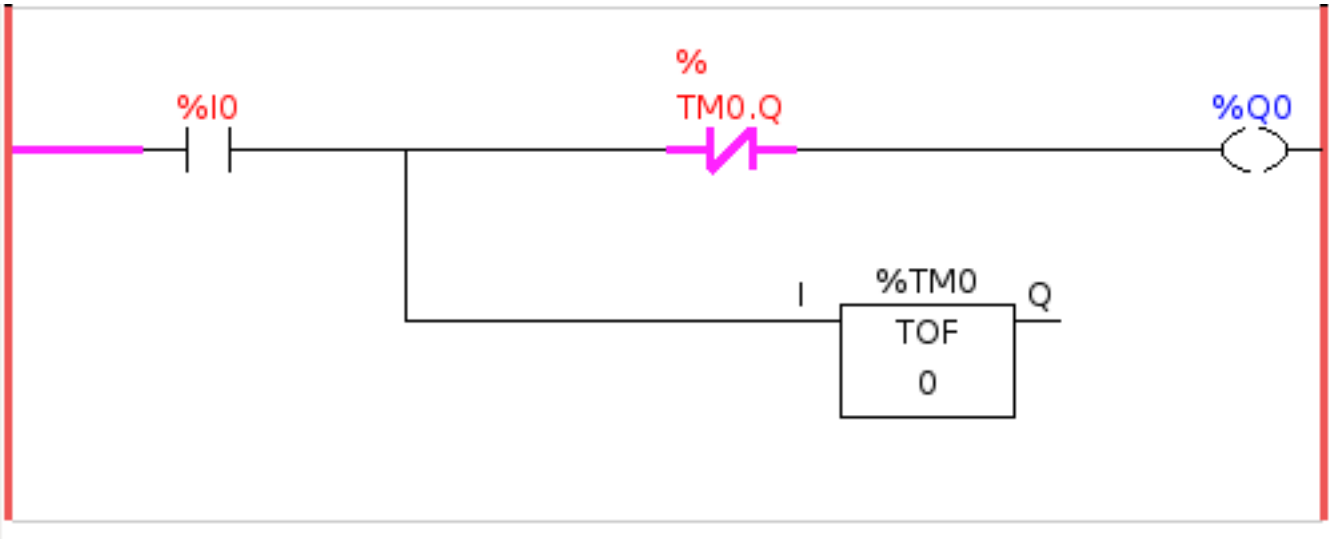



Figure 8.24: Отклонить дополнительный импульс

### 8.3.3 Внешний аварийный останов

Пример внешнего аварийного останова находится в папке /config/classicladder/cl-estop. Он использует панель PyVCP для моделирования внешних компонентов.

Для подключения внешнего аварийного останова к LinuxCNC и совместной работы внешнего аварийного останова с внутренним аварийным остановом требуется несколько подключений через ClassicLadder.

Сначала нам нужно открыть цикл аварийного останова в основном файле HAL, закомментировав его, добавив знак решетки, как показано, или удалив следующие строки.

```
# net estop-out <= iocontrol.0.user-enable-out
# net estop-out => iocontrol.0.emc-enable-in
```

Затем мы добавляем ClassicLadder в наш файл custom.hal, добавляя эти две строки:

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Далее мы запускаем нашу конфигурацию и строим лестницу, как показано здесь.

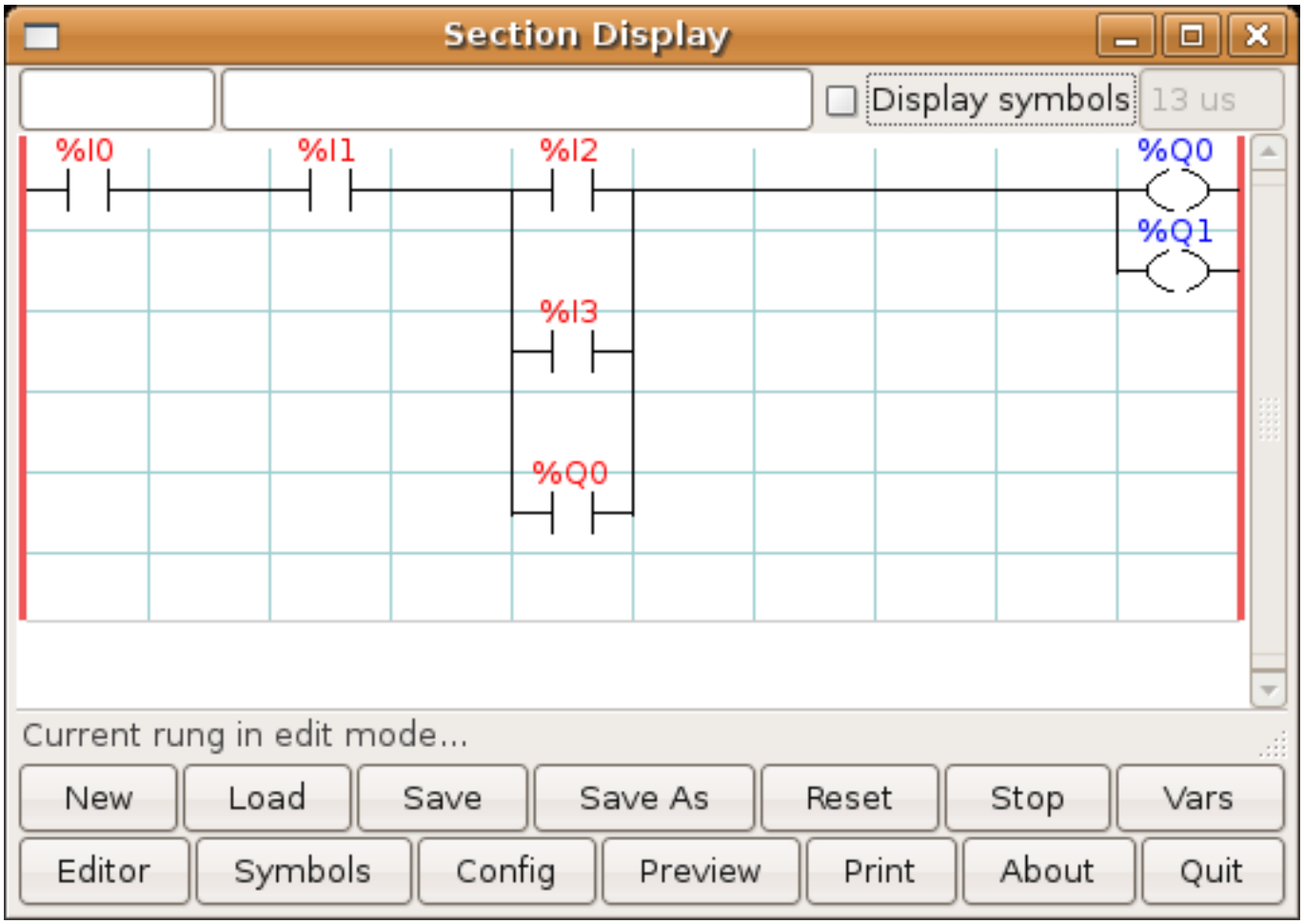



Figure 8.25: E-Stop Section Display

После построения лестницы выберите Save As и сохраните лестницу как estop.clp.

Теперь добавьте следующую строку в файл custom.hal.

```
# Load the ladder
loadusr classicladder --nogui estop.clp
```

I/O назначения

- %I0 = Ввод с панели PyVCP, имитирующий аварийный останов (флажок)
- %I1 = Ввод от аварийного останова LinuxCNC
- %I2 = Ввод импульса сброса аварийного останова LinuxCNC
- %I3 = Ввод от кнопки сброса панели PyVCP
- %Q0 = Вывод в LinuxCNC для разрешения
- %Q1 = Выход на контакт включения внешней платы драйвера (используйте выход H.3., если на вашей плате есть контакт запрета)

Затем мы добавляем следующие строки в файл custom_postgui.hal.

```
# E-Stop example using PyVCP buttons to simulate external components

# The PyVCP checkbutton simulates a normally closed external E-Stop
net ext-estop classicladder.0.in-00 <= pyvcp.py-estop

# Request E-Stop Enable from LinuxCNC
net estop-all-ok iocontrol.0.emc-enable-in <= classicladder.0.out-00

# Request E-Stop Enable from PyVCP or external source
net ext-estop-reset classicladder.0.in-03 <= pyvcp.py-reset

# This line resets the E-Stop from LinuxCNC
net emc-reset-estop iocontrol.0.user-request-enable => classicladder.0.in-02

# This line enables LinuxCNC to unlatch the E-Stop in ClassicLadder
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01

# This line turns on the green indicator when out of E-Stop
net estop-all-ok => pyvcp.py-es-status
```

Затем мы добавляем следующие строки в файл Panel.xml. Обратите внимание, что вам нужно открыть его с помощью текстового редактора, а не средства просмотра HTML по умолчанию.

```
<pyvcp>
<vbox>
<label><text>”E-Stop 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>”E-Stop”</text>
</checkbutton>
</vbox>
<button>
<halpin>”py-reset”</halpin>
<text>”Reset”</text>
</button>
</pyvcp>
```

Теперь запустите свою конфигурацию, и она должна выглядеть так.

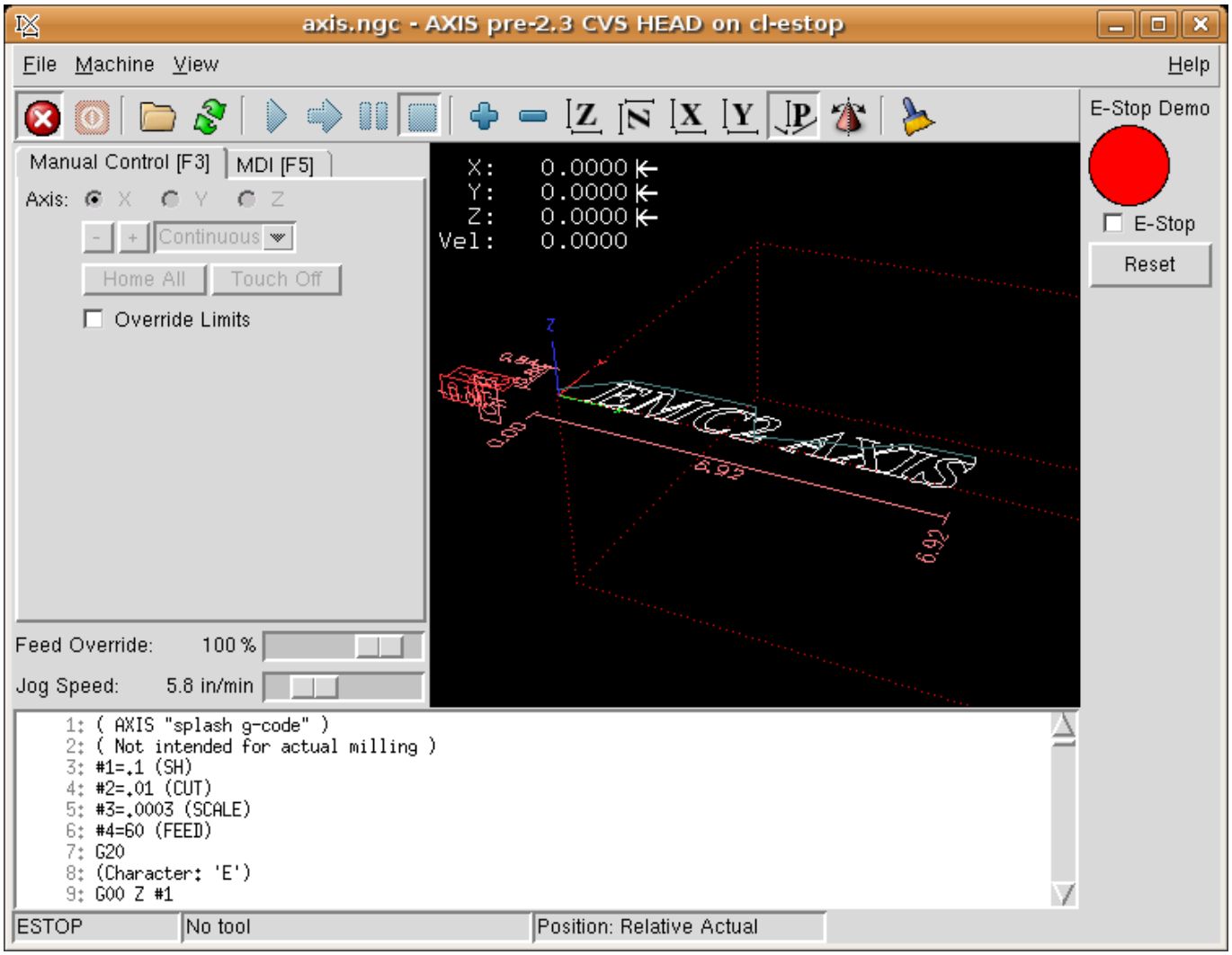



Figure 8.26: AXIS E-Stop

Обратите внимание, что в этом примере, как и в реальной жизни, вы должны очистить удаленный аварийный останов (имитируемый флажком), прежде чем аварийный останов AXIS или внешний сброс переведут вас в режим ВЫКЛ. Если кнопка аварийного останова на экране AXIS была нажата, вам придется нажать ее еще раз, чтобы очистить ее. Вы не можете выполнить сброс с внешнего устройства после выполнения аварийного останова в AXIS.

### 8.3.4 Timer/Operate пример

В этом примере мы используем блок Operate для присвоения значения предустановке таймера в зависимости от того, включен или выключен вход.

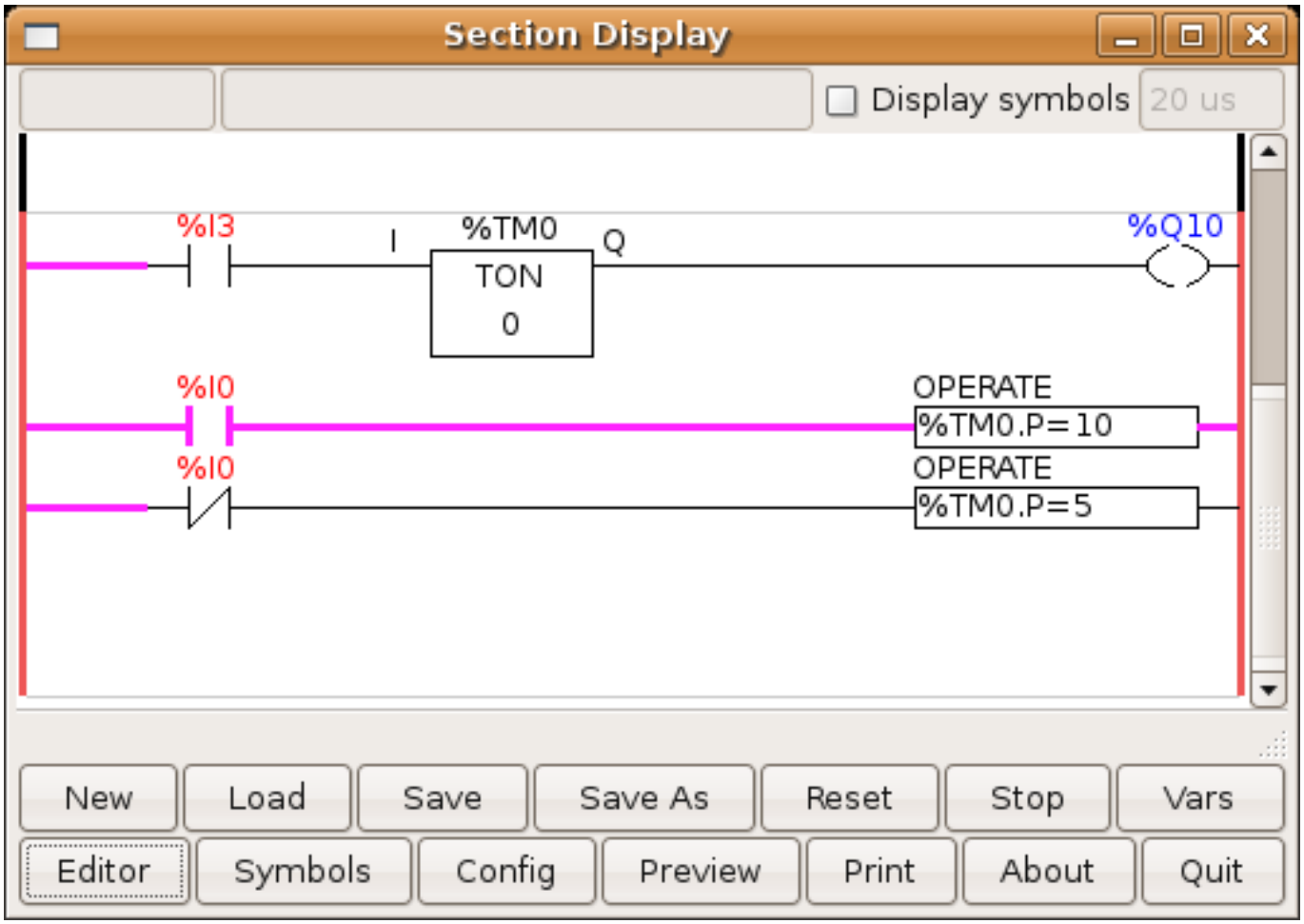



Figure 8.27: Timer/Operate пример

В этом случае %I0 имеет значение true, поэтому предустановленное значение таймера равно 10. Если %I0 было ложным, предустановленное значение таймера будет равно 5.

# Chapter 9

# Расширенные темы

## 9.1 Kinematics

### 9.1.1 Введение

Когда мы говорим о станках с ЧПУ, мы обычно думаем о станках, которым указано перемещаться в определенные места и выполнять различные задачи. Чтобы иметь единое представление о пространстве станка и соответствовать человеческой точке зрения в трехмерном пространстве, большинство станков (если не все) используют общую систему координат, называемую декартовой системой координат.

Декартова система координат состоит из трех осей (X, Y, Z), каждая из которых перпендикулярна двум другим footnote: [Слово ”оси” также часто (и ошибочно) используется, когда речь идет о станках с ЧПУ и относится к направлениям движения станка.].

Когда мы говорим о программе G-кода (RS274/NGC), мы говорим о ряде команд (G0, G1 и т. д.), которые имеют позиции в качестве параметров (X-Y-Z-). Эти позиции относятся именно к декартовым позициям. Часть контроллера движения LinuxCNC отвечает за перевод этих положений в положения, которые соответствуют кинематике станка footnote: [Кинематика: двусторонняя функция для преобразования из декартова пространства в пространство сочленения.].

#### 9.1.1.1 Сочленения против осей

Сочленение станка с ЧПУ является одной из физических степеней свободы станка. Оно может быть линейное (ШВП) или вращательное (поворотные столы, сочленения манипуляторов робота). На одном станке может быть любое количество сочленений. Например, у одного популярного робота 6 сочленений, а у типичного простого фрезерного станка — всего 3.

Существуют станки, в которых сочленения расположены в соответствии с кинематическими осями (соединение 0 по оси X, соединение 1 по оси Y, соединение 2 по оси Z), и такие станки называются Декартовы станки (или станки с (Тривиальной кинематикой). Это наиболее распространен станки, используемые при фрезеровании, но они не очень распространены в других областях управления машинами (например, сварка: роботы типа Puma).

LinuxCNC поддерживает оси с именами: X Y Z A B C U V W. Оси X Y Z обычно относятся к обычным декартовым координатам. Оси A B C относятся к координатам вращения относительно осей X Y Z соответственно. Оси U V W относятся к дополнительным координатам, которые обычно коллинеарны осям X Y Z соответственно.

### 9.1.2 Тривиальная кинематика

Простейшими станкаи являются станки, в которых каждое сочленение расположено вдоль одной из декартовых осей. На этих станках отображение декартова пространства (программы G-кода) в пространство сочленения (реальные исполнительные механизмы станка) тривиально. Это простое сопоставление 1:1:

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
```

В приведенном выше фрагменте кода можно увидеть, как выполняется отображение: позиция X идентична сочленению 0, позиция Y — сочленению 1 и т. д. Вышеупомянутое относится к прямой кинематике (одно направление преобразования). Следующий фрагмент кода относится к обратной кинематике (или обратному направлению преобразования):

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
```

В LinuxCNC тождественная кинематика реализована с помощью модуля кинематики *trivkins* и расширена до 9 осей. Соотношения по умолчанию между координатами осей и номерами сочленений следующие: footnote: [Если на станке (например, токарном станке) установлены только оси X, Z и A, а INI-файл LinuxCNC содержит только определения этих трех сочленений, то предыдущее утверждение неверно. Потому что в настоящее время у нас есть (joint0=X, Joint1=Z, Joint2=A), что предполагает, что Joint1=Y. Чтобы это работало в LinuxCNC, просто определите все оси (XYZA), LinuxCNC затем будет использовать простой цикл в HAL для неиспользуемой оси Y.] footnote: [Другой способ заставить это работать — изменить соответствующий код и перекомпилировать программное обеспечение. ]

```
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];
```

Аналогичным образом, отношения по умолчанию для обратной кинематики для trivkins таковы:

```
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;
```

Преобразование для тривиальной ”кинематики” (кинематика *trivkins*) или декартового станка выполнить несложно при условии, что в используемых буквах осей нет пропусков.

Ситуация становится немного сложнее, если на станке отсутствует одна или несколько букв оси. Проблемы с пропущенными буквами осей решаются с помощью параметра модуля *coordinates=* с модулем trivkins. Номера сочленений присваиваются последовательно каждой указанной координат Токарный станок можно описать с помощью *coordinates=xz*. Тогда назначения сочленения будут следующими:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.z
```

Использование параметра *coordinates=* рекомендуется для конфигураций, в которых отсутствуют буквы оси. [1]

Модуль кинематики *trivkins* также позволяет указывать одну и ту же координату для более чем одного сочленения. Эта функция может быть полезна на таких станках, как портальные, с двумя независимыми двигателями для координаты Y. Такой станок может использовать *coordinates=xyyz*, что приведет к назначеням сочленения:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.y
joints[2] = pos->tran.y
joints[3] = pos->tran.z
```

Дополнительную информацию смотрите на страницах руководства trivkins.

### 9.1.3 Нетривиальная кинематика

Типов настроек станков может быть довольно много (роботы: пума, скара; гексаподы и т. д.). Каждый из них настраивается с помощью линейных и поворотных сочленений. Эти сочленения обычно не совпадают с декартовыми координатами, поэтому нам нужна кинематическая функция, которая выполняет преобразование (фактически две функции: прямая и обратная кинематика).

Для иллюстрации вышеизложенного разберем простую кинематику, называемую бипод (упрощенный вариант трипода, представляющий собой упрощенный вариант гексапода).

[1] Исторически модуль trivkins не поддерживал параметр *coordinates=*, поэтому конфигурации токарных станков часто настраивались как станки XYZ. Неиспользуемая ось Y была настроена на 1) немедленное возвращение в исходное положение, 2) использование простой обратной связи для подключения контакта HAL команды положения к контакту HAL обратной связи по положению и 3) скрытие на ГИП дисплеев . Многие конфигурации SIM используют эти методы для совместного использования общих файлов HAL.

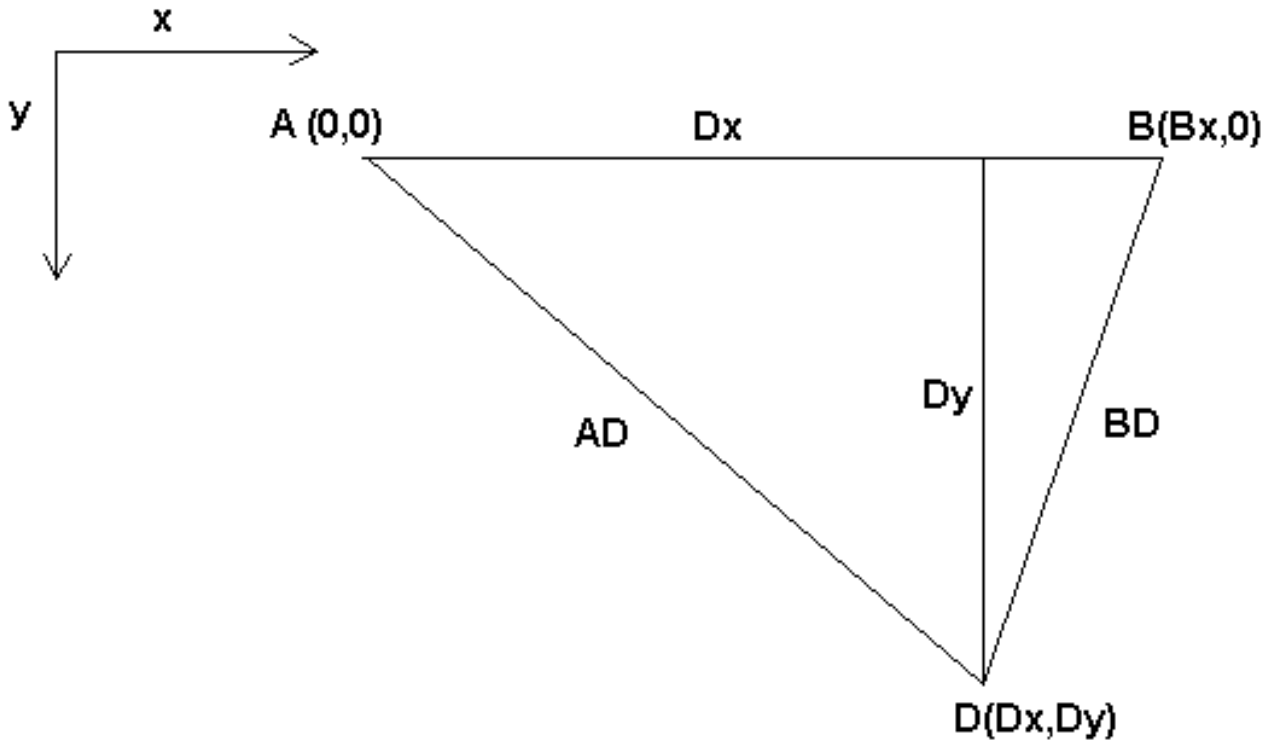



Figure 9.1: Настройка бипода

Бипод, о котором мы говорим, представляет собой устройство, состоящее из двух двигателей, размещенных на стене, к которой устройство подвешивается с помощью проволоки. Сочленениями в данном случае являются расстояния от двигателей до устройства (на рисунке обозначены AD и BD).

Положение двигателей фиксировано по соглашению. Двигатель A находится в (0,0), что означает, что его координата X равна 0, а его координата Y также равна 0. Двигатель B помещен в (Bx, 0), что означает, что его координата X равна Bx.

Наша режущая кромка инструмента будет находиться в точке D, которая определяется расстояниями AD и BD, а также декартовыми координатами Dx, Dy.

Задача кинематики заключается в преобразовании длин соединений (AD, BD) в декартовы координаты (Dx, Dy) и наоборот.

#### 9.1.3.1 Forward transformation

Для преобразования из пространства сочленения в декартово пространство мы воспользуемся некоторыми правилами тригонометрии (прямоугольные треугольники, определяемые точками (0,0), (Dx,0), (Dx,Dy) и треугольник (Dx,0), (Bx ,0) и (Dx,Dy)).

Мы легко можем это увидеть:

$$AD^2 = x^2 + y^2 \quad BD^2 = (Bx - x)^2 + y^2$$

так же:

$$BD^2 = (Bx - x)^2 + y^2$$

Если вычесть одно из другого, то получим:

$$AD^2 - BD^2 = x^2 + y^2 - x^2 + 2 * x * Bx - Bx^2 - y^2$$

и поэтому:

$$x = \frac{AD^2 - BD^2 + Bx^2}{2 * Bx}$$

и поэтому:

$$y = \sqrt{AD2 - x2}$$

Обратите внимание, что вычисление y включает квадратный корень из разницы, что может не привести к действительному числу. Если для этого совместного положения не существует единой декартовой координаты, то такое положение называется особенностью. В этом случае прямая кинематика возвращает -1.

Переведено на реальный код:

```
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 Обратное преобразование

Обратная кинематика в нашем примере намного проще, так как мы можем записать ее непосредственн

$$AD = \sqrt{x^2 + y^2}$$

$$BD = \sqrt{(Bx - x)^2 + y^2}$$

или переведено в реальный код:

```
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 Детали реализации

Модуль кинематики реализован как компонент HAL и может экспортировать контакты и параметры. Он состоит из нескольких функций ”C” (в отличие от функций HAL):

```
int kinematicsForward(const double *joint, EmcPose *world,
const KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Реализует forward kinematics function.

```
int kinematicsInverse(const EmcPose * world, double *joints,
const KINEMATICS_INVERSE_FLAGS *iflags,
KINEMATICS_FORWARD_FLAGS *fflags)
```

Реализует функцию обратной кинематики.

```
KINEMATICS_TYPE kinematicsType(void)
```

Возвращает идентификатор типа кинематики, например *KINEMATICS_BOTH*:

1. KINEMATICS_IDENTITY (каждый номер сочленения соответствует букве оси)
2. KINEMATICS_BOTH (предусмотрены функции прямой и обратной кинематики)
3. KINEMATICS_FORWARD_ONLY
4. KINEMATICS_INVERSE_ONLY

---

**Note**
ГИПы могут интерпретировать KINEMATICS_IDENTITY, чтобы скрыть различия между номерами сочленений и буквами осей в режиме сочленения (обычно до возврата в исходное положение).

---

```
int kinematicsSwitchable(void)
int kinematicsSwitch(int switchkins_type)
KINS_NOT_SWITCHABLE
```

Функция kinematicsSwitchable() возвращает 1, если поддерживается несколько типов кинематики. Функция kinematicsSwitch() выбирает тип кинематики. См. Switchable Kinematitcs.

---

**Note**
Большинство предоставляемых модулей кинематики поддерживают один тип кинематики и используют директиву ”**KINS_NOT_SWITCHABLE**” для предоставления значений по умолчанию для необходимых функций kinematicsSwitchable() и kinematicsSwitch().

---

```
int kinematicsHome(EmcPose *world, double *joint,
KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Функция кинематики исходного положения устанавливает для всех своих аргументов правильные значения в известной исходной позиции. При вызове им следует установить, если они известны, начальные значения, например, из INI-файла. Если кинематика исходного положения допускает произвольные начальные точки, следует использовать эти начальные значения.

```
int rtapi_app_main(void)
void rtapi_app_exit(void)
```

Это стандартные функции настройки и разборки модулей RTAPI.

Если модули кинематики содержатся в одном исходном файле, их можно скомпилировать и установить с помощью команды *halcompile*. Дополнительную информацию смотрите на странице руководства *halcompile(1)* или в руководстве по HAL.

#### 9.1.4.1 Модуль кинематики с использованием шаблона userkins.comp

Другой способ создать собственный модуль кинематики — адаптировать *userkins* компонента HAL. Этот компонент шаблона может быть изменен пользователем локально и может быть создан с помощью halcompile.

Дополнительную информацию см. на страницах руководства пользователя userkins.

Отметим, что для создания переключаемых кинематических модулей необходимые доработки несколько сложнее.

См. *millturn.comp* как пример переключаемого кинематического модуля, созданного с использованием шаблона *userkins.comp*.

## 9.2 Setting up ”modified” Denavit-Hartenberg (DH) parameters for *genserkins*

### 9.2.1 Prelude

LinuxCNC supports a number of kinematics modules including one that supports a generalized set of serial kinematics commonly specified via Denavit-Hartenberg parameters.

This document illustrates a method to set up the DH-parameters for a Mitsubishi RV-6SDL in LinuxCNC using *genserkins* kinematics.

---

**Note**
This document does not cover the creation of a *vismach* model which, while certainly very useful, requires just as much careful modeling if it is to match the *genserkins* model derived in this document.

---

---

**Note**
There may be errors and/or shortcomings — use at your own risk!

---

### 9.2.2 General

With the proliferation of industrial robots comes an increased interest to control used robots with LinuxCNC. A common type of robot used in industry and manufacturing is the ”serial manipulator” designed as a series of motorized joints connected by rigid links. Serial robots often have six joints as required for the six degrees of freedom needed to both position (XYZ) and orient (ABC or pitch, roll, yaw) an object in space. Often these robots have an arm structure that extends from a base to an end-effector.

Control of such a serial robot requires the calculation of the end-effector's position and orientation in relation to a reference coordinate system when the joint angles are known (**forward kinematics**) and also the more complex reverse calculation of the required joint angles for a given end-effector position and orientation in relation to the reference coordinate system (**inverse kinematics**). The standard mathematical tools used for these calculations are matrices which are basically tables of parameters and formulas that make it easier to handle the rotations and translations involved in forward and inverse kinematics calculations.

Detailed familiarity with the math is not required for a serial robot since LinuxCNC provides a kinematics module that implements an algorithm called *genserkins* to calculate the forward and inverse kinematics for a generic serial robot. In order to control a specific serial robot, *genserkins* must be provided with data so that it can build a mathematical model of the robot's mechanical structure and thus do the math.

The required data needs to be in a standardized form that has been introduced by Jacques Denavit and Richard Hartenberg back in the fifties and are called the DH-Parameters. Denavit and Hartenberg used four parameters to describe how one joint is linked to the next. These parameters describe basically two rotations (*alpha* and *theta*) and two translations (*a* and *d*).

### 9.2.3 Modified DH-Parameters

Как это часто бывает, этот "стандарт" был модифицирован другими авторами, которые ввели «модифицированные параметры DH», и нужно быть очень осторожным, поскольку *genserkins* использует «модифицированные параметры DH», как описано в публикации «Введение в Робототехник механику и управление» Джона Дж. Крейга. Будьте осторожны: существует много информации о «параметрах DH», но автор редко определяет, какое соглашение фактически используется. Кроме того, некоторые люди сочли необходимым изменить параметр с именем *a* на *r*, что еще больше усугубило путаницу. Этот документ соответствует соглашению, упомянутому в вышеупомянут публикации Крейга, с той разницей, что нумерация сочленений и параметров начинается с цифры «0», чтобы соответствовать *genserkins* и его контактам HAL.

Standard and Modified DH-Parameters consist of four numeric values for each joint (*a*, *d*, *alpha* and *theta*) that describe how the coordinate system (CS) sitting in one joint has to be moved and rotated to be aligned with the next joint. Aligned means that the Z-axis of our CS coincides with the axis of rotation of the joint and points in the positive direction such that, using the right hand rule with the thumb pointing in the positive direction of the Z-axis, the fingers point in the positive direction of rotation of the joint. It becomes clear that in order to do this, one must decide on the positive directions of all joints before starting to derive the parameters!

The difference between "standard" and "modified" notations is in how the parameters are allocated to the links. Using the "standard" DH-Parameters in *genserkins* will **not** give the correct mathematical model.

### 9.2.4 Modified DH-Parameters as used in *genserkins*

Note that *genserkins* does not handle offsets to theta-values — theta is the joint variable that is **controlled** by LinuxCNC. With the CS aligned with the joint, a rotation around its Z-Axis is identical to the rotation commanded to that joint by LinuxCNC. This makes it impossible to define the 0° position of our robots joints arbitrarily.

The three configurable parameters are:

1. **alpha** : positive or negative rotation (in radians) around the X-axis of the "current coordinate system"
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 or negative length along Z (also in *machine units*)

The parameter sets are always derived in the same order and a set is completed by setting the d-parameter. This does not leave the Z-axis of our CS aligned with the next joint! This may seem confusing but sticking to this rule will yield a working set of parameters. Once the **d**-parameter is set, the X-axis of our-CS needs to point to the axis of the next joint.

### 9.2.5 Numbering of joints and parameters

The first joint in LinuxCNC is joint-0 (because in software counting starts with 0) while most publications start with the number *1*. That goes for all the parameters as well. That is, numbering starts with a-0, alpha-0, d-0 and ends with a-5, alpha-5 and d-5. Keep this in mind when following a publication to set up *genserkins* parameters.

### 9.2.6 How to start

Convention is to start by placing the reference-CS in the base of the robot with it's Z-axis coinciding with the axis of the first joint and its X-axis pointing toward the next joint's axis.

This will also result in the DRO values in LinuxCNC being referenced to that point. Having done so sets a-0 and alpha-0 to 0. The above mentioned publication (Craig) also sets d-0 to 0, which is confusing when a displacement offset is needed in order to have the reference-CS at the bottom of the base. Setting d-0 = to the displacement gives correct results. In this manner, the first set of parameters are alpha-0 = 0, a-0 = 0, d0 = displacement, and the X-axis of the CS points to the axis of the next joint (joint-1).

Derivation of the net set (alpha-1, a-1, d-1) follows — always using the same sequence all the way to the sixth set (alpha-5, a-5, d-5).

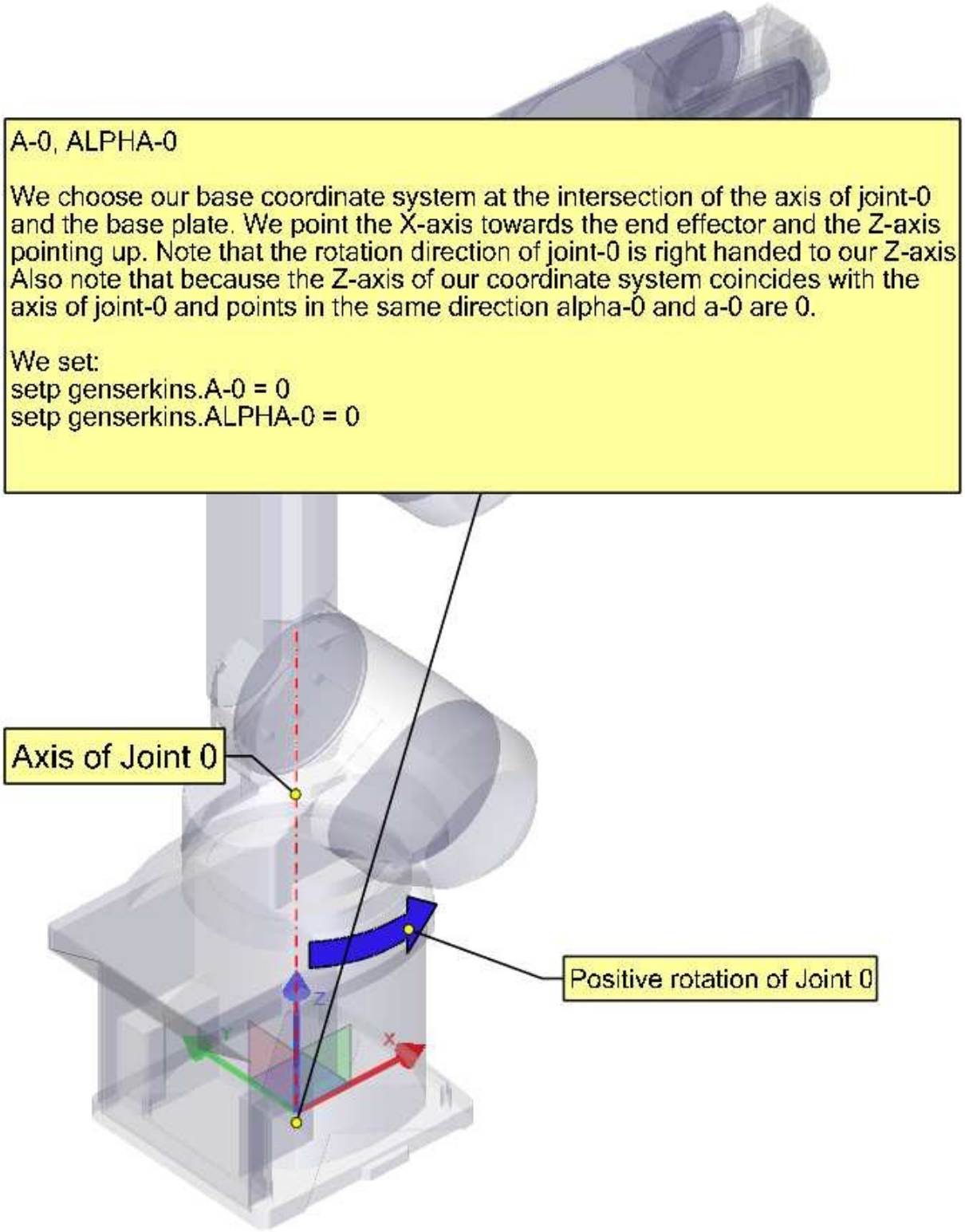
And thus, the TCP-CS of the end-effector is sitting in the center of the hand flange.

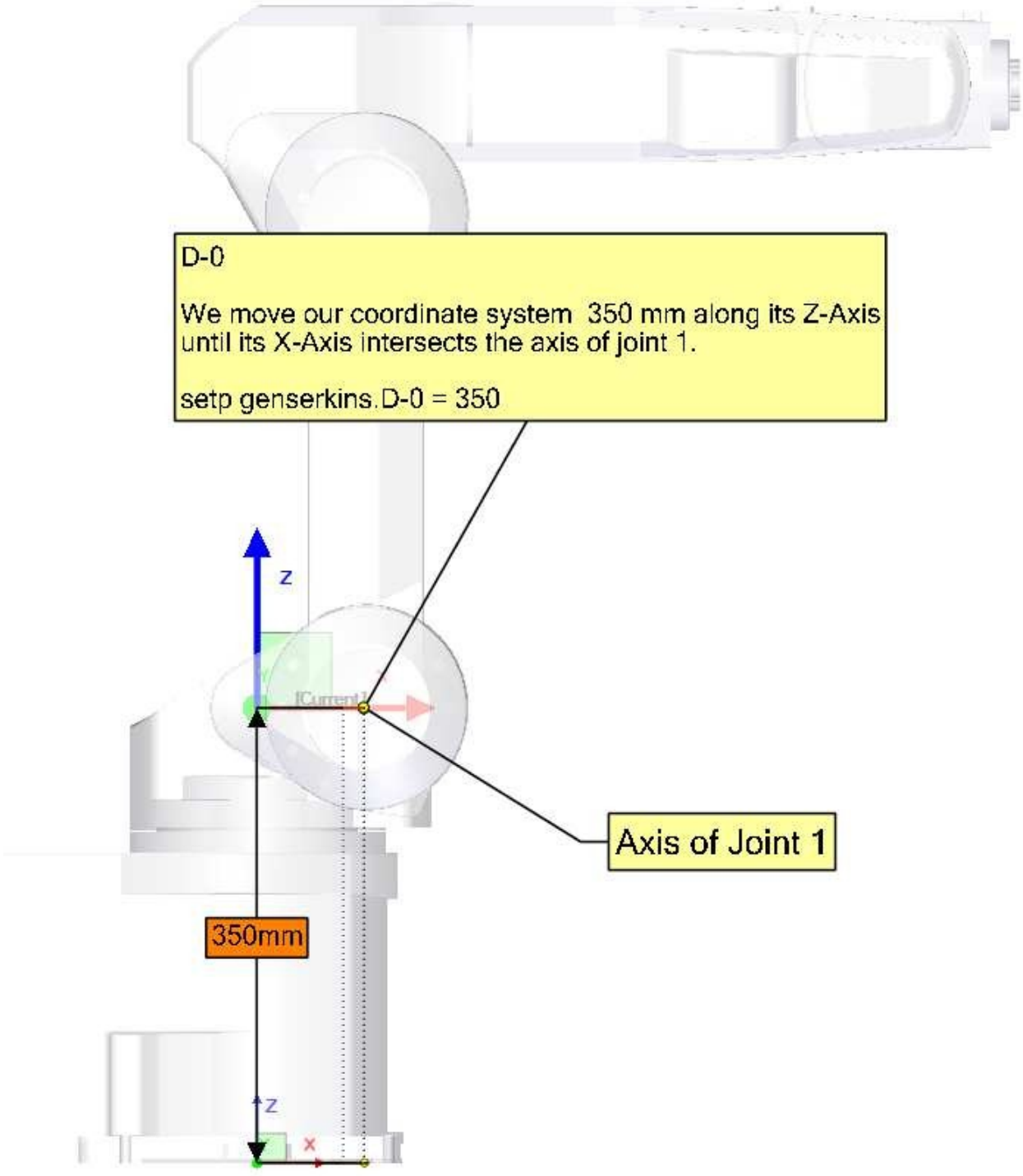
### 9.2.7 Special cases

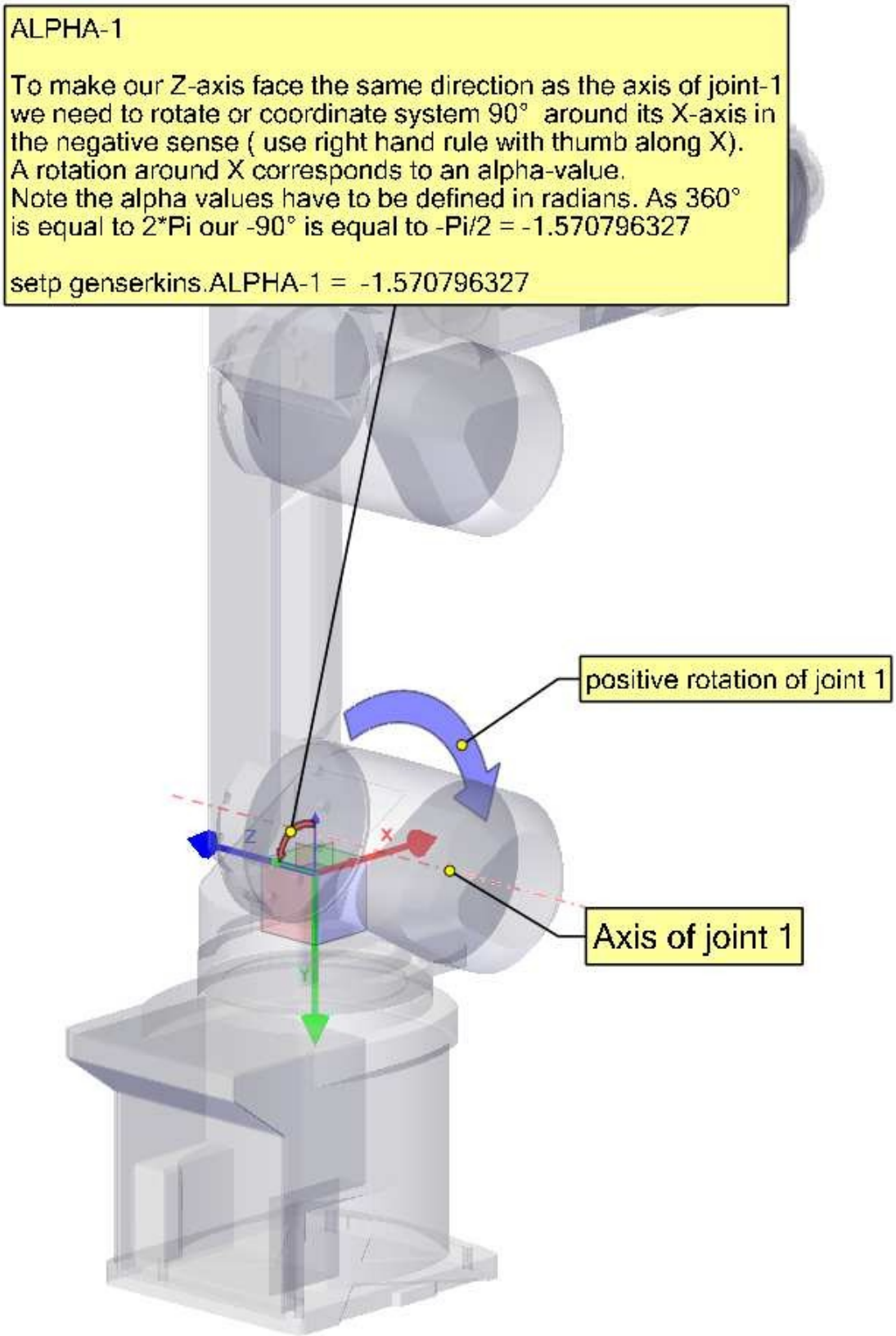
If the next joint-axis is parallel to the last then one could arbitrarily choose a value for the d-parameter but there is no point in setting it other than 0.

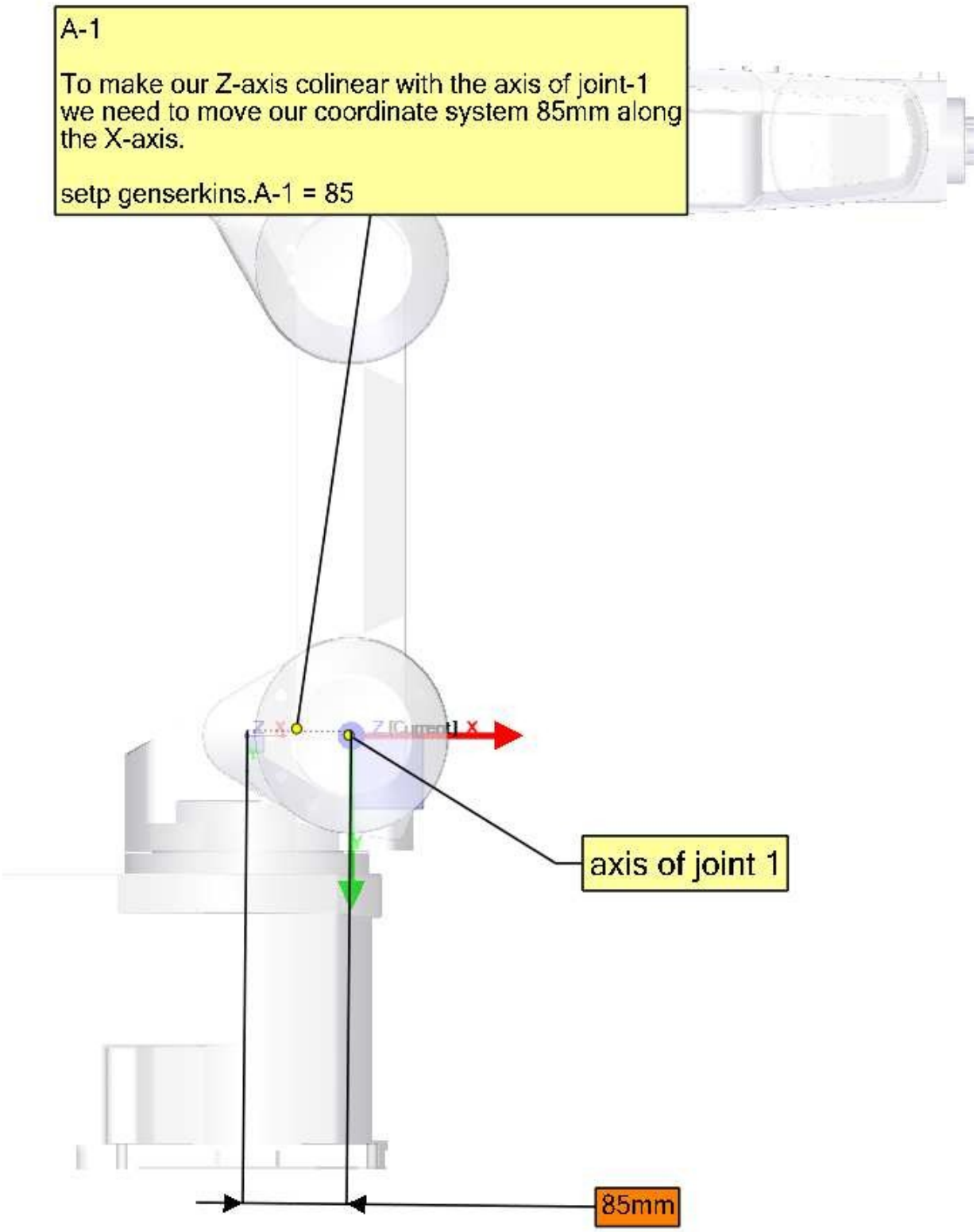
### 9.2.8 Detailed Example (RV-6SL)

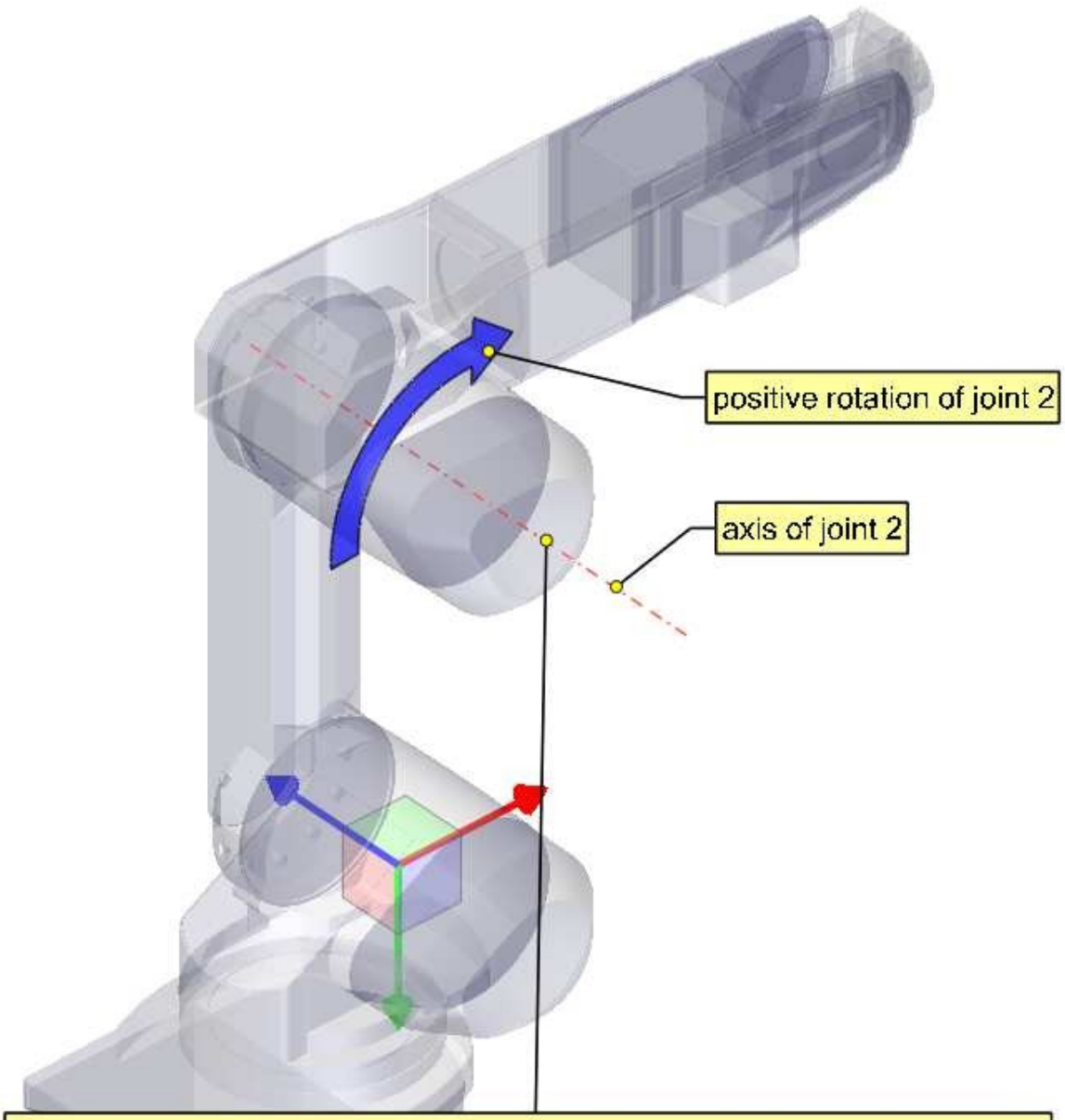
Described below is a method to derive the required "modified DH-parameters" for a Mitsubishi RV-6SDL and how to set the parameters in the HAL file to be used with the *genserkins* kinematics in LinuxCNC. The necessary dimensions are best taken from a dimensional drawing provided by the manufacturer of the robot.


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

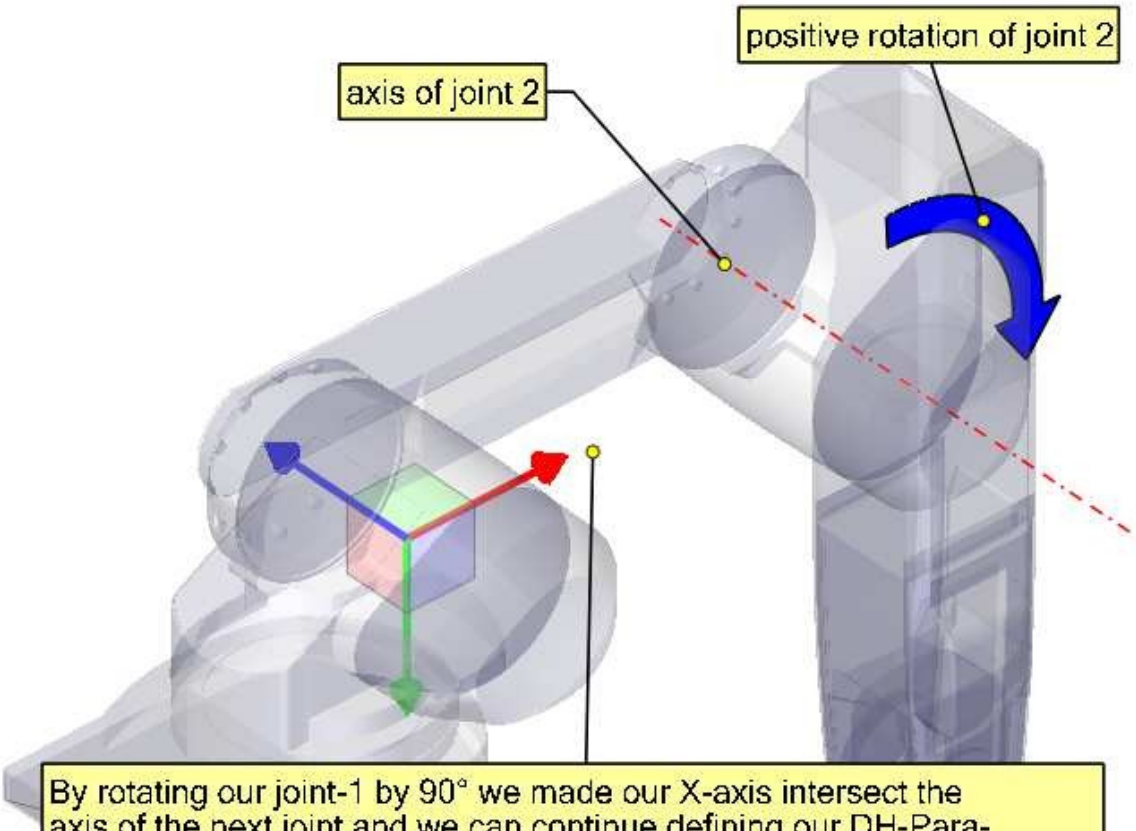

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


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

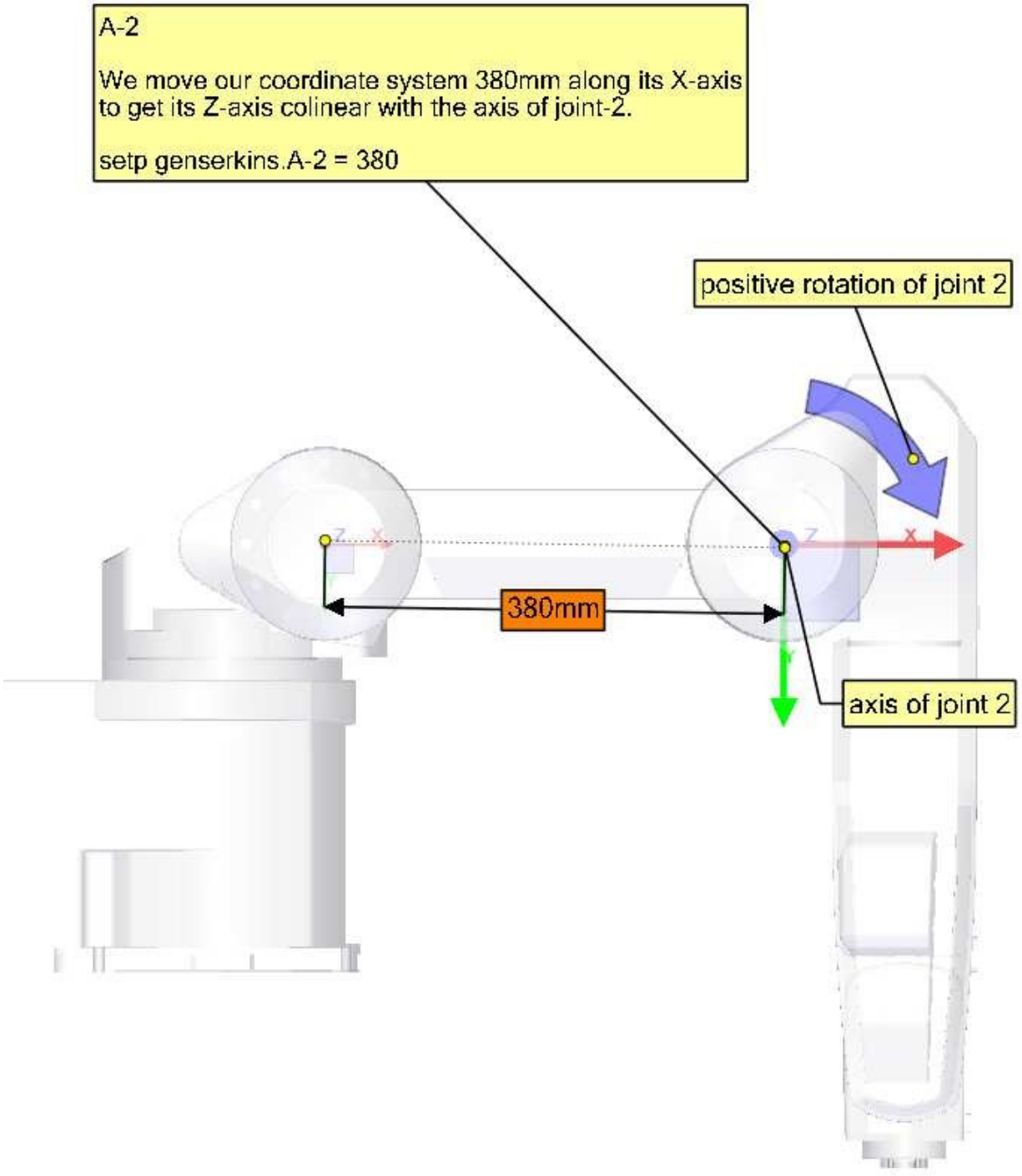
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

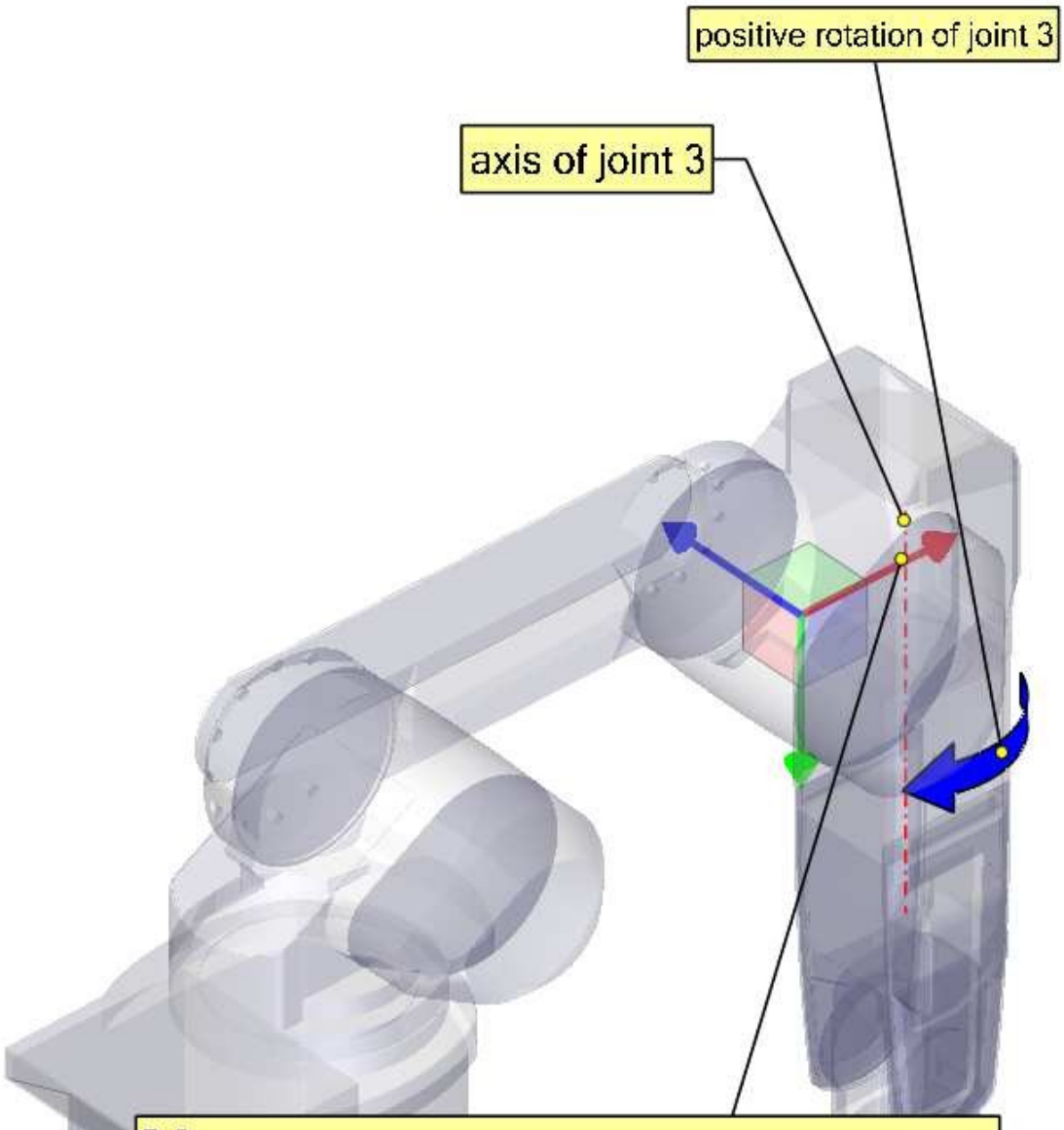
setp genserkins.D-1 = 0

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

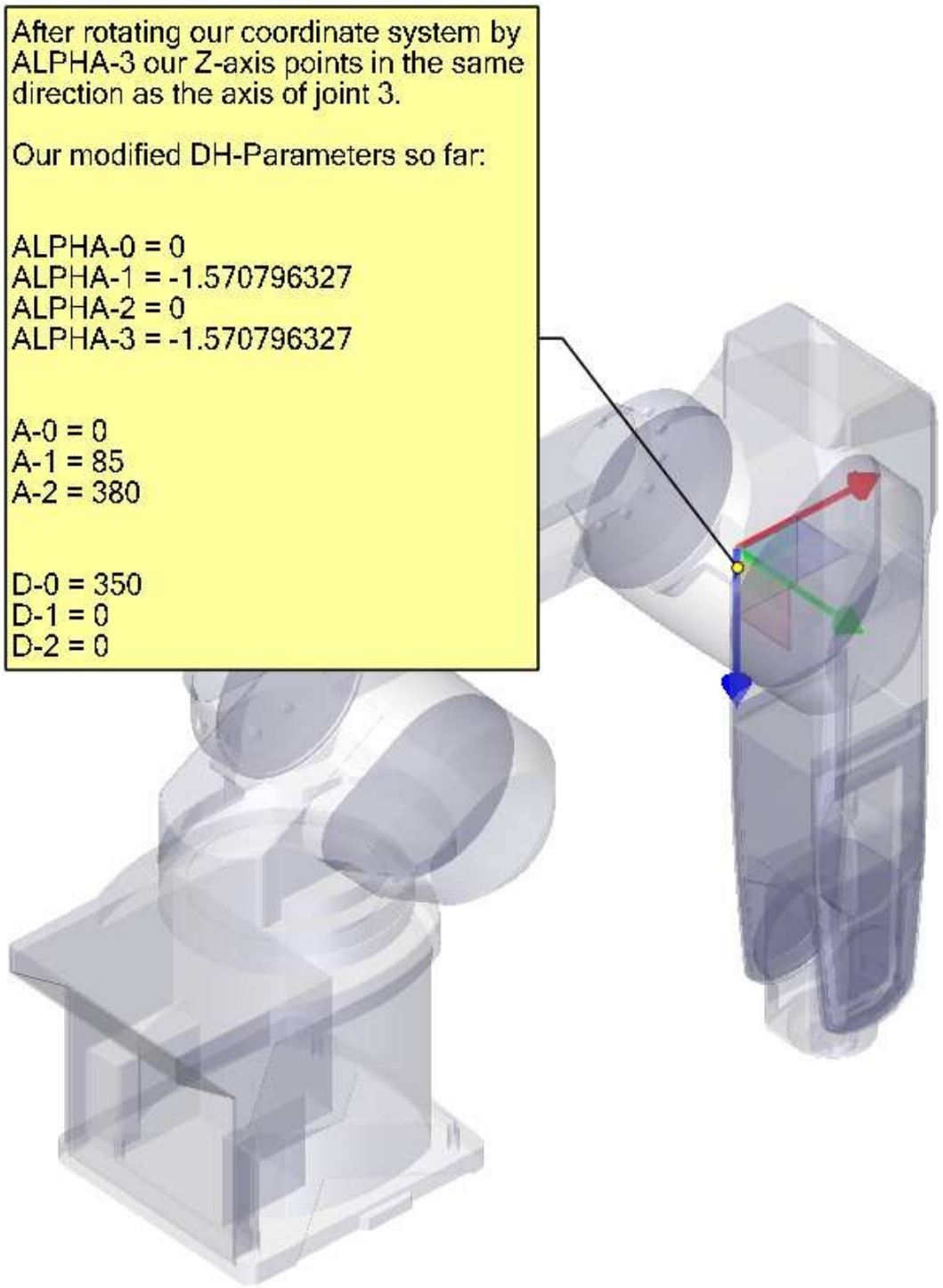
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.

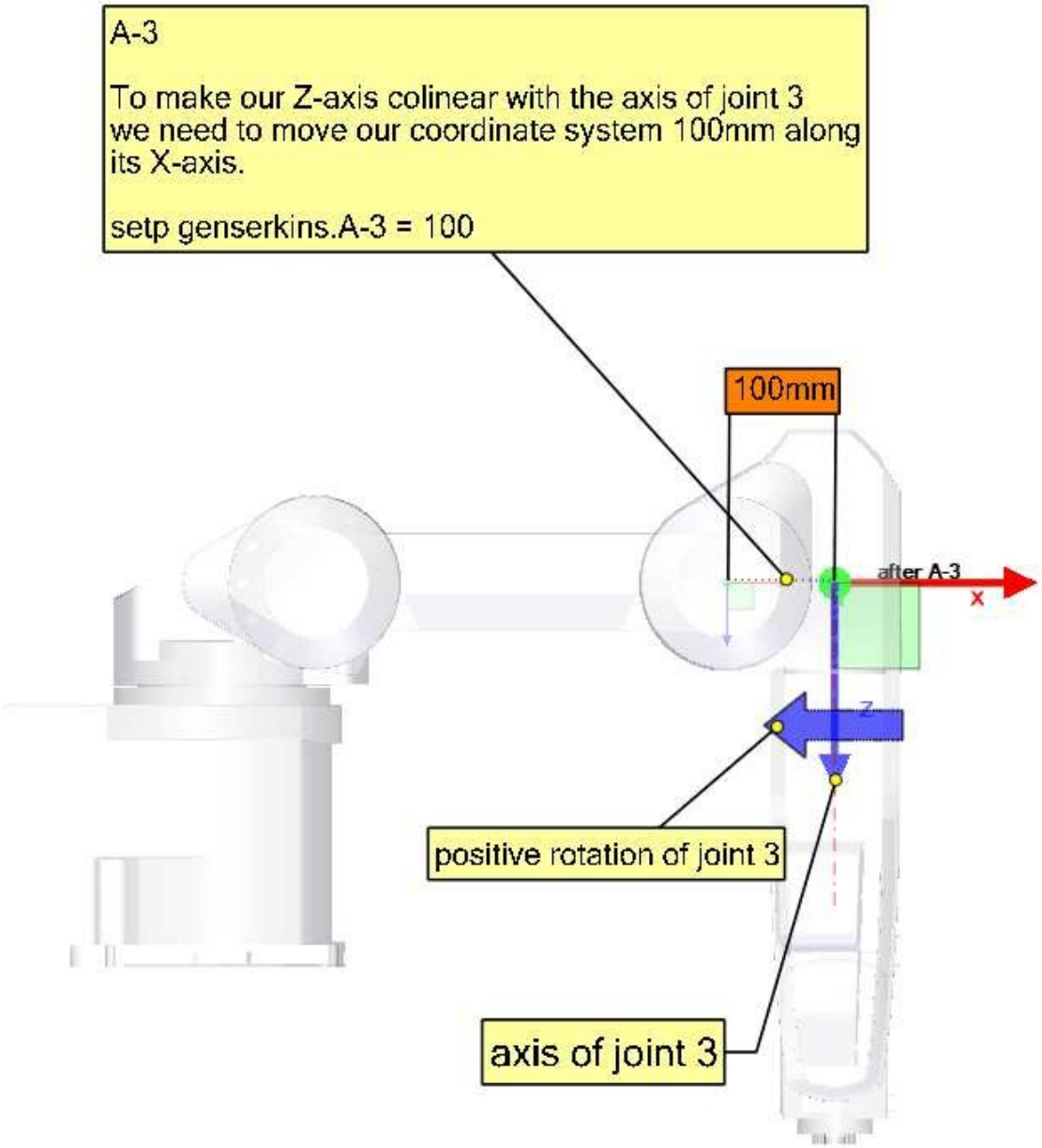
setp genserkins.D-2 = 0

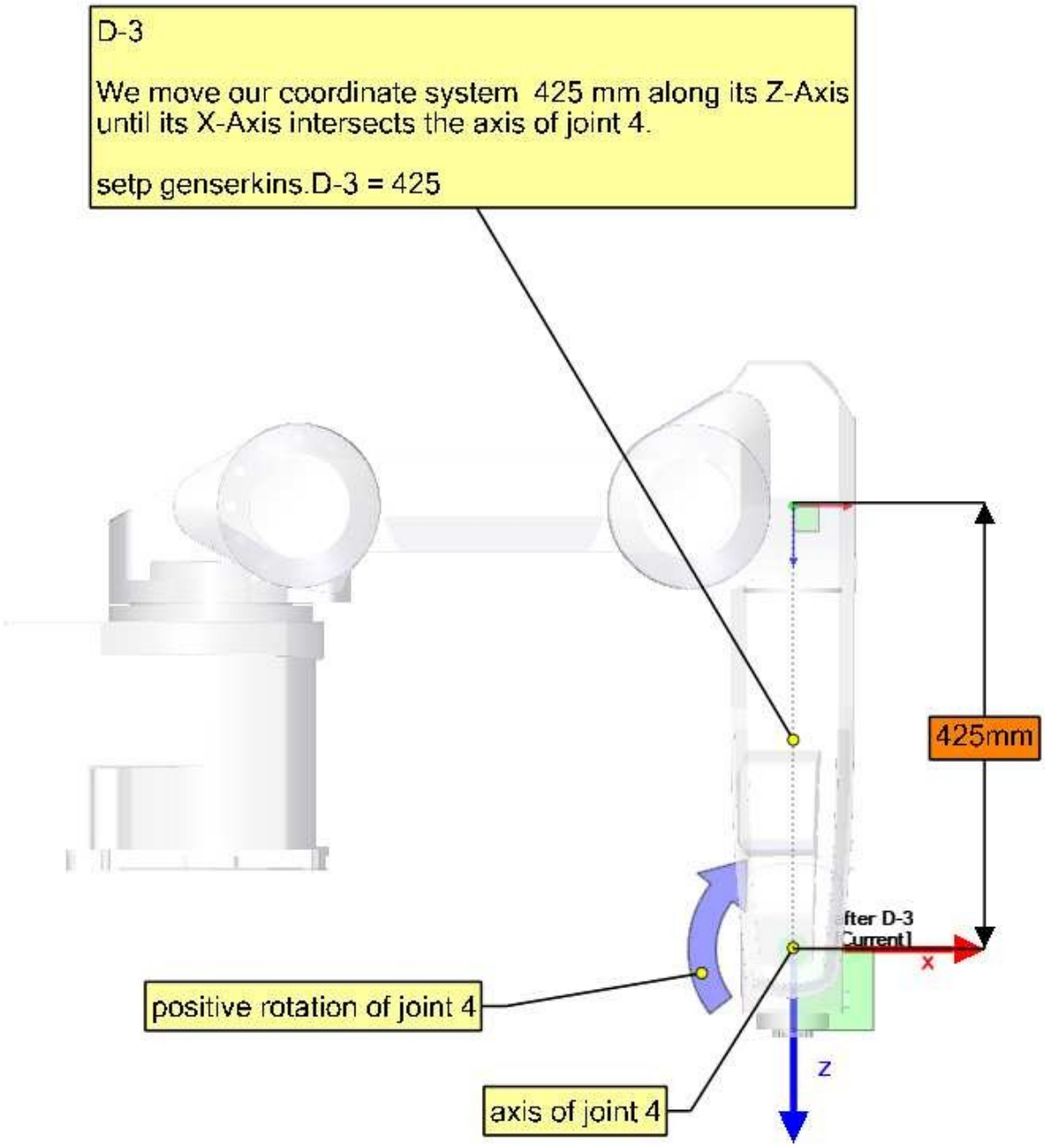
ALPHA-3

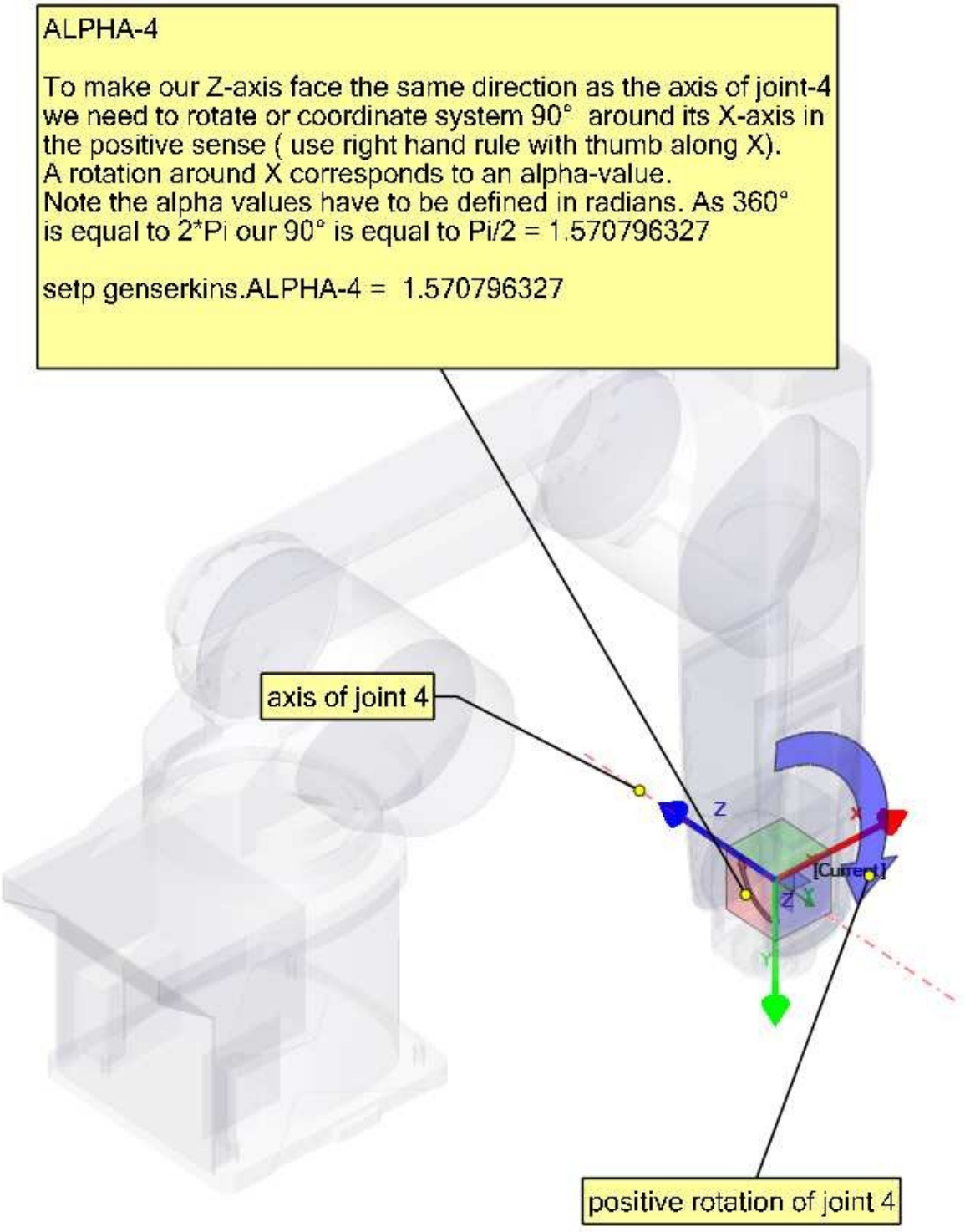
To make our Z-axis face the same direction as the axis of joint-3 we need to rotate or coordinate system 90° around its X-axis in the negative sense ( use right hand rule with thumb along X).
A rotation around X corresponds to an alpha-value.
Note the alpha values have to be defined in radians. As 360° is equal to 2*Pi our -90° is equal to -Pi/2 = -1.570796327

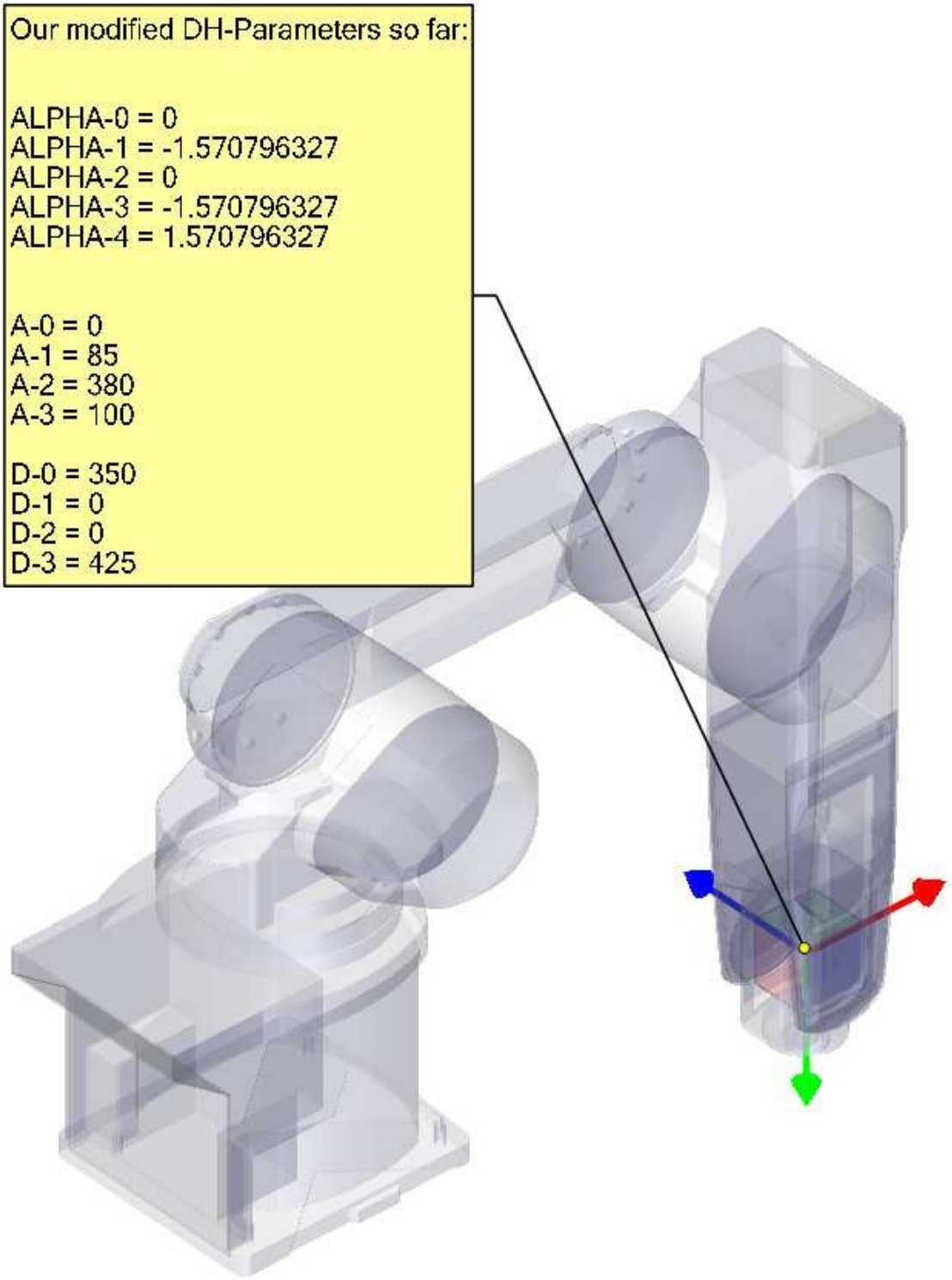
setp genserkins.ALPHA-3 = -1.570796327

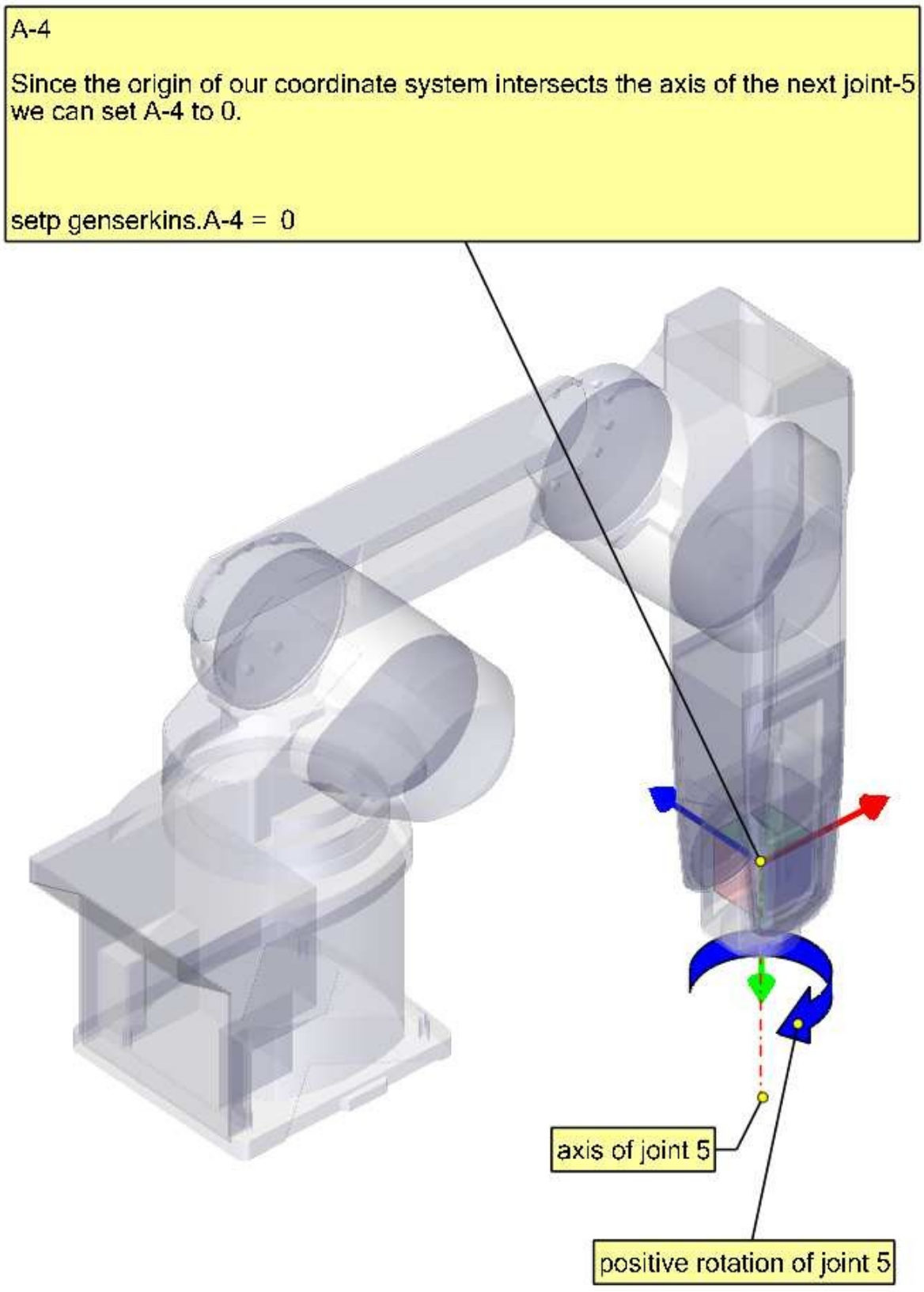

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


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

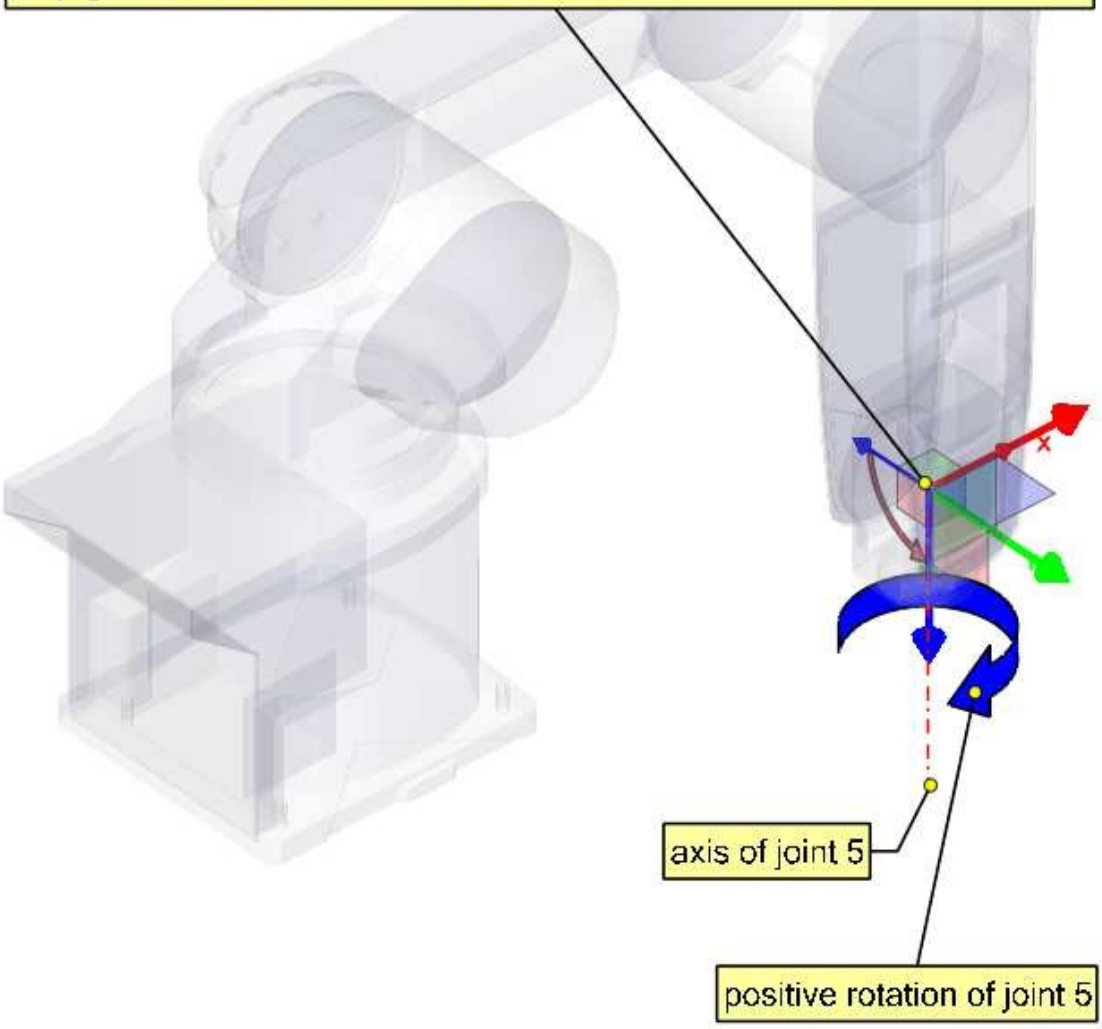
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

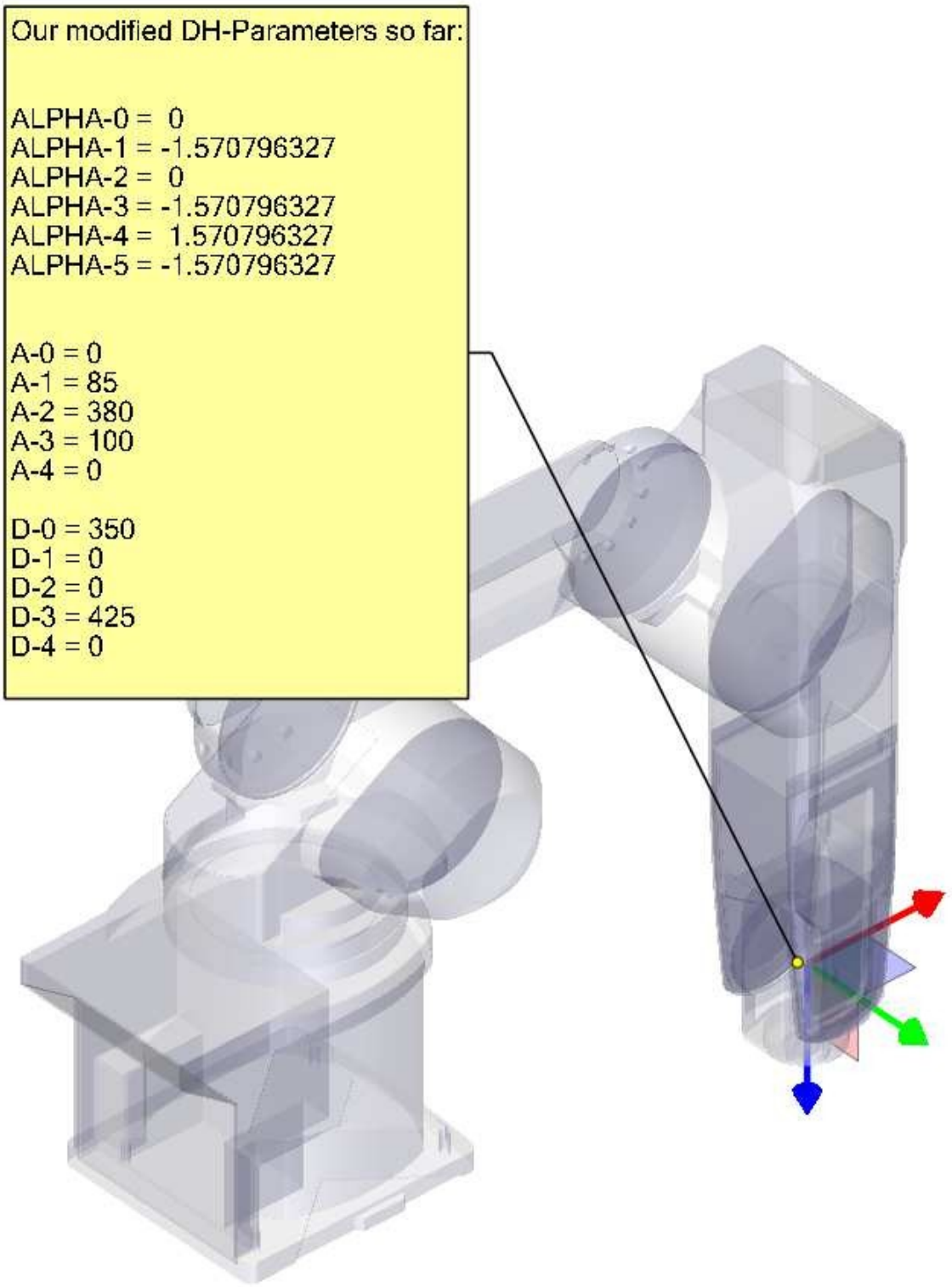
setp genserkins.D-4 = 0

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



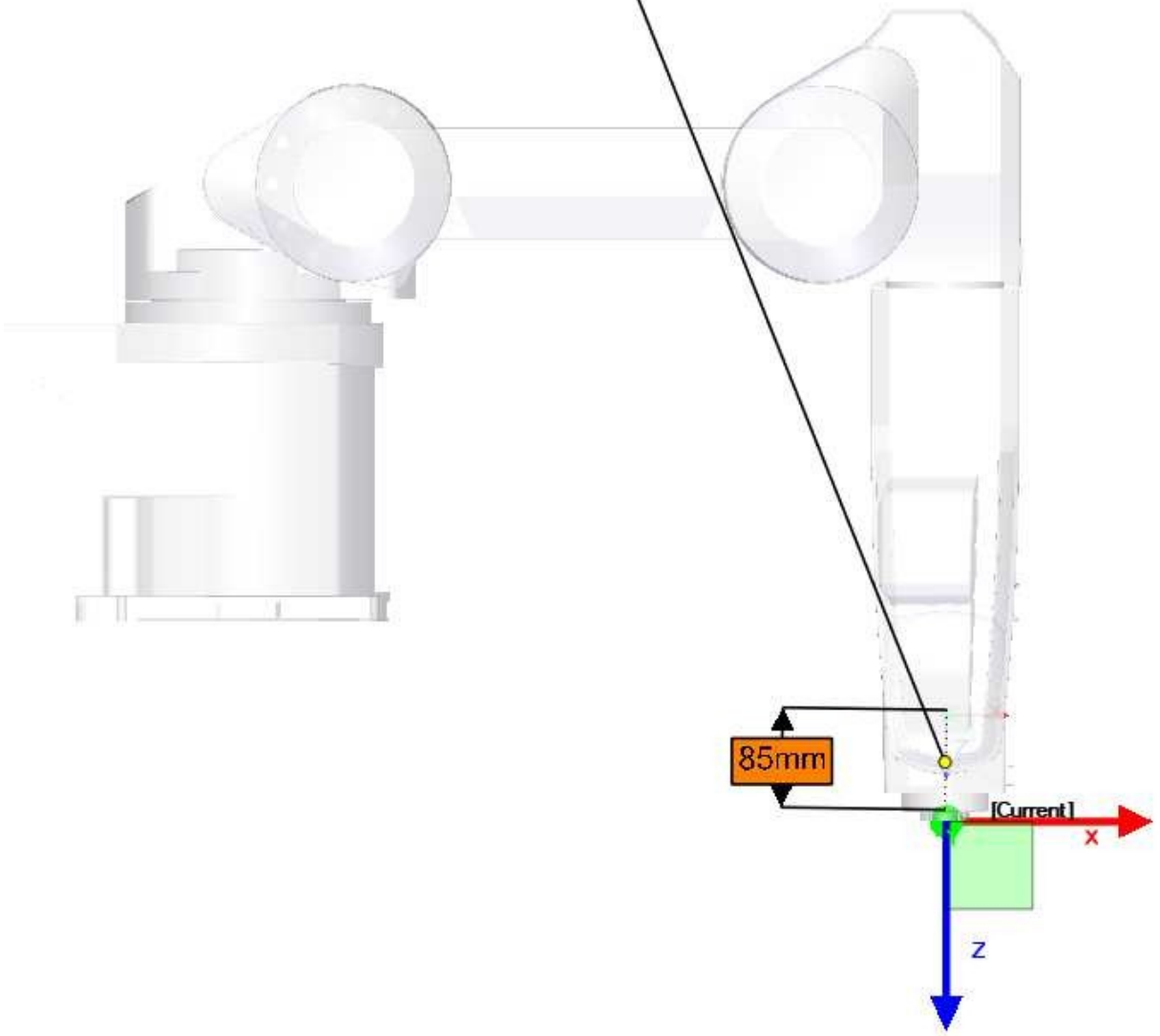

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0

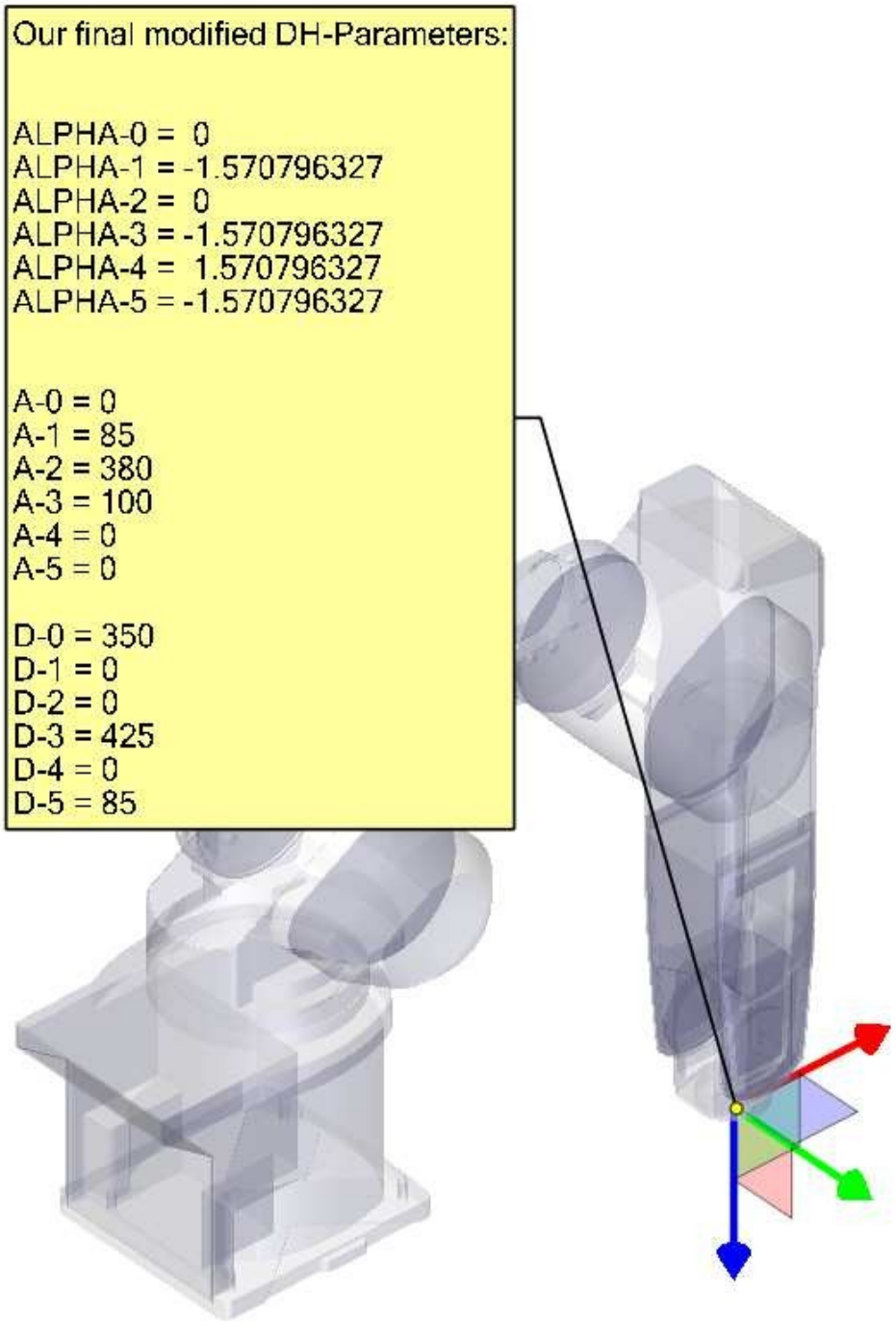
D-5

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 Credits

Thanks to user Aciera for all text and the graphics for the RV-6SL robot!

## 9.3 5-Axis Kinematics

### 9.3.1 Введение

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.

The kinematics components are given as well as vismach simulation models to demonstrate their behaviour on a computer screen. Examples of HAL file data are also given.

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-Axis Machine Tool Configurations

In this section we deal with the typical 5-axis milling or router machines with five joints or degrees-of-freedom which are controlled in coordinated moves.

3-axis machine tools cannot change the tool orientation, so 5-axis machine tools use two extra axes to set the cutting tool in an appropriate orientation for efficient machining of freeform surfaces.

Typical 5-axis machine tool configurations are shown in Figs. 3, 5, 7 and 9-11 [1,2] in section Figures.

The kinematics of 5-axes machine tools are much simpler than that of 6-axis serial arm robots, since 3 of the axes are normally linear axes and only two are rotational axes.

### 9.3.3 Tool Orientation and Location

CAD/CAM systems are typically used to generate the 3D CAD models of the workpiece as well as the CAM data for input to the CNC 5-axis machine. The tool or cutter location (CL) data, is composed of the cutter tip position and the cutter orientation relative to the workpiece coordinate system. Two vectors, as generated by most CAM systems and shown in Fig. 1, contain this information:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \quad \text{orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \quad \text{position vector} \qquad (1)$$

The K vector is equivalent to the 3rd vector from the pose matrix $E_6$ that was used in the 6-axis robot kinematics [3] and the Q vector is equivalent to the 4th vector of $E_6$. In MASTERCAM for example this information is contained in the intermediate output ”.nci” file.

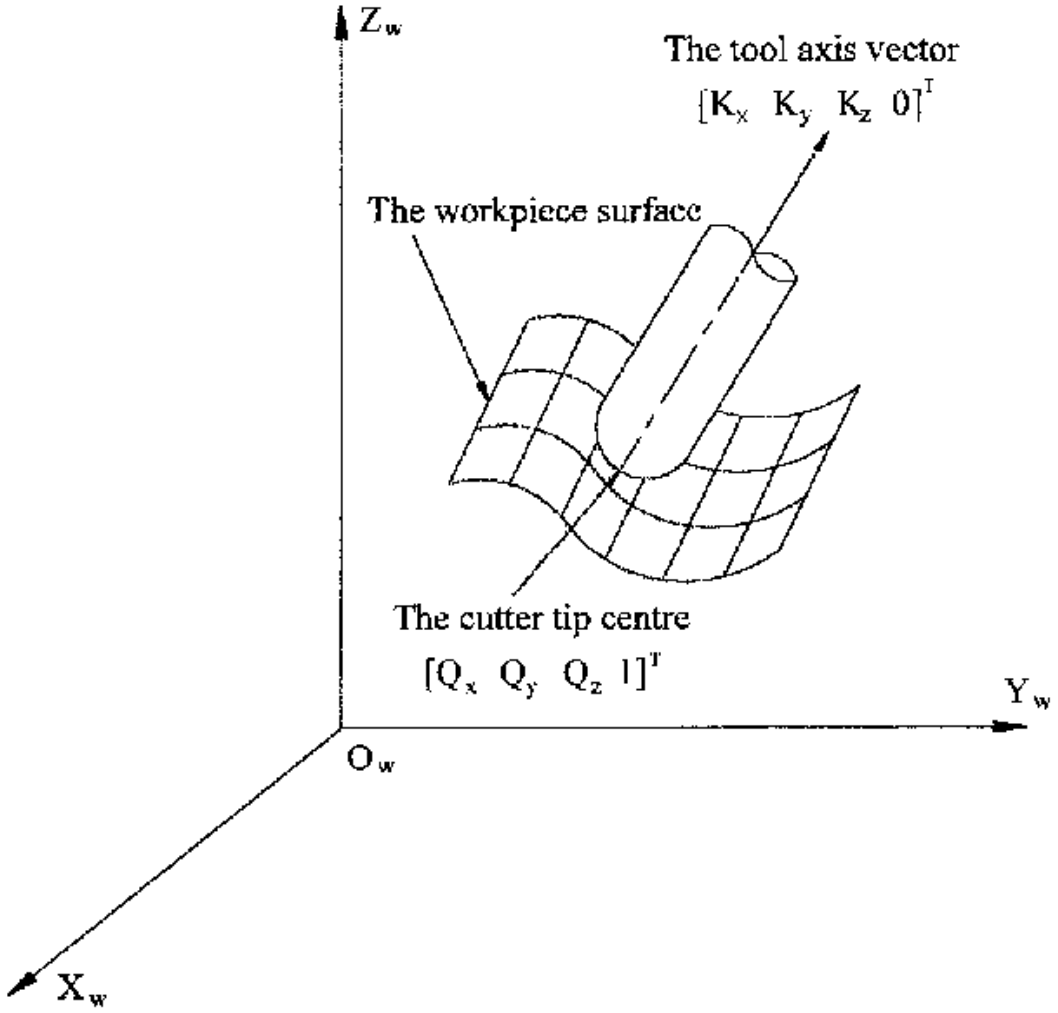



Figure 9.2: Cutter location data

### 9.3.4 Translation and Rotation Matrices

Homogeneous transformations provide a simple way to describe the mathematics of multi-axis machine kinematics. A transformation of the space H is a 4x4 matrix and can represent translation and rotation transformations. Given a point x,y,x described by a vector u = $\{x,y,z,1\}^T$, then its transformation v is represented by the matrix product

$$v = H \cdot u$$

There are four fundamental transformation matrices on which 5-axis kinematics can be based:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The matrix T(a,b,c) implies a translation in the X, Y, Z coordinate directions by the amounts a, b, c respectively. The R matrices imply rotations of the angle theta about the X, Y and Z coordinate axes respectively. The *C* and *S* symbols refer to cosine and sine functions respectively.

### 9.3.5 Table Rotary/Tilting 5-Axis Configurations

In these machine tools the two rotational axes mount on the work table of the machine. Two forms are typically used:

- A rotary table which rotates about the vertical Z-axes (C-rotation, secondary) mounted on a tilting table which rotates about the X- or Y-axis (A- or B-rotation, primary). The workpiece is mounted on the rotary table.
- A tilting table which rotates about the X- or Y-axis (A- or B-rotation, secondary) is mounted on a rotary table which rotates about the Z-axis (C-rotation, primary), with the workpiece on the tilting table.

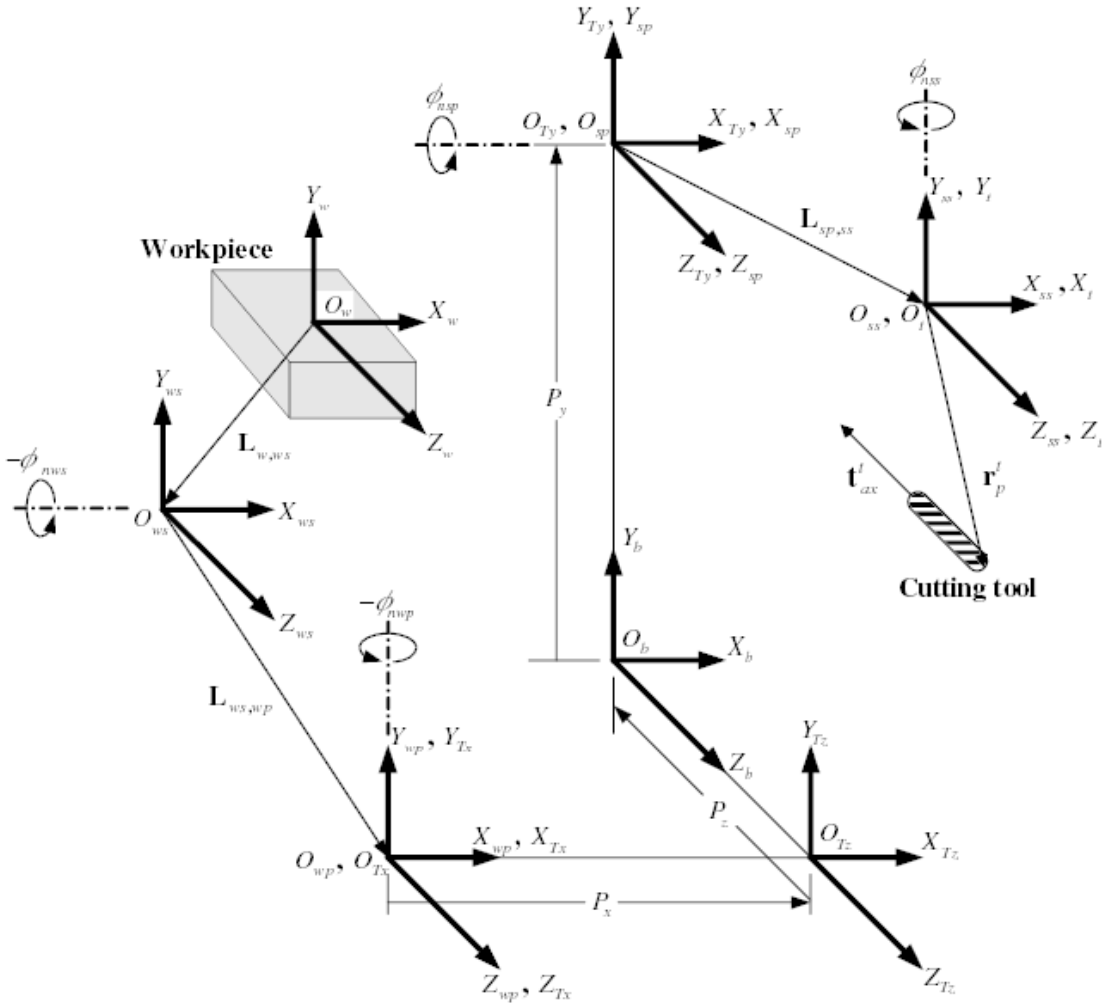



Figure 9.3: General configuration and coordinate systems

A multi-axis machine can be considered to consist of a series of links connected by joints. By embedding a coordinate frame in each link of the machine and using homogeneous transformations, we can describe the relative position and orientation between these coordinate frames

We need to describe a relationship between the workpiece coordinate system and the tool coordinate system. This can be defined by a transformation matrix ${}^{w}A_t$, which can be found by subsequent transformations between the different structural elements or links of the machine, each with its own defined coordinate system. In general such a transformation may look as follows:

$$ {}^{w}A_t = {}^{w}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \cdots {}^{n}A_t \qquad (4) $$

where each matrix ${}^{i-1}A_j$ is a translation matrix $T$ or a rotation matrix $R$ of the form (2,3).

Matrix multiplication is a simple process in which the elements of each row of the lefthand matrix A is multiplied by the elements of each column of the righthand matrix B and summed to obtain an element in the result matrix C, ie.

$$ C_{ij} = \sum_{k=1,n}^{n} A_{ik} B_{kj}; \quad i = 1, n; \quad j = 1, n $$

In Fig. 2 a generic configuration with coordinate systems is shown [4]. It includes table rotary/tilting axes as well as spindle rotary/tilting axes. Only two of the rotary axes are actually used in a machine tool.

First we will develop the transformations for the first type of configuration mentioned above, ie. a table tilting/rotary (trt) type with no rotating axis offsets. We may give it the name xyzac-trt configuration.

We also develop the transformations for the same type (xyzac-trt), but with rotating axis offsets.

Then we develop the transformations for a xyzbc-trt configuration with rotating axis offsets.

#### 9.3.5.1 Transformations for a xyzac-trt machine tool with work offsets

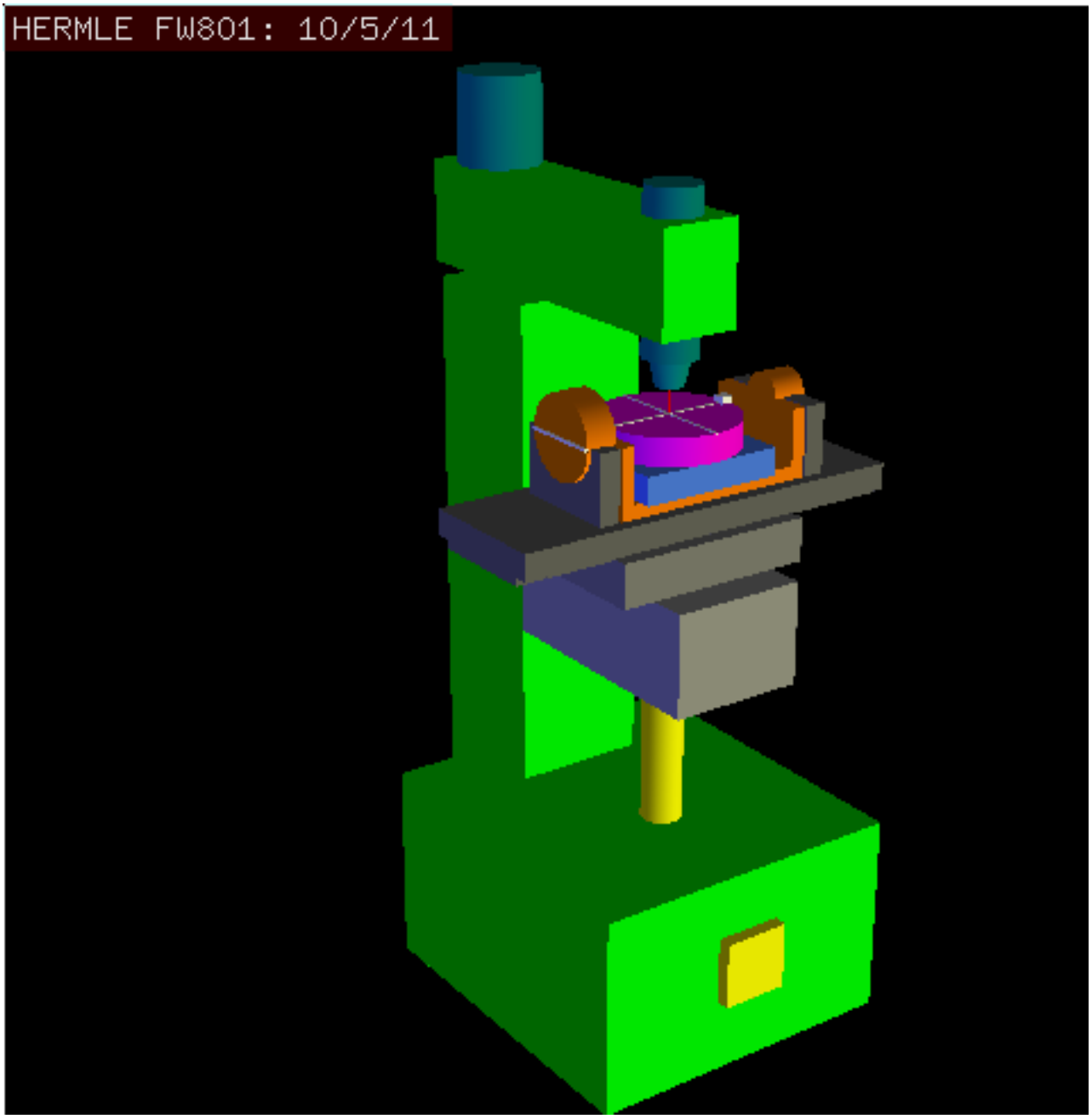



Figure 9.4: vismach model of xyzac-trt with coincident rotation axes

We deal here with a simplified configuration in which the tilting axis and rotary axis intersects at a point called the pivot point as shown in Fig. 4. therefore the two coordinate systems $O_{ws}$ and $O_{wp}$ of Fig. 2 are coincident.

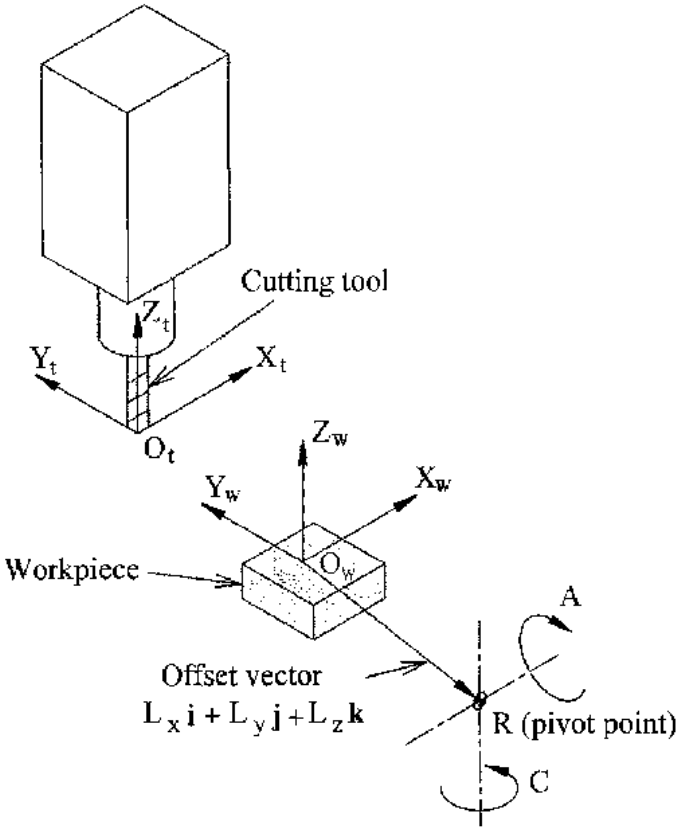



Figure 9.5: Table tilting/rotary configuration

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_t = {}^{w}A_C \cdot {}^{C}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (5) $$

with the matrices built up as follows:

$$ {}^{w}A_C = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{C}A_A = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

In these equations $L_x$, $L_y$, $L_z$ defines the offsets of the pivot point of the two rotary axes A and C relative to the workpiece coordinate system origin. Furthermore, $P_x$, $P_y$, $P_z$ are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is at the intersection of the two rotary axes. The signs of the $S_A$ and $S_C$ terms are different to those in [2,3] since there the table rotations are negative relative to the workpiece coordinate axes (note that sin(-theta) = -sin(theta), cos(-theta) = cos(theta)).

When multiplied in accordance with (5), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ 0 & -S_A & C_A & -S_A P_y + C_A P_z + L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

We can now equate the third column of this matrix with our given tool orientation vector K, ie.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (9) $$

From these equations we can solve for the rotation angles $theta_A$, $theta_C$. From the third row we find:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (10) $$

and by dividing the first row by the second row we find:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (11) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (8) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ -S_A P_y + C_A P_z + L_z \\ 1 \end{bmatrix} \qquad (12) $$

The vector on the right hand side can also be written as the product of a matrix and a vector resulting in:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & L_x \\ -S_C & C_C C_A & C_C S_A & L_y \\ 0 & -S_A & C_A & L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (13) $$

This can be expanded to give

$$ \begin{aligned} Q_x &= C_c P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ Q_y &= -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ Q_z &= -S_A P_y + C_A P_z + L_z \end{aligned} \qquad (14) $$

which is the *forward transformation* of the kinematics.

We can solve for P from equation (13) as $P = ({}^{Q}A_{P})^{-1} * Q$. Noting that the square matrix is a homogeneous 4x4 matrix containing a rotation matrix R and translation vector q, for which the inverse can be written as:

$$ {}^{q}A_{p} = \begin{bmatrix} R & q \\ 0 & 1 \end{bmatrix} \qquad ({}^{Q}A_{P})^{-1} = \begin{bmatrix} R^{T} & -R^{T}q \\ 0 & 1 \end{bmatrix} \qquad (15) $$

where R^T is the transpose of R (rows and columns swappped). We therefore obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & -C_C L_x + S_C L_y \\ S_C C_A & C_C C_A & -S_A & -S_C C_A L_x - C_C C_A L_y + S_A L_z \\ S_C S_A & C_C S_A & C_A & -S_C S_A L_x - C_C S_A L_y - C_A L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (16) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17) $$

#### 9.3.5.2 Transformations for a xyzac-trt machine with rotary axis offsets

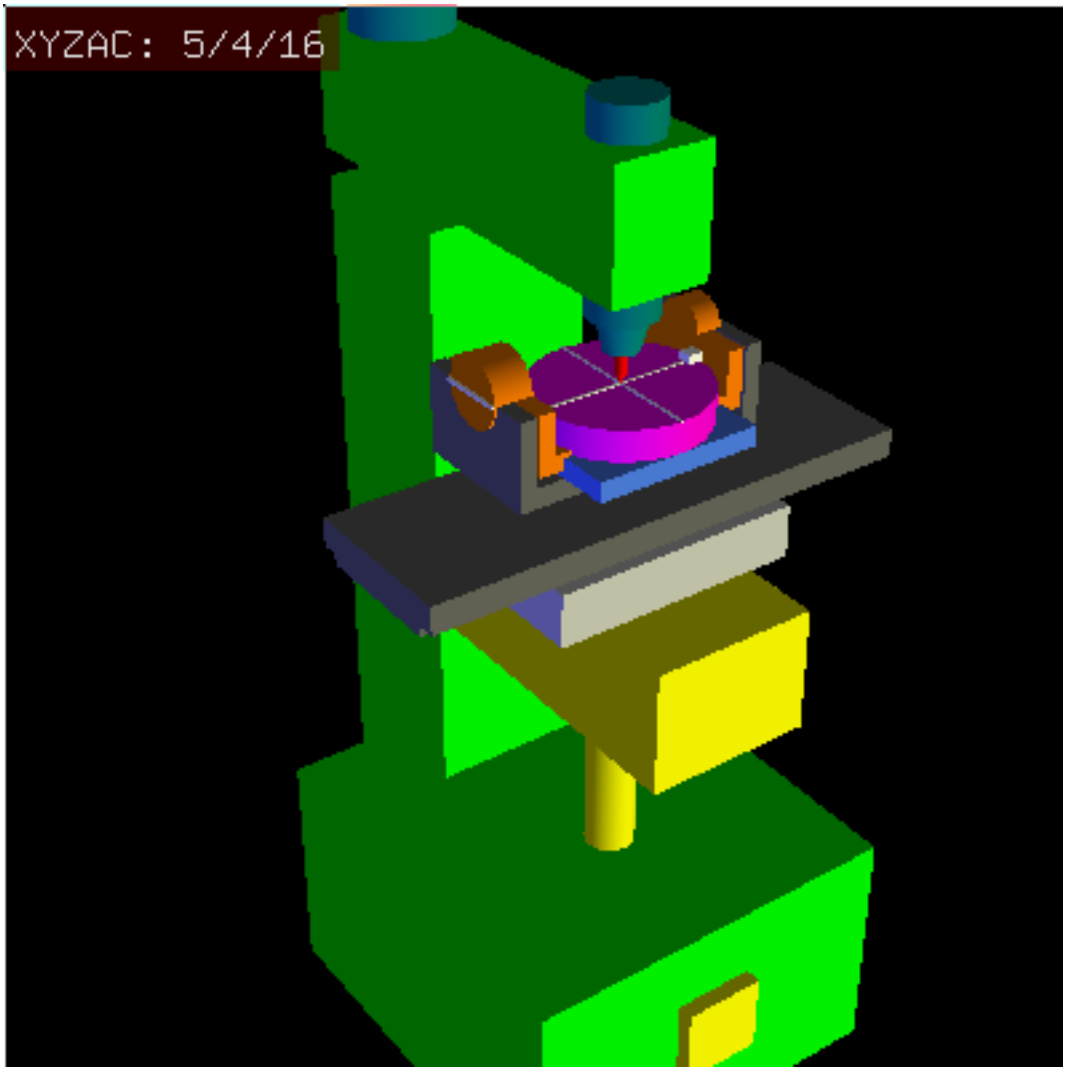



Figure 9.6: vismach model of xyzac-trt with rotational axis offsets (positive)

We deal here with a extended configuration in which the tilting axis and rotary axis do not intersect at a point but have an offset $D_y$. Furthermore, there is also an z-offset between the two coordinate systems $O_{ws}$ and $O_{wp}$ of Fig. 2, called $D_z$. A vismach model is shown in Fig. 5 and the offsets are shown in Fig. 6 (positive offsets in this example). To simplify the configuration, the offsets $L_x$, $L_y$, $L_z$ of the previous case are not included. They are probably not necessary if one uses the G54 offsets in LinuxCNC by means of the ”touch of” facility.

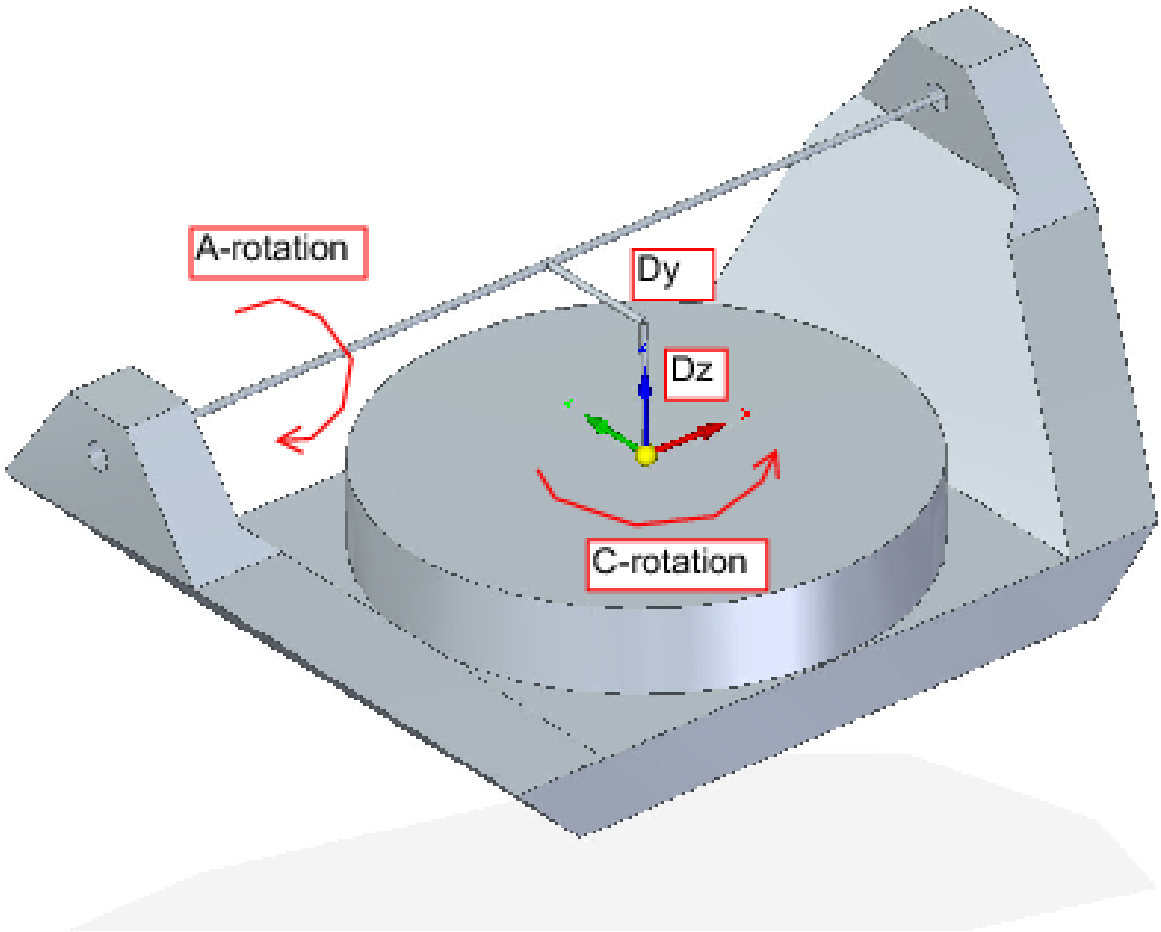



Figure 9.7: Table tilting/rotary xyzac-trt configuration, with axis offsets

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_{t} = {}^{w}A_{O} \cdot {}^{O}A_{A} \cdot {}^{A}A_{P} \cdot {}^{P}A_{t} \qquad (18) $$

with the matrices built up as follows:

$$ {}^{w}A_{O} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19) $$

$$ {}^{A}A_{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y - D_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20) $$

In these equations $D_y$, $D_z$ defines the offsets of the pivot point of the rotary axes A relative to the workpiece coordinate system origin. Furthermore, $P_x$, $P_y$, $P_z$ are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is on the A rotary axis.

When multiplied in accordance with (18), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ 0 & -S_A & C_A & -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21) $$

We can now equate the third column of this matrix with our given tool orientation vector K, ie.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (22) $$

From these equations we can solve for the rotation angles $theta_A$, $theta_C$. From the third row we find:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (23) $$

and by dividing the second row by the first row we find:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (24) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (21) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A (P_y - D_y) + S_C S_A (P_z - D_z) + S_C D_y \\ -S_C P_x + C_C C_A (P_y - D_y) + C_C S_A (P_z - D_z) + C_C D_y \\ -S_A (P_y - D_y) + C_A (P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (25) $$

The vector on the right hand side can also be written as the product of a matrix and a vector resulting in:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & -S_c C_A D_y - S_C S_A D_z + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -C_C C_A D_y - C_C S_A D_z + C_C D_y \\ 0 & -S_A & C_A & S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (26) $$

which is the *forward transformation* of the kinematics.

We can solve for P from equation (25) as $P = ({}^{Q}A_P)^{-1} * Q$ using (15) as before. We thereby obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & 0 \\ S_C C_A & C_C C_A & -S_A & -C_A D_y + S_A D_z + D_y \\ S_C S_A & C_S S_A & C_A & -S_A D_y - C_A D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (27) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28) $$

#### 9.3.5.3 Transformations for a xyzbc-trt machine with rotary axis offsets

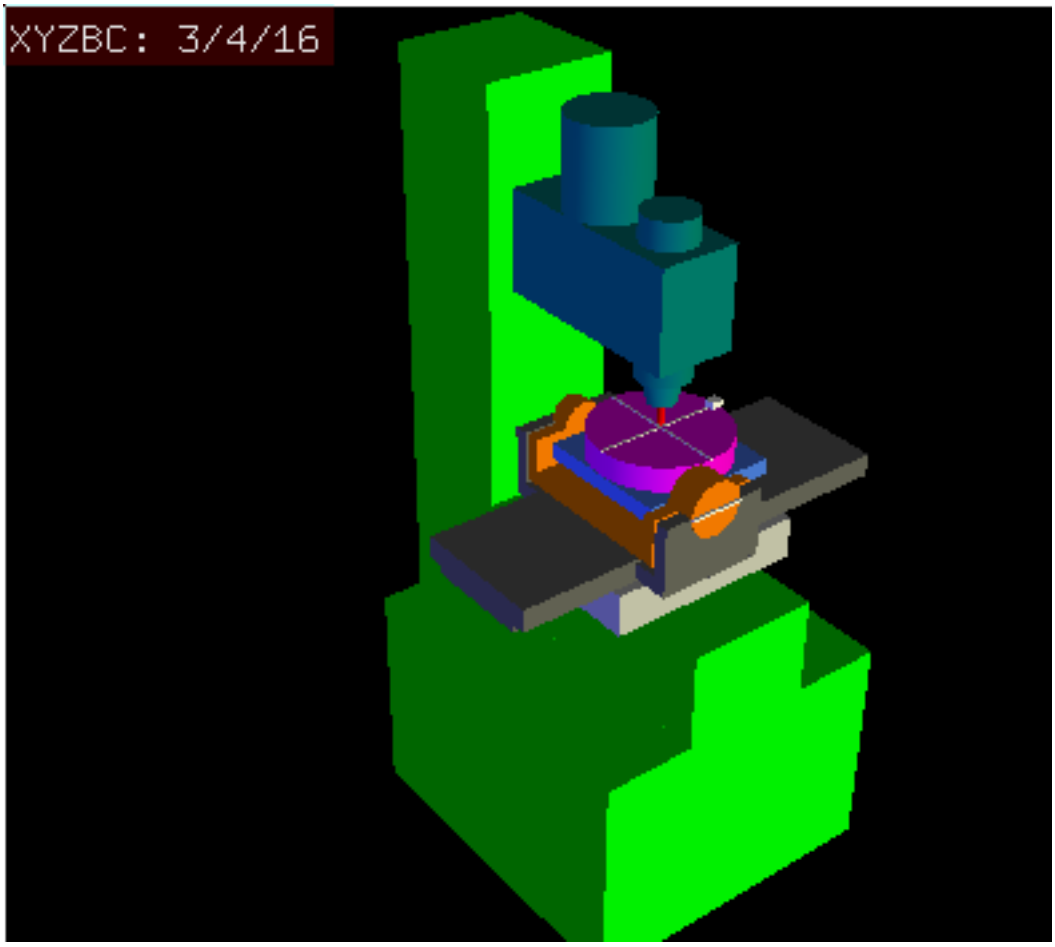



Figure 9.8: vismach model of xyzbc-trt with rotational axis offsets (negative)

We deal here again with a extended configuration in which the tilting axis (about the y-axis) and rotary axis do not intersect at a point but have an offset $D_x$. Furthermore, there is also an z-offset between the two coordinate systems $O_{ws}$ and $O_{wp}$ of Fig. 2, called $D_z$. A vismach model is shown in Fig. 7 (negative offsets in this example) and the positive offsets are shown in Fig. 8.

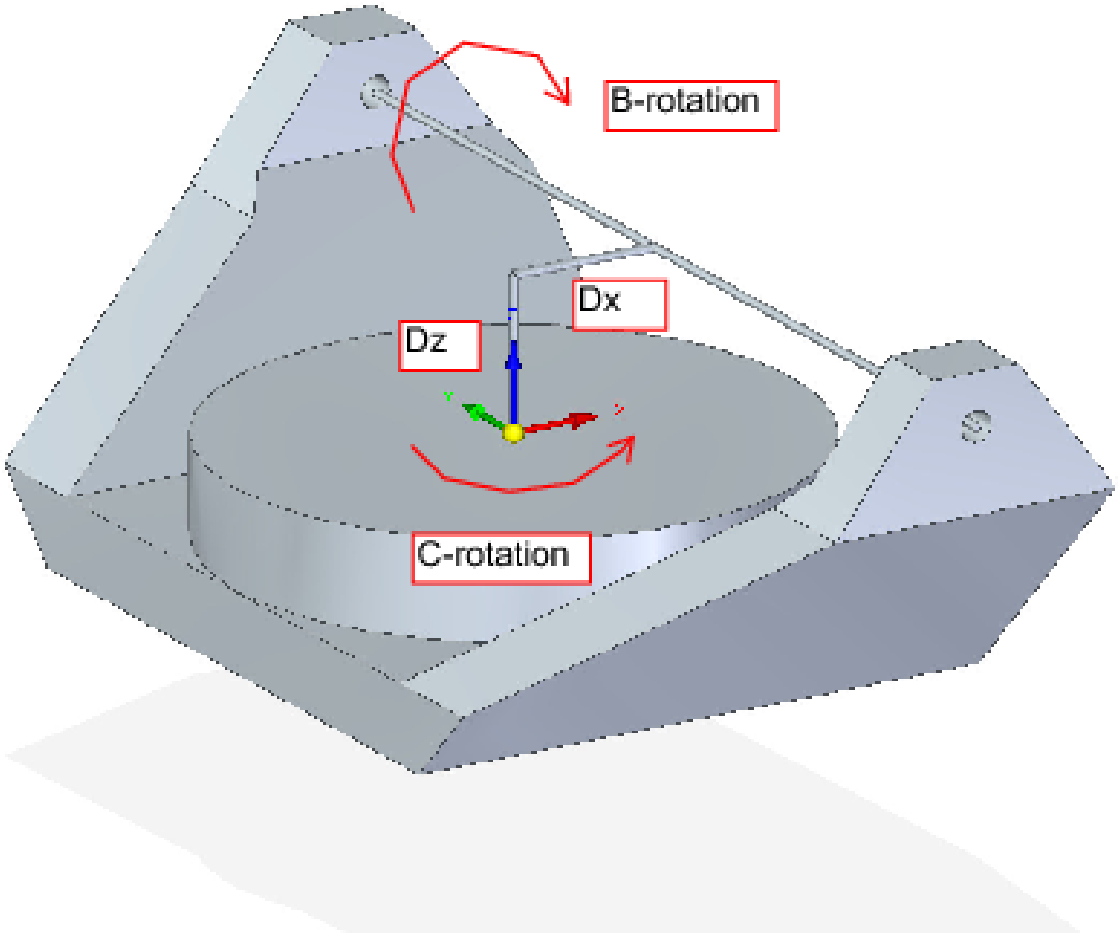



Figure 9.9: Table tilting/rotary xyzbc-trt configuration, with axis offsets

The transformation can be defined by the sequential multiplication of the matrices:

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_B \cdot {}^{B}A_P \cdot {}^{P}A_t \qquad (29) $$

with the matrices built up as follows:

$$ {}^{w}A_O = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_B = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (30) $$

$$ {}^{B}A_P = \begin{bmatrix} C_B & 0 & -S_B & 0 \\ 0 & 1 & 0 & 0 \\ S_B & 0 & C_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x - D_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (31) $$

In these equations $D_x$, $D_z$ defines the offsets of the pivot point of the rotary axes B relative to the workpiece coordinate system origin. Furthermore, $P_x$, $P_y$, $P_z$ are the relative distances of the pivot point to the cutter tip position, which can also be called the ”joint coordinates” of the pivot point. The pivot point is on the B rotary axis.

When multiplied in accordance with (29), we obtain:

$$ {}^{w}A_t = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B & C_C & S_C S_B & -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B & 0 & C_B & S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (32) $$

We can now equate the third column of this matrix with our given tool orientation vector K, i.e.:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} -C_C S_B \\ S_C S_B \\ C_B \\ 0 \end{bmatrix} \qquad (33) $$

From these equations we can solve for the rotation angles theta$_B$, theta$_C$. From the third row we find:

$$ \theta_B = cos^{-1}(K_z) \qquad (0 < \theta_B < \pi) \qquad (34) $$

and by dividing the second row by the first row we find:

$$ \theta_C = tan2^{-1}(K_y, K_x) \qquad (-\pi < \theta_C < \pi) \qquad (35) $$

These relationships are typically used in the CAM post-processor to convert the tool orientation vectors to rotation angles.

Equating the last column of (32) with the tool position vector Q, we can write:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (36) $$

The vector on the right hand side can also be written as the product of a matrix and a vector resulting in:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & -C_c C_B D_x + C_C S_B D_z + C_C D_x \\ -S_C C_B & C_C & S_C S_B & S_C C_B D_x - S_C S_B D_z - S_C D_x \\ S_B & 0 & C_B & -S_B D_x - C_B D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (37) $$

which is the *forward transformation* of the kinematics.

We can solve for P from equation (37) as $P = ({}^{Q}A_P)^{-1} * Q$.

With the same approach as before, we obtain:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & -S_C C_B & S_B & -C_B D_x - S_B D_z + D_x \\ S_C & C_C & 0 & 0 \\ -C_C S_B & S_C S_B & C_B & S_B D_x - C_B D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (38) $$

The desired equations for the *inverse transformation* of the kinematics thus can be written as:

$$ \begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \qquad (39) $$

## 9.3.6 Table Rotary/Tilting Examples

LinuxCNC includes kinematics modules for the *xyzac-trt* and *xyzbc-trt* topologies described in the mathematics detailed above. For interested users, the source code is available in the git tree in the *src/emc/kinematics/* directory.

Example xyzac-trt and xyzbc-trt simulation configurations are located in the Sample Configurations (*configs/sim/axis/vismach/5axis/table-rotary-tilting/*) directory.

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 Simulation Models

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
```

and connections are made using HAL commands like:

```
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 Tool-Length Compensation

In order to use tools from a tool table sequentially with tool-length compensation applied automatically, a further Z-offset is required. For a tool that is longer than the ”master” tool, which typically has a tool length of zero, LinuxCNC has a variable called ”motion.tooloffset.z”. If this variable is passed on to the kinematic component (and vismach python script), then the necessary additional Z-offset for a new tool can be accounted for by adding the component statement, for example:

$$D_z = D_z + \text{tool-offset}$$

The required HAL connection (for xyzac-trt) is:

```
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
```

where:

```
:tool-offset ---------------- signal name
motion.tooloffset.z --------- output HAL pin from LinuxCNC motion module
xyzac-trt-kins.tool-offset -- input  HAL pin to xyzac-trt-kins
```

## 9.3.7 Custom Kinematics Components

LinuxCNC implements kinematics using a HAL component that is loaded at startup of LinuxCNC. The most common kinematics module, *trivkins*, implements identity (trivial) kinematics where there is a one-to-one correspondence between an axis coordinate letter and a motor joint. Additional kinematics modules for more complex systems (including *xyzac-trt* and *xyzbc-trt* described above) are available.

See the kins manpage (**\$ man kins**) for brief descriptions of the available kinematics modules.

The kinematics modules provided by LinuxCNC are typically written in the C-language. Since a standard structure is used, creation of a custom kinematics module is facilitated by copying an existing source file to a user file with a new name, modifying it, and then installing.

Installation is done using halcompile:

```
sudo halcompile --install kinsname.c
```

where ”kinsname” is the name you give to your component. The sudo prefix is required to install it and you will be asked for your root password. See the halcompile man page for more information (**\$ 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
```

is replaced by

```
[KINS]
KINEMATICS = kinsname
```

where ”kinsname” is the name of your kins program. Additional HAL pins may be created by the module for variable configuration items such as the $D_x$, $D_y$, $D_z$, tool-offset used in the xyzac-trt kinematics module. These pins can be connected to a signal for dynamic control or set once with HAL connections like:

```
# set offset parameters
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
setp xyzac-trt-kins.y-offset 0
setp xyzac-trt-kins.z-offset 20
```

### 9.3.8 Figures

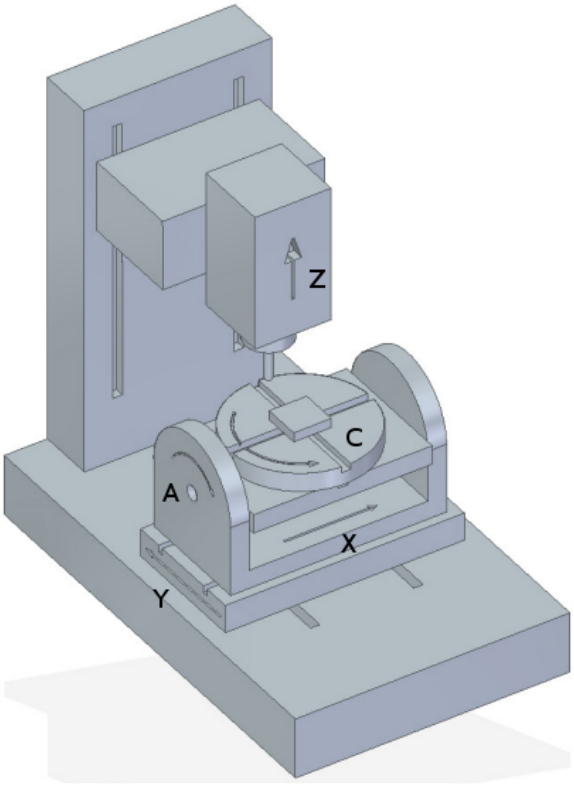



Figure 9.10: Table tilting/rotating configuration

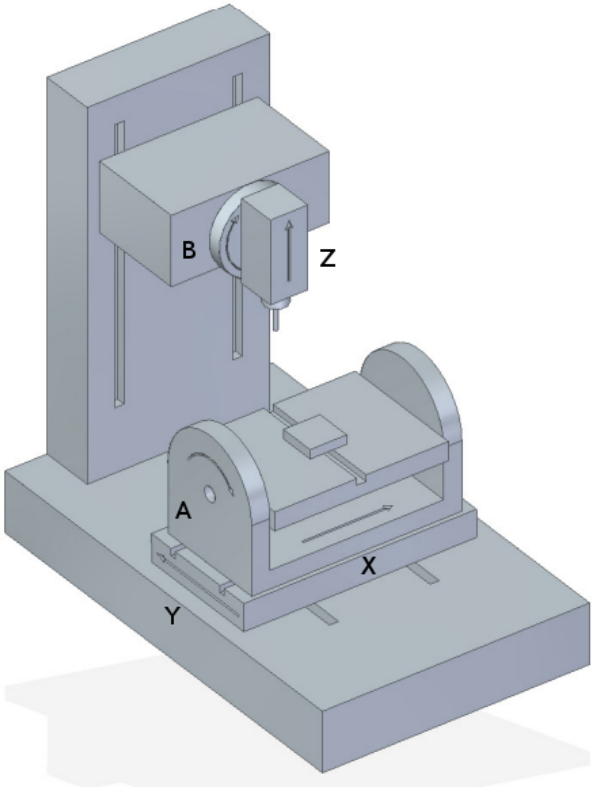



Figure 9.11: Spindle/table tilting configuration

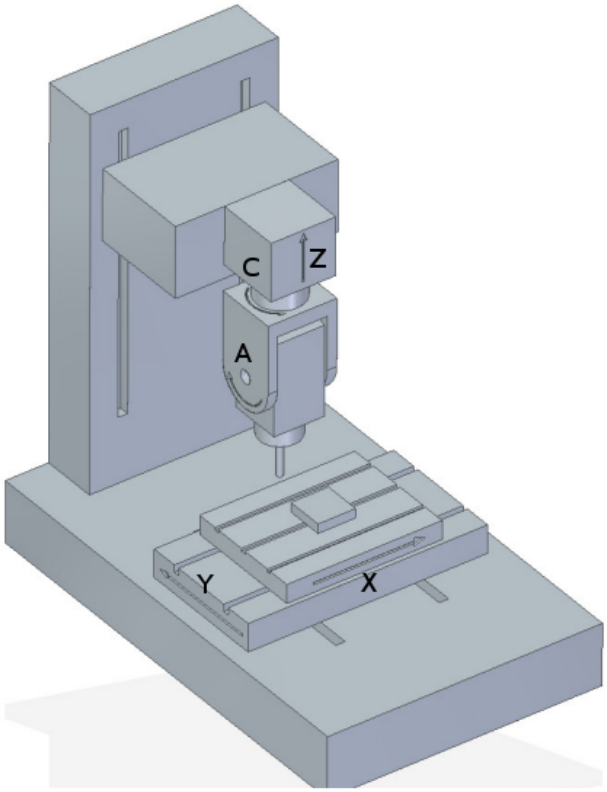



Figure 9.12: Spindle tilting/rotary configuration

### 9.3.9 REFERENCES

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 Switchable Kinematics (switchkins)

### 9.4.1 Введение

A number of kinematics modules support the switching of kinematics calculations. These modules support a default kinematics method (type0), a second built-in method (type1), and (optionally) a user-provided kinematics method (type2). Identity kinematics are typically used for the type1 method.

The switchkins functionality can be used for machines where post-homing joint control is needed during setup or to avoid movement near singularities from G-code. Such machines use specific kinematics

calculations for most operations but can be switched to identity kinematics for control of individual joints after homing.

The kinematics type is selected by a motion module HAL pin that can be updated from a G-code program or by interactive MDI commands. The halui provisions for activating MDI commands can be used to allow buttons to select the kinematics type using hardware controls or a virtual panel (PyVCP, GladeVCP, etc.).

When a kinematics type is changed, the G-code must also issue commands to **force synchronization** of the interpreter and motion parts of LinuxCNC. Typically, a HAL pin *read* command (M66 E0 L0) is used immediately after altering the controlling HAL pin to force synchronization.

## 9.4.2 Switchable Kinematic Modules

The following kinematics modules support switchable kinematics:

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 назначения идентифицирующей буквы

При использовании типа кинематики **identity** параметр модуля *координаты* можно использовать для назначения букв сочленениям в произвольном порядке из набора разрешенных букв координат. Примеры:

```
[KINS]
JOINTS = 6

# conventional identity ordering: joint0==x, joint1==y, ...
KINEMATICS = genhexkins coordinates=xyzabc

# custom identity ordering: joint0==c, joint1==b, ...
# KINEMATICS = genhexkins coordinates=cbazyx
```

---

**Note**
Если параметр coordinates= опущен, по умолчанию буква-сочленение присваиваемых идентификаторов будут joint0==x,joint1=y,...

---

Назначения сочленений, предусмотренные для кинематики **identity** при использовании параметра координат, идентичны тем, которые предусмотрены для модуля trivkins. Однако дублирование букв осей для назначения нескольких сочленений букве координат обычно не применимо для последовательной или параллельной кинематики (например, genserkins, pumakins, genhexkins и т. д.), где нет простой взаимосвязи между сочленениями и координатами.

Дублирование букв координат осей поддерживается в модулях кинематики xyzac-trt-kins, xyzbc-trt-kins и 5axiskins (bridgemill). Типичным применением дублирующихся координат являются портальные станки, в которых для поперечной оси используются два двигателя (сочленения).

#### 9.4.2.2 Обратная совместимость

Переключаемая кинематика инициализируется с помощью motion.switchkins-type==0, реализующего одноименный метод кинематики. Если контакт типа motion.switchkins не подключен (как в устаревших конфигурациях), доступен только тип кинематики по умолчанию.

### 9.4.3 HAL Контакты

Переключение кинематики контролируется входным контактом HAL модуля движения **motion.switch type**. Значение вывода с плавающей запятой усекается до целого числа и используется для выбора одного из предоставленных типов кинематики. Нулевое значение запуска выбирает тип кинематики по умолчанию type0.

---

**Note**
Входной контакт типа motion.switchkins имеет тип числа с плавающей запятой, чтобы облегчить подключение к выходным контактам модуля движения, таким как motion.analog-out-0n, которые управляются стандартными M-кодами (обычно M68EnL0).

---

Выходные контакты HAL предназначены для информирования ГИП о текущем типе кинематики. Эти контакты также можно подключить к цифровым входам, которые считываются программами G-кода для включения или отключения поведения программы в соответствии с активным типом кинематики.

#### 9.4.3.1 Сводная информация о контактах HAL

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 Использование

#### 9.4.4.1 HAL Connections

Функциональность switchkins разрешается контактом **motion.switchkins-type**. Обычно этот контакт подключается к аналоговому выходному контакту, например motion.analog-out-03, поэтому его можно установить с помощью команд M68. Пример:

```
net :kinstype-select <= motion.analog-out-03
net :kinstype-select => motion.switchkins-type
```

#### 9.4.4.2 Команды G-/M-кода

Выбор типа кинематики осуществляется с помощью подобных последовательностей G-кода:

```
...
M68 E3 Q1 ;update analog-out-03 to select kinstype 1
M66 E0 L0 ;sync interp-motion
...
...       ;user G-code
...
M68 E3 Q0 ;update analog-out-03 to select kinstype 0
M66 E0 L0 ;sync interp-motion
...
```

**Note**
Команда M66 *wait-on-input* обновляет переменную #5399. Если текущее значение этой переменной необходимо для последующих целей, его следует скопировать в дополнительную переменную перед вызовом M66.

Эти последовательности команд G-кода обычно реализуются в подпрограммах G-кода как переназначе
M-коды или с помощью обычных сценариев M-кода.

Предлагаемые коды (используемые в конфигурациях sim):

Обычные пользовательские M-коды:

1. M128 Выбор kinstype 0 (кинематика по умолчанию при запуске)
2. M129 Выбирает kinstype 1 (обычно идентичная кинематика)
3. M130 Выбирает kinstype 2 (кинематика, предоставляемая пользователем)

Переназначенные M-коды:

1. M428 Выбор kinstype 0 (кинематика по умолчанию при запуске)
2. M429 Выберите kinstype 1 (обычно идентичная кинематика)
3. M430 Выбор kinstype 2 (кинематика, предоставляемая пользователем)

**Note**
Обычные пользовательские M-коды (в диапазоне от M100 до M199) находятся в модальной группе 10. Переназначенные M-коды (в диапазоне от M200 до M999) могут указывать модальную группу. Дополнительную информацию см. в документации по переназначению.

#### 9.4.4.3 Настройки ограничений INI-файла

При планировании траектории LinuxCNC используются ограничения на положение (мин, максимум), скорость и ускорение для каждой применимой буквы координат, указанной в INI-файле конфигурации
Пример буквы L (в наборе *XYZABCUVW*):

```
[AXIS_L]
MIN_LIMIT =
MAX_LIMIT =
MAX_VELOCITY =
MIN_ACCELERATION =
```

Указанные ограничения файла INI применяются к типу кинематики по умолчанию типа 0, который активируется при запуске. Эти ограничения могут **не** применяться при переключении на альтернатив кинематику. Однако, поскольку при переключении кинематики требуется синхронизация движения интерпретатора, контакты INI-HAL можно использовать для установки пределов для ожидающего типа кинематики.

---

**Note**
Контакты INI-HAL обычно не распознаются во время работы программы G-кода, если не выдана команда синхронизации (queue-buster). Дополнительную информацию см. на странице руководства milltask ($ man milltask).

---

Соответствующие контакты INI-HAL для номера сочленения (*N*):

```
ini.N.min_limit
ini.N.max_limit
ini.N.max_acceleration
ini.N.max_velocity
```

Соответствующие контакты INI-HAL для координаты оси (*L*):

```
ini.L.min_limit
ini.L.max_limit
ini.L.max_velocity
ini.L.max_acceleration
```

---

**Note**
В общем, не существует фиксированных сопоставлений между номерами сочленений и буквами координат осей. Для некоторых модулей кинематики могут существовать специальные сопоставления, особенно для тех, которые реализуют идентичную кинематику (trivkins). Дополнительную информацию смотрите на странице руководства kins ($ man kins).

---

Предоставленный пользователем M-код может изменить любые или все пределы координат оси перед изменением контакта типа motion.switchkins и синхронизацией интерпретатора и движущихся частей LinuxCNC. Например, скрипт bash, вызывающий halcmd, может быть *hardcoded* для установки любого количества контактов HAL:

```
#!/bin/bash
halcmd -f <<EOF
setp ini.x.min_limit -100
setp ini.x.max_limit  100
# ... repeat for other limit parameters
EOF
```

Подобные скрипты могут быть вызваны как пользовательский M-код и использованы **перед** M-кодом переключения kins, который обновляет контакт HAL типа `motion.switchkins-type` и вызывает принудительную синхронизацию interp-motion. Обычно для каждого kinstype (0,1,2) используются отдельные скрипты.

Когда identity kinematics предоставляется в качестве средства управления отдельными сочленениями, может быть удобно устанавливать или восстанавливать пределы, указанные в системном INI-файле. Например, робот стартует со сложной (non-identity) кинематикой (type0) после возврата в исходное положение. Система настроена так, что ее можно переключить на identity kinematics (type1) для манипулирования отдельными сочленениями с помощью обычных букв из набора

*XYZABCUVW*. Настройки INI-файла ([AXIS_L]) **не** применимы при работе с identity (type1) kinematics. Для решения этого варианта использования пользовательские скрипты M-кода могут быть разработаны следующим образом:

**M129** (Переключение на identity type1)

1. прочитать и разобрать INI-файл
2. HAL: setp ограничительные контакты INI-HAL для каждой буквы оси ([AXIS_L]) в соответствии с настройкой ([JOINT_N]) *identity-referenced* номера сочленения INI файла
3. HAL: `setp motion.switchkins-type 1`
4. MDI: выполните синхронизирующий G-код (M66E0L0)

**M128** (восстановить по умолчанию robot kinematics type 0)

1. прочитать и разобрать INI-файл
2. HAL: setp ограничительные контакты INI-HAL для каждой буквы оси ([AXIS_L]) в соответствии с соответствующей настройкой INI-файла ([AXIS_L])
3. HAL: `setp motion.switchkins-type 0`
4. MDI: выполните синхронизирующий G-код (M66E0L0)

---

**Note**
Конфигурации моделирования vismach для робота Puma демонстрируют сценарии M-кода (M128, M129, M130) для этого примера использования.

---

#### 9.4.4.4 Соображения смещения системы координат

Как и настройки предельных значений INI-файла, смещения системы координат (G92, G10L2, G10L20, G43 и т. д.) обычно применимы только для type 0 типа kinematics при запуске по умолчанию. При переключении типов кинематики может быть **важно** либо сбросить все смещения перед переключением, либо обновить смещения в соответствии с требованиями конкретной системы.

#### 9.4.4.5 Внешние соображения смещения

External offsets (set to an axis (*L*) via axis.*L*.eoffset-request) are preserved during kinematics switches. When an offset is active on an axis before the switch (visible in axis.*L*.eoffset), the trajectory planner maintains that same offset after the switch, similar to how it maintains the commanded position from a G-code. This ensures consistent machine behavior regardless of the active kinematics.

If maintaining the offset will be an issue due to axis limit changes or other concerns, be sure to clear and possibly disable the eoffset before making a kinematics switch.

### 9.4.5 Конфигурации моделирования

Конфигурации моделирования (не требующие аппаратного обеспечения) снабжены иллюстративными дисплеями vismach в подкаталогах configs/sim/axis/vismach/ .

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 Условия пользовательской кинематики

Пользовательскую кинематику можно закодировать и протестировать в сборках Run-In-Place (RIP). В раздаче имеется файл шаблона src/emc/kinematics/userkfuncs.c. Этот файл можно скопировать в пользовательский каталог и отредактировать для предоставления пользовательской кинематики с kinstype==2.

Пользовательский файл кинематики можно скомпилировать из исходных расположений вне дерева для реализаций rt-preempt или путем замены файла шаблона в дереве (src/emc/kinematics/userkfuncs.c) для систем rtai.

Preempt-rt make пример:

```
$ userkfuncs=/home/myname/kins/mykins.c make && sudo make setuid
```

### 9.4.7 Предупреждения

Неожиданное поведение может возникнуть, если программа G-кода случайно запускается с несовместимым типом кинематики. Нежелательное поведение можно обойти в программах G-кода следующим образом:

1. Подключение соответствующих контактов HAL kinstype.is.N к контактам цифрового входа (например, motion.digital-in-0m).
2. Чтение контакта цифрового входа (M66 E0 Pm) в начале программы G-кода
3. Прерывание (M2) программы G-кода с сообщением (DEBUG, problem_message), если kinstype не подходит.

При интерактивном использовании возможностей медленной подачи или команд MDI оператору требуется соблюдать осторожность. В ГИП должны быть индикаторы для отображения текущего типа кинематики.

---

**Note**
Переключаемая кинематика может привести к существенным эксплуатационным изменениям, требующим тщательного проектирования, тестирования и подготовки к развертыванию. Управление смещениями координат, компенсацией инструмента и ограничениями файлов INI может потребовать сложных и нестандартных рабочих протоколов.

---

### 9.4.8 Примечания к коду

Кинематические модули, обеспечивающие функциональность switchkins, связаны с объектом switchkins.o (switchkins.c), который предоставляет *main* программу модуля (rtapi_app_main()) и связанные с ней функции. Эта *main* программа считывает (необязательные) параметры командной строки модуля (coordinates, sparm) и передает их в функцию switchkinsSetup(), предоставляемую модулем.

Функция switchkinsSetup() идентифицирует kinstype специфичные процедуры настройки и функции для прямого и обратного расчета для каждого kinstype (0,1,2) и устанавливает ряд настроек конфигурации.

После вызова switchkinsSetup() rtapi_app_main() проверяет предоставленные параметры, создает компонент HAL, а затем вызывает процедуру установки, определенную для каждого типа kinstype (0,1,2).

Каждая kinstype (0,1,2) процедура настройки может (опционально) создавать контакты HAL и устанавливать для них значения по умолчанию. Когда все процедуры установки завершаются, rtapi_app_main() устанавливает hal_ready(), чтобы компонент завершил создание модуля.

## 9.5 PID Настройка

### 9.5.1 ПИД-контроллер

Пропорционально-интегрально-дифференциальный контроллер (ПИД-контроллер) является распрост компонентом контура обратной связи в промышленных системах управления. footnote: [Этот подраздел взят из гораздо более обширной статьи, найденной по адресу https://en.wikipedia.org/-wiki/PID_controller]

Контроллер сравнивает измеренное значение процесса (обычно промышленного процесса) с опорным значением уставки. Разница (или сигнал *ошибки*) затем используется для расчета нового значения управляемого входного сигнала процесса, который возвращает измеренное значение процесса к желаемому заданному значению.

В отличие от более простых алгоритмов управления, ПИД-контроллер может регулировать выходные данные процесса на основе истории и скорости изменения сигнала ошибки, что обеспечивает более точное и стабильное управление. (Можно показать математически, что контур ПИД-контроллера обеспечит точное и стабильное управление в тех случаях, когда простое пропорциональн управление либо будет иметь установившуюся ошибку, либо приведет к колебаниям процесса).

#### 9.5.1.1 Основы контура управления

Интуитивно понятно, что контур ПИД пытается автоматизировать то, что сделал бы умный оператор с манометром и ручкой управления. Оператор будет считывать показания датчика, показывающего выходное измерение процесса, и использовать ручку для регулировки входных данных процесса (*действие*) до тех пор, пока выходное измерение процесса не стабилизируется на желаемом значении на датчике.

В старой литературе по управлению этот процесс настройки называется действием *сброса*. Положени стрелки на манометре является *измерением*, *значением процесса* или *переменной процесса*. Требуемое значение на манометре называется *уставка* (также называемая *заданным значением*). Разница между стрелкой манометра и заданным значением является *ошибка*.

Контур управления состоит из трех частей:

1. Измерение датчиком, подключенным к процессу (например, энкодером),

2. Решение в элементе контроллера,
3. Действие через устройство вывода, такое как двигатель.

Когда контроллер считывает показания датчика, он вычитает это измерение из *заданного значения*, чтобы определить *ошибку*. Затем он использует ошибку для расчета поправки к входной переменной процесса (*действие*), чтобы эта поправка устранила ошибку из выходного измерения процесса.

В контуре ПИД-регулятора коррекция рассчитывается на основе ошибки тремя способами: напрямую уравновешивается текущая ошибка (Пропорциональная), время, в течение которого ошибка продолжает оставаться неисправленной (Интегральная), и прогнозируется будущая ошибка на основе скорости изменения ошибки во времени (Дифференциальная).

ПИД-контроллер можно использовать для управления любой измеряемой переменной, на которую можно влиять путем манипулирования какой-либо другой переменной процесса. Например, его можно использовать для контроля температуры, давления, скорости потока, химического состава, скорости или других переменных. Автомобильный круиз-контроль является примером процесса за пределами промышленности, в котором используется грубое ПИД-регулирование.

Некоторые системы управления объединяют ПИД-контроллеры в каскады или сети. То есть *главный* элемент управления генерирует сигналы, используемые *подчиненными* контроллерами. Одной из распространенных ситуаций является управление двигателем: часто требуется, чтобы двигатель имел контролируемую скорость, а *ведомый* контроллер (часто встроенный в преобразовател частоты) напрямую управляет скоростью на основе пропорционального входного сигнала. Этот *ведомый* вход подается на выход *главного* контроллера, который управляет на основе связанной переменной.

#### 9.5.1.2 Теория

*ПИД* назван в честь трех корректирующих вычислений, которые увеличивают и корректируют контролируемую величину. Эти добавления на самом деле представляют собой *вычитания* ошибки, поскольку пропорции обычно отрицательны:

**Пропорциональная** Чтобы обработать текущю, ошибка умножается на (отрицательную) константу P (для *пропорциональной*) и добавляется к (вычитая ошибку из) контролируемой величины. P действует только в диапазоне, в котором выходной сигнал контроллера пропорционален ошибке системы. Обратите внимание, что когда ошибка равна нулю, выход пропорционального регулятора равен нулю.

**Интегральная** Чтобы оценить предшествующие данные, ошибка интегрируется (складывается) в течение определенного периода времени, а затем умножается на (отрицательную) константу И (что дает среднее значение) и добавляется к контролируемой величине (вычитается из нее). И усредняет измеренную ошибку, чтобы найти среднюю ошибку выходного сигнала процесса от заданного значения. Простая пропорциональная система либо колеблется, перемещаясь вперед и назад вокруг заданного значения, поскольку нет ничего, что могло бы устранить ошибку при ее превышении, либо колеблется и/или стабилизируется на слишком низком или слишком высоком значении. Добавляя отрицательную долю (т. е. вычитая часть) средней ошибки из входных данных процесса, средняя разница между выходными данными процесса и заданным значением всегда уменьшается. Таким образом, в конечном итоге выходной сигнал хорошо настроенного контура ПИД стабилизируется на заданном уровне.

**Дифференциальная** Чтобы оценить будущее, первая производная (наклон ошибки) с течением времени вычисляется и умножается на другую (отрицательную) константу Д, а также добавляется к контролируемой величине (вычитая ошибку). Дифференциальный член контролирует реакцию на изменение в системе. Чем больше производный член, тем быстрее контроллер реагирует на изменения выходных данных процесса.

С технической точки зрения контур ПИД можно охарактеризовать как фильтр, применяемый в сложной системе частотной области. Это полезно для того, чтобы рассчитать, действительно ли оно достигнет стабильного значения. Если значения выбраны неправильно, контролируемый

вход процесса может колебаться, а выход процесса может никогда не оставаться на заданном значении.

#### 9.5.1.3 Настройка контура

*Настройка* контура управления представляет собой настройку его параметров управления (усиление/ пропорциональности, интегральное усиление/сброс, производное усиление/скорость) до оптимальных значений для желаемого отклика управления. Оптимальное поведение при изменении процесса или заданного значения варьируется в зависимости от применения. Некоторые процессы не должны допускать превышения переменной процесса от заданного значения. Другие процессы должны минимизировать энергию, затрачиваемую на достижение новой заданной точки. Обычно требуется стабильность реакции, и процесс не должен колебаться при любой комбинации условий процесса и заданных значений.

Настройка контуров усложняется из-за времени отклика процесса; для того, чтобы изменение заданного значения дало стабильный эффект, может потребоваться несколько минут или часов. Некоторые процессы имеют определенную степень нелинейности, поэтому параметры, которые хорошо работают в условиях полной нагрузки, не работают, когда процесс запускается с холостого хода. В этом разделе описаны некоторые традиционные ручные методы настройки контура.

Существует несколько методов настройки контура ПИД. Выбор метода будет во многом зависеть от того, можно ли перевести контур в автономный режим для настройки, а также от скорости реакции системы. Если систему можно перевести в автономный режим, лучший метод настройки часто включает в себя подвергание системы ступенчатому изменению входных данных, измерение выходного сигнала как функции времени и использование этой реакции для определения параметров управления.

**Простой метод** Если система должна оставаться в режиме онлайн, один из методов настройки заключается в том, чтобы сначала установить значения И и Д на ноль. Увеличивайте П до тех пор, пока выходной сигнал контура не начнет колебаться. Затем увеличивайте И до тех пор, пока колебания не прекратятся. Наконец, увеличивайте Д до тех пор, пока цикл не будет достаточно быстро достигать уставки. Быстрая настройка контура ПИД-регулятора обычно слегка выходит за пределы диапазона, чтобы быстрее достичь заданного значения; однако некоторые системы не допускают перерегулирования.

| **Parameter (Параметр)** | **Время нарастания** | **Перерегулирование** | **Время урегулирования** | **Устойчивая ошибка** |
|---|---|---|---|---|
| P | Снижать | Увеличивать | Небольшое изменение | Снижать |
| I | Снижать | Увеличивать | Увеличивать | Устранять |
| D | Небольшое изменение | Снижать | Снижать | Небольшое изменение |

Эффекты увеличения параметров

**Метод Циглера-Николса Другой метод настройки формально известен как** *Метод Зиглера-Николса*, представленный Джоном Г. Зиглером и Натаниэлем Б. Николсом в 1942 году, сноска: [Зиглер, Дж. Г. и Николс, Н. Б. (1942), *Optimum Settings for Automatic Controllers*, Transactions of the ASME, link: https ://doi.org/10.1115/1.2899060[DOI 10.1115/1.2899060] и link: https://web.archive.org web/20170918055307/http://staff.guilan.ac.ir/staff/users/chaibakhsh /fckeditor_repo/file/documents/Optim архив].]. Все начинается так же, как и метод, описанный ранее: сначала установите усиление И и Д на ноль, а затем увеличьте усиление П и подвергайте контур внешнему вмешательству, например, постукиванию оси двигателя, чтобы заставить его выйти из равновесия по порядку. для определения критического усиления и периода колебаний до тех пор, пока выходной сигнал контура не начнет колебаться. Запишите критический коэффициент усиления ($K_c$) и период колебаний выходного сигнала ($P_c$). Затем отрегулируйте коэффициенты П, И и Д, как показано в таблице:

| Тип управления | P | I | D |
|---|---|---|---|
| P | $.5K_c$ | | |
| PI | $.45K_c$ | $P_c/1.2$ | |
| ПИД | $.6K_c$ | $P_c/2$ | $P_c/8$ |

**Заключительные шаги** После настройки оси проверьте следующую ошибку с помощью Halscope, чтобы убедиться, что она соответствует требованиям вашего станка. Дополнительную информацию о Halscope можно найти в руководстве пользователя HAL.

#### 9.5.1.4 Автоматическая настройка ПИД-регулятора

Начиная с LinuxCNC версии 2.9, компонент pid поддерживает автоматическую настройку с использова метода Relay [2]. Это замена уже удаленного и устаревшего компонента at_pid.

Компонент pid использует несколько констант для расчета выходного значения на основе текущего и желаемого состояния, наиболее важными из которых являются *Pgain*, *Igain*, *Dgain*, *bias*, *FF0*, *FF1*, *FF2*. и *FF3*. Все это должно иметь разумное значение, чтобы контроллер вел себя правильно.

Текущая реализация автоматической настройки реализует два разных алгоритма, выбираемых с помощью контакта tune-type. Когда тип настройки равен нулю, он влияет на *Pgain*, *Igain* и *Dgain*, в то время как *FF0*, *FF1* и *FF2* устанавливаются в ноль. Если tune-type равен 1, он влияет на *Pgain*, *Igain* and *FF1*, при этом *Dgain*, *FF0* и *FF2* устанавливаются в ноль. Примечание тип 1 требует установки масштабирования, чтобы выходные данные были в пользовательских единицах в секунду.

При автонастройке двигателя с tune-type 0 алгоритм будет генерировать прямоугольную волну, центрированную вокруг значения *bias* на выходном контакте ПИД-контроллера, перемещаясь от положительного экстремума к отрицательному экстремуму выходного диапазона. Это можно увидеть с помощью HAL Scope, предоставленного LinuxCNC. Для контроллера двигателя, принимающе +-10 В в качестве управляющего сигнала, это может на короткое время ускорить двигатель до полной скорости в одном направлении, прежде чем дать ему команду двигаться на полной скорости в противоположном направлении. Убедитесь, что по обе стороны от стартовой позиции достаточно места, и начните с низкого значения `tune-effort`, чтобы ограничить используемую скорость. Значение `tune-effort` определяет максимальное используемое значение `output`, поэтому, если `tune-effort` равно 1, значение `output` во время настройки изменится с 1 на -1. Другими словами, крайние точки волновой структуры контролируются контактом `tune-effort`. Использование слишком большого *tune-effort* может привести к перегрузке драйвера двигателя.

The number of cycles in the tune pattern is controlled by the *tune-cycles* pin. Of course, trying to change the direction of a physical object instantly (as in going directly from a positive voltage to the equivalent negative voltage in the motor controller case) do not change velocity instantly, and it take some time for the object to slow down and move in the opposite direction. This result in a more smooth wave form on the position pin, as the axis in question were vibrating back and forth. When the axis reached the target speed in the opposing direction, the auto tuner change direction again. After several of these changes, the average time delay between the ”peaks” and ”valleys” of this movement graph is used to calculate proposed values for `Pgain`, `Igain` and `Dgain`, and insert them into the HAL model to use by the pid controller. The auto tuned settings are not perfect, but might provide a good starting point for further parameter tuning.

FIXME: The author of these instructions have not tested automatic tuning with tune-type set to 1, so this approach remain to be documented.

Armed with this knowledge, it is time to look at how to do the tuning. Lets say the HAL configuration in question load the PID component for X, Y and Z like this, using named pin names instead of *count=3*:

```
loadrt pid names=pid.x,pid.y,pid.z
```

[2] Åström, Karl Johan and Hägglund, Tore (1984), *Automation paper Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins*, DOI 10.1016/0005-1098(84)90014-1

If the component had used *count=3* instead, all use of pid.x, pid,y and pid.z need to be changed to pid.1, pid.2 and pid.3 respectively. To start tuning the X axis, move the axis to the middle of its range, to make sure it do not hit anything when it start moving back and forth. You also want to extend the axis ferror limit (following error) to make LinuxCNC accept the higher position deviation during tuning. The sensible ferror limit depends on the machine and setup, but 1 inch or 20 mm might be useful starting points. Next, set the initial *tune-effort* to a low number in the output range, for example 1/100 of the maximum output, and slowly increase it to get more accurate tuning values. Assign 1 to the *tune-mode* value. Note, this will disable the pid controlling part and feed the *bias* value to the output pin, which can cause a lot of drift. It might be a good idea to tune the motor driver to ensure zero input voltage do not cause any motor rotation, or adjust the *bias* value for the same effect. Finally, after setting *tune-mode*, set *tune-start* to 1 to activate the auto tuning. If all go well, your axis will vibrate and move back and forth for a few seconds and when it is done, new values for Pgain, Igain and Dgain will be active. To test them, change *tune-mode* back to 0. Note that setting *tune-mode* back to zero might cause the axis to suddenly jerk as it bring the axis back to its commanded position, which it might have drifted away from during tuning. To summarize, these are the halcmd instructions you need to issue to do automatic tuning:

```
setp pid.x.tune-effort 0.1
setp pid.x.tune-mode 1
setp pid.x.tune-start 1
# wait for the tuning to complete
setp pid.x.tune-mode 0
```

A script to help doing the automatic tuning is provided in the LinuxCNC code repository as *scripts/run-auto-pid-tuner*. This will ensure the machine is powered on and ready to run, home all axes if it is not already done, check that the extra tuning pins are available, move the axis to its mid point, run the auto tuning and re-enable the pid controller when it is done. It can be run several times.

## 9.6 Remap Extending G-code

### 9.6.1 Introduction: Extending the RS274NGC Interpreter by Remapping Codes

#### 9.6.1.1 A Definition: Remapping Codes

By *remapping codes* we mean one of the following:

1. Define the semantics of new - that is, currently unallocated - M- or G-codes
2. Redefine the semantics of a - currently limited - set of existing codes.

#### 9.6.1.2 Why would you want to extend the RS274NGC Interpreter?

The set of codes (M,G,T,S,F) currently understood by the RS274NGC interpreter is fixed and cannot be extended by configuration options.

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 hodge-podge of special-case solutions.

A good example for this deficiency is tool change support in LinuxCNC. While random tool changers are supported well, 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 built-in code. Whenever the insufficient built-in code is to be executed, call the O-word procedure instead. While possible, this approach is cumbersome - it requires source-editing of NGC programs, replacing all calls to the deficient code by 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.

Generally, the behavior of a remapped code may be defined in the following ways:

- You define a O-word subroutine which implements the desired behavior
- Alternatively, you may employ a Python function which extends the interpreter's behavior.

**How to glue things together** M- and G-codes, and O-words subroutine calls have some fairly different syntax.

O-word procedures, for example, take positional parameters with a specific syntax like so:

```
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.

**How Embedded Python fits in** To make a simple situation easy and a complex situation solvable, the glue issue is addressed as follows:

- 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.

**A Word on Embedded 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 Getting started

Defining a code involves the following steps:

- Pick a code - either use an unallocated code, or redefine an existing code.
- Decide how parameters are handled.
- Decide if and how results are handled.
- Decide about the execution sequence.

### 9.6.2.1 Builtin Remaps

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.
- For G-codes, see the unallocated G-codes list.
- Existing codes that can be reassigned are listed in the remappable codes section.

There are currently two complete Python-only remaps that are available in stdglue.py:

- ignore_m6
- index_lathe_tool_with_wear

These are meant for use with lathe. Lathes don't use M6 to index the tools, they use the T command.

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.

Here is what you need in the INI to use them:

```
[RS274NGC]
REMAP=T python=index_lathe_tool_with_wear
REMAP=M6 python=ignore_m6

[PYTHON]
# where to find the Python code:

# code specific for this configuration
PATH_PREPEND=./
```

```
# generic support code - make sure this actually points to Python-stdglue
PATH_APPEND=../../nc_files/remap_lib/python-stdglue/

# import the following Python module
TOPLEVEL=toplevel.py

# the higher the more verbose tracing of the Python plugin
LOG_LEVEL = 0
```

You must also add the required Python file in your configuration folder.

Upgrade an existing configuration

#### 9.6.2.2 Picking a code

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.
- Existing codes which may be remapped are listed here.

#### 9.6.2.3 Parameter handling

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.

Referring to global variables is always possible, but they are easily overlooked.

Note there's a limited supply of words which may be used as parameters, so one might need to fall back to the second and third methods if many parameters are needed.

#### 9.6.2.4 Handling results

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 handlers help in processing results of remap procedures - see the reference section.

#### 9.6.2.5 Execution sequencing

Executable G-code words are classified into modal groups, which also defines their relative execution behavior.

If a G-code block contains several executable words on a line, these words are executed in a predefined order of execution, not in the order they appear in block.

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 An minimal example remapped 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
```

In a nutshell, this means:

- 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.

The file `myprocedure.ngc` is expected to exists in the `[DISPLAY]NC_FILES` or `[RS274NGC]SUBROUTINE_PATH` directory.

A detailed discussion of REMAP parameters is found in the reference section below.

### 9.6.3 Configuring Remapping

#### 9.6.3.1 The REMAP statement

To remap a code, define it using the `REMAP` option in `RS274NG` section of your INI file. Use one `REMAP` line per remapped code.

The syntax of the `REMAP` is:

**`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-коды**
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**
for these the modal group is fixed and any `modalgroup=` option is ignored.

**`argspec=<`*argspec*`>`**
See description of the argspec parameter options. Optional.

**`ngc=<`*ngc_basename*`>`**
Basename of an O-word subroutine file name. Do not specify an .ngc extension. Searched for in the directories specified in the directory specified in `[DISPLAY]PROGRAM_PREFIX`, then in `[RS274NGC]SUBROUTINE_PATH`. 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.

---

**Tip**
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`.

---

**Note**
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.

**Advanced example: Remapped codes in pure Python** The `interpreter` and `emccanon` modules expose most of the Interpreter and some Canon internals, so many things which so far required coding in *C/C+\+* can be now be done in Python.

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()`).

Assuming a definition like so (NB: this does not use argspec):

```
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 function with raw access to Interpreter internals """

    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

    # inspect controlling block for relevant words
    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
```

The examples described so far can be found in *configs/sim/axis/remap/getting-started* with complete working configurations.

### 9.6.4 Upgrading an existing configuration for remapping

The minimal prerequisites for using `REMAP` statements are as follows:

- 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.
- The Python interpreter needs to find the remap.py module above, so the path to the directory where your Python modules live needs to be added with `[PYTHON]PATH_APPEND=<path-to-your-local-Python-d`
- 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.

Assuming your configuration lives under `/home/user/xxx` and the INI file is `/home/user/xxx/xxx.ini`, execute the following commands.

```
$ 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
```

Now verify that LinuxCNC comes up with no error messages - from a terminal window execute:

```
$ cd /home/user/xxx
$ linuxcnc xxx.ini
```

## 9.6.5 Remapping tool change-related codes: T, M6, M61

### 9.6.5.1 Обзор

If you are unfamiliar with LinuxCNC internals, first read the How tool change currently works section (dire but necessary).

Note than when remapping an existing code, we completely disable this codes' built-in functionality of the interpreter.

So our remapped code will need to do a bit more than just generating some commands to move the machine as we like - it will also need to replicate those steps from this sequence which are needed to keep the interpreter and `task` happy.

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:

- Do we want to use an O-word procedure or do it all in Python code?
- Is the `iocontrol` HAL sequence (tool-prepare/tool-prepared and tool-change/tool-changed pins) good enough or do we need a different kind of HAL interaction for our tool changer (for example: more HAL pins involved with a different interaction sequence)?

Depending on the answer, we have four different scenarios:

- 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.

---

**Note**
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.

---

So the overall approach for our first example will be:

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 Understanding the role of `iocontrol` with remapped tool change codes

`iocontrol` provides two HAL interaction sequences we might or might not use:

- 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`:

---

**Note**
Устаревшие имена **selected_pocket** и **current_pocket** на самом деле ссылаются на последовательный индекс данных инструмента для элементов инструмента, загруженных из таблицы инструментов ([EMCIO]TOOL_TABLE) или через базу данных инструментов ([EMCIO]DB_PROGRAM).

---

```
def change_prolog(self, **words):
    try:
        if self.selected_pocket < 0:
            return "M6: no tool prepared"

        if self.cutter_comp_side:
            return "Cannot change tools with cutter radius compensation on"

        self.params["tool_in_spindle"] = self.current_tool
        self.params["selected_tool"] = self.selected_tool
        self.params["current_pocket"] = self.current_pocket
        self.params["selected_pocket"] = self.selected_pocket
        return INTERP_OK
    except Exception 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
```

**Caution**
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 Error handling: dealing with abort

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.

The reasons are defined in nml_intf/emc.hh

```
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
```

The suggested on_abort procedure would look like so (adapt to your needs):

```
o<on_abort> sub

G54 (origin offsets are set to the default)
G17 (select XY plane)
G90 (absolute)
G94 (feed mode: units/minute)
M48 (set feed and speed overrides)
G40 (cutter compensation off)
M5  (spindle off)
G80 (cancel modal motion)
M9  (mist and coolant off)

o100 if [#1 eq 5]
    (machine on)
o100 elseif [#1 eq 6]
    (machine off)
```

```
o100 elseif [#1 eq 7]
    (estopped)
o100 elseif [#1 eq 8]
    (msg, abort pressed)
o100 else
    (DEBUG, error parameter is [#1])
o100 endif

o<on_abort> endsub
m2
```

**Caution**
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.

Make sure `on_abort.ngc` is along the interpreter search path (recommended location: `SUBROUTINE_PATH` so as not to clutter your `NC_FILES` directory with internal procedures).

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`.

Note that terminating a remapped code by returning INTERP_ERROR from the epilog (see previous section) will also cause the `on_abort` procedure to be called.

#### 9.6.5.9 Error handling: failing a remapped code NGC procedure

If you determine in your handler procedure that some error condition occurred, do not use M2 to end your handler - see above:

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
```

**Note**
INI and HAL variable expansion is optional and can be disabled in the INI file

If more fine grained recovery action is needed, use the idiom laid out in the previous example:

- 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=`).

This error message will be displayed in the UI, and returning INTERP_ERROR will cause this error handled like any other runtime error.

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 Remapping other existing codes:

### 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 Adjusting the behavior of M0, M1

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.

### 9.6.6.3 Adjusting the behavior of M7, M8, M9

An example for remapping the built in behavior of M7/M8/M9 is the option to pass optional arguments like a P word for more complex coolant control (eg through tool vs external coolant flow).

See `configs/sim/axis/remap/extend-builtins/`, for an example of such an extension of the built in behavior for M7,M8 and M9.

## 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
```

Executing the following lines:

```
g17
(1)   g84.3 x1 y2 z3  r1
(2)   x3 y4 p2
(3)   x6 y7 z5
(4)   G80
```

causes the following (note *R* is sticky, and *Z* is sticky since the plane is *XY*):

1. `g843.ngc` is called with words x=1, y=2, z=3, r=1
2. `g843.ngc` is called with words x=3, y=4, z=3, p=2, r=1
3. `g843.ngc` is called with words x=6, y=7, z=3, r=1
4. The `G84.3` cycle is canceled.

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 Configuring Embedded Python

The Python plugin serves both the interpreter, and `task` if so configured, and hence has its own section `PYTHON` in the INI file.

### 9.6.8.1 Python plugin : INI file configuration

**[PYTHON]**

**TOPLEVEL = <*filename*>**
Filename of the initial Python script to execute on startup. This script is responsible for setting up the package name structure, see below.

**PATH_PREPEND = <*directory*>**
Prepend this directory to `PYTHON_PATH`. A repeating group.

**PATH_APPEND = <*directory*>**
Append this directory to `PYTHON_PATH`. A repeating group.

**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.

### 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 `this`, so you could also run:

```
;py,print(this.tool_table[0].toolno)
```

Here is an approach to use an O-word subroutine to read a preference file entry and add it as a G-code parameter.

```
(filename myofile.ngc)
o<myofile> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; find and print the preference file path
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; adjust for your preference file name
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; get an preference instance
;py,Pref = Access(PREFFILE)

; load a preference and print it
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])

; return the value
o<myofile> endsub [#<toolToLoad>]
M2
```

## 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.

### 9.6.9.2 The Interpreter as seen from Python

The interpreter is an existing C++ class (*Interp*) defined in *src/emc/rs274ngc*. Conceptually all `oword.<func` 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 The Interpreter `__init__` and `__delete__` functions

If the `TOPLEVEL` module defines a function `__init__`, it will be called once the interpreter is fully configured (INI file read, and state synchronized with the world model).

If the TOPLEVEL module defines a function __delete__, it will be called once before the interpreter is shutdown and after the persistent parameters have been saved to the PARAMETER_FILE.

Note_ at this time, the __delete__ handler does not work for interpreter instances created by importing the gcode module. If you need an equivalent functionality there (which is quite unlikely), please consider the Python atexit module.

```
# this would be defined in the TOPLEVEL module

def __init__(self):
    # add any one-time initialization here
    if self.task:
  # this is the milltask instance of interp
  pass
    else:
  # this is a non-milltask instance of interp
        pass

def __delete__(self):
    # add any cleanup/state saving actions here
    if self.task: # as above
  pass
    else:
        pass
```

This function may be used to initialize any Python-side attributes which might be needed later, for instance in remap or 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).

See also `tests/interp/value-returned` for an example.

**Calling conventions for prolog= and epilog= subroutines** Arguments are:

**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.

Example call:

```
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
```

Return values:

**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> )`.

**Calling conventions for python= subroutines** Arguments are:

**`self`**
: The interpreter instance.

**`words`**
: Keyword parameter dictionary. The same kwargs dictionary as prologs and epilogs (see above).

The minimum `python=` function example:

```
def useless(self,  **words): # in remap module
    return INTERP_OK
```

Return values:

**`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.

**Dealing with queue-buster: Probe, Tool change and waiting for a 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.

This demonstrates call continuation with a single point-of-restart:

```
def read_pin(self,*args):
    # wait 5secs for digital-input 00 to go high
    emccanon.WAIT(0,1,2,5.0)
    # cede control after executing the queue buster:
    yield INTERP_EXECUTE_FINISH
    # post-sync() execution resumes here:
    pin_status = emccanon.GET_EXTERNAL_DIGITAL_INPUT(0,0);
    print(”pin status=”,pin_status)
```

**Warning**
The *yield* feature is fragile. The following restrictions apply to the usage of *yield INTERP_EXECUTE_FINISH*:

- Python code executing a *yield INTERP_EXECUTE_FINISH* must be part of a remap procedure. Yield does not work in a Python oword procedure.
- A Python remap subroutine containing *yield INTERP_EXECUTE_FINISH* statement may not return a value, as with normal Python yield statements.
- 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 Calling conventions: Python to NGC

NGC code is executed from Python when

- the method `self.execute(<NGC code>[,<line number>])` is executed, or
- 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>])
```

Examples:

```
  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
- passing of local named parameters is not needed.

The recursive call feature is fragile.

**Interpreter Exception during execute()** if `interpreter.throw_exceptions` is nonzero (default 1), and self.execute() returns an error, the exception `InterpreterException` is raised. InterpreterException has the following attributes:

**`line_number`**
: where the error occurred

**`line_text`**
: the NGC statement causing the error

**`error_message`**
: the interpreter's error message

Errors can be trapped in the following Pythonic way:

```
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  # replace builtin error message
```

**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 Built in modules

The following modules are built in:

**`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`.

### 9.6.10 Adding Predefined Named Parameters

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.

To add or redefine a named parameter:

- 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
```

```
#<circumference> = [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.

Ònly functions with a leading underscore are added as parameters, since this is the RS274NGC convention for globals.

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 Standard Glue routines

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.

**Actions of prepare_prolog** The following parameters are made visible to the NGC procedure:

- `#<tool>` - the parameter of the `T` word
- `#<pocket>` - the corresponding pocket

If tool number zero is requested (meaning Tool unload), the corresponding pocket is passed as -1.

It is an error if:

- No tool number is given as T parameter,
- the tool cannot be found in the tool table.

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`

These wrap a NGC procedure for M6 Tool Change.

Actions of `change_prolog`

- If there was no preceding T command which caused a pocket to be selected, the prolog aborts with an error message.
- If cutter radius compensation is on, the prolog aborts with an error message.

Then, the following parameters are exported to the NGC procedure:

- `#<tool_in_spindle>` : the tool number of the currently loaded tool
- `#<selected_tool>` : the tool number selected

- #<selected_pocket> : the selected tool's tooldata index

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.
- 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.
- The new tool parameters (offsets, diameter etc) are set.

#### 9.6.11.3 G-code Cycles: cycle_prolog and 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.

The suggested argspec is as follows:

```
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

- Determine whether the words passed in from the current block fulfill the conditions outlined under Canned Cycle Errors.
  - 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.
- Determine whether this is the first invocation of a cycle G-code, if so:
  - 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.

- Export the set of sticky parameters to the NGC procedure.

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

The following flags are relevant for remapping and Python - related execution:

| | | |
|---|---|---|
| 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_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/debugflags.h*.

#### 9.6.12.5 Debugging Embedded 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.

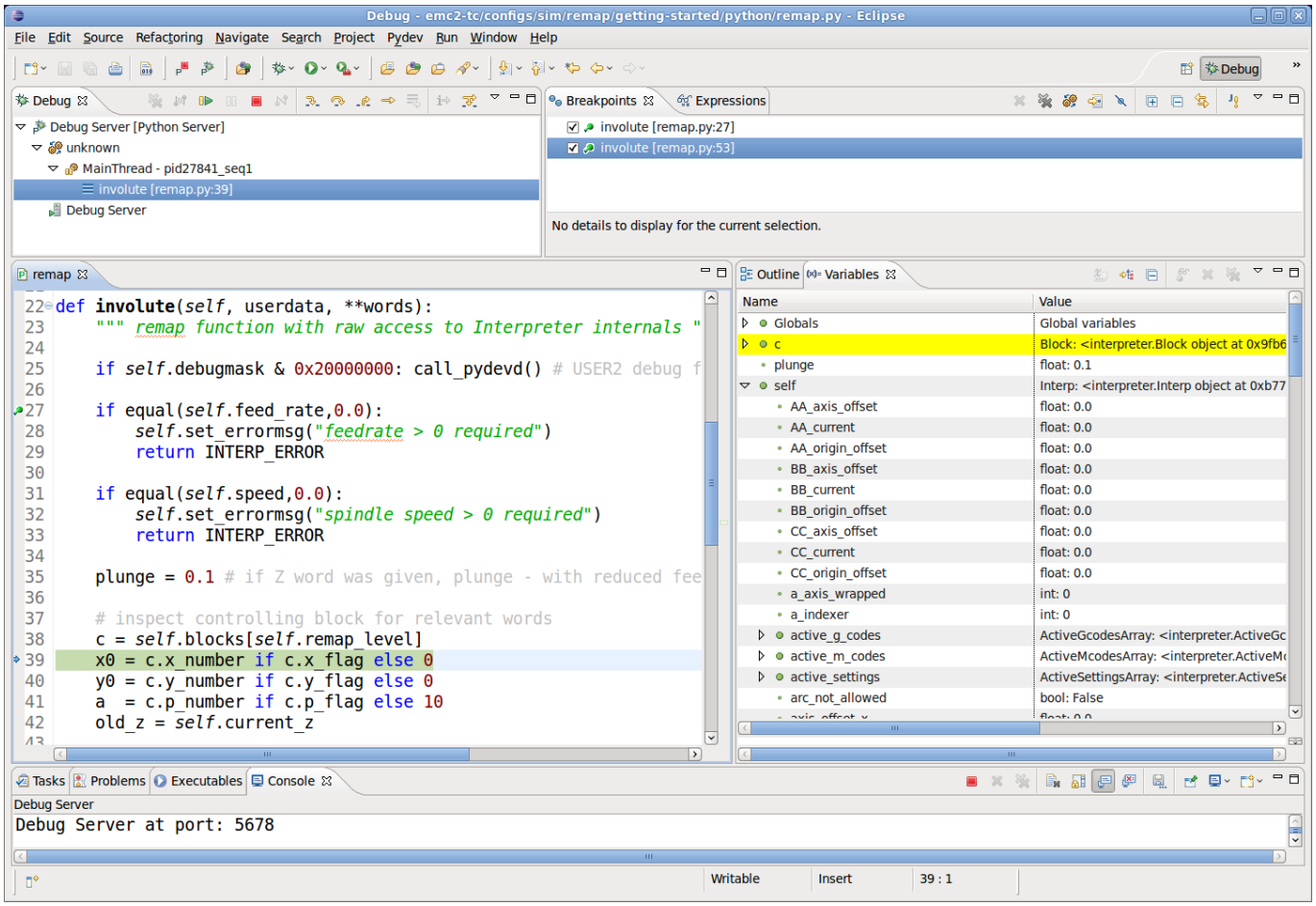
To use this approach:

- Install Eclipse via the the *Ubuntu Software Center* (choose first selection).
- 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.

**Caution**
`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)
- M7 (активировать туман охлаждающей жидкости)
- M8 (activate coolant flood)
- M9 (deactivate coolant mist and flood)
- M60 (exchange pallet shuttles and then pause a running program temporarily)
- M62 .. M65 (управление цифровым выводом)
- M66 (wait on input)
- M67, M68 (analog output control)
- S (set spindle speed)
- F (set feed)

### 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.)

Table 9.5: 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 | | | | | | | | | |

Table 9.5: (continued)

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *08* | G08 | | | | | | | | | |
| *09* | | | | | | | | | | |

Table 9.6: 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 | | | | | | | | |

Table 9.7: 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* | | | | | | | | | | |

Table 9.8: Table of Allocated G-codes 30-39

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *30* | G30 | G30.1 | | | | | | | | |
| *31* | | | | | | | | | | |
| *32* | | | | | | | | | | |
| *33* | G30 | G30.1 | | | | | | | | |
| *34* | | | | | | | | | | |
| *35* | | | | | | | | | | |
| *36* | | | | | | | | | | |
| *37* | | | | | | | | | | |
| *38* | | | | | | | | | | |
| *39* | | | | | | | | | | |

Table 9.9: Table of Allocated G-codes 40-49

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *40* | G40 | | | | | | | | | |
| *41* | G41 | G41.1 | | | | | | | | |
| *42* | G42 | G42.1 | | | | | | | | |
| *43* | G43 | G43.1 | | | | | | | | |
| *44* | | | | | | | | | | |
| *45* | | | | | | | | | | |
| *46* | | | | | | | | | | |
| *47* | | | | | | | | | | |
| *48* | | | | | | | | | | |
| *49* | G40 | | | | | | | | | |

Table 9.10: Table of Allocated G-codes 50-59

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *50* | | | | | | | | | | |
| *51* | | | | | | | | | | |
| *52* | | | | | | | | | | |
| *53* | G53 | | | | | | | | | |
| *54* | G54 | | | | | | | | | |
| *55* | G55 | | | | | | | | | |
| *56* | G56 | | | | | | | | | |
| *57* | G57 | | | | | | | | | |
| *58* | G58 | | | | | | | | | |
| *59* | G59 | G59.1 | G59.2 | G59.3 | | | | | | |

Table 9.11: Table of Allocated G-codes 60-69

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *60* | G60 | | | | | | | | | |
| *61* | G61 | G61.1 | | | | | | | | |
| *62* | | | | | | | | | | |
| *63* | | | | | | | | | | |
| *64* | G64 | | | | | | | | | |
| *65* | | | | | | | | | | |
| *66* | | | | | | | | | | |
| *67* | | | | | | | | | | |
| *68* | | | | | | | | | | |
| *69* | | | | | | | | | | |

Table 9.12: Table of Allocated G-codes 70-79

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *70* | G70 | | | | | | | | | |
| *71* | G71 | G71.1 | G71.2 | | | | | | | |
| *72* | G72 | G72.1 | G72.2 | | | | | | | |
| *73* | | | | | | | | | | |
| *74* | | | | | | | | | | |
| *75* | | | | | | | | | | |
| *76* | G76 | | | | | | | | | |
| *77* | | | | | | | | | | |
| *78* | | | | | | | | | | |
| *79* | | | | | | | | | | |

Table 9.13: Table of Allocated G-codes 80-89

| # | Gxx | Gxx.1 | Gxx.2 | Gxx.3 | Gxx.4 | Gxx.5 | Gxx.6 | Gxx.7 | Gxx.8 | Gxx.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *80* | G80 | | | | | | | | | |
| *81* | G81 | | | | | | | | | |
| *82* | G82 | | | | | | | | | |
| *83* | G83 | | | | | | | | | |
| *84* | G84 | | | | | | | | | |
| *85* | G85 | | | | | | | | | |
| *86* | G86 | | | | | | | | | |
| *87* | G87 | | | | | | | | | |
| *88* | G88 | | | | | | | | | |
| *89* | G89 | | | | | | | | | |

Table 9.14: Table of Allocated 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 Currently unallocated 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.)

Table 9.15: Table of Unallocated 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 |

All M-codes from `M100` to `M199` are user-defined M-codes already, and should not be remapped.

All M-codes from `M200` to `M999` are available for remapping.

## 9.6.15 A short survey of LinuxCNC program execution

To understand remapping of codes it might be helpful to survey the execution of `task` and interpreter as far as it relates to remapping.

### 9.6.15.1 Interpreter state

Conceptually, the interpreter's state consist of variables which fall into the following categories:

1. *Configuration information* (typically from INI file)
2. *The 'World model'* - a representation of actual machine state
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).

### 9.6.15.2 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*.

#### 9.6.15.3 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.

#### 9.6.15.4 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.

#### 9.6.15.5 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.

#### 9.6.15.6 Word order and execution order

One or several *words* may be present on an NGC *block* if they are compatible (some are mutually exclusive and must be on different lines). The execution model however prescribes a strict ordering of execution of codes, regardless of their appearance on the source line (G-code Order of Execution).

#### 9.6.15.7 Parsing

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.

#### 9.6.15.8 Execution

After successful parsing the block is executed by execute_block(), and here the different items are handled according to execution order.

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.

#### 9.6.15.9 Procedure execution

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).

#### 9.6.15.10 How tool change currently works

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.

**How tool information is communicated** Several processes are *interested* in tool information: `task` and its interpreter, as well as the user interface. Also, the *halui* process.

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])
```

You need to have LinuxCNC started from a terminal window to see the results.

#### 9.6.15.11 How T*x* (Prepare Tool) works

**Interpreter action on a Tx command**

All the interpreter does is evaluate the toolnumber parameter, looks up its corresponding tooldata index, remembers it in the `selected_pocket` variable for later, and queues a canon command (SELECT_TOOL). See *Interp::convert_tool_select* in *src/emc/rs274/interp_execute.cc*.

**Task action on SELECT_TOOL** When `task` gets around to handle a SELECT_TOOL, it sends a EMC_TOOL_PREPARE message to the `iocontrol` process, which handles most tool-related actions in LinuxCNC.

In the current implementation, `task` actually waits for `iocontrol` to complete the changer positioning operation, which is not necessary IMO since it defeats the idea that changer preparation and code execution can proceed in parallel.

**Iocontrol action on EMC_TOOL_PREPARE** When `iocontrol` sees the select pocket command, it does the related HAL pin wiggling - it sets the "tool-prep-number" pin to indicate which tool is next, raises the "tool-prepare" pin, and waits for the "tool-prepared" pin to go high.

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`.

#### 9.6.15.12 How M6 (Change tool) works

You need to understand this fully before you can adapt it. It is very relevant to writing a prolog and epilog handler for a remapped M6. Remapping an existing codes means you disable the internal steps taken normally, and replicate them as far as needed for your own purposes.

Even if you are not familiar with C, I suggest you look at the *Interp::convert_tool_change* code in *src/emc/rs274/interp_convert.cc*.

**Interpreter action on a M6 command**

When the interpreter sees an M6, it:

1. checks whether a T command has already been executed (test *settings->selected_pocket* to be >= 0) and fail with *Need tool prepared -Txx- for toolchange* message if not.
2. check for cutter compensation being active, and fail with *Cannot change tools with cutter radius compensation on* if so.

3. stop the spindle except if the ”TOOL_CHANGE_WITH_SPINDLE_ON” INI option is set.
4. generate a rapid *Z up* move if if the ”TOOL_CHANGE_QUILL_UP” INI option is set.
5. if TOOL_CHANGE_AT_G30 was set:
   a. move the A, B and C indexers if applicable
   b. generate rapid move to the G30 position
6. execute a CHANGE_TOOL canon command, with the selected pocket as a parameter. CHANGE_TOOL will:
   a. generate a rapid move to TOOL_CHANGE_POSITION if so set in INI
   b. enqueue an EMC_TOOL_LOAD NML message to `task`.
7. set the numberer parameters 5400-5413 according to the new tool
8. signal to `task` to stop calling the interpreter for readahead by returning INTERP_EXECUTE_FINISH since M6 is a queue buster.

**What task does when it sees a CHANGE_TOOL command** Again, not much more than passing the buck to `iocontrol` by sending it an EMC_TOOL_LOAD message, and waiting until `iocontrol` has done its thing.

Iocontrol action on EMC_TOOL_LOAD

1. it asserts the ”tool-change” pin
2. it waits for the ”tool-changed” pin to become active
3. when that has happened:
   a. deassert ”tool-change”
   b. set ”tool-prep-number” and ”tool-prep-pocket” pins to zero
   c. execute the *load_tool()* function with the pocket as parameter.

The last step actually sets the tooltable entries in the *emcStatus* structure. The actual action taken depends on whether the RANDOM_TOOLCHANGER INI option was set, but at the end of the process *toolTable[0]* reflects the tool currently in the spindle.

When that has happened:

1. `iocontrol` signals `task` to go ahead.
2. `task` tells the interpreter to execute a *synch()* operation, to see what has changed.
3. The interpreter *synch()* pulls all information from the world model needed, among it the changed tool table.

From there on, the interpreter has complete knowledge of the world model and continues with read ahead.

**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`.

#### 9.6.15.13 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`.

### 9.6.16 Status

1. The RELOAD_ON_CHANGE feature is fairly broken. Restart after changing a Python file.

### 9.6.17 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.

### 9.6.18 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 Компонент Moveoff

The moveoff HAL component is a HAL-only method for implementing offsets. See the manpage (*$ man moveoff*) for the IMPORTANT limitations and warnings.

Компонент перемещения используется для смещения позиций сочленений с использованием пользовательских соединений HAL. Реализация функции смещения во время паузы программы поддерживается соответствующими соединениями для входных контактов. Поддерживаются девять сочленений.

Значения смещения оси (offset-in-M) непрерывно применяются (с учетом ограничений на значение, скорость и ускорение) к выходным контактам (offset-current-M, pos-plusoffset-M, fb-minusoffset-M), когда оба входных контакта включения (apply-offsets и move-enable) имеют значение TRUE. Два разрешающих входа соединены внутренне как логическое ”И”. Становится активным *контакт предупреждения* и выдается сообщение, если контакт apply-offsets становится неактивным во время применения смещений. Контакт предупреждения остается TRUE до тех пор, пока смещения не будут удалены или не будет активен контакт apply-offsets.

Typically, the move-enable pin is connected to external controls and the apply-offsets pin is connected to halui.program.is-paused (for offsets only while paused) or set to TRUE (for continuously applied offsets).

Applied offsets are *automatically returned* to zero (respecting limits) when either of the enabling inputs is deactivated. The zero value tolerance is specified by the epsilon input pin value.

Путевые точки записываются, когда разрешен компонент перемещения. Путевые точки управляются с помощью контактов waypoint-sample-secs и waypoint-threshold. Когда контакт backtrack-enable в TRUE, auto-return путь следует по записанным путевым точкам. Когда память, доступная для путевых точек, исчерпана, смещения фиксируются и контакт waypoint-limit становится активным. Это ограничение применяется независимо от состояния контакта backtrack-enable. Разрешающий контакт должен быть неактивен, чтобы обеспечить возврат в исходное положение (положение без смещения).

Возврат через путевые точки приводит к «замедлению» скорости движения, поскольку перемещения выполняются от точки к точке с соблюдением настроек скорости и ускорения. Контактами ограничения скорости и ускорения можно управлять динамически, чтобы постоянно контролировать смещения.

When backtrack-enable is FALSE, the auto-return move is **NOT** coordinated, each axis returns to zero at its own rate. If a controlled path is wanted in this condition, each axis should be manually returned to zero before deasserting an enabling pin.

Контакты waypoint-sample-secs, waypoint-threshold и epsilon оцениваются только тогда, когда компонент простаивает.

The offsets-applied output pin is provided to indicate the current state to a GUI so that program resumption can be managed. If the offset(s) are non-zero when the apply-offsets pin is deasserted (for example when resuming a program when offsetting during a pause), offsets are returned to zero (respecting limits) and an *Error* message is issued.

**Caution**
Если смещения включены и применяются, а станок по какой-либо причине выключен, любая «внешняя» логика HAL, которая управляет разрешающими контактами и offset-in-M входами, отвечает за их состояние, когда станок впоследствии снова включается.

This HAL-only means of offsetting is typically not known to LinuxCNC nor available in GUI preview displays. **No protection is provided** for offset moves that exceed soft limits managed by LinuxCNC. Since soft limits are not honored, an offset move may encounter hard limits (or **CRASH** if there are no limit switches). Use of the offset-min-M and offset-max-M inputs to limit travel is recommended. Triggering a hard limit will turn off the machine — see **Caution** above.

The offset-in-M values may be set with INI file settings, controlled by a GUI, or managed by other HAL components and connections. Fixed values may be appropriate in simple cases where the direction and amount of offset is well-defined but a control method is required to deactivate an enabling pin in order to return offsets to zero. GUIs may provide means for users to set, increment, decrement, and accumulate offset values for each axis and may set offset-in-M values to zero before deasserting an enabling pin.

The default values for accel, vel, min, max, epsilon, waypoint-sample-secs, and waypoint-threshold may not be suitable for any particular application. This HAL component is unaware of limits enforced elsewhere by LinuxCNC. Users should test usage in a simulator application and understand all hazards before use on hardware.

Sim configurations that demonstrate the component and a GUI (moveoff_gui) are located in:

- configs/sim/axis/moveoff (axis-ui)
- configs/sim/touchy/ngcgui (touchy-ui)

### 9.7.1 Modifying an existing configuration

A system-provided HAL file (LIB:hookup_moveoff.tcl) can be used to adapt an existing configuration to use the moveoff component. Additional INI file settings support the use of a simple GUI (moveoff_gui) for controlling offsets.

When the system HAL file (LIB:hookup_moveoff.tcl) is properly specified in a configuration INI file, it will:

1. Disconnect the original joint.N.motor-pos-cmd and joint.N.motor-pos-fb pin connections
2. Load (loadrt) the moveoff component (using the name mv) with a personality set to accommodate all axes identified in the INI file
3. Add (addf) the moveoff component functions in the required sequence
4. Повторно соедините контакты Joint.N.motor-pos-cmd и Joint.N.motor-pos-fb, чтобы использовать компонент перемещения
5. Set the moveoff component operating parameters and limits for each axis in accordance with additional INI file settings

Note: The moveoff_gui application supports configurations that use known kinematics modules with KINEMATICS_TYPE=KINEMATICS_IDENTITY. Supported modules include: trivkins. With identity kins, moveoff_gui assigns each axis name specified with the command line parameter *-axes axisnames* to the corresponding joint.

Modify an existing configuration as follows:

Убедитесь, что существует запись в INI-файле для `[HAL]HALUI` и создайте новую запись [HAL]HALFILE` для LIB:hookup_moveoff.tcl. Запись для LIB:hookup_moveoff.tcl должна следовать за всеми записями `HALFILE==` для файлов HAL, которые соединяют контакты для `joint.N.motor-pos-cmd`, `joint.N.motor-` и любых компонентов, подключенных к этим контактам (например, компоненты ПИД и энкодер в сервосистеме).

```
[HAL]
HALUI   = halui
HALFILE = existing_configuration_halfile_1
...
HALFILE = existing_configuration_halfile_n
HALFILE = LIB:hookup_moveoff.tcl
```

Add INI file entries for the per-axis settings for each axis in use (if an entry is not defined, the corresponding entry from the [AXIS_n] section will be used, if no entry is found, then the moveoff component default will be used).

---

**Note**
Using component defaults or [AXIS_n] section values for per-axis offset settings is NOT recommended.

---

```
[MOVEOFF_n]
MAX_LIMIT =
MIN_LIMIT =
MAX_VELOCITY =
MAX_ACCELERATION =
```

Add INI file entries for moveoff component settings (omit to use moveoff defaults):

```
[MOVEOFF]
EPSILON =
WAYPOINT_SAMPLE_SECS =
WAYPOINT_THRESHOLD =
```

The moveoff_gui is used to make additional required connections and provide a popup GUI to:

1. Provide a control togglebutton to Enable/Disable offsets
2. Provide a control togglebutton to Enable/Disable backtracking
3. Provide control pushbuttons to Increment/Decrement/Zero each axis offset
4. Display each axis offset current value
5. Display current offset status (disabled, active, removing, etc)

The provided control buttons are optional depending upon the state of the moveoff component move-enable pin. Both a display and controls for enabling offsetting are provided if the pin mv.move-enable is NOT connected when the moveoff_gui is started. For this case, the moveoff_gui manages the moveoff component move-enable pin (named mv.move-enable) as well as the offsets (mv.move-offset-in-M) and the backtracking enable (mv.backtrack-enable)

Если контакт mv.move-enable подключен при запуске moveoff_gui, то moveoff_gui выдаст экран, но НЕ элементы управления. Этот режим поддерживает конфигурации, в которых используется маховичок медленной подачи или другие методы управления входами смещения и разрешающими контактами (mv.offset-in-M, mv.move-enable, mv.backtrack-enable).

moveoff_gui создает необходимые соединения для контактов компонента перемещения: mv.power_on и mv.apply-offsets. Вывод mv.power_on подключается к выводу с поддержкой motion.motion (при необходимости автоматически создается новый сигнал). mv.apply-offsets подключается к halui.program paused или имеет значение 1 в зависимости от параметра командной строки -mode [ onpause | always ]. При необходимости автоматически создается новый сигнал.

To use the moveoff_gui, add an entry in the INI file [APPLICATIONS] section as follows:

```
[APPLICATIONS]
# Note: a delay (specified in seconds) may be required if connections
# are made using postgui HAL files ([HAL]POSTGUI_HALFILE=)
DELAY = 0
APP = moveoff_gui option1 option2 ...
```

When the HAL file LIB:hookup_moveoff.tcl is used to load and connect the moveoff component, the mv.move-enable pin will not be connected and local controls provided by the moveoff_gui will be used. This is the simplest method to test or demonstrate the moveoff component when modifying an existing INI configuration.

Чтобы включить внешние элементы управления при использовании отображения moveoff_gui для значений смещения и состояния, файлы HAL, следующие за LIB:hookup_moveoff.tcl, должны установить дополнительные соединения. Например, в поставляемых демонстрационных конфигурации (configs/sim/axis/moveoff/*.ini) используется простой системный файл HAL (с именем LIB:moveoff_extern чтобы подключать контакты mv.move-enable, mv.offset-in-M. и mv.bactrack-enable к сигналам:

```
[HAL]
HALUI = halui
...
HALFILE = LIB:hookup_moveoff.tcl
HALFILE = LIB:moveoff_external.hal
```

The connections made by LIB:moveoff_external.hal (for a three axis configuration) are:

```
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
```

These signals (external_enable, external_offset_M, external_backtrack_en) may be managed by subsequent HALFILES (including POSTGUI_HALFILEs) to provide customized control of the component while using the moveoff_gui display for current offset values and offset status.

The moveoff_gui is configured with command line options. For details on the operation of moveoff_gui, see the man page:

```
$ man moveoff_gui
```

For a brief listing of command line options for moveoff_gui, use the command line help option:

```
$ moveoff_gui --help

Usage:
moveoff_gui [Options]

Options:
    [--help | -? | -- -h ]  (This text)

    [-mode [onpause | always]]  (default: onpause)
                                  (onpause: show gui when program paused)
                                  (always:  show gui always)

    [-axes axisnames]       (default: xyz (no spaces))
                              (letters from set of: x y z a b c u v w)
                              (example: -axes z)
                              (example: -axes xz)
                              (example: -axes xyz)
    [-inc incrementvalue]   (default: 0.001 0.01 0.10 1.0 )
                              (specify one per -inc (up to 4) )
                              (example: -inc 0.001 -inc 0.01 -inc 0.1 )
    [-size integer]         (default: 14)
                              (Overall gui popup size is based on font size)
    [-loc center|+x+y]      (default: center)
                              (example: -loc +10+200)
    [-autoresume]           (default: notused)
                              (resume program when move-enable deasserted)
    [-delay delay_secs]     (default: 5 (resume delay))

Options for special cases:
    [-noentry]              (default: notused)
                             (don't create entry widgets)
    [-no_resume_inhibit]    (default: notused)
                             (do not use a resume-inhibit-pin)
    [-no_pause_requirement] (default: notused)
                             (no check for halui.program.is-paused)
    [-no_cancel_autoresume] (default: notused)
                             (useful for retraact offsets with simple)
                             (external control)
    [-no_display]           (default: notused)
                             (Use when both external controls and displays)
                             (are in use (see Note))
```

```
Note: If the moveoff move-enable pin (mv.move-enable) is connected when
      moveoff_gui is started, external controls are required and only
      displays are provided.
```

## 9.8 Автономный интерпретатор

Автономный интерпретатор rs274 доступен для использования через командную строку.

### 9.8.1 Использование

```
b''Ив''b''сb''b''пв''b''ов''b''лв''b''ьв''b''зв''b''ов''b''вв''b''ав''b''нв''b''ив''b' ↩
    'eb'': rs274 [-p interp.so] [-t tool.tbl] [-v var-file.var] [-n 0|1|2]
            [-b] [-s] [-g] [b''вв''b''хb''b''ов''b''дв''b''нв''b''ов''b''йb'' b''фв''b' ↩
                'ав''b''йв''b''лв'' [b''вв''b''ыв''b''хb''b''ов''b''дв''b''нв''b''ов''b' ↩
                'йв'' b''фв''b''ав''b''йв''b''лв'']]

    -p: b''уb''b''кв''b''ав''b''жв''b''ив''b''тв''b''eb'' b''дв''b''лв''b''яв'' b''ив''b' ↩
        'сb''b''пв''b''ов''b''лв''b''ьв''b''зв''b''ов''b''вв''b''ав''b''нв''b''ив''b''яв'' b ↩
        ''пв''b''ов''b''дв''b''кв''b''лв''b''юв''b''чв''b''ав''b''eb''b''мв''b''ыв''b''йв''  ↩
        b''ив''b''нв''b''тв''b''eb''b''рb''b''пв''b''рb''b''eb''b''тв''b''ав''b''тв''b''ов'' ↩
        b''рb''
     -t: b''уb''b''кв''b''ав''b''жв''b''ив''b''тв''b''eb'' b''дв''b''лв''b''яв'' b''ив''b' ↩
         'сb''b''пв''b''ов''b''лв''b''ьв''b''зв''b''ов''b''вв''b''ав''b''нв''b''ив''b''яв''  ↩
         b''фв''b''ав''b''йв''b''лв'' .tbl (b''тв''b''ав''b''бв''b''лв''b''ив''b''цв''b' ↩
         'ав'' b''ив''b''нв''b''сb''b''тв''b''рb''b''уb''b''мв''b''eb''b''нв''b''тв''b''ов'' ↩
         b''вв'')
     -v: b''уb''b''кв''b''ав''b''жв''b''ив''b''тв''b''eb'' b''дв''b''лв''b''яв'' b''ив''b' ↩
         'сb''b''пв''b''ов''b''лв''b''ьв''b''зв''b''ов''b''вв''b''ав''b''нв''b''ив''b''яв''  ↩
         b''фв''b''ав''b''йв''b''лв'' .var (b''пв''b''ав''b''рb''b''ав''b''мв''b''eb''b' ↩
         'тв''b''рb''b''ов''b''вв'')
     -n: b''уb''b''кв''b''ав''b''жв''b''ив''b''тв''b''eb'' b''рb''b''eb''b''жв''b''ив''b' ↩
         'мв'' b''пв''b''рb''b''ов''b''дв''b''ов''b''лв''b''жв''b''eb''b''нв''b''ив''b' ↩
         'яв'':
           0: b''пв''b''рb''b''ов''b''дв''b''ов''b''лв''b''жв''b''ив''b''тв''b''ьв''
           1: b''вв''b''ов''b''йв''b''тв''b''ив'' b''вв'' b''рb''b''eb''b''жв''b''ив''b' ↩
               'мв'' MDI
           2: b''сb''b''тв''b''ов''b''пв'' (b''пв''b''ов'' b''уb''b''мв''b''ов''b''лв''b' ↩
               'чв''b''ав''b''нв''b''ив''b''юв'')
     -b: b''пв''b''eb''b''рb''b''eb''b''кв''b''лв''b''юв''b''чв''b''ив''b''тв''b''ьв'' b' ↩
         'фв''b''лв''b''ав''b''гb'' «b''уb''b''дв''b''ав''b''лв''b''ив''b''тв''b''ьв'' b' ↩
         'бв''b''лв''b''ов''b''кв''» (b''пв''b''ов'' b''уb''b''мв''b''ов''b''лв''b''чв''b' ↩
         'ав''b''нв''b''ив''b''юв'': b''Вв''b''Ыв''b''Кв''b''Лв'')
     -s: b''пв''b''eb''b''рb''b''eb''b''кв''b''лв''b''юв''b''чв''b''ив''b''тв''b''ьв'' b' ↩
         'фв''b''лв''b''ав''b''гb'' «b''сb''b''тв''b''eb''b''кв'' b''пв''b''eb''b''чв''b' ↩
         'ав''b''тв''b''ив''» (b''пв''b''ов'' b''уb''b''мв''b''ов''b''лв''b''чв''b''ав''b' ↩
         'нв''b''ив''b''юв'': b''Вв''b''Ыв''b''Кв''b''Лв'')
     -g: b''пв''b''eb''b''рb''b''eb''b''кв''b''лв''b''юв''b''чв''b''ив''b''тв''b''ьв'' b' ↩
         'фв''b''лв''b''ав''b''гb'' «b''пв''b''eb''b''рb''b''eb''b''йв''b''тв''b''ив'' b' ↩
         'вв'' (b''пв''b''ав''b''кв''b''eb''b''тв''b''нв''b''ыв''b''йв'' b''рb''b''eb''b' ↩
         'жв''b''ив''b''мв'')» (b''пв''b''ов'' b''уb''b''мв''b''ов''b''лв''b''чв''b''ав''b' ↩
         'нв''b''ив''b''юв'': b''Вв''b''Ыв''b''Кв''b''Лв'')
     -i: b''уb''b''кв''b''ав''b''зв''b''ав''b''тв''b''ьв'' b''фв''b''ав''b''йв''b''лв'' . ↩
         ini (b''пв''b''ов'' b''уb''b''мв''b''ов''b''лв''b''чв''b''ав''b''нв''b''ив''b' ↩
         'юв'': b''нв''b''eb''b''тв'' b''фв''b''ав''b''йв''b''лв''b''ав'' ini)
     -T: b''вв''b''ыв''b''зв''b''вв''b''ав''b''тв''b''ьв'' task_init()
```

```
     -l: b''yb''b''кb''b''ab''b''зb''b''ab''b''тb''b''ьb'' b''yb''b''pb''b''ob''b''вb''b' ↩
         'eb''b''нb''b''ьb'' b''жb''b''yb''b''pb''b''нb''b''ab''b''лb''b''иb''b''pb''b''ob'' ↩
         b''вb''b''ab''b''нb''b''иb''b''яb'' (b''пb''b''ob'' b''yb''b''мb''b''ob''b''лb''b' ↩
         'чb''b''ab''b''нb''b''иb''b''юb'': -1)
```

### 9.8.2 Пример

Например, чтобы увидеть результат цикла, мы можем запустить rs274 со следующим файлом и увидеть, что цикл никогда не заканчивается. Чтобы выйти из цикла, используйте Ctrl Z. Для запуска примера необходимы следующие два файла.

**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 b''кb''b''ob''b''нb''b''цb''b''eb''b''вb''b''ab''b''яb'' b''фb''b' ↩
    'pb''b''eb''b''зb''b''ab''
T2 P2 Z0.1 D0.0625 ;1/16 b''кb''b''ob''b''нb''b''цb''b''eb''b''вb''b''ab''b''яb'' b''фb''b' ↩
    'pb''b''eb''b''зb''b''ab''
T3 P3 Z1.273 D0.201 ;#7 b''cb''b''вb''b''eb''b''pb''b''лb''b''ob'' b''пb''b''ob''b''дb'' b' ↩
    'pb''b''eb''b''зb''b''ьb''b''6b''b''yb''
```

**command**

```
rs274 -g test.ngc -t test.tbl
```

## 9.9 Смещения внешней оси

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

For each axis letter (**L** in xyzabcuvw):

```
[AXIS_L]OFFSET_AV_RATIO = value (controls accel/vel for external offsets)
```

1. Allowed values: 0 <= value <= 0.9
2. Disallowed values are replaced with 0.1 with message to stdout

3. Default value: 0 (disables external offset).
   Consequence: omitted [AXIS_L]OFFSET_AV_RATIO disables external offset for the axis.
4. If nonzero, the OFFSET_AV_RATIO (**r**), adjusts the conventional (planning) max velocity and acceleration to preserve [AXIS_L] constraints:

```
planning max velocity        = (1-r) * MAX_VELOCITY
external offset velocity     = (  r) * MAX_VELOCITY

planning max acceleration    = (1-r) * MAX_ACCELERATION
external offset acceleration = (  r) * MAX_ACCELERATION
```

## 9.9.2 HAL Контакты

### 9.9.2.1 Per-Axis Motion HAL Pins

Для каждой буквы оси (***L*** в xyzabcuvw)

1. **axis.*L*.eoffset-enable** Input(bit): enable
2. **axis.*L*.eoffset-scale** Input(float): scale factor
3. **axis.*L*.eoffset-counts** Input(s32): input to counts register
4. **axis.*L*.eoffset-clear** Input(bit): clear requested offset
5. **axis.*L*.eoffset** Output(float): current external offset
6. **axis.*L*.eoffset-request** Output(float): requested external offset

### 9.9.2.2 Other Motion HAL Pins

1. **motion.eoffset-active** Output(bit): non-zero external offsets applied
2. **motion.eoffset-limited** Output(bit): motion inhibited due to soft limit

## 9.9.3 Использование

The axis input HAL pins (enable,scale,counts) are similar to the pins used for wheel jogging.

### 9.9.3.1 Offset Computation

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.

For many applications, the *axis*.L.*eoffset-scale* pin is constant and the net *axis*.L.*eoffset-request* response to *axis*.L.*eoffset-counts* is equivalent to the product of the accumulated value of *axis*.L.*eoffset-counts* and the (constant) *axis*.L.*eoffset-scale* pin values.

#### 9.9.3.2 Machine-off/Machine-on

When the machine is turned off, the **current position** with **external offsets** is **maintained** so that there is no unexpected motion at turn off or turn on.

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.

In other words, external offsets are **defined as ZERO at each startup** (machine-on) regardless of the value of the *axis*.L.*eoffset-counts* pins. To avoid confusion, it is recommended that all *axis*.L.*eoffset-counts* pins are set to zero when the machine is off.

#### 9.9.3.3 Програмные пределы

External axis offset movements are independently planned with velocity and acceleration settings specified by the *[AXIS_L]OFFSET_AV_RATIO*. The offsetting motion is not coordinated with teleop jogs nor with coordinated (G-code) motion. During teleop jogging and coordinated (G-code) motion, axis soft limits (*[AXIS_*L*]MIN_LIMIT,MAX_LIMIT*) restrict movement of the axis.

When external offsets are applied and motion reaches a soft limit (by external offset increases or teleop jogging or coordinated motion), the HAL pin *motion.eoffset-limited* is asserted and the axis value is held nominally to the soft limit. This HAL pin can be used by associated HAL logic to truncate additional eoffset counts or to stop the machine (connect to *halui.machine.off* for instance). If the axis is moved within the soft limit, the *motion.eoffset-limited* pin is reset.

When operating at a soft limit during coordinated motion that continues to change the planned axis value, the HAL output pin *axis*.L.*eoffset* will indicate the current offset — the distance needed to reach the limit instead of the computed offset request. This indicated value will change as the planned axis value changes.

The HAL pin *axis*.L.*eoffset-request* indicates the current requested offset that is the product of the internal counts register and the eoffset-scale. In general, the *axis*.L.*eoffset* pin value lags the *axis*.L.*eoffset-request* value since the external offset is subject to an acceleration limit. When operating at a soft limit, additional updates to the *axis*.L.*eoffset-counts* will continue to affect the requested external offset as reflected in the *axis*.L.*eoffset-request* HAL pin.

When teleop jogging with external offsets enabled **and** non-zero values applied, encountering a soft limit will stop motion in the offending axis **without a deacceleration interval**. Similarly, during coordinated motion with external offsets enabled, reaching a soft limit will stop motion with no deacceleration phase. For this case, it does not matter if the offsets are zero.

When motion is stopped with no deacceleration phase, system **acceleration limits may be violated** and lead to: 1) a following error (and/or a thump) for a servo motor system, 2) a loss of steps for a stepper motor system. In general, it is recommended that external offsets are applied in a manner to avoid approaching soft limits.

#### 9.9.3.4 Notes

External offsets apply to axis coordinate letters (xyzabcuvw). All joints must be homed before external axis offsets are honored.

If an *axis*.L.*eoffset-enable* HAL pin is reset when its offset is non-zero, the offset is maintained. The offset may be cleared by:

1. a *Machine-off/Machine on* toggle
2. reactivating the enable pin and incrementing/decrementing the *axis*.L.*eoffset-counts* HAL pin to return the offset to zero.
3. pulsing the *axis*.L.*eoffset-clear* HAL pin

External-offsets are intended for use with *small* offsets that are applied within the soft-limit bounds.

Soft limits are respected for both teleop jogging and coordinated motion when external offsets are applied. However, when a soft limit is reached during coordinated motion, reducing the offending external offset **may not move away** from the soft limit **if planned motion continues in the same direction**. This circumstance can occur since the rate of correcting offset removal (as set by *[AXIS_L]OFFSET_AV_RATIO*) may be less than the opposing planned rate of motion. In such cases, **pausing** (or stopping) the planned, coordinated motion will allow movement away from the soft limit when correcting changes are made in the offending external offset.

#### 9.9.3.5 Warning

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

Component to compute an external offset from a function based on a measured angle (rotary coordinate or spindle). See the man page for details (**$ man eoffset_per_angle**).

### 9.9.5 Testing

The external axis offset capability is enabled by adding an *[AXIS_L]* setting for each candidate axis. For example:

```
[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.

In the absence of a custom display, **halshow** can be started as an auxiliary application with a custom watch list.

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
```

Where the file z.halshow (in the configuration directory) is:

```
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 Examples

Provided simulation configurations demonstrate the use of external offsets in order to provide a starting point for user customization for real hardware.

The sim configurations utilize the INI setting *[HAL]HALFILE = LIB:basic_sim.tcl* to configure all routine HAL connections for the axes specified in the INI file *[TRAJ]COORDINATES*= setting. The HAL logic needed to demonstrate external offset functionality and the GUI HAL pin connections for a PyVCP panel are made in separate HAL files. A non-simulation configuration should replace the *LIB:basic_sim.tcl* item HALFILEs appropriate to the machine. The provided PyVCP files (.hal and .xml) could be a starting point for application-specific GUI interfaces.

### 9.9.6.1 eoffsets.ini

The sim config *sim/configs/axis/external_offsets/eoffsets.ini* demonstrates a cartesian XYZ machine with controls to enable external offsets on any axis.

Displays are provided to show all important position and offset values.

A sim_pin GUI provides controls for the axis offset pins: eoffset-scale & eoffset-counts (via signal e:<L>counts), eoffset-clear (via signal e:clearall)

A script (eoffsets_monitor.tcl) is used to set *axis.L.counts* pins to zero at Machine-off.

### 9.9.6.2 jwp_z.ini

The sim config *sim/configs/axis/external_offsets/jwp_z.ini* demonstrates a jog-while-pause capability for a single (Z) coordinate:

Panel LEDs are provided to show important status items.

Controls are provided to set the eoffset scale factor and to increment/decrement/clear the eoffset counts.

### 9.9.6.3 dynamic_offsets.ini

This sim config *sim/configs/axis/external_offsets/dynamic_offsets.ini* demonstrates dynamically applied offsets by connecting a sinusoidal waveform to the z coordinate external offset inputs.

Panel LEDs are provided to show important status items.

Controls are provided to alter INI file settings for the Z axis max velocity and max acceleration.

Controls are provided to set the waveform generator parameters.

A halscope app is started to show the applied waveform, the offset response, and the motor cmd response.

---

**Note**
changes to the z coordinate max-acceleration and max-velocity are not acknowledged while a program is running.

---

#### 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 transverse (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.

Panel LEDs are provided to show important status items.

Functions are provided for inside and outside polygons (nsides >= 3), sine waves and square waves. The functions can be multiplied in frequency using the fmul pin and modified in amplitude using the rfrac pin (fraction of reference radius).

Controls are provided to start/stop offset waveforms and to set the function type and its parameters.

## 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 file Settings

INI file settings enable the (optional) operation of a user-provided tool database program:

```
[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.

---

**Note**
INI file settings for [EMCIO]TOOL_TABLE are ignored when a **db_program** is specified.

---

---

**Note**
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)

When a **db_progam** is specified, operation is as follows:

1. At startup, LinuxCNC starts the **db_program** and connects to its stdin and 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. The **db_program** then enters an event wait loop to receive commands that indicate that tool data has been changed by LinuxCNC. Tool data changes include:
   - a) spindle loading(T*n* M6)/unloading(T0 M6)
   - b) tool parameter changes (G10L1P*n* for example)
   - c) tool substitutions (M61Q*n*).

When a tool data change occurs, LinuxCNC sends a command to the **db_program** consisting of an identifying command letter followed by a full or abbreviated tool data line. The **db_program** must respond with a reply to confirm receipt. If the reply includes the text ”*NAK*”, a message is printed to stdout but execution continues. The ”*NAK*” message signifies a lack of synchronization between the **db_program** and LinuxCNC — accompanying text should give an indication for the cause of the fault.

The commands issued for tool data changes are:

- ”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.

---

**Note**
When a NON_RANDOM tool changer is specified using [EMCIO]RANDOM_TOOL_CHANGER=0 (the default), the spindle_load command issued for T*n*M6 (or M61Q*n*) is: *l T*n *P0* (pocket 0 is the spindle). The spindle_unload command issued for T0M6 is *u T0 P0*.

---

---

**Note**
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 Использование

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.

---

**Note**
Removal of a tool number should only be done if the tool number is not currently loaded in spindle.

---

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()
```

---

**Note**
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 Конфигурации моделирования

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. updates of tool parameters
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 Notes

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.

# Part II

# Использование

# Chapter 10

# Интерфейсы пользователя

## 10.1 AXIS ГИП

### 10.1.1 Введение

AXIS — это графический интерфейс для LinuxCNC, который имеет предварительный просмотр в реальном времени и backplot. Он написан на Python и использует Tk и OpenGL для отображения пользовательского интерфейса.

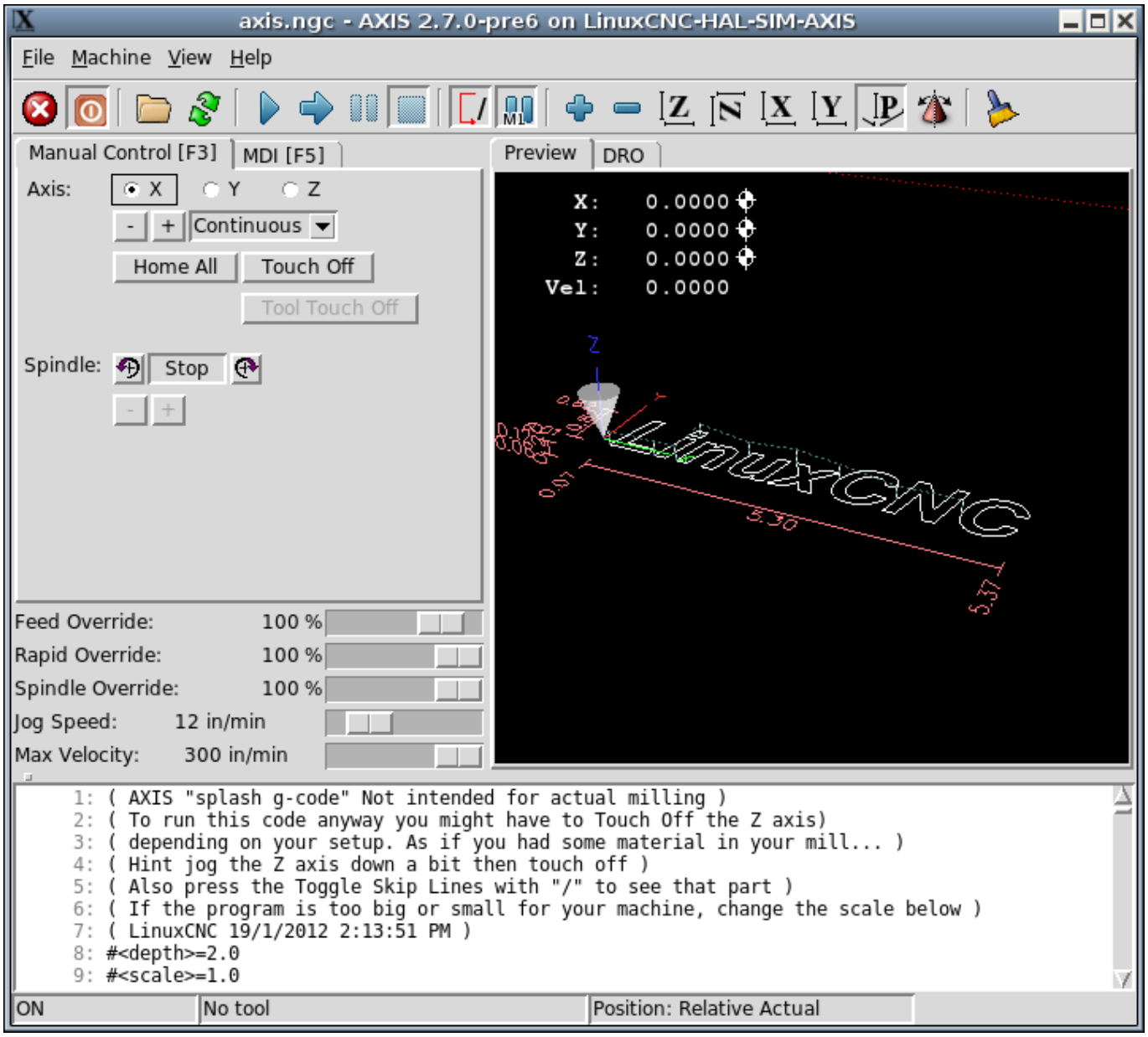



Figure 10.1: Окно AXIS

### 10.1.2 Начало

Если ваша конфигурация в настоящее время не настроена для использования AXIS, вы можете изменить ее, отредактировав файл .ini (INI-файл). В разделе `[DISPLAY]` измените строку `[DISPLAY]` на `DISPLAY = axis`.

Пример конфигурации *sim/axis.ini* уже настроен для использования AXIS в качестве внешнего интерфейса.

При запуске AXIS открывается окно, подобное показанному на рисунке Figure 10.1 выше.

#### 10.1.2.1 настройки INI

Для получения дополнительной информации о настройках файла INI, которые могут изменить работу AXIS, см. Display Section и Axis Section Главы конфигурации INI.

- *CYCLE_TIME* - Отрегулируйте скорость отклика ГИП в миллисекундах. Типичное значение 100, полезный диапазон 50–200.
  (по причинам, связанным с устаревшими версиями, время будет принято в секундах (0,05–0,2 – миллисекунды предпочтительнее для соответствия другим экранам).

```
[DISPLAY]
CYCLE_TIME = 100
```

- *PREVIEW_TIMEOUT* - Установите тайм-аут в секундах для загрузки предварительного просмотра G-кода. Если анализ G-кода длится дольше этого времени, отображается уведомление и на графическом дисплее отображается только начальная часть программы. Указание 0 или отказ от настройки приводит к отсутствию тайм-аута.

```
[DISPLAY]
PREVIEW_TIMEOUT = 5
```

#### 10.1.2.2 Типичная сессия

1. Запустите LinuxCNC и выберите файл конфигурации.
2. Отпустите кнопку аварийного останова (F1) и включите питание станка (F2).
3. Приведите все оси в исходное положение.
4. Загрузите файл G-кода.
5. Используйте график предварительного просмотра, чтобы убедиться в правильности программы.
6. Загрузите материал.
7. Установите правильное смещение для каждой оси, перемещая и используя кнопку `Touch Off` по мере необходимости.
8. Запустите программу.
9. Чтобы обработать тот же файл еще раз, вернитесь к шагу 6. Чтобы обработать другой файл, вернитесь к шагу 4.
10. Когда задание будет завершено, выйдите из AXIS.

---

**Note**
Повторный запуск той же программы зависит от ваших настроек и требований. Возможно, вам придется загрузить больше материала и установить смещения или перейти и установить смещение, а затем снова запустить программу. Если ваш материал зафиксирован, возможно, вам придется просто запустить программу еще раз. Дополнительную информацию о команде запуска см. в разделе Machine Menu.

---

### 10.1.3 Окно AXIS

Окно AXIS содержит следующие элементы:

- Область отображения, в которой отображается одно из следующего:
  - Предварительный просмотр загруженного файла (в данном случае *axis.ngc*), а также текущего местоположения *controlled point* станка с ЧПУ. Позже в этой области будет отображаться путь, по которому прошел станок с ЧПУ, называемый *backplot*.
  - Большой УЦИ, показывающий текущее положение и все смещения.
- Строка меню и панель инструментов, позволяющие выполнять различные действия
- *Manual Control Tab* - который позволяет вам заставить двигаться станок, включать или выключать шпиндель, а также включать или выключать подачу СОЖ, если она включена в INI-файл.
- *MDI Tab* - где программы G-кода можно вводить вручную, по одной строке за раз. Здесь также отображаются *Active G-codes*, которые показывают, какие модальные G-коды действуют.
- *Feed Override* - что позволяет масштабировать скорость запрограммированных движений. Максимал значение по умолчанию составляет 120 %, и в INI-файле можно установить другое значение. Дополнительную информацию см. в разделе Display файла INI.
- *Spindle Override* - что позволяет масштабировать скорость шпинделя вверх или вниз.
- *Jog Speed* - что позволяет установить скорость медленной подачи в пределах, заданных в INI-файле. Дополнительную информацию см. в разделе Display файла INI.
- *Max Velocity* - что позволяет ограничить максимальную скорость всех запрограммированных движений (кроме синхронизированного движения шпинделя).
- Область текстового отображения, показывающая загруженный G-код.
- Строка состояния, показывающая состояние станка. На этом снимке экрана станок включен, в него не вставлен инструмент, а отображаемое положение — *Relative* (показывающее все смещения) и *Actual* (показывающее положение обратной связи).

#### 10.1.3.1 Пункты меню

Некоторые пункты меню могут быть выделены серым цветом в зависимости от того, как настроен INI-файл. Для получения дополнительной информации о настройке см. INI Chapter.

- *Open...* - Открывает стандартное диалоговое окно для открытия файла G-кода для загрузки в AXIS. Если вы настроили LinuxCNC на использование программы-фильтра, вы также можете открыть ее. Дополнительную информацию см. в разделе FILTER конфигурации INI.
- *Recent Files* - Отображает список недавно открытых файлов.
- *Edit...* - Откройте текущий файл G-кода для редактирования, если в вашем INI-файле настроен редактор. См. раздел DISPLAY для получения дополнительной информации об указании используем редактора.
- *Reload* - перезагрузит текущий файл G-кода. Если вы его редактировали, вам необходимо перезагрузить его, чтобы изменения вступили в силу. Если вы остановили файл и хотите начать с начала, перезагрузите файл. Перезагрузка панели инструментов аналогична меню.
- *Save G-code as...* - Сохраните текущий файл под новым именем.
- *Properties* - Сумма быстрых и подающих перемещений. Не учитывает ускорение, смешивание или режим траектории, поэтому сообщаемое время никогда не будет меньше фактического времени выполнения.

- *Edit tool table...* - То же, что и Edit, если вы определили редактор, вы можете открыть таблицу инструментов и редактировать ее.
- *Reload tool table* - После редактирования таблицы инструментов необходимо перезагрузить ее.
- *Ladder editor* - Если вы загрузили ClassicLadder, вы можете редактировать его здесь. Дополнительную информацию см. в разделе ClassicLadder.
- *Quit* - Завершает текущий сеанс LinuxCNC.

- *Toggle Emergency Stop F1* - Изменяет состояние аварийной остановки.
- *Toggle Machine Power F2* - Изменяет состояние питания станка, если не включена аварийная остановка.
- *Run Program* - Запускает загруженную в данный момент программу с самого начала.
- *Run From Selected Line* - выбирает строку, с которой хотите начать. Используйте с осторожностью, так как это сначала переместит инструмент в ожидаемую позицию перед строкой, а затем выполнит остальную часть кода.

---

**Warning**
Не используйте *Run From Selected Line*, если ваша программа G-кода содержит подпрограммы.

---

- *Step* - Один шаг через программу.
- *Pause* - Приостановить программу.
- *Resume* - Возобновить работу после паузы.
- *Stop* - Остановить работающую программу. Если выбран запуск после остановки, программа запустится с начала.
- *Stop at M1* - Если достигнут M1 и это отмечено, выполнение программы остановится на строке M1. Нажмите Resume, чтобы продолжить.
- *Skip lines with "/"* - Если строка начинается с / и этот флажок установлен, строка будет пропущена.
- *Clear MDI history* - Очищает окно истории MDI.
- *Copy from MDI history* - Копирует историю MDI в буфер обмена
- *Paste to MDI history* - Вставка из буфера обмена в окно истории MDI
- *Calibration* - Запускает помощник калибровки (emccalib.tcl). При калибровке считывается файл HAL, и для каждого параметра *setp*, использующего переменную из файла INI, находящегося в разделе [AXIS_L], [JOINT_N], [SPINDLE_S] или [TUNE], создается запись, которую можно редактировать и тестировать. .
- *Show HAL Configuration* - Открывает окно конфигурации HAL, в котором вы можете отслеживать компоненты HAL, контакты, параметры, сигналы, функции и потоки.
- *HAL Meter* - Открывает окно, в котором вы можете отслеживать один HAL контакт, сигнал или параметр.
- *HAL Scope* - Открывает виртуальный осциллограф, который позволяет отображать зависимости значений HAL от времени.
- *Show LinuxCNC Status* - Открывает окно, показывающее статус LinuxCNC.

- *Set Debug Level* - Открывает окно, в котором можно просмотреть уровни отладки и настроить некоторые из них.
- *Homing* - Исходное положение одной или всех осей.
- *Unhoming* - Отменить исходное положение одной или всех осей.
- *Zero Coordinate System* - Установите все смещения на ноль в выбранной системе координат.
- Инструмент Touch Off
  - *Tool touch off to workpiece* - При выполнении Touch Off введенное значение относится к текущей системе координат заготовки (*G5x*), измененной смещением оси (*G92*). После завершения Touch Off Относительная координата выбранной оси станет введенным значением. См. G10 L10 в главе G-код.
  - *Tool touch off to fixture* - При выполнении Touch Off введенное значение относится к девятой (*G59.3*) системе координат, при этом смещение оси (*G92*) игнорируется. Это полезно, когда в фиксированном месте на станке имеется приспособление для касания инструмента, а девятая (*G59.3*) система координат установлена так, что острие инструмента нулевой длины находится в начале координат приспособления, когда Относительные координаты равны 0. См. G10 L11 в главе G-код.
- *Top View* - Top View (или вид Z) отображает G-код, смотрящий вдоль оси Z от положительного к отрицательному. Этот вид лучше всего подходит для просмотра X и Y.
- *Rotated Top View* - Rotated Top View (или повернутый вид по оси Z) также отображает G-код, смотрящий вдоль оси Z от положительного к отрицательному. Но иногда удобно отображать оси X и Y, повернутые на 90 градусов, чтобы лучше соответствовать дисплею. Этот вид также лучше всего подходит для просмотра X и Y.
- *Side View* - Вид сбоку (или вид X) отображает G-код, смотрящий вдоль оси X от положительного к отрицательному. Этот вид лучше всего подходит для просмотра Y и Z.
- *Front View* - Вид спереди (или вид Y) отображает G-код, смотрящий вдоль оси Y от отрицательного к положительному. Этот вид лучше всего подходит для просмотра X и Z.
- *Perspective View* - Перспективный вид (или вид P) отображает G-код, рассматривающий деталь с настраиваемой точки зрения, по умолчанию — X+, Y-, Z+. Положение регулируется с помощью мыши и переключателя перетаскивания/поворота. Это представление является компромиссным, и, хотя оно хорошо справляется с попыткой отобразить три (до девяти!) осей на двухмерном изображении, часто встречаются некоторые особенности, которые трудно увидеть, что требует изменения точки обзора. Этот вид лучше всего подходит, если вы хотите видеть все три (до девяти) осей одновременно.

**Точка зрения**

Меню выбора дисплея AXIS *View* относится к видам *Top*, *Front*, и *Side*. Эти термины верны, если ось Z станка с ЧПУ вертикальна, а положительная Z направлена вверх. Это справедливо для вертикальных фрезерных станков, которые, вероятно, являются наиболее популярным применением, а также почти для всех электроэрозионных станков и даже для вертикальных револьверных токарных станков, где деталь вращается под инструментом.

Термины *Top*, *Front*, и *Side* могут сбивать с толку на других станках с ЧПУ, таких как стандартный токарный станок, где ось Z горизонтальна, или горизонтальный фрезерный станок, опять же, где ось Z горизонтальна, или даже перевернутый вертикальный револьверный станок, на котором деталь вращается над инструментом, а положительное направление оси Z направлено вниз!

Просто помните, что положительная ось Z (почти) всегда находится вдали от детали. Поэтому ознакомьтесь с конструкцией вашего станка и интерпретируйте показания дисплея по мере необходимости.

- *Display Inches* - Установите масштаб дисплея AXIS в дюймах.
- *Display MM* - Установите масштаб дисплея AXIS в миллиметрах.
- *Show Program* - При желании дисплей предварительного просмотра загруженной программы G-кода можно полностью отключить.
- *Show Program Rapids* - На экране предварительного просмотра загруженной программы G-кода перемещения скорости подачи (G1,G2,G3) всегда будут отображаться белым цветом. А вот отображение быстрых ходов (G0) голубым цветом при желании можно отключить.
- *Alpha-blend Program* - Этот параметр облегчает просмотр сложных программ, но может привести к замедлению предварительного просмотра.
- *Show Live Plot* - При желании подсветку путей подачи (G1,G2,G3) при движении инструмента можно отключить.
- *Show Tool* - При желании отображение конуса/цилиндра инструмента можно отключить.
- *Show Extents* - При желании отображение размеров (максимального хода в каждом направлении оси) загруженной программы G-кода можно отключить.
- *Show Offsets* - Выбранное исходное положение смещения приспособления (G54-G59.3) может быть показано как набор из трех ортогональных линий, по одной красной, синей и зеленой. Это отображение начала смещения (или нуля приспособления) при желании можно отключить.
- *Show Machine Limits* - Максимальные пределы перемещения станка для каждой оси, установленные в INI-файле, показаны в виде прямоугольника, нарисованного красными пунктирными линиями. Это полезно при загрузке новой программы G-кода или при проверке того, какое смещение приспособления потребуется, чтобы привести программу G-кода в пределы перемещения вашего станка. Его можно отключить, если он не нужен.
- *Show Velocity* - Отображение скорости иногда полезно, чтобы увидеть, насколько близко ваш станок работает к расчетной скорости. При желании его можно отключить.
- *Show Distance to Go* - Оставшееся расстояние — очень полезный параметр, который нужно знать при первом запуске неизвестной программы с G-кодом. В сочетании с элементами управления быстрой коррекцией и коррекцией скорости подачи можно избежать нежелательного повреждения инструмента и станка. После отладки программы G-кода и ее бесперебойной работы отображение оставшегося пути при желании можно отключить.
- *Coordinates in large font…* - Предварительные координаты осей и скорость будут отображаться крупным шрифтом в представлении траектории инструмента.
- *Clear Live Plot* - По мере перемещения инструмента на дисплее AXIS путь G-кода подсвечивается. Чтобы повторить программу или лучше рассмотреть интересующую область, можно очистить ранее выделенные пути.
- *Show Commanded Position* - Это позиция, которую LinuxCNC попытается занять. Как только движение остановится, LinuxCNC попытается удержать эту позицию.
- *Show Actual Position* - Фактическое положение — это измеренное положение, считанное с энкодеров системы или смоделированное генераторами шагов. Это может немного отличаться от заданного положения по многим причинам, включая настройку ПИД-регулятора, физические ограничения или квантование положения.
- *Show Machine Position* - Это положение в несмещенных координатах, установленное приведением в исходную позицию.
- *Show Relative Position* - Это положение станка, измененное смещениями *G5x*, *G92*, и *G43*.
- *About AXIS* - Мы все знаем, что это такое.
- *Quick Reference* - Показывает сочетания клавиш клавиатуры.

#### 10.1.3.2 Кнопки панели инструментов

Слева направо на дисплее AXIS расположены кнопки панели инструментов (сочетания клавиш, показанные [в скобках]):

- Переключение аварийной остановки [F1] (также называется E-Stop)
- Переключение питания станка [F2]
- Открыть файл с G кодом [O]
- Перезагрузить текущий файл [Ctrl-R]
- Начать исполнение текущего файла [R]
- Выполнить следующую строку [T]
- Приостановить исполнение [P] Возобновить исполнение [S]
- Остановить выполнение программы [ESC]
- Переключение пропуска строк при помощи ”/” [Alt-M-/]
- Переключение опциональной паузы [Alt-M-1]
- Увеличить
- Уменьшить
- Вид сверху
- Повернутый вид сверху
- Вид сбоку
- Вид спереди
- Вид изометрия
- Переключение между режимом перетаскивания и вращения [D]
- Очистить текущий фон [Ctrl-K]

#### 10.1.3.3 Область графического отображения

**Отображение координат** В верхнем левом углу дисплея программы находится отображение положения координат для каждой оси. Справа от числа отображается символ исходного положения , если ось была перемещена в исходное положение.

Символ предела отображается справа от номера позиции координат, если ось находится на одном из концевых выключателей.

Чтобы правильно интерпретировать числа координат, обратитесь к индикатору *Position:* в строке состояния. Если позиция *Machine Actual*, то отображаемое число находится в системе координат станка. Если это *Relative Actual*, то отображаемое число находится в смещенной системе координат. Если отображаемые координаты являются относительными и установлено смещение, на дисплее будет отображаться голубой исходное положение станка маркер.

Если позиция *Commanded*, то отображаются точные координаты, заданные в команде G-кода. Если это *Actual*, то это положение, в которое машина фактически переместилась. Эти значения могут отличаться от заданного положения из-за ошибки рассогласования, зоны нечувствительности, разрешения энкодера или размера шага. Например, если вы задаете движение на X 0,0033 на вашем фрезерном станке, но один шаг вашего шагового двигателя или один отсчет энкодера равен 0,00125, тогда *Commanded* положение может быть 0,0033, а *Actual* положение будет 0,0025 (2 шага) или 0,00375 (3 шага).

**Предварительный просмотр графика** Когда файл загружен, его предварительный просмотр отображается в области отображения. Быстрые движения (например, выполняемые командой *G0*) показаны голубыми линиями. Перемещения со скоростью подачи (например, выполняемые командой G1) показаны сплошными белыми линиями. Задержки (например, созданные командой *G4*) отображаются в виде маленьких розовых отметок *X*.

Движения G0 (быстрые) перед перемещением подачи не будут отображаться на графике предварительн просмотра. Быстрые перемещения после T<n> (смена инструмента) не будут отображаться в предварительном просмотре до тех пор, пока не будет выполнено первое перемещение подачи. Чтобы отключить любую из этих функций, запрограммируйте G1 без каких-либо перемещений перед G0 перемещениями.

**Расширения программы** Показаны *расширения* программы по каждой оси. На концах указаны наименьшее и наибольшее значения координат. В середине показана разница между координатами.

Когда некоторые координаты превышают *программные пределы* в INI-файле, соответствующее измерение отображается другим цветом и заключено в рамку. На рисунке ниже максимальный программный предел превышен по оси X, как указано в рамке вокруг значения координаты. Минимальное перемещение X программы составляет -1,95, максимальное перемещение X составляет 1,88, а программа требует перемещения X 3,83 дюйма. Чтобы в этом случае переместить программу так, чтобы она находилась в пределах хода станка, выполните медленную подачу влево и снова сделайте касание (Touch Off) X.

Figure 10.2: Программные пределы

**Конус инструмента** Когда инструмент не загружен, положение кончика инструмента обозначается *конусом инструмента*. *Конус инструмента* не дает указаний относительно формы, длины или радиуса инструмента.

Когда инструмент загружается (например, с помощью команды MDI *T1 M6*), конус меняется на цилиндр, который показывает диаметр инструмента, указанный в файле таблицы инструментов.

**Траектория** Когда машина движется, она оставляет след, называемый траекторией. Цвет линии указывает тип движения: желтый для медленной подачи, бледно-зеленый для быстрых движений, красный для прямых движений со скоростью подачи и пурпурный для круговых движений со скоростью подачи.

**Сетка** AXIS может дополнительно отображать сетку в ортогональных видах. Включите или отключите сетку с помощью меню *Grid* в разделе *View*. Если этот параметр включен, сетка отображается в виде сверху и в повернутом виде сверху; когда система координат не повернута, сетка также отображается на видах спереди и сбоку. Пресеты в меню *Grid* контролируются элементом INI-файла `[DISPLAY]GRIDS`. Если не указано, по умолчанию используется значение `10 мм 20 мм 50 мм 100 мм 1 дюйм 2 дюйма 5 дюймов 10 дюймов`.

Указание очень маленькой сетки может снизить производительность.

**Взаимодействие** Если щелкнуть левой кнопкой мыши по части графика предварительного просмотра, линия будет выделена как в графическом, так и в текстовом отображении. Щелкнув левой кнопкой мыши по пустой области, подсветка будет снята.

При перетаскивании с нажатой левой кнопкой мыши график предварительного просмотра будет сдвинут (панорамирован).

При перетаскивании с нажатой левой кнопкой мыши или при перетаскивании с нажатым колесиком мыши график предварительного просмотра будет повернут. Когда линия выделена, центром вращения является центр линии. В противном случае центр вращения является центром всей программы.

Вращая колесо мыши, перетаскивая его с нажатой правой кнопкой мыши или перетаскивая с нажатой левой кнопкой мыши, можно увеличить или уменьшить масштаб графика предварительного просмотра.

Щелкнув одну из иконок *Preset View* или нажав кнопку *V*, можно выбрать несколько предустановленны видов.

#### 10.1.3.4 Область отображения текста

При щелчке левой кнопкой мыши по строке программы эта строка будет выделена как на графическом так и на текстовом дисплеях.

Когда программа запущена, строка, исполняемая в данный момент, выделяется красным цветом. Если пользователь не выбрал ни одну строку, текстовый дисплей автоматически прокрутится, чтобы отобразить текущую строку.

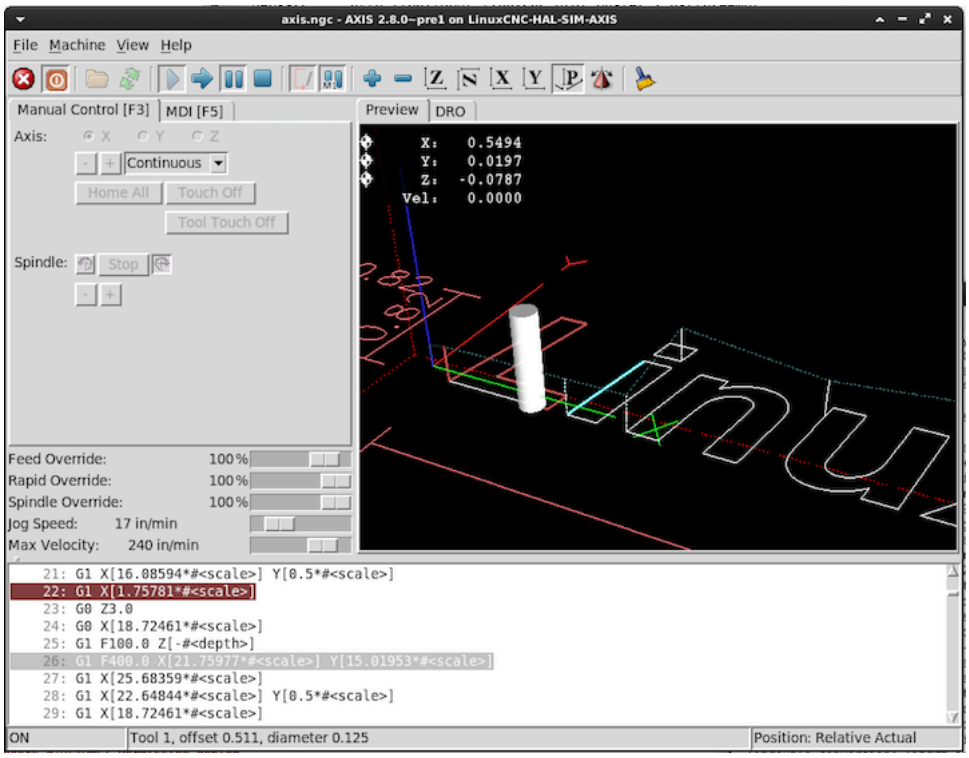



Figure 10.3: Текущие и выбранные строки

#### 10.1.3.5 Ручное управление

Когда станок включен, но не запущена программа, элементы на вкладке *Manual Control* можно использовать для перемещения станка или управления его шпинделем и СОЖ.

Когда станок не включен или когда запущена программа, ручное управление недоступно.

Многие из описанных ниже элементов полезны не на всех станках. Когда AXIS обнаруживает, что определенный контакт не подключен в HAL, соответствующий элемент на вкладке Manual Control удаляется. Например, если контакт HAL *spindle.0.brake* не подключен, кнопка *Brake* не появится на экране. Если установлена переменная среды *AXIS_NO_AUTOCONFIGURE*, это поведение отключено и все элементы появятся.

**Группа Axis** *AXIS* позволяет перемещать станок вручную. Это действие известно как *медленная подача* (*jogging*). Сначала выберите ось, которую нужно переместить, щелкнув на ней. Затем нажмите и удерживайте кнопку + или - в зависимости от желаемого направления движения. Первые четыре оси также можно перемещать с помощью клавиш со стрелками (X и Y), клавиш PAGE UP и PAGE DOWN (Z), а также клавиш [ и ] (A).

Если выбрано *Continuous*, движение будет продолжаться до тех пор, пока нажата кнопка или клавиша. Если выбрано другое значение, станок будет перемещаться точно на отображаемое расстояние при каждом нажатии кнопки или клавиши. По умолчанию доступны значения *0.1000, 0.0100, 0.0010, 0.0001*.

Дополнительную информацию о настройке приращений см. в разделе DISPLAY Section.

**Приведение в исходное положение (Identity Kinematics)** Параметр INI-файла [KINS]JOINTS определяет общее количество сочленений в системе. Сочленение может быть сконфигурировано с концевиком исходного положения или для *немедленного* возврата в исходное положение. Сочленения могут определять последовательность приведения в исходное положение, которая организует порядок для приведения в исходное положение групп сочленений.

Если **все** сочленения настроены для возврата в исходное положение и имеют действительные последовательности возврата в исходное положения, на кнопке возврата в исходное положение будет отображаться надпись *Home All*. Нажатие кнопки *Home All* (или клавиши Ctrl-HOME) инициирует возврат в исходное положение для всех сочленений, используя определенные для них последовательности возврата в исходное положение. Нажатие клавиши HOME приведет в исходное положение сочленение, соответствующее текущей выбранной оси, даже если последовательн возврата в исходное положение не определена.

Если не все оси имеют действительные последовательности возврата в исходное положение, на кнопке возврата отобразится *Home Axis*, и будет выполнено приведение в исходное положение сочленения только для выбранной в данный момент оси. Каждую ось необходимо выбирать и приводить в исходное положение отдельно.

Выпадающее меню Machine/Homing предоставляет альтернативный метод приведения в исходное положение осей. Выпадающее меню Machine/Unhoming предоставляет возможность для вывода осей из исходного положения.

Если на вашем станке в конфигурации не определены концевики исходного положения, кнопка *Home* установит текущее положение выбранной оси как абсолютное положение 0 для этой оси и установит бит *is-homed* для этой оси.

Дополнительную информацию см. в разделе Homing Configuration Chapter.

**Приведение в исходное положение (Non-Identity Kinematics)** Операция аналогична операции с Identity Kinematics, но перед возвратом в исходное положение кнопки с фиксацией выбирают сочленения по номеру. На кнопке возврата в исходное положение отобразится надпись *Home All*, если все сочленения настроены для приведения в исходное положение и имеют действительные последовательности приведения в исходное положение. В противном случае на кнопке возврата отобразится *Home Joint*.

Дополнительную информацию см. в разделе Homing Configuration Chapter.

**Касание**

При нажатии кнопки *Touch Off* или клавиши *END*, *G5x offset* для текущей оси изменяется так, что текущее значение оси будет заданной величиной. Выражения можно вводить, используя правила для программ rs274ngc, за исключением того, что нельзя ссылаться на переменные. Полученное значение отображается в виде числа.

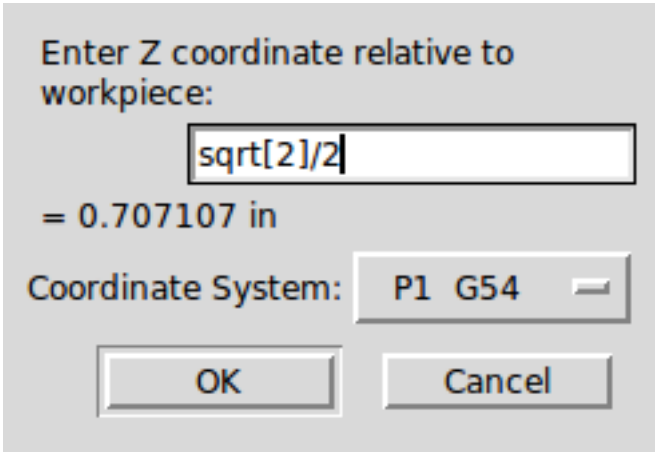



Figure 10.4: Окно касания

См. также пункты меню Machine: *Touch part* и *Touch part holder*.

**Текущая позиция Touch Off** An axis may be configured in the .INI file to incorporate the actual position value for an axis into the touch off calculation, either adding or subtracting this value. This

is primarily useful for machines which have a non-motorized axis such as a quill with an encoder. When this feature is enabled for an axis, the title bar of the touch off window will indicate **(system ACTUAL)**.

Дополнительную информацию смотрите Section 10.1.12.11 .

**Инструмент Touch Off** При нажатии кнопки *Tool Touch Off* длина и смещения текущего загруженного инструмента будут изменены так, чтобы текущее положение кончика инструмента соответствовало введенной координате.

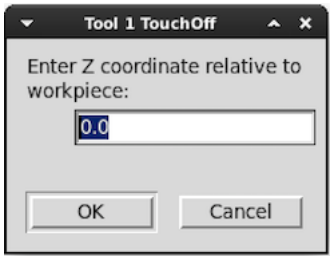



Figure 10.5: Окно инструмента касания

См. также параметры *Прикосновение инструмента к заготовке* и *Прикосновение инструмента к приспособлению* в меню *Станок*.

**Переопределение пределов** Нажав кнопку Override Limits, станку будет разрешена временно медленная подача от физического концевого выключателя. Этот флажок доступен только при срабатывании концевого выключателя. Коррекция сбрасывается после одной подачи. Если ось настроена с отдельными положительными и отрицательными концевыми выключателями, LinuxCNC разрешит медленную подачу только в правильном направлении. *Override Limits не позволит медленную подачу за программный предел. Единственный способ запретить программны предел на оси — вывести его из исходного положения.*

**Группа шпинделя** Кнопки первого ряда выбирают направление вращения шпинделя: Counterclockwise, Stopped, Clockwise. Counterclockwise будет отображаться только в том случае, если контакт *spindle.0.reverse* есть в файле HAL (это может быть *net Trick-axis spindle.0.reverse*). Кнопки следующего ряда увеличивают или уменьшают скорость вращения. Флажок в третьем ряду позволяет включить или отпустить тормоз шпинделя. В зависимости от конфигурации вашего компьютера могут отображаться не все элементы этой группы. Нажатие кнопки запуска шпинделя устанавливает скорость *S* на 1.

**Группа СОЖ** Две кнопки позволяют включать и выключать СОЖ *Mist* и *Flood*. В зависимости от конфигурации вашего компьютера могут отображаться не все элементы этой группы.

#### 10.1.3.6 MDI

MDI позволяет вводить команды G-кода вручную. Когда станок не включен или когда запущена программа, элементы управления MDI недоступны.

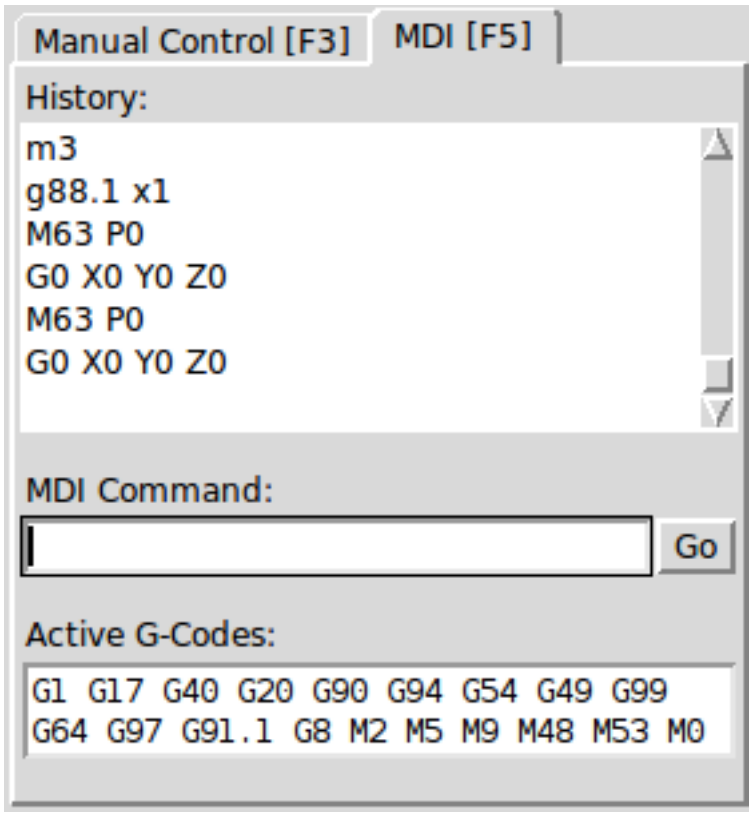



Figure 10.6: Вкладка MDI

- *History* - тут отображаются команды MDI, которые были набраны ранее в этом сеансе.
- *MDI Command* - здесь возможно ввести команду G-кода для выполнения. Выполните команду, нажав Enter или нажав *Go*.
- *Active G-codes* - тут отображаются *модальные коды*, активные в интерпретаторе. Например, *G54* указывает, что *G54 offset* применяется ко всем введенным координатам. В режиме *Авто* активные G-коды представляют собой коды после любого считывания интерпретатором.

#### 10.1.3.7 Переопределение подачи

Перемещая этот ползунок, можно изменить запрограммированную скорость подачи. Например, если программа запрашивает *F60*, а ползунок установлен на 120%, то результирующая скорость подачи будет равна 72.

#### 10.1.3.8 Spindle Speed Override

Перемещая этот ползунок, можно изменить запрограммированную скорость шпинделя. Например, если программа запрашивает S8000 и ползунок установлен на 80%, результирующая скорость шпинделя будет 6400. Этот элемент появляется только тогда, когда контакт HAL *spindle.0.speed-out* подключен.

#### 10.1.3.9 Скорость медленной подачи

Перемещая этот ползунок, можно изменить скорость медленной подачи. Например, если ползунок установлен на 1 дюйм/мин, то шаг на 0,01 дюйма будет выполнен примерно за 0,6 секунды или

1/100 минуты. С левой стороны (очень медленная подача) значения расположены близко друг к другу, тогда как с правой стороны (быстрая медленная подача) они расположены гораздо дальше друг от друга, что обеспечивает широкий диапазон скоростей медленной подачи с точным контролем, когда это наиболее важно.

На станках с поворотной осью показан второй ползунок скорости медленной подачи. Этот ползунок устанавливает скорость вращения поворотных осей (A, B и C).

#### 10.1.3.10 Макс скорость

Перемещая этот ползунок, можно установить максимальную скорость. Это ограничивает максимальну скорость для всех запрограммированных перемещений, кроме перемещений, синхронизированных со шпинделем.

### 10.1.4 Управление клавиатурой

Почти все действия в AXIS можно выполнить с помощью клавиатуры. Полный список сочетаний клавиш можно найти в Кратком справочнике AXIS, который можно просмотреть, выбрав Help > Quick Reference. Многие сочетания клавиш недоступны в режиме MDI.

#### 10.1.4.1 Клавиши переопределения подачи

**Note**
Подробную информацию о испанской раскладке клавиатуры можно найти в переведенной документации.

Клавиши переопределения подачи ведут себя по-другому в ручном режиме. Клавиши *12345678* выберут ось, если она запрограммирована. Если у вас 3 оси, то 0 выберет ось 0, 1 выберет ось 1, а 2 выберет ось 2. Остальные цифровые клавиши по-прежнему будут устанавливать коррекцию подачи. При запуске программы *1234567890* установит значение переопределение подачи 0–100%.

Наиболее часто используемые сочетания клавиш показаны в следующей таблице:

Table 10.1: Наиболее распространенные сочетания клавиш

| Нажатие клавиши | Действия | Mode |
|---|---|---|
| F1 | Переключает аварийный останов | Любой |
| F2 | Станок вкл/выкл | Любой |
| `, 1 .. 9, 0 | Устанавливает переопределение подачи от 0% до 100% | Варьируется |
| X, ` | Активирует первую ось | Ручной |
| Y, 1 | Активирует вторую ось | Ручной |
| Z, 2 | Активирует третью ось | Ручной |
| A, 3 | Активирует четвертую ось | Ручной |

Table 10.1: (continued)

| Нажатие клавиши | Действия | Mode |
|---|---|---|
| I | Устанавливает приращение медленной подачи | Ручной |
| C | Непрерывная медленная подача | Ручной |
| Control-Home | Выполнить последовательность приведения в исходное положение | Ручной |
| End | Touch off: Устанавливает G5x смещение для активой оси | Ручной |
| Left, Right | Медленная подача первой оси | Ручной |
| Up, Down | Медленная подача второй оси | Ручной |
| Pg Up, Pg Dn | Медленная подача третьей оси | Ручной |
| [, ] | Медленная подача четвертой оси | Ручной |
| O | Открыть файл | Ручной |
| Control-R | Перезагрузить файл | Ручной |
| R | Запустить файл | Ручной |
| P | Приостановить исполнение | Автоматический |
| S | Возобновить исполнение | Автоматический |
| ESC | Остановить исполнение | Автоматический |
| Control-K | Очистить траекторию | Автоматический/Ручной |
| V | Переключаться между предустановленными видами | Автоматический/Ручной |
| Shift-Left,Right | Быстро ось X | Ручной |
| Shift-Up,Down | Быстро ось Y | Ручной |
| Shift-PgUp, PgDn | Быстро ось Z | Ручной |
| @ | Переключить Текущее/Заданное | Любой |
| # | Переключить относительные/станочные | Любой |

### 10.1.5 Показать статус LinuxCNC (`linuxcnctop`)

В состав AXIS входит программа под названием *linuxcnctop*, которая показывает некоторые детали состояния LinuxCNC. Вы можете запустить эту программу, вызвав Machine > Show LinuxCNC Status

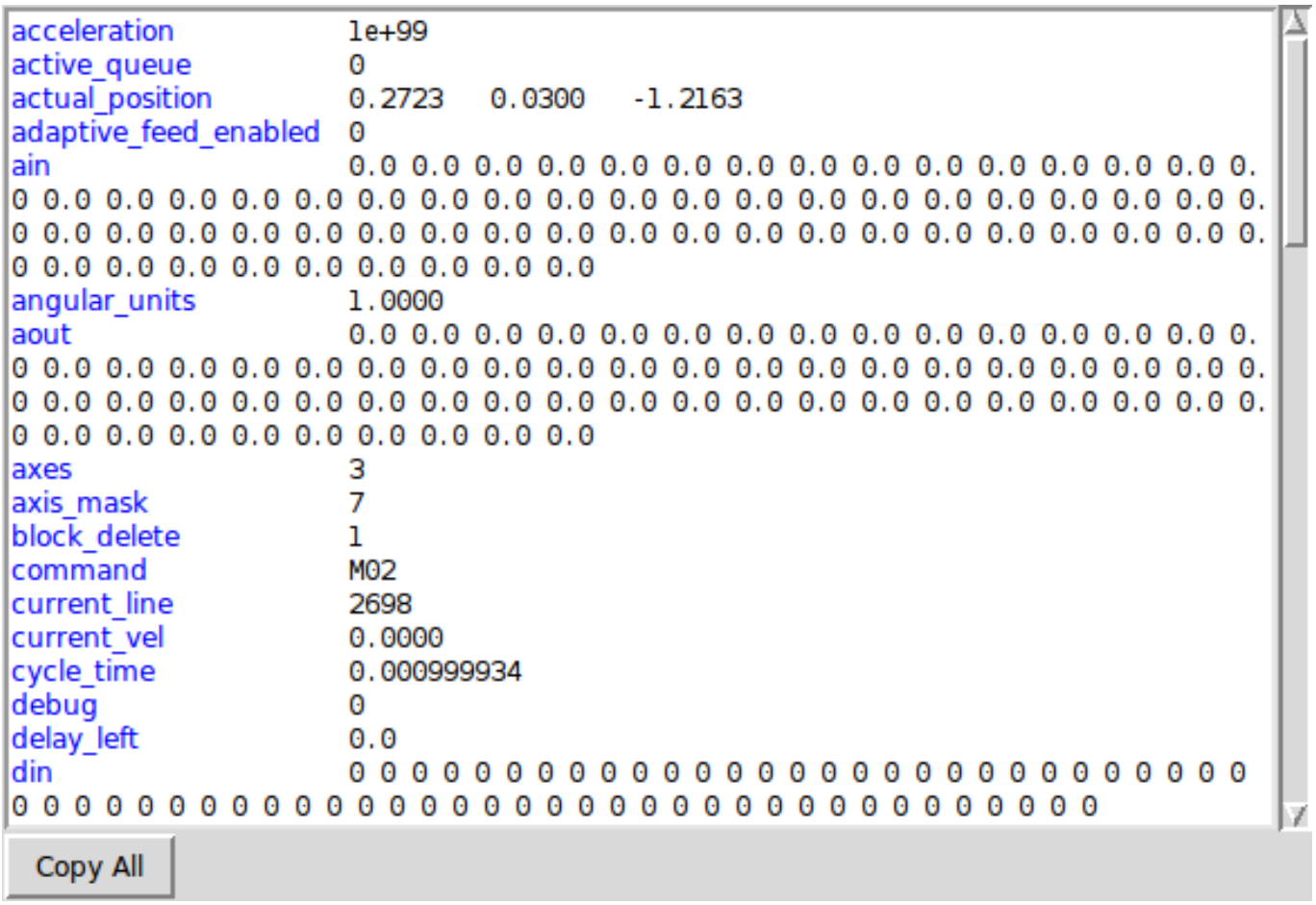



Figure 10.7: Окно статус LinuxCNC

Название каждого элемента отображается в левом столбце. Текущее значение отображается в правом столбце. Если значение недавно изменилось, оно отображается на красном фоне.

### 10.1.6 MDI интерфейс

В состав AXIS входит программа под названием `mdi`, сокращения для ”ручной ввод информации”, которая позволяет вводить команды MDI в текстовом режиме в работающий сеанс LinuxCNC. Вы можете запустить эту программу, напрямую с командной строки UNIX, открыв терминал и набрав:

```
mdi
```

После запуска отображается приглашение *MDI*>. При вводе пустой строки отображается текущее положение станка. Когда команда вводится, она отправляется в LinuxCNC для выполнения.

Это пример сеанса MDI:

```
$ mdi
MDI>
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI> G1 F5 X1
MDI>
(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI>
```

```
(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)
```

### 10.1.7 axis-remote

В состав AXIS входит программа под названием *axis-remote*, которая может отправлять определенные команды работающему AXIS. Доступные команды отображаются при запуске *axis-remote --help* и включают проверку работы AXIS (*--ping*), загрузку файла по имени, перезагрузку текущего загруженного файла (*--reload*) и выполнение выхода из AXIS (*--quit*).

### 10.1.8 Manual Tool Change

LinuxCNC включает компонент HAL, работающий не в режиме реального времени, под названием *hal_manualtoolchange*, который показывает окно с подсказкой, сообщающее вам, какой инструмент ожидается при вводе команды *M6*. После нажатия кнопки OK выполнение программы продолжится.

Компонент hal_manualtoolchange включает в себя контакт HAL для кнопки, которую можно подключит к физической кнопке, чтобы завершить смену инструмента и удалить подсказку окна (hal_manualtoolch

Файл конфигурации HAL *lib/hallib/axis_manualtoolchange.hal* показывает команды HAL, необходимые для использования этого компонента.

hal_manualtoolchange можно использовать, даже если AXIS не используется в качестве графического интерфейса. Этот компонент наиболее полезен, если у вас есть предустановленные инструменты и вы используете таблицу инструментов.

---

**Note**
Важное примечание: быстрые перемещения не будут отображаться в предварительном просмотре после выдачи T<n> до следующего хода подачи после M6. Это может сбить с толку большинство пользователей. Чтобы отключить эту функцию для текущей программы смены инструмента, введите G1 без перемещения после T<n>.

---

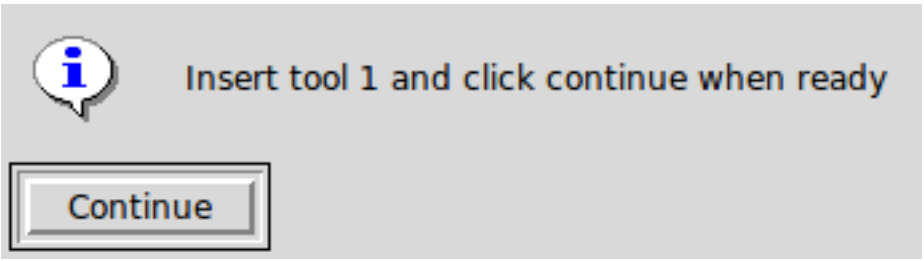



Figure 10.8: Окно ручной смены инструмента

### 10.1.9 Python Модули

AXIS включает в себя несколько модулей Python, которые могут быть полезны другим. Для получения дополнительной информации об одном из этих модулей используйте *pydoc <имя модуля>* или прочитайте исходный код. Эти модули включают в себя:

- *emc* обеспечивает доступ к каналам команд, состояний и ошибок LinuxCNC
- *gcode* предоставляет доступ к интерпретатору rs274ngc

- *rs274* предоставляет дополнительные инструменты для работы с файлами rs274ngc
- *hal* позволяет создавать компоненты HAL не реального времени, написанные на Python
- *_togl* предоставляет виджет OpenGL, который можно использовать в приложениях Tkinter

Чтобы использовать эти модули в своих собственных скриптах, вы должны убедиться, что каталог, в котором они находятся, находится в пути к модулю Python. При запуске установленной версии LinuxCNC это должно происходить автоматически. При запуске «на месте» это можно сделать с помощью *scripts/rip-environment*.

### 10.1.10 Использование AXIS в режиме токарного станка

Включив строку *LATHE = 1* в раздел [DISPLAY] INI-файла, AXIS выбирает режим токарного станка. Ось *Y* не отображается в показаниях координат, вид изменяется, чтобы показать ось Z, уходящую вправо, и ось X, уходящую к нижней части экрана, а некоторые элементы управления (например, для предустановленных видов) удалены. Показания координат для X заменяются диаметром и радиусом.

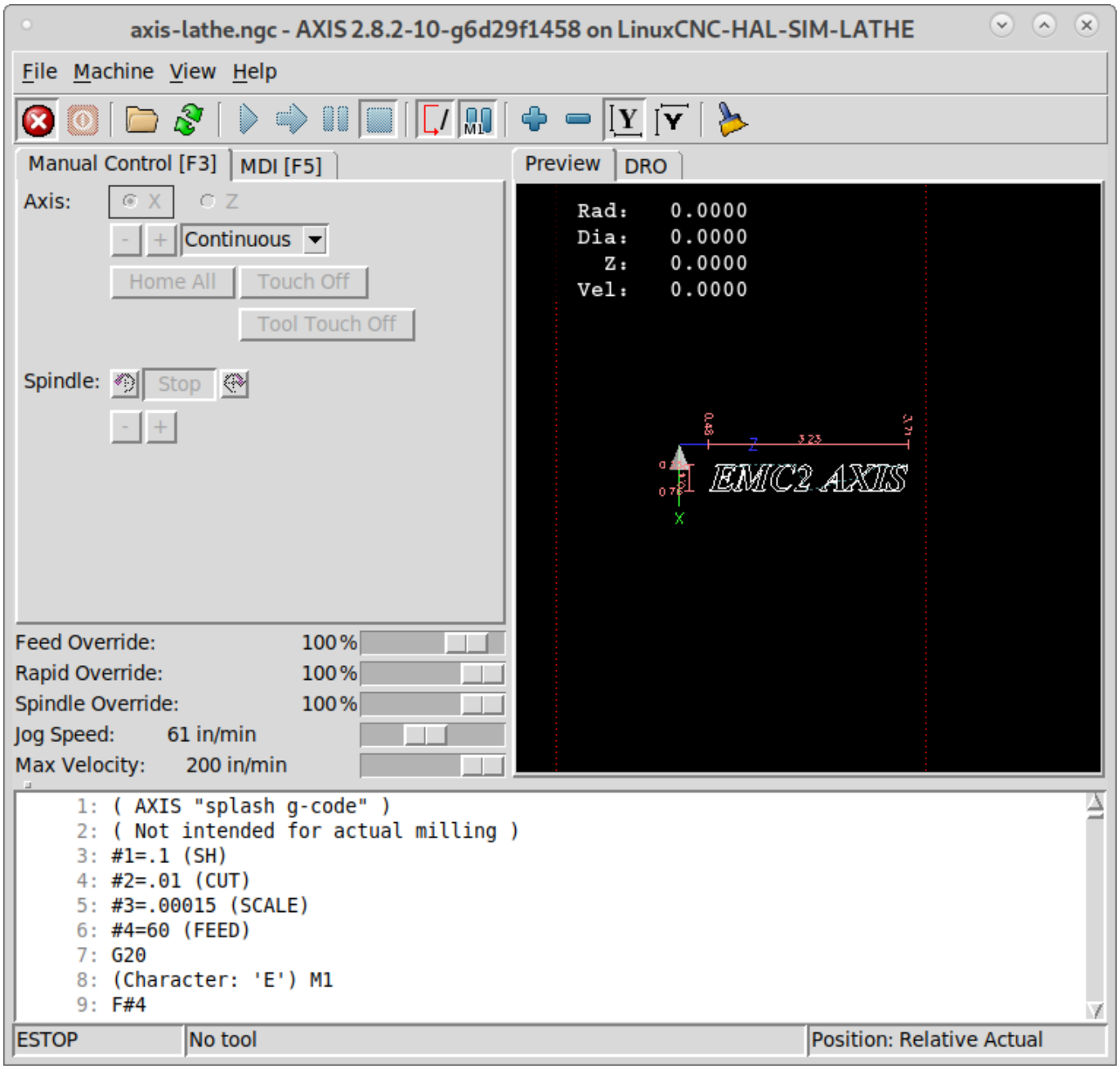



Figure 10.9: Режим токарного станка AXIS

Нажатие клавиши *V* уменьшает масштаб, чтобы отобразить весь файл, если он загружен.

В режиме токарного станка отображается форма загруженного инструмента (если есть).

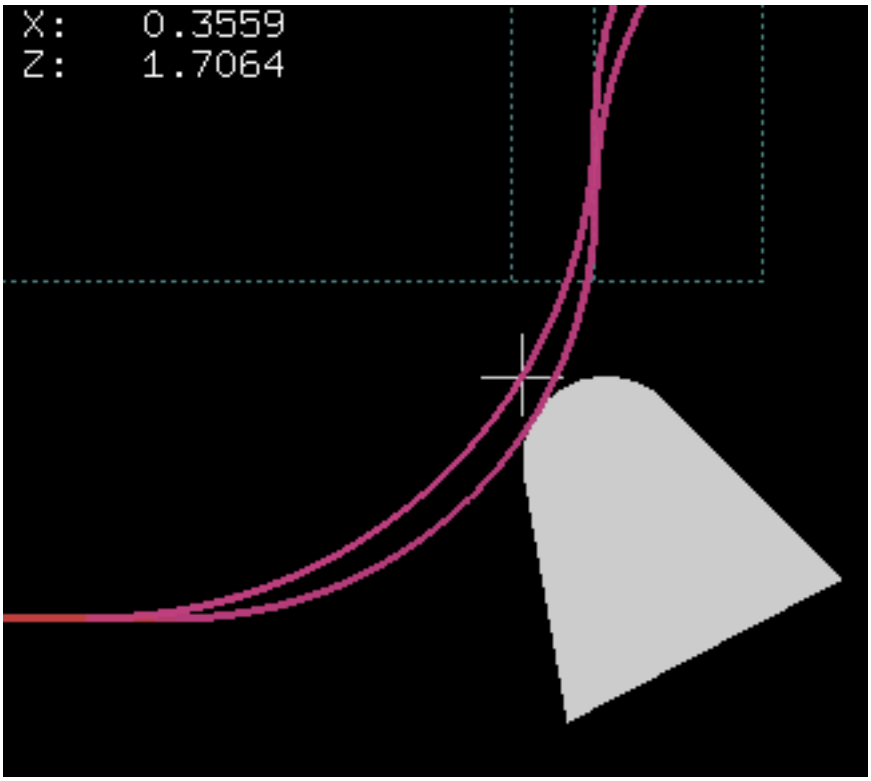



Figure 10.10: Форма инструмента токарного станка

Чтобы изменить отображение на токарный станок с задним инструментом, вам необходимо иметь как *LATHE = 1*, так и *BACK_TOOL_LATHE = 1* в разделе [DISPLAY]. Это инвертирует вид и поместит инструмент на обратную сторону оси Z.

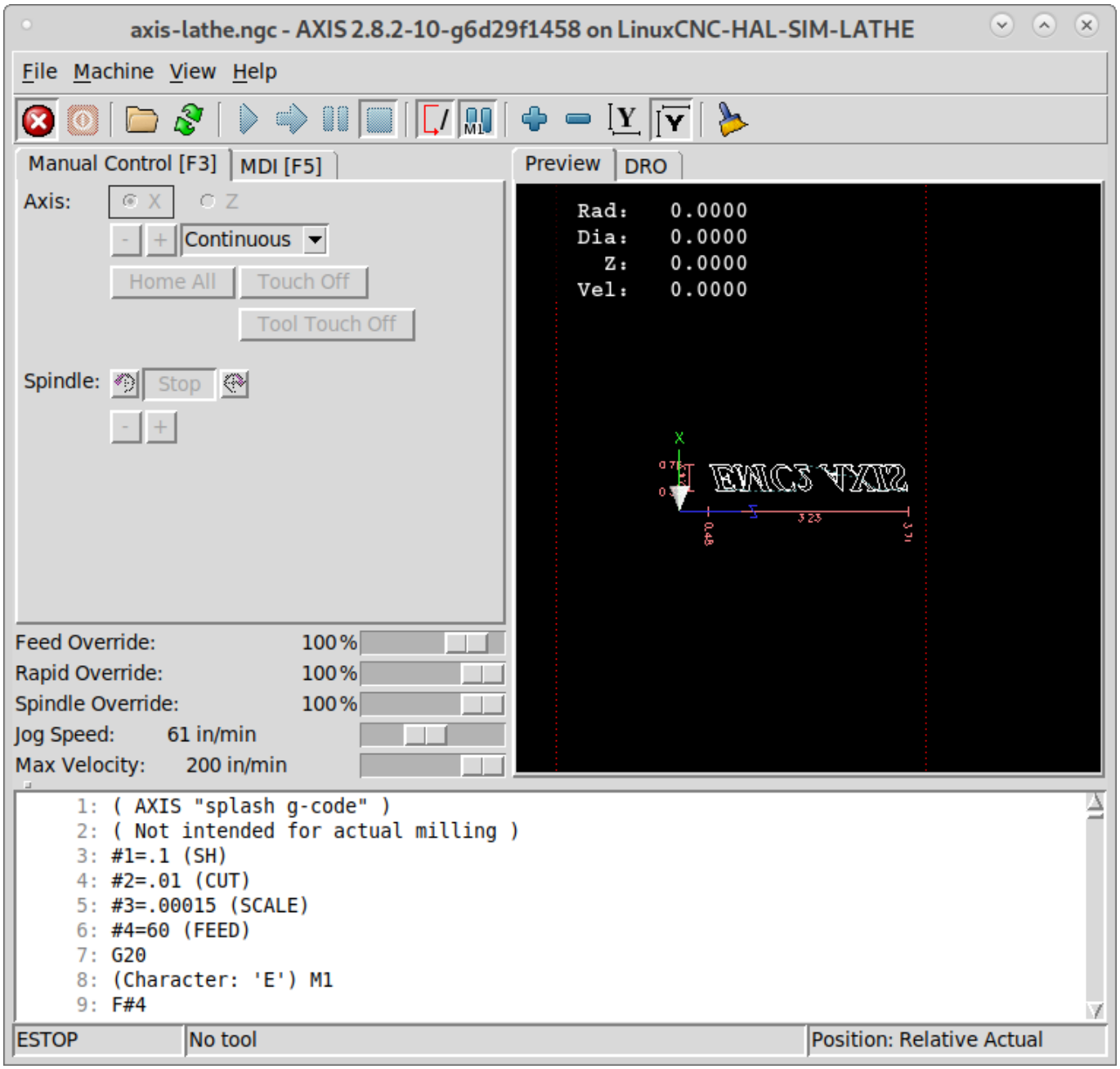



Figure 10.11: Форма заднего инструмента токарного станка

### 10.1.11 Использование AXIS в режиме резки вспененого материала

Включив строку *FOAM = 1* в раздел [DISPLAY] INI-файла, выбирает режим резки вспененого материала AXIS. В предварительном просмотре программы движения XY отображаются в одной плоскости, а UV-движения — в другой. На графике линии рисуются между соответствующими точками на плоскости XY и плоскости UV. Специальные комментарии (XY_Z_POS) и (UV_Z_POS) устанавливают координаты Z этих плоскостей, которые по умолчанию равны 0 и 1,5 станочным единицам.

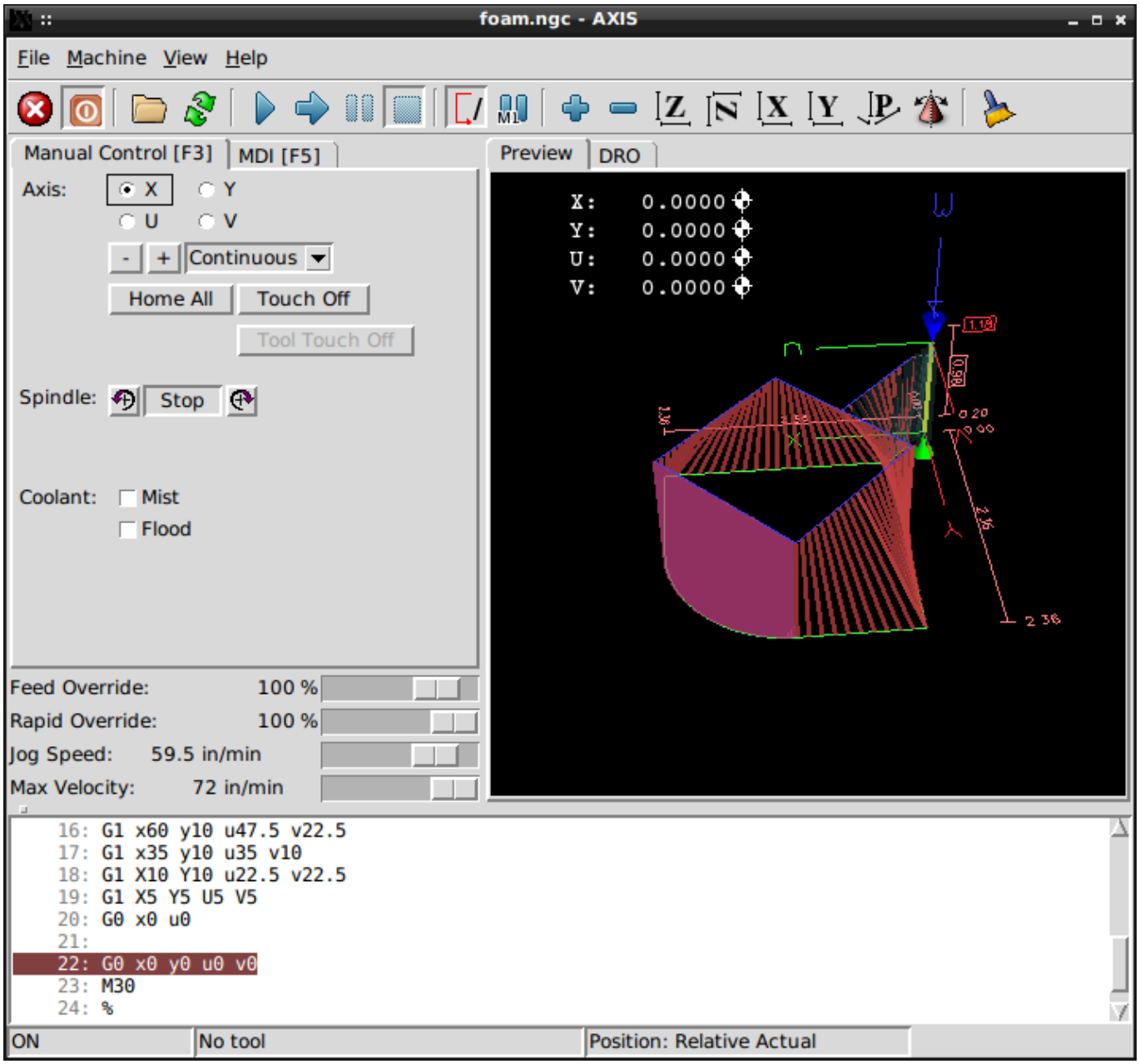



Figure 10.12: Режим резки вспененного материала

### 10.1.12 Расширенная конфигурация

При запуске AXIS создает контакты HAL для графического пользовательского интерфейса, а затем выполняет файл HAL, указанный в файле INI: *[HAL]POSTGUI_HALFILE=<filename>*. Обычно *<filename>* представляет собой базовое имя конфигурации + *_postgui* + *.hal*, например. *lathe_postgui.h* но может быть любым допустимым именем файла. Эти команды выполняются после построения экрана, гарантируя доступность контактов HAL виджета. Вы можете иметь несколько строк *POSTGUI_HALFILE=<filename>* в INI. Каждая из них будет запускаться одна за другой в том порядке, в котором они появляются.

Дополнительную информацию о настройках файла INI, которые могут изменить способ работы AXIS, см. в разделе Display Section главы о настройке INI.

#### 10.1.12.1 Программные фильтры

AXIS имеет возможность пропускать загруженные файлы через *программу-фильтр*. Этот фильтр может выполнять любую желаемую задачу: что-то простое, например проверка того, что файл заканчивается на *M2*, или что-то более сложное, например создание G-кода из изображения.

Раздел *[FILTER]* INI-файла управляет работой фильтров. Сначала для каждого типа файла напишите строку *PROGRAM_EXTENSION*. Затем укажите программу, которая будет выполняться для каждого типа файла. Этой программе присваивается имя входного файла в качестве первого аргумента, и она должна записать код rs274ngc в стандартный вывод. Этот вывод будет отображаться в текстовой области, просматриваться в области отображения и выполняться LinuxCNC при нажатии кнопки *Run*. Следующие строки добавляют поддержку конвертера *image-to-gcode*, включенно в LinuxCNC:

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

Также возможно указать интерпретатор:

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

Таким образом, любой скрипт Python может быть открыт, а его выходные данные обрабатываются как G-код. Один из таких примеров скрипта доступен по адресу *nc_files/holecircle.py*. Этот скрипт создает G-код для сверления серии отверстий по окружности.

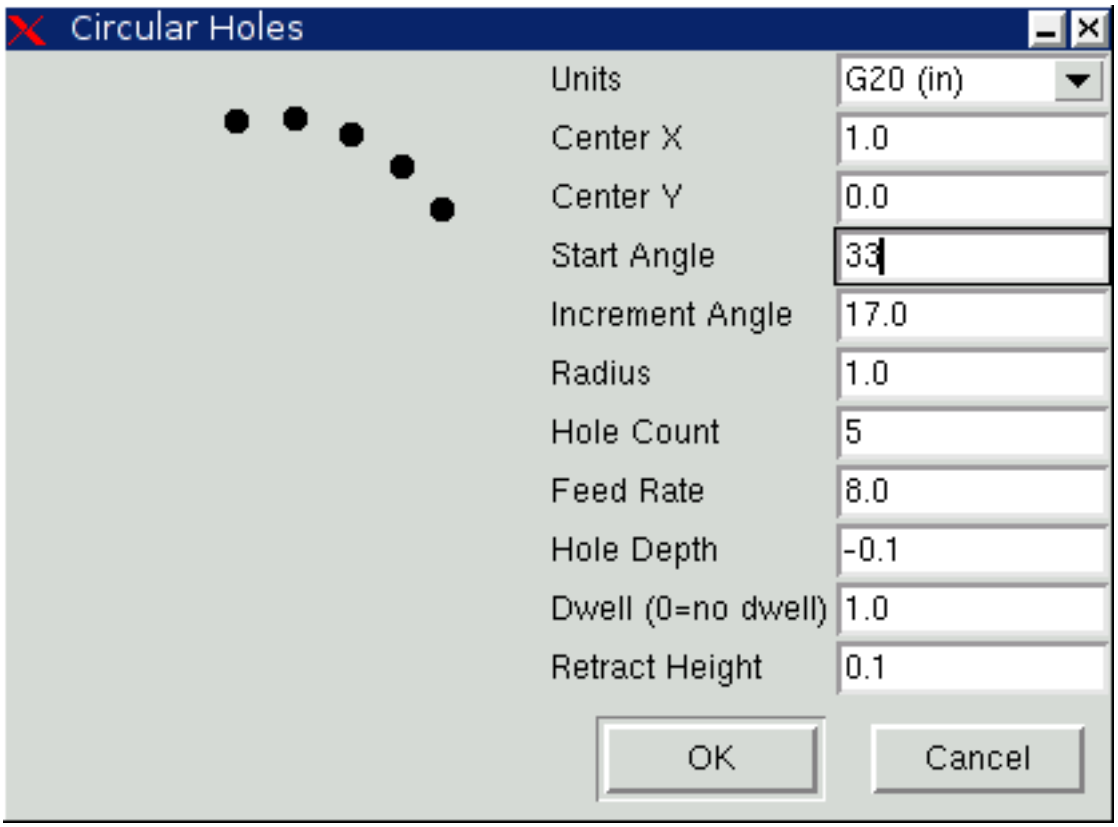



Figure 10.13: Отверстия по кругу

Если установлена переменная среды AXIS_PROGRESS_BAR, то в stderr записываются строки вида

```
FILTER_PROGRESS=%d
```

установит индикатор выполнения AXIS на заданный процент. Эту функцию следует использовать любому фильтру, работающему в течение длительного времени.

#### 10.1.12.2 База данных ресурсов X (X-windows)

Цвета большинства элементов пользовательского интерфейса AXIS можно настроить с помощью базы данных X Resource Database. Образец файла *axis_light_background* изменяет цвета окна траектории на схему *темные линии на белом фоне*, а также служит ссылкой для настраиваемых элементов в области отображения. В образце файла *axis_big_dro* значение положения изменяется на шрифт большего размера. Чтобы использовать эти файлы:

```
xrdb -merge /usr/share/doc/emc2/axis_light_background

xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

Информацию о других элементах, которые можно настроить в приложениях Tk, см. на страницах руководства Tk.

Поскольку современные среды рабочего стола автоматически вносят в базу данных ресурсов X некоторые настройки, которые отрицательно влияют на AXIS, то по умолчанию эти настройки игнорируются. Чтобы элементы базы данных X Resource переопределяли значения по умолчанию AXIS, включите следующую строку в свои X Resources:

```
*AXIS*optionLevel: widgetDefault
```

это приводит к созданию встроенных параметров на уровне параметра *widgetDefault*, чтобы ресурсы X (которые имеют уровень *userDefault*) могли их переопределить.

#### 10.1.12.3 Безупорный регулятор

Чтобы улучшить взаимодействие AXIS с физическим маховичком, текущая активная ось, выбранная в графическом интерфейсе, также выводится на *контакт HAL* с именем типа *axisui.jog.x*. За исключением короткого времени после изменения текущей оси, только один из этих контаков одновременно имеет значение *TRUE*, остальные остаются *FALSE*.

После того как AXIS создала эти *контакты HAL*, она запускает файл HAL, объявленный с помощью: [HAL]POSTGUI_HALFILE. Чем отличается от [HAL]HALFILE, который можно использовать только один раз.

#### 10.1.12.4 ~/.axisrc

Если он существует, содержимое `~/.axisrc` выполняется как исходный код Python непосредственно перед отображением ГИП AXIS. Детали того, что может быть записано в `~/.axisrc`, могут быть изменены в ходе цикла разработки.

Следующее добавляет Control-Q в качестве сочетания клавиш для выхода.

**Пример файла .axisrc**

```
root_window.bind("<Control-q>", "destroy .")
help2.append(("Control-Q", "Quit"))
```

Следующее действие останавливает диалоговое окно "Do you really want to quit".

```
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

#### 10.1.12.5 USER_COMMAND_FILE

Файл Python для конкретной конфигурации можно указать с помощью параметра INI-файла *[DISPLAY]USER_COMMAND_FILE=filename.py*. Как и файл `~/.axisrc`, этот файл создается непосредственно перед отображением графического интерфейса AXIS. Этот файл относится к конфигурации файла INI, а не к домашнему каталогу пользователя.

#### 10.1.12.6 user_live_update()

Графический интерфейс AXIS включает недействующую функцию (заполнитель) с именем user_live_update(), которая выполняется по завершении функции update() своего класса LivePlotter. Эту функцию можно реализовать в скрипте Python `~/.axisrc` или скрипте Python *[DISPLAY]USER_COMMAND_FILE* для выполнения пользовательских периодических действий. Подробности того, что может быть выполнено с помощью этой функции, зависят от реализации ГИП AXIS и могут быть изменены в ходе цикла разработки.

#### 10.1.12.7 user_hal_pins()

Графический интерфейс AXIS включает в себя неиспользуемую функцию (заполнитель) с именем *user_hal_pins()*.
Он выполняется сразу после вызова файла .axisrc и непосредственно перед инициализацией любых панелей/встроенных вкладок GladeVCP.
Эту функцию можно реализовать в скрипте Python `~/.axisrc` или скрипте Python *[DISPLAY]USER_COM*
для создания пользовательских выводов HAL, использующих *axisui.* префикс.
Используйте *comp* в качестве ссылки на экземпляр компонента HAL.
HAL comp.ready() вызывается сразу после возврата из этой функции.

#### 10.1.12.8 Внешний редактор

Пункты меню File > Edit... and File > Edit Tool Table... становятся доступными после определения редактора в разделе INI [DISPLAY]. Полезные значения включают `EDITOR=gedit` и `EDITOR=gnome-termi -e vim`. Дополнительную информацию см. в разделе «sub:ini:sec:display,Display» главы Конфигурация INI.

#### 10.1.12.9 Виртуальная панель управления

AXIS может отображать пользовательскую виртуальную панель управления либо в правом столбце, либо в нижней строке. Кроме того, одна или несколько панелей могут отображаться как встроенные вкладки. Вы можете запрограммировать кнопки, индикаторы, отображение данных и многое другое. Дополнительную информацию см. в главах PyVCP и GladeVCP.

#### 10.1.12.10 Управление предварительным просмотром

В файл G-кода можно вставлять специальные комментарии, чтобы контролировать поведение предварительного просмотра AXIS. В случае, если вы хотите ограничить прорисовку предварительного просмотра, используйте эти специальные комментарии. Все, что находится между (AXIS,hide) и (AXIS,show), не будет отображаться во время предварительного просмотра. (AXIS,hide) и (AXIS,show) должны использоваться парами, при этом (AXIS,hide) идет первым. Все, что находится после (AXIS,stop), не будет отображаться во время предварительного просмотра.

Эти комментарии полезны для упрощения отображения предварительного просмотра (например, при отладке большого файла G-кода можно отключить предварительный просмотр для определенных частей, которые уже работают нормально).

- (AXIS,hide) Останавливает предварительный просмотр (должен быть первым)
- (AXIS,show) Возобновляет предварительный просмотр (должен следовать за скрытием)
- (AXIS,stop) Останавливает предварительный просмотр отсюда до конца файла.
- (AXIS,notify,the_text) Отображает the_text как информационный дисплей

Это отображение может быть полезно при предварительном просмотре AXIS, когда комментарии (debug,message) не отображаются.

#### 10.1.12.11 Touch Off using Actual Position

The Touch Off feature can optionally incorporate the actual axis position value into the calculation for the offset. This is primarily used in cases where a non-motorized axis such as the quill in a milling machine provides feedback to LinuxCNC via an encoder, but there is no motor to control movement. This allows AXIS to provide a DRO display for such an axis with working touch off capability.

This feature is enabled on an axis by altering the appropriate *[AXIS_x]* section of the .INI file. Add an option named *TOUCHOFF_ACTUAL* and set the value to *PLUS* or *MINUS* depending on how you want to apply the actual position to the offset.

Example:

```
[AXIS_Z]
TOUCHOFF_ACTUAL = MINUS
```

Ordinarily, only the commanded position of an axis is used to set this offset, meaning it does not work properly because non-motorized axes are never commanded to move and thus their commanded position is always 0.

Touch off sends a *G10 L20* command to the MDI to set the new offset value. The value applied is normally just the value entered into the dialog box. When this feature is enabled, it will either add or subtract the current position value from the value entered in the dialog, depending on how it is configured.

### 10.1.13 Axisui

Чтобы улучшить взаимодействие AXIS с физическими маховичками, ось, выбранная в данный момент в графическом интерфейсе, также отображается на выводе с именем типа *axisui.jog.x*. Один из этих выводов одновременно имеет значение *TRUE*, а остальные — *FALSE*. Они предназначены для управления контактами разрешения перемещений медленной подачи.

**Axisui Контакты** AXIS имеет контакты HAL для указания того, какая кнопка с фиксацией выбрана на вкладке *Manual Control*.

```
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 имеет контакт HAL для обозначения приращения медленной подачи, выбранного на вкладке *Manual Tab*.

```
Type  Dir Name
float OUT axisui.jog.increment
```

AXIS имеет выходной контакт HAL, который указывает, когда произошло прерывание. Вывод *axisui.abort* будет иметь значение *TRUE* и вернется в состояние *FALSE* через 0,3 мс.

```
Type  Dir    Name
bit   OUT    axisui.abort
```

AXIS имеет выходной контакт HAL, который указывает, когда произошла ошибка. Вывод *axisui.error* останется *TRUE*, пока все уведомления об ошибках не будут отклонены.

```
Type  Dir    Name
bit   OUT    axisui.error
```

AXIS имеет входные контакты HAL для очистки всплывающих уведомлений об ошибках и информации.

```
Type  Dir    Name
bit   IN     axisui.notifications-clear
bit   IN     axisui.notifications-clear-error
bit   IN     axisui.notifications-clear-info
```

AXIS имеет входной контакт HAL, который отключает/включает функцию *Pause/Resume*.

```
Type  Dir    Name
bit   IN     axisui.resume-inhibit
```

### 10.1.14 Советы по настройке AXIS

AXIS — это довольно большая и сложная для проникновения база кода. Это помогает поддерживать стабильность кода, но затрудняет его настройку.
Здесь мы покажем фрагменты кода для изменения поведения или визуальных эффектов экрана.
Имейте в виду, что внутренний код AXIS может время от времени меняться.
работа этих фрагментов не гарантируется: возможно, их потребуется скорректировать.

#### 10.1.14.1 Функция обновления

В AXIS есть функция user_live_update, которая вызывается каждый раз, когда AXIS обновляется.
Вы можете использовать это для обновления своих собственных функций.

```
# continuous update function
def user_live_update():
    print('i am printed every update...')
```

#### 10.1.14.2 Отключить диалог закрытия

```
# disable the do you want to close dialog
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

#### 10.1.14.3 Изменить шрифт текста

```
# change the font

font = 'sans 11'
fname,fsize = font.split()
root_window.tk.call('font','configure','TkDefaultFont','-family',fname,'-size',fsize)

# redo the text in tabs so they resize for the new default font

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 font is independent

root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue')
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font)
#root_window.tk.call('.pane.bottom.t.text','configure','-foreground','blue','-font',font,'- ↩
    height','12')
```

#### 10.1.14.4 Измените быструю скорость с помощью сочетаний клавиш

```
# use control + ‘ or 1-0 as keyboard shortcuts for rapidrate and keep ‘ or 1-0 for feedrate
# also adds text to quick reference in help

help1.insert(10,("Control+ ‘,1..9,0", _("Set Rapid Override from 0% to 100%")),)

root_window.bind('<Control-Key-quoteleft>',lambda event: set_rapidrate(0))
root_window.bind('<Control-Key-1>',lambda event: set_rapidrate(10))
root_window.bind('<Control-Key-2>',lambda event: set_rapidrate(20))
root_window.bind('<Control-Key-3>',lambda event: set_rapidrate(30))
root_window.bind('<Control-Key-4>',lambda event: set_rapidrate(40))
root_window.bind('<Control-Key-5>',lambda event: set_rapidrate(50))
root_window.bind('<Control-Key-6>',lambda event: set_rapidrate(60))
root_window.bind('<Control-Key-7>',lambda event: set_rapidrate(70))
root_window.bind('<Control-Key-8>',lambda event: set_rapidrate(80))
root_window.bind('<Control-Key-9>',lambda event: set_rapidrate(90))
root_window.bind('<Control-Key-0>',lambda event: set_rapidrate(100))
root_window.bind('<Key-quoteleft>',lambda event: set_feedrate(0))
root_window.bind('<Key-1>',lambda event: set_feedrate(10))
root_window.bind('<Key-2>',lambda event: set_feedrate(20))
root_window.bind('<Key-3>',lambda event: set_feedrate(30))
root_window.bind('<Key-4>',lambda event: set_feedrate(40))
root_window.bind('<Key-5>',lambda event: set_feedrate(50))
root_window.bind('<Key-6>',lambda event: set_feedrate(60))
root_window.bind('<Key-7>',lambda event: set_feedrate(70))
root_window.bind('<Key-8>',lambda event: set_feedrate(80))
root_window.bind('<Key-9>',lambda event: set_feedrate(90))
root_window.bind('<Key-0>',lambda event: set_feedrate(100))
```

#### 10.1.14.5 Чтение INI-файла

```
# read an INI file item
machine = inifile.find('EMC','MACHINE')
print('machine name =',machine)
```

#### 10.1.14.6 Чтение статуса LinuxCNC

```
# LinuxCNC status can be read from s.
print(s.actual_position)
print(s.paused)
```

#### 10.1.14.7 Изменить текущий вид

```
# set the view of the preview
# valid views are view_x view_y view_y2 view_z view_z2 view_p
commands.set_view_z()
```

#### 10.1.14.8 Создание новых контактов AXISUI HAL

```
def user_hal_pins():
    comp.newpin('my-new-in-pin', hal.HAL_BIT, hal.HAL_IN)
    comp.ready()
```

#### 10.1.14.9 Создание нового HAL компонента и контактов

```
# create a component

mycomp = hal.component('my_component')
mycomp.newpin('idle-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.newpin('pause-led',hal.HAL_BIT,hal.HAL_IN)
mycomp.ready()

# connect pins

hal.new_sig('idle-led',hal.HAL_BIT)
hal.connect('halui.program.is-idle','idle-led')
hal.connect('my_component.idle-led','idle-led')

# set a pin

hal.set_p('my_component.pause-led','1')

# get a pin 2,8+ branch

value = hal.get_value('halui.program.is-idle')
print('value is a',type(value),'value of',value)
```

#### 10.1.14.10 Переключение вкладок с помощью контактов HAL

```
# HAL pins from a GladeVCP panel will not be ready when user_live_update is run
# to read them you need to put them in a try/except block

# the following example assumes 5 HAL buttons in a GladeVCP panel used to switch
# the tabs in the AXIS screen.
# button names are 'manual-tab', 'mdi-tab', 'preview-tab', 'dro-tab', 'user0-tab'
# the user_0 tab if it exists would be the first GladeVCP embedded tab
```

```
# for LinuxCNC 2.8+ branch

def user_live_update():
    try:
        if hal.get_value('gladevcp.manual-tab'):
            root_window.tk.call('.pane.top.tabs','raise','manual')
        elif hal.get_value('gladevcp.mdi-tab'):
            root_window.tk.call('.pane.top.tabs','raise','mdi')
        elif hal.get_value('gladevcp.preview-tab'):
            root_window.tk.call('.pane.top.right','raise','preview')
        elif hal.get_value('gladevcp.numbers-tab'):
            root_window.tk.call('.pane.top.right','raise','numbers')
        elif hal.get_value('gladevcp.user0-tab'):
            root_window.tk.call('.pane.top.right','raise','user_0')
    except:
        pass
```

#### 10.1.14.11 Добавить кнопку GOTO Home

```
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)

# make a button to home y axis
root_window.tk.call('button','.pane.top.tabs.fmanual.homey','-text','Home Y','-command',' ↩
    goto_home Y','-height','2')

# place the button
root_window.tk.call('grid','.pane.top.tabs.fmanual.homey','-column','1','-row','7','- ↩
    columnspan','2','-padx','4','-sticky','w')

# any function called from Tcl needs to be added to TclCommands
TclCommands.goto_home = goto_home
commands = TclCommands(root_window)
```

#### 10.1.14.12 Добавить кнопку в ручной фрейм

```
# make a new button and put it in the manual frame

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')

# the above send the "mybutton_clicked" command when clicked
# other options are to bind a press or release (or both) commands to the button
# these can be in addition to or instead of the clicked command
# if instead of then delete '-command','mybutton_clicked', from the first line

# Button-1 = left mouse button, 2 = right or 3 = middle

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')

# functions called from the buttons

def mybutton_clicked():
    print('mybutton was clicked')
def mybutton_pressed():
    print('mybutton was pressed')
def mybutton_released():
    print('mybutton was released')

# any function called from Tcl needs to be added to TclCommands

TclCommands.mybutton_clicked = mybutton_clicked
TclCommands.mybutton_pressed = mybutton_pressed
TclCommands.mybutton_released = mybutton_released
commands = TclCommands(root_window)
```

#### 10.1.14.13 Чтение внутренних переменных

```
# the following variables may be read from the vars instance

print(vars.machine.get())
print(vars.emcini.get())

    active_codes            = StringVar
    block_delete            = BooleanVar
    brake                   = BooleanVar
    coord_type              = IntVar
    display_type            = IntVar
    dro_large_font          = IntVar
    emcini                  = StringVar
    exec_state              = IntVar
    feedrate                = IntVar
    flood                   = BooleanVar
    grid_size               = DoubleVar
    has_editor              = IntVar
    has_ladder              = IntVar
    highlight_line          = IntVar
    interp_pause            = IntVar
    interp_state            = IntVar
    ja_rbutton              = StringVar
    jog_aspeed              = DoubleVar
    jog_speed               = DoubleVar
    kinematics_type         = IntVar
    linuxcnctop_command     = StringVar
    machine                 = StringVar
    max_aspeed              = DoubleVar
    max_maxvel              = DoubleVar
    max_queued_mdi_commands = IntVar
    max_speed               = DoubleVar
    maxvel_speed            = DoubleVar
    mdi_command             = StringVar
    metric                  = IntVar
    mist                    = BooleanVar
    motion_mode             = IntVar
    on_any_limit            = BooleanVar
    optional_stop           = BooleanVar
    override_limits         = BooleanVar
```

```
    program_alpha           = IntVar
    queued_mdi_commands     = IntVar
    rapidrate               = IntVar
    rotate_mode             = BooleanVar
    running_line            = IntVar
    show_distance_to_go     = IntVar
    show_extents            = IntVar
    show_live_plot          = IntVar
    show_machine_limits     = IntVar
    show_machine_speed      = IntVar
    show_program            = IntVar
    show_pyvcppanel         = IntVar
    show_rapids             = IntVar
    show_tool               = IntVar
    show_offsets            = IntVar
    spindledir              = IntVar
    spindlerate             = IntVar
    task_mode               = IntVar
    task_paused             = IntVar
    task_state              = IntVar
    taskfile                = StringVar
    teleop_mode             = IntVar
    tool                    = StringVar
    touch_off_system        = StringVar
    trajcoordinates         = StringVar
    tto_g11                 = BooleanVar
    view_type               = IntVar
```

#### 10.1.14.14 Скрыть виджеты

```
# hide a widget
# use 'grid' or 'pack' depending on how it was originally placed
root_window.tk.call('grid','forget','.pane.top.tabs.fmanual.jogf.zerohome.tooltouch')
```

#### 10.1.14.15 Изменить ярлык

```
# change label of a widget
root_window.tk.call('setup_widget_accel','.pane.top.tabs.fmanual.mist','Downdraft')

# make sure it appears (only needed in this case if the mist button was hidden)
root_window.tk.call('grid','.pane.top.tabs.fmanual.mist','-column','1','-row','5','- ↩
    columnspan','2','-padx','4','-sticky','w')
```

#### 10.1.14.16 Перенаправить существующую команду

```
# hijack an existing command
# originally the mist button calls the mist function
root_window.tk.call('.pane.top.tabs.fmanual.mist','configure','-command','hijacked_command' ↩
    )

# The new function
def hijacked_command():
    print('hijacked mist command')

# add the function to TclCommands
```

```
TclCommands.hijacked_command = hijacked_command
commands = TclCommands(root_window)
```

#### 10.1.14.17 Изменить цвет УЦИ

```
# change dro screen
root_window.tk.call('.pane.top.right.fnumbers.text','configure','-foreground','green','- ↩
    background','black')
```

#### 10.1.14.18 Изменение кнопок панели инструментов

```
# change the toolbar buttons

buW = '3'
buH = '2'
boW = '3'

root_window.tk.call('.toolbar.machine_estop','configure','-image','','-text','ESTOP','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.machine_power','configure','-image','','-text','POWER','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.file_open','configure','-image','','-text','OPEN','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.reload','configure','-image','','-text','RELOAD','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_run','configure','-image','','-text','RUN','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_step','configure','-image','','-text','STEP','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_pause','configure','-image','','-text','PAUSE','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_stop','configure','-image','','-text','STOP','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_blockdelete','configure','-image','','-text','Skip /' ↩
    ,'-width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.program_optpause','configure','-image','','-text','M1','- ↩
    width',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomin','configure','-image','','-text','Zoom+','-width' ↩
    ,buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_zoomout','configure','-image','','-text','Zoom-','-width ↩
    ',buW,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_z','configure','-image','','-text','Top X','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_z2','configure','-image','','-text','Top Y','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_x','configure','-image','','-text','Right','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_y','configure','-image','','-text','Front','-width',buW, ↩
    '-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.view_p','configure','-image','','-text','3D','-width',buW,'- ↩
    height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.rotate','configure','-image','','-text','Rotate','-width',buW ↩
    ,'-height',buH,'-borderwidth',boW)
root_window.tk.call('.toolbar.clear_plot','configure','-image','','-text','Clear','-width', ↩
    buW,'-height',buH,'-borderwidth',boW)
```

#### 10.1.14.19 Изменение цветов отрисовки

В формате RGBA, в следующем порядке: медленная подача, быстрое перемещение, подача, дуга, смена инструмента, измерительный щуп

```
# change plotter colors
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 Введение

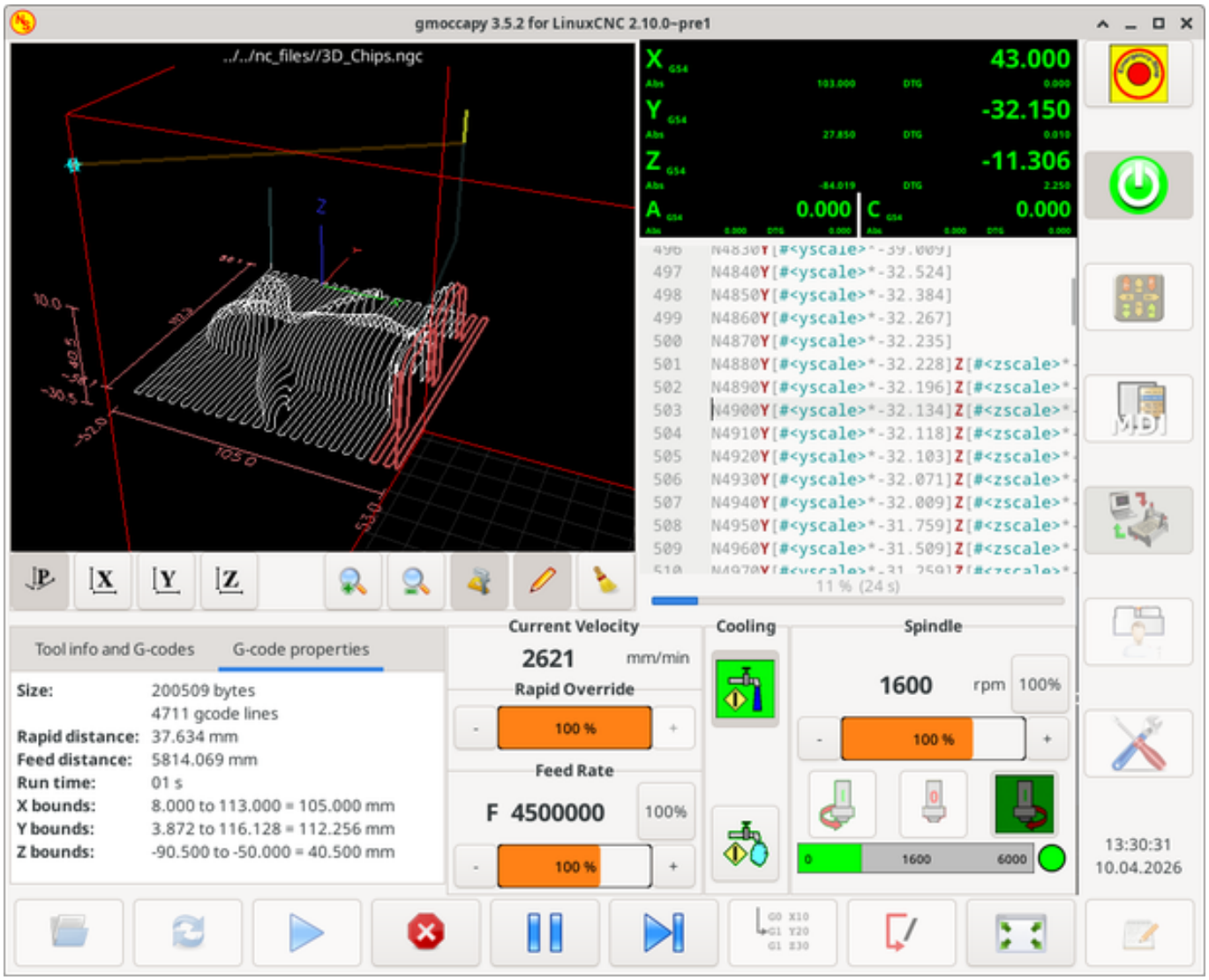
GMOCCAPY — это графический интерфейс для LinuxCNC, предназначенный для использования с сенсорным экраном, но его также можно использовать на обычных экранах с помощью мыши или аппаратных кнопок и маховичков РГИ, поскольку он предоставляет контакты HAL для наиболее распространенных нужд. Дополнительную информацию можно найти ниже.

Он предлагает возможность отображать до 9 осей, поддерживает режим токарного станка для обычного и обратного токарных станков и может быть адаптирован практически к любым потребностя поскольку GMOCCAPY поддерживает встроенные вкладки и боковые панели. В качестве хорошего примера см. gmoccapy_plasma.

GMOCCAPY 3 поддерживает до 9 осей и 9 сочленений. Поскольку код GMOCCAPY 3 был изменен для поддержки изменений сочленений/осей в LinuxCNC, он не работает в ветках 2.7 или 2.6!

Он поддерживает встроенную виртуальную клавиатуру (встроенную или экранную клавиатуру), поэтому нет необходимости в аппаратной клавиатуре или мыши, но ее также можно использовать с этим оборудованием. GMOCCAPY предлагает отдельную страницу настроек для настройки большинства параметров графического интерфейса без редактирования файлов.

GMOCCAPY можно очень легко локализовать, поскольку соответствующие файлы отделены от файлов linuxcnc.po, поэтому нет необходимости переводить ненужный материал. Если вы хотите внести свой вклад в перевод, воспользуйтесь ссылкой: веб-редактор переводов Weblate. Дополнительн информацию см. в разделе Переводы




## 10.2.2 Requirements

GMOCCAPY 3 был протестирован на Debian Jessie, Debian Stretch и MINT 18 с LinuxCNC master и версией 2.8. Он полностью поддерживает изменения сочленений/осей LinuxCNC, что делает его подходящим в качестве ГИП для Scara, роботов или любой другой конфигурации с большим количеством сочленений, чем осей. Таким образом, он также поддерживает портальные конфигураци Если вы используете другие версии, сообщите о проблемах и/или решениях на странице . Форум LinuxCNC или Немецкий форум CNC Ecke или LinuxCNC список рассылки пользователей.

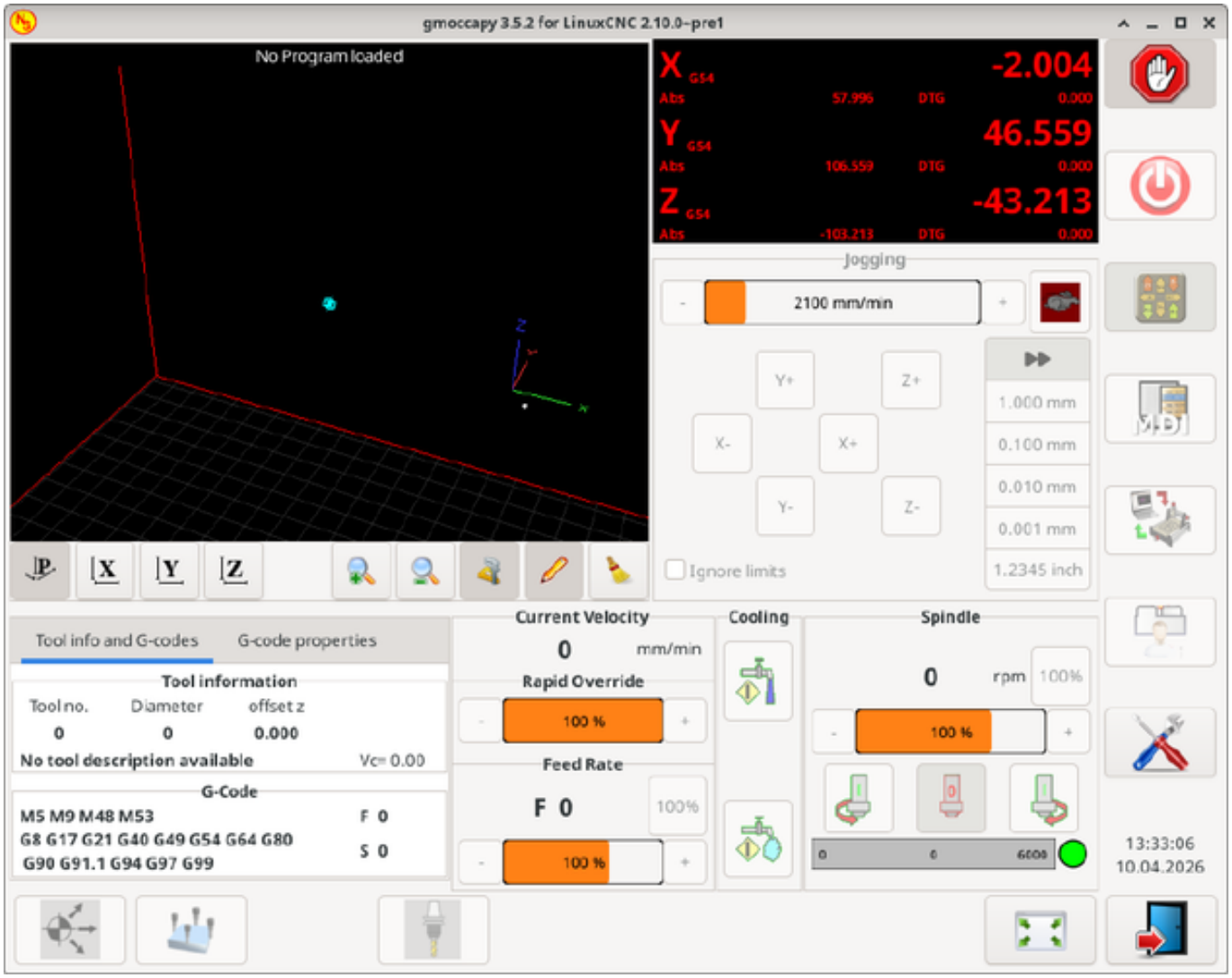
Минимальное разрешение экрана для GMOCCAPY для обычной планировки (без боковых панелей) составляет **980 x 750 пикселей**, поэтому оно должно соответствовать любому стандартному экрану. Рекомендуется использовать экраны с минимальным разрешением 1024x768. Существует также конфигурация, подходящая для экранов 800x600 (представленная в GMOCCAPY 3.4.8).

## 10.2.3 Как получить GMOCCAPY

GMOCCAPY 3 включен в стандартную поставку LinuxCNC начиная с версии 2.7. Поэтому самый простой способ получить GMOCCAPY на управляющем ПК — это просто загрузить ISO и установить его с CD/DVD/USB-накопителя. Это позволяет вам получать обновления вместе с обычными пакетами Debian.

По ссылке **примечания к выпуску**, также известной как список изменений, вы можете отслеживать последние исправления ошибок и функции.

Вы увидите экран, похожий на следующий (дизайн может отличаться в зависимости от вашей конфигурации):




## 10.2.4 Базовая конфигурация

GMOCCAPY 3 поддерживает следующие параметры командной строки:

- *-user_mode*: если установлено, кнопка настройки будет отключена, поэтому обычные операторы станка не смогут редактировать настройки станка.
- *-logo <путь к файлу логотипа>*: если задано, логотип скроет вкладку кнопки медленной подачи в ручном режиме. Это полезно только для машин с аппаратными кнопками для медленой подачи и выбора дистанции.

На самом деле не так уж много нужно настроить только для запуска GMOCCAPY, но есть некоторые моменты, на которые следует обратить внимание, если вы хотите использовать все функции ГИП.

Вы найдете множество конфигураций моделирования (INI-файлов), просто чтобы показать основы:

- 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
- и многое другое …

Имена должны объяснять основное предназначение различных INI-файлов.

Если вы используете существующую конфигурацию вашего станка, просто отредактируйте свой INI в соответствии с этим документом.

Итак, давайте подробнее рассмотрим INI-файл и то, что вам нужно включить для использования GMOCCAPY на вашем компьютере:

#### 10.2.4.1 Раздел DISPLAY

```
[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* - Это говорит LinuxCNC использовать GMOCCAPY.
- *PREFERENCE_FILE_PATH* - Указывает расположение и имя файла настроек, который будет использоваться. В большинстве случаев эта строка не нужна, она используется GMOCCAPY для хранения настроек графического пользовательского интерфейса, таких как темы, единицы измерения УЦИ, цвета, настройки клавиатуры и т. д., см. <<gmoccapy:settings-page,settingspage> > подробнее.

**Note**
Если путь или файл не указаны, GMOCCAPY будет использовать по умолчанию <your_machinename>.pref, если в вашем INI-файле не указано имя компьютера, будет использоваться gmoccapy.pref. Файл будет храниться в вашей директории с конфигами, поэтому настройки не будут смешиваться, если вы используете несколько конфигов. Если вы хотите использовать только один файл для нескольких машин, вам необходимо включить PREFERENCE_FILE_PATH в ваш INI.

- *MAX_FEED_OVERRIDE = 1.5* - Устанавливает максимальное переопределение подачи. В приведенно примере вам будет разрешено переопределять подачу на 150%.

**Note**
Если значение не указано, оно будет установлено на 1,0.

- *MIN_SPINDLE_OVERRIDE = 0.5* and *MAX_SPINDLE_OVERRIDE = 1.2* - Позволит изменить переопреде шпинделя в пределах от 50% до 120%.

**Note**
Если значения не указаны, MIN будет установлен на 0,1, а MAX на 1,0.

- *LATHE = 1* - Установите макет экрана для управления токарным станком.
- *BACK_TOOL_LATHE = 1* - Является необязательным и позволяет переключать ось X так, как вам нужно для токарного станка с задним инструментом. Кроме того, сочетания клавиш будут реагировать по-другому. С помощью GMOCCAPY разрешено также настраивать токарный станок с дополнительными осями, поэтому вы также можете использовать конфигурацию XZCW для токарного станка.

**Tip**
См. также раздел Раздел.

- *PROGRAM_PREFIX = ../../nc_files/* - Это запись сообщает LinuxCNC/GMOCCAPY где искать файлы NGC.

**Note**
Если не указано иное, GMOCCAPY будет искать файлы NGC в следующем порядке: сначала linuxcnc/nc_files, а затем домашний каталог пользователя.

- *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* - Устанавливает минимальную скорость медленного перемещения станка для поворотных осей.
- *MAX_ANGULAR_VELOCITY* - Устанавливает максимальную скорость медленного перемещения станка для поворотных осей.
- *DEFAULT_ANGULAR_VELOCITY* - Устанавливает скорость медленного перемещения станка для осей вращения по умолчанию.

#### 10.2.4.2 Раздел TRAJ

- *DEFAULT_LINEAR_VELOCITY = 85.0* - Устанавливает скорость перемещения станка по умолчанию.

> **Note**
> Если не установлено, будет использоваться половина значения *MAX_LINEAR_VELOCITY*. Если это значение также не указано, по умолчанию оно будет равно 180.

- *MAX_LINEAR_VELOCITY = 230.0* - Устанавливает максимальную скорость станка. Это значение также будет максимальной линейной скоростью медленного перемещения.

> **Note**
> По умолчанию 600, если не установлено.

#### 10.2.4.3 Кнопки макроса

Вы можете добавлять макросы в GMOCCAPY, аналогично методу Touchy. Макрос — это не что иное, как файл NGC. Вы можете выполнять полные программы ЧПУ в режиме MDI, просто нажав одну кнопку. Для этого сначала необходимо указать путь поиска макросов:

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

Это задает путь для поиска макросов и других подпрограмм. Несколько путей подпрограмм могут быть разделены знаком ":".

Затем вам просто нужно добавить такой раздел:

**Настройка пяти макросов для отображения в списке кнопок MDI**

```
[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
```

Затем вам необходимо предоставить соответствующие файлы NGC, которые должны соответствовать следующим правилам:

- Имя файла должно быть точно таким же, как имя, указанное в строке макроса, только с расширением ".ngc" (с учетом регистра).
- Файл должен содержать подпрограмму типа ***O<i_am_lost> sub***, имя подпрограммы должно точно соответствовать (с учетом регистра) имени макроса.
- Файл должен заканчиваться endsub ***O<i_am_lost> endsub***, за которым следует команда ***M2***.
- Файлы необходимо поместить в папку, указанную в вашем INI-файле с помощью *SUBROUTINE_PATH* в разделе RS274NGC

Код между sub и endsub будет выполнен нажатием соответствующей кнопки макроса.

**Note**
В графическом интерфейсе будет показано максимум 16 макросов. Из-за нехватки места вам может потребоваться нажать на стрелку, чтобы переключить страницу и отобразить скрытые кнопки макросов. Кнопки макросов будут отображаться в порядке записей INI. Размещение более 16 макросов в INI-файле не является ошибкой, они просто не будут отображаться.

**Note**
Вы найдете примеры макросов в папке с именем *macros*, расположенной в папке GMOCCAPY sim. Если вы указали несколько путей к подпрограммам, они будут искаться в порядке заданных путей. Будет использован первый найденный файл.

GMOCCAPY также принимает макросы, запрашивающие такие параметры, как:

```
[MACROS]
MACRO = go_to_position X-pos Y-pos Z-pos
```

Параметры должны быть разделены пробелами. В этом примере вызывается файл *go_to_position.ngc* со следующим содержимым:

```
; Test file ”go to position”
; will jog the machine to a given position

O<go_to_position> sub

G17
G21
G54
G61
G40
G49
G80
G90

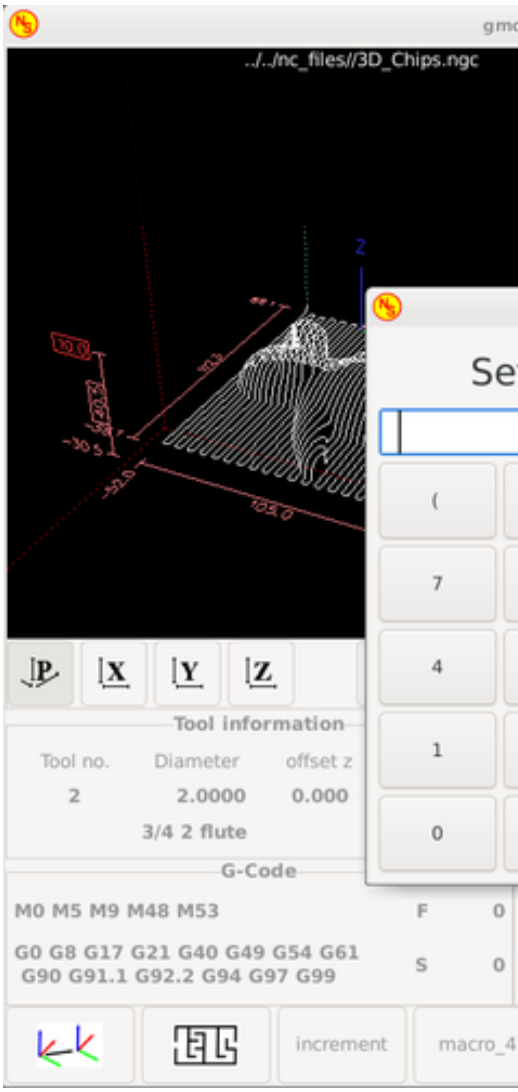
;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>

(DEBUG, Will now move machine to X = #1 , Y = #2 , Z = #3)
G0 X #1 Y #2 Z #3

O<go_to_position> endsub
M2
```

После нажатия кнопки ***выполнить макрос*** вам будет предложено ввести значения для ***X-pos Y-pos Z-pos***, и макрос запустится только в том случае, если все значения были заданы.

**Note**
Если вы хотите использовать макрос без каких-либо перемещений, см. также примечания в разделе известные проблемы.




**Пример макроса с использованием макроса ”go to position”**

### 10.2.4.4 Встроенные вкладки и панели

Вы можете добавлять встроенные программы в GMOCCAPY, как в AXIS, Touchy и Gscreen. Все делается GMOCCAPY автоматически, если вы добавите несколько строк в свой INI-файл в раздел DISPLAY.

Если вы никогда не использовали панель Glade, рекомендую прочитать отличную документацию по адресу Glade VCP.

**Пример встроенной вкладки**

```
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
```

Все, о чем вам нужно позаботиться, это включить для каждой вкладки или боковой панели упомянутые три строки:

- EMBED_TAB_NAME = Представляет имя вкладки или боковой панели. Какое имя вы используете, зависит от вас, но оно должно присутствовать!

- EMBED_TAB_LOCATION = Место, где будет размещена ваша программа в ГИП, см. на рисунке Расположение встроенных вкладок. Допустимые значения:
  - **ntb_user_tabs** (в качестве основной вкладки, закрывающей весь экран*(7)*)
  - **ntb_preview** (как вкладка на стороне предварительного просмотра **(1)**)
  - **hbox_jog** (скроет кнопки медленного перемещения и разместит здесь ваш файл Glade **(2)**)
  - **box_left** (слева, полная высота экрана **(10)**)
  - **box_right** (справа, между обычным экраном и списком кнопок **(11)**)
  - **box_tool_and_code_info** (скроет Tool information и поля G-code и разместит здесь ваш файл Glade. **(3)**)
  - **box_tool_info** (скроет поле информации об инструменте и разместит здесь ваш файл Glade **(3a)**)
  - **box_code_info** (скроет информационное поле G-code и разместит здесь ваш файл Glade **(3b)**)
  - **box_vel_info** (скроет поля скорости и разместит ваш Glade файл **(4)**)
  - **box_coolant_and_spindle** (скроет поля шпинделя и СОЖ и разместит здесь ваш файл Glade **(5)+(6)**)
  - **box_cooling** (скроет поле охлаждения и разместит ваш Glade файл **(5)**)
  - **box_spindle** (скроет поле шпинделя и разместит ваш glade файл **(6)**)
  - **box_custom_1** (расположит ваш Glade файл слева от vel_frame)
  - **box_custom_2** (разместит ваш файл Glade слева от cooling_frame)
  - **box_custom_3** (представит ваш файл Glade слева от spindle_frame)
  - **box_custom_4** (представит ваш Glade файл справа от spindle_frame)
  - **box_dro_side** (разместит ваш Glade файл справа от УЦИ **(8)**)
  - **ntb_preview** (как вкладка на стороне предварительного просмотра **(9)**)

---

**Note**
См. также прилагаемые примеры INI-файлов, чтобы увидеть различия.

---

- EMBED_TAB_COMMAND = Команда для выполнения, т.е.

```
gladevcp -x {XID} dro.glade
```

включает в себя собственный файл Glade под названием dro.glade в указанном месте. Файл необходимо поместить в папку конфигурации вашего компьютера.

```
gladevcp h_buttonlist.glade
```

просто откроет новое пользовательское окно под названием h_buttonlist.glade. Обратите внимание на разницу. Это автономный файл, и его можно перемещать независимо от окна GMOCCAPY.

```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

добавит панель manual-example.ui, включит собственный обработчик Python, hitcounter.py и установит все подключения после реализации панели в соответствии с manual-example.hal.

```
hide
```

скроет выбранное поле.

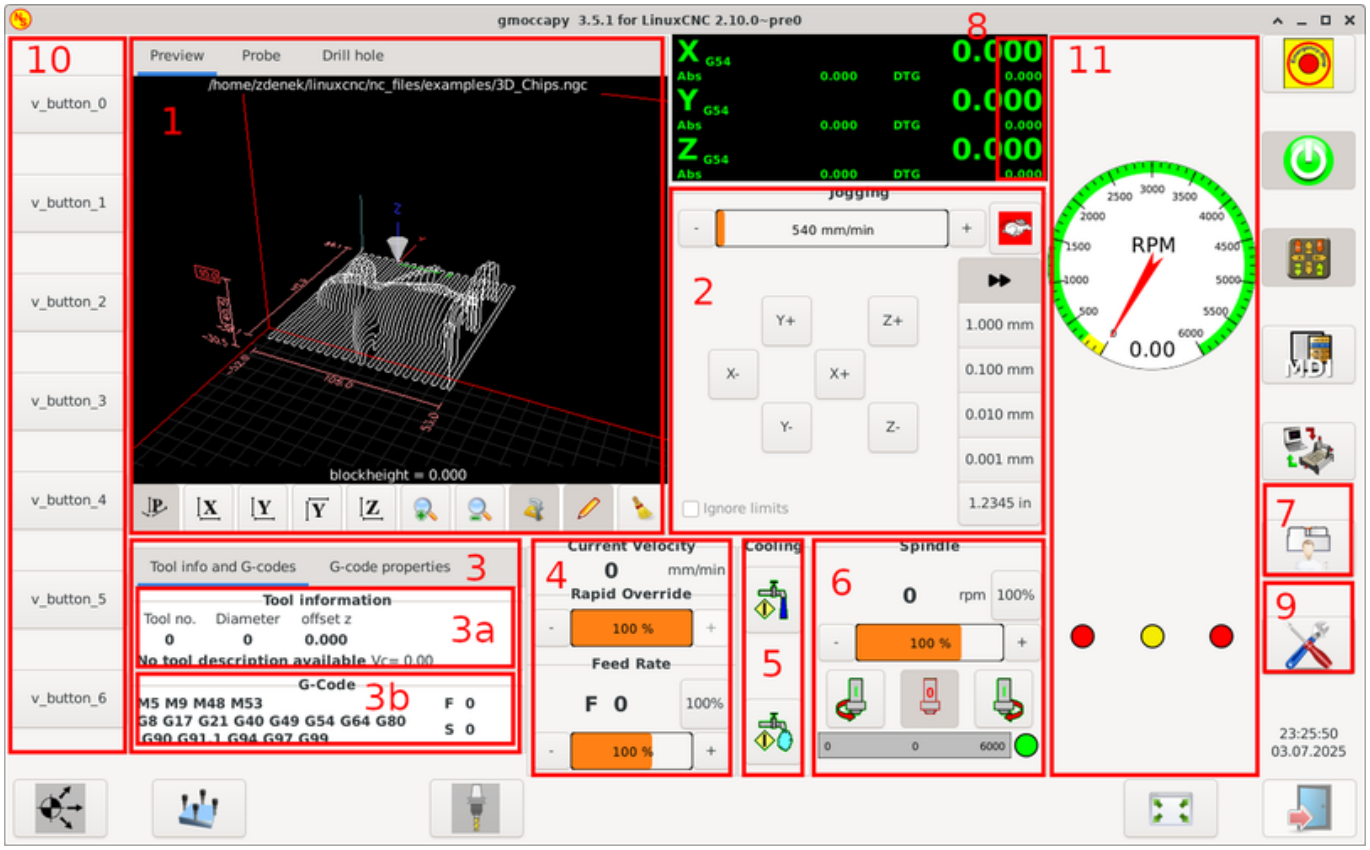
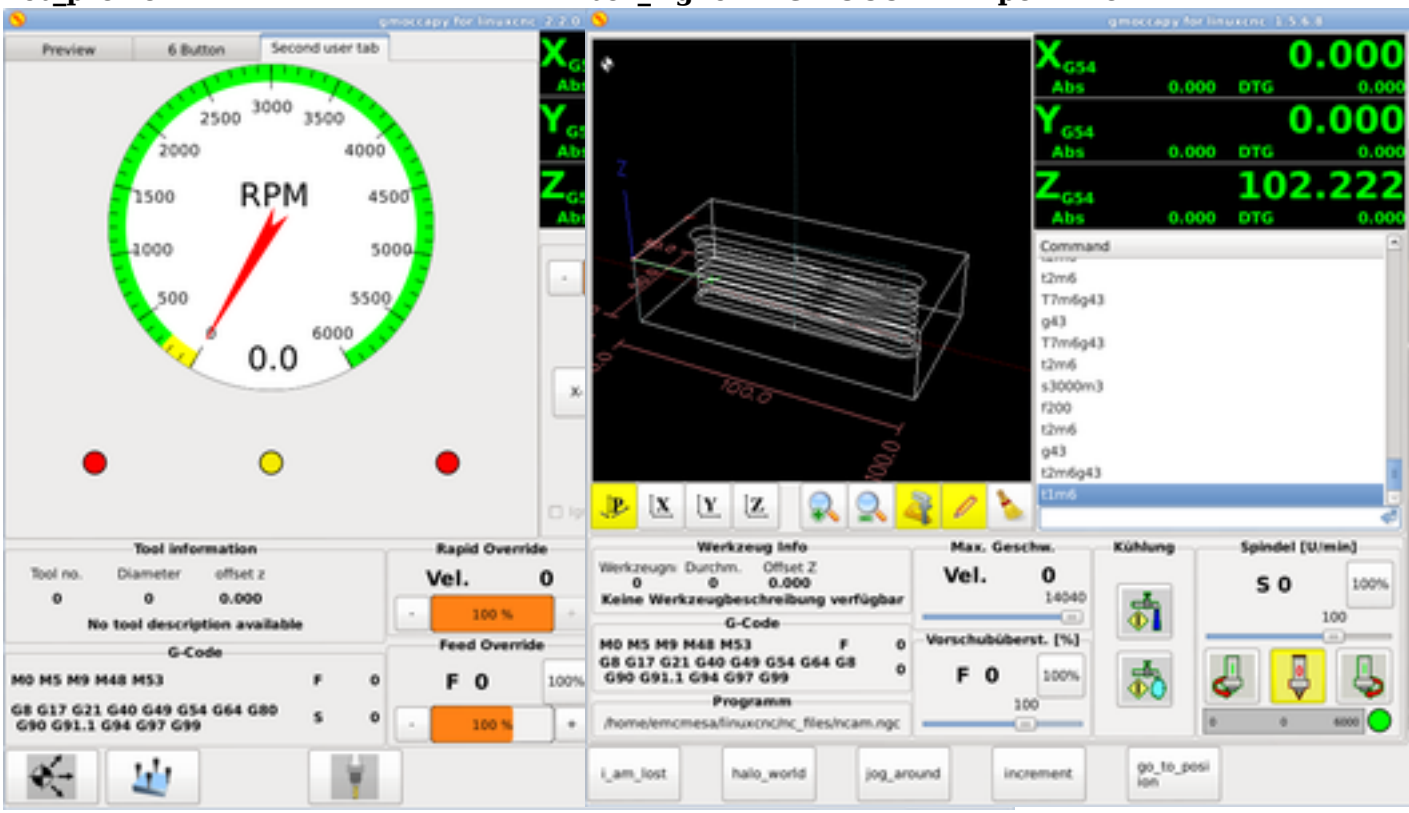



Figure 10.14: Расположение встроенных вкладок

**Note**
Если вы выполняете какие-либо подключения HAL к своей пользовательской панели Glade, вам необходимо сделать это в файле HAL, указанном в строке EMBED_TAB_COMMAND, в противном случае вы можете получить сообщение об ошибке, что контакт HAL не существует — это происходит из-за условий гонки, при загрузке HAL-файлов. Соединения с контактами GMOCCAPY HAL необходимо выполнить в файле HAL postgui, указанном в вашем INI-файле, поскольку эти контакты не существуют до реализации ГИП.

Вот некоторые примеры:

**ntb_preview** **box_right - и GMOCCAPY в режиме MDI**




### 10.2.4.5 Сообщения, созданные пользователем

GMOCCAPY имеет возможность создавать пользовательские сообщения на основе HAL. Чтобы их использовать, вам необходимо добавить несколько строк в раздел [DISPLAY] INI-файла.

Эти три строки необходимы для определения диалогового окна всплывающего сообщения пользовател

```
MESSAGE_TEXT    = The text to be displayed, may be pango markup formatted
MESSAGE_TYPE    = ”status” , ”okdialog” , ”yesnodialog”
MESSAGE_PINNAME = is the name of the HAL pin group to be created
```

Сообщения поддерживают язык разметки pango. Подробную информацию о языке разметки можно найти по адресу Pango Markup.

Доступны следующие три типа диалогов:

- **status** - Просто отобразит сообщение в виде всплывающего окна, используя систему обмена сообщениями GMOCCAPY.
- **okdialog** - Будет удерживать фокус на диалоговом окне сообщения и активирует HAL контакт `-waiting`.
- **yesnodialog** - Удержит фокус на диалоговом окне сообщения, активирует контакт HAL `-waiting` и предоставляет контакт HAL `-response`.

Для получения более подробной информации о контактах см. User Created Message HAL Pins.

**Пример конфигурации сообщений пользователя**

```
MESSAGE_TEXT = This is a <span background=”#ff0000” foreground=”#ffffff”>info-message</span ↩
    > test
```

```
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog

MESSAGE_TEXT = Text can be <small>small</small>, <big>big</big>, <b>bold</b <i>italic</i>,  ↩
    and even be <span color="red">colored</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog
```

**Note**
На данный момент форматирование не работает.

#### 10.2.4.6 Управление предварительным просмотром

Волшебные комментарии можно использовать для управления предварительным просмотром G-кода. В очень больших программах загрузка предварительного просмотра может занять много времени. Вы можете управлять тем, что отображается и что скрыто на графическом экране, добавляя в свой G-код соответствующие комментарии из этого списка:

```
(PREVIEW,hide)
<G-code to be hidden>
(PREVIEW,show)
```

#### 10.2.4.7 Пользовательский командный файл

Если файл `~/.gmoccapyrc` существует, его содержимое выполняется как исходный код Python сразу после отображения ГИП. Детали того, что может быть записано в `~/.gmoccapyrc`, могут быть изменены в ходе цикла разработки.

Файл Python для конкретной конфигурации можно указать с помощью параметра INI-файла

```
[DISPLAY]
USER_COMMAND_FILE=filename.py
```

Если этот файл указан, он будет считан сразу после отображения ГИП GMOCCAPY **вместо** `~/.gmoccapyrc`.

В следующем примере изменяется размер вертикальных кнопок: .Пример файла .gmoccapyrc

```
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)
```

Имена виджетов можно посмотреть в файле /usr/share/gmoccapy.glade

#### 10.2.4.8 Пользовательский CSS файл

Подобно командному файлу пользователя, на внешний вид можно влиять с помощью каскадных таблиц стилей (CSS). Если файл `~/.gmoccapy_css` существует, его содержимое загружается в поставщика таблицы стилей и таким образом применяется к графическому интерфейсу.

CSS-файл для конкретной конфигурации можно указать с помощью параметра INI-файла

```
[DISPLAY]
USER_CSS_FILE=filename.css
```

Если этот файл указан, он используется **вместо** `~/.gmoccapy_css`.

Информацию о том, чем можно управлять с помощью CSS, можно найти здесь: link:https://docs.gtk.org/overview.html [Обзор CSS в GTK]

Вот пример того, как можно установить желтый цвет отмеченных кнопок: Пример желтого цвета для отмеченных кнопок

```
button:checked {
    background: rgba(250,230,0,0.8);
}
```

#### 10.2.4.9 Ведение журнала

GMOCCAPY поддерживает указание уровня информации (log level), которая будет выводиться на консоль и в файл журнала.

Порядок: *VERBOSE*, *DEBUG*, *INFO*, *WARNING*, *ERROR*, *CRITICAL*. По умолчанию установлено значение *WARNING*, что означает, что печатаются *WARNING*, *ERROR* и *CRITICAL*.

Вы можете указать уровень детализации журнала в INI-файле следующим образом:

```
[DISPLAY]
DISPLAY = gmoccapy <log_level_param>
```

используя эти параметры:

```
Log level   <log_level_param>
DEBUG       -d
INFO        -i
VERBOSE     -v
ERROR       -q
```

**Пример: Настройка ведения журнала для записи только ошибок**

```
[DISPLAY]
DISPLAY = gmoccapy -q
```

Вы можете указать, где сохранить файл журнала:

```
[DISPLAY]
LOG_FILE = gmoccapy.log
```

Если `LOG_FILE` не установлен, журналирование происходит в `$HOME/<base_log_name>.log`.

### 10.2.5 HAL Контакты

GMOCCAPY экспортирует несколько контактов HAL, чтобы иметь возможность реагировать на аппаратные устройства. Цель состоит в том, чтобы получить ГИП, которым можно было бы управлять в магазине инструментов полностью/в основном без мыши или клавиатуры.

---

**Note**
Все подключения к контактам GMOCCAPY вам придется выполнить в файле postgui.hal. При запуске GMOCCAPY он создает контакты HAL для ГИП, а затем выполняет post-GUI файл HAL, указанный в INI файле:

```
[HAL]
POSTGUI_HALFILE=<filename>
```

Обычно `<filename>` представляет собой базовое имя конфигурации + `_postgui.hal`, например. `lathe_postgui.hal`, но может быть любым допустимым именем файла.
Эти команды выполняются после построения экрана, гарантируя доступность контактов HAL виджета.
В INI-файле может быть несколько строк `POSTGUI_HALFILE=<filename>`. Каждый из них будет запускаться один за другим в том порядке, в котором они появляются.

---

#### 10.2.5.1 Списки правых и нижних кнопок

На экране есть два основных списка кнопок: один справа, другой внизу. Правые кнопки не будут меняться во время работы, но список нижних кнопок будет меняться очень часто. Кнопки отсчитываются сверху вниз и слева направо, начиная с 0.

---

**Note**
Названия контактов были изменены в GMOCCAPY 2, чтобы лучше их упорядочить.

---

Контакты для правых (вертикальных) кнопок:

- **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)*

Для нижних (горизонтальных) кнопок это:

- **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)*

Поскольку кнопки в нижнем списке будут меняться в зависимости от режима и других воздействий, аппаратные кнопки будут активировать отображаемые функции. Таким образом, вам не нужно заботиться о переключении функций в HAL, потому что это полностью делает GMOCCAPY!

Для трехосного станка XYZ контакты HAL будут реагировать, как показано в следующих трех таблицах:

Table 10.3: Функциональное назначение горизонтальных кнопок (1)

| **Контакт** | **Ручной режим** | **Режим MDI** | **Автоматический режим** |
|---|---|---|---|
| gmoccapy.h-button.button-0 | открыть кнопку исходной позиции | macro 1 (if defined) | открыть файл |
| gmoccapy.h-button.button-1 | открыть меню касания | macro 2 (if defined) | перезагрузить программу |
| gmoccapy.h-button.button-2 | | macro 3 (if defined) | запуск |
| gmoccapy.h-button.button-3 | открыть инструментальные диалоги | macro 4 (if defined) | стоп |
| gmoccapy.h-button.button-4 | | macro 5 (if defined) | пауза |
| gmoccapy.h-button.button-5 | | macro 6 (if defined) | шаг за шагом |
| gmoccapy.h-button.button-6 | | macro 7 (if defined) | запускать из строки, если это разрешено в настройках, иначе ничего |
| gmoccapy.h-button.button-7 | | macro 8 (if defined) | опциональные блоки |
| gmoccapy.h-button.button-8 | полноэкранный предпросмотр | macro 9 или кнопка для отображения дополнительных макросов | полноэкранный предпросмотр |
| gmoccapy.h-button.button-9 | выйти, если станок выключен, иначе никакой реакции | открыть клавиатуру или прервать работу, если макрос запущен | редактировать код |

Table 10.4: Функциональное назначение горизонтальных кнопок (2)

| **Контакт** | **Режим настроек** | **Режим исходной позиции** | **Режим касания** |
|---|---|---|---|
| gmoccapy.h-button.button-0 | стереть историю MDI | | касание X |
| gmoccapy.h-button.button-1 | | вернуть все в исходную позицию | касание Y |

Table 10.4: (continued)

| Контакт | Режим настроек | Режим исходной позиции | Режим касания |
|---|---|---|---|
| gmoccapy.h-button.button-2 | | | касание Z |
| gmoccapy.h-button.button-3 | | исходная позиция x | |
| gmoccapy.h-button.button-4 | toggle язык релейных схем | исходная позиция y | |
| gmoccapy.h-button.button-5 | toggle HAL scope | исходная позиция z | |
| gmoccapy.h-button.button-6 | toggle HAL status | | |
| gmoccapy.h-button.button-7 | toggle HAL meter | | |
| gmoccapy.h-button.button-8 | toggle HAL calibration | вывести все из исходной позиции | |
| gmoccapy.h-button.button-9 | toggle HAL show | назад | назад |

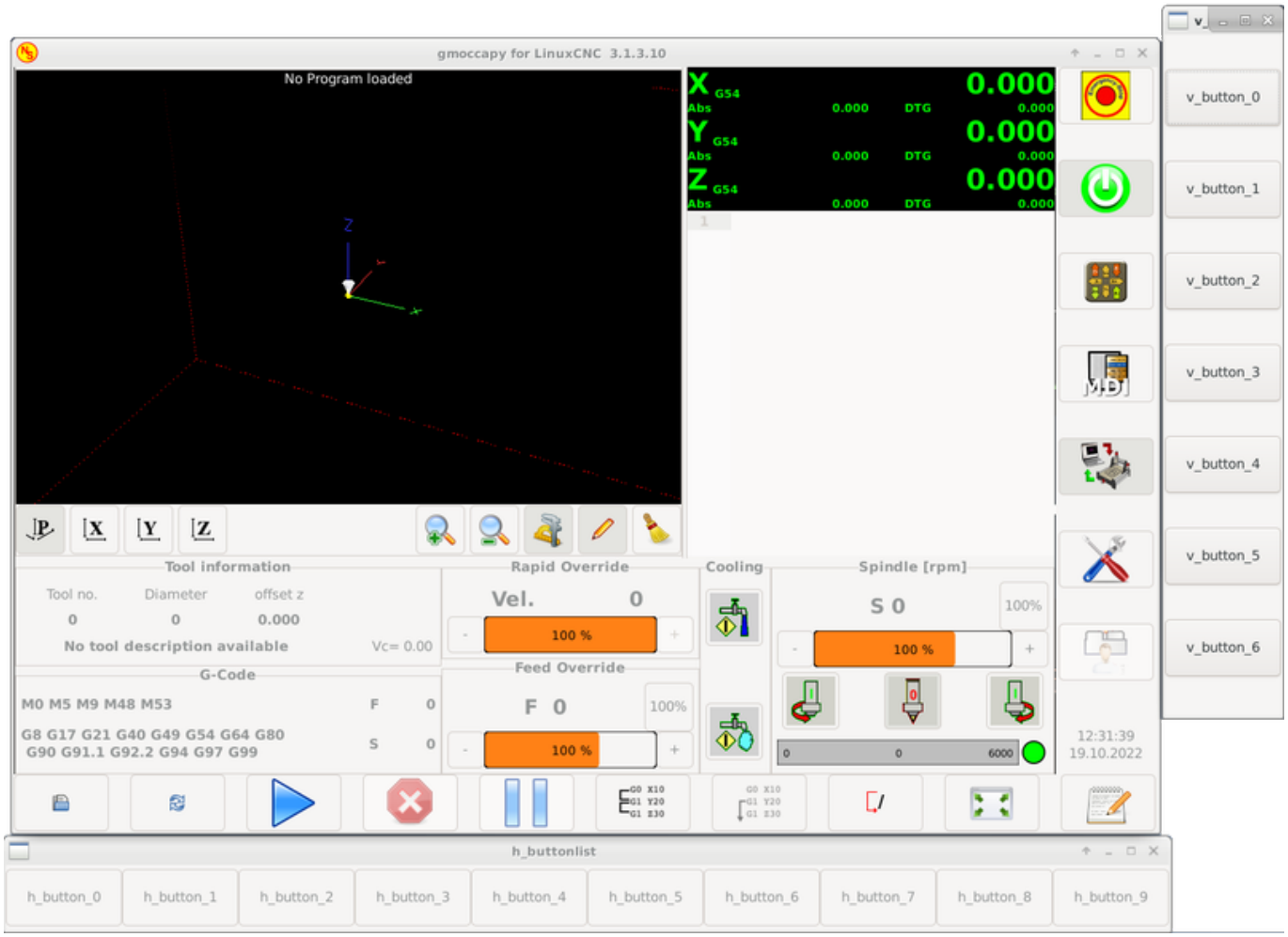
Table 10.5: Функциональное назначение горизонтальных кнопок (3)

| Контакт | Режим инструмента | Режим редактирования | Выбрать файл |
|---|---|---|---|
| gmoccapy.h-button.button-0 | | | перейти в домашний каталог |
| gmoccapy.h-button.button-1 | | перезагрузить файл | на один каталог выше |
| gmoccapy.h-button.button-2 | | сохранить | |
| gmoccapy.h-button.button-3 | | сохранить как | двигать выделение влево |
| gmoccapy.h-button.button-4 | сменить инструмент под номером T? M6 | | двигать выделение вправо |
| gmoccapy.h-button.button-5 | установить инструмент по номеру без изменения M61 Q? | | перейти в каталог, указанный в настройках |
| gmoccapy.h-button.button-6 | сменить инструмент на выбранный | Новый файл | |
| gmoccapy.h-button.button-7 | | | выбрать / ENTER |
| gmoccapy.h-button.button-8 | касание инструмента в Z | показать клавиатуру | |
| gmoccapy.h-button.button-9 | назад | назад | назад |

Итак, у нас есть 67 реакций всего с 10 контактами HAL!

Эти контакты предназначены для использования экрана без сенсорной панели или для защиты его от чрезмерного использования путем размещения аппаратных кнопок вокруг панели. Они доступны в примерной конфигурации, как показано на изображение ниже.

**Пример конфигурации ”gmoccapy_sim_hardware_button” с боковыми кнопками**




### 10.2.5.2 Скорости и переназначения

Все слайдеры от GMOCCAPY можно подключить к аппаратным энкодерам или аппаратным потенциоме

**Note**
В GMOCCAPY 3 имена некоторых контактов HAL изменились после реализации новых элементов управления. Максимальной скорости больше не существует, она была заменена быстрым переопределением по требованию многих пользователей.

Для подключения энкодеров экспортируются следующие контакты:

- **gmoccapy.jog.jog-velocity.counts** *(s32 IN)* - Скорость медленной подачи
- **gmoccapy.jog.jog-velocity.count-enable** *(bit IN)* - Должно быть True, чтобы разрешить отсчеты
- **gmoccapy.feed.feed-override.counts** *(s32 IN)* - переопределение подачи
- **gmoccapy.feed.feed-override.count-enable** *(bit IN)* - Должно быть True, чтобы разрешить отсчеты
- **gmoccapy.feed.reset-feed-override** *(bit IN)* - сбросить переопределение подачи на 0%
- **gmoccapy.spindle.spindle-override.counts** *(s32 IN)* - переопределение шпинделя
- **gmoccapy.spindle.spindle-override.count-enable** *(bit IN)* - Должно быть True, чтобы разрешить отсчеты

- **gmoccapy.spindle.reset-spindle-override** *(bit IN)* - сбросить переопределение шпинделя на 0%
- **gmoccapy.rapid.rapid-override.counts** *(s32 IN)* - Максимальная скорость станка
- **gmoccapy.rapid.rapid-override.count-enable** *(bit IN)* - Должно быть True, чтобы разрешить отсчеты

Для подключения потенциометров используйте следующие контакты:

- **gmoccapy.jog.jog-velocity.direct-value** *(float IN)* - Чтобы отрегулировать ползунок скорости медленной подачи
- **gmoccapy.jog.jog-velocity.analog-enable** *(bit IN)* - Должно быть True, чтобы разрешить аналоговые входы
- **gmoccapy.feed.feed-override.direct-value** *(float IN)* - Настройка ползунка переопределения подачи
- **gmoccapy.feed.feed-override.analog-enable** *(bit IN)* - Должно быть True, чтобы разрешить аналоговые входы
- **gmoccapy.spindle.spindle-override.direct-value** *(float IN)* - Чтобы отрегулировать ползунок переопределения шпинделя
- **gmoccapy.spindle.spindle-override.analog-enable** *(bit IN)* - Должно быть True, чтобы разрешить аналоговые входы
- **gmoccapy.rapid.rapid-override.direct-value** *(float)* - Чтобы отрегулировать ползунок максимальной скорости
- **gmoccapy.rapid.rapid-override.analog-enable** *(bit IN)* - Должно быть True, чтобы разрешить аналоговые входы

Кроме того, GMOCCAPY 3 предлагает дополнительные контакты HAL для управления новыми виджетами-слайдерами с помощью мгновенных переключателей. Значения того, насколько быстрым будет увеличение или уменьшение, необходимо задать в файле Glade. В будущем выпуске он будет интегрирован на страницу настроек.

СКОРОСТЬ

- **gmoccapy.spc_jog_vel.increase** *(bit IN)* - Пока значение True, значение ползунка будет увеличивать
- **gmoccapy.spc_jog_vel.decrease** *(bit IN)* - Пока значение True, значение ползунка будет уменьшаться
- **gmoccapy.spc_jog_vel.scale** *(float IN)* - Значение для масштабирования выходного значения (удобно менять единицы/мин на единицы/сек)
- **gmoccapy.spc_jog_vel.value** *(float OUT)* - Значение виджета
- **gmoccapy.spc_jog_vel.scaled-value** *(float OUT)* - Масштабированное значение виджета

ПОДАЧА

- **gmoccapy.spc_feed.increase** *(bit IN)* - Пока значение True, значение ползунка будет увеличиваться
- **gmoccapy.spc_feed.decrease** *(bit IN)* - Пока значение True, значение ползунка будет уменьшаться
- **gmoccapy.spc_feed.scale** *(float IN)* - Значение для масштабирования выходного значения (удобно менять единицы/мин на единицы/сек)
- **gmoccapy.spc_feed.value** *(float OUT)* - Значение виджета

- **gmoccapy.spc_feed.scaled-value** *(float OUT)* - Масштабированное значение виджета

SPINDLE

- **gmoccapy.spc_spindle.increase** *(bit IN)* - Пока значение True, значение ползунка будет увеличивать
- **gmoccapy.spc_spindle.decrease** *(bit IN)* - Пока значение True, значение ползунка будет уменьшатьс
- **gmoccapy.spc_spindle.scale** *(float IN)* - Значение для масштабирования выходного значения (удобно менять единицы/мин на единицы/сек)
- **gmoccapy.spc_spindle.value** *(float OUT)* - Значение виджета
- **gmoccapy.spc_spindle.scaled-value** *(float OUT)* - Масштабированное значение виджета

БЫСТРЫЕ ПЕРЕМЕЩЕНИЯ

- **gmoccapy.spc_rapid.increase** *(bit IN)* - Пока значение True, значение ползунка будет увеличиваться
- **gmoccapy.spc_rapid.decrease** *(bit IN)* - Пока значение True, значение ползунка будет уменьшаться
- **gmoccapy.spc_rapid.scale** *(float IN)* - Значение для масштабирования выходного значения (удобно менять единицы/мин на единицы/сек)
- **gmoccapy.spc_rapid.value** *(float OUT)* - Значение виджета
- **gmoccapy.spc_rapid.scaled-value** *(float OUT)* - Масштабированное значение виджета

Контакты типа float принимают значения от 0,0 до 1,0, что представляет собой процентное значение, которое вы хотите установить для ползунка.

**Warning**
Если вы используете оба типа подключения, не подключайте один и тот же ползунок к обоим контактам, поскольку влияние между ними не проверялось! К тому или иному типу соединения HAL могут быть подключены разные ползунки.

**Important**
Имейте в виду, что скорость медленной подачи зависит от состояния кнопки-черепахи. Это приведет к разным масштабам ползунка в зависимости от режима (черепаха или кролик). Пожалуйста, взгляните также на jog velocities и Turtle-Jog HAL контакт для получения более подробной информации.

**Example 10.1** Установка значения ползунка

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 HAL Контакты медленной подачи

Все оси, указанные в INI-файле, имеют контакты jog-plus и jog-minus, поэтому для медленной подачи осей можно использовать аппаратные переключатели мгновенного действия.

**Note**
Именование этих контактов HAL изменилось в GMOCCAPY 2.

Для стандартной конфигурации XYZ будут доступны следующие контакты HAL:

- **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)*

Если вы используете конфигурацию с 4 осями, тогда будет два дополнительных контакта:

- **gmoccapy.jog.jog-<your fourth axis letter >-plus** *(bit IN)*
- **gmoccapy.jog.jog-<your fourth axis letter >-minus** *(bit IN)*

Для оси C вы увидите:

- **gmoccapy.jog.axis.jog-c-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-c-minus** *(bit IN)*

#### 10.2.5.4 Скорость медленной подачи и контакт HAL Turtle-Jog

Скорость медленной подачи можно выбрать с помощью соответствующего ползунка. Масштаб ползунка будет изменен, если была переключена кнопка черепахи (та, которая показывает кролика или черепаху). Если кнопка не видна, возможно, она отключена на странице settings. Если на кнопке отображается значок кролика, шкала составляет от минимальной до максимальной скорости станка. Если на нем изображена черепаха, по умолчанию масштаб будет достигать только 1/20 максимальной скорости. Используемый разделитель можно настроить на странице settings.

Таким образом, с помощью сенсорного экрана гораздо проще выбирать меньшие скорости.

GMOCCAPY предлагает этот контакт HAL для переключения между скоростью подачи черепахи и кролика:

- **gmoccapy.jog.turtle-jog** *(bit IN)*

#### 10.2.5.5 HAL Контакты приращения медленной подачи

Приращения подачи, указанное в INI-файле, например

```
[DISPLAY]
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm
```

выбираются через контакты HAL, поэтому для выбора используемого приращения можно использовать аппаратный переключатель выбора. Для приращений, указанных в INI-файле, будет максимум 10 контактов HAL. Если вы укажете больше приращений в своем INI-файле, они будут недоступны из графического интерфейса, поскольку не будут отображаться.

Если у вас есть 6 приращений в вашем INI-файле, как в примере выше, вы получите **7** контактов:

- **gmoccapy.jog.jog-inc-0** *(bit IN)* - Этот фиксированный и будет представлять собой непрерывную медленную подачу.
- **gmoccapy.jog.jog-inc-1** *(bit IN)* - Первое приращение указано в INI-файле.
- **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 также предлагает контакт HAL для вывода выбранного приращения медленной подачи:

- **gmoccapy.jog.jog-increment** *(float OUT)*

#### 10.2.5.6 Контакт разблокировки оборудования

Чтобы иметь возможность использовать ключевой переключатель для разблокировки страницы настроек, экспортируется следующий контакт:

- **gmoccapy.unlock-settings** *(bit IN)* - Страница настроек разблокируется, если контакт находится в активном состоянии. Чтобы использовать этот контакт, вам необходимо активировать его на странице настроек.

#### 10.2.5.7 Контакты ошибок/предупреждений

- **gmoccapy.error** *(bit OUT)* - Указывает на ошибку, поэтому может загореться индикатор или даже может быть остановлен станок. Она будет сброшена при помощи контакта `gmoccapy.delete-me`
- **gmoccapy.delete-message** *(bit IN)* - Удалит первую ошибку и сбросит контакт `gmoccapy.error` в значение false после устранения последней ошибки.
- **gmoccapy.warning-confirm** *(bit IN)* - Подтверждает диалоговое окно с предупреждением подобно клику на OK

---

**Note**
Сообщения или информация пользователя не влияют на контакт `gmoccapy.error`, но контакт `gmoccapy.delete-message` удалит последнее сообщение, если не отображается ошибка!

---

#### 10.2.5.8 HAL Контакты созданного пользователем сообщения

GMOCCAPY можно настроить на реагирование на внешние ошибки, используя 3 различных сообщения пользователя:

статус

- **gmoccapy.messages.status** *(bit IN)* - Запускает диалог.

okdialog

- **gmoccapy.messages.okdialog** *(bit IN)* - Запускает диалог.
- **gmoccapy.messages.okdialog-waiting** *(bit OUT)* - Будет *1*, пока открыто диалоговое окно. Закрытие сообщения сбросит этот контакт.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bit IN)* - Запускает диалог.
- **gmoccapy.messages.yesnodialog-waiting** *(bit OUT)* - Будет *1*, пока открыто диалоговое окно. Закрытие сообщения сбросит этот контакт.
- **gmoccapy.messages.yesnodialog-response** *(bit OUT)* - Этот PIN-код изменится на *1*, если пользовате нажмет OK, а во всех остальных случаях он будет равен *0*. Этот контакт будет оставаться *1* до тех пор, пока диалоговое окно не будет вызвано снова.

Чтобы добавить сообщение, созданное пользователем, вам необходимо добавить сообщение в INI-файл в разделе DISPLAY. См. Конфигурация сообщений.

**Пример сообщения пользователя (INI-файл)**

```
MESSAGE_TEXT = LUBE FAULT
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = lube-fault

MESSAGE_TEXT = X SHEAR PIN BROKEN
MESSAGE_TYPE = status
MESSAGE_PINNAME = pin
```

Чтобы соединить эти новые контакты, вам нужно сделать это в HAL-файле postgui. Вот несколько примеров соединений, которые соединяют сигналы сообщения с другим местом в файле HAL.

**Пример подключения сообщений пользователя (HAL файл)**

```
net gmoccapy-lube-fault gmoccapy.messages.lube-fault
net gmoccapy-lube-fault-waiting gmoccapy.messages.lube-fault-waiting
net gmoccapy-pin gmoccapy.messages.pin
```

Для получения дополнительной информации о файлах HAL и команде net см. HAL Basics.

#### 10.2.5.9 Контакты обратной связи шпинделя

Для обратной связи шпинделя имеется два контакта:

- **gmoccapy.spindle_feedback_bar** *(float IN)* - Значок для отображения скорости шпинделя на шкале шпинделя.
- **gmoccapy.spindle_at_speed_led** *(bit IN)* - Контакт для включения is-at-speed-led.

#### 10.2.5.10 Контакты для отображения информации о ходе выполнения программы

Есть три контакта, которые дают информацию о ходе программы:

- **gmoccapy.program.length** *(s32 OUT)* - Показывает общее количество строк программы.
- **gmoccapy.program.current-line** *(s32 OUT)* - Указывает текущую рабочую строку программы.
- **gmoccapy.program.progress** *(float OUT)* - Показывает ход выполнения программы в процентах.

Значения могут быть неточными, если вы работаете с подпрограммами или большими процедурами переназначения. Также циклы будут вызывать разные значения.

#### 10.2.5.11 Контакты, относящиеся к инструменту

**Контакты смены инструмента** Эти контакты предназначены для использования внутреннего диалога смены инструмента GMOCCAPY, аналогичного тому, который известен в AXIS, но с некоторыми модификациями. Таким образом, вы получите не только сообщение о смене инструмента на *инструмент номер 3*, но и описание этого инструмента, например *фреза с канавками 7,5 мм 3*. Информация берется из таблицы инструментов, поэтому вам решать, что отображать.

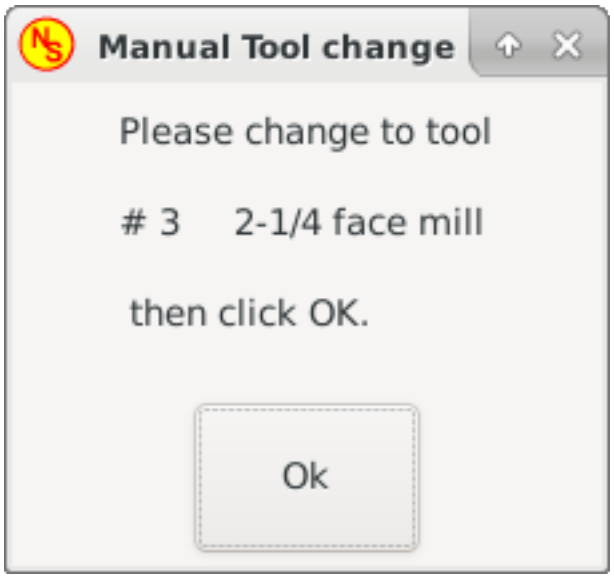



Figure 10.15: Диалоговое окно смены инструмента GMOCCAPY

- **gmoccapy.toolchange-number** *(s32 IN)* - Номер инструмента, подлежащего замене
- **gmoccapy.toolchange-change** *(bit IN)* - Указывает на необходимость замены инструмента
- **gmoccapy.toolchange-changed** *(bit OUT)* - Указывает на то, что инструмент был изменен
- **gmoccapy.toolchange-confirm** *(bit IN)* - Подтверждает смену инструмента

Обычно для ручной смены инструмента их подключают так:

```
net tool-change gmoccapy.toolchange-change <= iocontrol.0.tool-change
net tool-changed gmoccapy.toolchange-changed <= iocontrol.0.tool-changed
net tool-prep-number gmoccapy.toolchange-number <= iocontrol.0.tool-prep-number
net tool-prep-loop iocontrol.0.tool-prepare <= iocontrol.0.tool-prepared
```

**Note**
Пожалуйста, обратите внимание, что эти соединения должны быть выполнены в HAL-файле postgui.

**Контакты смещения инструмента** Эти контакты позволяют отображать активные значения коррекции инструмента по X и Z в информационном поле инструмента. Вы должны знать, что они активны только после отправки G43.

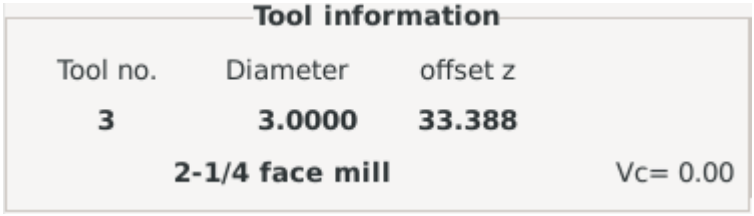



Figure 10.16: Область информации об инструменте

- **gmoccapy.tooloffset-x** *(float IN)*
- **gmoccapy.tooloffset-z** *(float IN)*

**Note**
Линия tooloffset-x не требуется на фрезерном станке и не будет отображаться на фрезерном станке с тривиальной кинематикой.

Чтобы отобразить текущие смещения, контакты должны быть соединены следующим образом в файле HAL postgui:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

**Important**
Обратите внимание, что GMOCCAPY самостоятельно обновляет смещения, отправляя G43 после любой смены инструмента, **но не в автоматическом режиме!**
Таким образом, написание программы возлагает на вас ответственность включать G43 после каждой смены инструмента!

### 10.2.6 Автоматическое измерение инструмента

GMOCCAPY предлагает интегрированное автоматическое измерение инструмента. Чтобы использовать эту функцию, вам потребуется выполнить некоторые дополнительные настройки, и вы можете использовать предлагаемый контакт HAL для получения значений в вашей собственной процедуре переназначения NGC.

**Important**
Перед началом первого теста не забудьте ввести высоту и скорость зонда на странице настроек! См. <<gmoccapy:tool-measurement, Страница настроек Измерение инструмента.

Возможно, было бы также неплохо взглянуть на видео об измерениях инструментов: см. видео.

Измерение инструментов в GMOCCAPY немного отличается от многих других ГИПов. Вам следует выполнить следующие шаги:

1. Коснитесь заготовки по X и Y.
2. Измерьте высоту вашего блока от основания, где расположен переключатель инструмента, до верхней поверхности блока (включая патрон и т. д.).
3. Нажмите кнопку высоты блока и введите измеренное значение.
4. Перейдите в автоматический режим и запустите программу.

Вот небольшой эскиз:

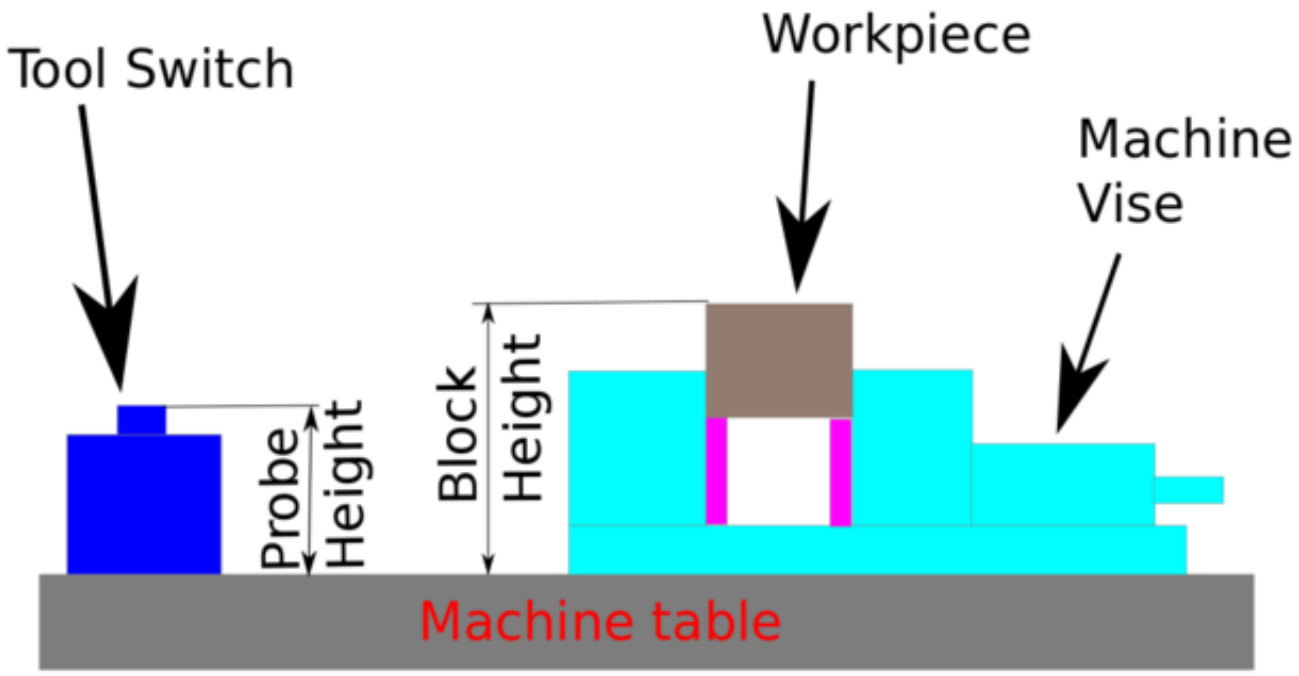



Figure 10.17: Данные измерений инструмента

При первой смене инструмента инструмент будет измерен, и смещение будет установлено автоматиче в соответствии с высотой блока. Преимущество метода GMOCCAPY в том, что вам не нужен калибровочный инструмент.

**Note**
Ваша программа должна содержать смену инструмента в начале! Инструмент будет измерен, даже если он уже использовался ранее, поэтому нет никакой опасности, если высота блока изменится. На YouTube есть несколько видеороликов, показывающих, как это сделать.

#### 10.2.6.1 Предоставленные контакты

GMOCCAPY предлагает пять контактов для измерения инструмента. Эти контакты в основном используются для чтения из подпрограммы G-кода, поэтому код может реагировать на разные значения.

- **gmoccapy.toolmeasurement** *(bit OUT)* - Разрешить или нет измерение инструмента
- **gmoccapy.blockheight** *(float OUT)* - Измеренное значение верхней части заготовки
- **gmoccapy.probeheight** *(float OUT)* - Высота переключения щупа
- **gmoccapy.searchvel** *(float OUT)* - Скорость поиска переключения датчика инструмента
- **gmoccapy.probevel** *(float OUT)* - Скорость измерения длины инструмента

#### 10.2.6.2 Изменения INI-файла

Измените свой INI-файл, включив в него следующие разделы.

**Раздел RS274NGC**

```
[RS274NGC]
# is the sub, with is called when a error during tool change happens, not needed on every  ↩
    machine configuration
ON_ABORT_COMMAND=O <on_abort> call

# The remap code
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

---

**Note**
Убедитесь, что для INI_VARS и HAL_PIN_VARS не установлено значение 0. По умолчанию для них установлено значение 1.

---

**Раздел Датчик инструмента** Положение датчика инструмента и начальное положение измерительн движения, все значения являются абсолютными координатами, кроме MAXPROBE, которое должно быть задано в относительном движении.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

**Раздел Изменить позицию** Это имя не называется TOOL_CHANGE_POSITION намеренно — **canon использует это имя и в противном случае будет мешать.** Позиция перемещения станка перед подачей команды смены инструмента. Все значения указаны в абсолютных координатах.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

**Раздел Python** Плагины Python служат интерпретатором и задачами.

```
[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 Необходимые файлы

Сначала создайте каталог "python" в папке конфигурации. Из <your_linuxcnc-dev_directory>/confi
скопируйте следующие файлы в только что созданную папку config_dir/python:

- toplevel.py
- remap.py
- stdglue.py

Из копии <your_linuxcnc-dev_directory>/configs/sim/gmoccapy/macros

- on_abort.ngc
- change.ngc

в каталог, указанный как SUBROUTINE_PATH, см. Раздел RS274NGC.

Откройте change.ngc в редакторе и раскомментируйте следующие строки (49 и 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

Возможно, вы захотите изменить этот файл, чтобы он больше соответствовал вашим потребностям.

#### 10.2.6.4 Необходимые соединения HAL

Подключите датчик инструмента к вашему файлу HAL следующим образом:

```
net probe  motion.probe-input <= <your_input_pin>
```

Строка может выглядеть так:

```
net probe  motion.probe-input <= parport.0.pin-15-in
```

В файле postgui.hal добавьте следующие строки:

```
# The next lines are only needed if the pins had been connected before
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.tool-changed
unlinkp iocontrol.0.tool-prep-number
unlinkp iocontrol.0.tool-prepared

# link to GMOCCAPY toolchange, so you get the advantage of tool description on change  ↩
    dialog
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 Страница настроек

Чтобы войти на страницу, вам нужно будет нажать на и ввести код разблокировки, который по умолчанию равен **123**. Если вы хотите изменить его сейчас, вам придется отредактировать скрытый файл настроек, подробности см. в разделе the displaysection.

Страница разделена на три основные вкладки:

#### 10.2.7.1 Внешний вид

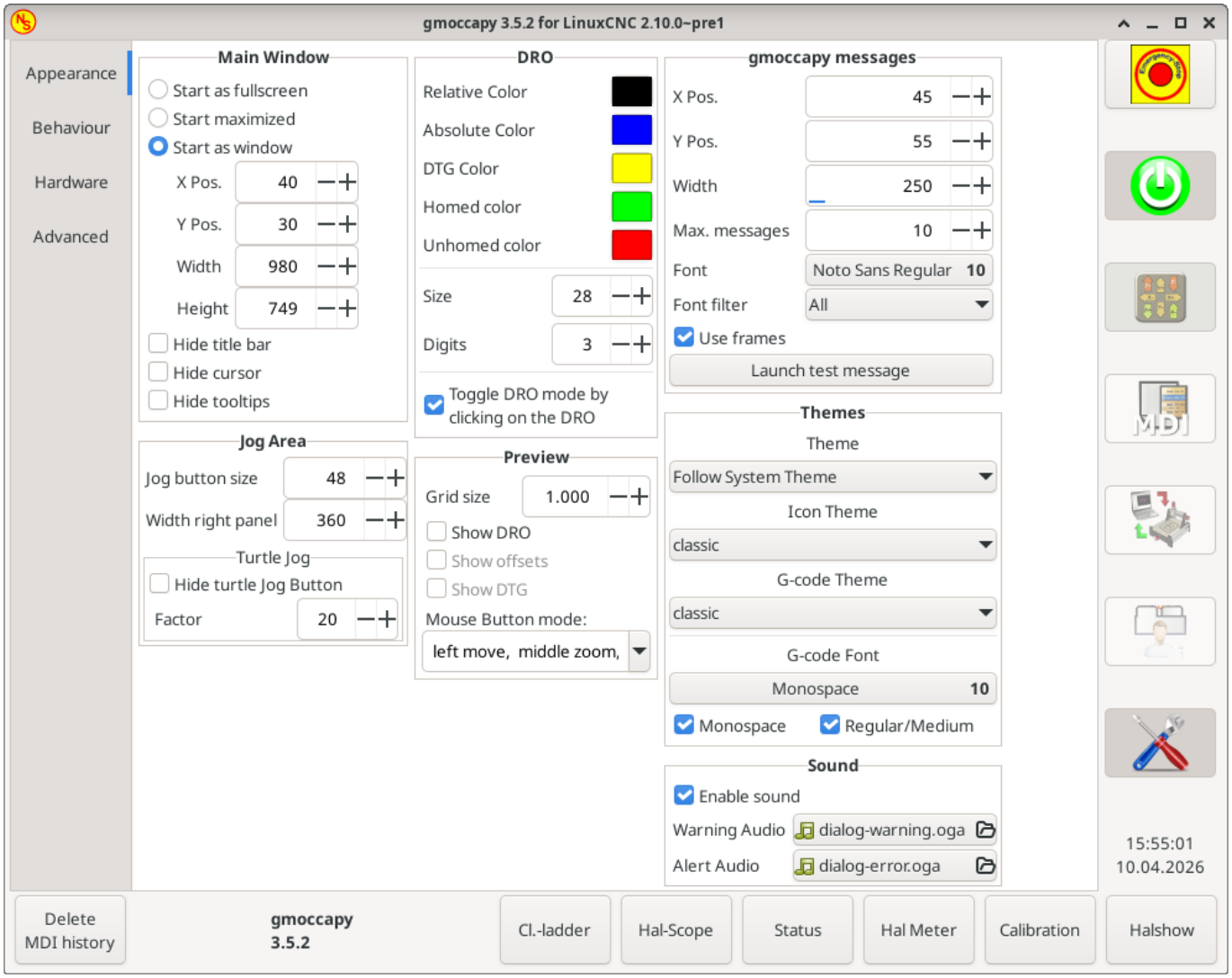



Figure 10.18: Страница настроек GMOCCAPY Внешний вид

На этой вкладке вы найдете следующие опции:

**Главное окно** Здесь вы можете выбрать способ запуска ГИП. Основной причиной этого было желание предоставить пользователю простой способ установки параметров запуска без необходимост изменения кода. У вас есть три варианта:

- *Start as full screen*
- *Start maximized*
- *Start as window* - Если вы выберете start as window, поля прокрутки для установки положения и размера станут активными. Будучи установленным, ГИП будет запускаться каждый раз на том же месте и с выбранным размером. Тем не менее, пользователь может изменить размер и положение с помощью мыши, но это не окажет никакого влияния на настройки.
- *hide title bar* - Позволяет скрыть строку заголовка. (по умолчанию: строка заголовка видна).

- *hide cursor* - Позволяет скрыть курсор, что очень полезно, если вы пользуетесь сенсорным экраном.
- *hide tooltips* - Скрывает подсказки инструментов.

Jog Area

- *Jog button size* - Change the size of the job buttons in the manual mode page
- *Width right panel* - Change the width of the right part of window (= width of the DRO)

**Очень медленная подача**

Эти настройки будут влиять на скорость медленной подачи.

- *Hide turtle jog button* - Скроет кнопку справа от ползунка скорости медленной подачи. Если вы скроете эту кнопку, позаботьтесь о том, чтобы был активирован ”режим кролика”, иначе вы не сможете подавать быстрее, чем скорость очень медленной (черепашей) подачи, которая рассчитывается с использованием коэффициента очень медленной подачи.
- *Turtle jog factor* - Устанавливает масштаб, применяемый для режима очень медленной (черепашей) подачи (нажатие кнопки показывает черепаху). Если вы установите коэффициент 20, скорость очень медленной подачи будет составлять 1/20 от максимальной. скорости станка.

---

**Note**
Этой кнопкой можно управлять с помощью контакта Turtle-Jog HAL.

---

**DRO** У вас есть возможность выбрать цвета фона для различных состояний УЦИ. Таким образом, пользователи, страдающие протанопией (слабостью красного и зеленого), могут выбирать правильные цвета.

По умолчанию цвета фона:

- *Relative Color* = черный
- *Absolute Color* = синий
- *DTG Color* = желтый

Цвет переднего плана УЦИ можно выбрать с помощью:

- *Homed Color* = green
- *Unhomed Color* = red

---

**Note**
Вы можете переключать режимы УЦИ (абсолютный, относительный, оставшееся расстояние), щелкая номер на УЦИ! Если вы щелкните левую букву УЦИ, появится всплывающее окно, которое позволит вам установить значение осей, что упростит установку значения, поскольку вам не нужно будет нажимать на нижнюю кнопку отключения касания.

---

- *size* - Позволяет установить размер шрифта УЦИ, по умолчанию — 28. Если вы используете экран большего размера, вы можете увеличить размер до 56. Если вы используете 4 оси, размер шрифта УЦИ будет составлять 3/4 значения, по соображениям пространства.
- *digits* - Устанавливает количество цифр УЦИ от 1 до 5.

**Note**
Имперская система покажет на одну цифру больше, чем метрическая. Таким образом, если вы используете имперские станочные единицы и установили цифровое значение на 1, вы вообще не получите цифр в метрических единицах.

- *Toggle DRO mode by clicking on the DRO* - If not active, a mouse click on the DRO will not take any action.
  By default this checkbox is active, so every click on any DRO will toggle the DRO readout from actual to relative to DTG (distance to go).
  Neverthereless a click on the axis letter will open the popup dialog to set the axis value.

Предварительный просмотр

- *Grid Size* - Устанавливает размер сетки окна предварительного просмотра. К сожалению, размер **должен быть установлен в дюймах**, даже если единицы измерения на вашем станке метрические. Мы надеемся исправить это в будущем выпуске.

**Note**
Сетка не будет отображаться в режиме перспективы.

- *Show DRO* - УЦИ также будет отображаться на панели предварительного просмотра, он всегда будет отображаться в полноразмерном предварительном просмотре.
- *Show DTG* - Будет показывать DTG (прямое расстояние до конечной точки) на панели предварительно просмотра, если параметр Show DRO активен. В противном случае только в полноразмерном просмотре.
- *Show Offsets* - Отображает смещения на панели предварительного просмотра, когда активен параметр Show DRO. В противном случае только в полноразмерном просмотре.
- *Mouse Button Mode* - Это поле со списком позволяет выбрать поведение кнопок мыши для вращения, перемещения или масштабирования в пределах предварительного просмотра:
  - левая вращение, средняя перемещение, правая масштабирование
  - левая масштабирование , средняя перемещение, правая вращение
  - левая перемещение, средняя вращение, правая масштабирование
  - левая масштабирование, средняя вращение, правая перемещение
  - левая перемещение, средняя масштабирование, правая вращение
  - левая вращение, средняя масштабирование, правая перемещение

По умолчанию — перемещение влево, масштабирование по центру, поворот вправо.
Колесо мыши по-прежнему будет масштабировать предварительный просмотр в каждом режиме.

**Tip**
Если вы выберете элемент в предварительном просмотре, выбранный элемент будет считаться центром вращения и в автоматическом режиме будет выделена соответствующая строка кода.

### Сообщения Gmoccapy

При этом отобразятся небольшие всплывающие окна с сообщением или текстом ошибки, подобные тем, которые известны в AXIS. Вы можете удалить конкретное сообщение, нажав на его кнопку закрытия. Если вы хотите удалить последнее, просто нажмите клавишу `WINDOWS` на клавиатуре или удалите все сообщения одновременно с помощью `Control + Space`.

Вы можете установить некоторые параметры:

- *X Pos* - Положение верхнего левого угла сообщения по оси X считается в пикселях от верхнего левого угла экрана.
- *Y Pos* - Положение верхнего левого угла сообщения по оси Y считается в пикселях от верхнего левого угла экрана.
- *Width* - Ширина окна сообщения.
- *Max. Messages* - Максимальное количество сообщений, которые вы хотите видеть одновременно. Если вы установите значение 10, $11^{e}$ сообщение удалит первое, поэтому вы увидите только последние 10.
- *Font* - Шрифт и размер, которые вы хотите использовать для отображения сообщений.
- *Font filter* - A filter for the font chooser above.
- *Use frames* - Если вы активируете флажок, каждое сообщение будет отображаться в рамке, поэтому различать сообщения будет намного проще. Но вам понадобится немного больше места.
- *Launch test message*-button - Появится сообщение, так что вы сможете увидеть изменения ваших настроек без необходимости генерировать ошибку.

Themes

- *Theme* - This lets the user select what desktop theme to apply. By default ”Follow System Theme” is set.
- *Icon Theme* - Change the icon theme. Currently there are three themes available:
  - classic
  - material
  - material light

  For more information and how to create custom icon themes, see section Icon Theme.
- *G-code Theme* - Select a colour theme for the G-code viewer/editor.
- *G-code Font* - Select a font for the G-code viewer/editor. Default is ”monospace 10”.

**Sound** Select what error and messages sounds should be played and if sound should be played at all.

#### 10.2.7.2 Behaviour

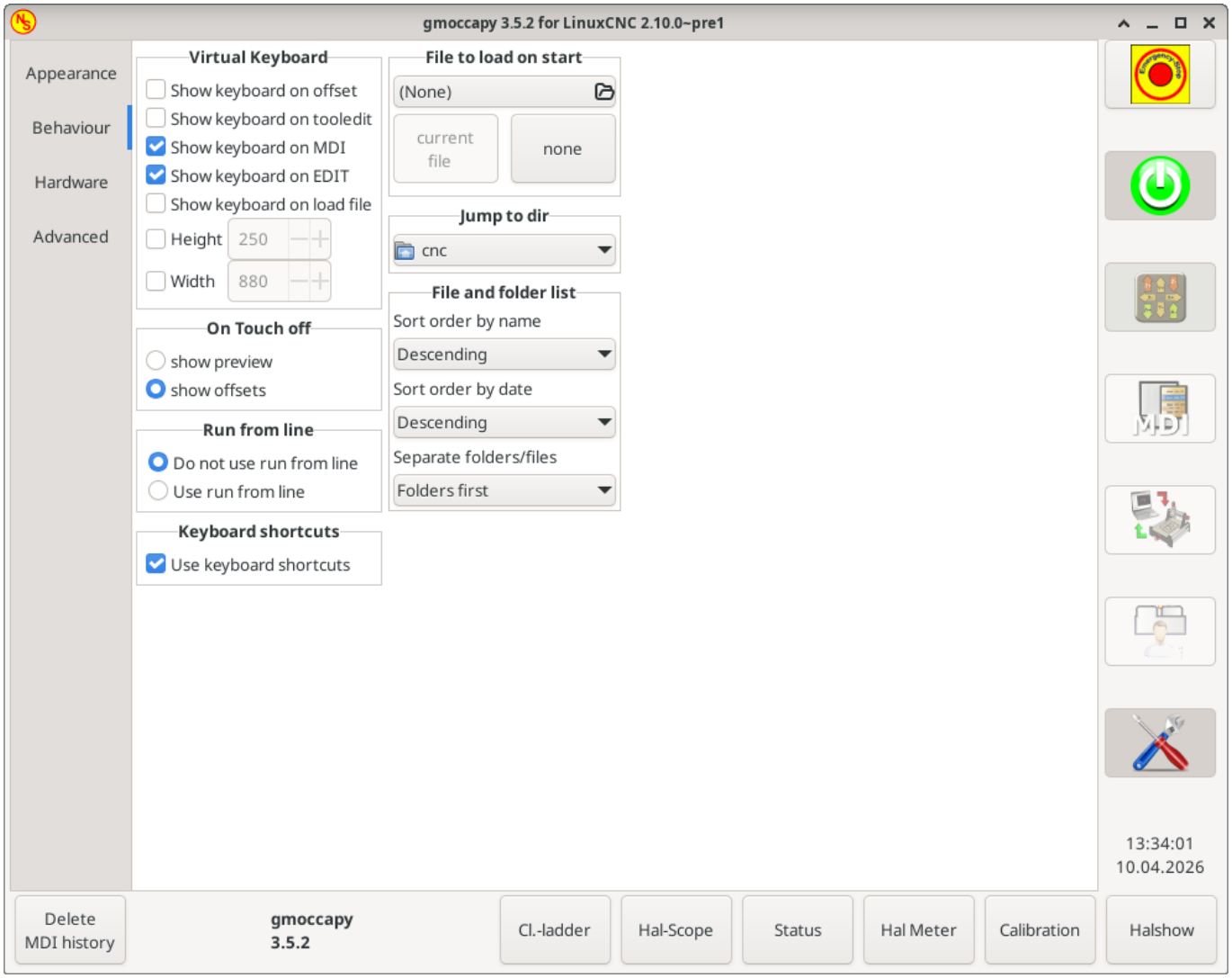



Figure 10.19: GMOCCAPY settings page Behaviour

**Virtual Keyboard** Флажки позволяют пользователю выбрать, хочет ли он, чтобы встроенная клавиатура отображалась немедленно при входе в режим MDI, на страницу смещений, в виджет редактирования инструментов или при открытии программы в режиме РЕДАКТИРОВАНИЯ. Эти настройки не повлияют на кнопку клавиатуры в нижнем списке кнопок, поэтому вы можете показать или скрыть клавиатуру, нажав кнопку.

Настройка по умолчанию:

- *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

**Note**
Если этот раздел не является изменяемым, у вас не установлена виртуальная клавиатура, поддерживаются только *onboard* и *matchbox-keyboard*.

**Note**
Если раскладка клавиатуры неправильная (т. е. нажатие Y дает Z, значит раскладка установлена неправильно в соответствии с настройками вашего региона). Для встроенной системы это можно решить с помощью небольшого командного файла следующего содержания:

```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

Буквы ”de” обозначают немецкий язык, вам придется установить их в соответствии с настройками вашего региона. Просто запустите этот файл перед запуском LinuxCNC, это также можно сделать, добавив стартер в локальную папку.

```
./config/autostart
```

Чтобы макет устанавливался автоматически при запуске.
Для миниатюрной клавиатуры придется делать свою раскладку, для немецкой раскладки спрашивайте на форуме.

**GMOCCAPY со встроенной клавиатурой в режиме редактирования**

**Включить касание** Это дает возможность выбрать, показывать ли вкладку предварительного просмотра или вкладку страницы смещения когда вы входите в режим касания путем нажатия соответствующей нижней кнопки.

- *show preview*
- *show offsets*

**Run from Line** Вы можете разрешить или запретить запуск с линии. Это сделает соответствующую кнопку нечувствительной (серой) и пользователь не сможет использовать эту опцию. По умолчанию запуск с линии отключен .

---

**Warning**
Не рекомендуется использовать запуск со строки, поскольку LinuxCNC не будет обрабатывать предыдущие строки кода перед начальной строкой. Поэтому ошибки или сбои вполне вероятны.

---

**Горячие клавиши** Некоторые пользователи хотят управлять станком с помощью кнопок клавиатуры, а есть другие, которые никогда этого не позволят. Таким образом, каждый может выбрать, использовать их или нет.
По умолчанию сочетания клавиш отключены. Их можно активировать флажком

- *Use keyboard shortcuts*

**Warning**
Не рекомендуется использовать клавиатуру для медленной подачи, так как это представляет серьезную опасность для оператора и станка.

Пожалуйста, будьте осторожны, если вы используете токарный станок, тогда сочетания клавиш будут другими, см. Lathe Specific Section.

**General**

- *F1* - Триггер Estop (будет работать, даже если сочетания клавиш отключены)
- *F2* - Включить/выключить станок
- *F3* - Ручной режим
- *F5* - MDI режим
- *ESC* - Прервать

**В ручном режиме**

- *Arrow_Left* или *NumPad_Left* - Медленная подача X минус
- *Arrow_Right* или *NumPad_Right* - Медленная подача X плюс
- *Arrow_up* или *NumPad_Up* - Медленная подача Y плюс
- *Arrow_Down* или *NumPad_Down* - Медленная подача Y минус
- *Page_Up* или *NumPad_Page_Up* - Медленная подача Z плюс
- *Page_Down* или *NumPad_Page_Down* - Медленная подача Z минус

**В автоматическом режиме**

- *R* или *r* - Запустить программу
- *P* или *p* - Приостановить программу
- *S* или *s* - Возобновить программу
- *Control* + *R* или *Control* + *r* - Перезагрузить загруженный файл

**Обработка сообщений (см. Поведение и внешний вид сообщения)**

- *WINDOWS* - Удалить последнее сообщение
- *Control* + *Space* - Удалить все сообщения

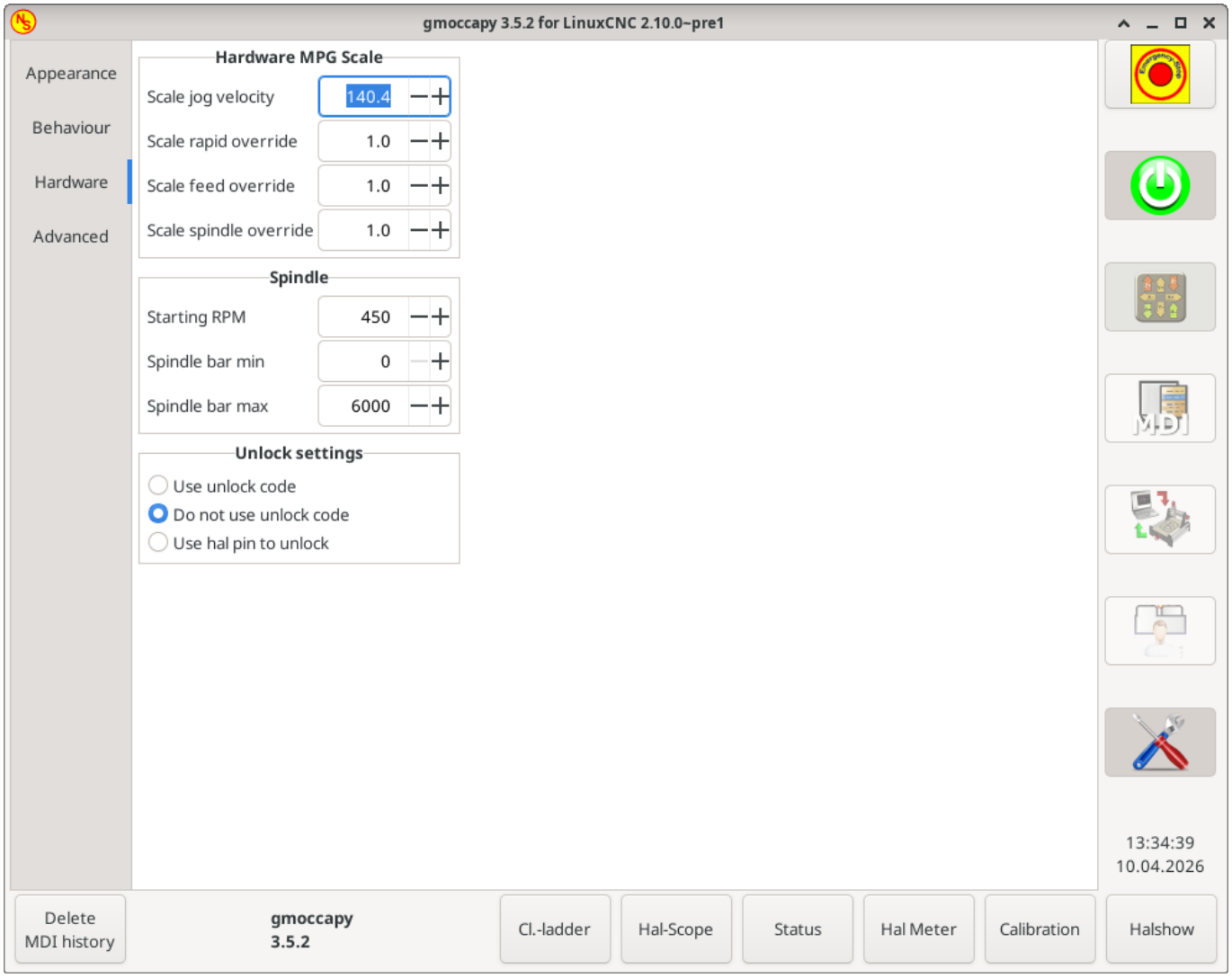
**Файл для загрузки при запуске** Выберите файл, который вы хотите загрузить при запуске. Если файл загружен, его можно установить, нажав текущую кнопку. Чтобы избежать загрузки какой-либо программы при запуске, просто нажмите кнопку None.

На экране выбора файла будут использоваться фильтры, которые вы установили в файле INI. Если фильтры не заданы, вы увидите только **файлы NGC**. Путь будет установлен в соответствии с настройками INI в `[DISPLAY] PROGRAM_PREFIX`.

**Перейти в каталог** Здесь вы можете указать каталог, в который будет осуществляться переход при нажатии соответствующей кнопки в диалоге выбора файла.

**File and folder list** For the file chooser page it is possible to change the sorting options with these parameters.

#### 10.2.7.3 Оборудование




**Масштаб аппаратного РГИ** Для различных контактов HAL для подключения маховичков РГИ вы можете выбрать отдельные масштабы, которые будут применяться. Основной причиной этого был мой собственный тест по решению этой проблемы через соединения HAL, в результате которого получился очень сложный файл HAL. Представьте себе пользователя, у которого есть маховичок РГИ со 100 имп/об и он хочет замедлить мак. скорость от 14000 до 2000 мм/мин, что требует 12000 импульсов, что соответствует 120 оборотам колеса! Или другой пользователь, у которого есть маховичок РГИ с 500 имп/об, и он хочет установить коррекцию шпинделя с пределами от 50 до 120%, чтобы он переходил от минимального до максимального значения за 70 импульсов, то есть даже не за 1/4 оборота.

По умолчанию все масштабы задаются с помощью расчета:

```
(MAX - MIN)/100
```

Spindle

- *Starting RPM* - Устанавливает частоту вращения, которая будет использоваться, если шпиндель запущен и значение S не установлено.

**Note**
Это значение будет задано в соответствии с вашими настройками в [DISPLAY] DEFAULT_SPINDLE_SPEED вашего INI-файла. Если вы измените настройки на странице настроек, с этого момента это значение будет использоваться по умолчанию, ваш INI-файл не будет изменен.

- *Spindle bar min* и *Spindle bar max* - Устанавливает пределы шпинделя, отображаемые в рамке INFO на главном экране.

  Значения по умолчанию:
  MIN = 0
  MAX = 6000

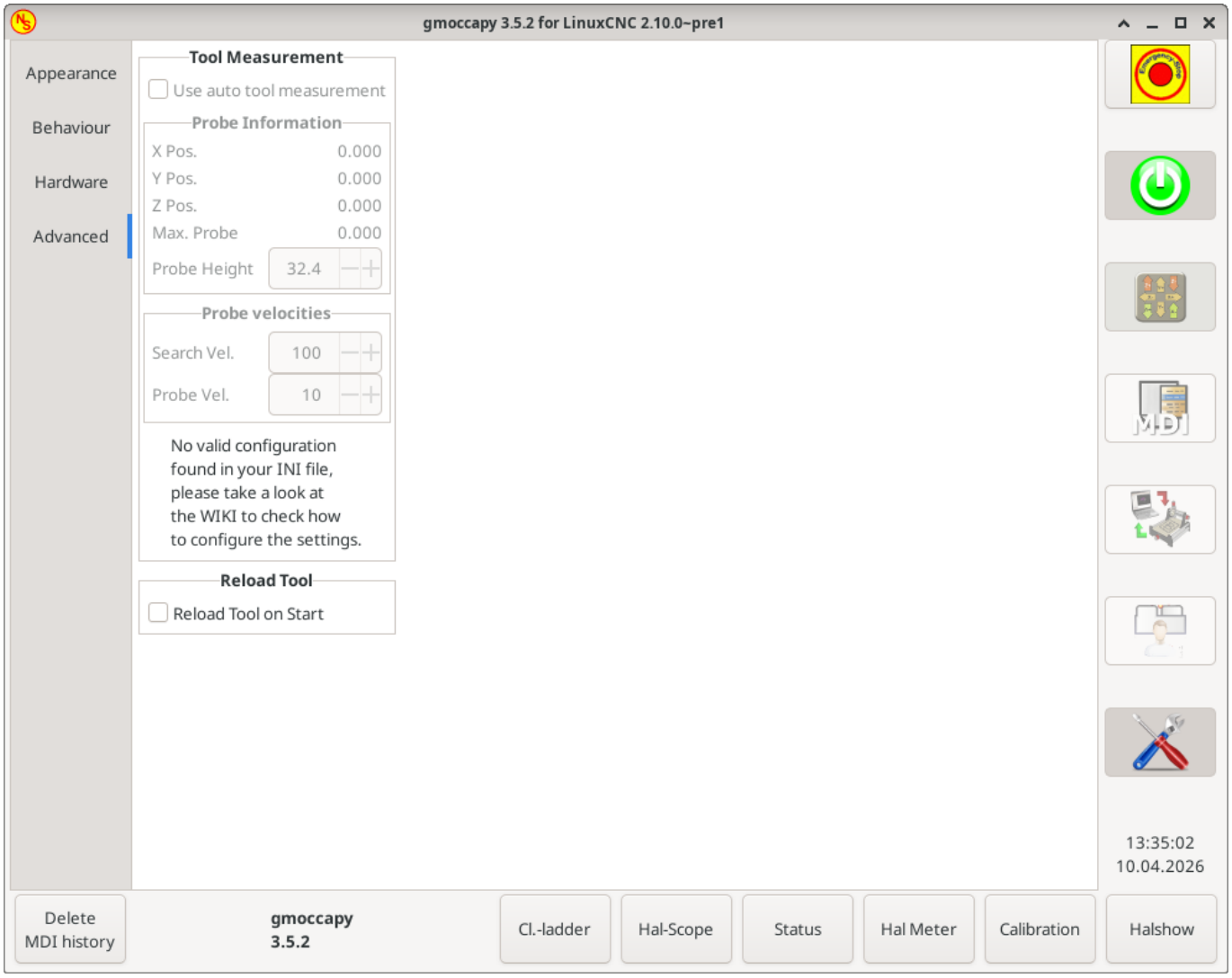
**Note**
Это не ошибка, дающая неверные значения. Если вы дадите максимум 2000, а ваш шпиндель делает 4000 об/мин, то только показания шкалы будут неверными на скоростях выше 2000 об/мин.

**Unlock settings** Есть три варианта разблокировки страницы настроек:

- *Use unlock code* - Пользователь должен ввести код для входа.
- *Do not use unlock code* - Никакой проверки безопасности не будет.
- *Use HAL pin to unlock* - Аппаратный контакт должен иметь высокий уровень, чтобы разблокировать настройки, см. аппаратный разблокирующий контакт.

По умолчанию *use unlock code* (код по умолчанию = **123**).

#### 10.2.7.4 Расширенные настройки




**Измерение инструмента**

Пожалуйста, проверьте автоматическое измерение инструмента

**Note**
Если эта часть не редактируема, у вас нет допустимой конфигурации INI-файла для использования измерения инструмента.

- *Use auto tool measurement* - Если этот флажок установлен, после каждой смены инструмента будет выполняться измерение инструмента, результат будет сохранен в таблице инструментов, а после замены будет выполняться G43.

**Информация о зонде**

Следующая информация взята из вашего INI-файла и должна быть указана в абсолютных координатах:

- *X Pos.* - Положение X переключателя инструмента.
- *Y Pos.* - Положение Y переключателя инструмента.

- *Z Pos.* - Положение Z переключателя инструмента, мы будем быстро перемещаться по этой координате.
- *Max. Probe* - Расстояние для поиска контакта, будет ошибка, если в этом диапазоне не будет задан ни один контакт. Расстояние должно быть задано в относительных координатах, начиная движение от позиции Z, поэтому вам нужно указать отрицательное значение, чтобы идти вниз!
- *Probe Height* - Вы можете измерить высоту вашего переключателя щупа. Просто коснитесь основания, где расположен щуп датчика, и установите его на ноль. Затем замените инструмент и посмотрите значение tool_offset_z, то есть высоту, которую вы должны ввести здесь.

**Скорости зонда**

- *Search Vel.* - Скорость поиска переключателя инструмента. После контакта инструмент снова поднимется вверх, а затем снова пойдет к датчику с увеличением скорости датчика, так что вы получите лучшие результаты.
- *Probe Vel.* - Скорость второго движения к переключателю. Это должно быть медленнее, чтобы получить лучшие результаты касания. В режиме моделирования это закомментировано в файле macros/change.ngc, иначе пользователю придется дважды нажать кнопку проверки.

Перезагрузка инструмента::

- *Reload Tool on Start* - Загружает последний инструмент при запуске после возврата в исходное положение.

If checked, the tool in spindle will be saved on each change in the preference file, making it possible to reload the last mounted tool on start up. The tool will be loaded after all axes are homed, because before it is not allowed to execute MDI commands. If you use NO_FORCE_HOMING you can not use this feature, because the needed all_homed_signal will never be emitted.

### 10.2.8 Тема иконок

Темы иконок используются для настройки внешнего вида иконок GMOCCAPY.

GMOCCAPY поставляется с тремя различными темами иконок:

- *classic* - Классические иконки GMOCCAPY.
- *material* - Современная тема значков, вдохновленная иконками материалов Google, которая автоматически принимает цвет из выбранной темы рабочего стола.
- *material-light* - Создано на основе материала, но оптимизировано для легких тем рабочего стола.

Тема иконок, используемая в GMOCCAPY, представляет собой обычную тему иконок GTK, соответствую спецификации темы иконок freedestktop. Таким образом, любую действующую тему иконок GTK можно использовать в качестве темы иконок GMOCCAPY, если она содержит необходимые иконки.

GMOCCAPY сканирует следующие каталоги на предмет тем иконок:

- linuxcnc/share/gmoccapy/icons
- ~/.icons

#### 10.2.8.1 Пользовательская тема значков

Создать собственную тему иконок довольно просто. Все, что вам нужно, это текстовый редактор и, конечно же, нужные иконки в виде пиксельной или векторной графики. Подробную информацию о том, как именно создается тема иконок, можно найти в https://specifications.freedesktop.org/-icon-theme-spec/icon-theme-spec-latest.html [Спецификация темы значков Freedesktop].

Начните с создания пустого каталога с именем темы иконок. Поместите каталог в один из каталогов тем иконок GMOCCAPY. Затем нам нужен файл index.theme в корневой папке нашей темы иконок, который содержит необходимые метаданные для темы. Это простой текстовый файл, содержащий как минимум следующие разделы:

- [Icon Theme]

```
[Icon Theme]
Name=YOUR_THEME_NAME
Comment=A DESCRIPTION OF YOUR THEME
Inherits=hicolor
Directories=16x16/actions,24x24/actions,32x32/actions,48x48/actions,scalable/actions
```

  - Name: Имя вашей темы иконок.
  - Comment: Описание темы иконок.
  - Inherits: Тема иконок может быть производной от другой темы иконок, по умолчанию используется hicolor.
  - Directories: Разделенный запятыми список всех каталогов вашей темы значков.
    Каждый каталог обычно содержит все значки темы определенного размера, например 16x16/actions должен содержать все значки категории ”actions” размером 16x16 пикселей в виде пиксельной графики (например, файлы png). Особым случаем является каталог под названием ”scalable/actions”, он содержит масштабируемые значки, не привязанные к определенному размеру (например, файлы svg).
    Предоставляя версии значков разного размера, мы можем гарантировать красивый вид значка, если размеры разные, а также у нас есть возможность изменять значок в соответствии с его размером, например, значок размером 64x64 пикселей может содержать больше деталей, чем его версия 16x16 пикселей. .

- Для каждого каталога нам также нужно написать раздел в файле index.theme:

```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

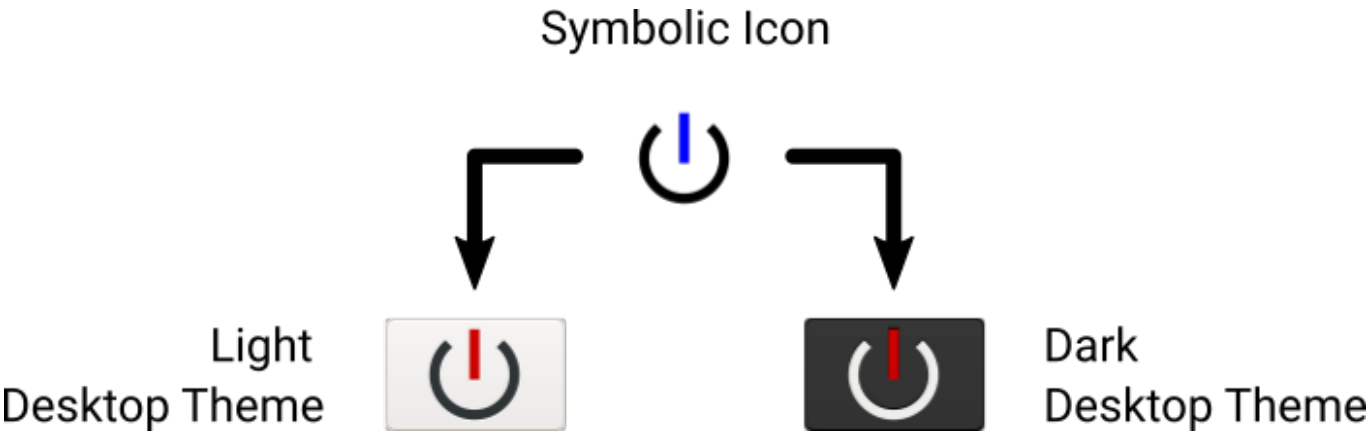
[scalable/actions]
Size=48
Type=Scalable
Context=Actions
```

  - Size: Номинальный размер значка в этом каталоге
  - Type: Фиксированный, пороговый или масштабируемый
  - Context: Предполагаемая ”категория” иконок

По сути, это все, что нужно для создания пользовательской темы иконок.

#### 10.2.8.2 Символические иконки

Символические иконки — это особый тип иконок, обычно монохромное изображение. Особенностью символических иконок является то, что во время выполнения иконки автоматически окрашиваются в соответствии с темой рабочего стола. Таким образом, можно создавать темы иконок, которые хорошо сочетаются как с темными, так и со светлыми темами рабочего стола (на самом деле, это не всегда лучший вариант, поэтому существует специальная тема ”material-light”).




Чтобы использовать символическую функцию, файл иконки должен иметь суффикс .symbolic.$ext (где $ext — это обычное расширение файла, например png), например ”power_on.symbolic.png”.

С таким именем GTK рассматривает это изображение как символическую иконку и применяет некоторую перекраску при загрузке иконки. Разрешено использовать только четыре цвета:

| Цвет | Hex Код | Description |
|---|---|---|
| черный | `#000000` | Основной цвет изменяется в соответствии с основным цветом темы рабочего стола. |
| красный | `#ff0000` | Успех: этот цвет указывает на ”успех” (обычно что-то зеленоватого цвета). |
| зеленый | `#00ff00` | Предупреждение: этот цвет означает ”предупреждение” (обычно что-то желтое/оранжевое). |
| синий | `#0000ff` | Ошибка: этот цвет указывает на ”ошибку” (обычно красный). |

---

**Tip**
Примеры символических иконок можно найти по адресу `linuxcnc/share/gmoccapy/icons/material`.

---

### 10.2.9 Специальный раздел для токарных станков

Если в INI-файле указано `LATHE = 1`, ГИП изменит свой внешний вид в соответствии с особыми потребностями токарного станка. В основном ось Y будет скрыта, а кнопки перемещения будут расположены в другом порядке.

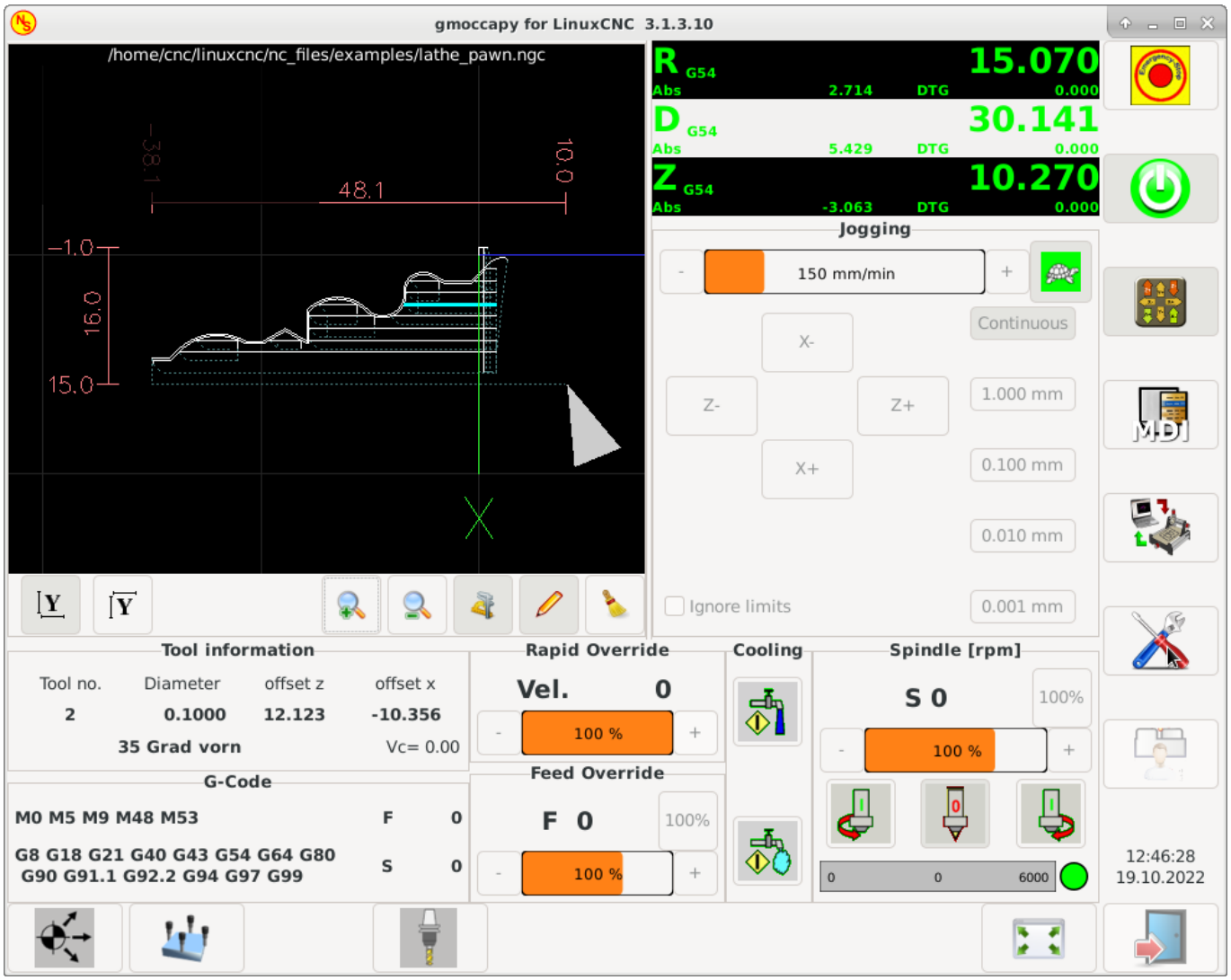



Figure 10.20: Обычный токарный станок

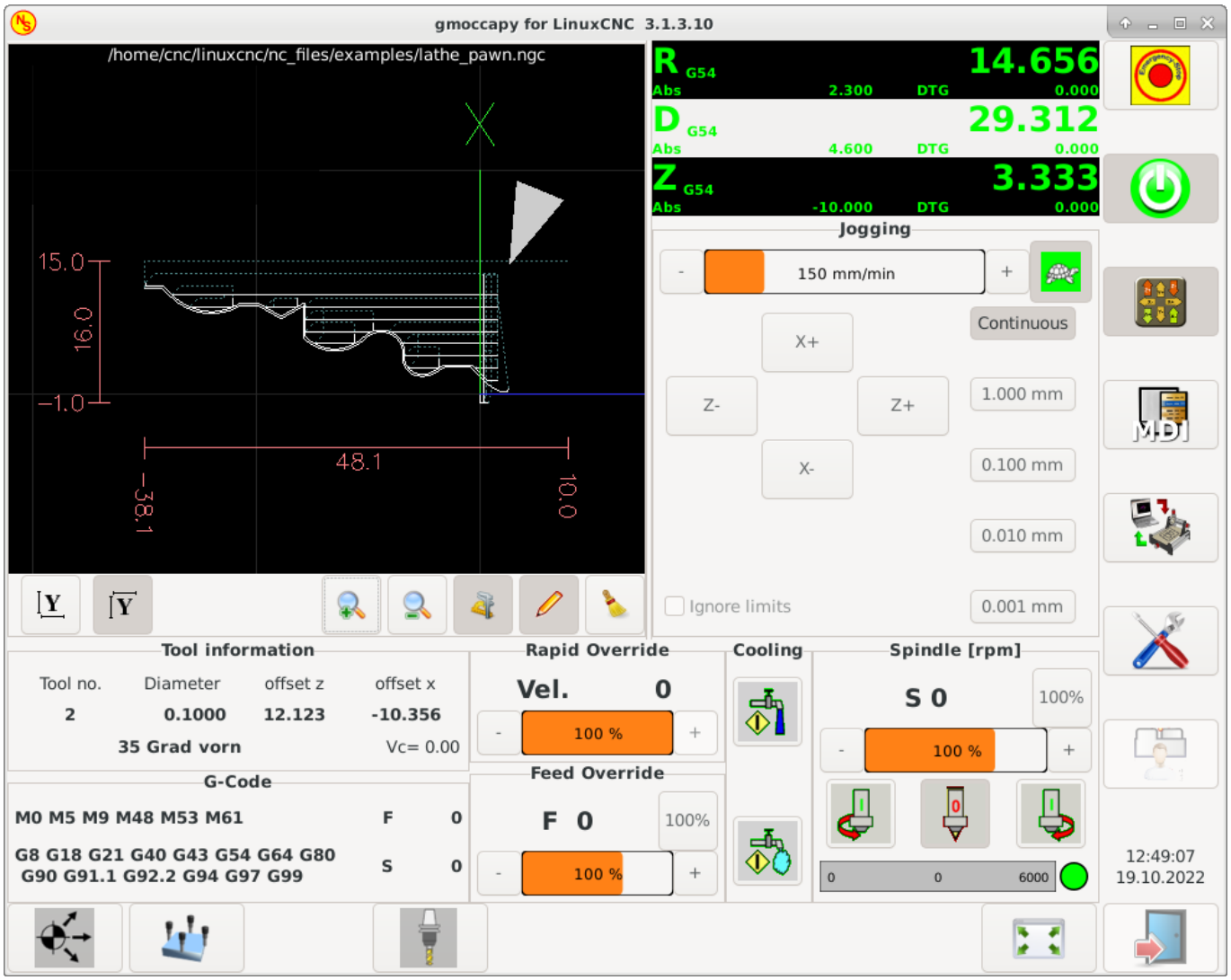



Figure 10.21: Задний инструмент токарного станка

Как вы видите, R УЦИ имеет черный фон, а D УЦИ — серый. Это будет меняться в зависимости от активного G-кода G7 или G8. Активный режим виден по черному фону, что означает, что на показанных изображениях G8 активен.

Следующее отличие от стандартного экрана — расположение кнопок поворота. X и Z поменялись местами, а Y исчезла. Вы заметите, что кнопки X+ и X- меняются местами в зависимости от обычного или токарного станка с задним инструментом.

Также изменится поведение клавиатуры:

Обычный токарный станок:

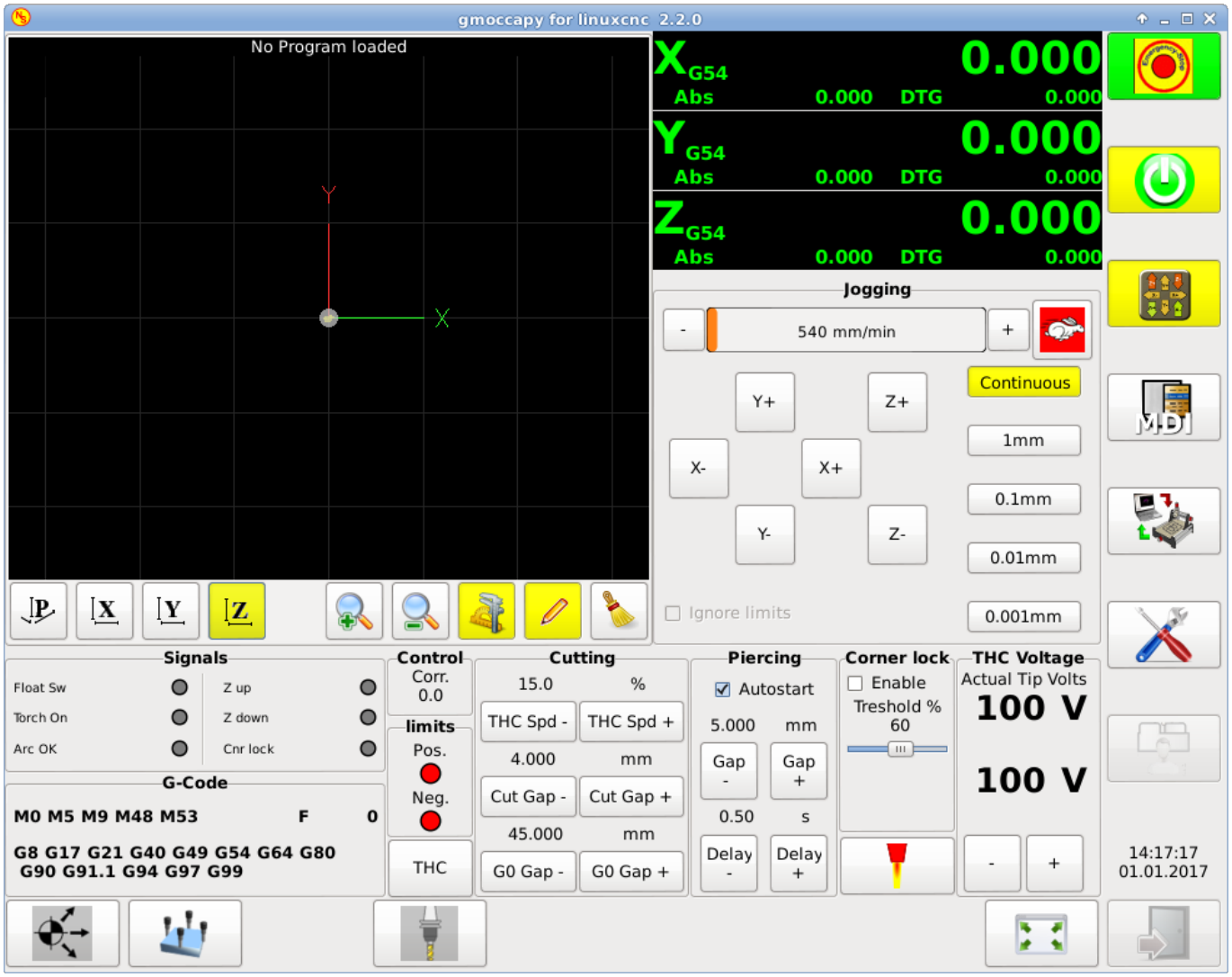
- *Arrow_Left* или *NumPad_Left* - Медленная подача Z минус
- *Arrow_Right* или *NumPad_Right* - Медленная подача Z плюс
- *Arrow_up* или *NumPad_Up* - Медленная подача X минус
- *Arrow_Down* или *NumPad_Down* - Медленная подача X плюс

Токарный станок с задним инструментом:

- *Arrow_Left* или *NumPad_Left* - Медленная подача Z минус
- *Arrow_Right* или *NumPad_Right* - Медленная подача Z плюс
- *Arrow_up* или *NumPad_Up* - Медленная подача X плюс
- *Arrow_Down* или *NumPad_Down* - Медленная подача X минус

В окне информации об инструменте будет отображаться не только смещение по оси Z, но и смещение по оси X, а в таблице инструментов отображается вся информация, относящаяся к токарному станку.

### 10.2.10 Специальный раздел плазмы




Существует очень хорошая WIKI, которая постоянно растет и поддерживается Мариусом, см. https://wiki.linuxcnc.org/cgi-bin/wiki.pl?Gmoccapy_plasma [вики-страница Plasma].

### 10.2.11 Видео на YouTube

Ниже представлена серия видеороликов, показывающих GMOCCAPY в действии. К сожалению, в этих видеороликах не показана последняя версия GMOCCAPY, но способ ее использования останется таким же, как и в текущей версии. Я буду обновлять видео как можно скорее.

#### 10.2.11.1 Базовое использование

https://youtu.be/O5B-s3uiI6g

#### 10.2.11.2 Имитация маховичков медленной подачи

https://youtu.be/ag34SGxt97o

#### 10.2.11.3 Страница настроек

https://youtu.be/AuwhSHRJoiI

#### 10.2.11.4 Имитированная аппаратная кнопка

German: https://youtu.be/DTqhY-MfzDE
English: https://youtu.be/ItVWJBK9WFA

#### 10.2.11.5 Пользовательские вкладки

https://youtu.be/rG1zmeqXyZI

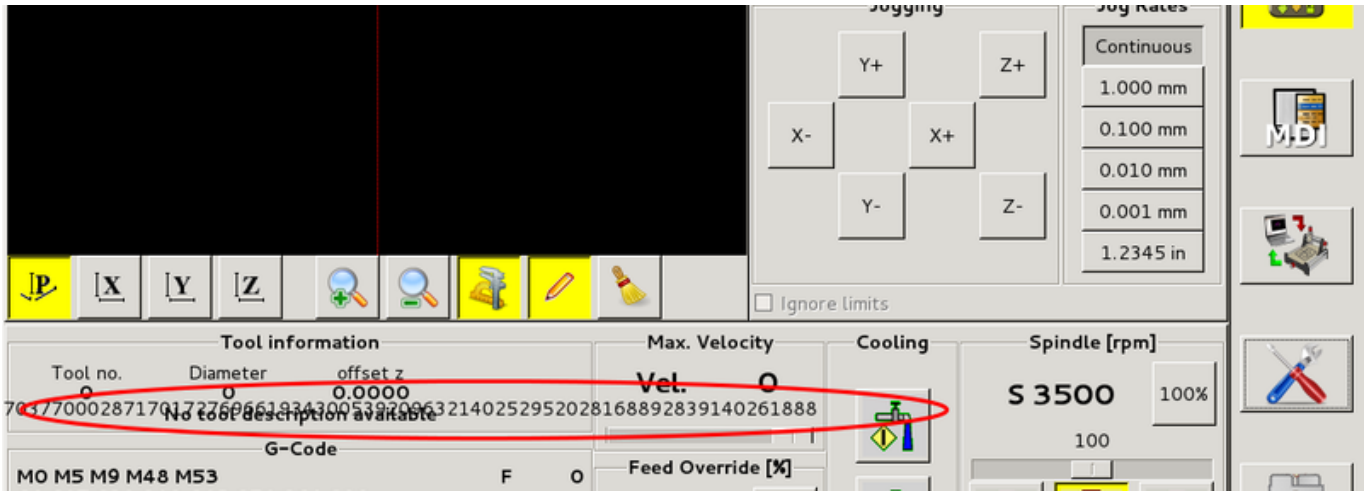
#### 10.2.11.6 Видео об измерениях инструментов

Моделирование автоматического измерения инструмента: https://youtu.be/rrkMw6rUFdk
Экран автоматического измерения инструмента: https://youtu.be/Z2ULDj9dzvk
Автоматический станок для измерения инструментов: https://youtu.be/1arucCaDdX4

### 10.2.12 Известные проблемы

#### 10.2.12.1 Странные числа в информационной зоне

Если вы получаете странные числа в информационной области GMOCCAPY, например:




Вы создали файл конфигурации, используя более старую версию StepConfWizard. Он сделал неверную запись в файле INI под [TRAJ] с именем MAX_LINEAR_VELOCITY = xxx. Измените эту запись на MAX_VELOCITY = xxx.

#### 10.2.12.2 Не завершающийся макрос

Если вы используете макрос без движения, например этот:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

o<zeroxy> endsub
m2
```

GMOCCAPY не увидит конец макроса, потому что интерпретатору необходимо изменить свое состояние на IDLE, но макрос даже не переводит интерпретатор в новое состояние. Чтобы избежать этого, просто добавьте линию G4 P0.1, чтобы получить необходимый сигнал. Правильный макрос будет такой:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

G4 P0.1

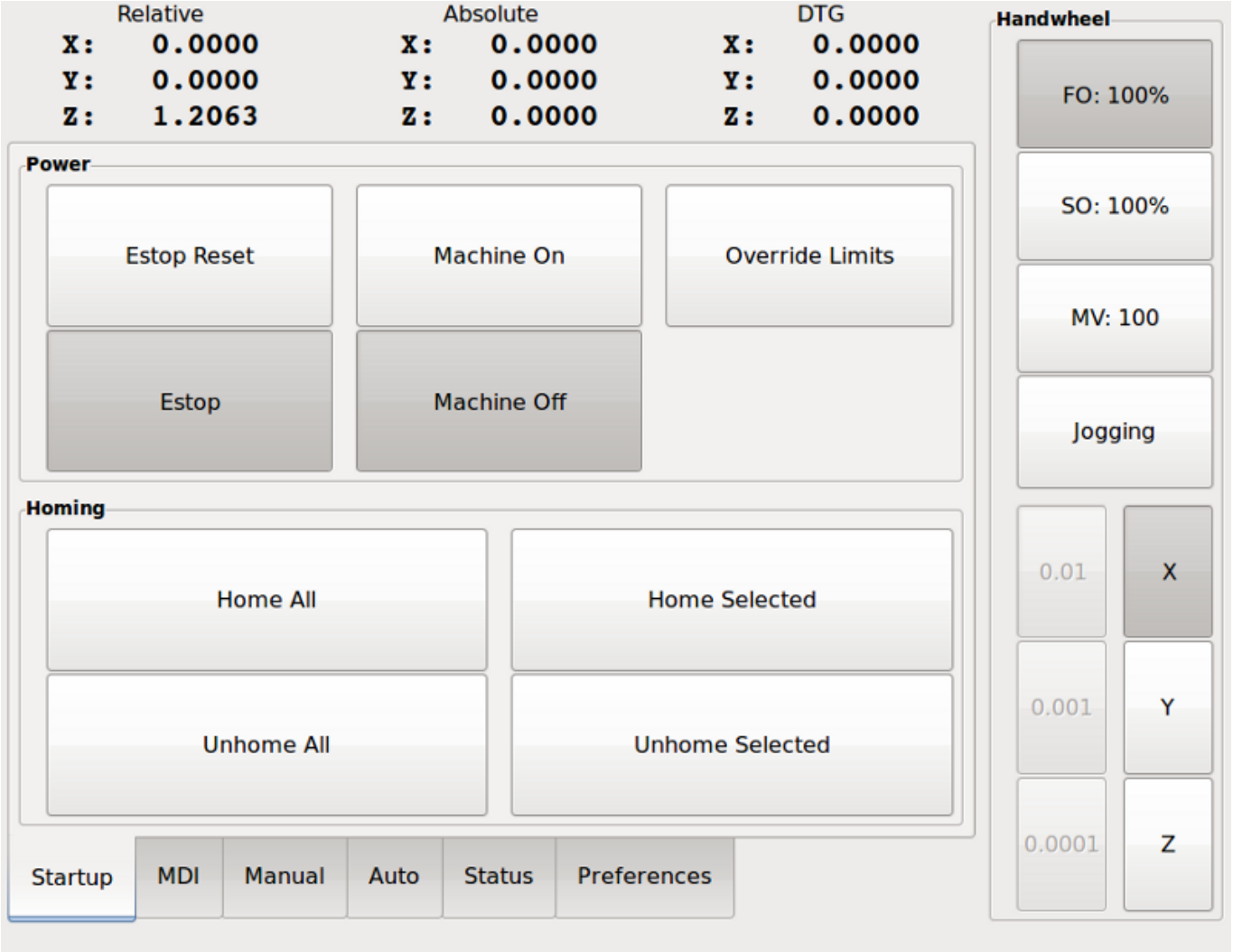
o<zeroxy> endsub
m2
```

## 10.3 Touchy ГИП

Touchy is a user interface for LinuxCNC meant for use on machine control panels, and therefore does not require keyboard or mouse.

It is meant to be used with a touch screen, and works in combination with a wheel/MPG and a few buttons and switches.

The *Handwheel* tab has radio buttons to select between *Feed Override*, *Spindle Override*, *Maximum Velocity* and *Jogging* functions for the wheel/MPG input. Radio buttons for axis selection and increment for jogging are also provided.




## 10.3.1 Panel Configuration

### 10.3.1.1 HAL connections

Touchy looks in the INI file, under the heading *[HAL]* for entries of *POSTGUI_HALFILE=<filename>*. Typically *<filename>* would be *touchy_postgui.hal*, but can be any legal filename. These commands are executed after the screen is built, guaranteeing the widget HAL pins are available. You can have multiple line of *POSTGUI_HALFILE=<filename>* in the INI. Each will be run one after the other in the order they appear in the INI file.

**Note**
Touchy used to require that you create a file named *touchy.hal* in your configuration directory (the directory your INI file is in). For legacy reasons this will continue to work, but INI based postgui files are preferred.

For more information on HAL files and the net command see the HAL Basics.

Touchy has several output pins that are meant to be connected to the motion controller to control wheel jogging:

- *touchy.jog.wheel.increment*, which is to be connected to the *axis*.N.*jog-scale* pin of each axis *N*.

- *touchy.jog.wheel*.N, which is to be connected to *axis*.N.*jog-enable* for each axis *N*.

**Note**
*N* represents the axis number 0-8.

- In addition to being connected to *touchy.wheel-counts*, the wheel counts should also be connected to *axis*.N.*jog-counts* for each axis *N*. If you use HAL component *ilowpass* to smooth wheel jogging, be sure to smooth only *axis*.N.*jog-counts* and not *touchy.wheel-counts*.

Required controls

- Abort button (momentary contact) connected to the HAL pin *touchy.abort*.
- Cycle start button (momentary contact) connected to *touchy.cycle-start*.
- Wheel/MPG, connected to *touchy.wheel-counts* and motion pins as described above.
- Single block (toggle switch) connected to *touchy.single-block*.

Optional controls

- For continuous jog, one center-off bidirectional momentary toggle (or two momentary buttons) for each axis, hooked to *touchy.jog.continuous.x.negative*, *touchy.jog.continuous.x.positive*, etc.
- If a quill up button is wanted (to jog Z to the top of travel at top speed), a momentary button connected to *touchy.quill-up*.

Optional panel lamps

- *touchy.jog.active* shows when the panel jogging controls are live.
- *touchy.status-indicator* is on when the machine is executing G-code, and flashes when the machine is executing but is in pause/feedhold.

#### 10.3.1.2 Recommended for any setup

- Estop button hardwired in the estop chain

### 10.3.2 Setup

#### 10.3.2.1 Enabling Touchy

To use Touchy, in the *[DISPLAY]* section of your INI file change the display selector line to *DISPLAY = touchy*.

#### 10.3.2.2 Preferences

When you start Touchy the first time, check the Preferences tab. If using a touchscreen, choose the option to hide the pointer for best results.

The Status Window is a fixed height, set by the size of a fixed font. This can be affected by the Gnome DPI, configured in System / Preferences / Appearance / Fonts / Details. If the bottom of the screen is cut off, reduce the DPI setting.

All other font sizes can be changed on the Preferences tab.

#### 10.3.2.3 Macros

Touchy can invoke O-word macros using the MDI interface. To configure this, in the *[MACROS]* section of the INI file, add one or more *MACRO* lines. Each should be of the following format:

```
MACRO=increment xinc yinc
```

In this example, increment is the name of the macro, and it accepts two parameters, named xinc and yinc.

Now, place the macro in a file named *increment.ngc*, in the *PROGRAM_PREFIX* directory or any directory in the *SUBROUTINE_PATH*.

It should look like:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Notice the name of the sub matches the file name and macro name exactly, including case.

When you invoke the macro by pressing the Macro button on the MDI tab in Touchy, you can enter values for xinc and yinc. These are passed to the macro as *#1* and *#2* respectively. Parameters you leave empty are passed as value 0.

If there are several different macros, press the Macro button repeatedly to cycle through them.

In this simple example, if you enter -1 for xinc and press cycle start, a rapid *G0* move will be invoked, moving one unit to the left.

This macro capability is useful for edge/hole probing and other setup tasks, as well as perhaps hole milling or other simple operations that can be done from the panel without requiring specially-written G-code programs.

## 10.4 Gscreen

### 10.4.1 Введение

Gscreen — это инфраструктура для отображения пользовательского экрана для управления LinuxCNC. Gscreen во многом заимствует у GladeVCP. GladeVCP использует редактор виджетов GTK GLADE для создания виртуальных панелей управления (VCP) методом укажи и кликни. Gscreen объединяет это с программированием на Python для создания экрана ГИП для работы на станке с ЧПУ.

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.

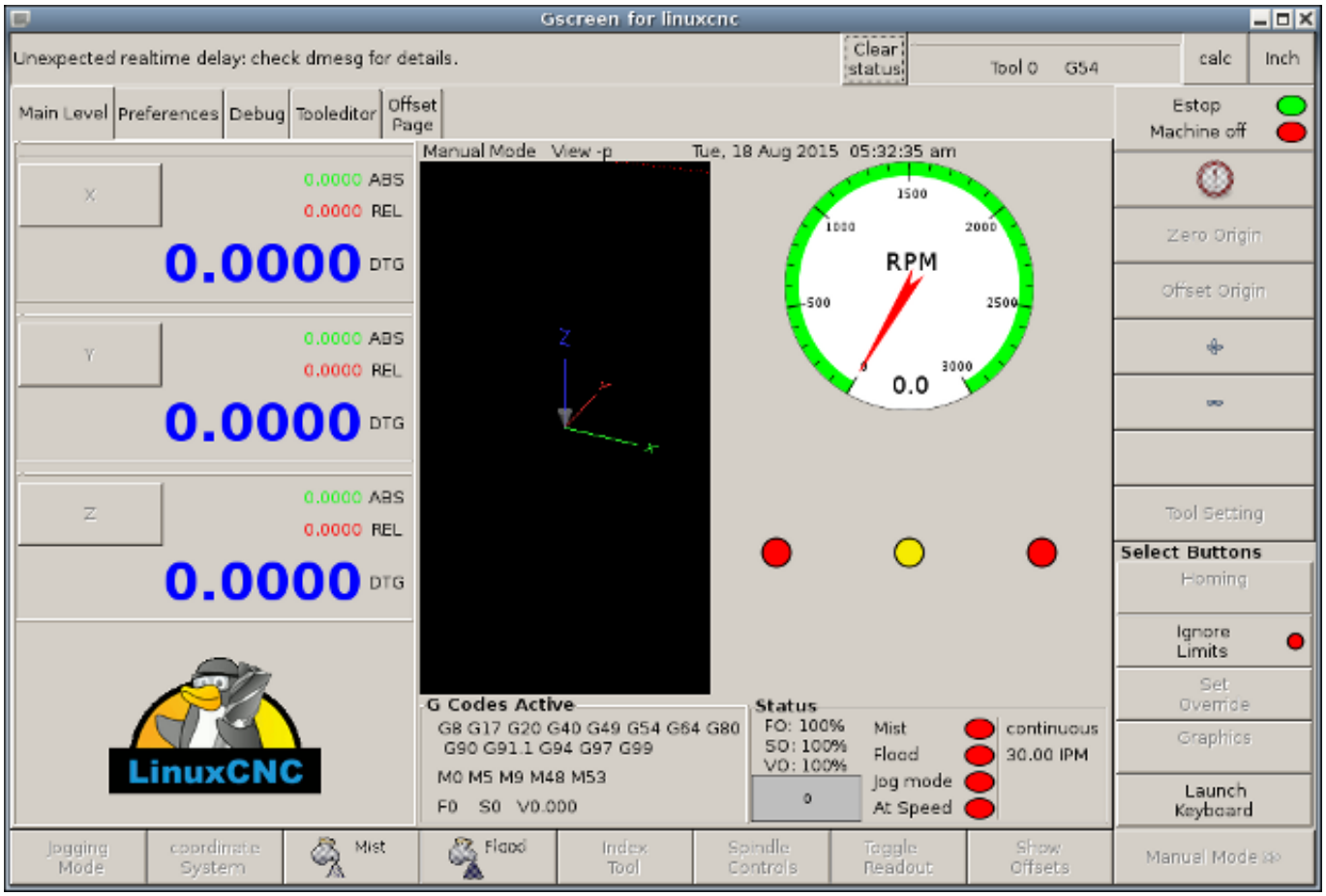



Figure 10.22: Gscreen Default Screen

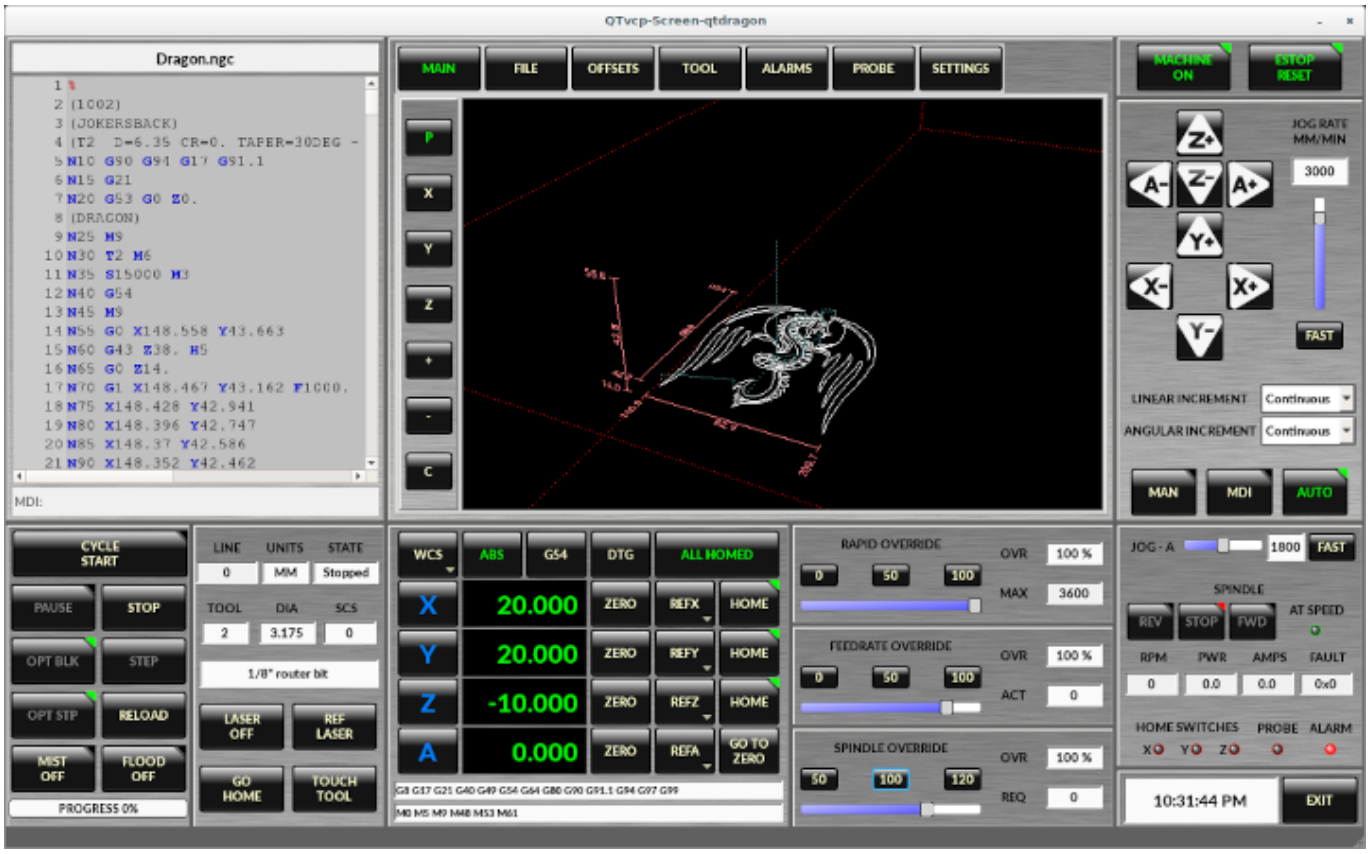



Figure 10.23: Gscreen Silverdragon Screen

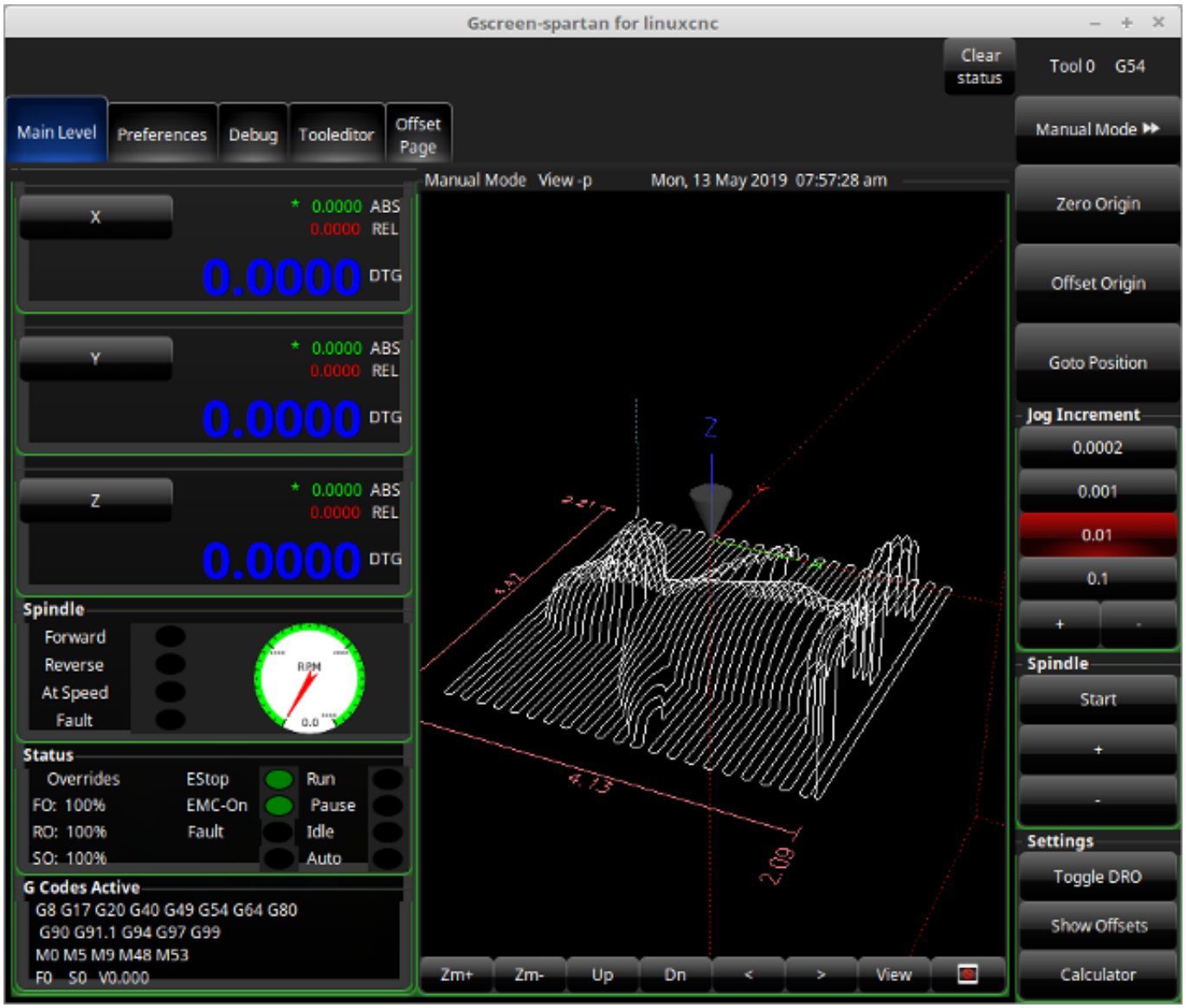



Figure 10.24: Gscreen Spartan Screen

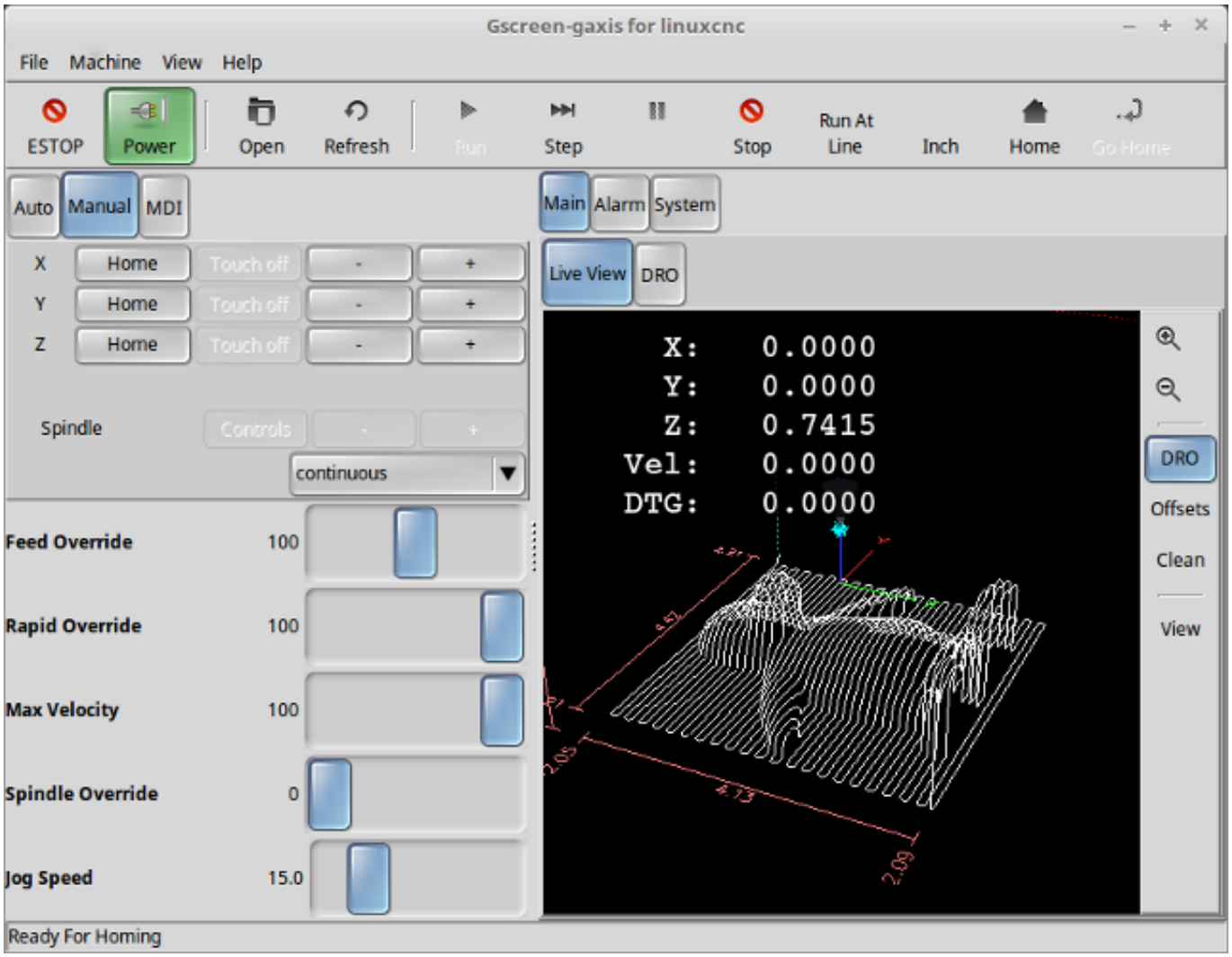



Figure 10.25: Gscreen Gaxis Screen

Figure 10.26: Gscreen Industrial Screen

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 File

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 Обзор

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.

**Local Glade Files** 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.

**Modifying Stock 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 Files** 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.

**Themes** Gscreen uses the PyGTK toolkit to display the screen. PyGTK is the Python language binding to GTK. GTK supports *themes*. Themes are a way to modify the look and feel of the widgets on the screen. For instance the color or size of buttons and sliders can be changed using themes. There are many GTK themes available on the web. Themes can also be customized to modify visuals of particular named widgets. This ties the theme file to the Glade file more tightly. Some of the sample screen skins allow the user to select any of the themes on the system. The sample *gscreen* is an example. Some will load the theme that is the same name in the config file. The sample *gscreen-gaxis* is an example. This is done by putting the theme folder in the config folder that has the INI and HAL files and naming it: SCREENNAME_theme (SCREENNAME being the base name of the files eg. gaxis_theme). Inside this folder is another folder call gtk-2.0, inside that is the theme files. If you add this file, Gscreen

will default to this theme on start up. gscreen-gaxis has a sample custom theme that looks for certain named widgets and changes the visual behavior of those specific widgets. The Estop and machine-on buttons use different colors then the rest of the buttons so that they stand out. This is done in the handler file by giving them specific names an by adding specific commands in the theme's gtkrc file. For some info on GTK theming (the sample theme uses the pixmap theme engine), see: GTK Themes, Pixmap Theme Engine.

#### 10.4.2.2 Build a GladeVCP Panel

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.

**Установленный 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 compiled commands** 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.

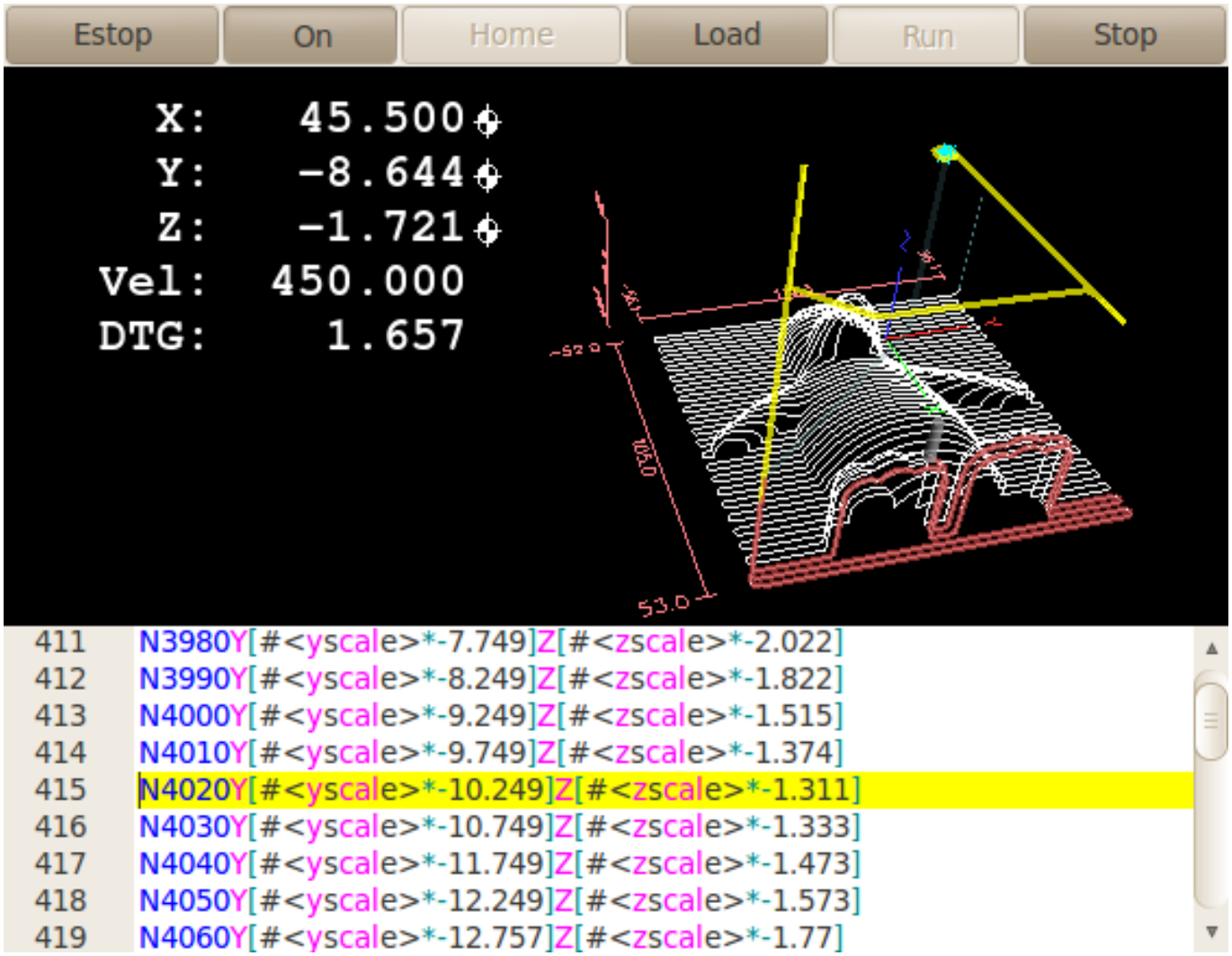
Ok you have loaded the stock Glade file and now can edit it. The first thing you notice is it does not look in the editor like what it is displayed like Gscreen uses some tricks, such as hiding all boxes of buttons except one and changing that one depending on the mode. The same goes for notebooks, some 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.

Remember, if you use a custom screen option YOU are responsible for fixing it (if required) when updating LinuxCNC.

### 10.4.3 Building a simple clean-sheet custom screen

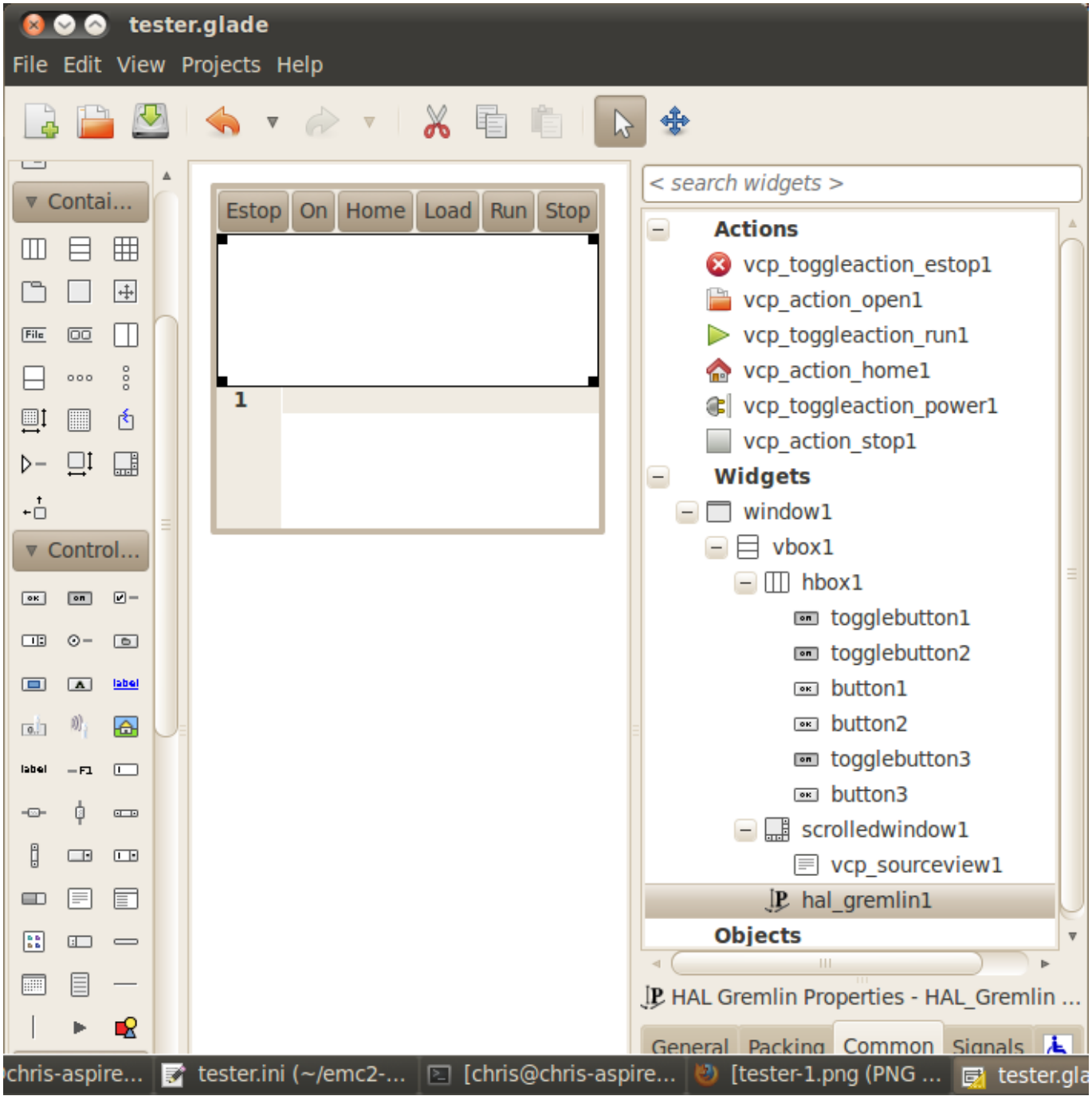



Lets build a simple usable screen. Build this in the Glade editor (if using a RIP package run it from a terminal after using . scripts/rip-environment ).

Things to note:

- The top level window must be called the default name, *window1* - Gscreen relies on this.
- 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.
- After adding the actions we must link the buttons to the actions for them to work (see below).
- The gremlin widget doesn't have a default size so setting a requested size is helpful (see below).
- 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.)
- The buttons will expand as the window is made larger which is ugly so we will set the box they are in, to not expand (see below).
- The button types to use depend on the VCP_action used -eg vcp_toggle_action usually require toggle buttons (Follow the example for now).

- The buttons in this example are regular buttons not HAL buttons. We don't need the HAL pins.




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.

Select the gremlin widget click the common tab and set the requested height to 100 and click the checkbox beside it.

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 Handler file example

There are special functions Gscreen checks the handler file for. If you add these in you handler file Gscreen will call them instead of gscreen's internal same-named functions.

- initialize_preferences(self): You can install new preference routines.
- 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): makes / initializes 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): allows a complete revamp of the manual toolchange system.
- 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): used to check what mode the screen is in. Returns a list[] 0 -manual 1- mdi 2- auto 3- jog.
- on_tool_change(self,widget): You can use this to override the manual tool change dialog -this is called when *gscreen.tool-change* changes state.
- 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.

You can also add you own functions to be called in this file. Usually you would add a signal to a widget to call your function.

#### 10.4.4.1 Adding Keybindings Functions

Our tester example would be more useful if it responded to keyboard commands. There is a function called keybindings() that tries to set this up. While you can override it completely, we didn't - but it assumes some things:

- It assumes the estop toggle button is call *button_estop* and that F1 key controls it.
- It assumes the power button is called *button_machine_on* and the F2 key controls it.

These are easily fixed by renaming the buttons in the Glade editor to match. But instead we are going to override the standard calls and add our own.

Add these command to the handler file:

```
# override Gscreen Functions
# keybinding calls
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
```

So now we have overridden Gscreen's function calls of the same name and deal with them in our handler file We now reference the widgets by the name we used in the Glade editor. We also added a built in gscreen function to make a sound when Estop changes. Note that we we call Gscreen's built in functions we must use self.gscreen.[FUNCTION NAME]() If we used self.[FUNCTION NAME]() it would call the function in our handler file.

Lets add another key binding that loads halmeter when F4 is pressed.

In the handler file under *def initialize_widgets(self):* change to:

```
  def initialize_widgets(self):
    self.gscreen.init_show_windows()
    self.gscreen.keylookup.add_conversion('F4','TEST','on_keycall_HALMETER')
```

Then add these functions under the *HandlerClass* class:

```
  def on_keycall_HALMETER(self,state,SHIFT,CNTRL,ALT):
    if state:
      self.gscreen.on_halmeter()
      return True
```

This adds a keybinding conversion that directs gscreen to call on_keycall_HALMETER when F4 is pressed. Then we add the function to the handle file to call a Gscreen builtin function to start halmeter.

#### 10.4.4.2 Linuxcnc State Status

The module *Gstat* polls LinuxCNC's state every 100ms and sends callback messages to user functions when state changes. You can register messages to act on specific state changes. As an example we will register to get *file-loaded* messages when LinuxCNC loads a new file. First we must import the module and instantiate it: In the import section of the handler file add:

```
from hal_glib import GStat
GSTAT = GStat()
```

In the handler file under *def __init__(self):* add:

```
GSTAT.connect('file-loaded', self.update_filepath)
```

Then in the *HandlerClass*, add the function:

```
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 Keys

There are no special widgets to do screen-button jogging, so we must do it with Python code. Under the connect_signals function add this code:

```
        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)
```

Add these functions under the HandlerClass class:

```
    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)
```

Finally add two buttons to the GLADE file for each axis - one for positive, one for negative direction jogging. Name these buttons xneg, xpos, yneg, ypos zneg, zpos respectively. add a SpeedControl widget to the GLADE file and name it jog_speed.

### 10.4.5 Gscreen Start Up

Gscreen is really just infrastructure to load a custom GladeVCP file and interact with it.

1. Gscreen reads the options it was started with.
2. Gscreen sets the debug mode and set the optional skin name.
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. Optional Audio is initialized if available.
6. It reads some of the INI file to initialize the units, and the number/type of axes.
7. Initializes Python's binding to HAL to build a non-realtime component with the Gscreen name.
8. GladeVCP's makepins is called to parse the XML file to build HAL pins for the HAL widgets and register the LinuxCNC connected widgets.
9. Checks for a *local* handler file in the configuration folder or else uses the stock one from the skin folder.
10. If there is a handler file gscreen parses it, and registers the function calls into Gscreen's namespace.
11. Glade matches/registers all signal calls to functions in gscreen and the handler file.
12. Gscreen checks the INI file for an option preference file name otherwise it uses *.gscreen_preferences* =.
13. Gscreen checks to see if there is a preference function call (*initialize_preferences(self)*) in the handler file otherwise it uses the stock Gscreen one.
14. Gscreen checks for ClassicLadder realtime component.
15. Gscreen checks for the system wide GTK theme.
16. Gscreen collects the jogging increments from the INI file.
17. Gscreen collects the angular jogging increments from the INI file.
18. Gscreen collects the default and max jog rate from the INI.
19. Gscreen collects the max velocity of any axes from the INI's TRAJ section.
20. Gscreen checks to see if there is angular axes then collects the default and max velocity from the INI file.
21. Gscreen collect all the override setting from the INI.
22. Gscreen checks if its a lathe configuration from the INI file.
23. Gscreen finds the name of the tool_table,tool editor and param file from the INI.
24. Gscreen checks the handler file for keybindings function (*initialize_keybindings(self)*) or else use Gscreen stock one.
25. Gscreen checks the handler file for pins function (*initialize_pins(self)*) or else use Gscreen stock one.
26. Gscreen checks the handler file for manual_toolchange function (*initialize_manual_toolchange(self)*) or else use Gscreen stock one.
27. Gscreen checks the handler file for connect_signals function (*initialize_connect_signals(self)*) or else use Gscreen stock one.
28. Gscreen checka the handler file for widgets function (*initialize_widgets(self)*) or else use Gscreen stock one.
29. Gscreen set up messages specified in the INI file.
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 sets the display cycle time based on the INI file.
32. Gscreen checks the handler file for *timer_interupt(self)* function call otherwise use Gscreen's default function call.

### 10.4.6 INI Settings

Under the [DISPLAY] heading:

```
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 User Dialog Messages

This function is used to display pop up dialog messages on the screen. These are defined in the INI file and controlled by HAL pins:

**MESSAGE_BOLDTEXT**
: is generally a title.

**MESSAGE_TEXT**
: is below that and usually longer.

**MESSAGE_DETAILS**
: is hidden unless clicked on.

**MESSAGE_PINNAME**
: is the basename of the HAL pins.

**MESSAGE_TYPE**
: specifies whether its a yes/no, ok, or status message

- Status messages
  - will be shown in the status bar and the notify dialog,
  - require no user intervention.
- ok messages
  - require the user to click ok to close the dialog.
  - have one HAL pin to launch the dialog and one to signify it is waiting for response.
- yes/no messages
  - require the user to select yes or no buttons to close the dialog.
  - have three HAL pins:
    1. one to show the dialog,
    2. one for waiting, and
    3. one for the answer.

Here is a sample INI code. It would be under the [DISPLAY] heading.

```
# This just shows in the status bar and desktop notify popup.
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

```
# This will pop up a dialog that asks a yes no question
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest

# This pops up a dialog that requires an ok response and it shows in the status bar and
# the desktop notify 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 Copy the Stock Handler/Glade File For Modification

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.

Here is a sample:

```
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']
```

The line:

```
**** GSCREEN INFO: handler file path: ['/home/chris/emc-dev/share/gscreen/skins/industrial/ ↩
    industrial_handler.py']
```

shows where the stock file lives. Copy this file to your config folder. This works the same for the Glade file.

# 10.5 QtDragon GUI

## 10.5.1 Введение

QtDragon и QtDragon_hd созданы с использованием инфраструктуры QtVCP. Это творческое видение форумной личности Persei8. Большая часть этого основана на превосходной работе других членов сообщества LinuxCNC. Версия LinuxCNC адаптирована из версий Persei8 на Github. В первую очередь он предназначен для станков с 3-5 осями, таких как фрезерные станки. Он хорошо работает с сенсорным экраном и/или мышью. QtDragon поддерживает несколько способов подключения инструментов и проверки заготовок. Вы можете использовать возможности внешних смещений LinuxCNC для автоматического подъема шпинделя во время паузы. Если вы используете опцию VersaProbe и переназначаете код, вы можете добавить автоматическое измерение длины инструмента при смене инструмента.

**Note**
QtDragon и QtVCP — относительно новые программы, добавленные в LinuxCNC. Возможны баги и странности. Пожалуйста, проводите тщательную проверку при использовании опасного станка. Пожалуйста, направляйте отчеты на форум или в список рассылки.

### 10.5.1.1 QtDragon

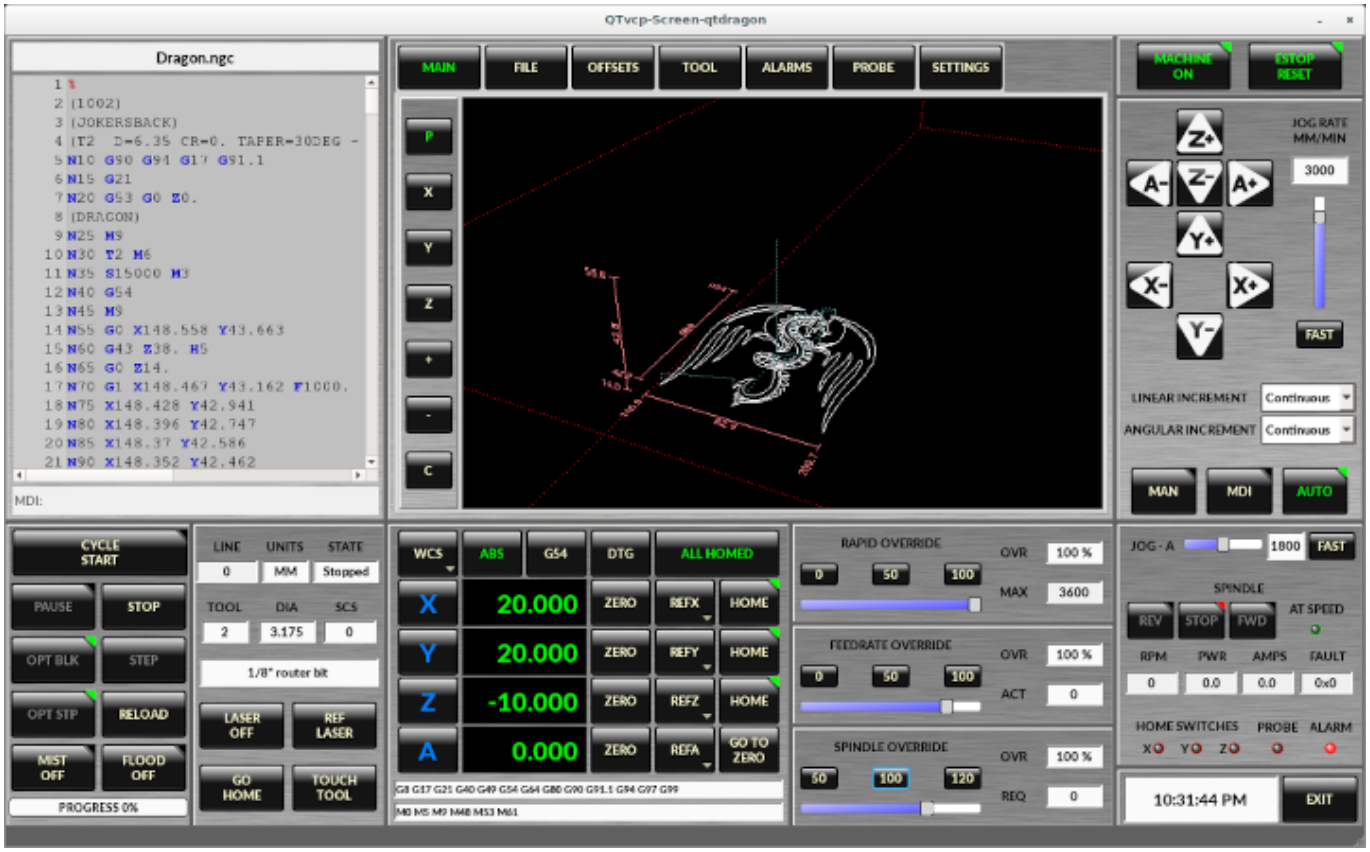
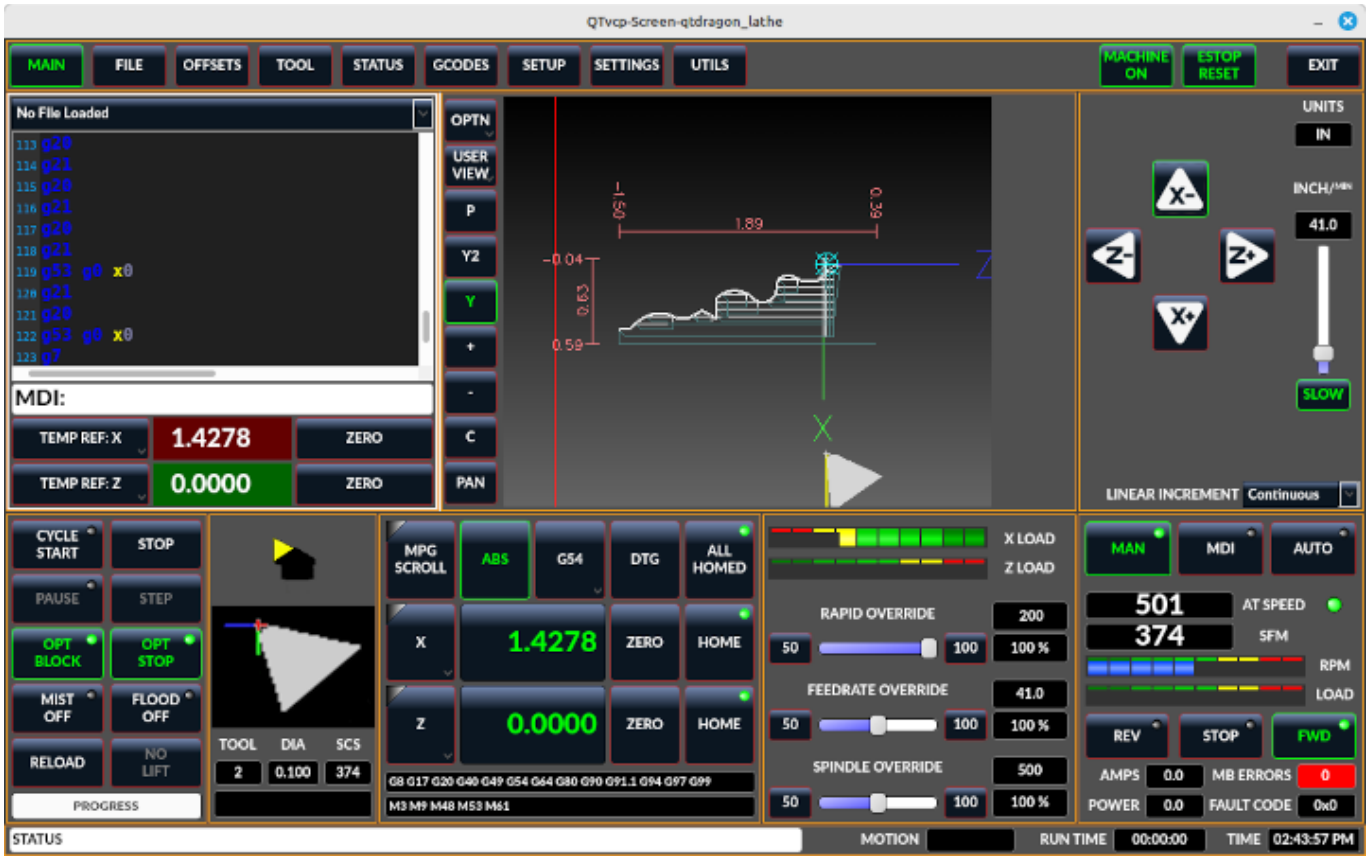



Figure 10.27: QtDragon — образец от 3 до 5 осей (1440x860) в серебристой теме

QtDragon is resizable from a resolution of 1280x768 to 1680x1200. It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

### 10.5.1.2 QtDragon_lathe




QtDragon_lathe is a modified version of QtDragon more suitable for lathes.
It is resizable from a resolution of 1280x768 to 1680x1200.
It will work in window mode on any monitor with higher resolution but not on monitors with lower resolution.

#### 10.5.1.3 QtDragon_hd

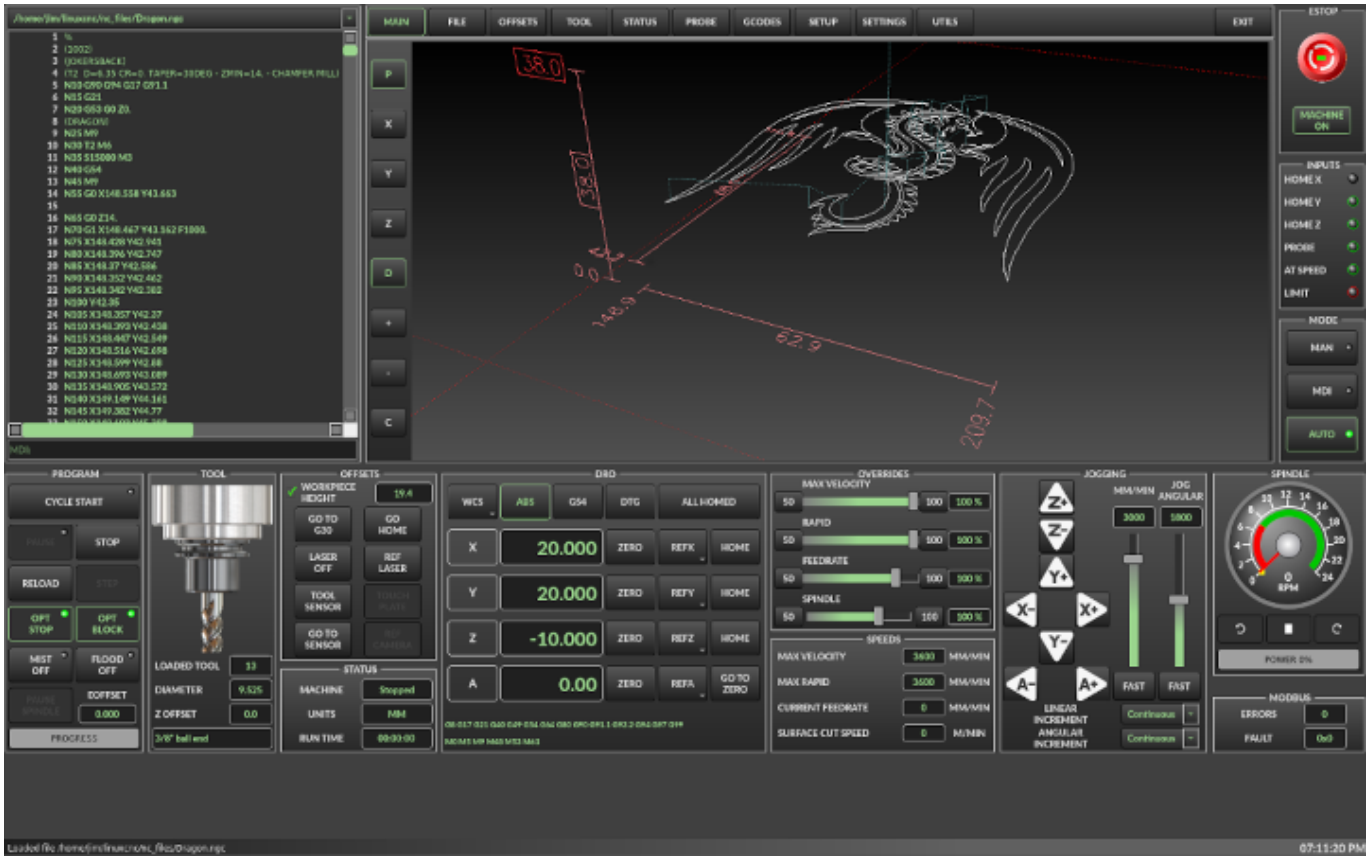



Figure 10.28: QtDragon_hd — пример от 3 до 5 осей для больших мониторов (1920x1056) в темной теме

QtDragon_hd is a similar design as QtDragon but modified to utilize the extra space of modern larger monitors. There are some small differences in layout and utility.

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.

#### 10.5.1.4 QtDragon_hd_vertical

QtDragon_hd_vertical is a vertical orientated version. It is not resizable.

### 10.5.2 Начало работы — INI-файл

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.

---

**Note**
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 Display

In the section [DISPLAY] change the DISPLAY = assignment to read:

- qtdragon for a small version
- qtdradon_hd for the large 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 Preferences

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 Ведение журнала

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 controls

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 Spindle controls

Spindle control settings (in rpm and watts):

```
[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 increments

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 Grid Increments

Set the available optional grid sizes for graphics display.
This will override the default sizes.
*mm* and *in* are used to specify units.
The grid selection box is shown when clicking the *OPTN* button on the graphics display.

```
[DISPLAY]
GRIDS = 0, .1 mm, 1 mm, 2 mm, 5 mm, 10 mm, .25 in, .5 in
```

#### 10.5.2.8 Jog speed

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
```

#### 10.5.2.9 User message dialog system

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 = This is the short text
MESSAGE_TEXT = This is the longer text of the both type test. It can be longer than the ↩
    status bar text
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = oktest
```

Multimessages use an s32 HAL pin to pop multiple defined messages.

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 1
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 1
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 1
```

```
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 2
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 2
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 2
MULTIMESSAGE_VFD_ICON = INFO
```

#### 10.5.2.10 Embed Custom VCP Panels

You can optionally embed QtVCP Virtual Control Panels into the QtDragon screens.
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`
and `WINDOW`.
Tab `EMBED_TAB_COMMAND` specifies what embed-able program to run, including any of its command line options.

If using the `tabWidget_utilities` or `tabWidget_setup` locations, an extra tab will appear with the panel.
If using `stackedWidget_mainTab`, a button labelled *User* will appear.
Pressing this button will cycle through displaying all available panels (specified for this location) on the main tab area.
You also use `WINDOW` to create a pop up window that cab be shown/hidden with an arrow button that will appear near the machine on button

**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
```

#### 10.5.2.11 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
```

#### 10.5.2.12 Управление предварительным просмотром

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)
```

#### 10.5.2.13 Program Extensions/Filters

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
```

#### 10.5.2.14 Probe/Touchplate/Laser Settings

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
```

#### 10.5.2.15 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
```

#### 10.5.2.16 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
```

#### 10.5.2.17 Кнопки макроса

QtDragon has up to ten convenience buttons for calling *macro actions*.
These are under the heading *[MDI_COMMAND_LIST]* as *MDI_COMMAND_MACRO0* = to *MDI_COMMAND_*
=
These could also call OWord routines if desired.
In the sample configurations they are labelled for moving between current user system origin (zero point) and Machine system origin.
User origin is the first MDI command in the INI list, machine origin is the second.
This example shows how to move Z axis up first. Commands separated by the *;* are run one after another.
The button label text can be set with any text after a comma, the \n symbol adds a line break.

```
[MDI_COMMAND_LIST]
# for macro buttons
MDI_COMMAND_MACRO0 = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND_MACRO1 = G53 G0 Z0;G53 G0 X0 Y0,Goto\nMachn\nZero
```

#### 10.5.2.18 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
```

#### 10.5.2.19 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. Adjust these to match your actual requirements.**

```
[HAL]
POSTGUI_HALCMD = loadusr qtvcp test_probe
POSTGUI_HALCMD = loadusr qtvcp test_led
POSTGUI_HALCMD = loadusr halmeter
```

#### 10.5.2.20 HAL Bridge

Hal Bridge is similar to HALUI - it has HAL pins that communicate with QtDragon.
These pins could be used with HALUI to built a more friendly control panel.

- It can report/change the current selected Axis button.
- The jog rate/increments will be reported.
- There is a cycle start and pause pin - these call the code in QtDragon rather then the motion controller.
- If there are macros defined in the INI, there will be pins available to initiate them.

**Sample entry. Remove -d to quiet debugging messages.**

```
[HAL]
HALBRIDGE= hal_bridge -d
```

**Typical HAL pins avaialble:**

```
Component Pins:
Owner   Type  Dir                Value  Name
    38  s32   OUT                    0  base-thread.time
    29  bit   OUT                FALSE  bridge.axis-x-is-selected
    29  bit   IN                 FALSE  bridge.axis-x-select
```

```
    29  bit   OUT                FALSE  bridge.axis-y-is-selected
    29  bit   IN                 FALSE  bridge.axis-y-select
    29  bit   OUT                FALSE  bridge.axis-z-is-selected
    29  bit   IN                 FALSE  bridge.axis-z-select
    29  bit   IN                 FALSE  bridge.cycle-pause-in
    29  bit   IN                 FALSE  bridge.cycle-start-in
    29  float OUT                    0  bridge.jog-increment
    29  float OUT                    0  bridge.jog-increment-angular
    29  float OUT                   15  bridge.jog-rate
    29  float OUT                  360  bridge.jog-rate-angular
    29  s32   OUT                   -1  bridge.joint-selected
    29  bit   IN                 FALSE  bridge.macro-cmd-MACRO0
    29  bit   IN                 FALSE  bridge.macro-cmd-MACRO1
```

#### 10.5.2.21 Builtin Sample Configurations

The sample configurations `sim/qtdragon/` or `sim/qtdragon_hd` are already configured to use Qt-Dragon as the screen. There are several examples that demonstrate various machine configurations.

### 10.5.3 Key Bindings

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 on/off
F2 - Machine on/off
F12 - Style Editor
Home - Home All Joint of the Machine
Escape - Abort Movement
Pause - Pause Machine Movement
```

### 10.5.4 Кнопки

Buttons that are checkable will change their text colour when checked. This is controlled by the stylesheet/theme

### 10.5.5 Virtual Keyboard

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:

- press the *HIDE* button on the virtual keyboard.
- click the MAIN page button
- go into AUTO mode

It should be noted that keyboard jogging is disabled when using the virtual keyboard.

### 10.5.6 HAL Контакты

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
```

This bit output pin can be connected to turn on a laser:

```
qtdragon.btn-laser-on
```

This float output pin indicates the camera rotation in degrees:

```
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
qtdragon.eoffset-is-active
```

These float output pins reflect the current slider jograte (in machine units):

```
qtdragon.slider-jogspeed-linear
qtdragon.slider-jogspeed-angular
```

These float output pins reflect the current slider override rates:

```
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
```

This pin will be true when the loaded tool equals the number set in the Versa Probe tool number in the preference file.
It can be used (for example) to inhibit the spindle when the probe is loaded by connecting it to `spindle.0.inhibit`.

```
qtversaprobe.probe-loaded
```

This output pin is available when setting the Basic Probe INI option.
This pin will be true when the loaded tool equals the number set in the Basic Probe tool number edit box.
It can be used (for example) to inhibit the spindle when the probe is loaded by connecting it to `spindle.0.inhibit`.

```
qtbasicprobe.probe-loaded
```

This input pin is available to toggle pause/resume of a running program.

```
qtdragon.external-pause
```

You can externally operate dialog responses on most qtdragon dialogs.

```
qtdragon.dialog-ok
qtdragon.dialog-no
qtdragon.dialog-cancel
```

### 10.5.7 HAL files

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 Manual Tool Changes

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_manualto*
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 Spindle

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 Auto Raise Z Axis on Program Pause

QtDragon can be set up to automatically raise and lower the Z axis and stop the spindle, when the program is paused.
You toggle the *SPINDLE LIFT* or *NO LIFT* button to select the option to raise the spindle in Z when paused.
Then when you press the *PAUSE* button the spindle will lift the amount set on the *Settings* tab and the spindle will stop.
Pressing *RESUME* will start the spindle and lower the spindle.
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 speed.
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).

---

**Note**
If using the Spindle lift option, HALUI can not be used to pause/resume the program. There is a pin, *QtDragon.external-pause* available to pause/resume from an external source. You must also enable external offsets. In the setting tab check *use external offsets* If you wish to inhibit the spindle when a probe tool is loaded, you will need to use an logical `or`-component to combine the two spindle inhibit signals to connect to `spindle.0.inhibit`.

---

This optional behaviour requires additions to the INI and the QtDragon_postgui HAL file.

In the INI, under the AXIS_Z heading.

```
[AXIS_Z]
OFFSET_AV_RATIO  = 0.2
```

This reserves 20% of max velocity and max acceleration for the external offsets.
This will limit max velocity of the machine by 20%

In the qtdragon_postgui.hal file add:

```
# 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
net eoffset-state    qtdragon.eoffset-is-active      <=  motion.eoffset-active

# Inhibit spindle when paused
net spindle-pause    qtdragon.spindle-inhibit        =>  spindle.0.inhibit

# uncomment for dragon_hd
#net limited          qtdragon.led-limits-tripped     <= motion.eoffset-limited

setp axis.z.eoffset-enable 1
setp axis.z.eoffset-scale 1.0
```

### 10.5.11 Z level compensation

QtDragon_hd can be set up to probe and compensate for Z level height changes by utilizing the external program *G-code Ripper*.

**Note**
This is only available in the QtDragon_hd version.

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 Using G-code Ripper for Z level Compensation

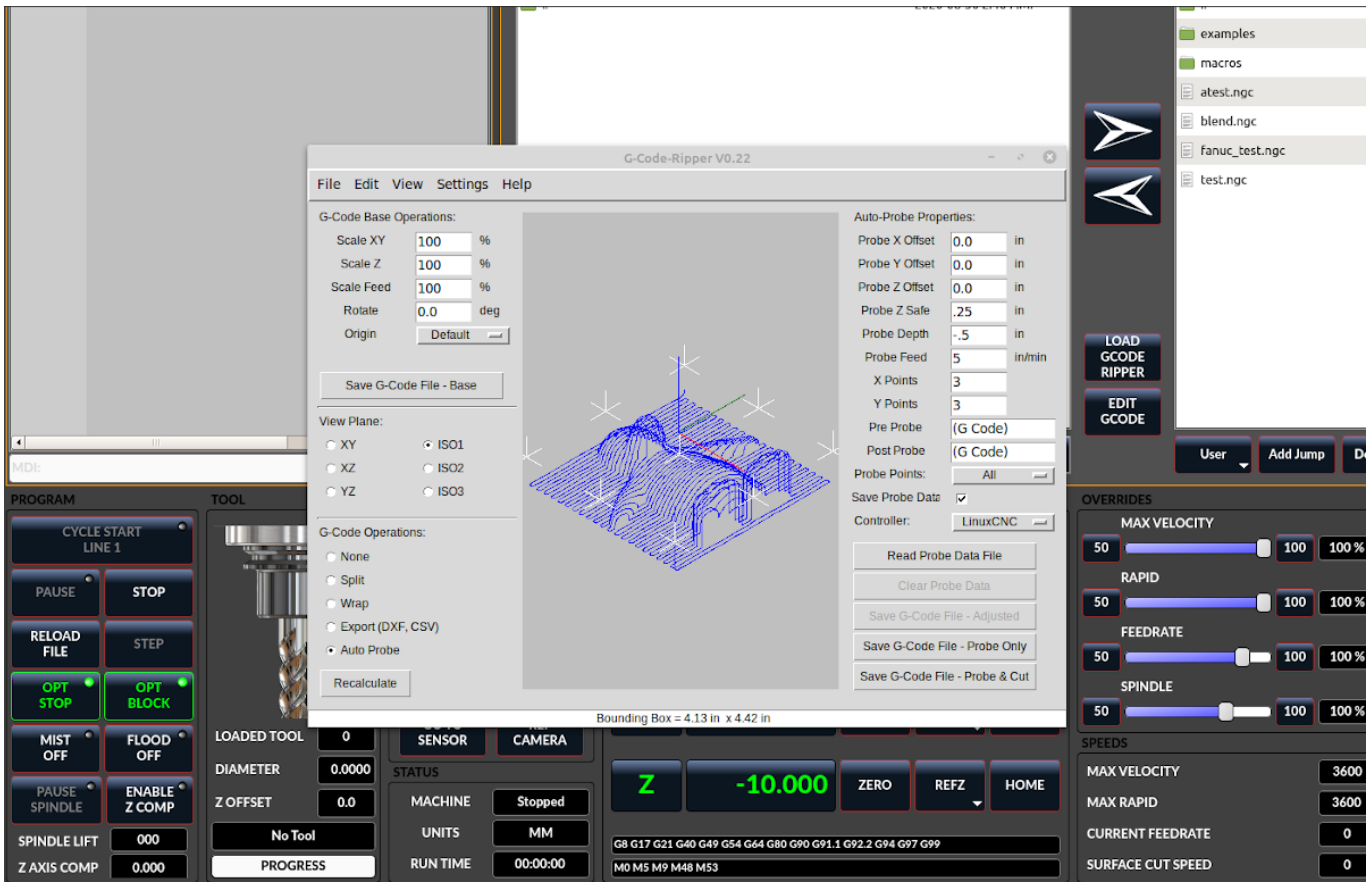



Figure 10.29: QtDragon_hd showing G-code Ripper

**Note**
G-code Ripper offers many functions that we will not go in to here. This is only available in the QtDragon_hd version.

- 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.
- File -> Open G-Code File (The file you will run after compensation)
- 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.

---

**Note**
If you use auto raise Z to lift the spindle on pause, you must combine the two with a HAL component and feed that to LinuxCNC's motion component.

---

### Sample postgui HAL file for combined spindle raise and Z Level compensation

```
# load components
#############################################################################

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

# connect signals to LinuxCNC's motion component
#############################################################################

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
net eoffset-state              qtdragon.eoffset-is-active      <=  motion.eoffset-active

## 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 Probing

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.

**Note**
Probing is very unforgiving to mistakes; be sure to check settings before using.

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

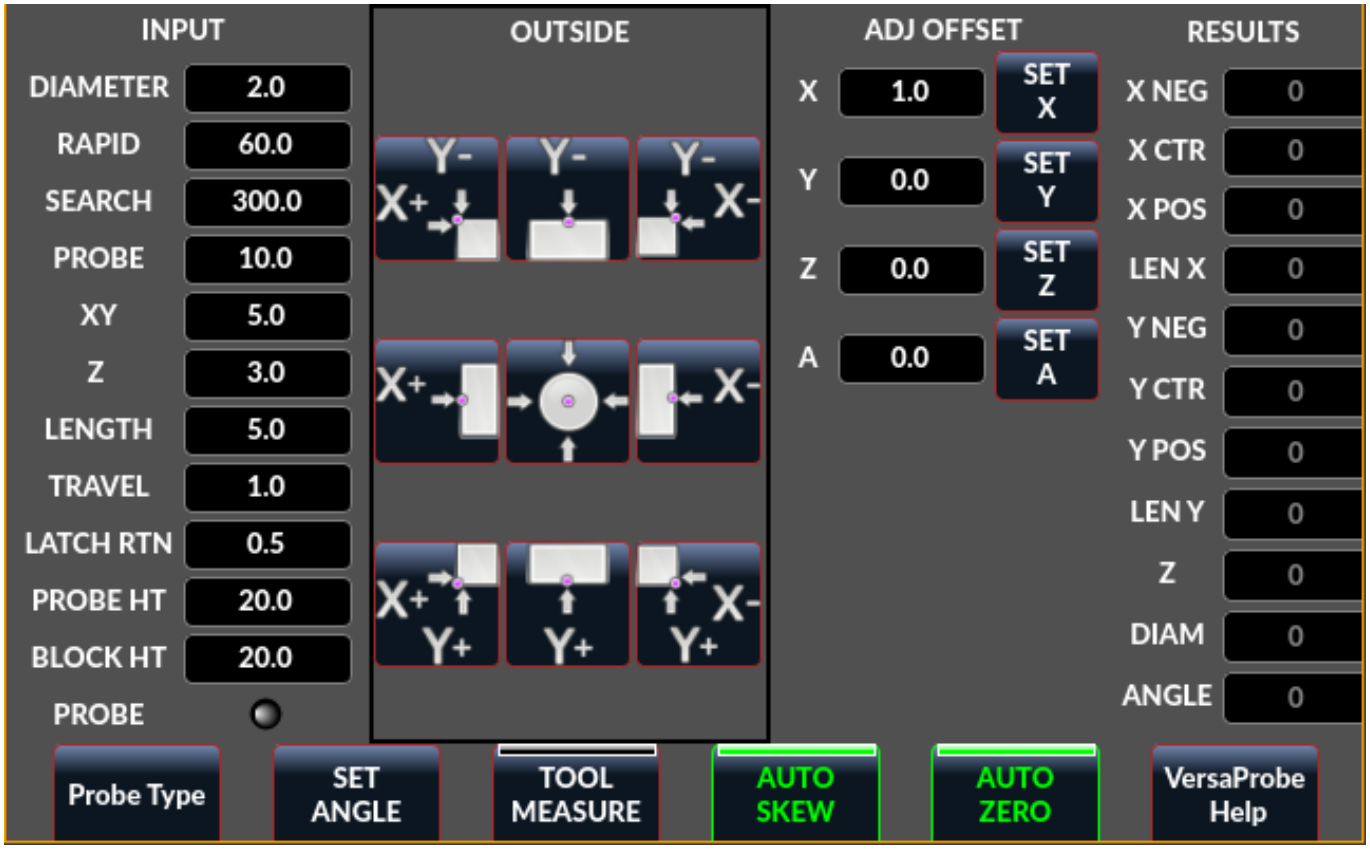



Figure 10.30: QtDragon - Versa Probe Option

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.

**’Инструмент измерение ’**
This (if integrated) turns auto tool probing on and off.

Versaprobe offers 5 output pins for tool measurement purpose and one that can be used to inhibit the spindle when the probe is loaded.
The 5 pins are used to be read from a remap G-code subroutine, so the code can react to different values.
Currently the probe tool number is only editable in the preference file:

```
[VERSA_PROBE_OPTIONS]
ps_probe_tool = 1
```

**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)**
**gmoccapy.searchvel** *(float OUT)* - Скорость поиска переключения датчика инструмента

**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.

**qtversaprobe.probe-loaded (HAL_BIT)**
Отразить, соответствует ли текущий инструмент номеру зонда файла настроек.

### 10.5.12.2 Basic probe

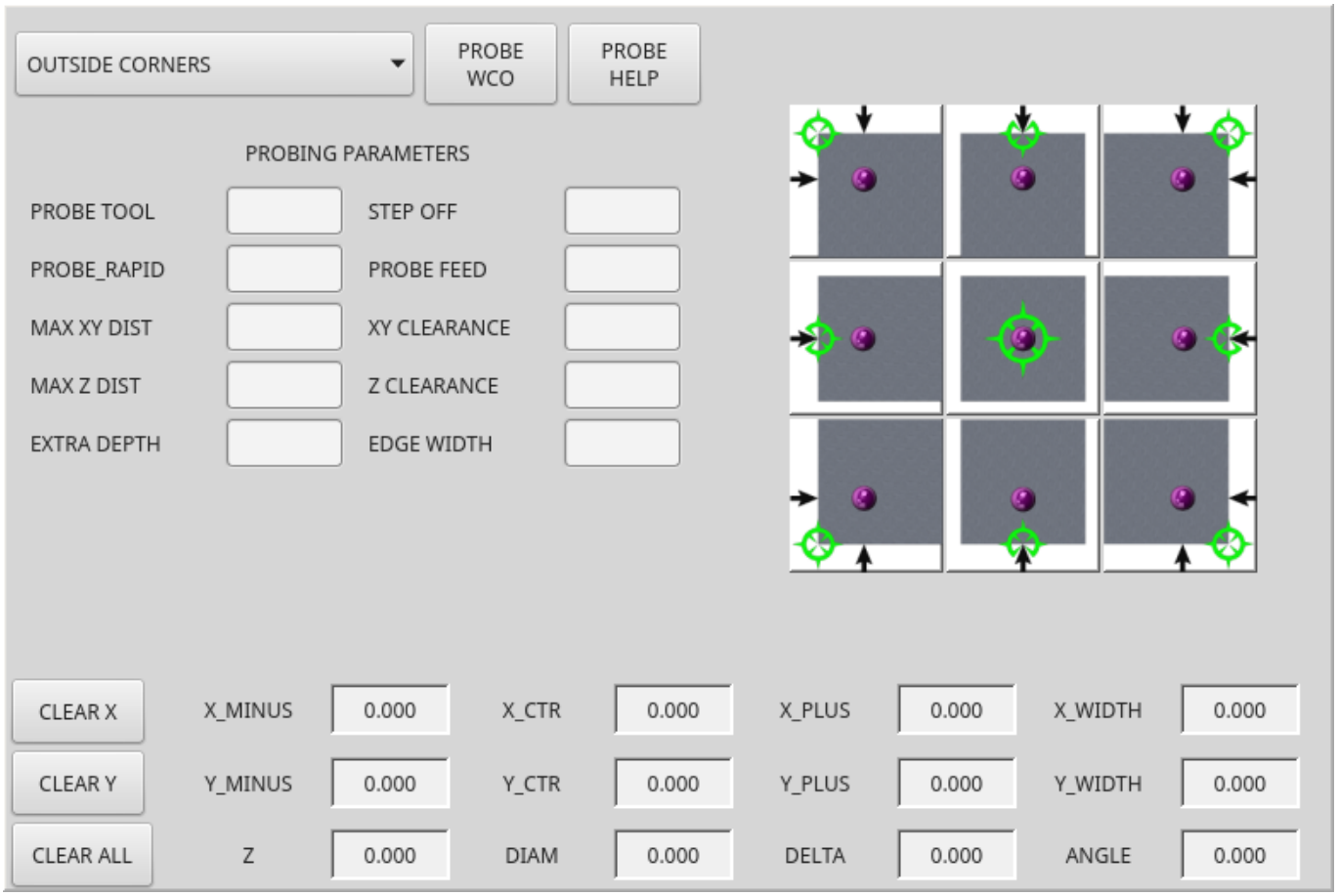



Figure 10.31: QtDragon - Basic Probe Option

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

***Диаметр зонда***
: Размер кончика щупа

***Probe Rapid***
: Скорость быстрых перемещений в единицах станка

***Probe Search***
: Скорость первого *грубого* поиска в единицах измерения станка

***Probe Feed***
: Скорость второго *точного* поиска в единицах измерения станка

***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 зазор***
: Clearance distance from probed to top of material

***Дополнительная глубина***
: 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

***Подсказка диаметра***
: Used by Round Boss or Round Pocket probing (start move: 1/2 diameter plus XY clearance)

***X Подсказка***
: Used by Rectangular Boss/Pocket probing (start move: 1/2 X length plus XY clearance)

***Y Подсказка***
: 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.

**Note**
Pressing the stop button or the keyboard escape key, will abort the probing.

This can be used to inhibit the spindle when the probe is loaded.
You would connect it to spindle.0.inhibit

```
qtbasicprobe.probe-loaded
```

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.

**Note**
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.

#### 10.5.12.3 Customizing Probe Screen Widget

It is possible to load a customized version of the probe widget.

There should be a folder in the config folder; for screens: named <CONFIG FOLDER>/qtvcp/.
There may be (or can be added) a folder *lib/* and *widgets/*
In the widgets folder you can copy *basic_probe.py* (or *versa_probe.py*) and *probe_subprog.py*
In the lib folder copy *touchoff_subprogram.py*
If these files are found they will be used instead of the originals.
You can modify the files to change behaviour.

### 10.5.13 Touch plate

Figure 10.32: QtDragon - 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.

**Note**
When using a conductive plate the search and probe velocity should be the same and slow. If using a tool setter that has spring loaded travel then you can set search velocity faster. LinuxCNC ramps speed down at the maximum acceleration rate, so there can be travel after the probe trip if the speed is set to high.

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 Автоматическое измерение инструмента

### 10.5.14.1 Обзор

QtDragon can be setup to do integrated auto tool measurement using the Versa Probe widget and remap code.

This feature assumes the use of two probes in concert:

1. A tool switch sensor, fixed somewhere on the machine (sometimes called a tool-setter), and
2. a spindle probe that is installed temporarily at the beginning of the job (sometimes called an xyz probe or a Renishaw probe).

These techniques are useful for machines that do not have repeatable tool holders and do not have automatic tool changing devices. (For machines with repeatable tool holders, see the section on measuring tool length. For machines with automatic tool changing devices, consult work done under the LinuxCNC repository at `configs/sim/axis/remap/rack-toolchange`.)

Чтобы использовать эту функцию, вам нужно будет сделать некоторые дополнительные настройки, и вы можете захотеть использовать предлагаемые контакты HAL для получения значений в вашей собственной процедуре переназначения ngc. Эти настройки рассматриваются далее в разделе.

First, this document covers how to use this technique. Second, this document covers how to set up for this technique at the beginning of a production run.

### 10.5.14.2 Обзор рабочего процесса

A detailed workflow walkthrough follows this overview.

Шаги настройки включают в себя:

- Entering the probe velocities on the versa probe settings page.
- Enabling ”Use Tool Measurement” on the Versa Probe tab.
- Enabling ”Use Tool Sensor” under Settings.

**Important**
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.

Tool Measurement in QtDragon is organized into the following steps which will be explained in more detail in the following section:

1. Zero the probe tool by measuring the tool setter with the spindle probe installed
2. Коснитесь заготовки по X и Y.
3. 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.).
4. In the Versa probe tab, enter the measured value for block height.
5. Перейдите в автоматический режим и запустите программу.

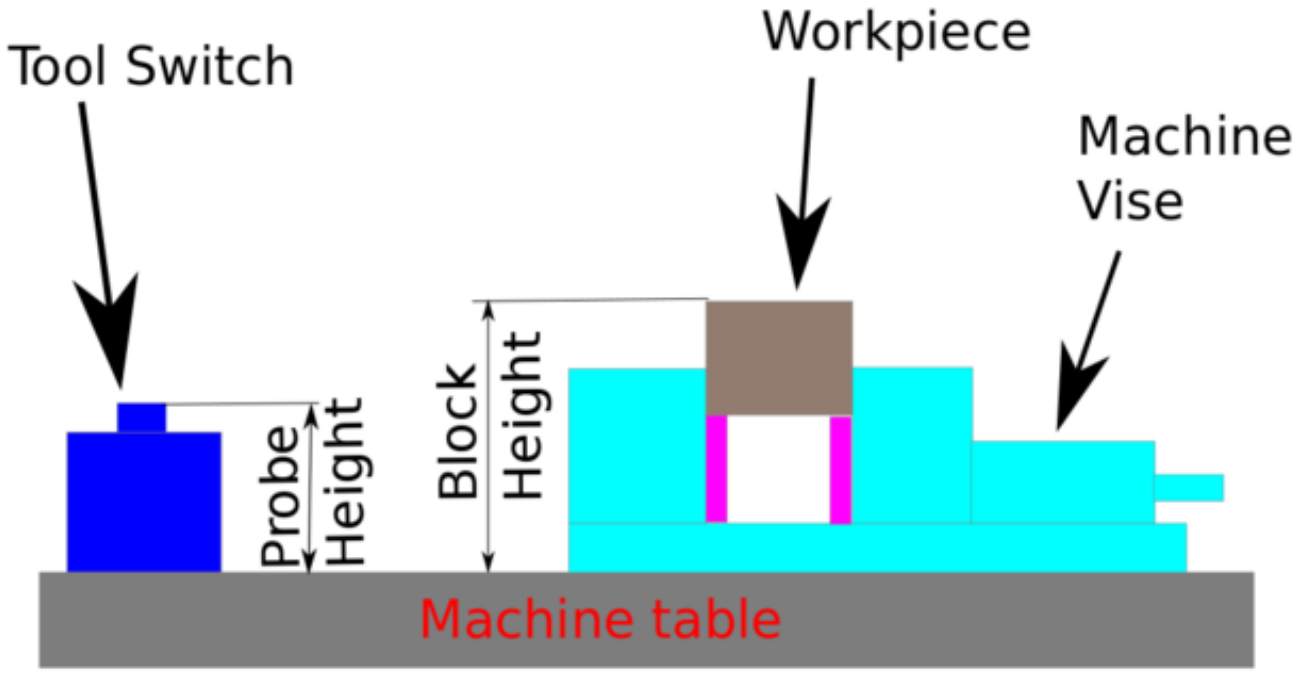



Figure 10.33: Auto tool measurement

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 this way is, that you do not need a reference tool.

**Note**
Ваша программа должна содержать смену инструмента в начале. Инструмент будет измерен, даже если он уже использовался ранее, поэтому нет никакой опасности, если высота блока изменилась. На you tube есть несколько видеороликов, демонстрирующих технику с использованием GMOCCAPY. Экран GMOCCAPY был пионером этой техники.

The sequence of events (using the default files in the default setting):

1. Rapid move in Z to position defined in the INI’s [TOOL_CHANGE] Z.
2. Rapid move in X and Y to number defined in INI’s [TOOL_CHANGE] X and Y.
3. Perform normal M6 tool change, i.e., stop spindle, send message to user to change the tool.
4. Rapid move in X and Y to position defined in the INI’s [VERSA_TOOLSETTER] X and Y.
5. Rapid move down in Z to position defined in the INI’s [VERSA_TOOLSETTER] Z.
6. Probe down in Z to maximum defined in the INI’s [VERSA_TOOLSETTER] MAXPROBE.
7. Changes the offset of the current work coordinate system to match the difference between the previous tool and the currently measured tool.
8. Rapid move up in Z to position defined in the INI’s [VERSA_TOOLSETTER] Z_MAX_CLEAR.
9. Rapid move to the X Y position when the tool change was called.
10. Rapid move down to the Z position when the tool change was called.

---

**Note**
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.

---

#### 10.5.14.3 Подробный пример рабочего процесса

The following setups need only be done once as long as they remain in effect:

1. Under Probe Tool Screens: Ensure reasonable values for ”Rapid” and ”Search,” these are the speeds at which the probing will be performed and are in machine units per minute.

2. Under Probe Tool Screens: Ensure that ”Tool Measure” is enabled (this is a button that must be highlighted)
3. Under Settings: Ensure that ”Use Tool Sensor” is enabled (this is a tick-box that must be checked)
4. In the Tool Table: Set up a tool for the spindle probe and ensure that its Z offset is set to zero.

---

**Note**
It is possible to use a non-zero tool length for the tool probe, but this requires extra steps and is easy to make mistakes. The following procedure assumes a zero tool probe length.

---

The following setup is done before beginning a program that has M6 tool change commands inside it.

1. Physically load the spindle probe into the spindle.
2. Logically load the spindle probe into the spindle with the M61 Qx command where x is the number in the tool table for the spindle probe (there is a button inside the tool table tab that can also be used)

3. Position to the Toolsetter: Use the button under the Probe Screens for ”Go To Toolsetter” to position the spindle above the Toolsetter.
4. Toolsetter Measure: Use the button under the Probe Screens for ”Probe Tool Setter Z Height.” Note that this will set and display on the Probe Settings screen the ”Probe HT” value in ABS coordinates
5. Двигайтесь к заготовке.
6. Workpiece Measure: Use the button under the Probe Screens for ”Probe Z Height of Material:” this will set and display on the Probe Settings screen the ”Block Ht” value in ABS coordinates. (Typically, this will now also be the zero Z for your Work Coordinate System)
7. Set Work Coordinate System (ie, G54, or other): Use the Probe Tool and whatever probe screen or other method is appropriate to set the X, Y, and Z coordinate system needed for your job.
8. If your program begins with a TnM6 command before spinning the spindle, you may leave the spindle probe installed. You may also issue a TnM6 command to change out the spindle probe, and if the program issues the same one, it will skip the tool change.

**Caution**
Take care not to leave the spindle probe in the spindle if a program may start the spindle.

Once those steps are complete, a program with multiple TnM6 toolchanges can be started and will operate with automatic pauses for manual tool change followed by automated tool measurement.

**Note**
After probing the new tool length using the tool-setter, this remap code uses a G43 which applies the offset to the Work Coordinate system which was in effect when the M6 command was issued. Because remapping has adjusted the Work Coordinate system by the offset between the previous and the current tool, the tool tip will end up at the same point in space as the tip of the previous tool was when the tool change was called.

#### 10.5.14.4 Work Piece Height Probing in QtDragon_hd

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.

### 10.5.14.5 Work Piece Height Probing

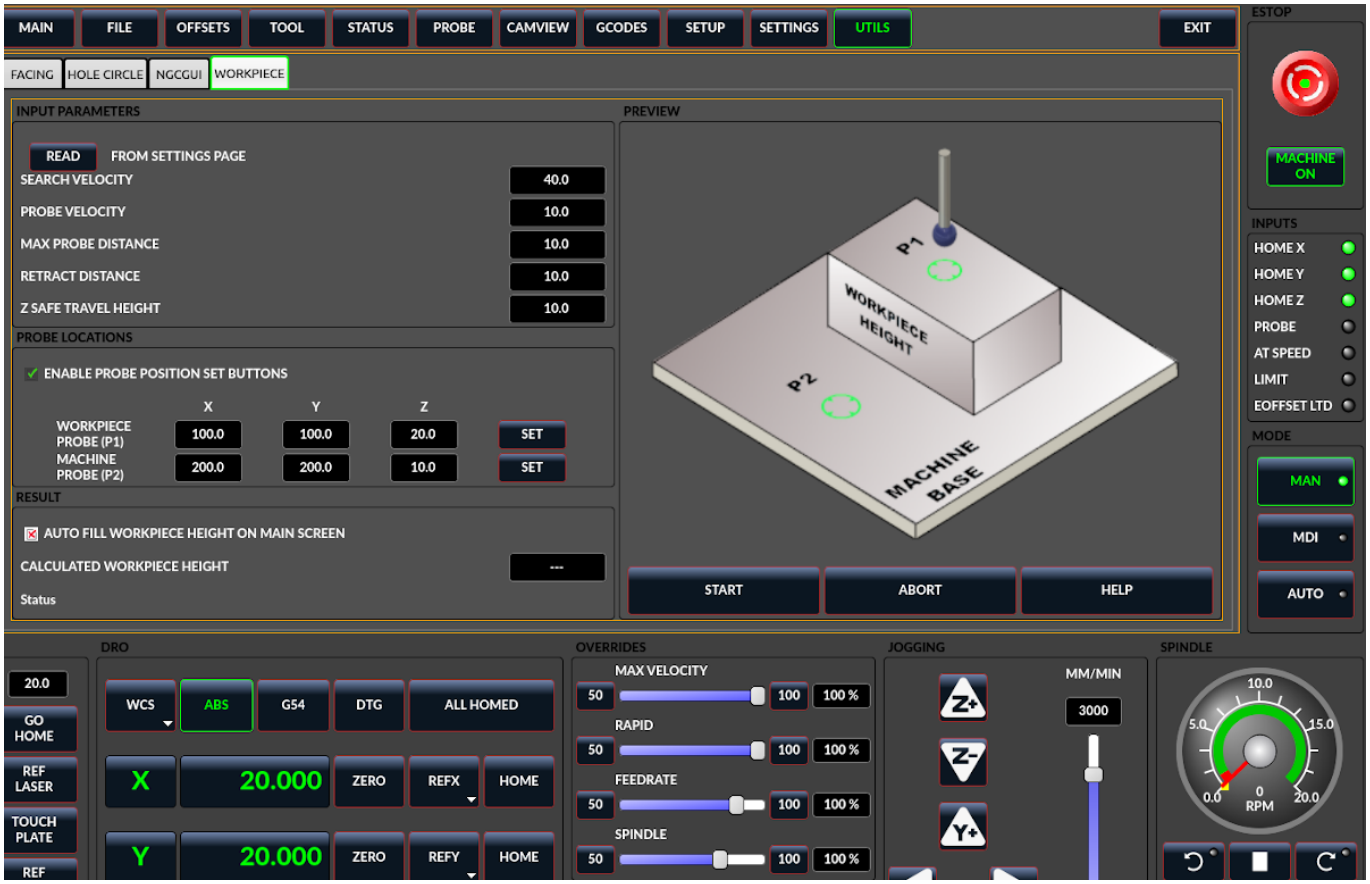



Figure 10.34: QtDragon_hd - Work piece Height probing

This program probes 2 user specified locations in the Z axis and calculates the difference in heights.

---

**Note**
This is only available in the QtDragon_hd version.

---

Enable Probe Position Set Buttons

- When checked, the SET buttons are enabled.
- This allows the user to automatically fill in the X, Y and Z parameters with the current position as displayed on the DROs.

Autofill Workpiece Height on Main Screen

- When checked, the calculated height is automatically transferred to the Workpiece Height field in the main screen.
- Otherwise, the main screen is not affected.

Workpiece Probe At

- the X, Y and Z coordinates specify where the first probing routine should start, in current WCS

Machine Probe At

- the X, Y and Z coordinates specify where the second probing routine should start, in current WCS

Z Safe Travel Height

- The machine is raised to the Z safe travel height before jogging to the X and Y coordinates.
- The spindle then lowers to the specified Z coordinate.
- It should be selected so that the tool clears all obstructions while jogging.

START button

- The machine will jog to the first location and then probe down.
- The machine then jogs to the second location and probes down again.
- The difference in probed values is reported as Calculated Workpiece Height.
- The parameters for search velocity, probe velocity, maximum probe distance and return distance are read from the main GUI Settings page.

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.

**Caution**
Setting incorrect values can lead to crashes into fixtures on the machine work surface. Initial testing with no tool and safe heights is recommended.

#### 10.5.14.6 Tool Measurement 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)**
**gmoccapy.searchvel** *(float OUT)* - Скорость поиска переключения датчика инструмента

**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.

#### 10.5.14.7 Tool Measurement INI File Modifications

Modify your INI file to include the following:

QtDragon allows you to select one of two styles of touch probe routines. Versa probe works with a M6 remap to add auto tool probing.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

RS274NGC Section Details
Remap Statement Details
Remap Abort Details

---

**Note**
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

# is the sub, with is called when a error during tool change happens.
ON_ABORT_COMMAND=O <on_abort> call

# The remap code for QtVCP's versaprobe's automatic tool probe of Z
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

Положение датчика инструмента и начальное положение измерительного движения.
Все значения являются абсолютными (G53) координатами, за исключением MAXPROBE, который выражается как абсолютная длина перемещения.
Все значения указаны в собственных единицах измерения станка.
X, Y и Z задают местоположение измерительного датчика наладчика инструмента.

Auto probe action sequence in the default qt_auto_probe_tool example remap defined above (this behavior can be changed by modifying either the remap statement in the RS274NGC section, or by modifying the qt_auto_probe_tool.ngc code.):

1. rapid move to the INI’s [CHANGE_POSITION] Z position (this is a relative move, it adds this Z value to the current Z coordinate)
2. rapid move to the INI’s [CHANGE_POSITION] X & Y position.
3. wait for manual tool change acknowledgement
4. rapid move to the INI’s [VERSA_TOOLSETTER] X & Y position
5. rapid move to the INI’s [VERSA_TOOLSETTER] Z_MAX_CLEAR Z position
6. fast probe
7. slow probe
8. rapid move to the INI’s [VERSA_TOOLSETTER] Z_MAX_CLEAR 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:

1. rapid move to [VERSA_TOOLSETTER] Z_MAX_CLEAR Z position
2. rapid move to [VERSA_TOOLSETTER] XY position
3. rapid move to [VERSA_TOOLSETTER] Z position.

Примеры настроек:

```
[VERSA_TOOLSETTER]
X = 10
Y = 10
Z = -20
Z_MAX_CLEAR = -2
MAXPROBE =  -20
```

Это имя не называется TOOL_CHANGE_POSITION намеренно — **canon использует это имя и в противном случае будет мешать.** Положение, в которое следует переместить станок перед подачей команды на смену инструмента. Все значения указаны в оригинальных единицах станка.

```
[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
```

#### 10.5.14.8 Required HAL Connections

Make sure to connect the tool probe input in your HAL file: If connected properly, you should be able to toggle the probe LED in QtDragon if you press the probe stylus.

```
net probe  motion.probe-input <= <your_input_pin>
```

### 10.5.15 Run from Line

A G-code program can be started at any line by clicking on the desired line in the G-code display while in AUTO mode. It is the operator's responsibility to ensure the machine is in the desired operational mode. A dialog will be shown allowing the spindle direction and speed to be preset. The start line is indicated in the box labelled LINE, next to the CYCLE START button. The run from line feature can be disabled in the settings page.

---

**Note**
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 Tabs Description

Tabs allow the user to select the most appropriate info/control on the top three panels. If the on screen keyboard is showing and the user wishes to hide it but keep the current tab, they can do that by pressing the *HIDE* button on the virtual keyboard. In QtDragon, there is a splitter handle between the G-code text display and the G-code graphical display. One can use this to split the size between the two areas. This can be set differently in each tab and in each mode. The positions will be remembered.

#### 10.5.17.1 Main tab

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 File Tab

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 Offsets Tab

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 Tool Tab

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 Status Tab

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 Probe Tab

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 Tab

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 (in percent), camera port number, API backend, or requested resolution look in the preference file for:

```
[CUSTOM_FORM_ENTRIES]
Camview xscale = 100
Camview yscale = 100
Camview cam number = 0
Camview cam api = V4L2
Camview cam resolution = 1280,720
```

Scales are in percent, usually the range will be 100 - 200 in one axis.
Negating these scales can be used to flip the image in X, Y or both axes.
API comes from openCV, the available backends will be listed if -V debugging is used. Set to *ANY* for opencv to choose.
Set resolution to DEFAULT for opencv to choose. Available resolutions will be listed if -V debugging is used.

**Note**
The preference file can only be edited when QtDragon is not running.

#### 10.5.17.8 G-codes Tab

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 Setup Tab

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*.

Figure 10.35: QtDragon - Setup Tab Sample

#### 10.5.17.10 Settings Tab

The settings tab is used to set running options, probing/touchplate/laser/camera offsets and load debugging external programs.

#### 10.5.17.11 Utilities Tab

This tab will display another tab selection of G-code utility programs (and any embedded panels):

- *Facing*: allows quick face milling of a definable area at angles of 0,45 and 90 degrees.
- *Hole Circle*: allows quick setting of a program to drill a bolt circle of definable diameter and number of holes.
- *NGCGUI*: is a QtVCP version of the popular G-code subroutine builder/selector, see Widgets-NGCGUI.
- *Hole Enlarge*: allows milling a preexisting hole larger.

These tabs are detachable from the main screen by pressing the small arrow key on the far right of the tab header.
You can close the panel by pressing the arrow again or closing the window with the x button.
The last size and location of the detached window will be remembered each time the window is closed.
Custom QtVCP panels can be displayed here by setting the EMBED_TAB_LOCATION option to `tabWidget_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 Styles

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.

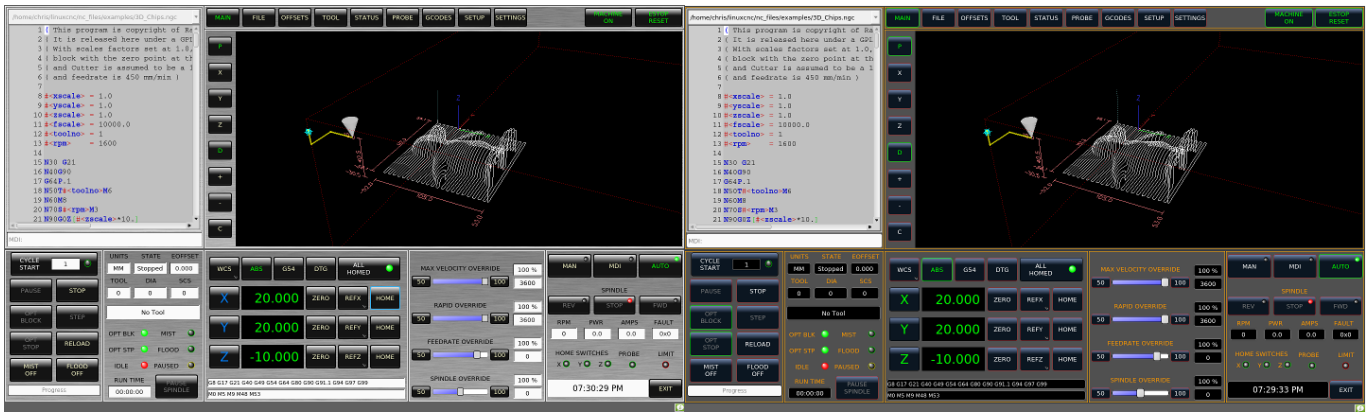


Figure 10.36: QtDragon - Two Style Examples

### 10.5.19 Internationalisation

It is possible to create translation files for QtDragon to display in the language of the current locale.

To create and or edit a translation file requires that LinuxCNC has been installed as run in place.

The following assumes that the LinuxCNC git directory is ~/linuxcnc-dev.

---

**Note**
If using QtDragon_hd substitute *qtdragon_hd* for *qtdragon*

---

All language files are kept in ~/linuxcnc-dev/share/screens/qtdragon/languages.

The qtdragon.py file is a Python version of the GUI file used for translation purposes.

The .ts files are the translation source files for the translations. These are the files that require creating/editing for each language.

The .qm files are the compiled translation files used by pyqt.

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 `qtdragon_xx.ts` in this document would actually be `qtdragon_it.ts` for an Italian translation.

The default locale for QtDragon is `_en` which means that any translation files created as `qtdragon_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
```

Change to the languages directory:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtdragon/languages
```

If any text changes have been made to the GUI then run the following to update the GUI Python file:

```
pyuic5 ../qtdragon.ui > qtdragon.py
```

The user can either create a new translation source file for a non-existing language translation or modify an existing translation source file due to changes being made to some text in a QtDragon source file. If modifying an existing translation that has had no source file changes then this step is not required.

Create or edit a .ts file:

```
./langfile xx
```

---

**Note**
this command is a script which runs the following: $ pylupdate5 **.py ..**/.py ../../../../../lib/python/qtvcp/lib/qtdragon/*.py -ts qtdragon_xx.ts

---

The editing of the translation is done with the linguist application:

```
linguist
```

1. Open the TS file and translate the strings

It is not necessary to provide a translation for every text string, if no translation is specified for a string then the original string will be used in the application. The user needs to be careful with the length of strings that appear on widgets as space is limited. If possible try to make the translation no longer than the original.

When editing is complete save the file:
`File -> Save`

Then create the .qm file:
`File -> Release`

QtDragon will be translated to the language of the current locale on the next start so long as a .qm file exists in that language.

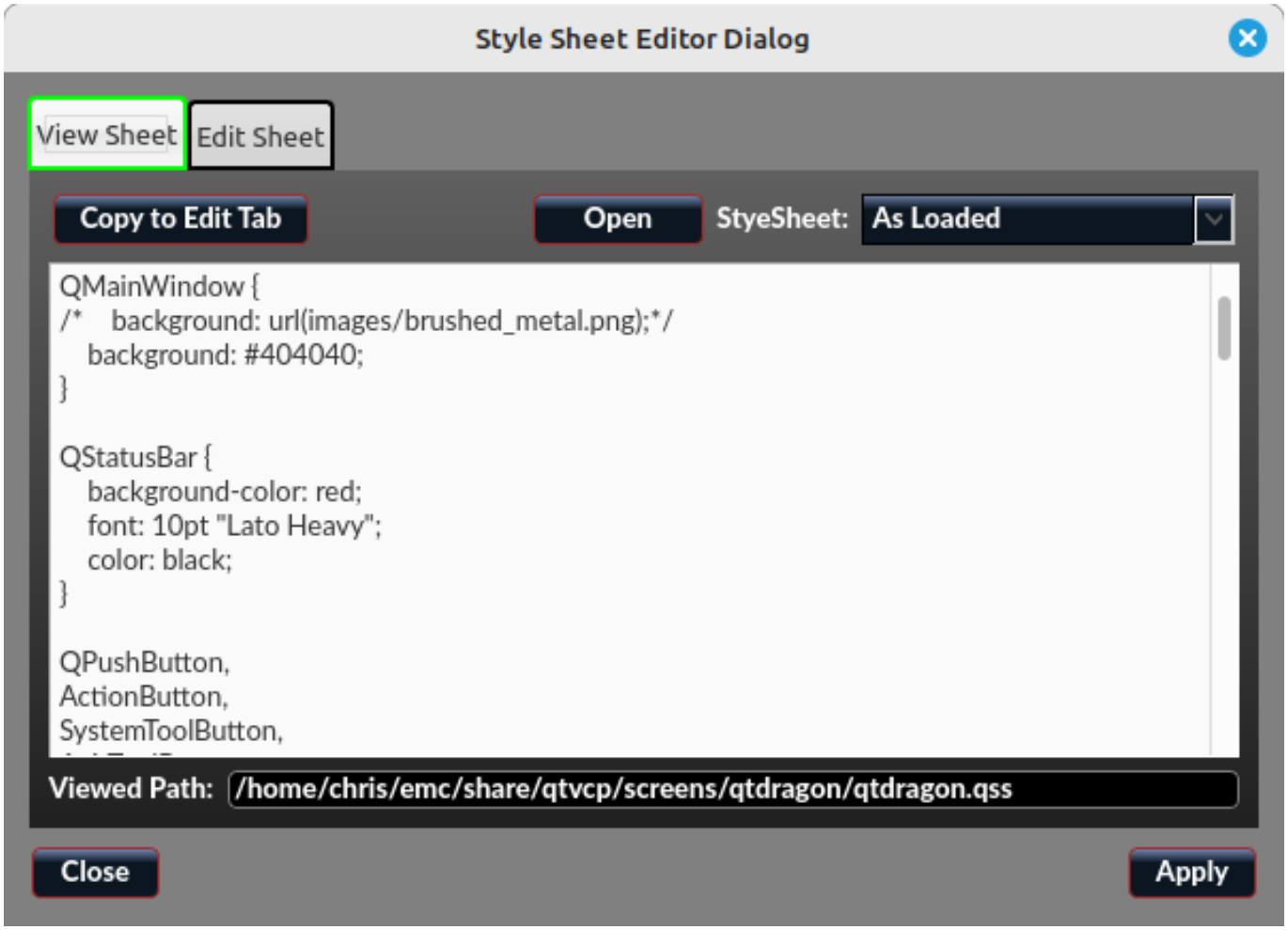
Users are welcome to submit translation files for inclusion into QtDragon. The preferred method is to submit a pull request from the users GitHub account as described in the contributing to LinuxCNC documentation. The only files required to be committed are qtdragon_xx.ts and qtdragon_xx.qm.

### 10.5.20 Customization

A general overview of Customizing Stock Screens.

#### 10.5.20.1 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.
The *View Sheet* tab will allow you to select and apply what stylesheet QtDragon will use when it's first loaded.
Press the button *Copy to Edit Tab* to copy the current stylesheet to the edit tab.
The *Edit Sheet* tab allows editing, applying and saving of changes od the displayed stylesheet.




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';

    /*Adjust the menu options */
    qproperty-showLast: true;
    qproperty-showDivide : false;
    qproperty-showGotoOrigin: false;
    qproperty-showZeroOrigin: false;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}
```

Change the axis select button's click menu items:

```
AxisToolButton {
    /* Adjust all the menu options */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

To change the text of the mist button to *air* (add these lines)

```
#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;
}
```

To change status highlight/selection colors:

```
#screen_options {
    qproperty-user1Color: white;     /*  default status     */
    qproperty-user2Color: #ff9000;   /*  warning status     */
    qproperty-user3Color: #ff8a96;   /*  critical status    */
    qproperty-user4Color: #ffaa00;   /*  MPG select         */
    qproperty-user5Color: #ff0000;   /*  Cycle Start select */
}
```

Change the G-code text display colors/fonts:

```
EditorBase{
background:black;
qproperty-styleColorCursor:white;
qproperty-styleColorBackground:grey;
qproperty-styleColor0: black;
qproperty-styleColor1: darkblue;
qproperty-styleColor2: blue;
qproperty-styleColor3: red;
qproperty-styleColor4: lightblue;
qproperty-styleColor5: white;
qproperty-styleColor6: lightGreen;
qproperty-styleColor7: yellow ;
qproperty-styleColorSelectionText: white;
qproperty-styleColorSelectionBackground: blue;
qproperty-styleFont0: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont1: ”Times,15,-1,5,90,1,0,1,0,0”;
qproperty-styleFont2: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont3: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont4: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont5: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont6: ”Times,15,-1,5,90,0,0,1,1,0”;
qproperty-styleFont7: ”Times,15,-1,5,90,0,0,1,1,0”;
}
```

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;
}
```

#### 10.5.20.2 Qt Designer and 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.
Custom modifications can be added by *subclassing* the handler file. This adds code on top of the original.
QtDragon can also utilize QtVCP’s rc file to do minor python code modifications without using a custom handler file.

See Modifying Screens for more information about customization.

Some widget customization is available for basic probe and versa probe.

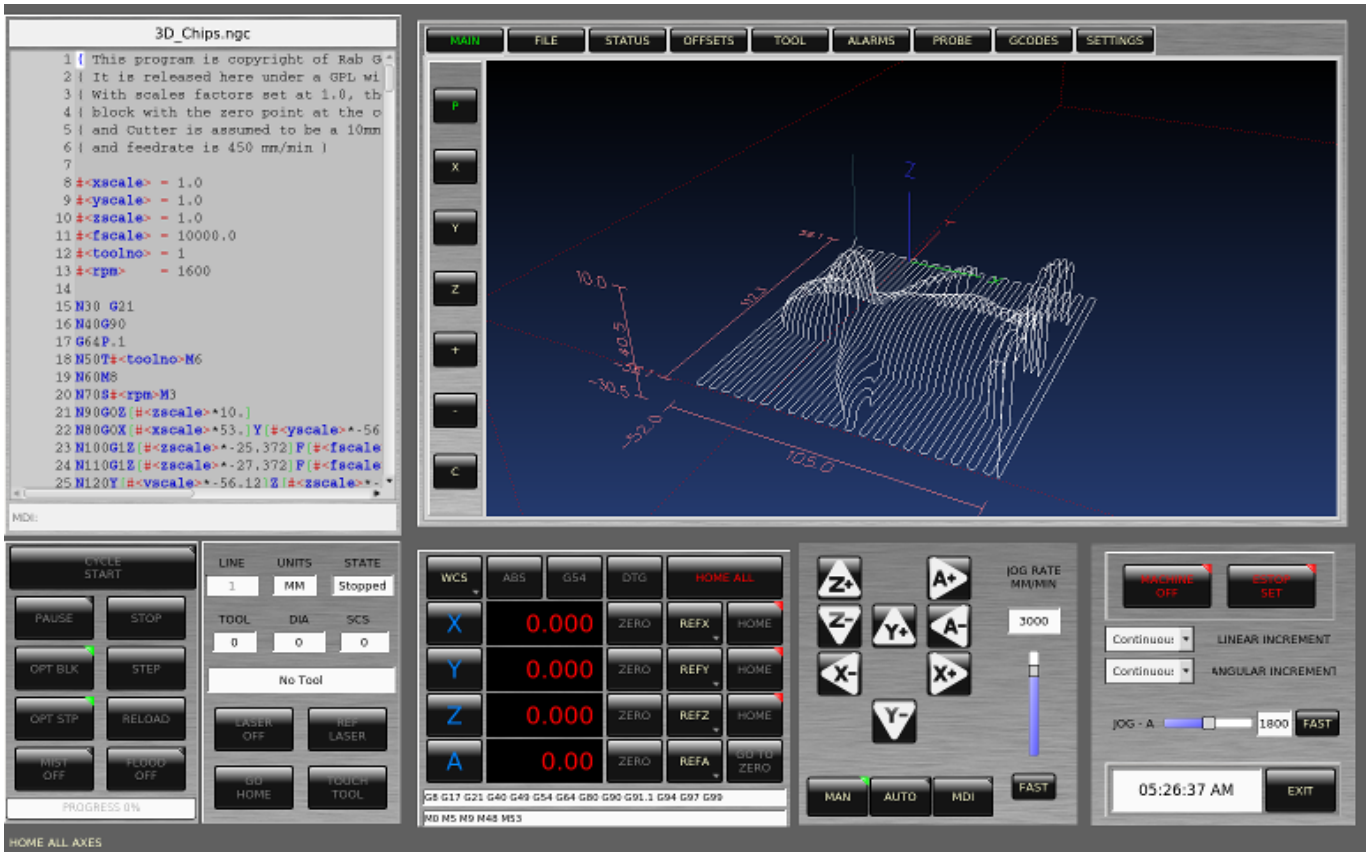



Figure 10.37: QtDragon - Customized QtDragon

## 10.6 NGCGUI

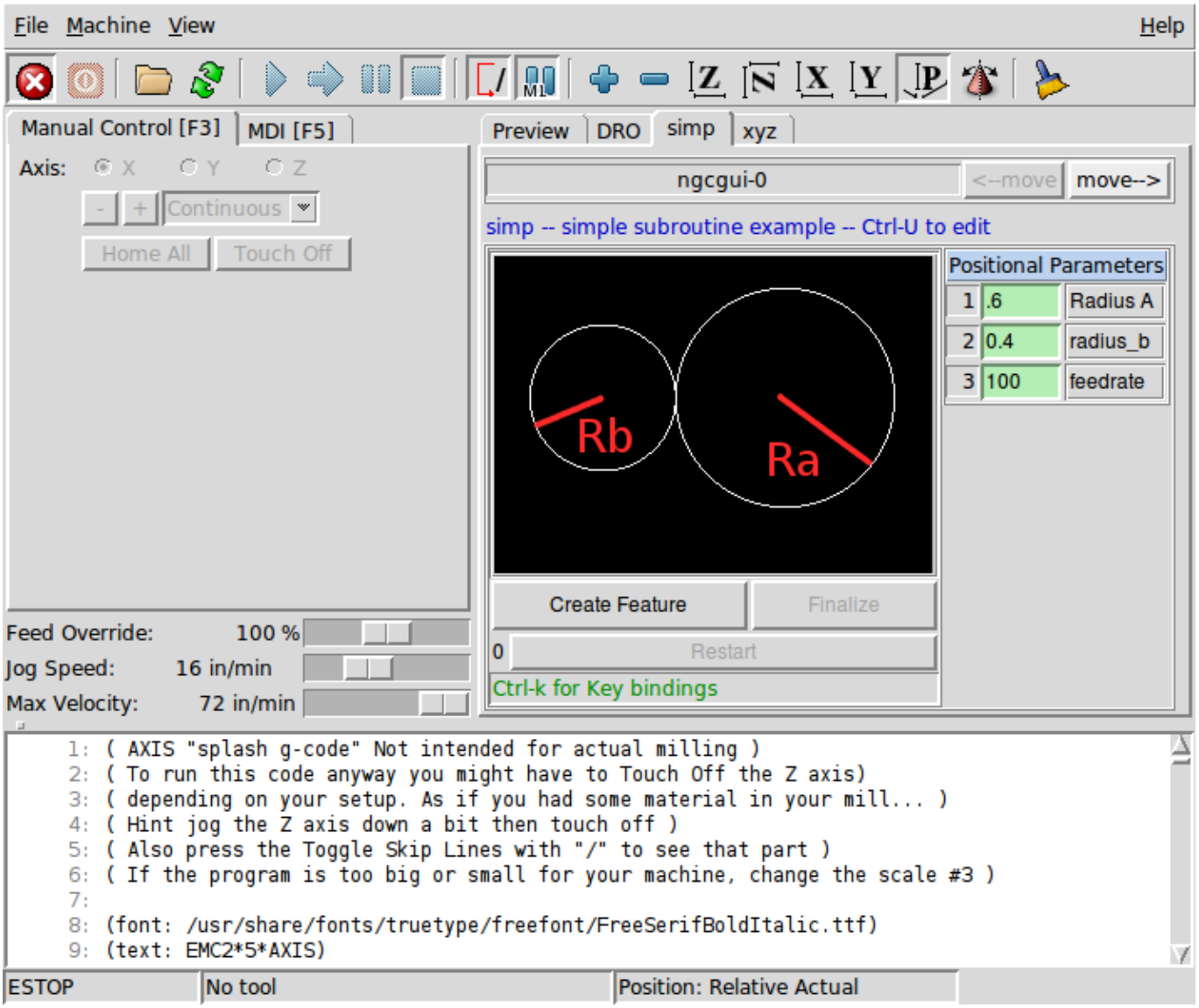



Figure 10.38: NGCGUI embedded into AXIS

### 10.6.1 Обзор

* *NGCGUI* is a Tcl application to work with subroutines. It allows you to have a conversational interface with LinuxCNC. You can organize the subroutines in the order you need them to run and concatenate the subroutines into one file for a complete part program. * *NGCGUI* can run as a standalone application or can be embedded in multiple tab pages in the AXIS GUI. * *PyNGCGUI* is an alternate, Python implementation of NGCGUI. * *PyNGCGUI* can run as a standalone application or can be embedded as a tab page (with its own set of multiple subroutine tabs) in any GUI that supports embedding of GladeVCP applications AXIS, Touchy, Gscreen and GMOCCAPY.

Using NGCGUI or PyNGCGUI:

- Tab pages are provided for each subroutine specified in the INI file.

- New subroutines tab pages can be added on the fly using the custom tab.
- Each subroutine tab page provides entry boxes for all subroutine parameters.
- The entry boxes can have a default value and an label that are identified by special comments in the subroutine file.
- Subroutine invocations can be concatenated together to form a multiple step program.
- Any single-file G-code subroutine that conforms to NGCGUI conventions can be used.
- 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)

**Note**
NGCGUI and PyNGCGUI implement the same functions and both process .ngc and .gcmc files that conform to a few NGCGUI-specific conventions. In this document, the term *NGCGUI* generally refers to either application.

### 10.6.2 Demonstration Configurations

A number of demonstration configurations are located in the sim directory of the Sample Configurations offered by the LinuxCNC configuration picker. The configuration picker is on the system's main menu: Applications > CNC > LinuxCNC

Examples are included for the AXIS, Touchy, gscreen, and GMOCCAPY. These examples demonstrate both 3-axis (XYZ) cartesian configurations (like mills) and lathe (XZ) setups. Some examples show the use of a pop up keyboard for touch screen systems and other examples demonstrate the use of files created for the gcmc (G-code Meta Compiler) application. The touchy examples also demonstrate incorporation of a GladeVCP back plot viewer (gremlin_view).

The simplest application is found as:

`Sample Configurations/sim/axis/ngcgui/ngcgui_simple`

A comprehensive example showing gcmc compatibility is at:

`Sample Configurations/sim/axis/ngcgui/ngcgui_gcmc`

A comprehensive example embedded as a GladeVCP app and using gcmc is at:

`Sample Configurations/sim/gscreen/ngcgui/pyngcgui_gcmc`

The example sim configurations make use of library files that provide example G-code subroutine (.ngc) files and G-code-meta-compiler (.gcmc) files:

- *nc_files/ngcgui_lib*
  - *ngcgui.ngc* - An easy to understand example using subroutines
  - *arc1.ngc* - basic arc using cutter radius compensation
  - *arc2.ngc* - arc speced by center, offset, width, angle (calls arc1)
  - *backlash.ngc* - routine to measure an axis backlash with dial indicator
  - *db25.ngc* - creates a DB25 plug cutout
  - *gosper.ngc* - a recursion demo (flowsnake)
  - *helix.ngc* - helix or D-hole cutting

  - *helix_rtheta.ngc* - helix or D-hole positioned by radius and angle
  - *hole_circle.ngc* - equally spaced holes on a circle
  - *ihex.ngc* - internal hexagon
  - *iquad.ngc* - internal quadrilateral
  - *ohex.ngc* - outside hexagon
  - *oquad.ngc* - outside quadrilateral
  - *qpex_mm.ngc* - demo of qpockets (mm based)
  - *qpex.ngc* - demo of qpockets (inch based)
  - *qpocket.ngc* - quadrilateral pocket
  - *rectangle_probe.ngc* - probe a rectangular area
  - *simp.ngc* - a simple subroutine example that creates two circles
  - *slot.ngc* - slot from connecting two endpoints
  - *xyz.ngc* - machine exerciser constrained to a box shape
  - *Custom* - Creates custom tabs
  - *ttt* - True Type Tracer, to create texts to be engraved
- *nc_files/ngcgui_lib/lathe*
  - *ngcgui-lathe* - Example lathe subroutine
  - *g76base.ngc* - GUI for G76 threading
  - *g76diam.ngc* - threading speced by major, minor diameters
  - *id.ngc* - bores the inside diameter
  - *od.ngc* - turns the outside diameter
  - *taper-od.ngc* - turns a taper on the outside diameter
  - *Custom* - Creates custom tabs
- *nc_files/gcmc_lib*
  - *drill.gcmc* - drill holes in rectangle pattern
  - *square.gcmc* - simple demo of variable tags for gcmc files
  - *star.gcmc* - gcmc demo illustrating functions and arrays
  - *wheels.gcmc* - gcmc demo of complex patterns

To try a demonstration, select a sim configuration and start the LinuxCNC program.

If using the AXIS GUI, press the *E-Stop* then *Machine Power* then *Home All*. Pick a NGCGUI tab, fill in any empty blanks with sensible values and press *Create Feature* then *Finalize*. Finally press the *Run* button to watch it run. Experiment by creating multiple features and features from different tab pages.

To create several subroutines concatenated into a single file, go to each tab fill in the blanks, press *Create Feature* then using the arrow keys move any tabs needed to put them in order. Now press *Finalize* and answer the prompt to create

Other GUIs will have similar functionality but the buttons and names may be different.

**Note**
The demonstration configs create tab pages for just a few of the provided examples. Any GUI with a custom tab can open any of the library example subroutines or any user file if it is in the LinuxCNC subroutine path.
To see special key bindings, click inside an NGCGUI tab page to get focus and then press Control-k.
The demonstration subroutines should run on the simulated machine configurations included in the distribution. A user should always understand the behavior and purpose of a program before running on a real machine.

### 10.6.3 Library Locations

In LinuxCNC installations installed from deb packages, the simulation configs for NGCGUI use symbolic links to non-user-writable LinuxCNC libraries for:

- *nc_files/ngcgui_lib* NGCGUI-compatible subfiles
- *nc_files/ngcgui_lib/lathe* NGCGUI-compatible lathe subfiles
- *nc_files/gcmc_lib* NGCGUI-gcmc-compatible programs
- *nc_files/ngcgui_lib/utilitysubs* Helper subroutines
- *nc_files/ngcgui_lib/mfiles* User M files

These libraries are located by INI file items that specify the search paths used by LinuxCNC (and NGCGUI):

```
[RS274NGC]
SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/ ↩
    ngcgui_lib/utilitysubs
USER_M_PATH     = ../../nc_files/ngcgui_lib/mfiles
```

**Note**
These are long lines (not continued on multiple lines) that specify the directories used in a search patch. The directory names are separated by colons (:). No spaces should occur between directory names.

A user can create new directories for their own subroutines and M-files and add them to the search path(s).

For example, a user could create directories from the terminal with the commands:

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

And then create or copy system-provided files to these user-writable directories. For instance, a user might create a NGCGUI-compatible subfile named:

```
/home/myusername/mysubs/example.ngc
```

To use files in new directories, the INI file must be edited to include the new subfiles and to augment the search path(s). For this example:

```
[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 (and NGCGUI) use the first file found when searching directories in the search path. With this behavior, you can supersede an ngcgui_lib subfile by placing a subfile with an identical name in a directory that is found earlier in the path search. More information can be found in the INI chapter of the Integrators Manual.

## 10.6.4 Standalone Usage

### 10.6.4.1 Standalone NGCGUI

For usage, type in a terminal:

```
ngcgui --help
Usage:
  ngcgui --help | -?
  ngcgui [Options] -D <nc files directory name>
  ngcgui [Options] -i <LinuxCNC INI file name>
  ngcgui [Options]

Options:
  [-S subroutine_file]
  [-p preamble_file]
  [-P postamble_file]
  [-o output_file]
  [-a autosend_file]            (autosend to AXIS default:auto.ngc)
  [--noauto]                    (no autosend to AXIS)
  [-N | --nom2]                 (no m2 terminator (use %))
  [--font [big|small|fontspec]] (default: ”Helvetica -10 normal”)
  [--horiz|--vert]              (default: --horiz)
  [--cwidth comment_width]      (width of comment field)
  [--vwidth varname_width]      (width of varname field)
  [--quiet]                     (fewer comments in outfile)
  [--noiframe]                  (default: frame displays image)
```

**Note**
As a standalone application, NGCGUI handles a single subroutine file which can be invoked multiple times. Multiple standalone NGCGUI applications can be started independently.

### 10.6.4.2 Standalone PyNGCGUI

For usage, type in a terminal:

```
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.
```

**Note**
As a standalone application, PyNGCGUI can read an INI file (or a running LinuxCNC application) to create tab pages for multiple subfiles.

## 10.6.5 Embedding NGCGUI

### 10.6.5.1 Embedding NGCGUI in AXIS

The following INI file items go in the [DISPLAY] section. (See additional sections below for additional items needed)

- *TKPKG = Ngcgui 1.0* - the NGCGUI package
- *TKPKG = Ngcguittt 1.0* - the True Type Tracer package for generating text for engraving (optional, must follow TKPKG = Ngcgui).
- *NGCGUI_FONT = Helvetica -12 normal* - Sets the font used
- *NGCGUI_PREAMBLE = in_std.ngc* - The preamble file to be added at the beginning of the subroutine. When several subroutines are concatenated, it is only added once.
- *NGCGUI_SUBFILE = simp.ngc* - Creates a tab from the named subroutine.
- *NGCGUI_SUBFILE = ””* - Creates a custom tab
- *#NGCGUI_OPTIONS = opt1 opt2 …* - NGCGUI options:
  - *nonew* — Prohibits creation of new custom tab
  - *noremove* — Prohibits deleting a tab page
  - *noauto* — Do not run automatically (makeFile, then manual run)
  - *noiframe* — No internal image, image on separate top level
- *TTT = truetype-tracer* - name of the truetype tracer program (it must be in user PATH)
- *TTT_PREAMBLE = in_std.ngc* - Optional, specifies filename for preamble used for ttt created subfiles. (alternate: mm_std.ngc)

**Note**
The optional truetype tracer items are used to specify an NGCGUI-compatible tab page that uses the application truetype-tracer. The truetype-tracer application must be installed independently and located in the user PATH.

#### 10.6.5.2 Embedding PyNGCGUI as a GladeVCP tab page in a GUI

The following INI file items go in the [DISPLAY] section for use with the AXIS, Gscreen, or Touchy GUIs. (See additional sections below for additional items needed)

EMBED_ Items

- `EMBED_TAB_NAME = PyNGCGUI` - name to appear on embedded tab
- `EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui` - invokes GladeVCP
- `EMBED_TAB_LOCATION = name_of_location` - where the embedded page is located

---

**Note**
The EMBED_TAB_LOCATION specifier is not used for the AXIS GUI. While PyNGCGUI can be embedded in AXIS, integration is more complete when using NGCGUI (using TKPKG = Ngcgui 1.0). To specify the EMBED_TAB_LOCATION for other GUIs, see the DISPLAY Section of the INI Configuration Chapter.

---

---

**Note**
The truetype tracer GUI front-end is not currently available for GladeVCP applications.

---

#### 10.6.5.3 Additional INI File items required for NCGUI or PyNGCGUI

The following INI file items go in the [DISPLAY] section for any GUI that embeds either NGCGUI or PyNGCGUI.

- *NGCGUI_FONT = Helvetica -12 normal* - specifies the font name,size, normal|bold
- *NGCGUI_PREAMBLE = in_std.ngc* - the preamble file to be added in front of the subroutines. When concatenating several common subroutine invocations, this preamble is only added once. For mm-based machines, use mm_std.ngc
- *NGCGUI_SUBFILE = filename1.ngc* - creates a tab from the filename1 subroutine
- *NGCGUI_SUBFILE = filename2.ngc* - creates a tab from the filename2 subroutine
- *... etc.*
- *NGCGUI_SUBFILE = gcmcname1.gcmc* - creates a tab from the gcmcname1 file
- *NGCGUI_SUBFILE = gcmcname2.gcmc* - creates a tab from the gcmcname2 file
- *... etc.*
- *NGCGUI_SUBFILE = ””* - creates a custom tab that can open any subroutine in the search path
- *NGCGUI_OPTIONS = opt1 opt2 ...* - NGCGUI options
  - *nonew* - disallow making a new custom tab
  - *noremove* - disallow removing any tab page
  - *noauto* - no autosend (use makeFile, then save or manually send)
  - *noiframe* - no internal image, display images on separate top level widget
  - *nom2* - do not terminate with m2, use % terminator. This option eliminates all the side effects of m2 termination

- *GCMC_INCLUDE_PATH = dirname1:dirname2* - search directories for gcmc include files

This is an example of embedding NGCGUI into AXIS. The subroutines need to be in a directory specified by the [RS274NGC]SUBROUTINE_PATH. Some example subroutines use other subroutines so check to be sure you have the dependences, if any, in a SUBROUTINE_PATH directory. Some subroutines may use custom M-files which must be in a directory specified by the [RS274NGC]USER_M_PATH.

The G-code-meta-compiler (gcmc) can include statements like:

```
include(”filename.inc.gcmc”);
```

By default, gcmc includes the current directory which, for LinuxCNC, will be the directory containing the LinuxCNC INI file. Additional directories can be prepended to the gcmc search order with the GCMC_INCLUDE_PATH item.

**Sample AXIS-GUI-based 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 must precede Ngcguittt

NGCGUI_FONT       = Helvetica -12 normal
# specify filenames only, files must be in [RS274NGC]SUBROUTINE_PATH
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
# specify ”” for a custom tab page
NGCGUI_SUBFILE    = ””
#NGCGUI_SUBFILE   = ”” use when image frame is specified if
#                     opening other files is required
#                     images will be put in a top level window
NGCGUI_OPTIONS    =
#NGCGUI_OPTIONS   = opt1 opt2 ...
# opt items:
#   nonew      -- disallow making a new custom tab
#   noremove   -- disallow removing any tab page
#   noauto     -- no auto send (makeFile, then manually send)
#   noiframe   -- no internal image, image on separate top level
GCMC_INCLUDE_PATH = /home/myname/gcmc_includes

TTT               = truetype-tracer
TTT_PREAMBLE      = in_std.ngc

PROGRAM_PREFIX    = ../../nc_files
```

**Note**
The above is not a complete AXIS GUI INI — the items show are those used by NGCGUI. Many additional items are required by LinuxCNC to have a complete INI file.

#### 10.6.5.4 Truetype Tracer

Ngcgui_ttt provides support for truetype-tracer (v4). It creates an AXIS tab page which allows a user to create a new NGCGUI tab page after entering text and selecting a font and other parameters. (Truetype-tracer must be installed independently).

To embed ngcgui_ttt in AXIS, specify the following items in addition to NGCGUI items:

```
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 = directory_name
```

followed by multiple directories specified by:

```
[RS274NGC]SUBROUTINE_PATH = directory1_name:directory1_name:directory3_name ...
```

**Directories** Directories may be specified as absolute paths or relative paths.

- Example: `[DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files`
- Example: `[DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/nc_files`
- Example: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`

**Absolute Paths** An absolute path beginning with a ”/” specifies a complete filesystem location. A path beginning with a ”~/” specifies a path starting from the user’s home directory. A path beginning with ”~username/” specifies a path starting in username’s home directory.

**Относительные пути** Relative paths are based on the startup directory which is the directory containing the INI file. Using relative paths can facilitate relocation of configurations but requires a good understanding of Linux path specifiers.

- `./d0` is the same as d0, e.g., a directory named d0 in the startup directory
- `../d1` refers to a directory d1 in the parent directory
- `../../d2` refers to a directory d2 in the parent of the parent directory
- `../../../d3` etc.

Multiple directories can be specified with [RS274NGC]SUBROUTINE_PATH by separating them with colons. The following example illustrates the format for multiple directories and shows the use of relative and absolute paths.

**Multiple Directories Example:**

```
[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs ↩
    :/tmp/tmpngc
```

This is one long line, do not continue on multiple lines. When LinuxCNC and/or NGCGUI searches for files, the first file found in the search is used.

LinuxCNC (and NGCGUI) must be able to find all subroutines including helper routines that are called from within NGCGUI subfiles. It is convenient to place utility subs in a separate directory as indicated in the example above.

The distribution includes the ngcgui_lib directory and demo files for preambles, subfiles, postambles and helper files. To modify the behavior of the files, you can copy any file and place it in an earlier part of the search path. The first directory searched is [DISPLAY]PROGRAM_PREFIX. You can use this directory but it is better practice to create dedicated directory(ies) and put them at the beginning of the [RS274NGC]SUBROUTINE_PATH.

In the following example, files in /home/myname/linuxcnc/mysubs will be found before files in ../../nc_files/ngc

**Adding User Directory Example:**

```
[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 Summary of INI File item details for NGCGUI usage

**[RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...**
*Example*: `[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_l`
*Note*: Optional, but very useful to organize subfiles and utility files.

**[RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...**
*Example*: `[RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles`
*Note*: Optional, needed to locate custom user M-files.

**[DISPLAY]EMBED_TAB_NAME = name to display on embedded tab page**
*Example*: `[DISPLAY]EMBED_TAB_NAME = Pyngcgui`
*Note*: The entries: `EMBED_TAB_NAME`, `EMBED_TAB_COMMAND`, `EMBED_TAB_LOCATION` define an embedded application for several LinuxCNC GUIs.

**[DISPLAY]EMBED_TAB_COMMAND = programname followed by arguments**
*Example*: `[DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui`
*Note*: For GladeVCP applications, see the GladeVCP Chapter.

**[DISPLAY]EMBED_TAB_LOCATION = name_of_location**
*Example*: `[DISPLAY]EMBED_TAB_LOCATION = notebook_main`
*Note*: See example INI files for possible locations.
Not required for the AXIS GUI.

**[DISPLAY]PROGRAM_PREFIX = dirname**
*Example*: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`
*Note*: Mandatory and needed for numerous LinuxCNC functions.
It is the first directory used in the search for files.

**[DISPLAY]TKPKG = NGCGUI version_number**
*Example*: `[DISPLAY]TKPKG = Ngcgui 1.0`
*Note*: Required only for AXIS GUI embedding.
Specifies loading of NGCGUI AXIS tab pages.

**[DISPLAY]NGCGUI_FONT = font_descriptor**
*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 .

**[DISPLAY]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.

**[DISPLAY]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 ...**
*Example*: [DISPLAY]NGCGUI_OPTIONS = nonew noremove
*Note*: Multiple options are separated by blanks.
By default, NGCGUI configures tab pages so that:
1) a user can make new tabs;
2) a user can remove tabs (except for the last remaining one);
3) finalized files are automatically sent to LinuxCNC;
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).

The options `nonew`, `noremove`, `noauto`, `noiframe`, `nom2` respectively disable these default behaviors.

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_inc`
*Note*: Optional, each directory will be included when gcmc is invoked using the option: `--include dirname`.

## 10.6.6 File Requirements for NGCGUI Compatibility

### 10.6.6.1 Single-File Gcode (.ngc) Subroutine Requirements

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.

The first non-comment line should be a `sub` statement.
The last non-comment line should be a `endsub` statement.

**examp.ngc:**

```
(info: info_text_to_appear_at_top_of_tab_page)
; comment line beginning with semicolon
( comment line using parentheses)
o<examp> sub
  BODY_OF_SUBROUTINE
o<examp> endsub
; comment line beginning with semicolon
( comment line using parentheses)
```

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).

**Parameter Numbering**

```
#<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.

Each defining statement may optionally include a special comment and a default value for the parameter.

**Statement Prototype**

```
#<vname> = #n (=default_value)
or
#<vname> = #n (comment_text)
or
#<vname> = #n (=default_value comment_text)
```

**Parameter Examples**

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z start setting)
```

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.

**Global Named Parameters** Notes on global named parameters and NGCGUI:

(global named parameters have a leading underscore in the name, like #<_someglobalname>)

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 supports some input global named parameter but their usage is obsolete and not documented here.

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.

**Global Named Parameters Example**

```
o<examp> sub
...
#<_examp:result> = #5410       (return the current tool diameter)
...
o<helper> call [#<x1>] [#<x2>] (call a subroutine)
#<xresult> = #<_helper:answer> (immediately localize the helper global result)
#<_helper:answer> = 0.0        (nullify global named parameter used by subroutine)
...
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.

**Additional Features** A special *info* comment can be included anywhere in an NGCGUI-compatible subfile. The format is:

```
(info: info_text)
```

The info_text is displayed near the top of the NGCGUI tab page in AXIS.

Files not intended for use as subfiles can include a special comment so that NGCGUI will reject them automatically with a relevant message.

```
(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.

None of the conventions required for making an NGCGUI-compatible subfile preclude its use as general purpose subroutine file for LinuxCNC.

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).

Additional user sumitted subroutines can be found on the Forum in the Subroutines Section.

#### 10.6.6.2 Gcode-meta-compiler (.gcmc) file requirements

Files for the Gcode-meta-compiler (gcmc) are read by NGCGUI and it creates entry boxes for variables tagged in the file. When a feature for the file is finalized, NGCGUI passes the file as input to the gcmc compiler and, if the compile is successful, the resulting G-code file is sent to LinuxCNC for execution. The resulting file is formatted as single-file subroutine; .gcmc files and .ngc files can be intermixed by NGCGUI.

The variables identified for inclusion in NGCGUI are tagged with lines that will appear as comments to the gcmc compiler.

**Variable Tags Formats**

```
//ngcgui: varname1 =
//ngcgui: varname2 = value2
//ngcgui: varname3 = value3, label3;
```

**Variable Tags Examples**

```
//ngcgui: zsafe =
//ngcgui: feedrate = 10
//ngcgui: xl = 0, x limit
```

For these examples, the entry box for varname1 will have no default, the entry box for varname2 will have a default of value2, and the entry box for varname 3 will have a default of value 3 and a label label3 (instead of varname3). The default values must be numbers.

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.

**Alternate Variable Tags Formats**

```
//ngcgui: varname2 = value2;
//ngcgui: varname3 = value3; //, label3;
```

**Alternate Variable Tags Examples**

```
//ngcgui: feedrate = 10;
//ngcgui: xl = 0; //, x limit
```

An info line that will appear at the top of a tab page may be optionally included with a line tagged as:

**Info tag**

```
//ngcgui: info: text_to_appear_at_top_of_tab_page
```

When required, options can be passed to the gcmc compiler with a line tagged:

**Option line tag format**

```
//ngcgui: -option_name [ [=] option_value]
```

**Option line tag Examples**

```
//ngcgui: -I
//ngcgui: --imperial
//ngcgui: --precision 5
//ngcgui: --precision=6
```

Options for gcmc are available with the terminal command:

```
gcmc --help
```

A gcmc program by default uses metric mode. The mode can be set to inches with the option setting:

```
//ngcgui: --imperial
```

A preamble file, if used, can set a mode (g20 or g21) that conflicts with the mode used by a gcmc file. To ensure that the gcmc program mode is in effect, include the following statement in the .gcmc file:

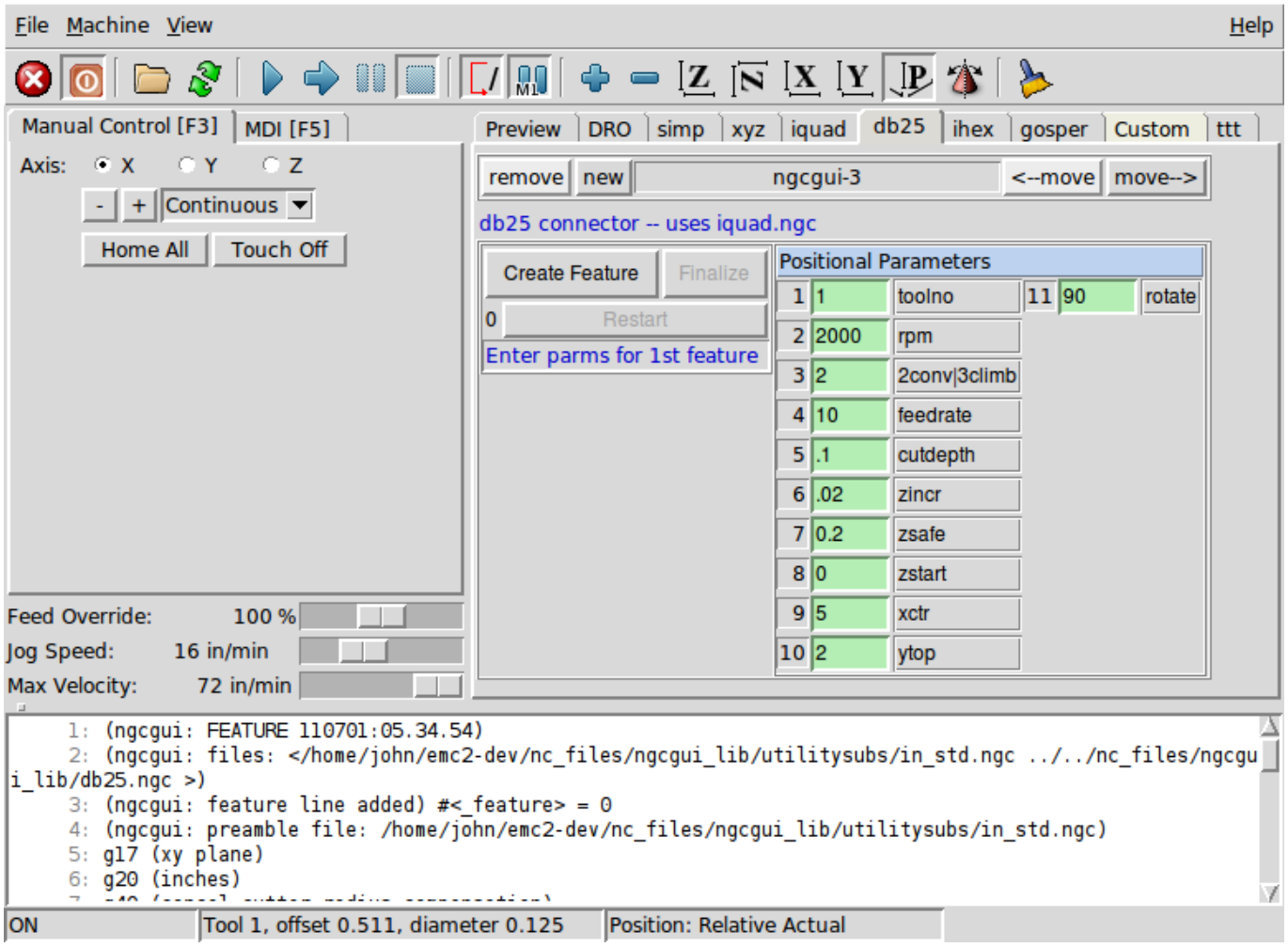
```
include(”ensure_mode.gcmc”)
```

and provide a proper path for gcmc include_files in the INI file, for example:

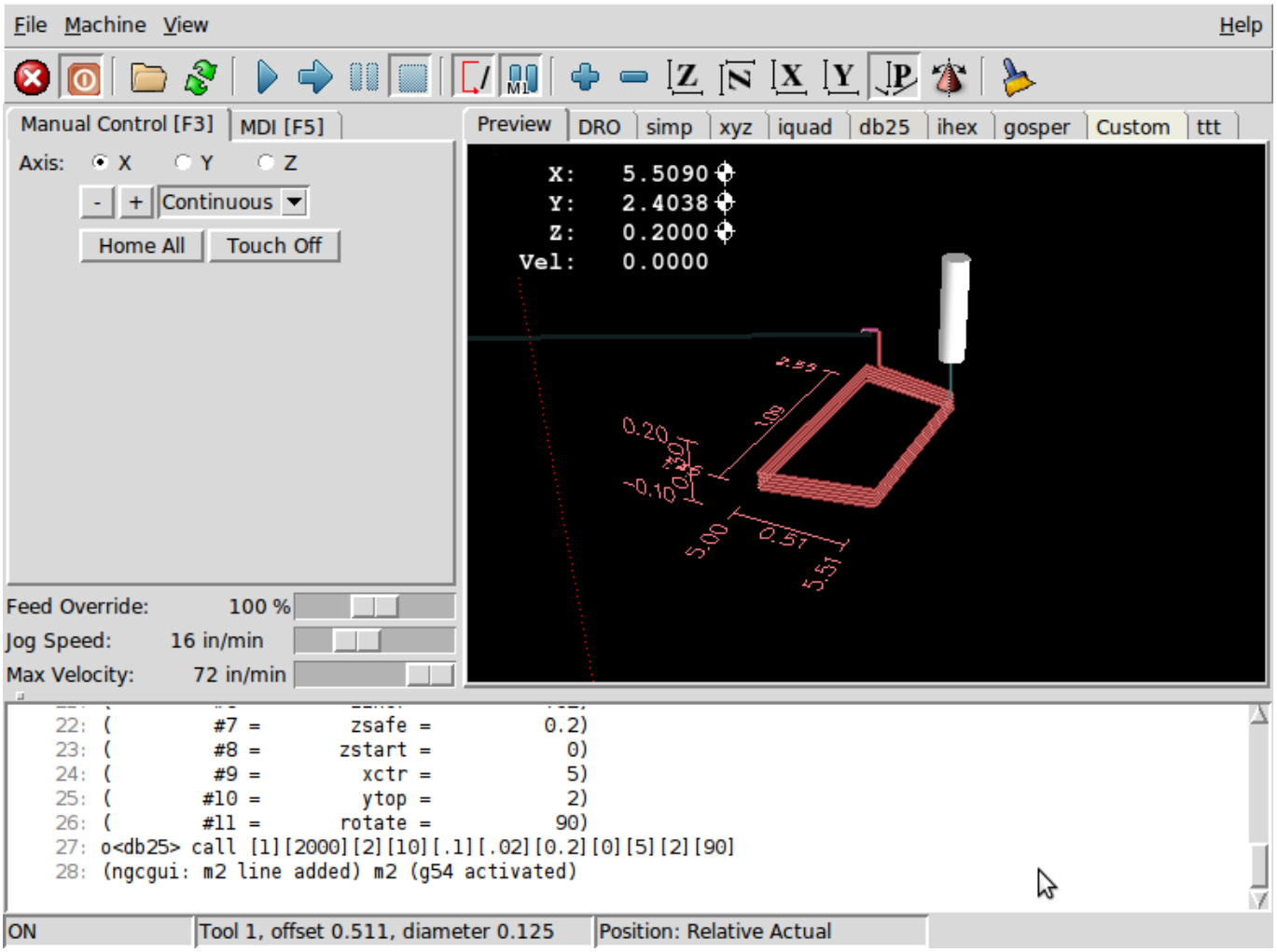
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

### 10.6.7 DB25 Example

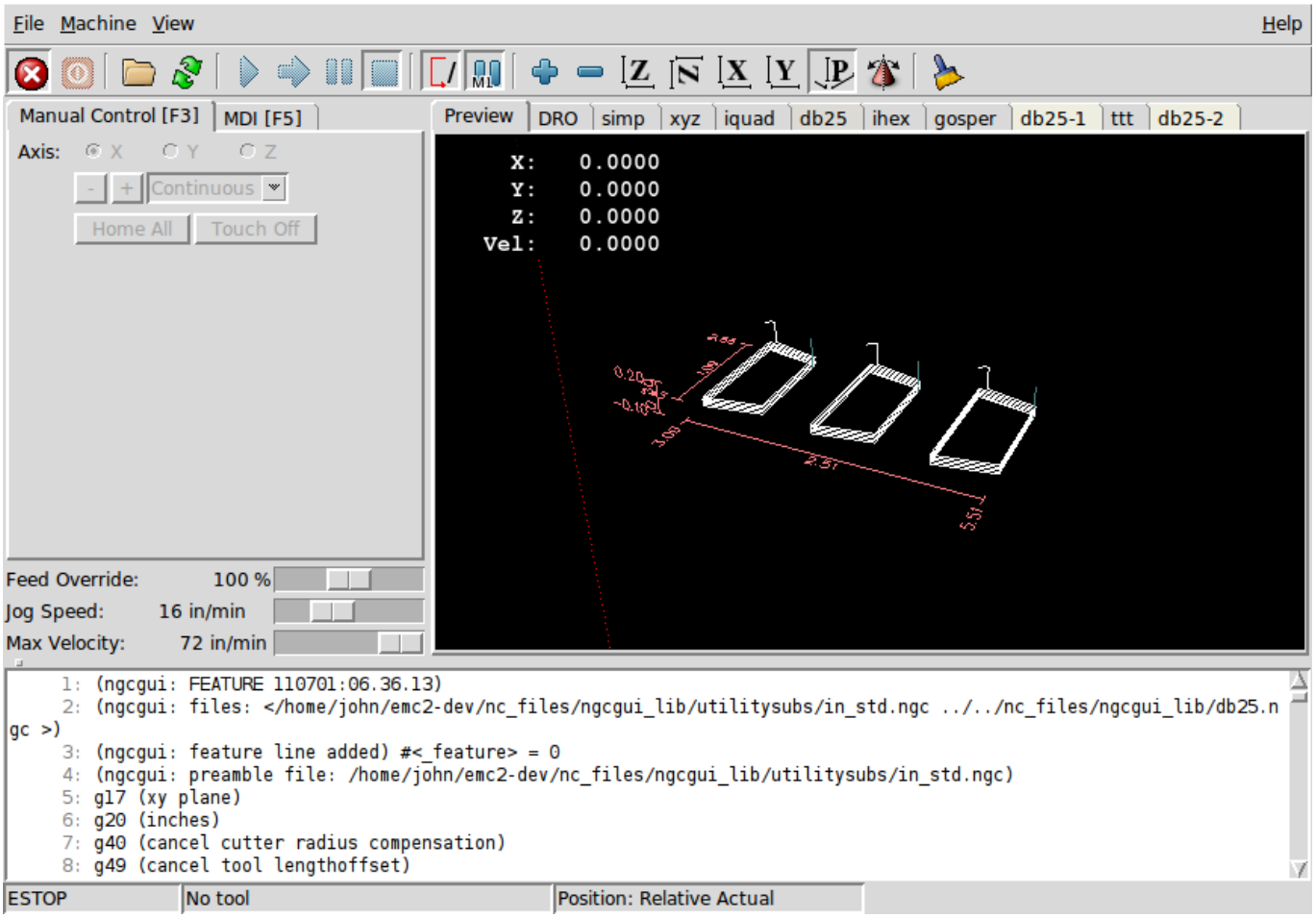
The following shows the DB25 subroutine. In the first photo you see where you fill in the blanks for each variable.




This photo shows the backplot of the DB25 subroutine.




This photo shows the use of the new button and the custom tab to create three DB25 cutouts in one program.




## 10.6.8 Creating a subroutine

- For creating a subroutine for use with NGCGUI, the filename and the subroutine name must be the same.
- The file must be placed in the subdirectory pointed to in the INI file.
- On the first line there may be a comment of type `info:`
- The subroutine must be surrounded by the `sub` and `endsub` tags.
- The variables used must be numbered variables and must not skip number.
- Comments and presets may be included.

### Subroutine Skeleton Example

```
(info: simp -- simple exemple de sous-programme -- Ctrl-U pour éditer)
o<simp> sub
  #<ra>       = #1 (=.6 Rayon A) ;Example de paramètre avec un commentaire
  #<radius_b> = #2 (=0.4)         ;Example de paramètre sans commentaire
  #<feedrate> = #3 (Feedrate)     ;Example de paramètre sans preset
  g0x0y0z1
  g3 i#<ra> f#<feedrate>
  g3 i[0-#<radius_b>]
o<simp> endsub
```

## 10.7 TkLinuxCNC GUI

### 10.7.1 Введение

TkLinuxCNC is one of the first graphical front-ends for LinuxCNC. It is written in Tcl and uses the Tk toolkit for the display. Being written in Tcl makes it very portable (it runs on a multitude of platforms). A separate backplot window can be displayed as shown.

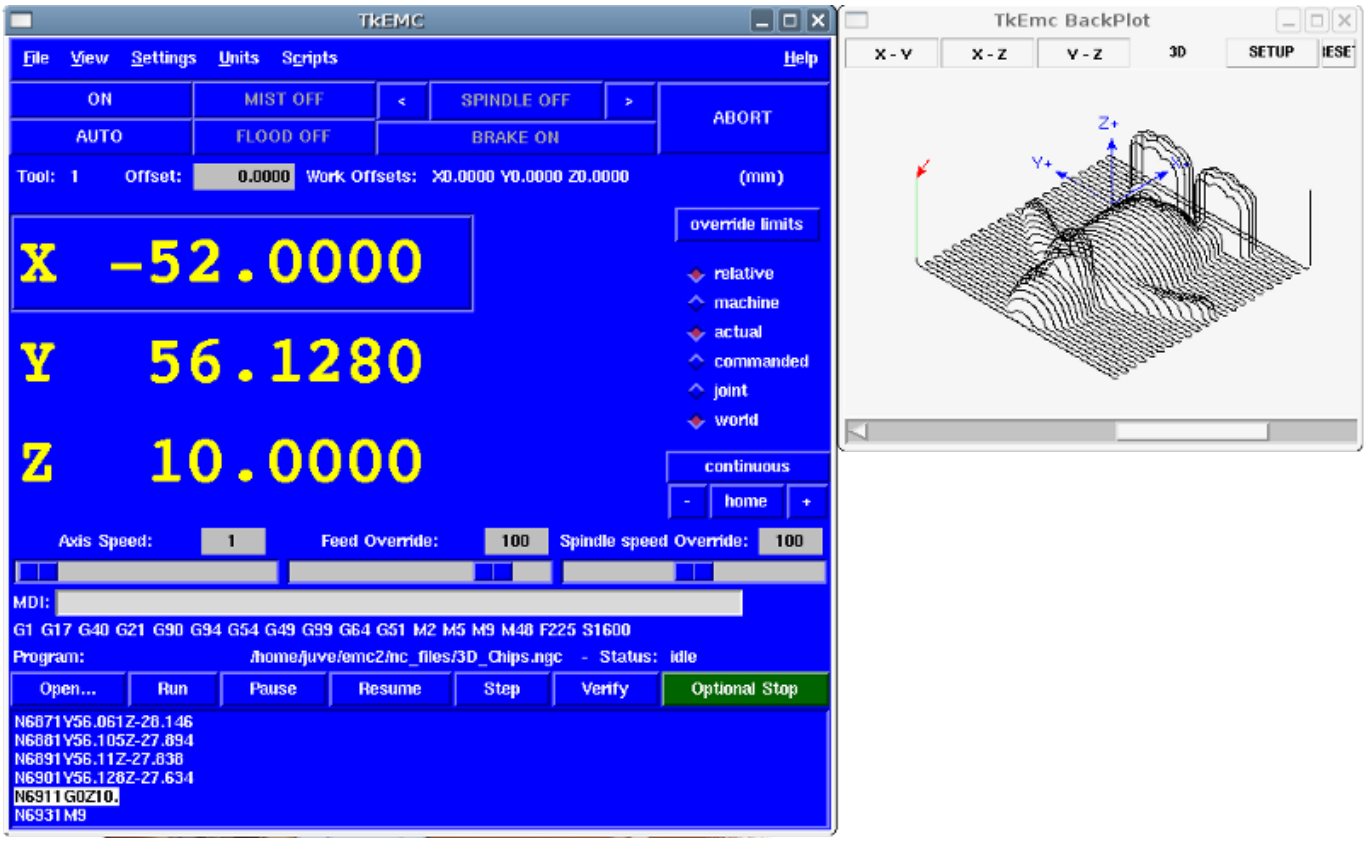



Figure 10.39: TkLinuxCNC Window

### 10.7.2 Начало

To select TkLinuxCNC as the front-end for LinuxCNC, edit the INI file. In the section *[DISPLAY]* change the *DISPLAY* line to read

```
DISPLAY = tklinuxcnc
```

Then, start LinuxCNC and select that INI file. The sample configuration *sim/tklinuxcnc/tklinuxcnc.ini* is already configured to use TkLinuxCNC as its front-end.

When LinuxCNC is launched the TKLinuxCNC window is opened.

#### 10.7.2.1 A typical session with TkLinuxCNC

1. Start LinuxCNC and select a configuration file.
2. Clear the *E-STOP* condition and turn the machine on (by pressing F1 then F2).

3. *Home* each axis.
4. Load the file to be milled.
5. Put the stock to be milled on the table.
6. Set the proper offsets for each axis by jogging and either homing again or right-clicking an axis name and entering an offset value. [1]
7. Запустите программу.
8. To mill the same file again, return to step 6. To mill a different file, return to step 4. When you're done, exit LinuxCNC.

### 10.7.3 Elements of the TkLinuxCNC window

The TkLinuxCNC window contains the following elements:

- A menubar that allows you to perform various actions
- A set of buttons that allow you to change the current working mode, start/stop spindle and other relevant I/O
- Status bar for various offset related displays
- Coordinate display area
- A set of sliders which control *Jogging speed*, *Feed Override*, and *Spindle speed Override* which allow you to increase or decrease those settings
- Manual data input text box *MDI*
- Status bar display with active G-codes, M-codes, F- and S-words
- Interpreter related buttons
- A text display area that shows the G-code source of the loaded file

#### 10.7.3.1 Main buttons

From left to right, the buttons are:

- Machine enable: *ESTOP > ESTOP RESET > ON*
- Toggle mist coolant
- Decrease spindle speed
- Set spindle direction *SPINDLE OFF > SPINDLE FORWARD . SPINDLE REVERSE*
- Increase spindle speed
- Прерывание

then on the second line:

- Operation mode: *MANUAL > MDI > AUTO*
- Toggle flood coolant
- Toggle spindle brake control

[1] For some of these actions it might be necessary to change the mode LinuxCNC is currently running in.

#### 10.7.3.2 Offset display status bar

The Offset display status bar displays the currently selected tool (selected with Txx M6), the tool length offset (if active), and the work offsets (set by right-clicking the coordinates).

#### 10.7.3.3 Coordinate Display Area

The main part of the display shows the current position of the tool. The color of the position readout depends on the state of the axis. If the axis is unhomed the axis will be displayed in yellow letters. Once homed it will be displayed in green letters. If there is an error with the current axis TkLinuxCNC will use red letter to show that. (for example if an hardware limit switch is tripped).

To properly interpret these numbers, refer to the radio boxes on the right. If the position is *Machine*, then the displayed number is in the machine coordinate system. If it is *Relative*, then the displayed number is in the offset coordinate system. Further down the choices can be *actual* or *commanded*. Actual refers to the feedback coming from encoders (if you have a servo machine), and the *commanded* refers to the position command send out to the motors. These values can differ for several reasons: Following error, deadband, encoder resolution, or step size. For instance, if you command a movement to X 0.0033 on your mill, but one step of your stepper motor is 0.00125, then the *Commanded* position will be 0.0033 but the *Actual* position will be 0.0025 (2 steps) or 0.00375 (3 steps).

Another set of radio buttons allows you to choose between *joint* and *world* view. These make little sense on a normal type of machine (e.g. trivial kinematics), but help on machines with non-trivial kinematics like robots or stewart platforms. (you can read more about kinematics in the Integrator Manual).

**Траектория** When the machine moves, it leaves a trail called the backplot. You can start the backplot window by selecting View→Backplot.

#### 10.7.3.4 TkLinuxCNC Interpreter / Automatic Program Control

TkLinuxCNC Interpreter / program control

**Control Buttons** The buttons in the lower part of TkLinuxCNC are used to control the execution of a program:

+ * *Open* to load a program, * *Verify* to check it for errors, * *Run* to start the actual cutting, * *Pause* to stop it while running, * *Resume* to resume an already paused program, * *Step* to advance one line in the program and * *Optional Stop* to toggle the optional stop switch (if the button is green the program execution will be stopped on any M1 encountered).

**Text Program Display Area** When the program is running, the line currently being executed is highlighted in white. The text display will automatically scroll to show the current line.

#### 10.7.3.5 Ручное управление

**Implicit keys** TkLinuxCNC allows you to manually move the machine. This action is known as *jogging*. First, select the axis to be moved by clicking it. Then, click and hold the + or - button depending on the desired direction of motion. The first four axes can also be moved by the keyboard arrow keys (X and Y), the PAGE UP and PAGE DOWN keys (Z) and the *[* and *]* keys (A/4th).

+ If *Continuous* is selected, the motion will continue as long as the button or key is pressed. If another value is selected, the machine will move exactly the displayed distance each time the button is clicked or the key is pressed. The available values are:

+

```
1.0000, 0.1000, 0.0100, 0.0010, 0.0001
```

+ By pressing *Home* or the HOME key, the selected axis will be homed. Depending on your configuration, this may just set the axis value to be the absolute position 0.0, or it may make the machine move to a specific home location through use of *home switches*. See the Homing Chapter for more information.

+ By pressing *Override Limits*, the machine will temporarily be permitted to jog outside the limits defined in the INI file. (Note: if *Override Limits* is active the button will be displayed using a red color).

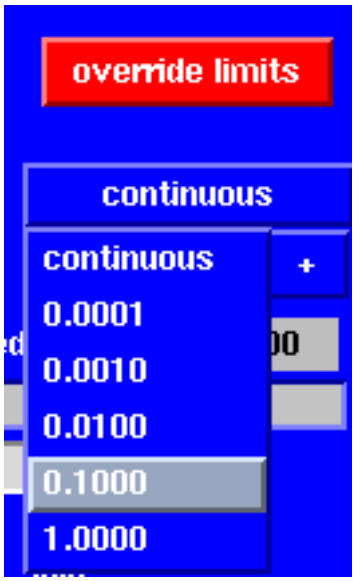


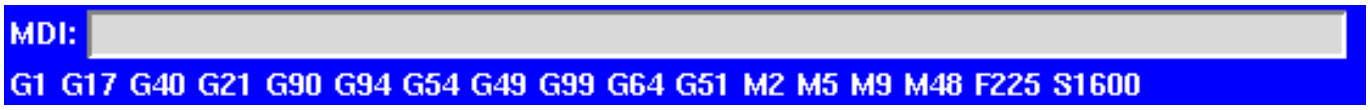

Figure 10.40: TkLinuxCNC Override Limits & Jogging increments example

**Группа шпинделя** The button on the first row selects the direction for the spindle to rotate: Counterclockwise, Stopped, Clockwise. The buttons next to it allow the user to increase or decrease the rotation speed. The button on the second row allows the spindle brake to be engaged or released. Depending on your machine configuration, not all the items in this group may have an effect.

**Группа СОЖ** The two buttons allow the *Mist* and *Flood* coolants to be turned on and off. Depending on your machine configuration, not all the items in this group may appear.

#### 10.7.3.6 Code Entry

Manual Data Input (also called MDI), allows G-code programs to be entered manually, one line at a time. When the machine is not turned on, and not set to MDI mode, the code entry controls are unavailable.




This allows you to enter a G-code command to be executed. Execute the command by pressing Enter.

**Active 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 Скорость медленной подачи

By moving this slider, the speed of jogs can be modified. The numbers above refer to axis units / second. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

#### 10.7.3.8 Переопределение подачи

By moving this slider, the programmed feed rate can be modified. For instance, if a program requests *F60* and the slider is set to 120%, then the resulting feed rate will be 72. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

#### 10.7.3.9 Spindle speed Override

The spindle speed override slider works exactly like the feed override slider, but it controls to the spindle speed. If a program requested S500 (spindle speed 500 RPM), and the slider is set to 80%, then the resulting spindle speed will be 400 RPM. This slider has a minimum and maximum value defined in the INI file. If those are missing the slider is stuck at 100%. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

### 10.7.4 Управление клавиатурой

Almost all actions in TkLinuxCNC can be accomplished with the keyboard. Many of the shortcuts are unavailable when in MDI mode.

The most frequently used keyboard shortcuts are shown in the following table.

Table 10.7: Most Common Keyboard Shortcuts

| **Нажатие клавиши** | **Действия** |
|---|---|
| F1 | Переключает аварийный останов |
| F2 | Станок вкл/выкл |
| `, 1 .. 9, 0 | Устанавливает переопределение подачи от 0% до 100% |
| X, ` | Активирует первую ось |
| Y, 1 | Активирует вторую ось |
| Z, 2 | Активирует третью ось |
| A, 3 | Активирует четвертую ось |
| Home | Send active axis Home |
| Left, Right | Медленная подача первой оси |
| Up, Down | Медленная подача второй оси |
| Pg Up, Pg Dn | Медленная подача третьей оси |
| [, ] | Медленная подача четвертой оси |
| ESC | Остановить исполнение |

# 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 Введение

The development branch version of QtPlasmaC is a GUI for plasma cutting which utilises the plasmac component for controlling a plasma table using the master branch (development) version of LinuxCNC (v2.10) using the Debian Bullseye or later 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 development branch version of the QtPlasmaC GUI will run on any hardware that is supported by the master branch version of LinuxCNC (v2.10) 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:

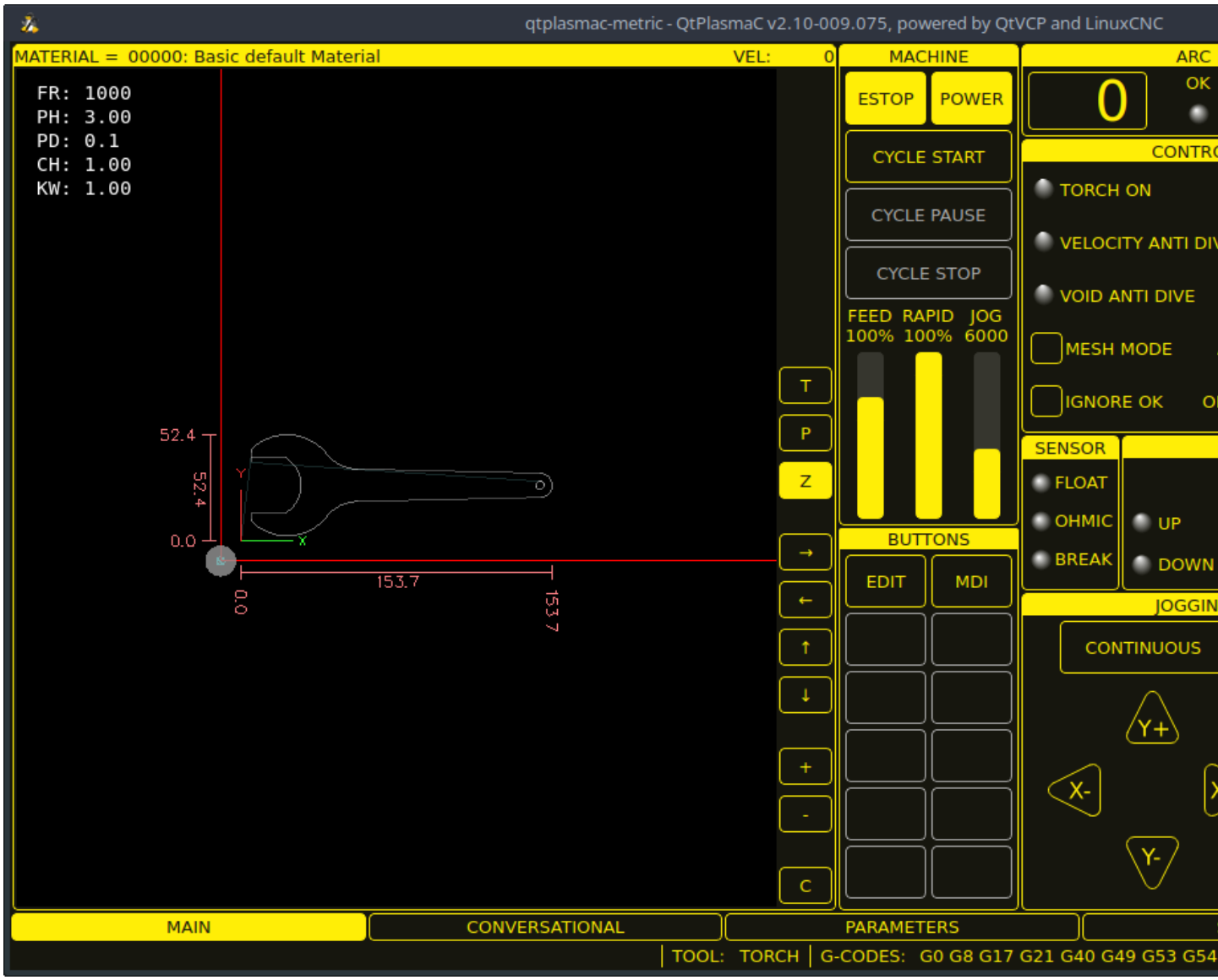
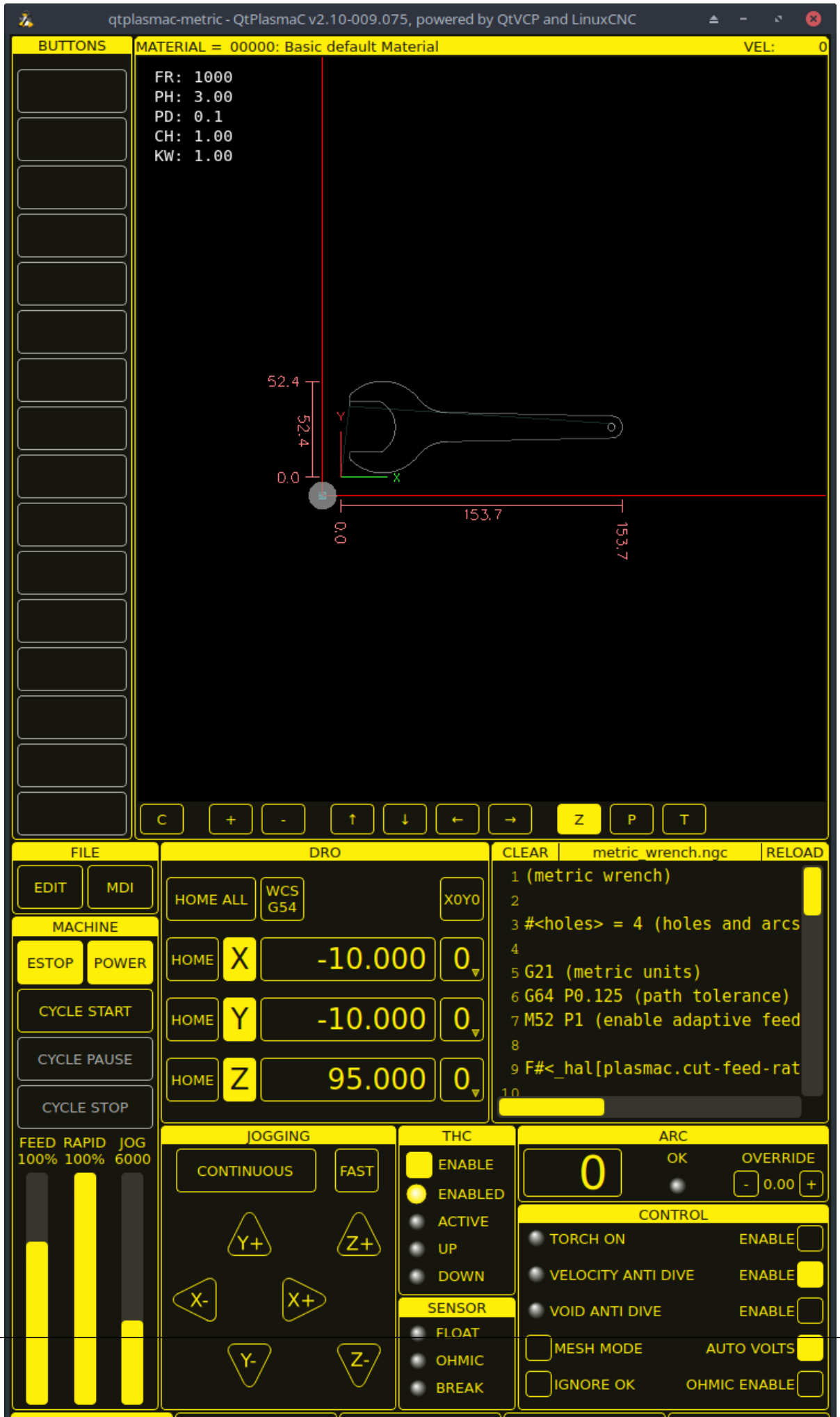



Figure 10.41: **16:9**


qtplasmac-metric - QtPlasmaC v2.10-009.075, powered by QtVCP and LinuxCNC
BUTTONS
MATERIAL = 00000: Basic default Material
VEL: 0
FR: 1000
PH: 3.00
PD: 0.1
CH: 1.00
KW: 1.00
52.4
52.4
0.0
153.7
0.0
153.7
C
+
-
Z
P
T
FILE
EDIT
MDI
MACHINE
ESTOP
POWER
CYCLE START
CYCLE PAUSE
CYCLE STOP
DRO
HOME ALL
WCS G54
X0Y0
HOME
X
-10.000
HOME
Y
-10.000
HOME
Z
95.000
CLEAR
metric_wrench.ngc
RELOAD
1 (metric wrench)
3 #<holes> = 4 (holes and arcs
5 G21 (metric units)
6 G64 P0.125 (path tolerance)
7 M52 P1 (enable adaptive feed
9 F#<_hal[plasmac.cut-feed-rat
FEED 100%
RAPID 100%
JOG 6000
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

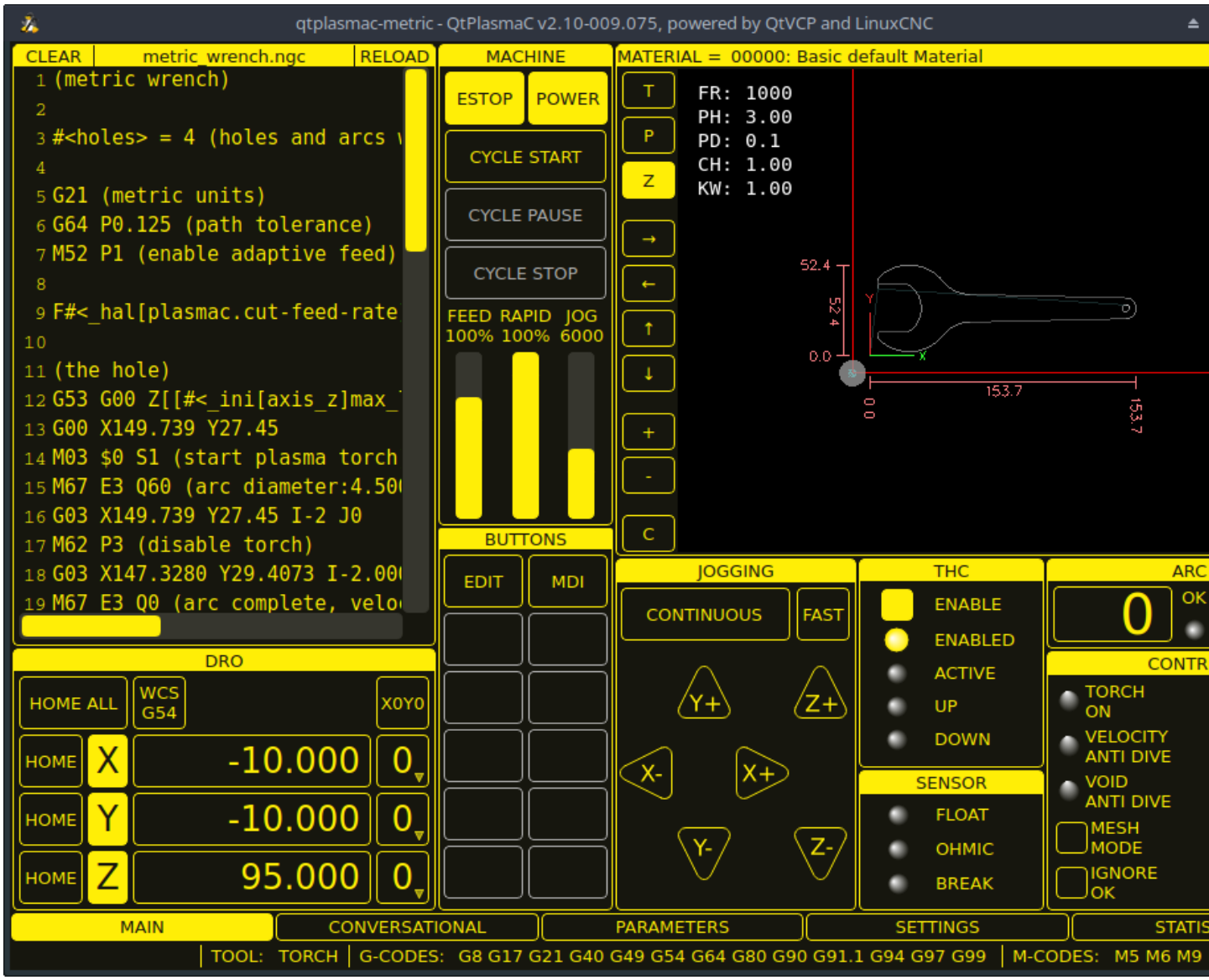



Figure 10.43: **4:3**

### 10.8.4 Установка LinuxCNC

The preferred method for installing LinuxCNC is via an ISO image as described below.

---

**Note**
It is possible to install and run LinuxCNC on a variety of Linux distributions however that is beyond the scope of this User Guide. If the user wishes to install a Linux distribution other than those recommended, they will first need to install their preferred Linux distribution and then install the master branch version of LinuxCNC (v2.10) along with any required dependencies. It should also be noted that Bullseye is the earliest Debian distribution that is supported by the master branch of LinuxCNC (v2.10). Buster is no longer supported.

---

#### 10.8.4.1 If The User Does Not Have Linux Installed

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). The user will then have to follow the appropriate instructions to upgrade to the master branch version of LinuxCNC (v2.10).

#### 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 12 (Bookworm) or Debian 11 (Bullseye)

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 master branch (v2.10) Bookworm repository.

```
deb     http://buildbot2.highlab.com/ bookworm master-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 Creating A QtPlasmaC Configuration

Prior to creating a QtPlasmaC configuration, it is important that the user has a firm understanding of the operating modes available, as well as the I/O's that are required for successful plasma operation.

#### 10.8.5.1 Modes

QtPlasmaC requires the selection of one of following three operating modes:

| Mode | Description |
|---|---|
| 0 | Uses an external arc voltage input to calculate both Arc Voltage (for Torch Height Control) and Arc OK. |
| 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. |

**Important**
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.

**Note**
For fine tuning of Mode 0 Ark OK see Tuning Mode 0 Arc OK in the Advanced Topics section of the manual.

#### 10.8.5.2 Available I/Os

**Note**
This section only touches on the hardware I/O's required for QtPlasmaC. Base machine requirements such as limit switches, home switches, etc. are in addition to these.

| Имя | Modes | Description |
|---|---|---|
| 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. |
| Arc 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. |

| Имя | Modes | Description |
|---|---|---|
| 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**.

---

**Note**
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 Diagrams 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.

The machine specific settings are not described here, refer to the documentation for the particular configuration wizard that is being used.

There are LinuxCNC forum sections available for these wizards:

StepConf Wizard

PnCconf Wizard

Fill in the required entries to suit the machine wiring/breakout board configuration.

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.

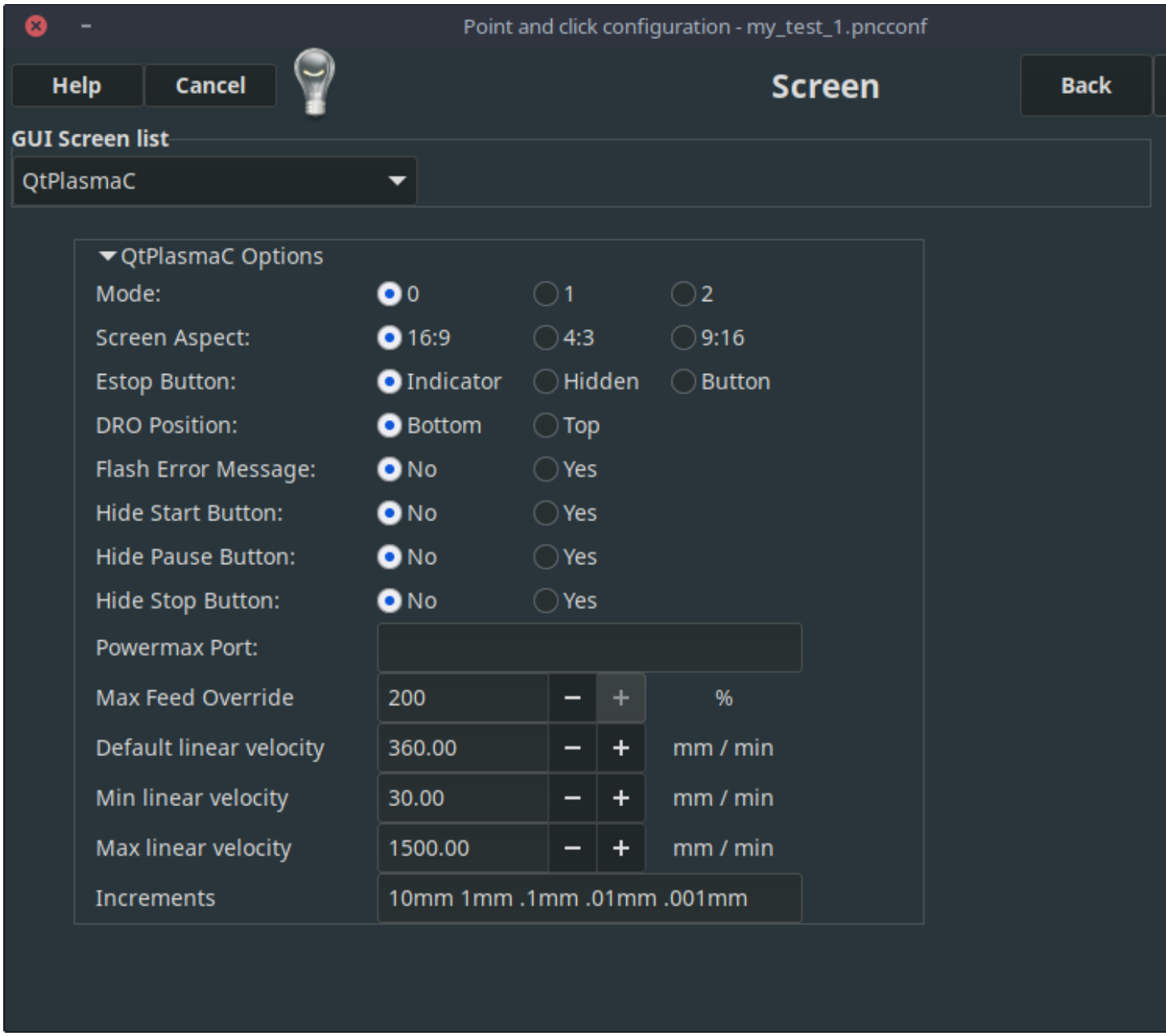



Figure 10.44: PnCConf QtPlasmaC Options

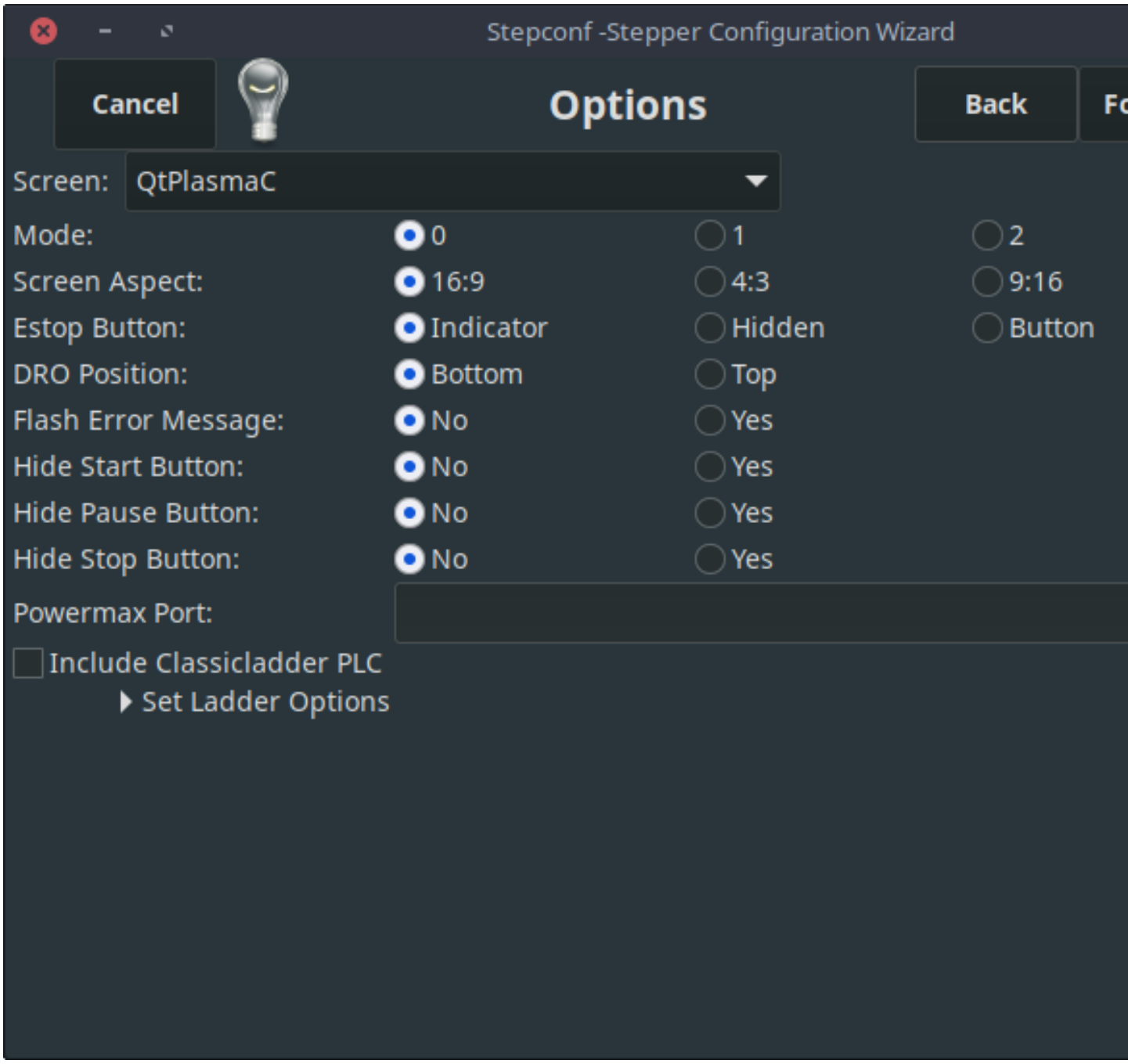



Figure 10.45: StepConf QtPlasmaC Options

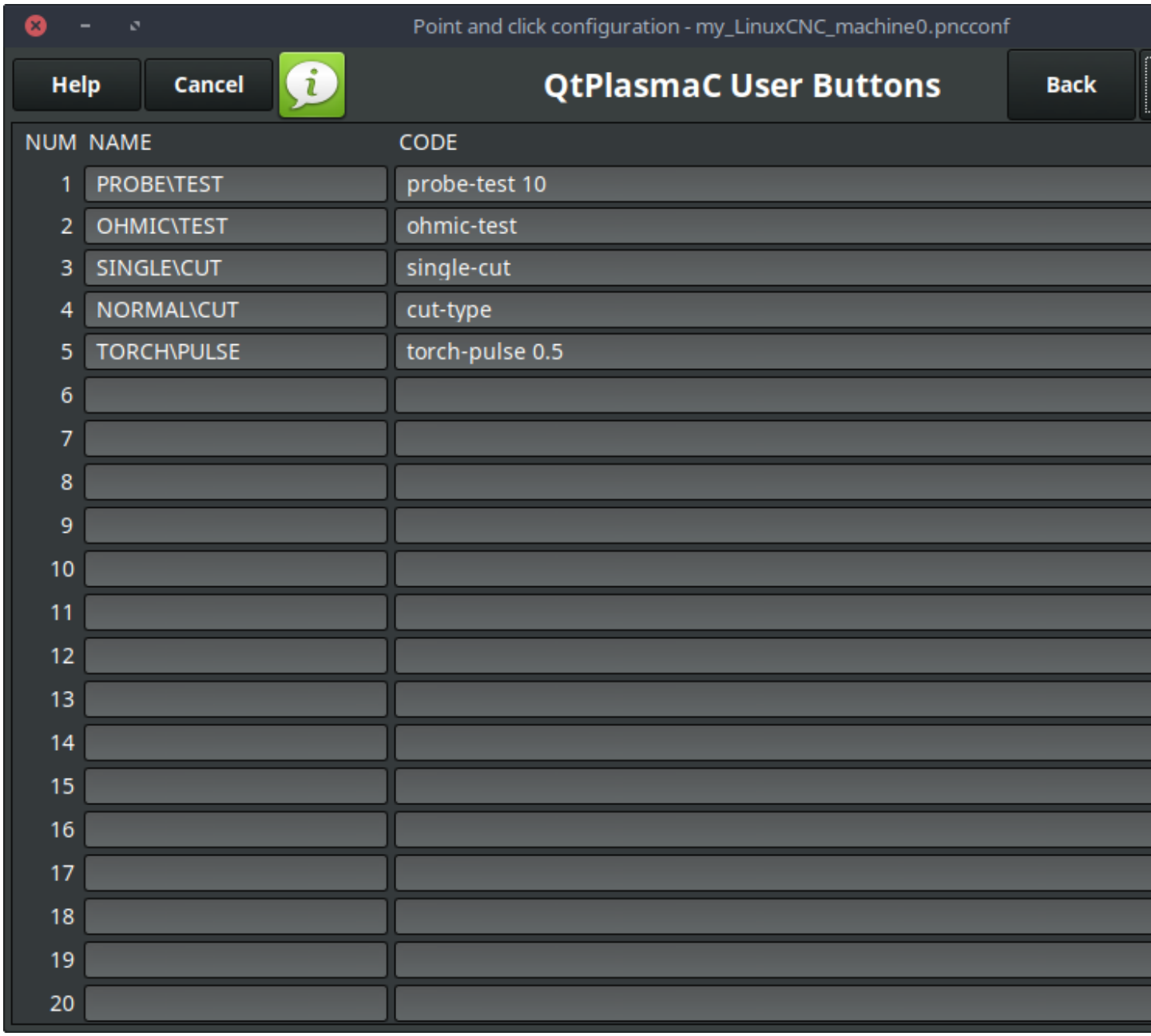



Figure 10.46: QtPlasmaC User Buttons

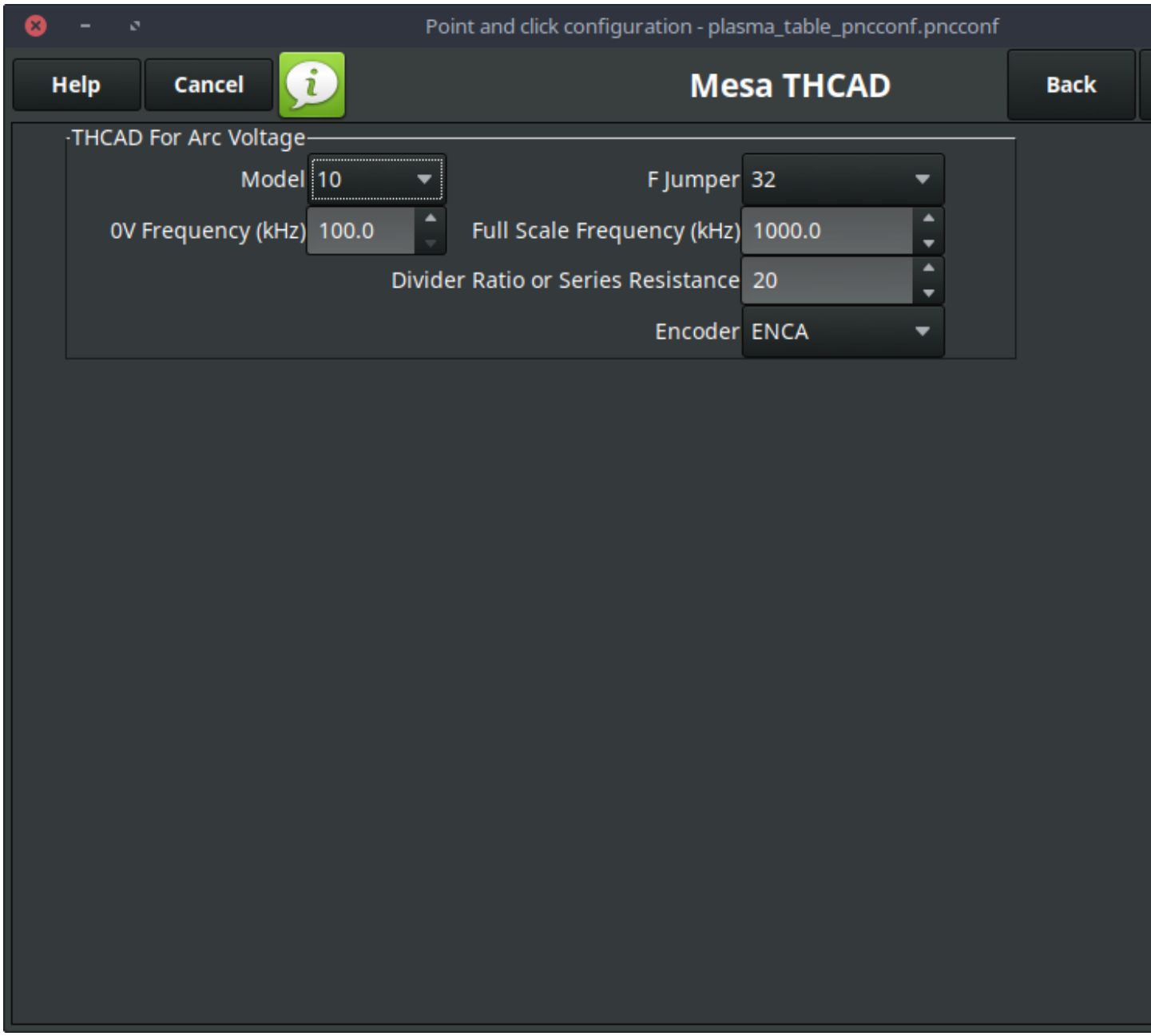



Figure 10.47: QtPlasmaC THCAD

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 on the way LinuxCNC was installed:

For a package installation (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
```

After running the above command LinuxCNC should be running with the QtPlasmaC GUI visible.

**Important**
BEFORE PROCEEDING, THE USER SHOULD BE ABLE TO HOME THE MACHINE, ZERO EACH AXIS, JOG ALL AXES TO SOFT LIMITS WITHOUT CRASHING, AND RUN TEST G-CODE PROGRAMS WITHOUT ANY ERRORS.

ONLY WHEN this criteria is met should the user proceed with the QtPlasmaC initial setup.

**Note**
It is possible to create a sim configuration using StepConf but it is not possible to have tandem joints in the sim configuration.

#### 10.8.5.5 Qt Dependency Errors

If any Qt dependency errors are encountered while attempting to run the QtPlasmaC configuration, the user may need to run the QtVCP installation script to resolve these issues.

For a package installation (Buildbot) enter the following command in a terminal window:

```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

For a run in place installation enter the following command in a terminal window:

```
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
```

#### 10.8.5.6 Initial Setup

The following heights diagrams will help the user visualize the different heights involved in plasma cutting and how they are measured. There are two different scenarios based on if the user chooses to use **Probe Height** only, or if the user chooses to use **Slat Height** AND **Material Thickness**.

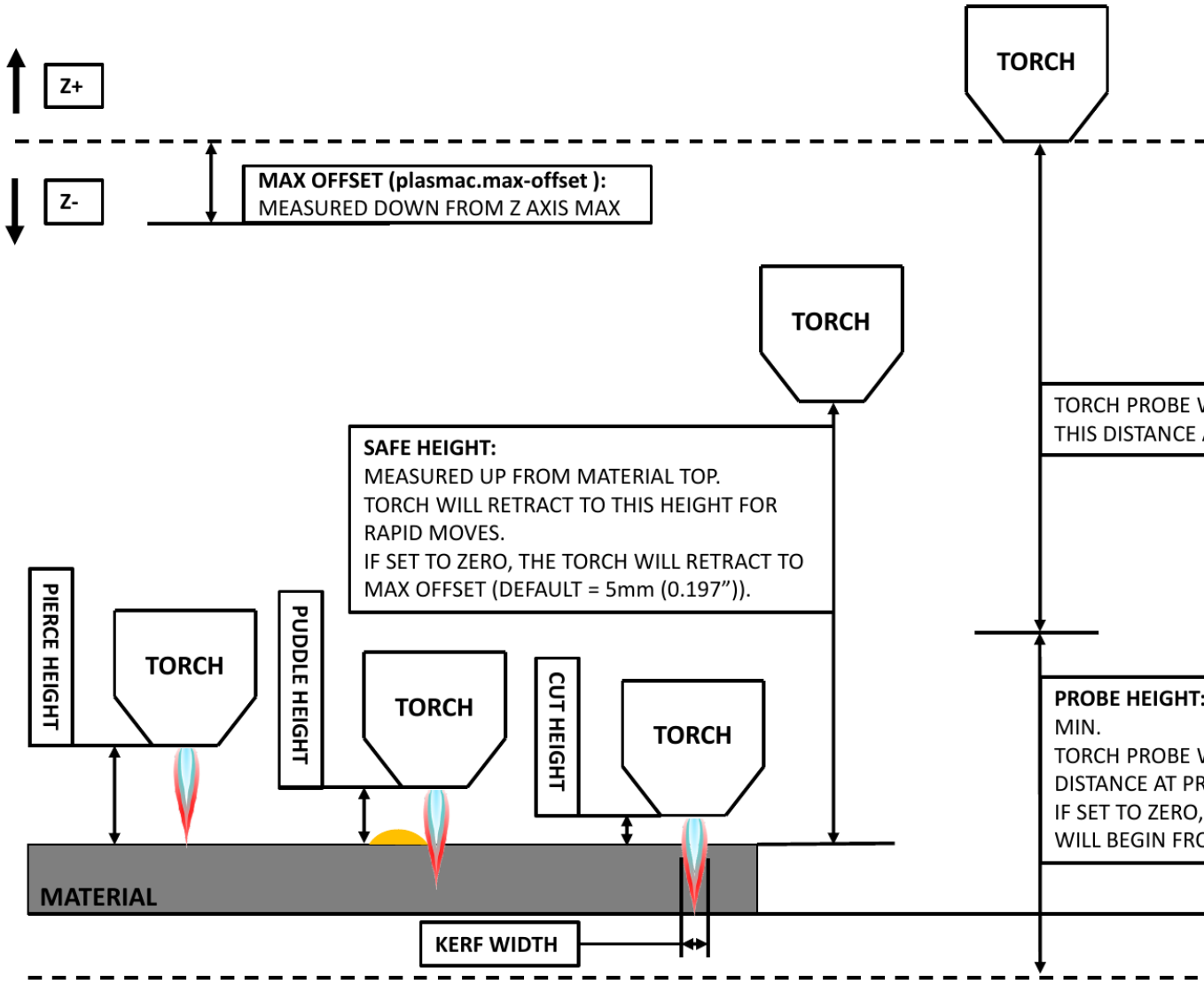



Figure 10.48: **Probe Height Only**

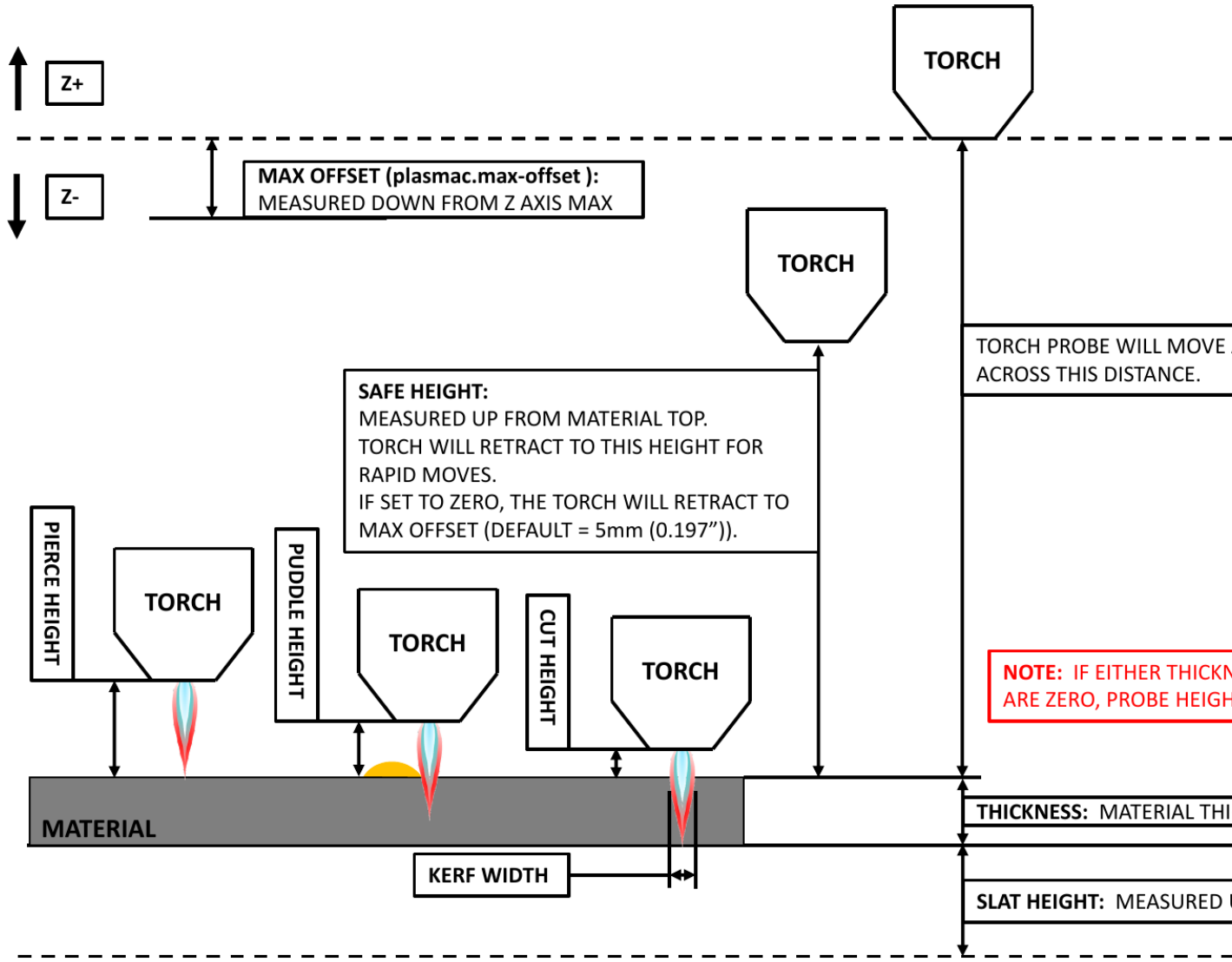



Figure 10.49: **Slat Height and Material Thickness**

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.

**Note**
The user should be familiar with the recommended Z Axis Settings.

1. Home the Z axis.

2. 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 step only serves to allow the user to more easily visualize and adjust **Probe Height** this value is measured from the Z axis MINIMUM_LIMIT up.
3. Home the Z axis again.

**Probe Test**

If the machine is equipped with a float switch, then the user will need to set the offset in the **CONFIGURATION** section of the **PARAMETERS** tab. This will be done by running a ”Probe Test” cycle.

1. Check that the Probe Speed and the Probe Height in the **CONFIGURATION** section of the **PARAMETERS** tab are correct. QtPlasmaC can probe at the full Z axis velocity so long as the machine has enough movement in the float switch to absorb any overrun. If the machine is suitable, the user could set the Probe Height to a value near the Z axis minimum and do all probing at full speed.
2. If the machine is not already homed and in the home position, home the machine.
3. Place some material on the slats under the torch.
4. Press the **PROBE TEST** button.
5. The Z axis will probe down, find the material then move up to the specified **Pierce Height** as set by the currently selected material. The torch will wait in this position for the time set in the <*machine_name*>.prefs file. The default probe test hold time is 10 seconds, this value may be edited in the <*machine_name*>.prefs file. After this the torch will return to the starting height.
6. Measure the distance between the material and the tip of the torch while the torch is waiting at **Pierce Height**.
7. If the measurement is greater than the **Pierce Height** of the currently selected material, then reduce the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the measured value and the specified value. If the measurement is less than **Pierce Height** of the currently selected material, then increase the ”Float Travel” in the **CONFIGURATION** section of the **PARAMETERS** tab by the difference between the specified value and the measured value.
8. After the adjustments to the ”Float Travel” have been made, repeat the process from #4 above until the measured distance between the material and the torch tip matches the **Pierce Height** of the currently selected material.
9. If the table has a laser or camera for sheet alignment, a scribe, or uses offset probing then the required offsets need to be applied by following the procedure described in Peripheral Offsets. The LASER and/or CAMERA buttons will not be visible until the user has set the appropriate offset(s) and they are recorded in the <*machine_name*>.prefs file.
10. CONGRATULATIONS! The user should now have a working QtPlasmaC Configuration.

---

**Note**
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.

---

---

**Important**
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**.

---

**Warning**
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)

Automated migration to QtPlasmaC from PlasmaC is no longer supported. The user will either need to convert the PlasmaC configuration manually, or create a new configuration using the configuration wizard.

### 10.8.7 Other QtPlasmaC Setup Considerations

#### 10.8.7.1 Фильтр нижних частот

The plasmac HAL component has a built in low-pass 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 low-pass 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 low-pass 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 low-pass 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).

Например:

```
setp plasmac.lowpass-frequency 100
```

В приведенном выше примере частота среза составит 100 Гц.

#### 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 over-written 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)[2].

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 = calculated resistance

[2] In the US, the letter V is commonly used as a symbol (Voltage) and as a unit (Volt).

- $U_s$ = 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 = power
- $U_s$ = 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 Launcher

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.

After the launcher appears on the desktop, right click on it and then edit it with the user's editor of choice. Edit the file so it looks similar to:

```
[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
```

If the user would like a terminal window to open behind the GUI window, then change the Terminal line to:

```
Terminal=true
```

Displaying a terminal can be handy for error and information messages.

#### 10.8.7.5 QtPlasmaC Files

After a successful QtPlasmaC installation, the following files are created in the configuration directory:

| **Filename** | **Function** |
|---|---|
| <*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. |

| Filename | Function |
|---|---|
| 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. |

**Note**
<*machine_name*> is whatever name the user entered into the ”Machine Name” field of the configuration wizard program.

**Note**
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 | Function |
|---|---|
| <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 style sheet for the currently loaded session of QtPlasmaC. |

**Note**
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.

**Note**
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]** Раздел

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
```

**Note**
for a imperial config replace G21 above with G20.

**Note**
both the above paths show the minimum requirements.

**Important**
SEE PATH TOLERANCE FOR RS274NGC_STARTUP_CODE INFORMATION RELATED TO G64.

**[HAL]** Раздел

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)
```

**Note**
The user could place custom HAL commands in the custom.hal file as this file is not overwritten by QtPlasmaC updates.

**[DISPLAY]** Раздел

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]** Раздел

This variable is mandatory.

```
SPINDLES = 3
```

**[AXIS_X]** Раздел

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]** Раздел

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]** Раздел

These variables are mandatory.

```
MIN_LIMIT        = just below the top of the table's slats
MAX_VELOCITY     = double the value in the corresponding joint
MAX_ACCELERATION = double the value in the corresponding joint
OFFSET_AV_RATIO  = 0.5
```

**Note**
With the exception of tube cutting with an angular A, B, or C axis, QtPlasmaC uses the LinuxCNC External Offsets feature for all Z axis motion, and for moving the X and/or Y axis for a consumable change or a cut recovery while paused. For more information on this feature, please read External Axis Offsets in the LinuxCNC documentation.

## 10.8.8 QtPlasmaC GUI Overview

The following sections will give a general overview of the QtPlasmaC layout.

### 10.8.8.1 Exiting QtPlasmaC

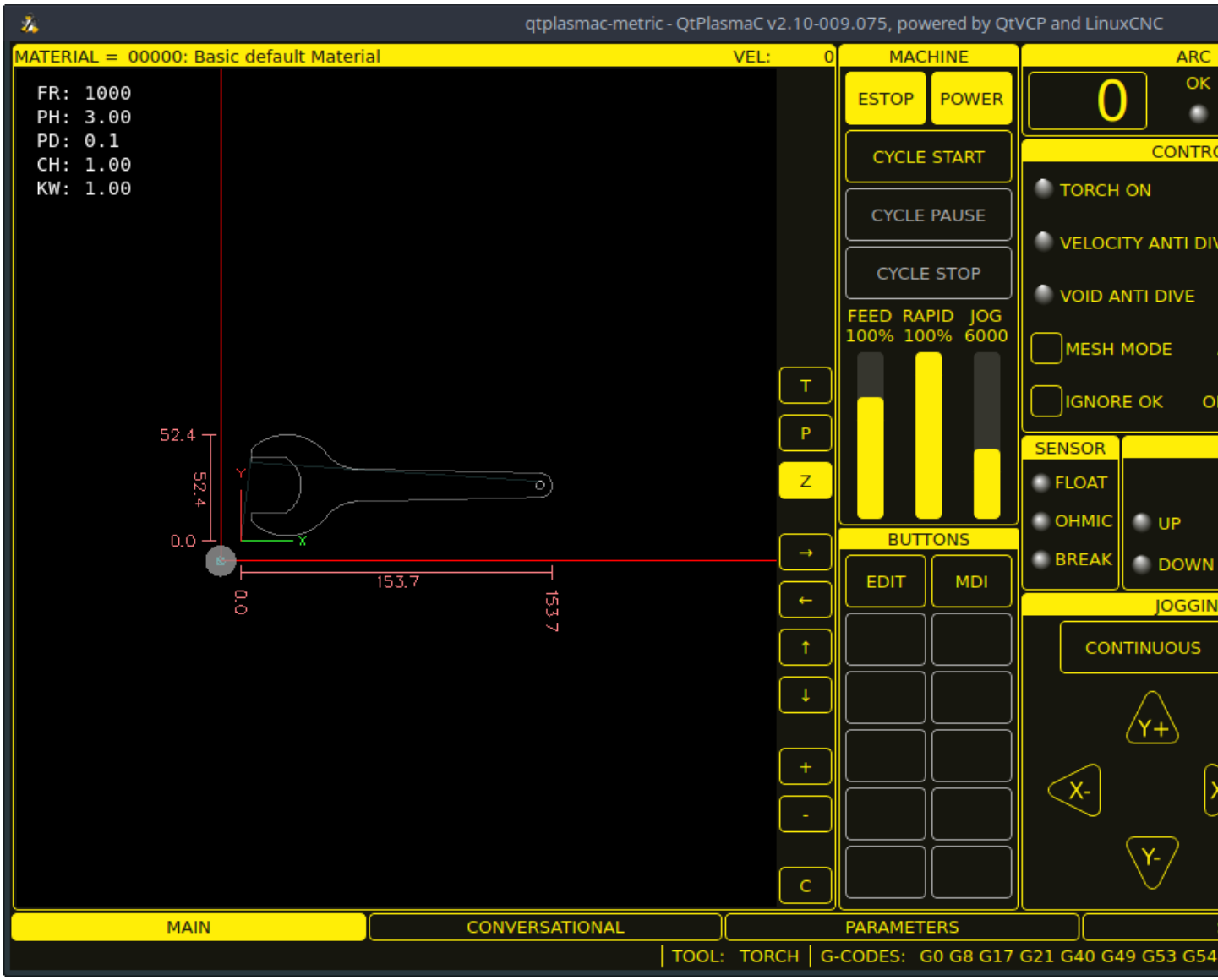
Exiting or shutting down QtPlasmaC is done by either:

1. Click the window shutdown button on the window title bar
2. Long press the **POWER** button on the MAIN Tab.

A shutdown warning can be displayed on every shutdown by checking the **Exit Warning** checkbox on the SETTINGS Tab.

### 10.8.8.2 MAIN Tab

Screenshot example of the QtPlasmaC MAIN Tab in **16:9** aspect ratio:




Some functions/features are only used for particular modes and are not displayed if they are not required by the chosen QtPlasmaC mode.

Table 10.12: Features of the **PREVIEW WINDOW**

| **Имя** | **Description** |
|---|---|
| Material | The top header is clickable in this area to reveal a drop-down menu. It 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. |
| VEL: | Displays the current feed rate of the table, including both rapid and cutting moves. While cutting, if a Velocity Reduction is active, this label updates to reflect the percentage of the original feed rate being used. For example, ”VEL@20%:” means the table is cutting at 20% of the programmed feed rate - an 80% reduction. |
| FR: | If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Feed Rate. |
| PH: | If ”View Material” is selected on the SETTINGS Tab, this displays the currently selected material’s Pierce Height. |

Table 10.12: (continued)

| **Имя** | **Description** |
|---|---|
| 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 | This button changes the preview to a top down full table view. |
| P | This button changes the preview to an isometric view. |
| Z | This button changes the preview to a top down view. |
| → | This button pans the preview right. |
| ← | This button pans the preview left. |
| ↑ | This button pans the preview up. |
| ↓ | This button pans the preview down. |
| + | This button zooms the preview. |
| - | This button zooms the preview. |
| C | This button clears the live plot. |

Table 10.13: **MACHINE** representation

| **Имя** | **Description** |
|---|---|
| ESTOP | 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 | This button turns the GUI on and allows QtPlasmaC/LinuxCNC to control the hardware.<br>Pressing and holding the **POWER** button for longer than two seconds will bring up a dialog to exit the QtPlasmaC application. |
| CYCLE START | This button starts the cycle for any loaded G-code file. |
| CYCLE PAUSE | This button pauses the cycle for any loaded G-code file.<br>If a cycle is paused, this button will display **CYCLE RESUME** and flash.<br>Pressing **CYCLE RESUME** will resume the cycle. |
| 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 |
| ПОДАЧА | 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%. |

Table 10.13: (continued)

| Имя | Description |
|---|---|
| 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.

| Имя | Description |
|---|---|
| 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. |

Table 10.15: **ARC**

| Имя | Modes | Description |
|---|---|---|
| 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. |

Table 10.16: **CONTROL**

| Имя | Modes | Description |
|---|---|---|
| TORCH ON | 0, 1, 2 | Indicates the status of the Torch On output signal. |

Table 10.16: (continued)

| Имя | Modes | Description |
|---|---|---|
| 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”. If the user has a laser installed, then it is also possible to dry run with the laser. See the LASER section for detailed instructions. |
| 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 | This box enables or disables the ohmic probe input.<br>If the Ohmic Probe input is disabled, the Ohmic Probe LED will still show the status of the probe input, but the Ohmic Probe results will be ignored. |
| RS485 | 0, 1, 2 | This box will enable or disable the communications to a PowerMax. This button is only visible if a PM_PORT option is configured in the `[POWERMAX]` section of the *<machine_name>*.prefs file. |

Table 10.16: (continued)

| **Имя** | **Modes** | **Description** |
|---|---|---|
| Status | 0, 1, 2 | When PowerMax communications are enabled, this will display one of the following: **CONNECTING**, **CONNECTED**, **COMMS ERROR**, or a **Fault Code**. For more information, see the PowerMax Communications section. |

Table 10.17: **SENSOR**

| **Имя** | **Description** |
|---|---|
| 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. |

Table 10.18: **THC**

| **Имя** | **Description** |
|---|---|
| ENABLE | 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. |

**Note**
During Paused Motion, this section will become CUT RECOVERY

| **Имя** | **Description** |
|---|---|
| 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. |

**Note**
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.

| Имя | Description |
|---|---|
| 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. |
| ПОДАЧА | 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. |

Table 10.21: **G-CODE WINDOW**

| Имя | Description |
|---|---|
| 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.<br>Previous error messages, and the error status will be cleared.<br>Cut type will be set to NORMAL CUT. |
| OPEN | This button will open a FILE OPEN panel over the PREVIEW WINDOW. |
| RELOAD | This button will reload the currently loaded G-code File. |

Table 10.22: **DRO**

| Имя | Description |
|---|---|
| HOME ALL | This button will home all of the axes in the order set by HOME_SEQUENCE in the <*machine_name*>.ini file. |
| WCS G54 | This drop-down button will change the current work offset. |
| CAMERA | This button will display a CAMVIEW panel on top of the PREVIEW WINDOW and will allow the user to set an origin with or without rotation. See the CAMERA section for detailed instructions. This button will not be visible until a CAMERA offset is set in the <*machine_name*>.prefs file. |
| 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. This button will not be visible until a LASER offset is set in the <*machine_name*>.prefs file. |
| X0 Y0 | This button will set the current position to X0 Y0. |
| HOME [AXIS] | This button will home the corresponding axis. |

Table 10.22: (continued)

| **Имя** | **Description** |
|---|---|
| 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.

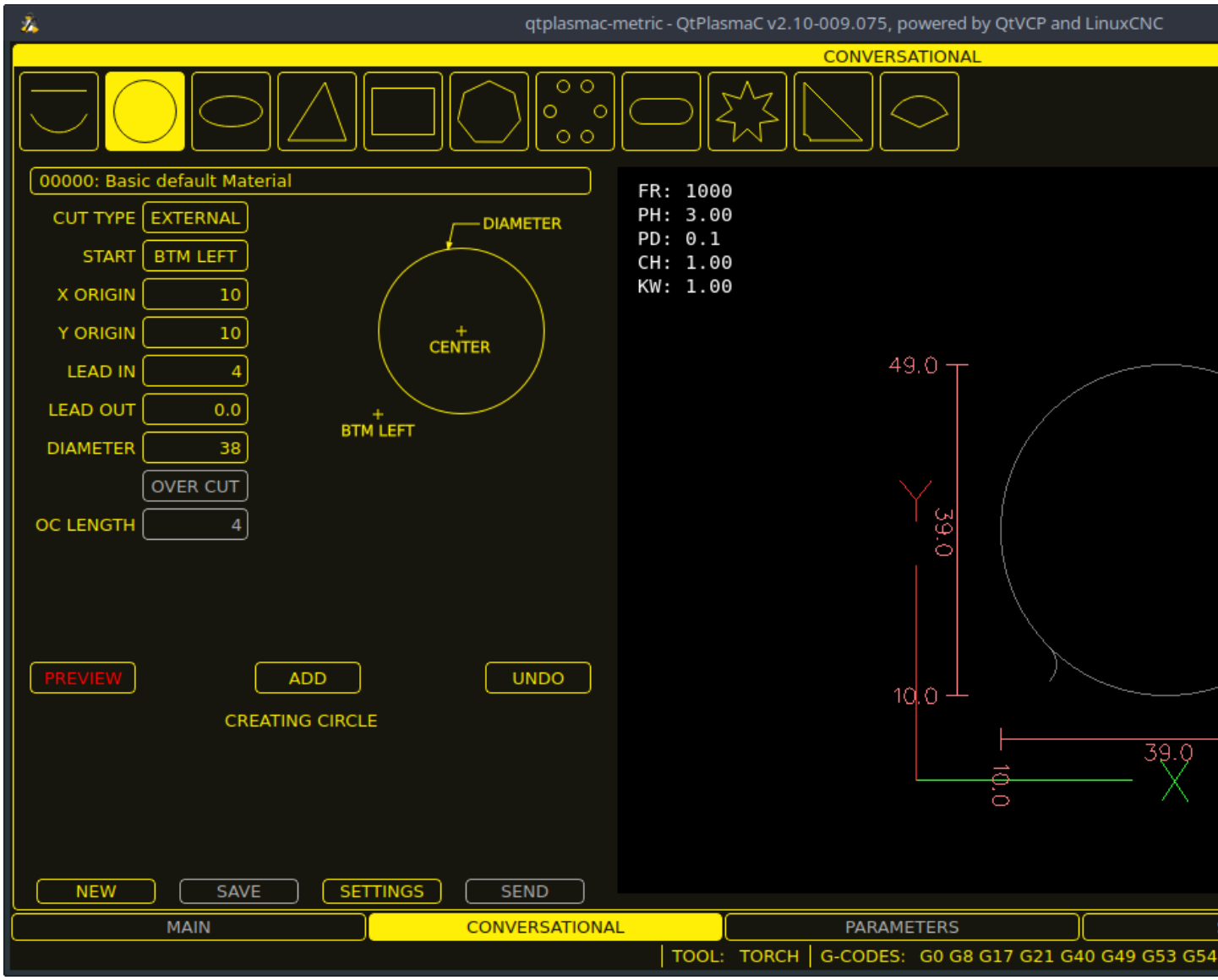
When a G-code file is loaded, the display will change to the selected view.

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:

Some functions/features are only used for particular modes and are not displayed if they are not required by the chosen QtPlasmaC mode.

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
```

Table 10.23: **CONFIGURATION - ARC**

| **Имя** | **Modes** | **Description** |
|---|---|---|
| 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. |

Table 10.23: (continued)

| Имя | Modes | Description |
|---|---|---|
| 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. |

**Note**
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.

Table 10.24: **CONFIGURATION - PROBING**

| Имя | Description |
|---|---|
| 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. |
| Probe Speed | 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. If set to zero, then the torch will move at Probe Speed from the current position. This may be adventageous when using Slat Height and Material Thickness as the machine will default to Probe Height if either Slat Height or Material Thickness are zero. Refer to the Heights Diagrams for a visual representation. |
| Slat Height | This sets the height of the slats, measured up from Z axis minimum limit. This must be used in conjunction with Material Thickness. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation. |
| 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. |
| Offset Speed | This sets the speed at which the probe will move to the offset position in the X axis and Y axis. |

**Note**
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.

Table 10.25: **CONFIGURATION - SAFETY**

| **Имя** | **Description** |
|---|---|
| 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 Max Offset (plasmac.max-offset) will be used for the safe height. This defaults to 5mm (0.197”). Refer to the Heights Diagrams for a visual representation. |

Table 10.26: **CONFIGURATION - SCRIBING**

| **Имя** | **Description** |
|---|---|
| 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. |

Table 10.27: **CONFIGURATION - SPOTTING**

| **Имя** | **Description** |
|---|---|
| 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. |

Table 10.28: **CONFIGURATION - PIERCE ONLY**

| **Имя** | **Description** |
|---|---|
| X Offset | Moves the pierce point this distance along the X axis when piercing in Pierce Only mode. |
| Y Offset | Moves the pierce point this distance along the Y axis when piercing in Pierce Only mode. |

Table 10.29: **CONFIGURATION - MOTION**

| **Имя** | **Description** |
|---|---|
| Setup Speed | The Z axis velocity for setup moves (movements to Probe Height, Pierce Height, Cut Height, etc.). |

**Note**
Setup Speed has no effect on THC speed which is capable of the velocity displayed in the Max. Speed field.

Table 10.30: **CONFIGURATION - THC**

| **Имя** | **Modes** | **Description** |
|---|---|---|
| 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. |

Two methods of THC activation are available and are selected with the **Auto Activation** check-button. 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.

**Note**
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.

Table 10.31: **MATERIAL** - The parameters which are active for the current cut.

| **Имя** | **Description** |
|---|---|
| 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. |
| Thickness | This sets the thickness for the currently selected material. This must be used in conjunction with Slat Height. If either Slat Height or Material Thickness are zero then the machine will default to Probe Height. Refer to the Heights Diagrams for a visual representation. |
| Kerf Width | This sets the kerf width for the currently selected material. Refer to the Heights Diagrams for a visual representation. |
| Pierce Height | This sets the pierce height for the currently selected material. Refer to the Heights Diagrams for a visual representation. |
| Pierce Delay | This sets the pierce delay (in seconds) for the currently selected material. |
| Cut Height | This sets the cut height for the currently selected material. Refer to the Heights Diagrams 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 Diagrams 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. |

Table 10.31: (continued)

| Имя | Description |
|---|---|
| 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 |

**Note**
See the thick materials section for more information on puddle jump.

**SAVE, RELOAD, NEW, & DELETE Buttons** The **SAVE** button will save the current material set to the <*machine_name*>_material.cfg file.

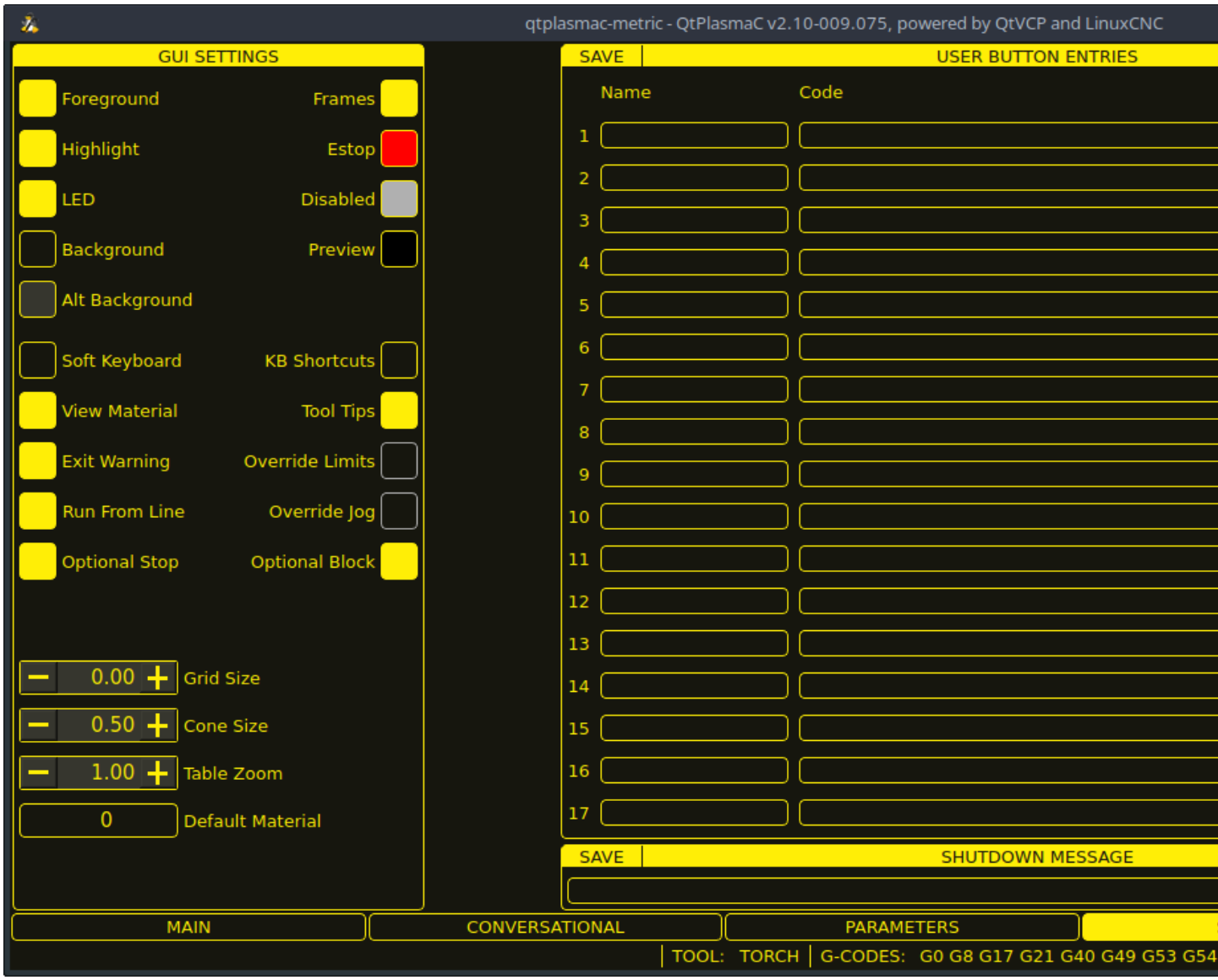
The **RELOAD** button will reload the material set from the <*machine_name*>_material.cfg file.

The **NEW** button will allow a new material to be added to the material file. The user will be prompted for a material number and a material name, all other parameters will be read from the currently selected material. Once entered, QtPlasmaC will reload the material file and display the new material. The Cut Parameters for the new material will then need to be adjusted and saved.

The **DELETE** this button is used to delete a material. After pressing it, the user will be prompted for a material number to be deleted, and prompted again to ensure the user is sure. After deletion, the material file will be reloaded and the drop-down list will display the default material.

#### 10.8.8.6 SETTINGS Tab

Screenshot example of the QtPlasmaC SETTINGS Tab in **16:9** aspect ratio:




This tab is used to display GUI configuration parameters, button text, and shutdown text that are modified infrequently as well as some utility buttons.

It is possible to 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.

Table 10.32: **GUI SETTINGS** Parameters that effect the GUI appearance and GUI behaviors.

| **Имя** | **Description** |
|---|---|
| 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. |

Table 10.32: (continued)

| **Имя** | **Description** |
|---|---|
| LED | This button allows the user to change the color of the GUI LED. |
| Background | This button allows the user to change the color of the GUI Background. |
| 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. |
| Предварительный просмотр | 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.<br>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. |
| Переопределение пределов | 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.

| **Имя** | **Code** |
|---|---|
| The text that is displayed on the button | The code that is run when the button is pressed. |

**Note**
There are 20 user buttons available but not all may be displayed depending on the window size.

**EXIT WARNING MESSAGE**

This section shows the text that appears on the shutdown dialog if the **Exit Warning** is enabled.

The text may be edited at any time and will be loaded ready for use if the **SAVE** button is clicked.

To return the **EXIT WARNING MESSAGE** text to the last saved value press the **RELOAD** button.

**UTILITIES** Some standard LinuxCNC utilities are provided as an aid in the diagnosis of issues that may arise:

- Halshow
- Halscope
- Halmeter
- hm2_pci
- Status

In addition the following two QtPlasmaC specific utilities are provided:

The **SET OFFSETS** button is used if the table has a laser or camera for sheet alignment, a scribe, or uses offset probing. The required offsets for these peripherals need to be applied by following the procedure described in Peripheral Offsets.

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 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). For this reason, G92 offsets have been disabled across the GUI.

QtPlasmaC will automatically add a line of G-code to move the Z axis to the correct height at the beginning of every G-code program.

---

**Note**
It is possible to keep Z motion for use with different tools by adding the magic comment #<keep-z-motion>=1. If using an angular A, B, or C axis for tube cutting then Z axis motion is required in the G-code file.

---

**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 Units Systems

All settings and parameters in QtPlasmaC are required to be in the same units as specified in the <*machine_name*>.ini file, being either metric or imperial.

If the user is attempting to run a G-code file that is in the "other" unit's 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 Preamble and Postamble 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:

Metric:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Imperial:

```
G20 G40 G49 G64p0.004 G80 G90 G92.1 G94 G97
```

A detailed explanation of each G-code can be found in the docs here.

Note that throughout this user guide there are several additional recommendations for codes that are prudent to add to both the preamble and postamble depending on the features the user wishes to utilize.

#### 10.8.9.3 Mandatory 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 Coordinates

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.

If the machine has home switches, then it will move to the specified home coordinates when the Joints are homed.

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.

If the user intends to place the material randomly on the table, then the user must **Touch Off** the X and Y axes at the appropriate position before starting the program.

#### 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]>
```

It is possible to use the standard G-code **F** word to set the cut feed rate as follows:

```
F 1000
```

If the **F** word is used, and the **F** word value does not match the cut feed rate of the selected material then a warning dialog will indicate this during loading of the G-code file.

#### 10.8.9.6 Material File

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.

Material numbers in the materials file do not need to be consecutive nor do they need to be in numerical order.

The following variables are mandatory, and an error message will appear if any are not found when the material file is loaded.

- PIERCE_HEIGHT
- PIERCE_DELAY
- CUT_HEIGHT
- CUT_SPEED

**Note**
If doing tube cutting using the #<tube_cut>=1 magic comment then the only mandatory variable is PIERCE_DELAY, all other variables are optional.

The following variables are optional. If they are not detected or have no value assigned, they will be assigned a value of 0 and no error message will appear.

- ИМЯ
- KERF_WIDTH
- THC
- PUDDLE_JUMP_HEIGHT
- PUDDLE_JUMP_DELAY
- CUT_AMPS
- CUT_VOLTS
- PAUSE_AT_END
- GAS_PRESSURE
- CUT_MODE

**Note**
Material numbers 1000000 and above are reserved for temporary materials.

**Warning**
It is the responsibility of the operator to ensure that the variables are included if they are a requirement for the G-code to be run.

The material file uses the following 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 Manual Material Handling

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
```

The following code is the minimum code necessary to have a successful cut using the manual material selection method:

```
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

**Note**
Manual material handling will restrict the user to only one material for the entire job.

#### 10.8.9.8 Automatic Material Handling

For automatic material handling, the user would add commands to their G-code file which will enable QtPlasmaC to change the material automatically.

The following codes may be used to allow QtPlasmaC to automatically change materials:

- **M190 P*n*** - Changes the currently displayed material to material number *n*.
- **M66 P3 L3 Q1** - Adds a small delay (1 second in this example) to wait for QtPlasmaC to confirm that it successfully changed materials.

- **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
```

**Note**
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 Addition Via Magic Comments In G-code

By using ”magic comments” in a G-code file it is possible to do the following:

- Add new materials to the <*machine_name*>_material.cfg file.
- Edit existing materials in the <*machine_name*>_material.cfg file.
- Use one or more temporary materials.

Temporary materials are numbered automatically by QtPlasmaC and the material change will also be done by QtPlasmaC and should not be added to the G-code file by CAM software or otherwise. The material numbers begin at 1000000 and are incremented for each temporary material. It is not possible to save a temporary material, however the user could create a new material while a temporary material is displayed, and it will use the settings from the temporary material as the defaults.

**Tip**
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.

- The entire comment must be in parentheses.
- The beginning of the magic comment must be: **(o=**
- The equals sign must immediately follow each parameter with no space.
- The mandatory parameters must be in the magic comment (for option 0, **na** is optional and **nu** is not used).
- There can be any number and type of magic comments in a G-code file.
- If option 0 is to be used in addition to option 1 and/or option 2 then all option 0 must appear after all option 1 or all option 2 in the G-code file.

The options are:

| Вариант | Description |
|---|---|
| 0 | Creates a temporary default material.<br>Material information added with this option will be discarded by a LinuxCNC restart or materials reload. They may also be overwritten by a new G-code file that has temporary materials. |
| 1 | Adds a new material if the number specified does not exist. |
| 2 | Overwrites an existing material if the number specified exists.<br>Adds a new material if the number specified does not exist. |

Mandatory parameters are:

| Имя | Description |
|---|---|
| o | Selects the option to be used. |
| nu | Sets the material number (not used for option 0). |
| na | Sets the material name (optional for option 0). |
| ph | Sets the pierce height. |
| pd | Sets the pierce delay. |
| ch | Sets the cut height. |
| fr | Sets the feed rate. |

Optional parameters are:

| Имя | Description |
|---|---|
| mt | Задает толщину материала. |
| kw | Sets the kerf width. |
| th | Sets the THC status (0=disabled, 1=enabled). |
| ca | Sets the cut amps. |
| cv | Sets the cut voltage. |
| pe | Sets the pause at end delay. |
| gp | Sets the gas pressure (PowerMax). |
| cm | Sets the cut mode (PowerMax). |
| jh | Sets the puddle jump height. |
| jd | Sets the puddle jump delay. |

A complete example (metric):

```
(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, mt=5, kw=0.5, th=1, ca ↩
    =40, cv=110, pe=0.1, gp=5, cm=1, jh=0, jd=0)
```

A complete example (imperial):

```
(o=0, nu=2, na=0.197” Mild Steel 40A, ph=0.122, pd=0.1, ch=0.029, fr=118, mt=0.197, kw ↩
    =0.020, th=1, ca=40, cv=110, pe=0.1, gp=72, cm=1, jh=0, jd=0)
```

If a temporary material has been specified in a G-code file then the material change line (M190...) and wait for change line (M66...) will be added by the G-code filter and are not required in the G-code file.

#### 10.8.9.10 Material Converter

This application is used to convert existing tool tables into QtPlasmaC material files. It can also create a material file from manual user input to entry fields.

At this stage the only conversions available are for tool tables exported from either SheetCam or Fusion 360.

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.

If the user has a format from a different CAM software they would like converted, create a **New Topic** in the PlasmaC forum section of the LinuxCNC forum to request this addition.

Material Converter may be run from a terminal using one of the two following methods.

For a package installation (Buildbot) enter the following command in a terminal window:

```
qtplasmac-materials
```

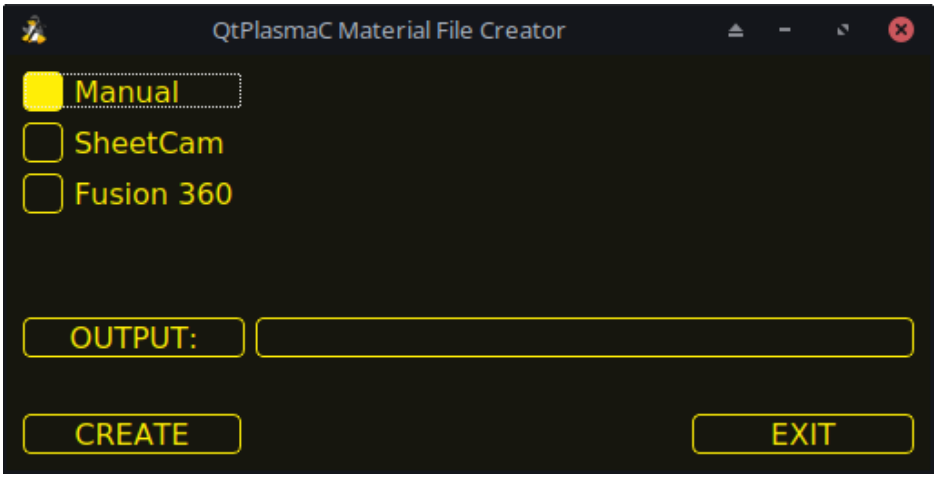
For a run in place installation enter the following two commands in a terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

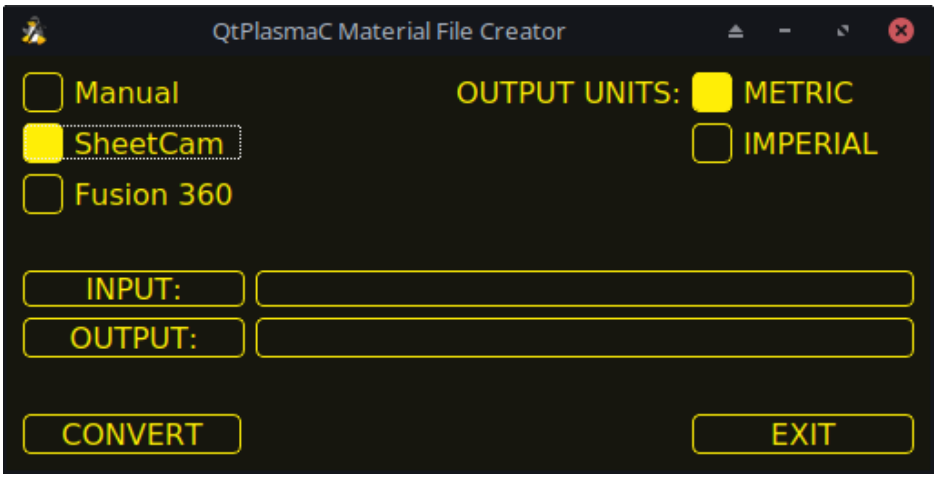
This will bring up the Material Converter Main dialog box with Manual selected as the default.

Select one of:

- **Manual** - to manually create a new material file.

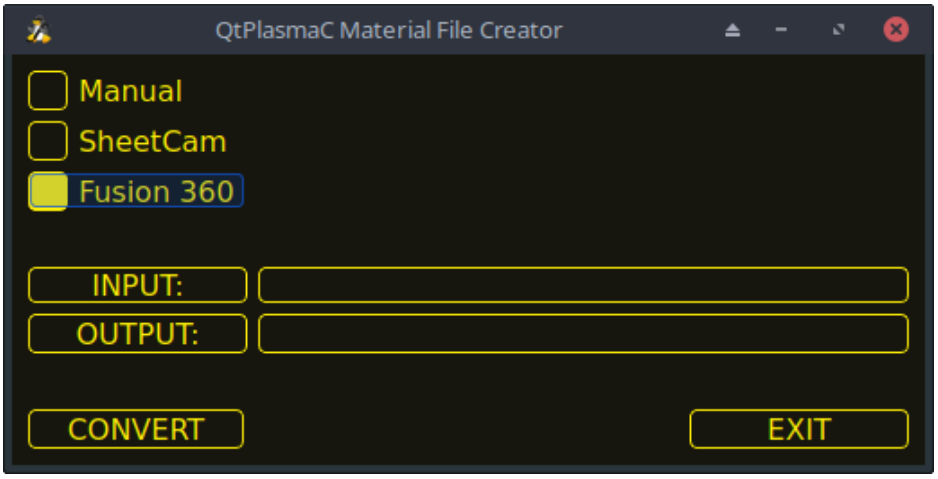



- **SheetCam** - to convert a SheetCam tool file.



For SheetCam only, select whether the user requires a metric or imperial output file.

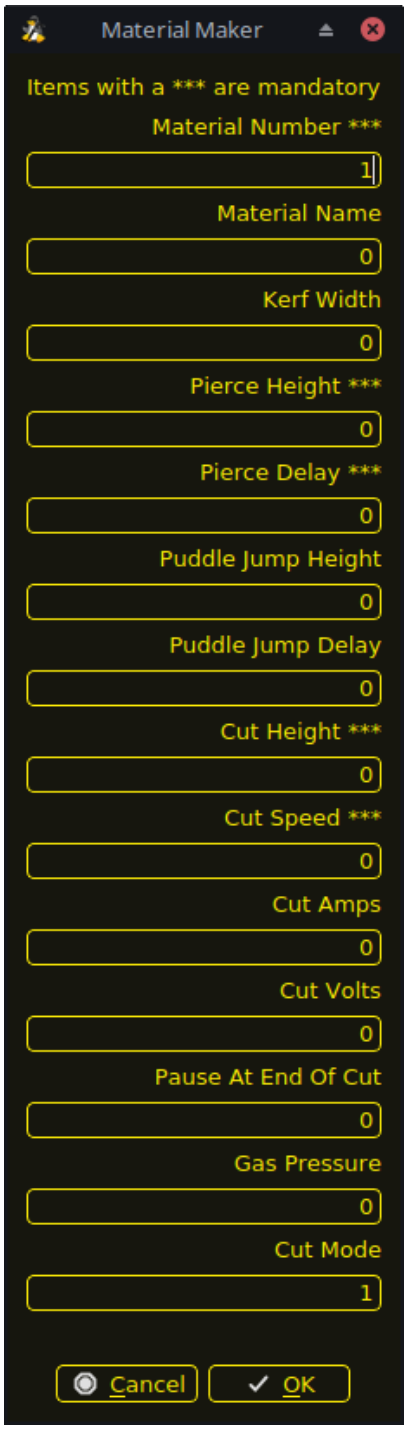
- **Fusion 360** - to convert a Fusion 360 tool file.




To convert:

1. Select the Input File to be converted, press **INPUT** to bring up a file selector or directly enter the file in the entry box.
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. Click **CREATE/CONVERT** and the new material file will be created.

For both a Manual creation or a Fusion 360 conversion, a dialog box will show with all available parameters displayed for input. Any entry marked with *** is mandatory and all other entries are optional depending on the user's configuration needs.




**Note**
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. This button will not be visible until a LASER offset is set in the <*machine_name*>.prefs file.

To use this feature, the user must set the laser'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 **[LASER_OFFSE** 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.

Additionally, the laser can be tied to any available output to turn the laser on and off via a HAL pin with the following name:

```
qtplasmac.laser_on
```

**To set the origin with zero rotation:**

1. Click the **LASER** button.
2. **LASER** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned on.
3. Jog until the laser cross hairs are on top of the desired origin point.
4. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN**.
5. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned off.
6. The torch will now move to the X0 Y0 position.
7. The offset is now successful set.

**To set the origin with rotation:**

1. Click the **LASER** button.
2. **LASER** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned on.
3. Jog until the laser cross hairs are at the edge of the material a suitable distance away from the desired origin point.
4. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN**.
5. Jog until the laser cross hairs are at the origin point of the material.
6. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the HAL pin named qtplasmac.laser_on will be turned off.
7. The torch will now move to the X0 Y0 position.
8. The offset is now successfully set.

**To turn the laser off and cancel an alignment:**

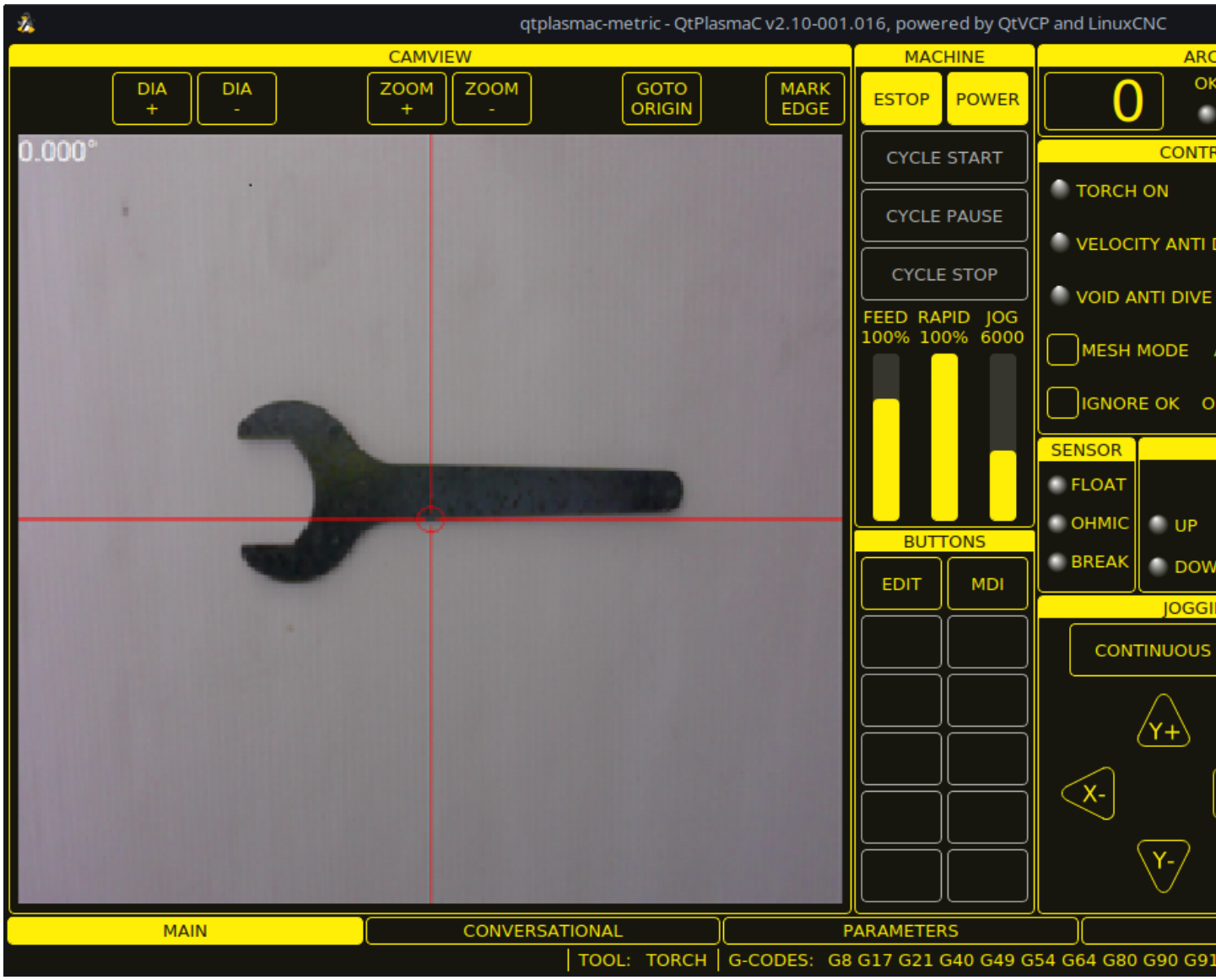
1. Press the **LASER** button and hold for longer than 750 ms.
2. **LASER** button label will change to **LASER** and the HAL pin named qtplasmac.laser_on will be turned off.

3. Release the **LASER** button.

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.

**To dry run the G-code file with the laser:** . Ensure there are no bounds errors and CYCLE START is enabled. . Press the **LASER** button and hold for longer than 750 ms, the laser will turn on and the dry run will start. . Release the **LASER** button.

#### 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. This button will not be visible until a CAMERA offset is set in the <*machine_name*>.prefs file.

To use this feature, the user must set the camera'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 axes options in the **[CAMERA_OFFSET]** section of the <*machine_name*>.prefs file:

```
X axis = n.n
Y axis = n.n
Camera port = 0
```

where *n.n* is distance from the center line of the torch to the camera's cross hairs.

**To set the origin with zero rotation:**

1. Jog until the cross hairs are on top of the desired origin point.
2. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN** and the **GOTO ORIGIN** button will be disabled.
3. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the **GOTO ORIGIN** button will be enabled.
4. The torch will now move to the X0 Y0 position.
5. The offset is now successful set.

**To set the origin with rotation:**

1. Jog until the cross hairs are at the edge of the material a suitable distance away from the desired origin point.
2. Press **MARK EDGE**. The **MARK EDGE** button label will change to **SET ORIGIN** and the **GOTO ORIGIN** button will be disabled.
3. Jog until the cross hairs are at the origin point of the material.
4. Press **SET ORIGIN**. The **SET ORIGIN** button label will change to **MARK EDGE** and the **GOTO ORIGIN** button will be enabled.
5. The torch will now move to the X0 Y0 position.
6. The offset is now successfully set.

In the CAMVIEW panel, the mouse can affect the cross hairs, and the zoom level as follows:

- Mouse Wheel Scroll - Change cross hair diameter.
- Mouse Wheel Button Double Click - Restores cross hair diameter to default.
- Mouse Left Button Clicked + Wheel Scroll - Changes camera zoom level.
- Mouse Left Button Clicked + Wheel Button Double Click - Restores default camera zoom level.

#### 10.8.9.13 Path Tolerance

Path tolerance is set with a G64 command and a following P value. The P value corresponds to the amount that the actual cut path followed by the machine may deviate from the programmed cut path.

The default LinuxCNC path tolerance is set for maximum speed which will severely round corners when used with normal plasma cutting speeds.

It is recommended that the path tolerance is set by placing the appropriate G64 command and P value in the header of each G-code file.

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`.

**For Metric:**

```
G64 P0.1
```

**For Imperial:**

```
G64 P0.004
```

#### 10.8.9.14 Paused Motion

QtPlasmaC has the ability to allow the repositioning of the X and Y axes along the current cut path while the G-code program is paused, aiding in Cut Recovery.

In order to use this feature, LinuxCNC's Adaptive Feed Control (M52) must be turned on (P1). This is also a requirement for Hole Cutting Velocity Reduction.

To enable **Paused Motion** The preamble of the G-code must contain the following line:

```
M52 P1
```

To turn off **Paused Motion** at any point, use the following command:

```
M52 P0
```

#### 10.8.9.15 Pause At End Of 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 Multiple Tools

QtPlasmaC has the ability to allow the use of more than one type of plasma tool by utilizing LinuxCNC spindles as a plasma tool when running a G-code program.

Valid plasma tools for use are:

| Имя | TOOL # | Description |
|---|---|---|
| Plasma Torch | 0 | Used for normal Plasma cutting. |
| Scribe | 1 | Used for material engraving. |
| Plasma Torch | 2 | Used for spotting (creating dimples to aid in drilling). |

A LinuxCNC spindle number (designated by $\$n$) is required to be in the starting command and also the end command to be able to start and stop the correct plasma tool. Examples:

- `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.

It is permissible to use **M5 $-1** in lieu of the M5 $*n* codes above to stop all tools.

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 Velocity Reduction

There is a HAL pin available named **motion.analog-out-03** that can be changed in G-code with the **M67 (Synchronized with Motion)/M68 (Immediate)** commands. This pin will reduce the velocity of the original feed rate to the percentage specified in the command.

The ”VEL:” label at the top right of the preview window will update to reflect the percentage of the original feed rate being used. For example, ”VEL@20%:” means the table is cutting at 20% of the programmed feed rate - an 80% reduction.

**Note**
Because of differences between the GUI and the PlasmaC component polling intervals, velocity label updates may lag (typically by up to 100 ms).

It is important to thoroughly understand the difference between **Synchronized with Motion** and **Immediate**:

- `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.

Examples:

- `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**.

Q values that are less than or equal to 0 or greater than or equal to 100 will be set to 100.

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.

**Important**
**G-CODE THC** AND **VELOCITY BASED THC** ARE NOT ABLE TO BE USED IF **CUTTER COMPENSATION** IS IN EFFECT; AN ERROR MESSAGE WILL BE DISPLAYED.

**Warning**
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 feed rate 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.

**Note**
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 (Torch Height Controller)

The THC can be enabled or disabled from the THC frame of the MAIN Tab.

The THC can also be enabled or disabled directly from the G-code program.

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. If **Use Auto Volts** is checked, then the actual cut voltage is sampled at the end of the THC **Delay** time and this is used as the target voltage to adjust the height of the torch.
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)

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.

#### 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.

Velocity based THC prevents the torch height being changed when velocity is reduced for a sharp corner or a small hole.

It is important to note that Velocity Reduction affects the Velocity Based THC in the following ways:

1. If Velocity Reduction is invoked in the middle of the cut, the THC will be locked.
2. The THC will remain locked until the velocity reduction is canceled by returning it to a value that is above the **VAD Threshold**, and the torch actually reaches 99.9% of the **CutFeedRate**.

### 10.8.9.19 Cutter Compensation

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

**Important**
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 Probing

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.

---

**Note**
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 <*machine_name*>.prefs file with either X or Y not equal to zero, then QtPlasmaC will do **all** Ohmic Probing as Offset Probing. If Offset Probing is valid then the feed rate at which the X and Y axes move to the offset position may be adjusted by the use of the **Offset Speed** parameter in the PROBING frame of the PARAMETERS Tab.

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.

**Important**
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.

There are three methods available for improving the quality of small holes:

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. **Over cut** - Turning the torch off at the end of the hole then continue along the path.

---

**Note**
If both **Arc Dwell** and **Over cut** are active at the same time, then **Over cut** will take precedence.

---

---

**Important**
**OVER CUT** IS NOT ABLE TO BE USED IF CUTTER COMPENSATION IS IN EFFECT; AN ERROR MESSAGE WILL BE DISPLAYED.

---

#### 10.8.9.25 Hole Cutting

G-code commands can be set up by either by a CAM Post Processor (PP) or by hand coding.

**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**.

In order to use this feature, LinuxCNC’s Adaptive Feed Control (M52) must be turned on (P1). This is also a requirement for Paused Motion during Cut Recovery.

To enable **Hole Cutting Velocity Reduction** The preamble of the G-code must contain the following line:

```
M52 P1
```

To turn off **Hole Cutting Velocity Reduction** at any point, use the following command:

```
M52 P0
```

See the Velocity Based THC section.

**Sample code for hole cutting with reduced velocity.**

```
G21 (metric)
G64 P0.005
M52 P1 (enable adaptive feed)
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 (enable adaptive feed)
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 4 mm)
M63 P3 (allow torch to be turned on)
M67 E3 Q0 (restore feed rate to 100%)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

#### 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 Hole Sensing is disabled by default. It can be enabled/disabled by using the following G-code parameters to select the desired hole sensing mode:

- `#<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 (enable adaptive feed)
F#<_hal[plasmac.cut-feed-rate]> (feed rate from cut parameters)
#<holes> = 2 (over cut for holes)
#<oclength> = 6.5 (optional, 6.5 mm over cut length)
G0 X10 Y10
M3 $0 S1 (start cut)
G1 X0
G3 I10 (the hole)
M5 $0 (end cut)
G0 X0 Y0
M2 (end job)
```

**Note**
It is OK to have multiple and mixed hole commands in a single G-code file.

#### 10.8.9.27 Single Cut

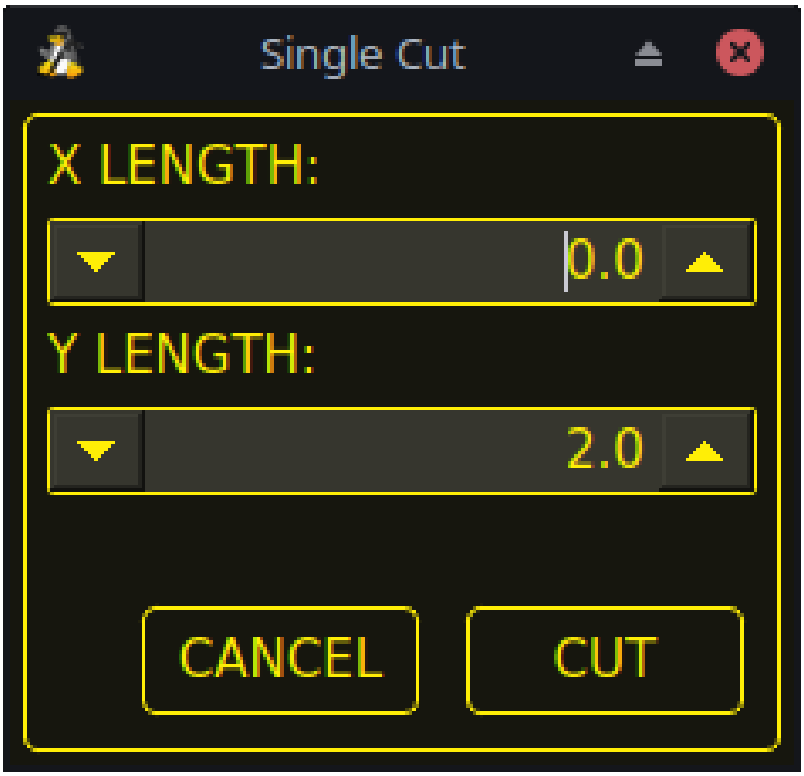
A single cut is a single unidirectional cutting move often used to cut a sheet into smaller pieces prior to running a G-code program.

The machine needs to be homed before commencing a single cut.

A single cut will commence from the machine's current X/Y position.

**Automatic Single Cut**

This is the preferred method. The parameters for this method are entered in the following dialog box that is displayed after pressing a user button which has been coded to run single cut:



1. Jog to the required X/Y start position.
2. Set required appropriate material, or edit the Feed Rate for the default material in the PARAMETERS Tab.
3. Press the assigned single cut user button.
4. Enter the length of the cut along the X and/or Y axes.
5. Press the **CUT** button and the cut will commence.

**Pendant Single Cut** If the machine is equipped with a pendant that can start and stop the spindle plus jog the X and Y axes, the user can manually perform a single cut.

1. Jog to the required X/Y start position.
2. Set the required feed rate with the Jog Speed slider.
3. Start the cut process by starting the spindle.
4. After probing the torch will fire.
5. When the Arc OK is received the machine can be jogged along the cut line using the jog buttons.
6. When the cut is complete stop the spindle.
7. The torch will turn off and the Z axis will return to the starting position.

**Manual Single Cut**

Manual single cut requires that either keyboard shortcuts are enabled in the GUI SETTINGS section of the SETTINGS Tab, or a custom user button is specified as a manual cut button.

If the user is using a custom user button then, substitute **F9** with **User Button** in the following description.

1. Jog to the required X/Y start position.
2. Start the procedure by pressing **F9**. The jog speed will be automatically set to the feed rate of the currently selected material. The jog label will blink to indicate that the jog speed is temporarily being overridden (jog speed manipulation will be disabled while a manual cut is active). **CYCLE START** will change to **MANUAL CUT** and blink.
3. After probing the torch will fire.
4. When the Arc OK is received the machine can be jogged along the cut line using the jog keys.
5. The Z height will remain locked at the cut height for the duration of the manual cut, regardless of the Torch Height Controller **ENABLE** status.
6. When the cut is complete press **F9** or **Esc** or the **CYCLE STOP** button.
7. The torch will turn off and the Z axis will return to the starting position.
8. The jog speed will automatically be returned to the value it was prior to initiating the manual cut process, the label will stop blinking and the jog speed manipulation will be enabled. **MANUAL CUT** will stop blinking and revert to **CYCLE START**.

---

**Note**
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.

There are several functions built into QtPlasmaC to help alleviate these issues, Pierce Only and Puddle Jump described in this section as well as Wiggle Pierce and Ramp Pierce described in the Moving Pierce section.

**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.

The pierce location during **Pierce Only** mode may be offset in the X and/or Y axes to ensure that the arc is able to transfer correctly when piercing after returning to the **Normal Cut** mode. The parameters for the X and Y Offsets are in the PIERCE ONLY frame of the CONFIGURATION section of the PARAMETERS Tab

**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.

**Important**
**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)

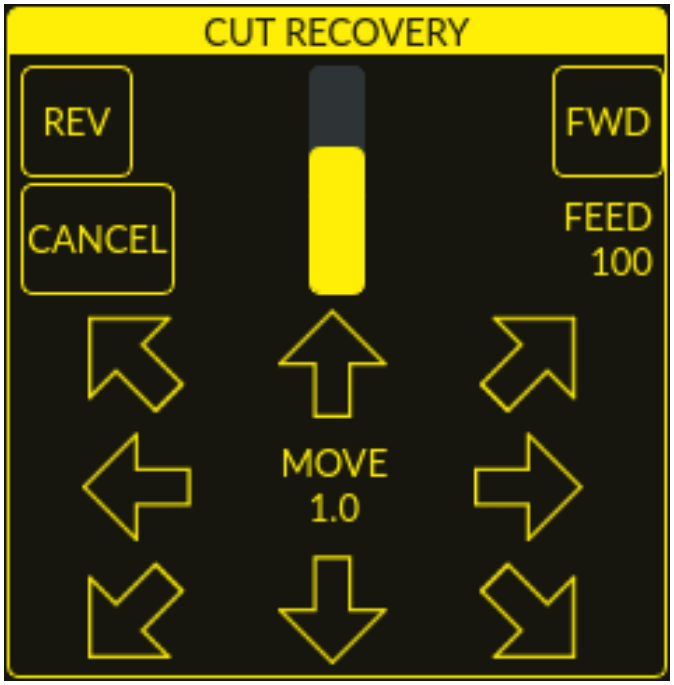
It is important to thoroughly understand the difference between **Synchronized with motion** and **Immediate**:

- `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. Click the **LASER** button.
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.

**Note**
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.

**Important**
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

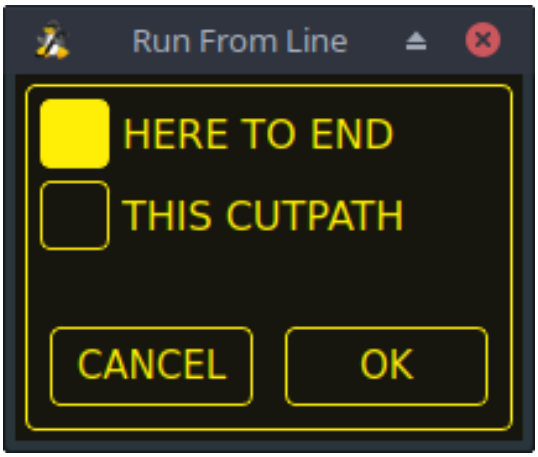
It is important to note that G-code programs can be run from any selected line using this method, however a lead-in may not be possible depending on the line selected. In this case, an error message will be displayed to let the user know the lead-in calculation was not possible.

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:

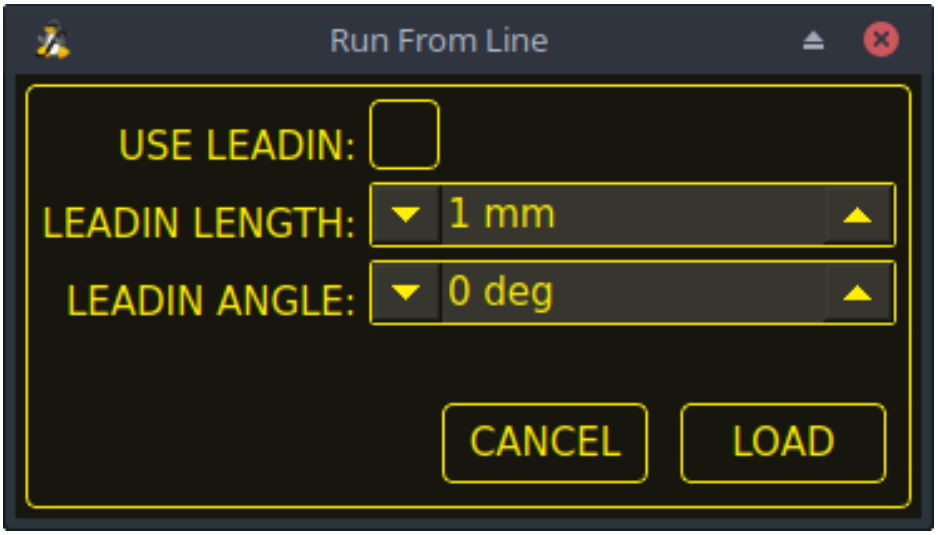
It is not possible to use Run From Line from within a subroutine. If the user selects a line within a subroutine and clicks **"SELECTED *nn*"** then an error message will be displayed that includes the O-code name of the subroutine.

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.




**HERE TO END** will run from the beginning of the selected line to the end of the G-code file. The user will be presented with the option of adding a lead-in if the selected line falls within an "active" cutting operation (between an M3 and M5).



| Имя | Description |
|---|---|
| USE LEADIN | This radio button will allow the user to start the selected line with a lead-in. |
| 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 lead-in.<br>The angle is measured such that positive increases in value move the lead-in counterclockwise:<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 lead-in parameters applied.<br>If the lead-in cannot be calculated for the selected line, the following error message will be displayed:<br>"Unable to calculate a lead-in for this cut<br>Program will run from selected line with no lead-in 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.

**THIS CUTPATH** will run only the cutpath that the selected segment is a part of.

The blinking "SELECTED *nn*" button will change to **RUN FROM LINE CYCLE START** button. Click this button to run the selected cutpath.

**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. Clicking **RELOAD** in the G-code window header - this method will cancel the Run From Line process if LOAD was clicked on the Run From Line dialog box and "rfl.ngc" is displayed as the loaded file name in the G-code window header. This will return the user to the originally loaded file.

---

**Note**
Although Run From Line allows the user to begin execution at any line, not every possible scenario can be fully tested. Most testing has focused on recovering typical cutting operations. Therefore when recovering outside of cutting operations, users should select a starting line that provides the GUI with the most context about the operation. For example, when restarting a spotting operation, choose the G0 line before the M3 command rather than the M3 itself or the G1 X0.000001 move in the middle of the spotting operation.

---

#### 10.8.9.33 Scribe

A scribe may be operated by QtPlasmaC in addition to the plasma torch.

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.

To stop the scribe, use the G-code command **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 (enable adaptive feed)
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.

**Warning**
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
```

**Note**
The **high feed rate** of 99999 is to ensure that the motion is at the machine's highest feed rate.

**Important**
SOME PLASMA CUTTERS WILL NOT BE SUITABLE FOR THIS FEATURE.
IT IS RECOMMENDED THAT THE USER CARRY OUT SOME TEST SPOTTING TO ENSURE THAT THE PLASMA CUTTER IS CAPABLE OF UTILIZING THIS FEATURE.

#### 10.8.9.35 Tube Cutting

Tube cutting with an angular A, B, or C axis is achieved with the following in the G-code file:

- #<tube_cut>=1 magic comment before any motion command.
- All material probing must be done using the G38 straight probe codes.
- All Z axis motion is required, PlasmaC does no internal Z axis motion during tube cutting.
- PIERCE_DELAY is the only required material parameter
- Start a cut with M3 $0 S1.
- End a cut with M5 $0

#### 10.8.9.36 Virtual Keyboard Custom Layouts

Virtual keyboard support is available for only the ”onboard” onscreen keyboard. If it is not already on the system, it may be installed by typing the following in a terminal:

```
sudo apt install onboard
```

The following two custom layouts are used for soft key support:

Figure 10.50: Number keypad - used for the CONVERSATIONAL Tab and the PARAMETERS Tab

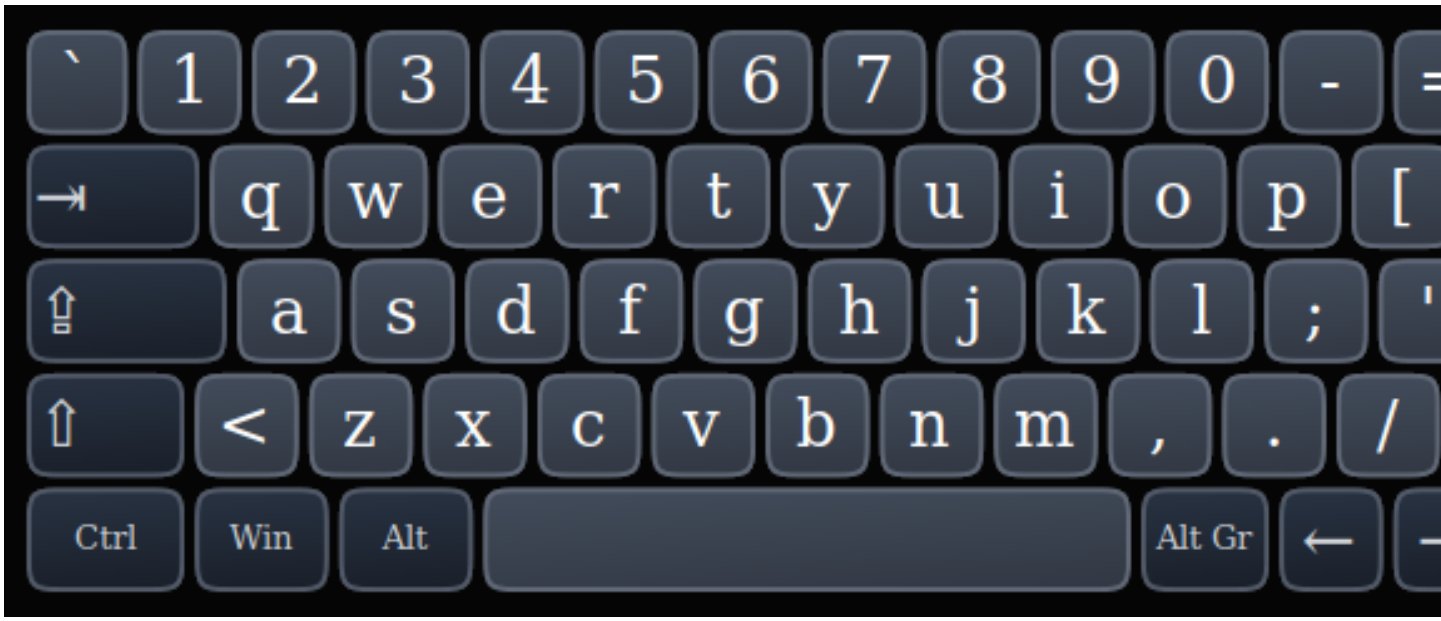



Figure 10.51: Alphanumeric keypad - used for G-code editing and file management.

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.

#### 10.8.9.37 Keyboard Shortcuts

Below is a list of all available keyboard shortcuts in QtPlasmaC.

**Note**
All keyboard shortcuts are disabled by default.

In order to utilize them, **KB Shortcuts** must be enabled in the **GUI SETTINGS** section of the SETTINGS Tab.

| **Keyboard Shortcut** | **Действие** |
|---|---|
| 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 style sheet 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. |

| **Keyboard Shortcut** | **Действие** |
|---|---|
| SHIFT+`, 1-9, 0 | Changes rapid speed to 0%, 10%-90%, 100%. |
| CTRL+1-9, 0 | Changes feed rate to 10%-90%, 100%. |
| CTRL+HOME | Homes all axes if they are not yet homed and have a homing sequence set in the <*machine_name*>.ini file. If they are already homed, they will no longer be homed. |
| CTRL+R | Cycle Start if the program is not already running. Cycle Resume if the program is paused. |
| END | Touches off X and Y to 0. |
| DEL | Allows the user to use a laser to set an origin with or without rotation. See the LASER section for detailed instructions. |
| SPACE BAR | Pauses motion. |
| CTRL+SPACE BAR | Clears notifications. |
| O | Opens a new program. |
| L | Loads the previously opened program if no program is loaded. Reloads the current program if there is a program loaded. |
| → | Jogs the X axis positive. |
| ← | Jogs the X axis negative. |
| ↑ | Jogs the Y axis positive. |
| ↓ | Jogs the Y axis negative. |
| PAGE UP | Jogs the Z axis positive. |
| PAGE DOWN | Jogs the Z axis negative. |
| [ | Jogs the A axis positive. |
| ] | Jogs the A axis negative. |
| . | Jogs the B axis positive. |
| , | Jogs the B axis negative. |
| SHIFT (+ Jog Key) | The shift key is used with any jog key to invoke a rapid jog. |
| + (+Jog Key) | The plus key can be used with any jog key to invoke a rapid jog (behaves the same as SHIFT). |
| - (+Jog Key) | The minus key can be used with any jog key to invoke a slow jog (10% of the displayed jog speed)<br>If SLOW jogging is already active, the axis will jog at the displayed jog speed. |

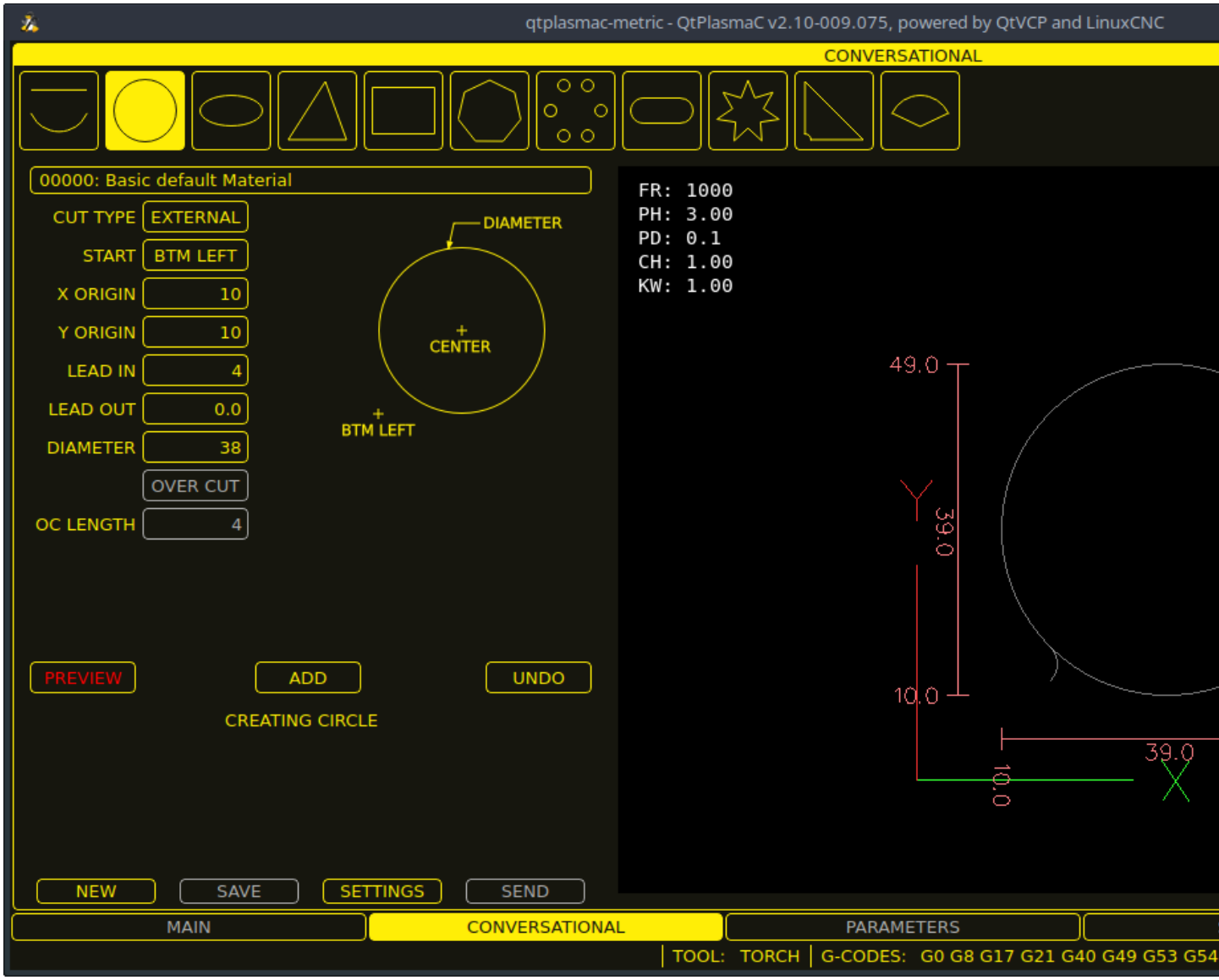
#### 10.8.9.38 MDI

In addition to the typical G and M codes that are allowed by LinuxCNC in MDI mode, the MDI in QtPlasmaC can be used to access several other handy features. The following link outlines the features and their use: Section 12.7.2.15[MDILine Widget]

**Note**
M3, M4, and M5 are not allowed in the QtPlasmaC MDI.

**In addition, pressing RETURN (or ENTER) with no entry in the MDI will close the MDI window.**

### 10.8.10 Conversational Shape Library




The **Conversational Shape Library** consists of several basic shapes and functions to assist the user with generating quick G-code at the machine to cut simple shapes quickly. This feature is found on the CONVERSATIONAL Tab.

**Note**
The Conversational Library is not meant to be a CAD/CAM replacement as there are limitations to what can be achieved.

Blank entries in the shape input boxes will use the current setting at the time the G-code was generated. For example, if **X start** was left blank then the current X axis position would be used.

All lead-ins and lead-outs are arcs except for **Circles** and **Stars**:

**Circles:**

- If the circle is external then any lead-in or lead-out will be an arc.

- If the circle is internal and a **small hole** then any lead-in will be perpendicular and there will be no lead out.
- If the circle is internal and not a **small hole** then any lead-in and let-out will be an arc. If the lead-in has a length greater than half the radius then the lead-in will revert to perpendicular and there will be no let-out. If the lead-out has a length greater than half the radius then there will be no leadout.

**Stars:**

- The lead-in is at the same angle as the first cut and the leadout is at the same angle as the last cut.

---

**Note**
A **small hole** is a circle that is smaller than the SMALL HOLE DIAMETER specified in the CONVERSATIONAL SETTINGS page.

---

---

**Note**
The holes in a BOLT CIRCLE shape will also abide by the above rules.

---

**The cut order will occur in the same order as the shape was built.**

Pressing **Return** on the keyboard while editing parameters will automatically show the preview of the shape if there are enough parameters entered to create the shape. Clicking any of the available check boxes will do the same.

The general functions are as follows:

| **Имя** | **Description** |
|---|---|
| Material Drop-Down | Allows the user to select the desired material for cutting. If "VIEW MATERIAL" is selected on the SETTINGS Tab, a visual reference showing key material cut settings will be displayed on the Conversational Preview Window. Examples are: Feed Rate, Pierce Height, Pierce Delay, Cut Height, and Kerf Width (for Conversational only). Cut Amps will be shown if PowerMax communications are enabled. |
| NEW | Removes the current G-code file and load a blank G-code file. |
| SAVE | Opens a dialog box allowing the current shape to be saved as a G-code file. |
| SETTINGS | Allows the changing of the global settings. |
| SEND | Loads the current shape into LinuxCNC (QtPlasmaC). If the last edit was not added, then it will be discarded. |
| PREVIEW | Displays a preview of the current shape provided the required information is present. |
| CONTINUE | This button is used for lines and arcs only. Allows another segment to be added to the current segment/segments. |
| ADD | Stores the current shape into the current job. |
| UNDO | Reverts to the previously stored state. |
| RELOAD | Reloads the original G-code file or a blank file if none was loaded. |

If there is a G-code file loaded in LinuxCNC (QtPlasmaC) when the CONVERSATIONAL Tab is selected, that code will be imported into the conversational as the first shape of the job. If this code is not required, then it can be removed by pressing the **NEW** button.

If there is an added shape that is unsaved or unsent then it is not possible to switch tabs in the GUI. To re-enable switching tabs, it is necessary to either **SAVE** the shape, **SEND** the shape, or press **NEW** to remove the shape.

If **NEW** is pressed to remove an added shape that is unsaved or unsent then a warning dialog will be displayed.

**Note**
All distances are in machine units relative to the current User Coordinate System and all angles are in degrees.

#### 10.8.10.1 Conversational Settings

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:

- **Preamble**
- **Postamble**
- **Origin (Center or Bottom Left)**
- **Lead-in length**
- **Leadout length**
- **Small hole diameter**
- **Small hole speed**
- **Preview Window Grid Size**

Any internal circle that has a diameter less than **Small hole diameter** is classified as a small hole and will have a straight lead-in with a length that is the lesser of either the radius of the hole or the specified lead-in length. It will also have its feed rate set to **Small hole speed**.

Preamble and Postamble may be entered as a string of G-Codes and M-Codes separate by spaces. If the user wishes for the generated G-code to have each code on an individual line, then this is made possible by separating the codes with **\n**.

This will place all codes on the same line:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

This will place each code on its own line:

```
G21\nG40\nG49\nM52P1\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

Note that LinuxCNC does not allow multiple **P** words on the same line.

Pressing the **RELOAD** button will discard any changed but unsaved settings.

Pressing the **SAVE** button will save all the settings as displayed.

Pressing the **EXIT** button will close the setting panel and return to the previous shape.

#### 10.8.10.2 Conversational Lines And Arcs

Lines and arcs have an additional option in that they may be strung together to create a complex shape.

There are two line types and three arc types available:

1. **Line** given a start point and an end point.
2. **Line** given a start point, length, and angle.
3. **Arc** given a start point, way point, and end point.
4. **Arc** given a start point, end point, and radius.
5. **Arc** given a start point, length, angle, and radius.

To use lines and arcs:

1. Select the **Lines and Arcs** icon.
2. Select the type of line or arc to create.
3. Choose the material from the MATERIAL drop-down. If no material is chosen, the default material (00000) will be used.
4. Enter the desired parameters.
5. Press **PREVIEW** to see the shape.
6. If satisfied with the shape press **CONTINUE**.
7. Change the line or arc type if needed and continue this procedure until the shape is complete.
8. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

If the user wishes to create a closed shape, they will need to create any required leadin as the first segment of the shape. If a leadout is required, it will need to be the last segment of the shape.

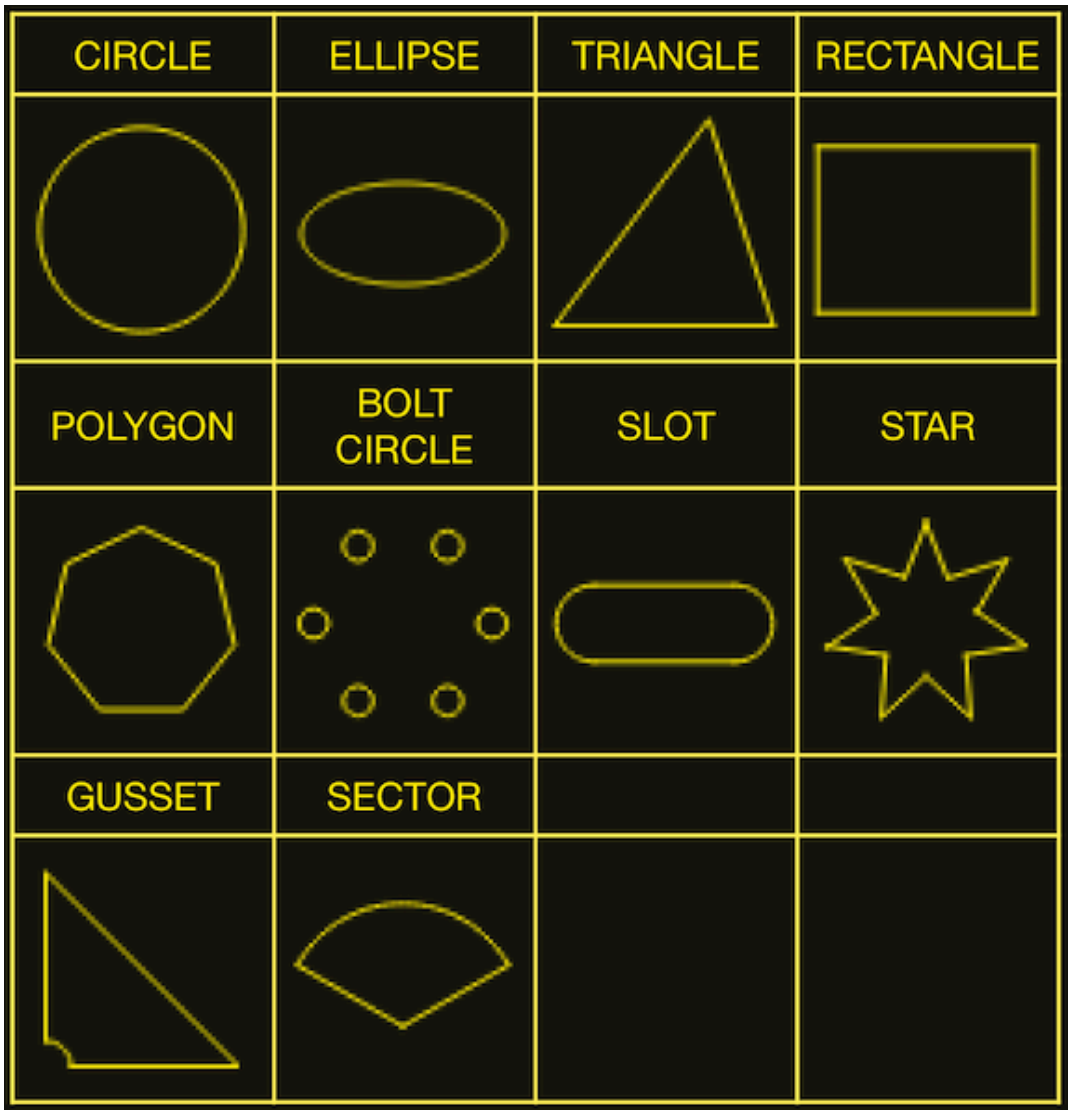
---

**Note**
At this stage there is no automatic option for a lead-in/leadout creation if the shape is closed.

---

#### 10.8.10.3 Conversational Single Shape

The following shapes are available for creation:




To create a shape:

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. Enter the appropriate values and press **PREVIEW** to display the shape.
4. If the shape is not correct, edit the values and press **PREVIEW** and the new shape will be displayed. Repeat until satisfied with the shape.
5. Press **ADD** to add the shape to the G-code file.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

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.

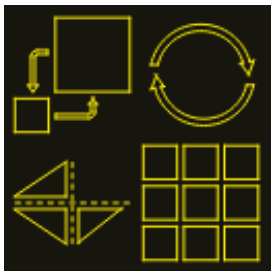
Once a shape is added to the group it cannot be edited or removed.

Groups cannot have shapes removed, only added to.

To create a group of shapes:

1. Create the first shape as in **Single Shape**.
2. Press **ADD** and the shape will be added to the group.
3. If the user wishes to add another version of the same shape, then edit the required parameters and press **ADD** when satisfied with the shape.
4. If the user wishes to add a different shape, select that shape and create it as in **Single Shape**.
5. Repeat until all the required shapes to complete the group have been added.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting.

#### 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.

All block operations are done in the machines native units. If a file of a different units system has been opened then the copy loaded into the CONVERSATIONAL Tab will be converted to the machines native units when the CONVERSATIONAL Tab becomes active.

Block operations:

- Rotate
- Шкала с ползунком
- Array
- Mirror
- Flip

To create a block:

1. Create a shape, a group, or use a previously loaded G-code file.
2. Click the **Block** icon to open the Block tab.
3. Enter the appropriate values in the Block tab and press **PREVIEW** to display the resulting changes.
4. If the result is not correct, edit the values and press **PREVIEW** and the new result will be shown. Repeat until satisfied with the result.
5. Press **ADD** to complete the procedure.
6. Press **SEND** to send the G-code file to LinuxCNC (QtPlasmaC) for cutting, or **SAVE** to save the G-code file.

**COLUMNS & ROWS**
: specifies the number of duplicates of the original shape arranged in columns and rows as well as the spacing between each shape's origin.

**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.

If the result is an array of shapes, then the cut order of the result is from the left column to the right column, starting at the bottom row and ending at the top row.

#### 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 Error Messages

### 10.8.11.1 Error Logging

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
```

**Note**
<*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**

These errors indicate the corresponding sensor was activated before probing commenced:

- **ohmic probe detected before probing program is paused**

- **float switch detected before probing program is paused**
- **breakaway switch detected before probing program is paused**

The Arc OK signal was lost during cutting motion, before the **M5** command was reached:

- **valid arc lost program is paused**

The Z axis reached the bottom limit before the work piece was detected:

- **bottom limit reached while probing down program is paused**

The work piece is too high for any safe rapid removes:

- **material too high for safe traverse, program is paused**

One of these values in MATERIAL section of the PARAMETERS Tab is invalid (For example: if they are set to zero):

- **invalid pierce height or invalid cut height or invalid cut volts, program is paused**

No arc has been detected after attempting to start the number of times indicated by **Max Starts** in the ARC frame of the CONFIGURATION section of the PARAMETERS Tab:

- **no arc detected after <*n*>d start attempts program is paused**
- **no arc detected after <*n*>d start attempts manual cut is stopped**

THC has caused the bottom limit to be reached while cutting:

- **bottom limit reached while THC moving down program is paused**

THC has caused the top limit to be reached while cutting:

- **top limit reached while THC moving up program is paused**

These errors indicate move to pierce height would exceed the Z Axis MAX_LIMIT for the corresponding probe method:

- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during float switch probing**
- **pierce height would exceed Z axis maximum limit condition found while moving to probe height during ohmic probing**

These errors indicate the move to pierce height would exceed the Z axis maximum safe height for the corresponding probe method:

- **pierce height would exceed Z axis maximum safe height condition found while float switch probing**
- **pierce height would exceed Z axis maximum safe height condition found while ohmic probing**

#### 10.8.11.4 Warning 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!**

**Warning**
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**

This indicates that the safe height has been reduced due to THC raising the Z axis during cutting:

- **safe traverse height has been reduced**

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.

**Important**
Any manual modification to the <*machine_name*>.ini and <*machine_name*>.hal files will not be registered in PnCConf or StepConf.

**Note**
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 style sheets and some customization may be achieved by the use of a custom style sheet. 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 style sheets 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 style sheet 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 style sheets 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 style sheet 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 style sheet 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 Style-sheet 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.

| **Цвет** | **Parameter (Параметр)** | **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 | background of G-code display’s active line |

#### 10.8.14.3 Returning To The Default Styling

The user may return to the default styling at any time by following the following steps:

1. Close QtPlasmaC if open.
2. Delete qtplasmac.qss from the machine config directory.
3. Delete qtplasmac_custom.qss from the machine config directory (if it exists).
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.

The next time QtPlasmaC is loaded all custom styling will be removed and the default styling will return.

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 machine's 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:

| **Имя** | **Function** |
|---|---|
| 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
```

---

**Note**
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 Advanced Topics

### 10.8.15.1 Custom User 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.

The number of user buttons varies by display type and resolution as follows:

- 16:9 and 4:3 - Minimum 8, Maximum 20
- 9:16 - Minimum 15, Maximum 20

The user will need to run QtPlasmaC at the desired screen size to determine how many user buttons are available for use.

All <*machine_name*>.prefs file settings for the buttons are found in the **[BUTTONS]** section.

**Button Names** The text that appears on the button is set the following way:

```
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
```

If an ampersand is required to be displayed as text then two consecutive ampersands are required:

```
n Name = PIERCE&&CUT
```

**Button Code** Buttons can run the following:

1. External commands
2. External python scripts
3. G-code commands
4. Dual code
5. Toggle a HAL pin
6. Toggle the alignment laser HAL pin
7. Pulse a HAL pin
8. Probe test
9. Ohmic Test
10. Cut Type
11. Change consumables
12. Load a G-code program
13. Pulse the torch on
14. Single unidirectional cut
15. Framing a job
16. Begin/End a manual cut

17. Display/Hide an offsets viewer
18. Load the latest modified NGC file found in a directory
19. Display/Hide the online HTML user manual
20. Toggle between joint and teleop modes

#### 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 user's home directory.

```
n Code = %~/user_script.py
```

#### G-code

To run G-code, just enter the code to be run.

```
n Code = G0 X100
```

To run an existing subroutine.

```
n Code = o<the_subroutine> call
```

<*machine_name*>.ini file variables can be entered by using 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
```

#### Двойной код

Dual Code allows the running of two code snippets alternately with each button press. The button text will alternate with each button press and the indicator light may be optionally enabled.

It is mandatory to specify the button code in the following order: "dual-code", the first code, the alternate button text, and the second code separated by double semicolons. If an indicator is required then optionally add ";; true" at the end.

```
n Code = dual-code ;; code1 ;; name1 ;; code2 ;; true
```

On the first button press, code1 will be run, the button text will change to name1, and if ”true” is specified the indicator will light.

On the second button press, code2 will be run, the button text will change to n Name, and the indicator will extinguish if lit.

code1 and code2 both follow the rule of the preceding code explanations, External commands, Python code, and G-code. Multiple codes as well as mixing codes are allowed.

The following code will allow the user to use a single button to run two code snippets alternately each button press:

```
n Name = X+10
n Code = dual-code ;; G91\G0X10\G90 ;; X-10 ;; G91\G0X-10\G90
```

The original label will be X+10, when pressed the torch will move positive 10 in the X axis and the label will change to X-10. When pressed again the torch will move negative 10 in the X axis and the label will change to X+10.

**Специальные команды** The following commands must be a single command per user button, and the button code must start with the special command. The exception is toggle-laser which may be appear anywhere in the code as demonstrated below.

**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.

It is also possible for the user to specify alternate text which will display on the button while ever it is in the latched-on condition. To specify the alternate text, use a double semicolon followed by the required text. This must be the last item in the button code.

```
n Code = toggle-halpin the-hal-pin-name ;; PIN\TOGGLED
```

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 sequence by adding ”cutcritical” after the HAL pin in the button code. If **TORCH ENABLE** is checked and **CYCLE START**, **MANUAL CUT**, or **SINGLE CUT** are initiated while the ”cutcritical” button is not ”ON” then the user will receive a dialog warning them as such and asking to CONTINUE or CANCEL. The dialog will list all untoggled buttons with a corresponding checkbox and allow the user to choose which of the buttons should be toggled automatically upon clicking CONTINUE.

```
n Code = toggle-halpin the-hal-pin-name cutcritical
```

**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.

**Probe 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
```

---

**Note**
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.

---

### Ohmic 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
```

**Note**
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
```

### Change Consumables

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 X*nnn* Y*nnn* F*nnn*. 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
```

**Note**
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.

**Torch Pulse**

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
```

---

**Note**
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.

---

**Note**
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
```

or:

```
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
```

### Toggle Joint Mode

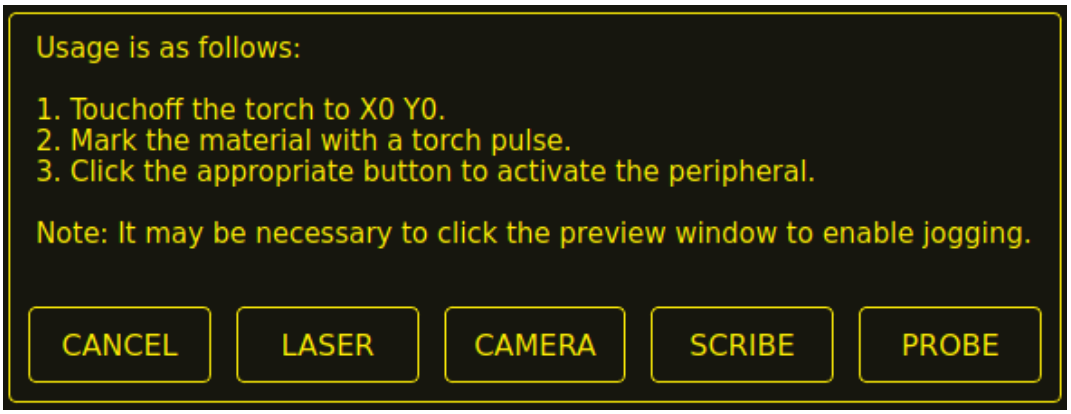
This allows the toggling between joint mode and teleop mode. The machine must be on and homed for this button to be active.

```
n Code = toggle-joint
```

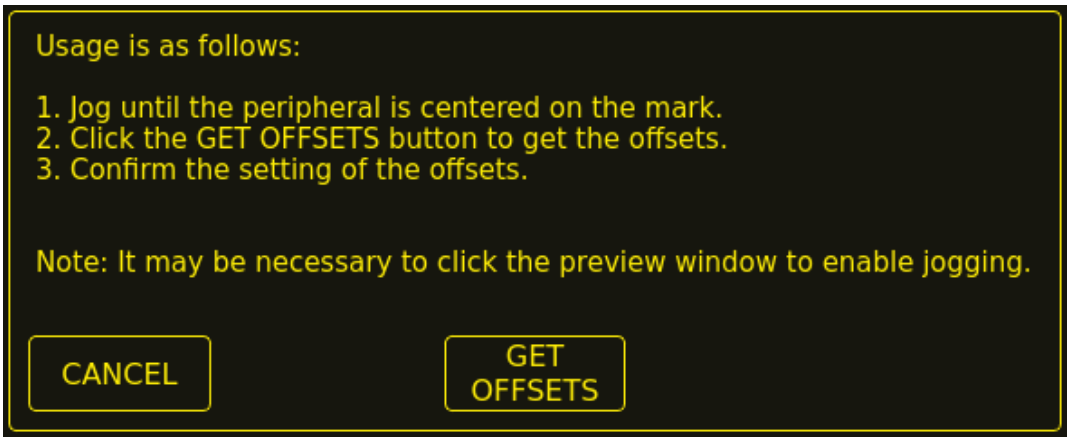
#### 10.8.15.2 Peripheral Offsets (Laser, Camera, Scribe, Offset Probe)

Use the following sequence to set the offsets for a laser, camera, scribe, or offset probe:

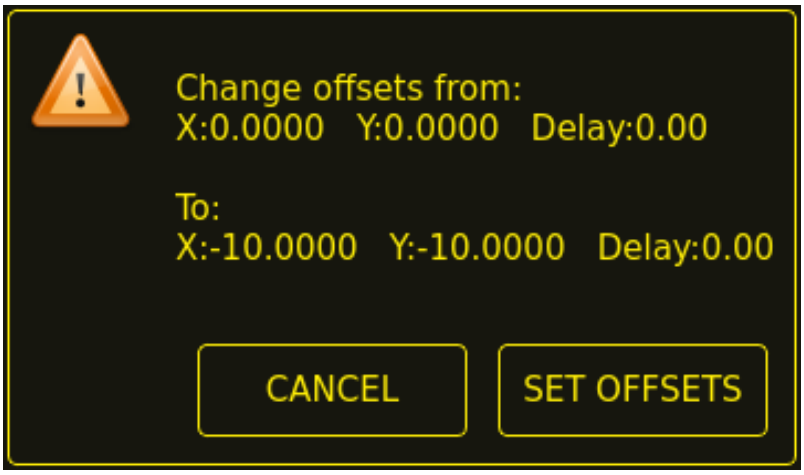
1. Place a piece of scrap material under the torch.
2. The machine must be homed and idle before proceeding.
3. Open the SETTINGS tab.
4. Click the SET OFFSETS button which opens the Set Peripheral Offsets dialog.




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.



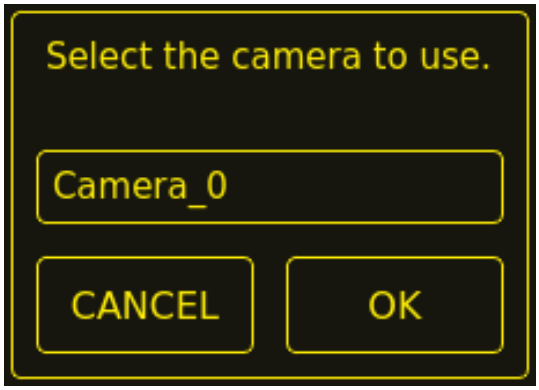

9. Raise the Z axis so the torch and peripheral are clear of the material.
10. Jog the X/Y axes so that the peripheral is centered in the mark from the torch.
11. Click the GET OFFSETS button to get the offsets and a confirmation dialog will open.



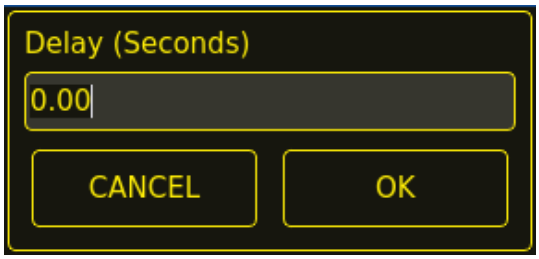

12. Click SET OFFSETS and the offsets will now be saved.

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.




**Note**
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, 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
```

This can be used two different ways: . In a G-code file with no M3 (cutting, scribing, or spotting) commands. In this case, #<keep-z-motion> = 1 can be placed anywhere before the first Z movement and all subsequent Z motion will be retained. The G-code filter will not add any initial Z movements. . In a G-code file that also contains subsequent spotting and/or cutting operations. In this case, the marking portion of the file that contains Z movements needs to preceed the spotting and/or cutting operations, and the marking section needs to have #<keep-z-motion> = 1 at the beginning and #<keep-z-motion> = 0 at the end. In this case, the G-code filter will automatically add an initial Z movement for the first M3 command after #<keep-z-motion> = 0.

In either case, M3 commands are not supported while #<keep-z-motion> is active.

Omitting #<keep-z-motion>, or setting #<keep-z-motion> to anything but 1 will cause QtPlasmaC the default behavior of stripping all Z motion from a loaded G-code file and adding an initial Z movement to bring the torch near the top of travel before the first M3 command.

#### 10.8.15.4 External HAL Pins

QtPlasmaC creates some HAL pins that may be used to connect a momentary external button or pendant etc.

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.

The following HAL bit pins are always created. The HAL pin has the identical behaviour of the related QtPlasmaC GUI button.

| **User Button Function** | **HAL Pin** | **GUI Function** |
|---|---|---|
| Toggle machine power | `qtplasmac.ext_power` | POWER |
| Run the loaded G-code program | `qtplasmac.ext_run` | CYCLE START |
| Pause/Resume the loaded G-code program | `qtplasmac.ext_pause` | CYCLE PAUSE/CYCLE RESUME |
| Приостановить загруженную программу G-кода | `qtplasmac.ext_pause_only` | CYCLE PAUSE |
| Resume the loaded G-code program | `qtplasmac.ext_resume` | CYCLE RESUME |
| Abort the loaded G-code program | `qtplasmac.ext_abort` | CYCLE STOP |
| Touchoff X & Y axes to zero | `qtplasmac.ext_touchoff` | TOUCHOFF X0Y0 |
| Use a laser to set an origin with or without rotation | `qtplasmac.ext_laser_touchoff` | LASER |
| Toggle qtplasmac.laser_on pin | `qtplasmac.ext_laser_toggle` | N/A |
| Run/Pause/Resume the loaded G-code program | `qtplasmac.ext_run_pause` | CYCLE START, CYCLE PAUSE, CYCLE RESUME in sequence |
| Torch height override plus | `qtplasmac.ext_height_ov_plus` | OVERRIDE + |
| Torch height override minus | `qtplasmac.ext_height_ov_minus` | OVERRIDE - |
| Torch height override reset | `qtplasmac.ext_height_ov_reset` | OVERRIDE RESET TO 0.00 |
| Torch height override scale | `qtplasmac.ext_height_ov_scale` | N/A |

| User Button Function | HAL Pin | GUI Function |
|---|---|---|
| Toggle jogging speed between fast and slow | qtplasmac.ext_jog_slow | JOGGING FAST/SLOW |
| Toggle THC enable | qtplasmac.ext_thc_enable | THC ENABLE |
| Toggle torch enable | qtplasmac.ext_torch_enable | TORCH ENABLE |
| Toggle corner Lock enable | qtplasmac.ext_cornerlock_enable | VELOCITY ANTI DIVE ENABLE |
| Toggle voidlock enable | qtplasmac.ext_voidlock_enable | VOID ANTI DIVE ENABLE |
| Toggle use auto volts | qtplasmac.ext_auto_volts_enable | AUTO VOLTS |
| Toggle ohmic probe enable | qtplasmac.ext_ohmic_probe_enable | OHMIC ENABLE |
| Toggle mesh mode | qtplasmac.ext_mesh_mode | MESH MODE |
| Toggle arc ignore OK | qtplasmac.ext_ignore_arc_OK | IGNORE OK |
| Forward along the programmed path | qtplasmac.ext_cutrec_fwd | CUT RECOVERY FWD |
| Reverse along the programmed path | 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 |

The following HAL pins which allow the use of an MPG to control height override are always created.

| Function | HAL Pin |
|---|---|
| Enable MPG height control | qtplasmac.ext_height_ovr_count_enable |
| MPG height change | 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 |
|---|---|
| Probe Test | qtplasmac.ext_probe |
| Torch Pulse | qtplasmac.ext_pulse |
| Ohmic Test | qtplasmac.ext_ohmic |
| Change Consumables | 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 Hide Program Buttons

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 Mode 0 Arc OK

Mode 0 Arc OK relies on the arc voltage to set the Arc OK signal. This is accomplished by sampling the arc voltage every servo thread cycle. There needs to be a specified number of consecutive samples, all within a specified threshold for the Arc OK signal to be set. These voltages are also required to be within a specified range.

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
```

**Warning**
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.

Table 10.47: Different states QtPlasmaC could encounter

| Состояние | Имя | Description |
|---|---|---|
| 0 | IDLE | idle and waiting for a start command |
| 1 | PROBE_HEIGHT | move down to probe height |
| 2 | PROBE_DOWN | probe down until material sensed |
| 3 | PROBE_UP | probe up until material not sensed, this sets the zero height |
| 4 | ZERO_HEIGHT | not used at present |
| 5 | PIERCE_HEIGHT | move up to pierce height |
| 6 | TORCH_ON | turn the torch on |
| 7 | ARC_OK | wait until arc OK detected |
| 8 | PIERCE_DELAY | wait for pierce delay time |
| 9 | PUDDLE_JUMP | xy motion begins, move to puddle jump height |
| 10 | CUT_HEIGHT | move to cut height |
| 11 | CUT_MODE_01 | cutting in either mode 0 or mode 1 |
| 12 | CUT_MODE_2 | cutting in mode 2 |
| 13 | PAUSE_AT_END | pause motion at end of cut |
| 14 | SAFE_HEIGHT | move to safe height |
| 15 | MAX_HEIGHT | move to maximum height |
| 16 | END_CUT | end the current cut |
| 17 | END_JOB | end the current job |
| 18 | TORCHPULSE | a torch pulse is active |
| 19 | PAUSED_MOTION | cut recovery motion is active while paused |
| 20 | OHMIC_TEST | an ohmic test is active |
| 21 | PROBE_TEST | a probe test is active |
| 22 | SCRIBING | a scribing job is active |
| 23 | CONSUMABLE_CHANGE_ON | move to consumable change coordinates |
| 24 | CONSUMABLE_CHANGE_OFF | return from consumable change coordinates |
| 25 | CUT_RECOVERY_ON | cut recovery is active |
| 26 | CUT_RECOVERY_OFF | cut recovery is deactivated |

The DEBUG state is for testing purposes only and will not normally be encountered.

#### 10.8.15.14 QtPlasmaC Debug Print

The plasmac HAL component has a HAL pin named **plasmac.debug-print** which if set to 1 (true) will print to terminal every state change as a debug aid.

#### 10.8.15.15 Hypertherm PowerMax Communications

Communications can be established with a Hypertherm PowerMax plasma cutter that has a RS485 port. This feature enables the setting of **Cut Mode**, **Cutting Amperage** and **Gas Pressure** automatically from the **Cut Parameters** of the material file. In addition, the user will be able to view the PowerMax's **Arc On Time** in hh:mm:ss format on the STATISTICS Tab.

If **Gas Pressure** is set to zero, then the PowerMax will automatically calculate the required pressure from the **Cut Mode**, **Cut Current**, torch type, and torch length.

Changing the cutting mode will set the gas pressure to zero causing the machine to use its automatic gas pressure mode.

The maximum and minimum values of these parameters are read from the plasma cutter and the related spin-buttons in the Cut Parameters are then limited by these values. Gas pressure cannot be changed from zero until communications have been established.

This feature is enabled by setting the correct port name for the PM_PORT option in the **[POWERMAX]** section of the <*machine_name*>.prefs file. If the PM_PORT option is not set in the <*machine_name*>.prefs file then the widgets associated with this feature will not be visible.

Example showing enabling the Hypertherm PowerMax Communications on USB0:

```
[POWERMAX]
Port = /dev/ttyusb0
```

If the user is unsure of the name of the port, there is a Python script in the configuration directory that will show all available ports and can also be used to test communications with the plasma unit prior to enabling this feature in the QtPlasmaC GUI.

To use the test script, follow these instructions:

For a package installation (Buildbot) enter the following command in a terminal window:

```
pmx485-test
```

For a run in place installation enter the following two commands in a terminal window:

```
source ~/linuxcnc-dev/scripts/rip-environment
pmx485-test
```

The gas pressure units display (psi or bar) is determined by the data received during initial setup of the communication link and is then shown next to the Gas Pressure setting in the MATERIAL section of the PARAMETERS Tab.

The PowerMax machine will go into remote mode after communications have been established and may only be controlled remotely (via the QtPlasmaC GUI) at this point. The connection can be validated by observing the PowerMax display.

To switch the PowerMax back to local mode the user can either:

1. Disable PowerMax Comms from the MAIN Tab
2. Close LinuxCNC which will put the PowerMax into local mode during shutdown.
3. Turn the PowerMax off for 30 seconds and then power it back on.

**Tip**
If PowerMax communications is active then selecting Mesh Mode will automatically select CPA mode on the PowerMax unit.

**Note**
To use the PowerMax communications feature it is necessary to have the Python pyserial module installed.
If pyserial is not installed an error message will be displayed.

To install pyserial, enter the following command into a terminal window:

```
sudo apt install python3-serial
```

A typical connection diagram is shown in the appendix of this document as well as confirmed working interfaces.

#### 10.8.15.16 Moving Pierce

A moving pierce allows the torch to move during the pierce delay period. This has an advantage in that it allows for thicker materials to be pierced than can be achieved with a stationary pierce. It may also support longer consumable life by allowing a style of motion that helps prevent molten material being sprayed up into the torch nozzle.

Through the use of M159 a moving pierce can be configured.

The syntax for the M159 command is as follows:

```
M159 Pn Qn
```

| **Action Code (P)** | **Действие** | **Description** | **Value (Q)** |
|---|---|---|---|
| 601 | Pierce Type | 0=Normal, 1=Wiggle, 2=Ramp | 0,1,2 |
| 602 | Pierce Motion Delay | Delay before Z motion starts to Pierce End Height. Expressed as a % of Pierce Delay. | Целое от 0 до 100 |
| 603 | Pierce End Height | Target pierce height at end of Pierce Delay. Normally lower than Pierce Height. Expressed in machine units. | Float |
| 604 | Cut Height Delay | Delay at the end of transition to Pierce End Height before transition to Cut Height. Expressed in seconds. | Float |
| 605 | Скорость зарезки | Скорость выемки. Выражается в единицах станка/мин. | Float |
| 606 | Gouge Distance | Длина бороздки. Выражается в единицах измерения станка. | Float |
| 607 | Скорость замедления | Velocity of creep which takes effect after gouge has finished. Expressed in machine units. | Float |
| 608 | Creep Distance | Длина ползучести. Выражается в единицах измерения станка. | Float |

| Action Code (P) | Действие | Description | Value (Q) |
|---|---|---|---|
| 609 | Reset | Resets the values for action codes 601-608 back to 0, returning to default behaviour. | Not Required |

Доступны следующие модели подвижного прожига:

The model supported is the same as that created by Sheetcam’s wiggle pierce. Given a straight lead-in to the main cut the wiggle pierce is expected to move back and forth for some distance along the lead-in. Strictly speaking this straight movement is arbitrary. Technically any X/Y motion is available during the pierce delay, and it is up the CAM tooling or the user to program.

The constraint is that this motion is expected to be completed during the pierce delay value. If not then the torch will transition to normal cut height on completion of pierce delay and potentially before the wiggle motion is completed.

Therefore, the length of the wiggle and the feed rate need to be considered in calculating the pierce delay, or the size of the wiggle constrained based on feed rate and pierce delay.

Например:

- A feed rate of 1080 mm/min (18 mm/s).
- A wiggle movement of 4 mm for 3 oscillations.

This means that the length of the wiggle is 4 x 3 = 12 mm. At the 18 mm/s feed rate, the pierce delay needs to be approx 0.7 seconds to support the wiggle distance at pierce height.

The G-code needed invoke this behaviour is:

```
M159 P601 Q1
```

The G-code needed to reset to standard behaviour is:

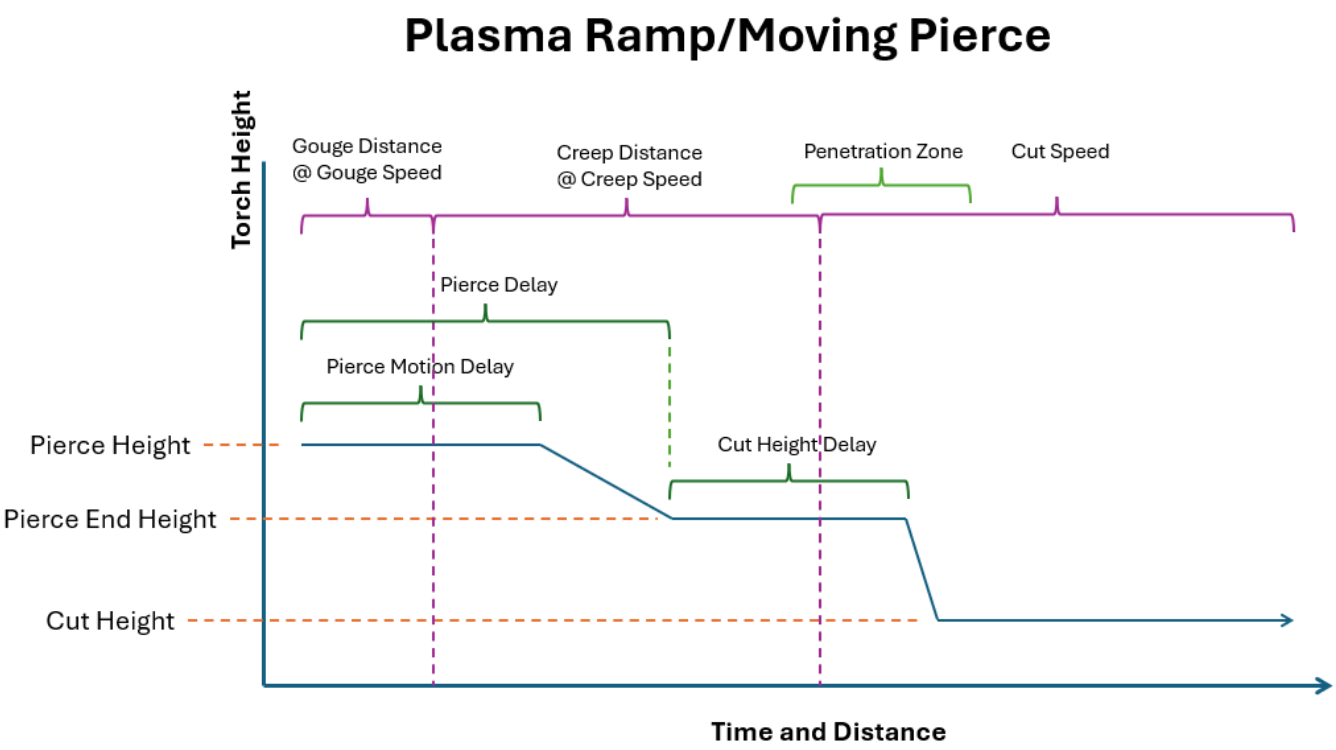
```
M159 P609
```

A ramping pierce combines a range of parameters so as to generate a sloped trough that causes the molten material to be evacuated. The resulting evacuation of material is sometimes likened to a ”rooster tail” as it is very directional. Careful consideration of the lead-in can allow evacuation of material in a safe direction for workers and machine components.

As the elements combine to drive the shape of the ramping pierce it is key that all of these elements are carefully considered when designing the ramp pierce and the parameters that set it. Inevitably there will need to be experimentation to build recipes that work with the plasma power source being used in conjunction with the material to be cut.

Моменты, которые следует учитывать:

1. The gouge and creep speeds and distances in relation to the sum of Pierce Delay from the material and the Cut Height Delay action.
2. Pierce height from the material and the End Pierce Height action in relation to Pierce Delay from the material and the speeds in effect over the various distances during the Pierce Delay time.
3. The plasma source manufacturer’s cut charts for the material type and thickness.




As with wiggle pierce it is up the CAM tooling or the user to program a ramping pierce.
Below is a sample of code to setup a ramp pierce:

```
(o=0,kw=2, ph=4, pd=1, ch=1.5, fr=490, th=1, cv=99, pe=0.3, jh=0, jd=0)

M159 P601 Q2
M159 P602 Q50
M159 P603 Q2.5
M159 P604 Q1
M159 P605 Q980
M159 P606 Q5
M159 P607 Q245
M159 P608 Q3
```

Этот код показывает нам следующую информацию по порядку:

1. Material magic comment.
   - Pierce Height is 4 mm.
   - Pierce Delay is 1 second.
   - Cut Height is 1.5 mm.
   - Cut Feed Rate is 490 mm/min.
2. Mode is set to 2 which is ramp pierce.
3. Pierce Motion Delay is 50% of Pierce Delay (0.5 seconds).
4. Pierce End Height is 2.5 mm
5. Cut Height Delay is 1 second.

6. Gouge Speed is 980 mm/min.
7. Gouge Distance is 5 mm
8. Creep Speed is 245 mm/min.
9. Creep Distance is 3 mm.

With this information the following behavior can be described:

It is important to note that accelerations and decelerations are omitted from the following calculations.

The torch will start at Pierce Height (4 mm from the material) and start traveling at a Gouge Speed of 980 mm/min (16.3 mm/s) for a Gouge Distance of 5 mm which will consume 0.3 seconds (5 mm / 16.3 mm/s = 0.3 s) of the 0.5 s Pierce Motion Delay.

When the Gouge Distance is reached, the torch speed is set to a Creep Speed of 245 mm/min (4 mm/s) for a Creep Distance of 3 mm. The Creep Distance will take roughly 0.7 seconds to complete (3 mm / 4 mm/s = 0.7 s).

The torch height will remain at 4 mm for another 0.2 seconds (0.5 s (Pierce Motion Delay) - 0.3 s (Gouge Distance at Gouge Speed) = 0.2 s) after which the torch will begin descending to a Pierce End Height of 2.5 mm over the remaining 0.5 seconds of the material's Pierce Delay. Since there are 0.5 seconds of the material's Pierce Delay remaining, as well as 0.5 seconds left at Creep Speed, the Creep Distance will be covered at the same time the Pierce End Height is reached.

When the Creep Distance has been reached, the torch speed will be set to the material's Cut Feed Rate of 490 mm/min. Since there is a 1 second Cut Height Delay that started at the end of the material's Pierce Delay the transition to a Cut Height of 1.5 mm will occur after the remaining 1 second Cut Height Delay has expired.

The above text should demonstrate that there is quite a bit of configuration and subtlety can be achieved through experimentation and the careful use of different parameter combinations.

### 10.8.16 Internationalisation

It is possible to create translation files for QtPlasmaC to display in the language of the current locale.

To create and or edit a translation file requires that LinuxCNC has been installed as run in place.

The following assumes that the LinuxCNC git directory is ~/linuxcnc-dev.

All language files are kept in ~/linuxcnc-dev/share/screens/qtplasmac/languages.

The qtplasmac.py file is a Python version of the GUI file used for translation purposes.

The .ts files are the translation source files for the translations. These are the files that require creating/editing for each language.

The .qm files are the compiled translation files used by PyQt.

The directories qtplasmac_4x3/languages and qtplasmac_9x16/languages are only for links to the .qm files in qtplasmac/languages.

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
```

Change to the languages directory:

```
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
```

The user can either create a new translation source file for a non-existing language translation or modify an existing translation source file due to changes being made to some text in a QtPlasmaC source file. If modifying an existing translation that has had no source file changes then this step is not required.

Create or edit a .ts file:

```
./langfile xx
```

---

**Note**
this command is a script which runs the following: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

---

The editing of the translation is done with the linguist application:

```
linguist
```

1. Open the TS file and translate the strings

It is not necessary to provide a translation for every text string, if no translation is specified for a string, then the original string will be used in the application. The user needs to be careful with the length of strings that appear on widgets as space is limited. If possible try, to make the translation no longer than the original.

When editing is complete save the file:
`File -> Save`

Then create the .qm file:
`File -> Release`

Close linguist.

Then create links to the compiled .qm file for the other QtPlasmaC GUIs.

```
$ ./langlink xx
```

---

**Note**
This command is a script which creates a link in both qtplasmac_4x3/languages and qtplasmac_9x16/languages to the above .qm file and then renames the link to match the GUI name.

---

QtPlasmaC will be translated to the language of the current locale on the next start so long as a .qm file exists in that language.

Users are welcome to submit translation files for inclusion into QtPlasmaC. An easy method is to post the up to date qtplasmac_xx.ts file on the forum and the maintainers will install the translations.

The preferred method is to submit a pull request from the users GitHub account as described in the contributing to LinuxCNC documentation. The files required to be committed are qtplasmac_xx.ts and qtplasmac_xx.qm in the qtplasmc/languages directory plus the links in both the qtplasmac_4x3/languages and qtplasmac_9x16/languages directories.

## 10.8.17 Appendix

### 10.8.17.1 Примеры конфигураций

There are example configuration files which use the QtPlasmaC GUI to simulate plasma cutting machines.

They can be found in the LinuxCNC chooser under: Sample Configurations -> sim -> qtplasmac

Three versions are available in both metric and imperial units:

1. qtplasmac_l - 16:9 format, minimum resolution 1366x768
2. qtplasmac_p - 9:16 format, minimum resolution 786x1366
3. qtplasmac_s - 4:3 format, minimum resolution 1024x768

Each sample configuration includes a popup control panel to simulate various inputs to the GUI such as:

1. ARC VOLTAGE
2. OHMIC SENSE
3. FLOAT SWITCH
4. BREAKAWAY SWITCH
5. ESTOP

### 10.8.17.2 NGC Samples

There are some sample G-code files in the ~/linuxcnc/nc_files/examples/plasmac directory.

### 10.8.17.3 QtPlasmaC Specific G-codes

| **Description** | **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.` |
| Wait for material change confirmation. | `M66 PG 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) |

| Description | Code |
|---|---|
| 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) |
| 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

| Description | Пример |
|---|---|
| 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 |
| Set velocity to 40% of CutFeedRate | M67 E3 Q40 |
| Cut a hole with 60% reduced speed using velocity setting | G21 (metric)<br>G64 P0.05<br>M52 P1 (enable adaptive feed)<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) |
| Cut a hole with 60% reduced speed using the #<holes> command | G21 (metric)<br>G64 P0.05<br>M52 P1 (enable adaptive feed)<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) |
| Cut a hole with over cut using torch disable | G21 (metric)<br>G64 P0.05<br>M52 P1 (enable adaptive feed)<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) |

| Description | Пример |
|---|---|
| Cut a hole with over cut using the #<holes> command | G21 (metric)<br>G64 P0.05<br>M52 P1 (enable adaptive feed)<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 (enable adaptive feed)<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) |
| Select scribe and select torch at end of scribing | .<br>.<br>M52 P1 (enable adaptive feed)<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) |
| Hole center spotting. | (Requires a small motion command or nothing happens)<br>G21 (metric)<br>F99999 (high feed rate)<br>G0 X10 Y10<br>M3 $2 S1 (spotting on)<br>G91 (relative distance mode)<br>G1 X0.000001<br>G90 (absolute distance mode)<br>M5 $2 (spotting off)<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) |
| Edit material, if not existing create a new one | (o=2, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000, kw=1.0) |

#### 10.8.17.5 Mesa THCAD

The Mesa THCAD is a common way of obtaining the arc voltage from a plasma cutter and is also useful for ohmic sensing of the material during probing. The THCAD may be used for parallel port configurations as well as configurations using Mesa Electronics hardware. The THCAD is available in three different models, THCAD-5, THCAD-10, and THCAD-300.

There is a mode jumper on each THCAD card which should be set to **UNIPOLAR**

There is a frequency divider jumper on each THCAD card which should be set according to the hardware type:

| Input Device | Recommended Setting |
|---|---|
| Parallel Port with very low latency | F/32 |
| Parallel Port recommended starting point | F/64 |
| Parallel Port with higher latency, or when cutting thick material | F/128 |
| Mesa Card | F/32 |

This value is required to be entered into PnCConf during installation.

**Note**
If using a parallel port it may be necessary for the user to adjust the jumper setting and the subsequent scaling values on the Parameters Tab to achieve optimal results. Symptoms may include random torch raises or dives during otherwise stable cutting. Halscope plots may be useful in diagnosing these issues.

Located on the rear of the THCAD is a calibration sticker showing:

```
   THCAD-nnn

0V  121.1 kHz
5V  925.3 kHz
```

or similar values, these values are required to be entered into PnCConf during installation.

PnCConf has entries for all required THCAD parameters and will calculate and configure any required settings. The calculations used are as follows:

**Voltage Scale**

```
vs = r / ((f - z) / d / v)
```

**Voltage Offset**

```
vo = z / d
```

$r$ = divider ratio (see below).

$f$ = full scale value from calibration sticker.

$z$ = 0 V value from calibration sticker.

$d$ = value from jumper above.

$v$ = full scale voltage of THCAD

**Divider Ratio** *THCAD-5 or THCAD-10*

If connecting to a plasma CNC port, then the divider ratio is selected from the plasma machine. A common ratio used is 20:1.

If connecting to the plasma machines full arc voltage, then a common setup for a THCAD-10 is to use a 1 MΩ resistor from arc negative to THCAD negative and a 1 MΩ resistor from arc positive to THCAD positive. The divider ratio is obtained by:

```
r = (total_resistance + 100000) / 100000
```

*THCAD-300*

```
r = 1
```

**Important**
IF THE USER IS USING A HF START PLASMA POWER SUPPLY, THEN EACH OF THESE RESISTANCES SHOULD BE MADE UP OF SEVERAL HIGH VOLTAGE RESISTORS.

**Caution**
IF THE USER IS USING A HF START PLASMA POWER SUPPLY, THEN OHMIC SENSING IS NOT RECOMMENDED.

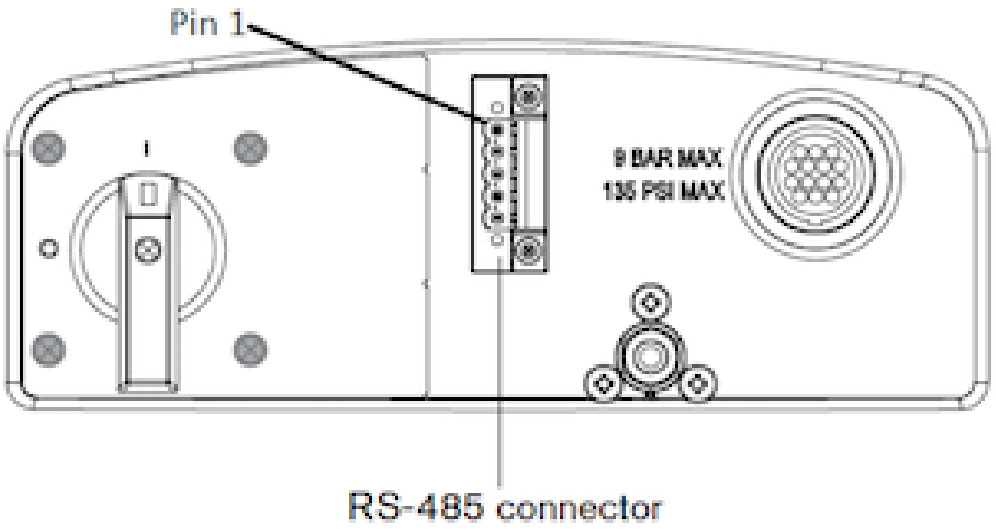
**Note**
These values can be calculated by using this online calculator.

**Note**
There is a lowpass filter available which may be useful if using a THCAD and there is a lot of noise on the returned arc voltage.

#### 10.8.17.6 RS485 Connections

Hypertherm RS485 Wiring Diagram (wire colors inside the Hypertherm in parentheses):

| Connection at Machine Pin # | Connection at Breakout Board |
|---|---|
| **1** - Tx+ (Red) | ->RXD+ |
| **2** - Tx- (Black) | ->RXD- |
| **3** - Rx+ (Brown) | ->T/R+ |
| **4** - Rx- (White) | ->T/R- |
| **5** - GND (Green) | ->GND |



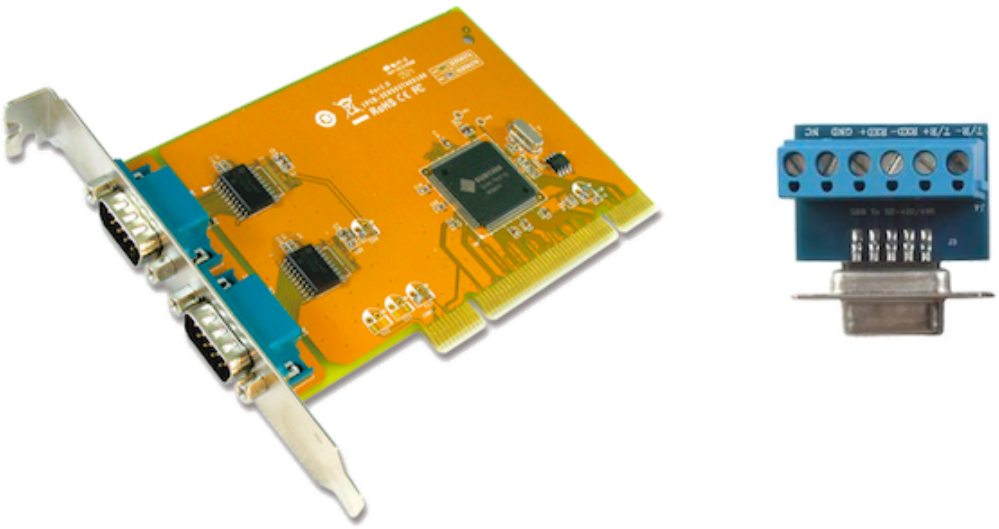
RS485 interfaces that are known to work:

DTECH DT-5019 USB to RS-485 converter adapter:

The following is necessary to convert a motherboard Serial connection or Serial card (RS232) to RS485:

DTECH RS-232 to RS-485 converter:

Serial card example (Sunnix SER5037A PCI Card shown with Breakout Board):

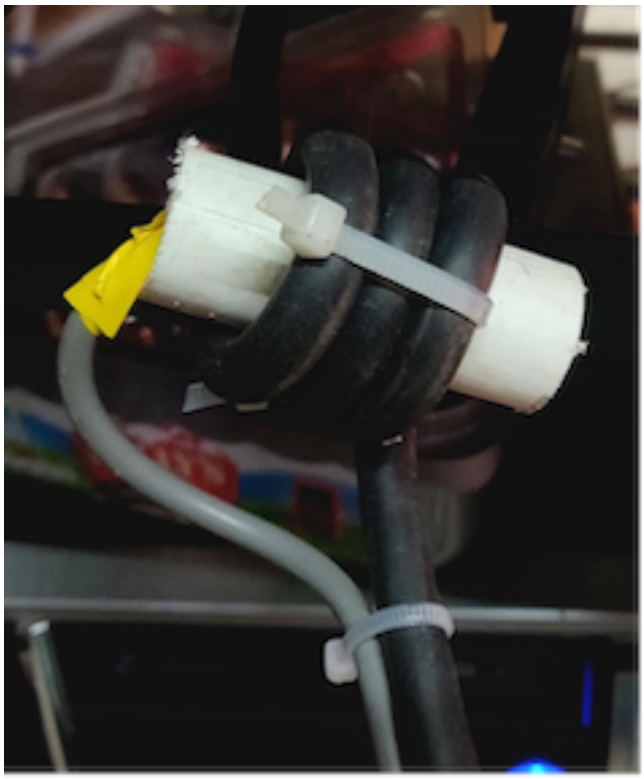


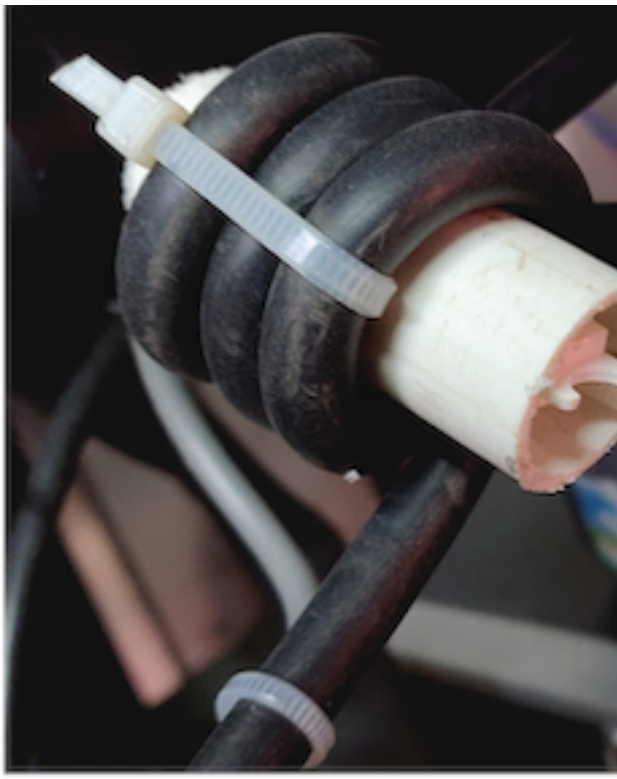
#### 10.8.17.7 Arc OK With A Reed Relay

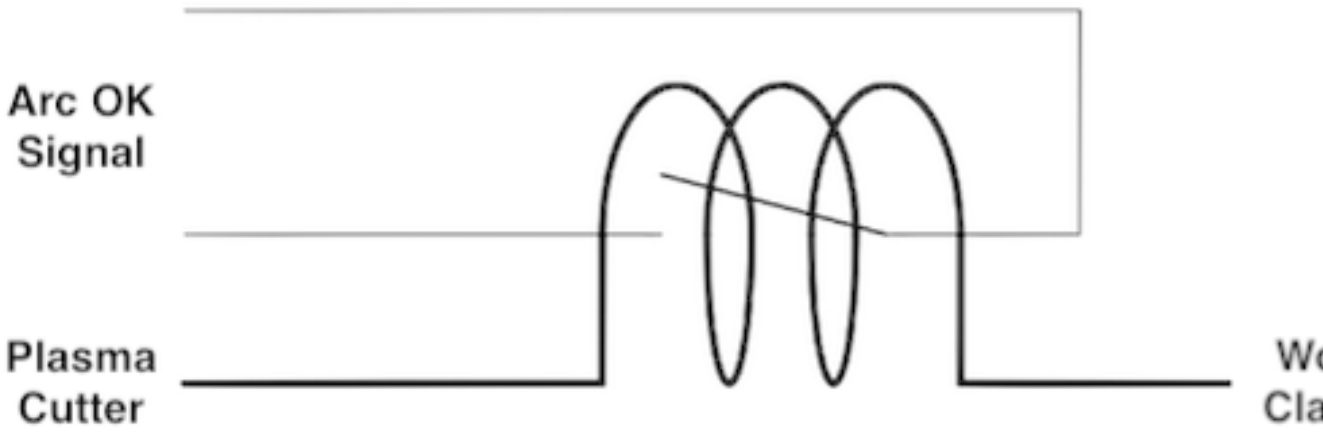
An effective and very reliable method of obtaining an Arc OK signal from a plasma power supply without a CNC port is to mount a reed relay inside a non-conductive tube and wrap and secure three turns of the work lead around the tube.

This assembly will now act as a relay that will switch on when current is flowing through the work lead which only occurs when a cutting arc has been established.

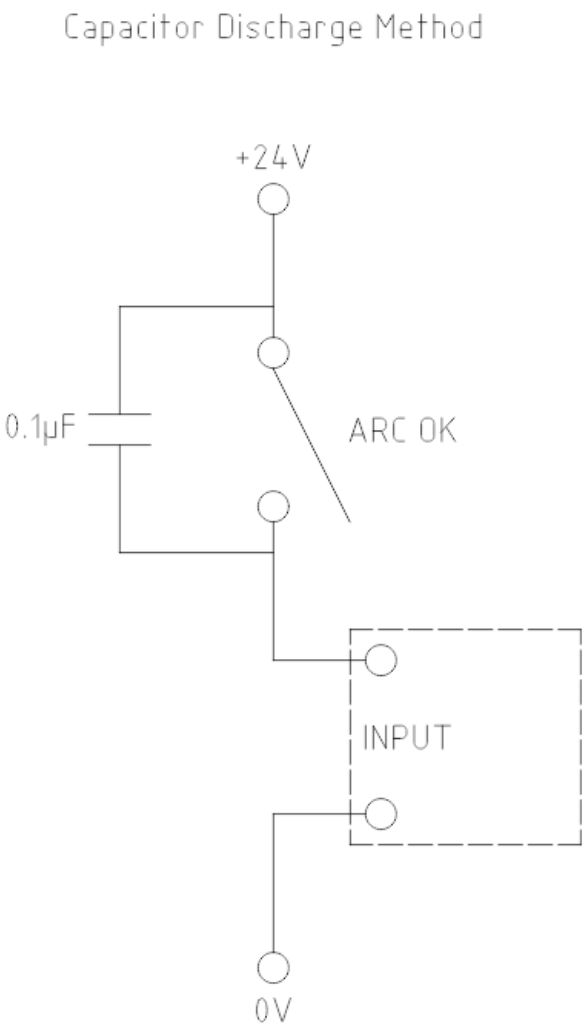
This will require that QtPlasmaC be operated in Mode 1 rather than Mode 0. See the QtPlasmaC Modes sections for more information.

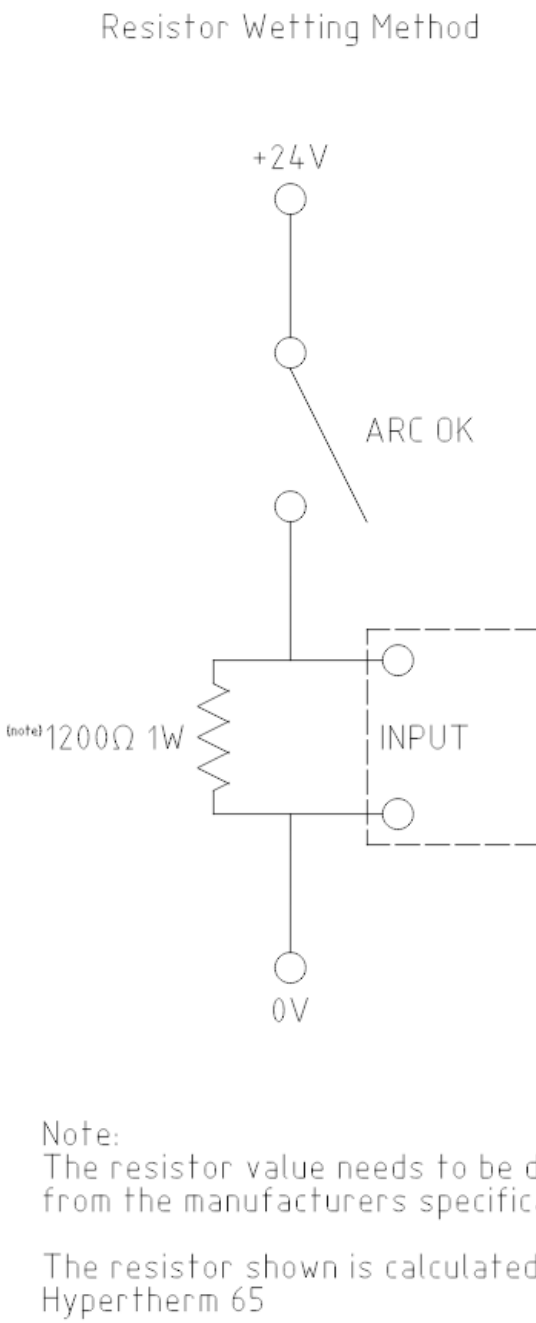





Arc OK
Signal
Plasma
Cutter

#### 10.8.17.8 Contact Load Schematics




A full description is at Contact Load.

### 10.8.18 Known Issues

#### 10.8.18.1 Keyboard Jogging

There is a known issue with some combinations of hardware and keyboards that may affect the autorepeat feature of the keyboard and will then affect keyboard jogging by intermittent stopping and starting during jogging. This issue can be prevented by disabling the Operating System's autorepeat feature for all keys. QtPlasmaC uses this disabling feature by default for all keys only when the MAIN

Tab is visible, with the following exceptions when autorepeat is allowed with the MAIN Tab visible: G-code editor is active, MDI is active. When QtPlasmaC is shut down, the Operating System's autorepeat feature will be enabled for all keys.

If the user wishes to prevent QtPlasmaC from changing the Operating System's autorepeat settings, enter the following option in the **[GUI_OPTIONS]** section of the <*machine_name*>.prefs file:

```
Autorepeat all == True
```

This issue does not affect any jogging using the GUI jog buttons.

---

**Note**
Disconnecting and reconnecting a keyboard during an active QtPlasmaC session will cause the autorepeat feature to re-enable itself automatically which may cause intermittent stopping and starting during jogging. The user must restart QtPlasmaC to disable the autorepeat feature again.

---

#### 10.8.18.2 NO_FORCE_HOMING

QtPlasmaC does not currently adhere to the following stanza in the <*machine_name*>.ini file:

```
NO_FORCE_HOMING = 1
```

Regardless of this setting, QtPlasmaC requires that the machine must be homed before executing MDI commands or running programs.

### 10.8.19 Contributing Code To QtPlasmaC

Bugfixes and enhancements to QtPlasmaC are always welcome. The preferred method to contribute code is to submit a pull request (PR) comprised of a single commit to the LinuxCNC GitHub repository. For more information on creating a PR, see the LinuxCNC documentation, the only pre-requisite is that you sign up for a GitHub account. All PR's are verified and then committed by one of the developers. If you are uncomfortable with submitting a PR then attaching the code changes on a LinuxCNC Forum thread is an acceptable method.

Bugfixes are accepted for both the latest released branch and master branch. If a bugfix applies to both branches, then it is only necessary to submit a PR for the latest released branch as it will be merged into master branch by a developer.

Enhancements are accepted for master branch only.

Every PR, except for changes to the QtPlasmaC documentation only, requires that the appropriate version number is incremented and also that the version history is updated. Version numbers are located in the following locations:

| **Расположение** | **Формат** | **инкрементирован когда** |
|---|---|---|
| src/hal/components/plasmac.comp | nnn | component code changes |
| share/qtvcp/screens/qtplasmac/qtplasmac_handler.py | nnn.*nnn* | component code changes<br>*GUI code changes* |

The version history is located at share/qtvcp/screens/qtplasmac/versions.html.

### 10.8.20 Support

Online help and support is available from the PlasmaC section of the 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 Введение

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 is mouse and touch screen friendly.

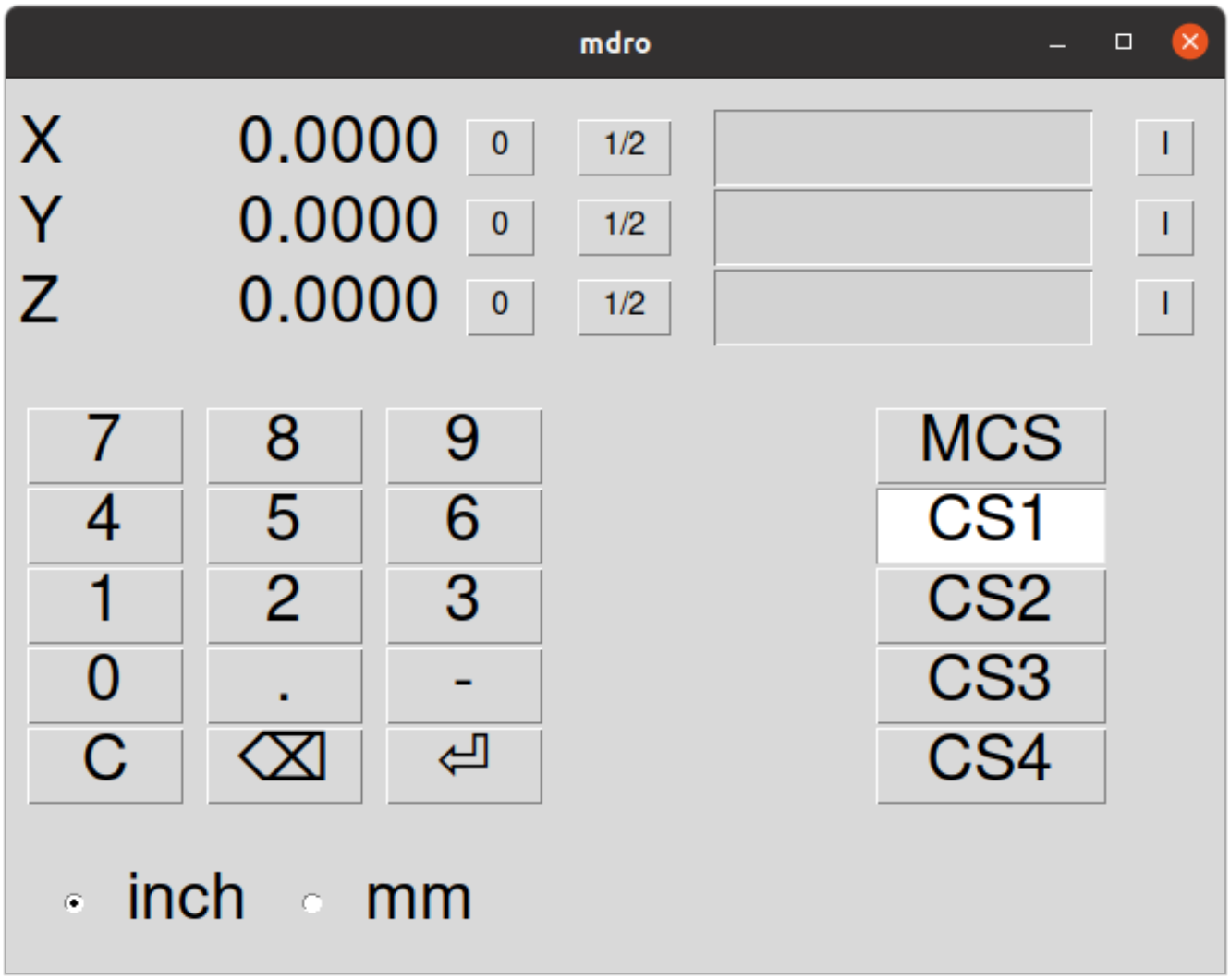



Figure 10.52: MDRO Window

## 10.9.2 Начало

If your configuration is not currently set up to use MDRO, you can change it by editing the INI file. In the `[DISPLAY]` section, change the `DISPLAY =` line to read `DISPLAY = mdro`. MDRO defaults to XYZ for the axes but that can be changed. Set `[DISPLAY]` section `GEOMETRY = XYZ` for a 3 axis mill. A lathe with with DRO scales on the X and Z axes might use `GEOMETRY = XZ`.

When MDRO starts, a window like the one in the figure Figure 10.52 above opens.

### 10.9.2.1 INI File Options

Other options that can be included on the `[DISPLAY]` section include:

- `MDRO_VAR_FILE = <file.var>` - preload G54 - G57 coordinate system data.
  - Preload a .var file. This is typically the .var file used by the operational code.
- `POINT_SIZE = <n>` - Set text point size.

  - This option sets the size of the font used which sets the overall size of the window. The default point size is 20, Typical sizes are 20 to 30.

- `MM = 1` Set this if the DRO scales provide data scaled in millimeters.

#### 10.9.2.2 Command Line Options

MDRO can be started by a `loadusr` command in a HAL file. Options equivalent to those in the INI file can be set on the command line:

- `-l <file.var>` - preload G54 - G57 coordinate system data.
- `-p <n>` - Set text point size.
- `-m` - Set this if the DRO scales provide data scaled in millimeters.
- `<axes>` - axes to display. See `GEOMETRY` above.

#### 10.9.2.3 Контакты

Using an example of ”XYZA” for an AXES argument, these pins will be created when MDRO starts:

```
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 this example, the first row of the display will be labeled `X` and will show the data from the DRO scale connected to pin `mdro.axis.0`. The `mdro.index-enable.n` pins should be connected to the index pins of the DRO if the DRO supports them.

The pins must be connected in the file specified in the `POSTGUI_HALFILE` entry of the INI file when the program is started from an INI file. They can be set directly after the loadusr command if the program is started in a HAL file.

### 10.9.3 MDRO Window

The MDRO window contains the following elements:

- A row for each axis. Each row includes:
  - the name of the axis,
  - the current value,
  - a ”z” button that zeros the value,
  - a ”1/2” button that halves the value,
  - a entry field that can be used to set a user-defined value. This field can be set from the keyboard or from the on-screen keypad.
  - A ”I” button that starts an index operation (see below),
- a keypad used to set values in the entry field via a mouse or touchscreen,

- coordinate system selection buttons:
  - The ”mcs” button selects the machine coordinate system. These are the raw values from the encoders connected to the `mdro.axis.n` pins.
  - The ”cs1” - ”cs4” buttons allow the user to select among one of four user-defined coordinate systems. If the program is started with the `MDRO_VAR_FILE =` option, the labels will be changed to ”g54” - ”g57” and the values from the specified .var file will be preloaded. Note that any changes to the values are not persistent: the .var file is never changed.
- Inch/Millimeter selection buttons.

### 10.9.4 Index operations

`MDRO` supports DRO scales with index marks. Hit the ”I” button on the axis row then crank the axis to the index position. The machine coordinate will be zeroed. This is easiest to see at startup or when the ”mcs” coordinate system has been selected.

### 10.9.5 Simulation

The easiest way to see how `MDRO` works is to try it in a simulation environment. Add this section to the end of your simulation HAL file, usually ”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
```

# Chapter 11

# Программирование в G-кодах

## 11.1 Coordinate Systems

### 11.1.1 Введение

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.

We will also show that it is very interesting to use a reference point on the blank or the part and to make the program work from this point, without having to take into account where the part is placed on the table.

This chapter introduces you to offsets as they are used by the LinuxCNC. These include:

- Machine Coordinates (G53)
- Nine Coordinate System Offsets (G54-G59.3)
- Global Offsets (G92) and Local Offsets (G52)

### 11.1.2 Machine Coordinate System

When LinuxCNC is started the positions of each axis is the machine origin. Once an axis is homed, the machine origin for that axis is set to the homed position. The machine origin is the machine coordinate system on which all other coordinate systems are based. The G53 G-code can be used to move in the machine coordinate system.

#### 11.1.2.1 Machine coordinates moves: G53

Regardless of any offset that may be active, a G53 in a line of code tells the interpreter to move to the actual axes positions (absolute positions) specified. For example:

```
G53 G0 X0 Y0 Z0
```

will move from the current position to the position where the machine coordinates of the three axes will be at zero. You can use this command if you have a fixed position for the tool change or if your machine has an automatic tool changer. You can also use this command to clear the work area and access the workpiece in the vise.

G53 is a non modal command. It must be used in every block where a move in machine coordinate system is desired.

### 11.1.3 Coordinate Systems

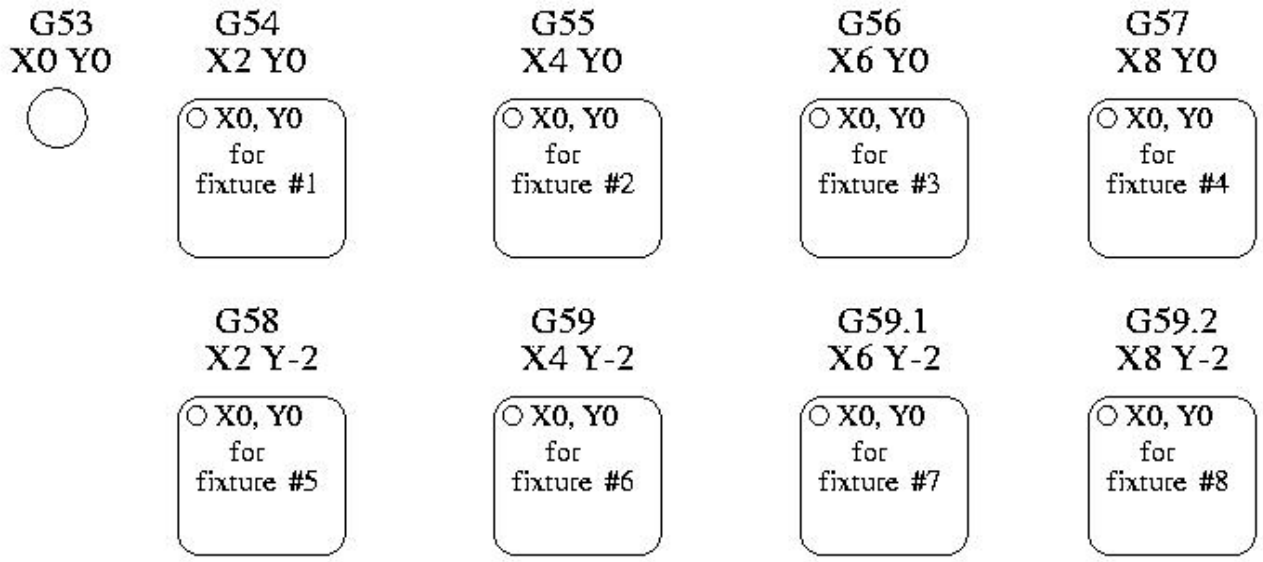



Figure 11.1: Coordinate Systems Example

Coordinate System Offsets

- G54 - use coordinate system 1
- G55 - use coordinate system 2
- G56 - use coordinate system 3
- G57 - use coordinate system 4
- G58 - use coordinate system 5
- G59 - use coordinate system 6
- G59.1 - use coordinate system 7
- G59.2 - use coordinate system 8
- G59.3 - use coordinate system 9

Coordinate system offsets are used to shift the coordinate system from the machine coordinate system. This allows the G-code to be programmed for the part without regard to the part location on the machine. Using coordinate system offsets would allow you to machine parts in multiple locations with the same G-code.

The values for offsets are stored in the VAR file that is requested by the INI file during the startup of an LinuxCNC. In the example below, which uses G55, the position of each axis for G55 origin is stored in a numbered variable.

In the VAR file scheme, the first variable number stores the X offset, the second the Y offset and so on for all nine axes. There are numbered sets like this for each of the coordinate system offsets.

Each of the graphical interfaces has a way to set values for these offsets. You can also set these values by editing the VAR file itself and then restart LinuxCNC so that the LinuxCNC reads the new values however this is not the recommended way. Using G10, G52, G92, G28.1, etc are better ways to set the variables. For our example, we will directly edit the file so that G55 will take the following values:

Table 11.1: Example of G55 parameters

| Axis | Variable | Value |
|---|---|---|
| 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 |

You should read this as moving the zero positions of G55 to X = 2 units, Y= 1 unit, and Z = -2 units away from the absolute zero position.

Once there are values assigned, a call to G55 in a program block would shift the zero reference by the values stored. The following line would then move each axis to the new zero position. Unlike G53, G54 through G59.3 are modal commands. They will act on all blocks of code after one of them has been set. The program that might be run using fixture offsets would require only a single coordinate reference for each of the locations and all of the work to be done there. The following code is offered as an example of making a square using the G55 offsets that we set above.

```
G55 ; use coordinate system 2
G0 X0 Y0 Z0
G1 F2 Z-0.2000
X1
Y1
X0
Y0
G0 Z0
G54 ; use coordinate system 1
G0 X0 Y0 Z0
M2
```

In this example the G54 near the end leaves the G54 coordinate system with all zero offsets so that there is a modal code for the absolute machine based axis positions. This program assumes that we have done that and use the ending command as a command to machine zero. It would have been possible to use G53 and arrive at the same place but that command would not have been modal and any commands issued after it would have returned to using the G55 offsets because that coordinate system would still be in effect.

[source,”]

```
G54     uses parameters of coordinate system 1
G55     uses parameters of coordinate system 2
G56     uses parameters of coordinate system 3
G57     uses parameters of coordinate system 4
G58     uses parameters of coordinate system 5
G59     uses parameters of coordinate system 6
G59.1   uses parameters of coordinate system 7
G59.2   uses parameters of coordinate system 8
G59.3   uses parameters of coordinate system 9
```

#### 11.1.3.1 Default Coordinate System

One other variable in the VAR file becomes important when we think about offset systems. This variable is named 5220. In the default files its value is set to 1.00000. This means that when the

LinuxCNC starts up it should use the first coordinate system as its default. If you set this to 9.00000 it would use the ninth offset system as its default for start up and reset. Any value other than an integer (decimal really) between 1 and 9, or a missing 5220 variable will cause the LinuxCNC to revert to the default value of 1.00000 on start up.

#### 11.1.3.2 Setting Coordinate System Offsets

The G10 L2x command can be used to set coordinate system offsets:

- *G10 L2 P(1-9)* - Set offset(s) to a value. Current position irrelevant (see G10 L2 for details).
- *G10 L20 P(1-9)* - Set offset(s) so current position becomes a value (see G10 L20 for details).

---

**Note**
We only give a brief overview here, refer to the G-code sections for a full description.

---

### 11.1.4 Local and Global Offsets

#### 11.1.4.1 The G52 command

*G52* is used in a part program as a temporary "local coordinate system offset" within the workpiece coordinate system. An example use case is when machining several identical features at different locations on a part. For each feature, *G52* programs a local reference point within workpiece coordinates, and a subprogram is called to machine the feature relative to that point.

*G52* axis offsets are programmed relative to workpiece coordinate offsets *G54* through *G59.3*. As a local offset, *G52* is applied after the workpiece offset, including rotation. Thus, a part feature will be machined identically on each part regardless of the part's orientation on the pallet.

---

**Caution**
As a temporary offset, set and unset within the localized scope of a part program, in other G-code interpreters *G52* does not persist after machine reset, *M02* or *M30*. In LinuxCNC, *G52* shares parameters with *G92*, which, for historical reasons, **does** persist these parameters. See G92 Persistence Cautions below.

---

**Caution**
*G52* and *G92* share the same offset registers. Therefore, setting *G52* will override any earlier *G92* setting, and *G52* will persist across machine reset when *G92* persistence is enabled. These interactions may result in unexpected offsets. See G92 and G52 Interaction Cautions below.

---

Programming *G52 X1 Y2* offsets the current workpiece coordinate system X axis by 1 and Y axis by 2. Accordingly, on the DRO, the current tool position's X and Y coordinates will be reduced by 1 and 2, respectively. Axes unset in the command, such as Z in the previous example, will be unaffected: any previous *G52* Z offset will remain in effect, and otherwise the Z offset will be zero.

The temporary local offset may be canceled with *G52 X0 Y0*. Any axes not explicitly zeroed will retain the previous offset.

*G52* shares the same offset registers as *G92*, and thus *G52* is visible on the DRO and preview labeled with *G92*.

### 11.1.5 G92 Axes Offsets

G92 is the most misunderstood and cleverest command programmable with LinuxCNC. The way it works has changed a bit between the first versions and the current one. These changes have doubt baffled many users. They should be seen as a command producing a temporary offset, which applies to all the other offsets.

#### 11.1.5.1 The G92 commands

*G92* is typically used in two conceptually different ways: as a ”global coordinate system offset” or as a ”local coordinate system offset”.

The *G92* set of commands includes:

- *G92* - This command, when used with axis names, sets values to offset variables.
- *G92.1* - This command sets zero values to the G92 variables.
- *G92.2* - This command suspends but does not zero out the G92 variables.
- *G92.3* - This command applies offset values that have been suspended.

As a global offset, *G92* is used to shift all workpiece coordinate systems *G54* through *G59.3*. An example use case is when machining several identical parts in fixtures with known locations on a pallet, but the pallet location may change between runs or between machines. Each fixture location offset, relative to a reference point on the pallet, is preset in one of the workpiece coordinate systems, *G54* through *G59.3*, and *G92* is used to ”touch off” on the pallet reference point. Then, for each part, the corresponding workpiece coordinate system is selected and the part program is executed.

---

**Note**
*G10 R-* workpiece coordinate system rotation is specific to the *rs274ngc* interpreter, and the *G92* offset is applied *after* rotation. When using *G92* as a global offset, workpiece coordinate system rotations may have unexpected results.

---

As a local coordinate system, *G92* is used as a temporary offset within the workpiece coordinate system. An example use case is when machining a part with several identical features at different locations. For each feature, *G92* is used to set a local reference point, and a subprogram is called to machine the feature starting at that point.

---

**Note**
The use of *G92* is discouraged for programming with local coordinate systems in a part program. Instead, see G52, a local coordinate system offset more intuitive when desired offset relative to the workpiece is known but current tool location may not be known.

---

Programming *G92 X0 Y0 Z0* sets the current tool location to the coordinates X0, Y0, and Z0, without motion. G92 **does not** work from absolute machine coordinates. It works from **current location**.

*G92* also works from current location as modified by any other offsets that are in effect when the *G92* command is invoked. While testing for differences between work offsets and actual offsets it was found that a *G54* offset could cancel out a *G92* and thus give the appearance that no offsets were in effect. However, the *G92* was still in effect for all coordinates and did produce expected work offsets for the other coordinate systems.

By default, *G92* offsets are restored after the machine is started. Programmers that wish for Fanuc behavior, where *G92* offsets are cleared at machine start and after a reset or program end, may disable *G92* persistence by setting *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.

**Note**
It is good practice to clear the *G92* offsets at the end of their use with *G92.1* or *G92.2*. When starting up LinuxCNC with *G92* persistence enabled (the default), any offsets in the *G92* variables will be applied when an axis is homed. See G92 Persistence Cautions below.

#### 11.1.5.2 Setting G92 Values

There are at least two ways to set G92 values:

- With a right click on the position displays in tklinuxcnc, a window opens where it is possible to enter a value.
- With the G92 command

Both work from the current position of the axis that should be moved.

Programming *G92 X Y Z A B C U V W* sets the values of the G92 variables so that each axis takes the value associated with its name. Those values are assigned to the current position of the axes. These results satisfy to paragraphs one and two of the NIST document.

G92 commands work from current axis location and add and subtract correctly to give the current axis position the value assigned by the G92 command. The effects work even though previous offsets are in.

So if the X axis is currently showing 2.0000 as its position a *G92 X0* will set an offset of -2.0000 so that the current location of X becomes zero. A *G92 X2* will set an offset of 0.0000 and the displayed position will not change. A *G92 X5.0000* will set an offset of 3.0000 so that the current displayed position becomes 5.0000.

#### 11.1.5.3 G92 Persistence Cautions

By default, the values of a *G92* offset will be saved in the VAR file and be restored after a machine reset or startup.

The G92 parameters are:

- 5210 - Enable/disable flag (1.0/0.0)
- 5211 - X Axis Offset
- 5212 - Y Axis Offset
- 5213 - Z Axis Offset
- 5214 - A Axis Offset
- 5215 - B Axis Offset
- 5216 - C Axis Offset
- 5217 - U Axis Offset
- 5218 - V Axis Offset
- 5219 - W Axis Offset

where 5210 is the *G92* enable flag (1 for enabled, 0 for disabled) and 5211 to 5219 are the axis offsets. If you are seeing unexpected positions as the result of a commanded move, as a result of storing an offset in a previous program and not clearing them at the end then issue a G92.1 in the MDI window to clear the stored offsets.

If G92 values exist in the VAR file when LinuxCNC starts up, the G92 values in the var file will be applied to the values of the current location of each axis. If this is home position and home position is set as machine zero everything will be correct. Once home has been established using real machine switches, or by moving each axis to a known home position and issuing an axis home command, any G92 offsets will be applied. If you have a G92 X1 in effect when you home the X axis the DRO will read *X: 1.000* instead of the expected *X: 0.000* because the G92 was applied to the machine origin. If you issue a G92.1 and the DRO now reads all zeros then you had a G92 offset in effect when you last ran LinuxCNC.

Unless your intention is to use the same G92 offsets in the next program, the best practice is to issue a G92.1 at the end of any G code files where you use G92 offsets.

When a program is aborted during processing that has *G92* offsets in effect a startup will cause them to become active again. As a safeguard, always have your preamble to set the environment as you expect it. Additionally, *G92* persistence may be disabled by setting *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.

#### 11.1.5.4 G92 and G52 Interaction Cautions

*G52* and *G92* share the same offset registers. Unless *G92* persistence is disabled in the INI file (see G92 Commands), *G52* offsets will also persist after machine reset, *M02* or *M30*. Beware that a *G52* offset in effect during a program abort may result in unintended offsets when the next program is run. See G92 Persistence Cautions above.

### 11.1.6 Sample Programs Using Offsets

#### 11.1.6.1 Sample Program Using Workpiece Coordinate Offsets

This sample engraving project mills a set of four .1 radius circles in roughly a star shape around a center circle. We can setup the individual circle pattern like this.

```
G10 L2 P1 X0 Y0 Z0 (ensure that G54 is set to machine zero)
G0 X-0.1 Y0 Z0
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

We can issue a set of commands to create offsets for the four other circles like this.

```
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)
```

We put these together in the following program:

```
(a program for milling five small circles in a diamond shape)

G10 L2 P1 X0 Y0 Z0 (ensure that G54 is machine zero)
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
```

```
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)

G54 G0 X-0.1 Y0 Z0 (center circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G55 G0 X-0.1 Y0 Z0 (first offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G56 G0 X-0.1 Y0 Z0 (second offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G57 G0 X-0.1 Y0 Z0 (third offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G58 G0 X-0.1 Y0 Z0 (fourth offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0

M2
```

Now comes the time when we might apply a set of G92 offsets to this program. You'll see that it is running in each case at Z0. If the mill were at the zero position, a G92 Z1.0000 issued at the head of the program would shift everything an inch. You might also shift the whole pattern around in the XY plane by adding some X and Y offsets with G92. If you do this you should add a G92.1 command just before the M2 that ends the program. If you do not, other programs that you might run after this one will also use that G92 offset. Furthermore it would save the G92 values when you shut down the LinuxCNC and they will be recalled when you start up again.

#### 11.1.6.2 Sample Program Using G52 Offsets

(To be written)

## 11.2 Компенсация инструмента

### 11.2.1 Касание

Using the Touch Off Screen in the AXIS interface you can update the tool table automatically.

Typical steps for updating the tool table:

- After homing load a tool with *T*n *M6* where *n* is the tool number.
- Move tool to an established point using a gauge or take a test cut and measure.
- Click the *Touch Off* button in the Manual Control tab (or hit the *End* button on your keyboard).
- Select *Tool Table* in the Coordinate System drop down box.

- Enter the gauge or measured dimension and select OK.

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*.

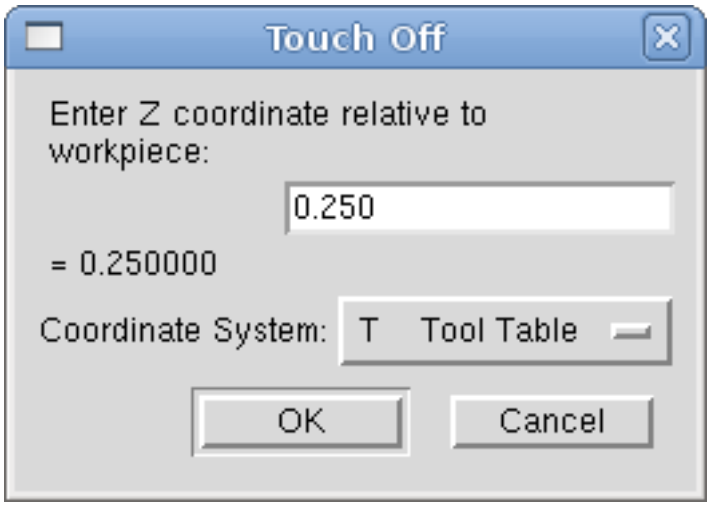



Figure 11.2: Touch Off Tool Table

#### 11.2.1.1 Using G10 L1/L10/L11

The G10 L1/L10/L11 commands can be used to set tool table offsets:

- `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).

---

**Note**
This is only a brief presentation, refer to the reference guide of the G-code for more detailed explanations.

---

### 11.2.2 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. 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 Tool Table Format

.Tool Формат таблицы

| T# | P# | X | Y | Z | A | B | C | U | V | W | Dia | FA | BA | Ori | Rem |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ; | (no data after opening semicolon) | | | | | | | | | | | | | | |
| 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 |

In general, the tool table line format is:

- T - tool number (tool numbers must be unique)
- P - pocket number, 1-1000 (pocket numbers must be unique, Pocket 0 represents the spindle)
- X..W - tool offset on specified axis - floating-point
- D - tool diameter - floating-point, absolute value
- I - front angle (lathe only) - floating-point
- J - back angle (lathe only) - floating-point
- Q - tool orientation (lathe only) - integer, 0-9
- ; - beginning of comment or remark - text

Tool numbers should be unique. Lines beginning with a semicolon are ignored.

The units used for the length, diameter, etc., are in machine units.

You will probably want to keep the tool entries in ascending order, especially if you are going to be using a randomizing tool changer. Although the tool table does allow for tool numbers in any order.

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.

The meanings of the entries and the type of data to be put in each are as follows.

**Tool Number (required)**
: 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.

**Pocket Number (required)**
: 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.

**Data Offset Numbers (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).

Смотрите Информация для пользователя токарного станка главу для дополнительной информации.

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.

---

**Note**
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.

---

#### 11.2.2.2 Инструментальный ввод/вывод

The non-realtime program **iocontrol** is conventionally used for tool changer management (and other io functions for enabling LinuxCNC and the control of coolant 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***.

The io program provides HAL output pins to facilitate toolchanger management:

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. Tool number *n*==0 indicates no tool.
2. The pocket number for a tool is set when tooldata is loaded/reloaded from its data source ([EMCIO]TOOL_TABLE or [EMCIO]DB_PROGRAM).
3. At G-code **T*n*** (*n* != 0) command:
   a. **iocontrol.0.tool-prep-index** = *idx* (index based on tooldata load sequence)
   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = the pocket number for *n*
4. At G-code **T0** (*n* == 0 remove) command:
   a. **iocontrol.0.tool-prep-index** = 0
   b. **iocontrol.0.tool-prep-number** = 0
   c. **iocontrol.0.tool-prep-pocket** = 0
5. At M-code **M6** (following iocontrol.0.tool-changed pin 0-->1):
   a. **iocontrol.0.tool-from-pocket** = pocket number used to retrieve tool

1. Tool number *n*==0 is **not special**.
2. Pocket number 0 is **special** as it indicates the **spindle**.
3. The **current** pocket number for tool *n* is the tooldata index (*idx*) for tool *n*.
4. At G-code command **T*n***:
   a. **iocontrol.0.tool-prep-index** = tooldata index (idx) for tool *n*
   b. **iocontrol.0.tool-prep-number** = *n*
   c. **iocontrol.0.tool-prep-pocket** = номер кармана для инструмента *n*
5. At M-code **M6** (following iocontrol.0.tool-changed pin 0-->1):
   a. **iocontrol.0.tool-from-pocket** = pocket number used to retrieve tool

---

**Note**
At startup, **iocontrol.0.tool-from-pocket** = 0. An M61Q*n* (*n*!=0) command does not change the **iocontrol.0.tool-from-pocket**. An M61Q0 (*n*==0) command sets **iocontrol.0.tool-from-pocket** to 0.

---

#### 11.2.2.3 Устройства смены инструмента

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.

**Manual Tool Changer** 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
```

**Устройство смены инструмента с фиксированным расположением** 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.

---

**Note**
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`.

---

**Random Location Tool Changers** 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.

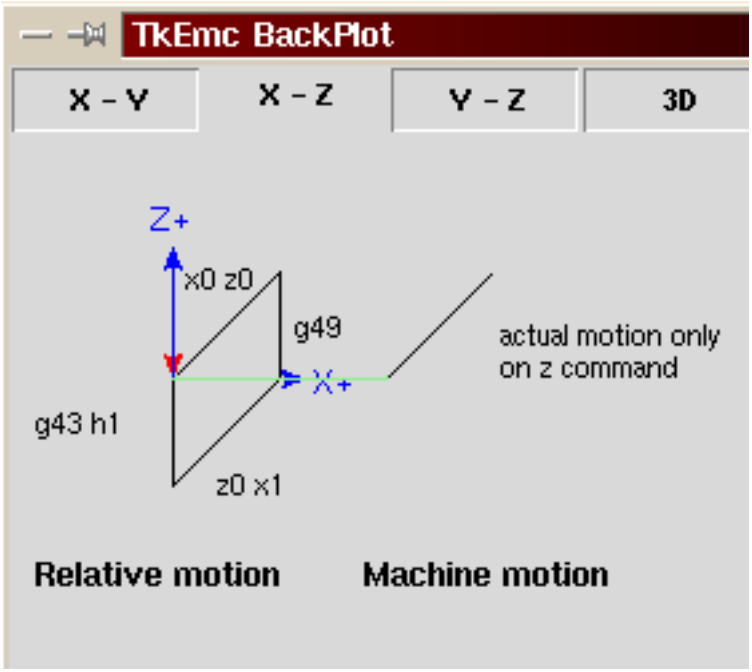
### 11.2.3 Компенсация длины инструмента

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.

**Tool length test program. Tool #1 is one inch long.**

```
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
```




With this program, in most cases, the machine will apply the offset in the form of a ramp during the movement in xyz following the word G43.

### 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.

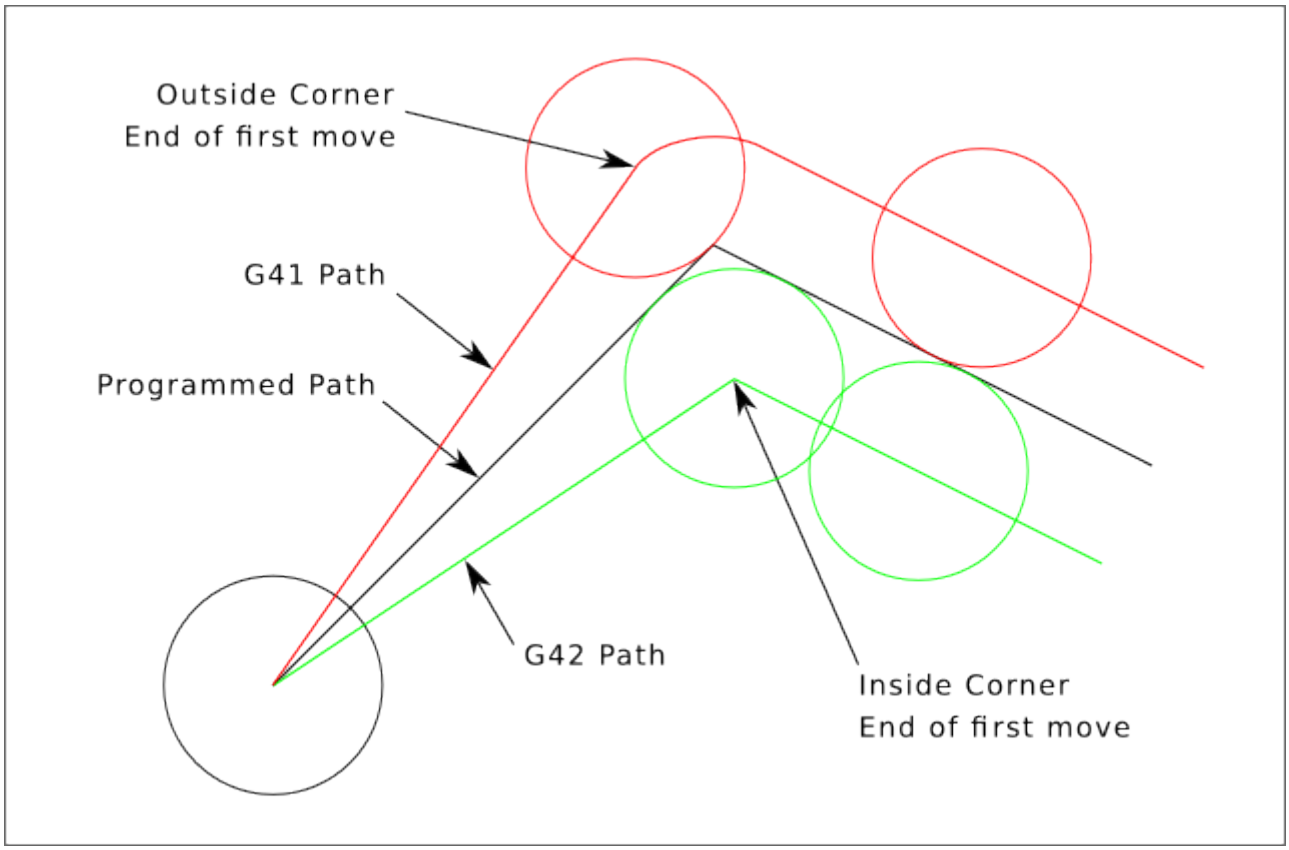



Figure 11.3: Compensation End Point

#### 11.2.4.1 Обзор

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.

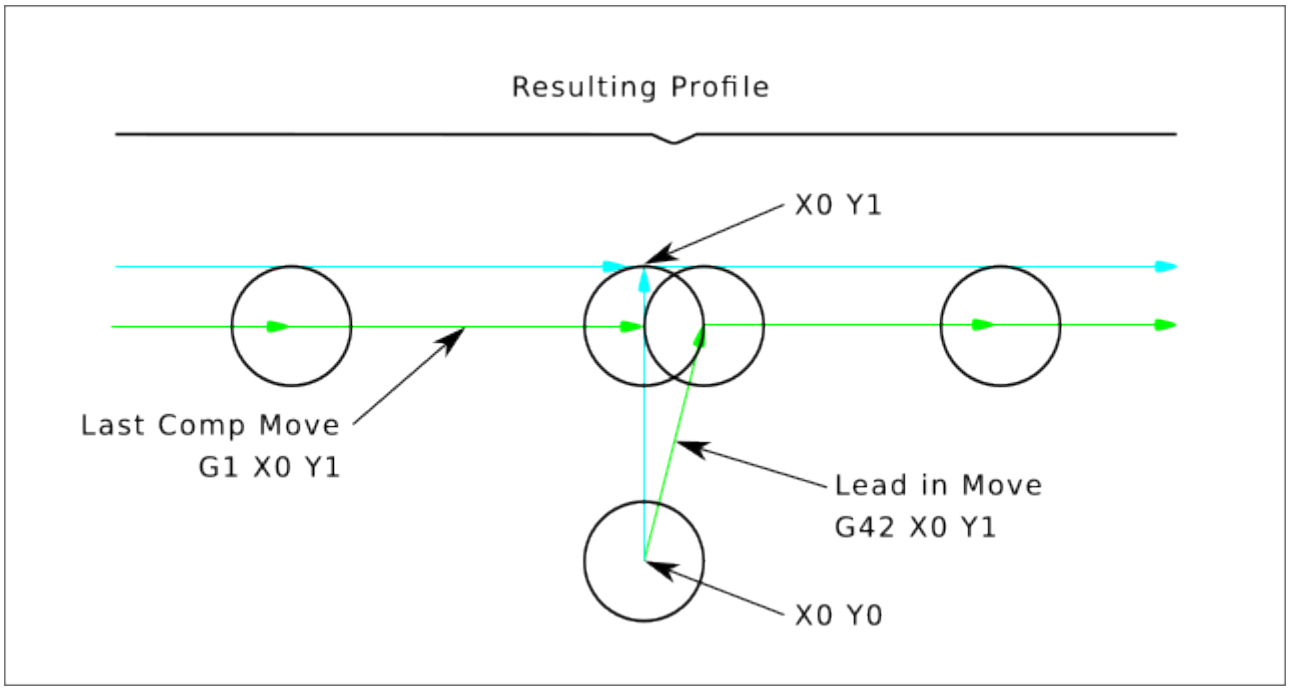



Figure 11.4: Entry Move

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.

Rapid moves may be programmed while compensation is turned on.

Start a program with G40 to make sure compensation is off.

#### 11.2.4.2 Examples

```
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 )
```

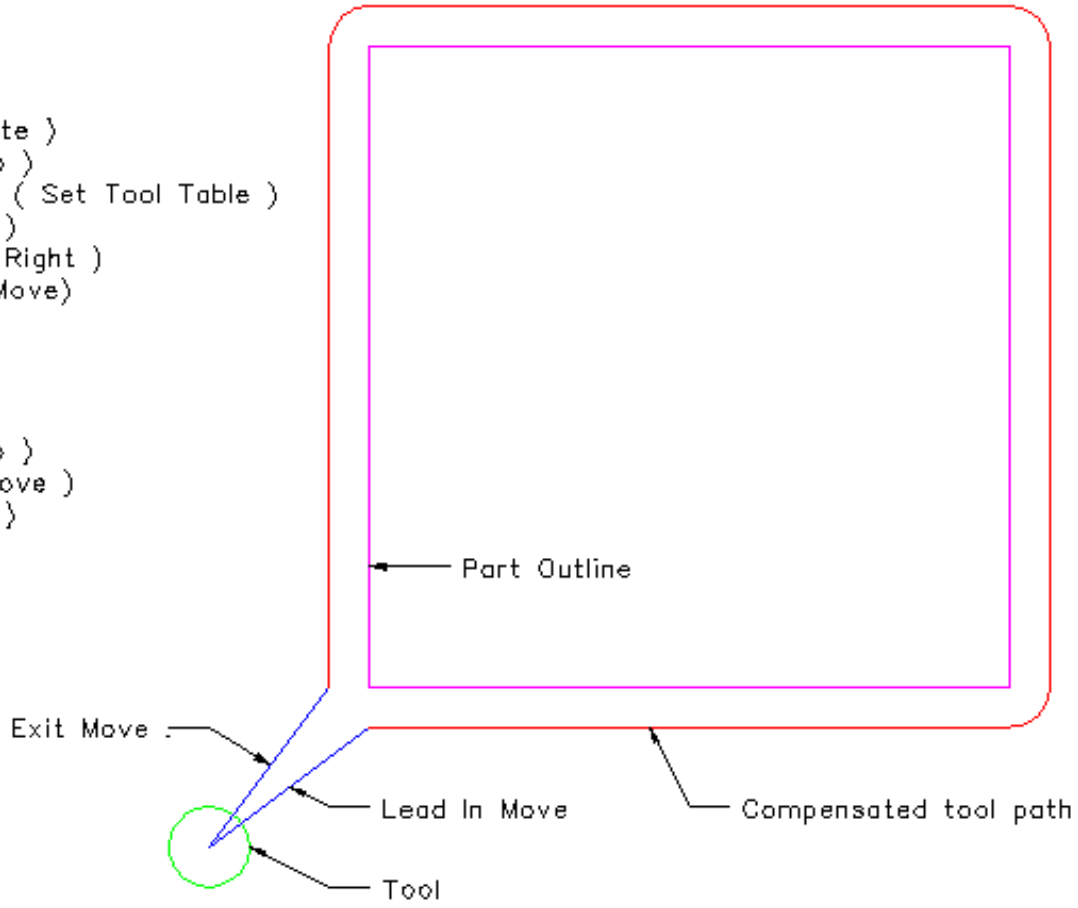



Figure 11.5: Outside Profile

```
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 )
```

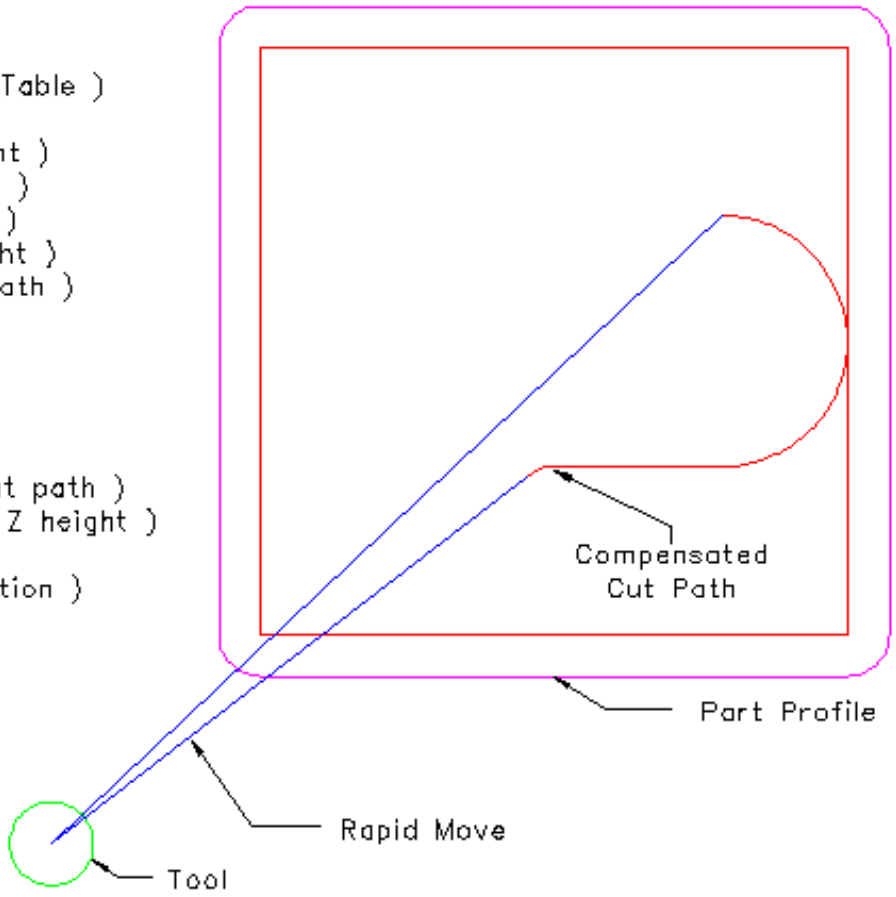



Figure 11.6: Inside Profile

## 11.3 Tool Edit GUI

### 11.3.1 Обзор

**Note**
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.

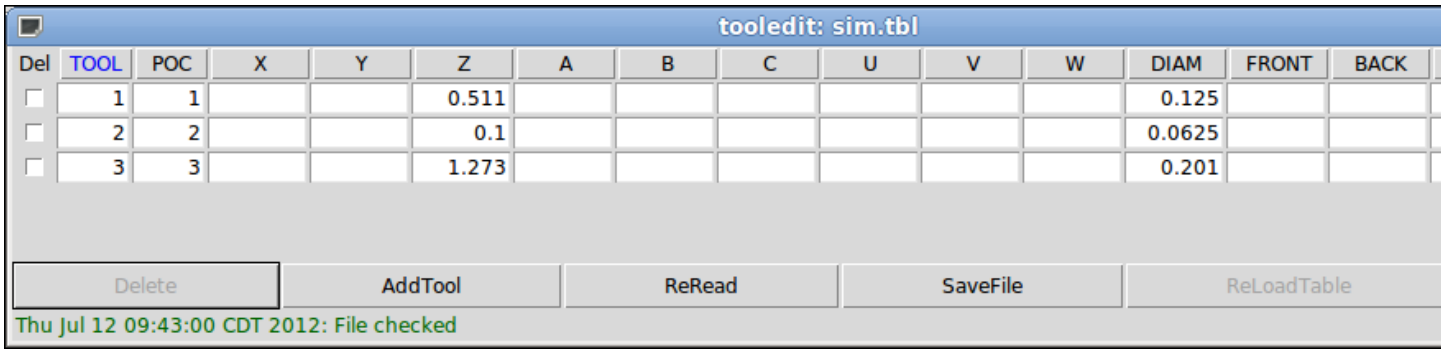



Figure 11.7: Tool Edit GUI - Overview

The *tooledit* program can update the tool table file with edited changes by using the SaveFile button. The SaveFile button updates the system file but a separate action is required to update the tool table data used by a running LinuxCNC instance. With the AXIS GUI, both the file and the current tool table data used by LinuxCNC can be updated with the ReloadTable button. This button is enabled only when the machine is ON and IDLE.

### 11.3.2 Column Sorting

The tool table display can be sorted on any column in ascending order by clicking on the column header. A second click sorts in descending order. Column sorting requires that the machine is configured with the default Tcl version >= 8.5.

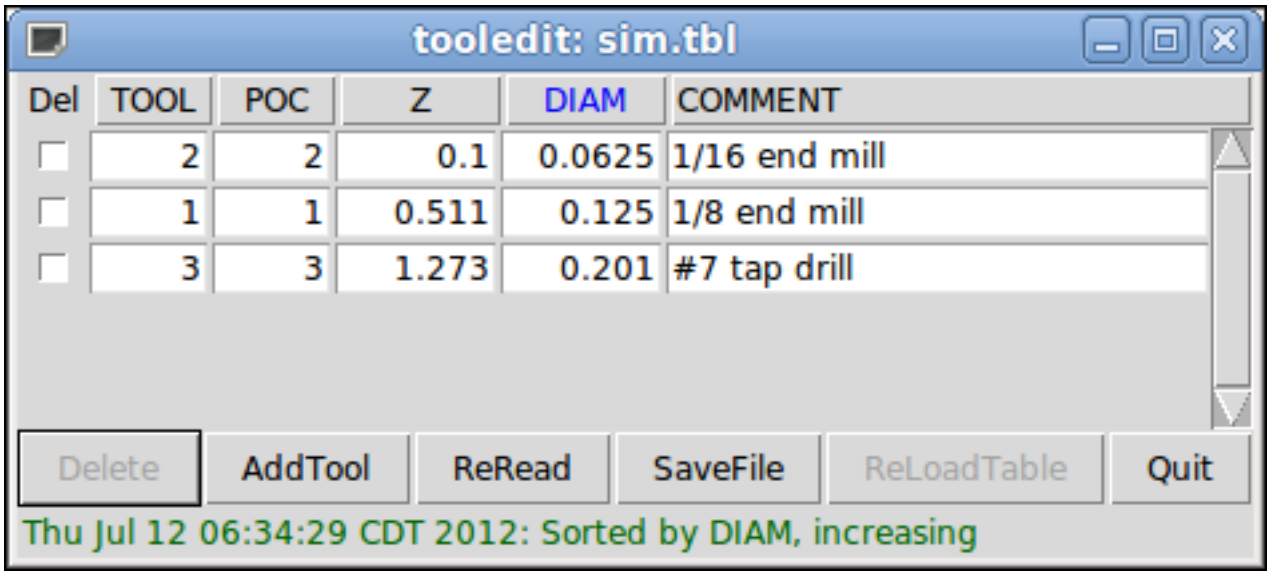



Figure 11.8: Tool Edit GUI - Column Sorting

In Ubuntu Lucid 10.04 Tcl/Tk8.4 it is installed by default. The installation is performed as follows:

```
sudo apt-get install tcl8.5 tk8.5
```

Depending upon other applications installed on the system, it may be necessary to enable Tcl/Tk8.5 with the commands:

```
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 Columns Selection

By default, the *tooledit* program will display all possible tool table parameter columns. Since few machines use all parameters, the columns displayed can be limited with the following INI file setting:

**Syntax of INI file**

```
[DISPLAY]
TOOL_EDITOR = tooledit column_name column_name ...
```

**Example for Z and DIAM columns**

```
[DISPLAY]
TOOL_EDITOR = tooledit Z DIAM
```

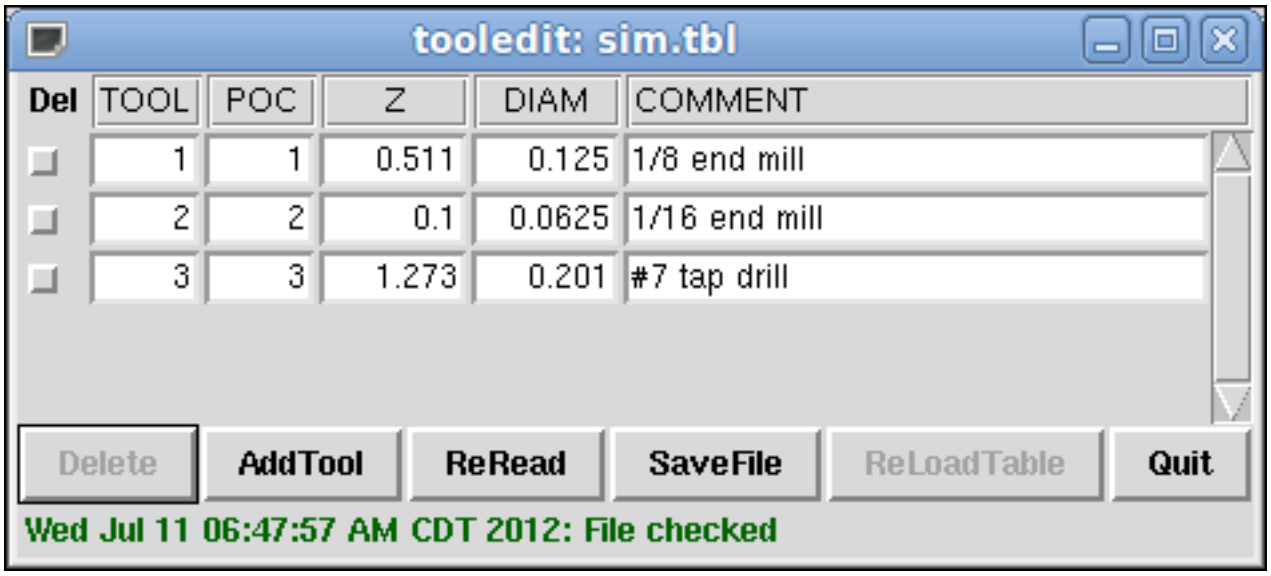



Figure 11.9: Tool Edit GUI - Columns Selection Example

### 11.3.4 Stand Alone Use

The *tooledit* program can also be invoked as a standalone program. For example, if the program is in the user PATH, typing *tooledit* will show the usage syntax:

**Автономность**

```
tooledit
Usage:
       tooledit filename
       tooledit [column_1 ... column_n] filename

Valid column names are: x y z a b c u v w diam front back orien
```

To synchronize a standalone *tooledit* with a running LinuxCNC application, the filename must resolve to the same [EMCIO]TOOL_TABLE filename specified in the LinuxCNC INI file.

When using the program *tooledit* while LinuxCNC is running, G-code command execution or other programs may alter tool table data and the tool table file. File changes are detected by *tooledit* and a message is displayed:

```
Warning: File changed by another process
```

The *tooledit* tool table display can be updated to read the modified file with the ReRead button.

The tool table is specified in the INI file with an entry:

```
[EMCIO]TOOL_TABLE = tool_table_filename
```

The tool table file can be edited with any simple text editor (not a word processor).

The AXIS GUI can optionally use an INI file setting to specify the tool editor program:

```
[DISPLAY]TOOL_EDITOR = path_to_editor_program
```

By default, the program named *tooledit* is used. This editor supports all tool table parameters, allows addition and deletion of tool entries, and performs a number of validity checks on parameter values.

## 11.4 Обзор программирования G-кода

### 11.4.1 Обзор

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*.

The LinuxCNC language has no indicator for the start of a program. The Interpreter, however, deals with files. A single program may be in a single file, or a program may be spread across several files. A file may demarcated with percents in the following way. The first non-blank line of a file may contain nothing but a percent sign, *%*, possibly surrounded by white space, and later in the file (normally at the end of the file) there may be a similar line. Demarcating a file with percents is optional if the file has an *M2* or *M30* in it, but is required if not. An error will be signaled if a file has a percent line at the beginning but not at the end. The useful contents of a file demarcated by percents stop after the second percent line. Anything after that is ignored.

The LinuxCNC G-code language has two commands (*M2* or *M30*), either of which ends a program. A program may end before the end of a file. Lines of a file that occur after the end of a program are not to be executed. The interpreter does not even read them.

### 11.4.2 Format of a line

A permissible line of input code consists of the following, in order, with the restriction that there is a maximum (currently 256) to the number of characters allowed on a line.

1. an optional block delete character, which is a slash /.
2. an optional line number.
3. Any number of:
    1. words,
    2. настройки параметров,
    3. subroutine codes, and
    4. комментарии.
4. an end of line marker (carriage return or line feed or both).

Any input not explicitly allowed is illegal and will cause the Interpreter to signal an error.

Spaces and tabs are allowed anywhere on a line of code and do not change the meaning of the line, except inside comments. This makes some strange-looking input legal. The line *G0X +0. 12 34Y 7* is equivalent to *G0 x+0.1234 Y7*, for example.

Blank lines are allowed in the input. They are to be ignored.

Input is case insensitive, except in comments, i.e., any letter outside a comment may be in upper or lower case without changing the meaning of a line.

#### 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, it is also possible to enable block delete with the following icon:

**AXIS Block Delete Icon**

#### 11.4.2.2 Необязательный номер строки

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.

---

**Note**
Line numbers are not recommended. See Best Practices.

---

#### 11.4.2.3 Words, Parameters, Subroutines, Comments

A word is a letter other than N or O (”o”) followed by a real value.

Words may begin with any of the letters shown in the following Table. The table includes N and O for completeness, even though, as defined above, line numbers and program flow parameters are not words. Several letters (I, J, K, L, P, R) may have different meanings in different contexts. Letters which refer to axis names are not valid on a machine which does not have the corresponding axis.

Table 11.3: Words and their meanings

| Letter | Meaning |
|---|---|
| A | A axis of machine |
| B | B axis of machine |
| C | C axis of machine |
| D | Tool radius compensation number |
| F | Feed rate |
| G | General function (See table G-code Modal Groups) |
| H | Tool length offset index |
| I | X offset for arcs and G87 canned cycles |
| J | Y offset for arcs and G87 canned cycles |
| K | Z offset for arcs and G87 canned cycles. |
| | Spindle-Motion Ratio for G33 synchronized movements. |
| L | generic parameter word for G10, M66 and others |
| M | Miscellaneous function (See table M-code Modal Groups) |
| N | Line number (not recommended, see Best Practices) |
| O | o-codes for program flow control (See o-Codes) |
| P | Dwell time in canned cycles and with G4. |
| | Key used with G10. |
| Q | Feed increment in G73, G83 canned cycles |
| R | Arc radius or canned cycle plane |
| S | Spindle speed |
| T | Tool selection |
| U | U axis of machine |
| V | V axis of machine |
| W | W axis of machine |
| X | X axis of machine |
| Y | Y axis of machine |
| Z | Z axis of machine |

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.

**Note**
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.

### 11.4.3 Числа

The following rules are used for (explicit) numbers. In these rules a digit is a single character between 0 and 9.

- A number consists of:
  - an optional plus or minus sign, followed by
  - zero to many digits, followed, possibly, by
  - one decimal point, followed by
  - zero to many digits - provided that there is at least one digit somewhere in the number.
- There are two kinds of numbers:
  - Integers, that does not have a decimal point,
  - Decimals, that do have a decimal point.
- Numbers may have any number of digits, subject to the limitation on line length. Only about seventeen significant figures will be retained, however (enough for all known applications).
- A non-zero number with no sign but the first character is assumed to be positive.

Notice that initial (before the decimal point and the first non-zero digit) and trailing (after the decimal point and the last non-zero digit) zeros are allowed but not required. A number written with initial or trailing zeros will have the same value when it is read as if the extra zeros were not there.

Numbers used for specific purposes in RS274/NGC are often restricted to some finite set of values or some to some range of values. In many uses, decimal numbers must be close to integers; this includes the values of indices (for parameters and carousel slot numbers, for example), M-codes, and G-codes multiplied by ten. A decimal number which is intended to represent an integer is considered close enough if it is within 0.0001 of an integer value.

### 11.4.4 Параметры

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.

Parameters differ in syntax, scope, behavior when not yet initialized, mode, persistence and intended use.

**Syntax**

There are three kinds of syntactic appearance:

- *numbered* - #4711
- *named local* - #<localvalue>
- *named global* - #<_globalvalue>

**Scope**

The scope of a parameter is either global, or local within a subroutine. Subroutine parameters and local named variables have local scope. Global named parameters and numbered parameters starting from number 31 are global in scope. RS274/NGC uses *lexical scoping* - in a subroutine only the local variables defined therein, and any global variables are visible. The local variables of a calling procedure are not visible in a called procedure.

**Behavior of uninitialized parameters**

- Uninitialized global parameters, and unused subroutine parameters return the value zero when used in an expression.

- Uninitialized named parameters signal an error when used in an expression.

**Mode**
Most parameters are read/write and may be assigned to within an assignment statement. However, for many predefined parameters this does not make sense, so they are are read-only - they may appear in expressions, but not on the left-hand side of an assignment statement.

**Persistence**
When LinuxCNC is shut down, volatile parameters lose their values. All parameters except numbered parameters in the current persistent range [1] are volatile. Persistent parameters are saved in the .var file and restored to their previous values when LinuxCNC is started again. Volatile numbered parameters are reset to zero.

**Intended Use**

- user parameters - numbered parameters in the range 31..5000, and named global and local parameters except predefined parameters. These are available for general-purpose storage of floating-point values, like intermediate results, flags etc, throughout program execution. They are read/write (can be assigned a value).
- subroutine parameters - these are used to hold the actual parameters passed to a subroutine.
- numbered parameters - most of these are used to access offsets of coordinate systems.
- system parameters - used to determine the current running version. They are read-only.

#### 11.4.4.1 Пронумерованные параметры

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.

A value is stored in a parameter with the = operator; for example:

```
#3 = 15 (set parameter 3 to 15)
```

A parameter setting does not take effect until after all parameter values on the same line have been found. For example, if parameter 3 has been previously set to 15 and the line *#3=6 G1 X#3* is interpreted, a straight move to a point where X equals 15 will occur and the value of parameter 3 will be 6.

The # character takes precedence over other operations, so that, for example, *#1+2* means the number found by adding 2 to the value of parameter 1, not the value found in parameter 3. Of course, *#[1+2]* does mean the value found in parameter 3. The # character may be repeated; for example *##2* means the value of the parameter whose index is the (integer) value of parameter 2.

- *31-5000* - G-code user parameters. These parameters are global in the G code file, and available for general use. Volatile.
- *5061-5069* - Coordinates of a G38 probe result (X, Y, Z, A, B, C, U, V & W). Coordinates are in the coordinate system in which the G38 took place. Volatile.
- *5070* - G38 probe result: 1 if success, 0 if probe failed to close. Used with G38.3 and G38.5. Volatile.
- *5161-5169* - ”G28” Home for X, Y, Z, A, B, C, U, V & W. Persistent.
- *5181-5189* - ”G30” Home for X, Y, Z, A, B, C, U, V & W. Persistent.

[1] persistent_range,The range of persistent parameters may change as development progresses. This range is currently 5161- 5390. It is defined in the *required_parameters array* in file the src/emc/rs274ngc/interp_array.cc .

[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.

- *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* - Shared ”G52” and ”G92” offset for X, Y, Z, A, B, C, U, V & W. Volatile by default; persistent if *DISABLE_G92_PERSISTENCE = 1* in the *[RS274NGC]* section of the INI file.
- *5220* - Coordinate System number 1 - 9 for G54 - G59.3. Persistent.
- *5221-5230* - Coordinate System 1, G54 for X, Y, Z, A, B, C, U, V, W & R. R denotes the XY rotation angle around the Z axis. Persistent.
- *5241-5250* - Coordinate System 2, G55 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5261-5270* - Coordinate System 3, G56 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5281-5290* - Coordinate System 4, G57 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5301-5310* - Coordinate System 5, G58 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5321-5330* - Coordinate System 6, G59 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5341-5350* - Coordinate System 7, G59.1 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5361-5370* - Coordinate System 8, G59.2 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5381-5390* - Coordinate System 9, G59.3 for X, Y, Z, A, B, C, U, V, W & R. Persistent.
- *5399* - Result of M66 - Check or wait for input. Volatile.
- *5400* - Tool Number. Volatile.
- *5401-5409* - Tool Offsets for X, Y, Z, A, B, C, U, V & W. Set by G43. Volatile.
- *5410* - Tool Diameter. Volatile.
- *5411* - Tool Front Angle. Volatile.
- *5412* - Tool Back Angle. Volatile.
- *5413* - Tool Orientation. Volatile.
- *5420-5428* - Current relative position in the active coordinate system including all offsets and in the current program units for X, Y, Z, A, B, C, U, V & W, volatile.
- *5599* - Flag for controlling the output of (DEBUG,) statements. 1=output, 0=no output; default=1. Volatile.

**Numbered Parameters Persistence** The values of parameters in the persistent range are retained over time, even if the machining center is powered down. LinuxCNC uses a parameter file to ensure persistence. It is managed by the Interpreter. The Interpreter reads the file when it starts up, and writes the file when it exits.

The format of a parameter file is shown in Table Parameter File Format.

The Interpreter expects the file to have two columns. It skips any lines which do not contain exactly two numeric values. The first column is expected to contain an integer value (the parameter’s number). The second column contains a floating point number (this parameter’s last value). The value is represented as a double-precision floating point number inside the Interpreter, but a decimal point is not required in the file.

Parameters in the user-defined range (31-5000) may be added to this file. Such parameters will be read by the Interpreter and written to the file as it exits.

Missing Parameters in the persistent range will be initialized to zero and written with their current values on the next save operation.

The parameter numbers must be arranged in ascending order. An *Parameter file out of order* error will be signaled if they are not in ascending order.

The original file is saved as a backup file when the new file is written.

Table 11.4: Parameter File Format

| Parameter Number | Parameter Value |
|---|---|
| 5161 | 0.0 |
| 5162 | 0.0 |

#### 11.4.4.2 Коды и параметры подпрограмм

Subroutine codes, or o-codes (sometimes also called o-words), provide for logic and flow control in NGC programs (as in if-else logic). They are called Subroutine codes because they can also form called subroutines (as in sub-endsub).

See the chapter on o-Codes.

**Note**
If o-codes are used to form subroutines, then o-codes can also call those subroutines and pass up to 30 parameters, which are local to the subroutine and volatile. (Again, see o-Codes for fuller treatment and examples.)

**Note**
While both lower and upper case o- are valid, best practice is using lower case ”o-” because it disambiguates 0 (zero) and O (capital o).

#### 11.4.4.3 Именованные параметры

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.

#<*named parameter*> is a local named parameter. By default, a named parameter is local to the scope in which it is assigned. You can’t access a local parameter outside of its subroutine. This means that two subroutines can use the same parameter names without fear of one subroutine overwriting the values in another.

#<*_global named parameter*> is a global named parameter. They are accessible from within called subroutines and may set values within subroutines that are accessible to the caller. As far as scope is concerned, they act just like regular numeric parameters. They are not stored in files.

Examples:

**Declaration of named global variable**

```
#<_endmill_dia> = 0.049
```

**Reference to previously declared global variable**

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

**Mixed literal and named parameters**

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]
```

Named parameters spring into existence when they are assigned a value for the first time. Local named parameters vanish when their scope is left: when a subroutine returns, all its local parameters are deleted and cannot be referred to anymore.

It is an error to use a non-existent named parameter within an expression, or at the right-hand side of an assignment. Printing the value of a non-existent named parameter with a DEBUG statement - like *(DEBUG, <no_such_parameter>)* will display the string #.

Global parameters, as well as local parameters assigned to at the global level, retain their value once assigned even when the program ends, and have these values when the program is run again.

The EXISTS function tests whether a given named parameter exists.

#### 11.4.4.4 Предопределенные именованные параметры

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>* - Major package version. If current version was 2.5.2 would return 2.5.
- *#<_vminor>* - Minor package version. If current version was 2.6.2 it would return 0.2.
- *#<_line>* - Sequence number. If running a G-code file, this returns the current line number.
- *#<_motion_mode>* - Return the interpreter's current motion mode:

| **Motion mode** | **return value** |
|---|---|
| G1 | 10 |
| G2 | 20 |
| G3 | 30 |
| G33 | 330 |
| G38.2 | 382 |
| G38.3 | 383 |
| G38.4 | 384 |
| G38.5 | 385 |
| G5.2 | 52 |
| G73 | 730 |
| G76 | 760 |
| G80 | 800 |
| G81 | 810 |
| G82 | 820 |
| G83 | 830 |
| G84 | 840 |
| G85 | 850 |
| G86 | 860 |
| G87 | 870 |
| G88 | 880 |
| G89 | 890 |

- *#<_plane>* - returns the value designating the current plane:

| Plane | return value |
|---|---|
| G17 | 170 |
| G18 | 180 |
| G19 | 190 |
| G17.1 | 171 |
| G18.1 | 181 |
| G19.1 | 191 |

- *#<_ccomp>* - Status of cutter compensation. Return values:

| Mode | return value |
|---|---|
| G40 | 400 |
| G41 | 410 |
| G41.1 | 411 |
| G41 | 410 |
| G42 | 420 |
| G42.1 | 421 |

- *#<_metric>* - Return 1 if G21 is on, else 0.
- *#<_imperial>* - Return 1 if G20 is on, else 0.
- *#<_absolute>* - Return 1 if G90 is on, else 0.
- *#<_incremental>* - Return 1 if G91 is on, else 0.
- *#<_inverse_time>* - Return 1 if inverse feed mode (G93) is on, else 0.
- *#<_units_per_minute>* - Return 1 if Units/minute feed mode (G94) is on, else 0.
- *#<_units_per_rev>* - Return 1 if Units/revolution mode (G95) is on, else 0.
- *#<_coord_system>* - Return a float of the current coordinate system name (G54..G59.3). For example if your in G55 coordinate system the return value is 550.000000 and if your in G59.1 the return value is 591.000000.

| Mode | return value |
|---|---|
| G54 | 540 |
| G55 | 550 |
| G56 | 560 |
| G57 | 570 |
| G58 | 580 |
| G59 | 590 |
| G59.1 | 591 |
| G59.2 | 592 |
| G59.3 | 593 |

- *#<_tool_offset>* - Return 1 if tool offset (G43) is on, else 0.
- *#<_retract_r_plane>* - Return 1 if G98 is set, else 0.
- *#<_retract_old_z>* - Return 1 if G99 is on, else 0.

#### 11.4.4.5 Параметры системы

* <_spindle_rpm_mode> - Return 1 if spindle rpm mode (G97) is on, else 0. * <_spindle_css_mode> - Return 1 if constant surface speed mode (G96) is on, else 0. * <_ijk_absolute_mode> - Return 1 if Absolute Arc distance mode (G90.1) is on, else 0. * <_lathe_diameter_mode> - Return 1 if this is a lathe configuration and diameter (G7) mode is on, else 0. * <_lathe_radius_mode> - Return 1 if this is a lathe configuration and radius (G8) mode is on, else 0. * <_spindle_on> - Return 1 if spindle currently running (M3 or M4) else 0. * <_spindle_cw> - Return 1 if spindle direction is clockwise (M3) else 0. * <_mist> - Return 1 if mist (M7) is on. * <_flood> - Return 1 if flood (M8) is on. * <_speed_override> - Return 1 if feed override (M48 or M50 P1) is on, else 0. * <_feed_override> - Return 1 if feed override (M48 or M51 P1) is on, else 0. * <_adaptive_feed> - Return 1 if adaptive feed (M52 or M52 P1) is on, else 0. * <_feed_hold> - Return 1 if feed hold switch is enabled (M53 P1), else 0. * <_feed> - Return the current value of F, not the actual feed rate. * <_rpm> - Return the current value of S, not the actual spindle speed. * <_x> - Return current relative X coordinate including all offsets. Same as 5420. In a lathe configuration, it always returns radius. * <_y> - Return current relative Y coordinate including all offsets. Same as 5421. * <_z> - Return current relative Z coordinate including all offsets. Same as 5422. * <_a> - Return current relative A coordinate including all offsets. Same as 5423. * <_b> - Return current relative B coordinate including all offsets. Same as 5424. * <_c> - Return current relative C coordinate including all offsets. Same as 5425. * <_u> - Return current relative U coordinate including all offsets. Same as 5426. * <_v> - Return current relative V coordinate including all offsets. Same as 5427. * <_w> - Return current relative W coordinate including all offsets. Same as 5428. * <_abs_x> - Return current absolute X coordinate (G53) including no offsets. * <_abs_y> - Return current absolute Y coordinate (G53) including no offsets. * <_abs_z> - Return current absolute Z coordinate (G53) including no offsets. * <_abs_a> - Return current absolute A coordinate (G53) including no offsets. * <_abs_b> - Return current absolute B coordinate (G53) including no offsets. * <_abs_c> - Return current absolute C coordinate (G53) including no offsets. * <_current_tool> - Return number of the current tool in spindle. Same as 5400. * <_current_pocket> - Return the tooldata index for the current tool. * <_selected_tool> - Return number of the selected tool post a T code. Default -1. * <_selected_pocket> - Return the tooldata index of the selected pocket post a T code. Default -1 (no pocket selected). * <_value> - Return value from the last O-code *return* or *endsub*. Default value 0 if no expression after *return* or *endsub*. Initialized to 0 on program start. * <_value_returned> - 1.0 if the last O-code *return* or *endsub* returned a value, 0 otherwise. Cleared by the next O-code call. * <_task> - 1.0 if the executing interpreter instance is part of milltask, 0.0 otherwise. Sometimes it is necessary to treat this case specially to retain proper preview, for instance when testing the success of a probe (G38.n) by inspecting #5070, which will always fail in the preview interpreter (e.g. Axis). * <_call_level> - current nesting level of O-code procedures. For debugging. * #<_remap_level> - current level of the remap stack. Each remap in a block adds one to the remap level. For debugging.

### 11.4.5 HAL Контакты и INI значения

If enabled in the INI file G-code has access to the values of INI file entries and HAL pins.

- *#<_ini[section]name>* Returns the value of the corresponding item in the INI file.

For example, if the INI file looks like so:

```
[SETUP]
XPOS = 3.145
YPOS = 2.718
```

you may refer to the named parameters `#<_ini[setup]xpos>` and `#<_ini[setup]ypos>` within G-code.

`EXISTS` can be used to test for presence of a given INI file variable:

```
o100 if [EXISTS[#<_ini[setup]xpos>]]
  (debug, [setup]xpos exists: #<_ini[setup]xpos>)
o100 else
  (debug, [setup]xpos does not exist)
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]>* Allows G-code programs to read the values of HAL pins Variable access is read-only, the only way to *set* HAL pins from G-code remains M62-M65, M67, M68 and custom M100-M199 codes. Note that the value read will not update in real-time, typically the value that was on the pin when the G-code program was started will be returned. It is possible to work round this by forcing a state synch. One way to do this is with a dummy M66 command: M66E0L0

Example:

```
(debug, #<_hal[motion-controller.time]>)
```

Access of HAL items is read-only. Currently, only all-lowercase HAL names can be accessed this way.

`EXISTS` can be used to test for the presence of a given HAL item:

```
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
```

This feature was motivated by the desire for stronger coupling between user interface components like `GladeVCP` and `PyVCP` to act as parameter source for driving NGC file behavior. The alternative - going through the M6x pins and wiring them - has a limited, non-mnemonic namespace and is unnecessarily cumbersome just as a UI/Interpreter communications mechanism.

### 11.4.6 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]]]*.

### 11.4.7 Бинарные операторы

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*).

The binary operations are divided into several groups according to their precedence. If operations in different precedence groups are strung together (for example in the expression *[2.0 / 3 * 1.5 - 5.5*

*/ 11.0]*), operations in a higher group are to be performed before operations in a lower group. If an expression contains more than one operation from the same group (such as the first / and * in the example), the operation on the left is performed first. Thus, the example is equivalent to: *[ [ [2.0 / 3] * 1.5] - [5.5 / 11.0] ]* , which is equivalent to to *[1.0 - 0.5]* , which is *0.5*.

The logical operations and modulus are to be performed on any real numbers, not just on integers. The number zero is equivalent to logical false, and any non-zero number is equivalent to logical true.

Table 11.9: Operators Precedence

| **Operators** | **Precedence** |
|---|---|
| ** | *highest* |
| * / MOD | |
| + - | |
| EQ NE GT GE LT LE | |
| AND OR XOR | *lowest* |

### 11.4.8 Equality and floating-point values

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).

### 11.4.9 Функции

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.

Table 11.10: G-code Functions

| **Function Name** | **Function result** |
|---|---|
| ATAN[arg]/[arg] | Four quadrant inverse tangent |
| ABS[arg] | Absolute value |
| ACOS[arg] | Inverse cosine |
| ASIN[arg] | Inverse sine |
| COS[arg] | Cosine |
| EXP[arg] | e raised to the given power |
| FIX[arg] | Round down to integer |
| FUP[arg] | Round up to integer |
| ROUND[arg] | Round to nearest integer |
| LN[arg] | Base-e logarithm |
| SIN[arg] | Sine |
| SQRT[arg] | Square Root |
| TAN[arg] | Tangent |
| EXISTS[arg] | Check named Parameter |

The *FIX* function rounds towards the left (less positive or more negative) on a number line, so that *FIX[2.8] =2* and *FIX[-2.8] = -3*.

The *FUP* operation rounds towards the right (more positive or less negative) on a number line; *FUP[2.8] = 3* and *FUP[-2.8] = -2*.

The *EXISTS* function checks for the existence of a single named parameter. It takes only one named parameter and returns 1 if it exists and 0 if it does not exist. It is an error if you use a numbered parameter or an expression. Here is an example for the usage of the EXISTS function:

```
o<test> sub
o10 if [EXISTS[#<_global>]]
    (debug, _global exists and has the value #<_global>)
o10 else
    (debug, _global does not exist)
o10 endif
o<test> endsub

o<test> call
#<_global> = 4711
o<test> call
m2
```

### 11.4.10 Repeated Items

A line may have any number of G words, but two G words from the same modal group may not appear on the same line. See the Modal Groups section for more information.

A line may have zero to four M words. Two M words from the same modal group may not appear on the same line.

For all other legal letters, a line may have only one word beginning with that letter.

If a parameter setting of the same parameter is repeated on a line, *#3=15 #3=6*, for example, only the last setting will take effect. It is silly, but not illegal, to set the same parameter twice on the same line.

If more than one comment appears on a line, only the last one will be used; each of the other comments will be read and its format will be checked, but it will be ignored thereafter. It is expected that putting more than one comment on a line will be very rare.

### 11.4.11 Item order

The three types of item whose order may vary on a line (as given at the beginning of this section) are word, parameter setting, and comment. Imagine that these three types of item are divided into three groups by type.

The first group (the words) may be reordered in any way without changing the meaning of the line.

If the second group (the parameter settings) is reordered, there will be no change in the meaning of the line unless the same parameter is set more than once. In this case, only the last setting of the parameter will take effect. For example, after the line *#3=15 #3=6* has been interpreted, the value of parameter 3 will be 6. If the order is reversed to *#3=6 #3=15* and the line is interpreted, the value of parameter 3 will be 15.

If the third group (the comments) contains more than one comment and is reordered, only the last comment will be used.

If each group is kept in order or reordered without changing the meaning of the line, then the three groups may be interleaved in any way without changing the meaning of the line. For example, the line *g40 g1 #3=15 (foo) #4=-7.0* has five items and means exactly the same thing in any of the 120 possible orders (such as *#4=-7.0 g1 #3=15 g40 (foo)*) for the five items.

### 11.4.12 Commands and Machine Modes

Many commands cause the controller to change from one mode to another, and the mode stays active until some other command changes it implicitly or explicitly. Such commands are called *modal*. For example, if coolant is turned on, it stays on until it is explicitly turned off. The G-codes for motion are also modal. If a G1 (straight move) command is given on one line, for example, it will be executed again on the next line if one or more axis words is available on the line, unless an explicit command is given on that next line using the axis words or canceling motion.

*Non-modal* codes have effect only on the lines on which they occur. For example, G4 (dwell) is non-modal.

### 11.4.13 Полярные координаты

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.

In Absolute Mode the distance and angle is from the XY zero position and the angle starts with 0 on the X Positive axis and increases in a CCW direction about the Z axis. The code G1 @1^90 is the same as G1 Y1.

In Relative Mode the distance and angle is also from the XY zero position but it is cumulative. This can be confusing at first how this works in incremental mode.

For example if you have the following program you might expect it to be a square pattern:

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

You can see from the following figure that the output is not what you might expect. Because we added 0.5 to the distance each time the distance from the XY zero position increased with each line.

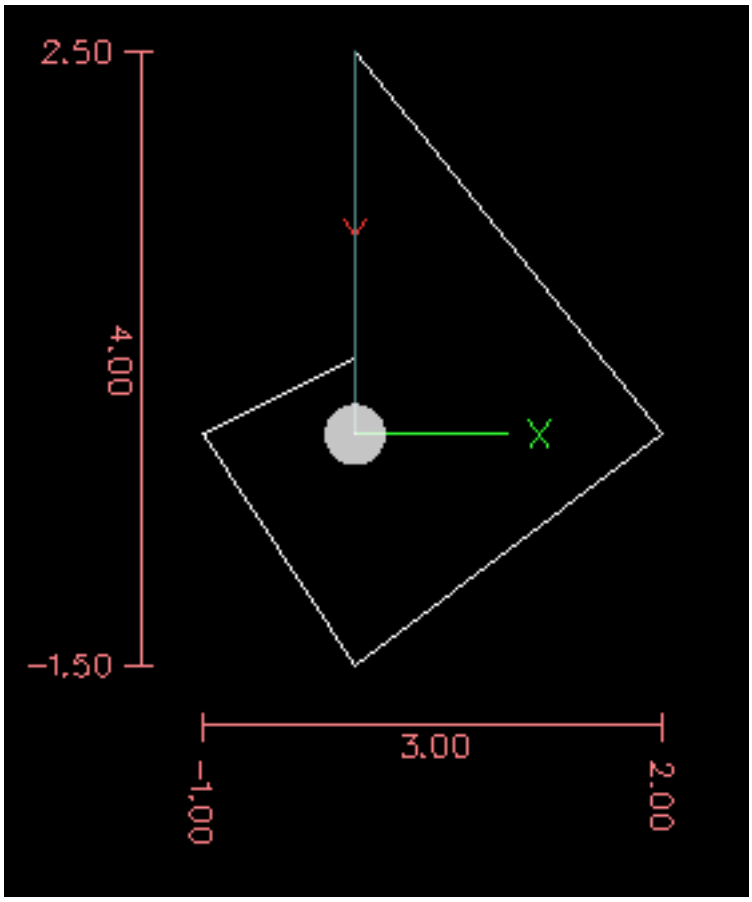



Figure 11.10: Polar Spiral

The following code will produce our square pattern:

```
F100 G1 @.5 ^90
G91 ^90
^90
^90
^90
G90 G0 X0 Y0 M2
```

As you can see by only adding to the angle by 90 degrees each time the end point distance is the same for each line.

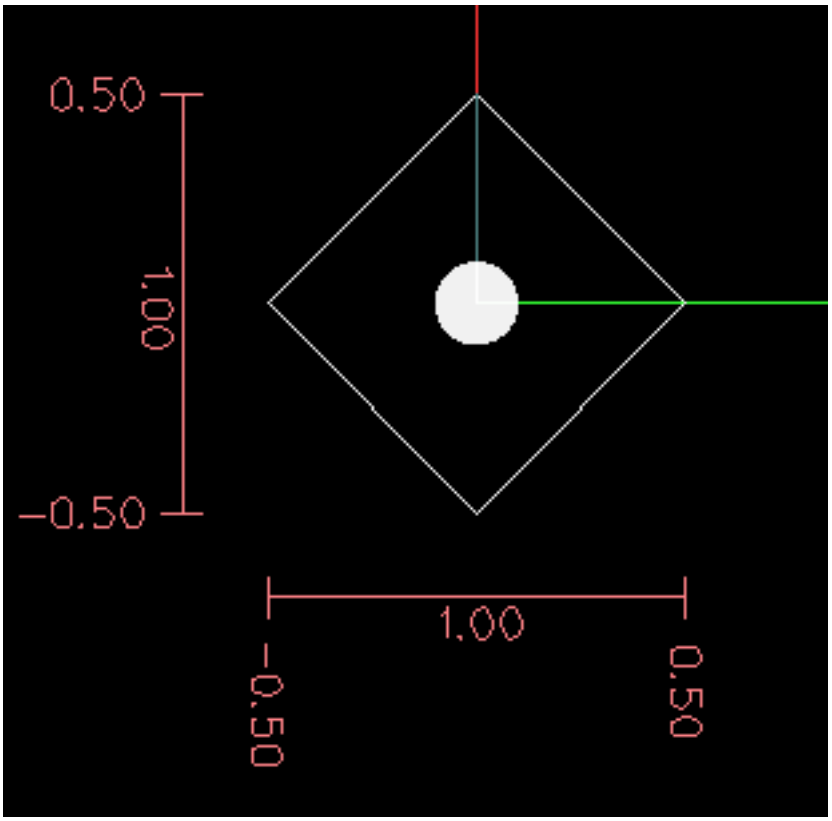



Figure 11.11: Polar Square

It is an error if:

- An incremental move is started at the origin
- A mix of Polar and X or Y words are used

### 11.4.14 Modal Groups

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 versus 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.

Table 11.11: G-code Modal Groups

| **Modal Group Meaning** | **Member Words** |
|---|---|
| Non-modal codes (Group 0) | G4, G10 G28, G30, G52, G53, G92, G92.1, G92.2, G92.3, |
| Motion (Group 1) | G0, G1, G2, G3, G33, G38.n, G73, G76, G80, G81 G82, G83, G84, G85, G86, G87, G88, G89 |
| Plane selection (Group 2) | G17, G18, G19, G17.1, G18.1, G19.1 |
| Distance Mode (Group 3) | G90, G91 |

Table 11.11: (continued)

| **Modal Group Meaning** | **Member Words** |
|---|---|
| Arc IJK Distance Mode (Group 4) | G90.1, G91.1 |
| Feed Rate Mode (Group 5) | G93, G94, G95 |
| Units (Group 6) | G20, G21 |
| Cutter Diameter Compensation (Group 7) | G40, G41, G42, G41.1, G42.1 |
| Tool Length Offset (Group 8) | G43, G43.1, G49 |
| Canned Cycles Return Mode (Group 10) | G98, G99 |
| Coordinate System (Group 12) | G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3 |
| Control Mode (Group 13) | G61, G61.1, G64 |
| Spindle Speed Mode (Group 14) | G96, G97 |
| Lathe Diameter Mode (Group 15) | G7, G8 |

Table 11.12: M-code Modal Groups

| **Modal Group Meaning** | **Member Words** |
|---|---|
| Stopping (Group 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* |
| Spindle (Group 7) | M3, M4, M5 |
| Coolant (Group 8) | (M7 M8 can both be on), M9 |
| Override Switches (Group 9) | M48, M49 |
| User Defined (Group 10) | M100-M199 |

For several modal groups, when a machining center is ready to accept commands, one member of the group must be in effect. There are default settings for these modal groups. When the machining center is turned on or otherwise re-initialized, the default values are automatically in effect.

Group 1, the first group on the table, is a group of G-codes for motion. One of these is always in effect. That one is called the current motion mode.

It is an error to put a G-code from group 1 and a G-code from group 0 on the same line if both of them use axis words. If an axis word-using G-code from group 1 is implicitly in effect on a line (by having been activated on an earlier line), and a group 0 G-code that uses axis words appears on the line, the activity of the group 1 G-code is suspended for that line. The axis word-using G-codes from group 0 are G10, G28, G30, G52 and G92.

It is an error to include any unrelated words on a line with *O*- flow control.

### 11.4.15 Comments

Comments are purely informative and have no influence on machine behaviour.

Comments can be added to lines of G-code to help clear up the intention of the programmer. Comments can be embedded in a line using parentheses () or for the remainder of a line using a semi-colon. The semi-colon is not treated as the start of a comment when enclosed in parentheses.

Comments may appear between words, but not between words and their corresponding parameter. So, *S100(set speed)F200(feed)* is OK while *S(speed)100F(feed)* is not.

Here is an example of a commented program:

```
G0 (Rapid to start) X1 Y1
G0 X1 Y1 (Rapid to start; but don't forget the coolant)
M2 ; End of program.
```

There are several *active* comments which look like comments but cause some action, like *(debug,..)* or *(print,..)*. If there are several comments on a line, only the last comment will be interpreted according to these rules. Hence, a normal comment following an active comment will in effect disable the active comment. For example, *(foo) (debug,#1)* will print the value of parameter *#1*, however *(debug,#1)(foo)* will not.

A comment introduced by a semicolon is by definition the last comment on that line, and will always be interpreted for active comment syntax.

**Note**
Inline comments on O-codes should not be used see the O-code comments section for more information.

### 11.4.16 Messages

* *(MSG,)* - displays message if *MSG* appears after the left parenthesis and before any other printing characters. Variants of *MSG* which include white space and lower case characters are allowed. The rest of the characters before the right parenthesis are considered to be a message. Messages should be displayed on the message display device of the user interface if provided.

**Message Example**

```
(MSG, This is a message)
```

### 11.4.17 Ведение журнала зонда

* *(PROBEOPEN filename.txt)* - will open filename.txt and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it. * *(PROBECLOSE)* - will close the open probelog file.

For more information on probing see the G38 section.

### 11.4.18 Ведение журнала

* *(LOGOPEN,filename.txt)* - opens the named log file. If the file already exists, it is truncated. * *(LOGAPPEND,filename)* - opens the named log file. If the file already exists, the data is appended. * *(LOGCLOSE)* - closes an open log file. * *(LOG,)* - everything past the , is written to the log file if it is open. Supports expansion of parameters as described below.

Examples of logging are in *nc_files/examples/smartprobe.ngc* and in *nc_files/ngcgui_lib/rectange_probe.ngc* sample G-code files.

### 11.4.19 Сообщения об отмене

* *(ABORT,)* - displays a message like *(MSG,)* with the addition of special handling for comment parameters as described below and of aborting the current running process.

### 11.4.20 Debug Messages

* *(DEBUG,)* - displays a message like *(MSG,)* with the addition of special handling for comment parameters as described below.

### 11.4.21 Print Messages

* *(PRINT,)* - messages are output to *stderr* with special handling for comment parameters as described below.

### 11.4.22 Параметры комментариев

In the DEBUG, PRINT and LOG comments, the values of parameters in the message are expanded.

For example: to print a named global variable to stderr (the default console window).

**Parameters Example**

```
(print,endmill dia = #<_endmill_dia>)
(print,value of variable 123 is: #123)
```

Inside the above types of comments, sequences like *#123* are replaced by the value of the parameter 123. Sequences like *#<named parameter>* are replaced by the value of the named parameter. Named parameters will have white space removed from them. So, *#<named parameter>* will be converted to *#<namedparameter>*.

Parameter numbers can be formatted, e.g.:

```
(DEBUG, value = %d#<some_value>)
```

will print the value rounded to an integer.

- %lf is default if there is no formatting string.
- %d = 0 decimals
- %f = four decimals
- %.xf = x (0-9) number of decimals

The formatting will be performed on all parameters in the same line unless changed, i.e., multiple formatting is allowed in one line.

The formatting string does not need to be right beside the parameter.

If the formatting string is created with the wrong pattern it will be printed as characters.

### 11.4.23 Требования к файлам

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.

If a program end code is not used a pair of percent signs % with the first percent sign on the first line of the file followed by one or more lines of G-code and a second percent sign. Any code past the second percent sign is not evaluated.

**Warning**
Using % to wrap a G-code file will not do the same thing as using a program end. The machine will be in what ever state the program left it in using %, the spindle and coolant may still be on and things like G90/91 are left as the last program set them. If you don't use a proper preamble the next program could start in a dangerous condition.

**Note**
The file must be created with a text editor like Gedit and not a word processor like Open Office Word Processor.

### 11.4.24 Размер файла

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.

### 11.4.25 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-code commands (optionally followed by a comment but no other words allowed on the same line)
- Comment (including message)
- Set feed rate mode (G93, G94).
- Set feed rate (F).
- Set spindle speed (S).
- Select tool (T).
- HAL pin I/O (M62-M68).
- Change tool (M6) and Set Tool Number (M61).
- Spindle on or off (M3, M4, M5).
- Save State (M70, M73), Restore State (M72), Invalidate State (M71).
- Coolant on or off (M7, M8, M9).
- Enable or disable overrides (M48, M49,M50,M51,M52,M53).

- User-defined Commands (M100-M199).
- Dwell (G4).
- Set active plane (G17, G18, G19).
- Set length units (G20, G21).
- Cutter radius compensation on or off (G40, G41, G42)
- Cutter length compensation on or off (G43, G49)
- Coordinate system selection (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
- Set path control mode (G61, G61.1, G64)
- Set distance mode (G90, G91).
- Set retract mode (G98, G99).
- Go to reference location (G28, G30) or change coordinate system data (G10) or set axis offsets (G52, G92, G92.1, G92.2, G92.3).
- Perform motion (G0 to G3, G33, G38.n, G73, G76, G80 to G89), as modified (possibly) by G53.
- Stop (M0, M1, M2, M30, M60).

### 11.4.26 G-code Best Practices

**Use an appropriate decimal precision** Use at least 3 digits after the decimal when milling in millimeters, and at least 4 digits after the decimal when milling in inches.

In particular, tolerance checks of arcs are done for .001 and .0001 according to the active units.

**Use consistent white space** G-code is most legible when at least one space appears before words. While it is permitted to insert white space in the middle of numbers, there is no reason to do so.

**Use Center-format arcs** Center-format arcs (which use *I- J- K-* instead of *R-* ) behave more consistently than R-format arcs, particularly for included angles near 180 or 360 degrees.

**Use a Preamble set modal groups** When correct execution of your program depends on modal settings, be sure to set them at the beginning of the part program. Modes can carry over from previous programs and from the MDI commands.

**Example Preamble for a Mill**

```
G17 G20 G40 G49 G54 G80 G90 G94
```

G17 use XY plane, G20 inch mode, G40 cancel diameter compensation, G49 cancel length offset, G54 use coordinate system 1, G80 cancel canned cycles, G90 absolute distance mode, G94 feed/minute mode.

Perhaps the most critical modal setting is the distance units—If you do not include G20 or G21, then different machines will mill the program at different scales. Other settings, such as the return mode in canned cycles may also be important.

**Don't put too many things on one line** Ignore everything in section Order of Execution, and instead write no line of code that is the slightest bit ambiguous.

**Don't set & use a parameter on the same line** Don't use and set a parameter on the same line, even though the semantics are well defined. Updating a variable to a new value, such as *#1=[#1+#2]* is OK.

**Don't use line numbers** Line numbers offer no benefits. When line numbers are reported in error messages, the numbers refer to the line number in the file, not the N-word value.

**When several coordinate systems are moved** Consider using the inverse time speed mode.

Because the meaning of an *F* word in meters per minute varies depending on the type of axis to be moved and because the amount of removed material does not depend only on the feed rate, it can be simpler to use G93, inverse speed of time, to achieve the removal of desired material.

### 11.4.27 Linear and Rotary Axis

Because the meaning of an F-word in feed-per-minute mode varies depending on which axes are commanded to move, and because the amount of material removed does not depend only on the feed rate, it may be easier to use G93 inverse time feed mode to achieve the desired material removal rate.

### 11.4.28 Common Error Messages

- *G-code out of range* - A G-code greater than G99 was used, the scope of G codes in LinuxCNC is 0 - 99. Not every number between 0 and 99 is a valid G-code.
- *Unknown G-code used* - A G-code was used that is not part of the LinuxCNC G-code language.
- *i,j,k word with no Gx to use it* - i, j and k words must be used on the same line as the G-code.
- *Cannot use axis values without a G-code that uses them* - Axis values can not be used on a line without either a modal G-code in effect or a G-code on the same line.
- *File ended with no percent sign or program end* - Every G-code file must end in a M2 or M30 or be wrapped with the percent sign %.

## 11.5 G-Codes

### 11.5.1 Conventions

Conventions used in this section

In the G-code prototypes the hyphen (-) stands for a real value and (<>) denotes an optional item.

If *L-* is written in a prototype the - will often be referred to as the *L number*, and so on for any other letter.

In the G-code prototypes the word *axes* stands for any axis as defined in your configuration.

An optional value will be written like this <*L-* >.

A real value may be:

- An explicit number, *4*
- An expression, *[2+2]*
- A parameter value, *#88*
- A unary function value, *acos[0]*

In most cases, if *axis* words are given (any or all of *X Y Z A B C U V W*), they specify a destination point.

Axis numbers are in the currently active coordinate system, unless explicitly described as being in the absolute coordinate system.

Where axis words are optional, any omitted axes will retain their original value.

Any items in the G-code prototypes not explicitly described as optional are required.

The values following letters are often given as explicit numbers. Unless stated otherwise, the explicit numbers can be real values. For example, *G10 L2* could equally well be written *G[2*5] L[1+1]*. If the value of parameter 100 were 2, *G10 L#100* would also mean the same.

If *L-* is written in a prototype the - will often be referred to as the *L number*, and so on for any other letter.

### 11.5.2 G-Code Quick Reference Table

| Code | Description |
|---|---|
| G0 | Coordinated Motion at Rapid Rate |
| G1 | Coordinated Motion at Feed Rate |
| G2 G3 | Coordinated Helical Motion at Feed Rate |
| G4 | Dwell |
| G5 | Cubic Spline |
| G5.1 | Quadratic B-Spline |
| G5.2 | NURBS, add control point |
| G7 | Diameter Mode (lathe) |
| G8 | Radius Mode (lathe) |
| G10 L0 | Reload Tool Table Data |
| G10 L1 | Set Tool Table Entry |
| G10 L10 | Set Tool Table, Calculated, Workpiece |
| G10 L11 | Set Tool Table, Calculated, Fixture |
| G10 L2 | Coordinate System Origin Setting |
| G10 L20 | Coordinate System Origin Setting Calculated |
| G17 - G19.1 | Plane Select |
| G20 G21 | Set Units of Measure |
| G28 - G28.1 | Go to Predefined Position |
| G30 - G30.1 | Go to Predefined Position |
| G33 | Синхронизированное движение шпинделя |
| G33.1 | Rigid Tapping |
| G38.2 - G38.5 | Probing |
| G40 | Cancel Cutter Compensation |
| G41 G42 | Cutter Compensation |
| G41.1 G42.1 | Dynamic Cutter Compensation |
| G43 | Use Tool Length Offset from Tool Table |
| G43.1 | Dynamic Tool Length Offset |
| G43.2 | Apply additional Tool Length Offset |
| G49 | Cancel Tool Length Offset |
| G52 | Local Coordinate System Offset |
| G53 | Move in Machine Coordinates |
| G54-G59.3 | Select Coordinate System (1 - 9) |
| G61 | Exact Path Mode |
| G61.1 | Exact Stop Mode |
| G64 | Path Control Mode with Optional Tolerance |
| G70 | Lathe finishing cycle |
| G71-G72 | Lathe roughing cycle |
| G73 | Drilling Cycle with Chip Breaking |
| G74 | Left-hand Tapping Cycle with Dwell |
| G76 | Multi-pass Threading Cycle (Lathe) |
| G80 | Cancel Motion Modes |
| G81 | Drilling Cycle |

| Code | Description |
|---|---|
| G82 | Drilling Cycle with Dwell |
| G83 | Drilling Cycle with Peck |
| G84 | Right-hand Tapping Cycle with Dwell |
| G85 | Boring Cycle, No Dwell, Feed Out |
| G86 | Boring Cycle, Stop, Rapid Out |
| G87 | Back-boring Cycle *(not yet implemented)* |
| G88 | Boring Cycle, Stop, Manual Out *(not yet implemented)* |
| G89 | Boring Cycle, Dwell, Feed Out |
| G90 G91 | Distance Mode |
| G90.1 G91.1 | Arc Distance Mode |
| G92 | Coordinate System Offset |
| G92.1 G92.2 | Cancel G92 Offsets |
| G92.3 | Restore G92 Offsets |
| G93 G94 G95 | Feed Modes |
| G96 G97 | Spindle Control Mode, Constant Surface vs Rotation Speed (IPM or m/min vs RPM) |
| G98 G99 | Canned Cycle Z Retract Mode |

## 11.5.3 G0 Rapid Move

```
G0 <axes>
```

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 Rapid Velocity Rate

The MAX_VELOCITY setting in the INI file [TRAJ] section defines the maximumrapid traverse rate. The maximum rapid traverse rate can be higher than the individual axes MAX_VELOCITY setting during a coordinated move. The maximum rapid traverse rate can be slower than the MAX_VELOCITY setting in the [TRAJ] section if an axis MAX_VELOCITY or trajectory constraints limit it.

**G0 Example**

```
G90 (set absolute distance mode)
G0 X1 Y-2.3 (Rapid linear move from current location to X1 Y-2.3)
M2 (end program)
```

- See G90 & M2 sections for more information.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

If *G53* is programmed on the same line, the motion will also differ; see the G53 section for more information.

The path of a G0 rapid motion can be rounded at direction changes and depends on the trajectory control settings and maximum acceleration of the axes.

It is an error if:

- An axis letter is without a real value.
- An axis letter is used that is not configured.

### 11.5.4 G1 Линейное перемещение

```
G1 axes
```

For linear (straight line) motion at programmed feed rate (for cutting or not), program *G1 'axes'*, where all the axis words are optional. The *G1* is optional if the current motion mode is *G1*. This will produce coordinated motion to the destination point at the current feed rate (or slower).

**G1 Example**

```
G90 (set absolute distance mode)
G1 X1.2 Y-3 F10 (linear move at a feed rate of 10 from current position to X1.2 Y-3)
Z-2.3 (linear move at same feed rate from current position to Z-2.3)
Z1 F25 (linear move at a feed rate of 25 from current position to Z1)
M2 (end program)
```

- See G90 & F & M2 sections for more information.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

If *G53* is programmed on the same line, the motion will also differ; see the G53 section for more information.

It is an error if:

- No feed rate has been set.
- An axis letter is without a real value.
- An axis letter is used that is not configured

### 11.5.5 G2, G3 Arc Move

```
G2 or G3 axes offsets (center format)
G2 or G3 axes R- (radius format)
G2 or G3 offsets|R- <P-> (full circles)
```

A circular or helical arc is specified using either *G2* (clockwise arc) or *G3* (counterclockwise arc) at the current feed rate. The direction (CW, CCW) is as viewed from the positive end of the axis about which the circular motion occurs.

The axis of the circle or helix must be parallel to the X, Y, or Z axis of the machine coordinate system. The axis (or, equivalently, the plane perpendicular to the axis) is selected with *G17* (Z-axis, XY-plane), *G18* (Y-axis, XZ-plane), or *G19* (X-axis, YZ-plane). Planes *17.1*, *18.1*, and *19.1* are not currently supported. If the arc is circular, it lies in a plane parallel to the selected plane.

To program a helix, include the axis word perpendicular to the arc plane, for example, if in the *G17* plane, include a *Z* word. This will cause the *Z* axis to move to the programmed value during the circular *XY* motion.

To program an arc that gives more than one full turn, use the *P* word specifying the number of full turns plus the programmed arc. The *P* word must be an integer. If *P* is unspecified, the behavior is as if *P1* was given that is, only one full or partial turn will result. For example, if a 180 degree arc is programmed with a P2, the resulting motion will be 1 1/2 rotations. For each P increment above 1 an extra full circle is added to the programmed arc. Multi turn helical arcs are supported and give motion useful for milling holes or threads.

**Warning**
If the pitch of the helix is very small (less than the naive CAM tolerance) then the helix might be converted into a straight line. Bug #222

If a line of code makes an arc and includes rotary axis motion, the rotary axes turn at a constant rate so that the rotary motion starts and finishes when the XYZ motion starts and finishes. Lines of this sort are hardly ever programmed.

If cutter compensation is active, the motion will differ from the above; see the Cutter Compensation section.

The arc center is absolute or relative as set by G90.1 or G91.1 respectively.

Two formats are allowed for specifying an arc: Center Format and Radius Format.

It is an error if:

- No feed rate has been set.
- The P word is not an integer.

#### 11.5.5.1 Center Format Arcs

Center format arcs are more accurate than radius format arcs and are the preferred format to use.

The end point of the arc along with the offset to the center of the arc from the current location are used to program arcs that are less than a full circle. It is OK if the end point of the arc is the same as the current location.

The offset to the center of the arc from the current location and optionally the number of turns are used to program full circles.

When programming arcs an error due to rounding can result from using a precision of less than 4 decimal places (0.0000) for inch and less than 3 decimal places (0.000) for millimeters.

**Incremental Arc Distance Mode** Arc center offsets are a relative distance from the start location of the arc. Incremental Arc Distance Mode is default.

One or more axis words and one or more offsets must be programmed for an arc that is less than 360 degrees.

No axis words and one or more offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Incremental Arc Distance Mode see the G91.1 section.

**Absolute Arc Distance Mode** Arc center offsets are the absolute distance from the current 0 position of the axis.

One or more axis words and *both* offsets must be programmed for arcs less than 360 degrees.

No axis words and both offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Absolute Arc Distance Mode see the G90.1 section.

**XY-plane (G17)**

```
G2 or G3 <X- Y- Z- I- J- P->
```

- *Z* - helix

- *I* - X offset
- *J* - Y offset
- *P* - number of turns

#### XZ-plane (G18)

```
G2 or G3 <X- Z- Y- I- K- P->
```

- *Y* - helix
- *I* - X offset
- *K* - Z offset
- *P* - number of turns

#### YZ-plane (G19)

```
G2 or G3 <Y- Z- X- J- K- P->
```

- *X* - helix
- *J* - Y offset
- *K* - Z offset
- *P* - number of turns

It is an error if:

- No feed rate is set with the F word.
- No offsets are programmed.
- When the arc is projected on the selected plane, the distance from the current point to the center differs from the distance from the end point to the center by more than (.05 inch/.5 mm) OR ((.0005 inch/.005mm) AND .1% of radius).

Deciphering the Error message *Radius to end of arc differs from radius to start:*

- *start* - the current position
- *center* - the center position as calculated using the i, j, or k words
- *end* - the programmed end point
- *r1* - radius from the start position to the center
- *r2* - radius from the end position to the center

#### 11.5.5.2 Center Format Examples

Calculating arcs by hand can be difficult at times. One option is to draw the arc with a CAD program to get the coordinates and offsets. Keep in mind the tolerance mentioned above, you may have to change the precision of your CAD program to get the desired results. Another option is to calculate the coordinates and offset using formulas. As you can see in the following figures a triangle can be formed from the current position the end position and the arc center.

In the following figure you can see the start position is X0 Y0, the end position is X1 Y1. The arc center position is at X1 Y0. This gives us an offset from the start position of 1 in the X axis and 0 in the Y axis. In this case only an I offset is needed.

**G2 Example Line**

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (clockwise arc in the XY plane)
```

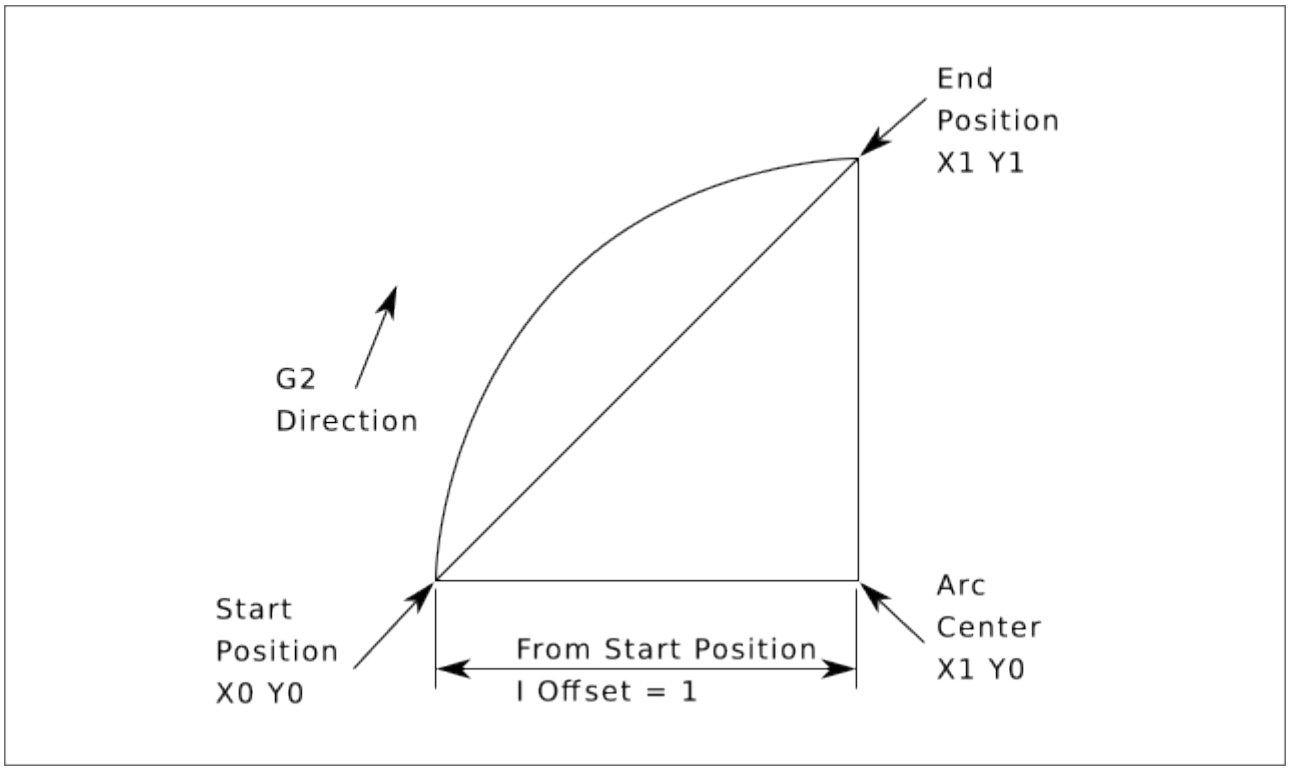



Figure 11.12: G2 Example

In the next example we see the difference between the offsets for Y if we are doing a G2 or a G3 move. For the G2 move the start position is X0 Y0, for the G3 move it is X0 Y1. The arc center is at X1 Y0.5 for both moves. The G2 move the J offset is 0.5 and the G3 move the J offset is -0.5.

**G2-G3 Example Line**

```
G0 X0 Y0
G2 X0 Y1 I1 J0.5 F25 (clockwise arc in the XY plane)
G3 X0 Y0 I1 J-0.5 F25 (counterclockwise arc in the XY plane)
```

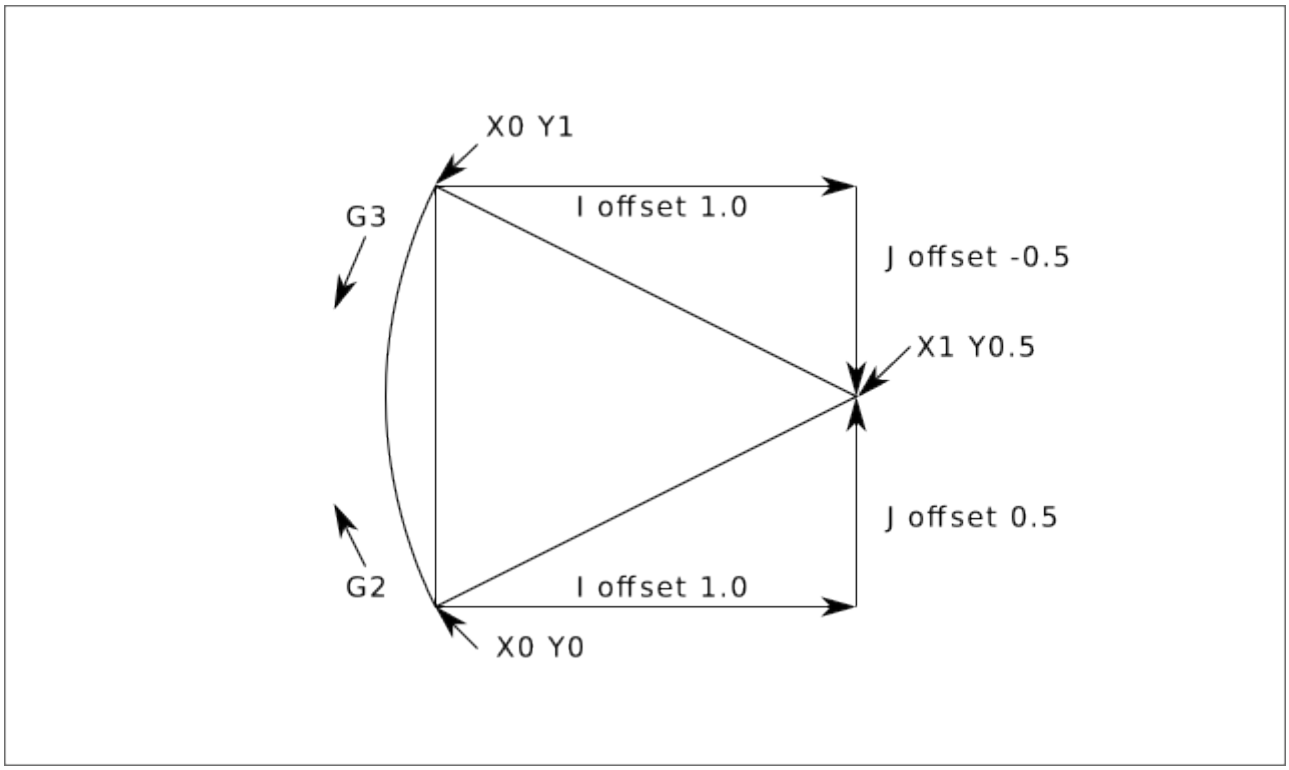



Figure 11.13: G2-G3 Example

In the next example we show how the arc can make a helix in the Z axis by adding the Z word.

**G2 Example Helix**

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (helix arc with Z added)
```

In the next example we show how to make more than one turn using the P word.

**P word Example**

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

In the center format, the radius of the arc is not specified, but it may be found easily as the distance from the center of the circle to either the current point or the end point of the arc.

#### 11.5.5.3 Radius Format Arcs

```
G2 or G3 axes R- <P->
```

- *R* - radius from current position

It is not good practice to program radius format arcs that are nearly full circles or nearly semicircles because a small change in the location of the end point will produce a much larger change in the location of the center of the circle (and, hence, the middle of the arc). The magnification effect is large enough that rounding error in a number can produce out-of-tolerance cuts. For instance, a 1% displacement of the endpoint of a 180 degree arc produced a 7% displacement of the point 90 degrees

along the arc. Nearly full circles are even worse. Other size arcs (in the range tiny to 165 degrees or 195 to 345 degrees) are OK.

In the radius format, the coordinates of the end point of the arc in the selected plane are specified along with the radius of the arc. Program *G2 axes R-* (or use *G3* instead of *G2* ). R is the radius. The axis words are all optional except that at least one of the two words for the axes in the selected plane must be used. The R number is the radius. A positive radius indicates that the arc turns through less than 180 degrees, while a negative radius indicates a turn of more than 180 degrees. If the arc is helical, the value of the end point of the arc on the coordinate axis parallel to the axis of the helix is also specified.

It is an error if:

- both of the axis words for the axes of the selected plane are omitted
- the end point of the arc is the same as the current point.

**G2 Example Line**

```
G17 G2 X10 Y15 R20 Z5 (radius format with arc)
```

The above example makes a clockwise (as viewed from the positive Z-axis) circular or helical arc whose axis is parallel to the Z-axis, ending where X=10, Y=15, and Z=5, with a radius of 20. If the starting value of Z is 5, this is an arc of a circle parallel to the XY-plane; otherwise it is a helical arc.

### 11.5.6 G4 Dwell

```
G4 P-
```

- *P* - seconds to dwell (floating point)

The P number is the time in seconds that all axes will remain unmoving. The P number is a floating point number so fractions of a second may be used. G4 does not affect spindle, coolant and any I/O.

**G4 Example Line**

```
G4 P0.5 (wait for 0.5 seconds before proceeding)
```

It is an error if:

- the P number is negative or not specified.

### 11.5.7 G5 Cubic Spline

```
G5 X- Y- <I- J-> P- Q-
```

- *I* - X incremental offset from start point to first control point
- *J* - Y incremental offset from start point to first control point
- *P* - X incremental offset from end point to second control point
- *Q* - Y incremental offset from end point to second control point

G5 creates a cubic B-spline in the XY plane with the X and Y axes only. P and Q must both be specified for every G5 command.

For the first G5 command in a series of G5 commands, I and J must both be specified. For subsequent G5 commands, either both I and J must be specified, or neither. If I and J are unspecified, the starting direction of this cubic will automatically match the ending direction of the previous cubic (as if I and J are the negation of the previous P and Q).

For example, to program a curvy N shape:

**G5 Sample initial cubic spline**

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

A second curvy N that attaches smoothly to this one can now be made without specifying I and J:

**G5 Sample subsequent cubic spline**

```
G5 P0 Q-3 X2 Y2
```

It is an error if:

- P and Q are not both specified.
- Just one of I or J are specified.
- I or J are unspecified in the first of a series of G5 commands.
- An axis other than X or Y is specified.
- The active plane is not G17.

### 11.5.8 G5.1 Quadratic Spline

```
G5.1 X- Y- I- J-
```

- *I* - X incremental offset from start point to control point
- *J* - Y incremental offset from start point to control point

G5.1 creates a quadratic B-spline in the XY plane with the X and Y axis only. Not specifying I or J gives zero offset for the unspecified axis, so one or both must be given.

For example, to program a parabola, through the origin, from X-2 Y4 to X2 Y4:

**G5.1 Sample quadratic spline**

```
G90 G17
G0 X-2 Y4
G5.1 X2 I2 J-8
```

It is an error if:

- both I and J offset are unspecified or zero
- An axis other than X or Y is specified
- The active plane is not G17

### 11.5.9 G5.2 G5.3 NURBS Block

```
G5.2 <P-> <X- Y-> <L->
X- Y- <P->
...
G5.3
```

**Warning**
G5.2, G5.3 is experimental and not fully tested.

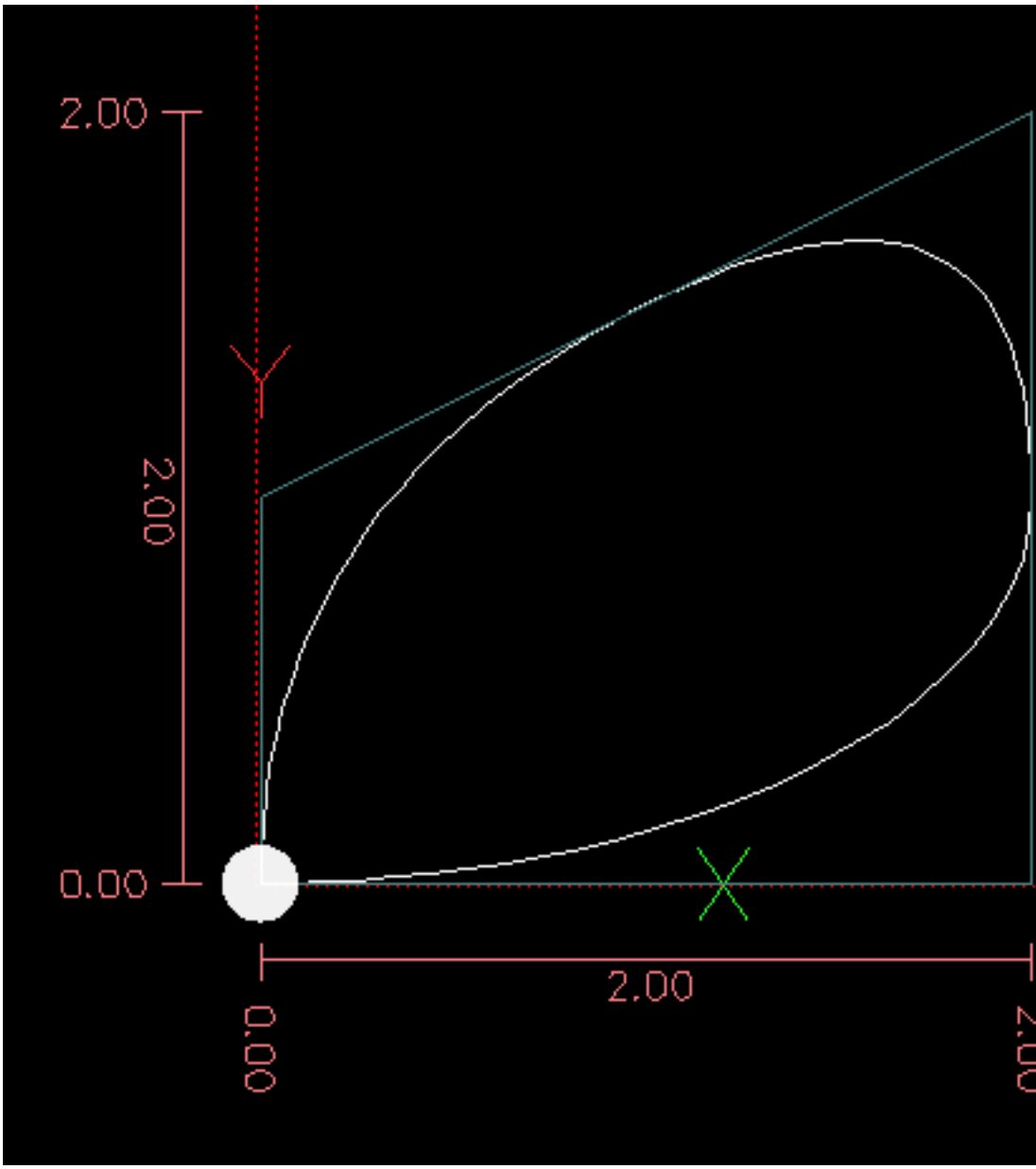
G5.2 is for opening the data block defining a NURBS and G5.3 for closing the data block. In the lines between these two codes the curve control points are defined with both their related *weights* (P) and the parameter (L) which determines the order of the curve.

The current coordinate, before the first G5.2 command, is always taken as the first NURBS control point. To set the weight for this first control point, first program G5.2 P- without giving any X Y.

The default weight if P is unspecified is 1. The default order if L is unspecified is 3.

**G5.2 Example**

```
G0 X0 Y0 (rapid move)
F10 (set feed rate)
G5.2 P1 L3
     X0 Y1 P1
     X2 Y2 P1
     X2 Y0 P1
     X0 Y0 P2
G5.3
; The rapid moves show the same path without the NURBS Block
G0 X0 Y1
   X2 Y2
   X2 Y0
   X0 Y0
M2
```




**Sample NURBS Output**

More information on NURBS can be found here:

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

### 11.5.10 Режим диаметра токарного станка G7

```
G7
```

Program G7 to enter the diameter mode for axis X on a lathe. When in the diameter mode the X axis moves on a lathe will be 1/2 the distance to the center of the lathe. For example X1 would move the cutter to 0.500” from the center of the lathe thus giving a 1” diameter part.

### 11.5.11 Радиусный режим токарного станка G8

```
G8
```

Program G8 to enter the radius mode for axis X on a lathe. When in Radius mode the X axis moves on a lathe will be the distance from the center. Thus a cut at X1 would result in a part that is 2” in diameter. G8 is default at power up.

### 11.5.12 G10 L0 Перезагрузить данные таблицы инструментов

```
G10 L0
```

G10 L0 reload all tool table data. Requires that there is no current tool loaded in spindle.

**Note**
When using G10 L0, tool parameters (#5401-#5413) will be updated immediately and any altered tool diameters will be used for subsequent G41,42 cutter radius compensation commands. Existing G43 tool length compensation values will remain in effect until updated by new G43 commands.

### 11.5.13 G10 L1 Таблица набора инструментов

```
G10 L1 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L1 sets the tool table for the *P* tool number to the values of the words.

A valid G10 L1 rewrites and reloads the tool table for the specified tool.

**G10 L1 Example Line**

```
G10 L1 P1 Z1.5 (set tool 1 Z offset from the machine origin to 1.5)
G10 L1 P2 R0.015 Q3 (lathe example setting tool 2 radius to 0.015 and orientation to 3)
```

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

For more information on cutter orientation used by the *Q* word, see the Lathe Tool Orientation diagram.

### 11.5.14 G10 L2 Set Coordinate System

```
G10 L2 P- <axes R->
```

- *P* - coordinate system (0-9)
- *R* - rotation about the Z axis

G10 L2 offsets the origin of the axes in the coordinate system specified to the value of the axis word. The offset is from the machine origin established during homing. The offset value will replace any current offsets in effect for the coordinate system specified. Axis words not used will not be changed.

Program P0 to P9 to specify which coordinate system to change.

Table 11.14: Coordinate System

| P Value | Coordinate System | G-code |
|---|---|---|
| 0 | Active | n/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 |

Optionally program R to indicate the rotation of the XY axis around the Z axis. The direction of rotation is CCW as viewed from the positive end of the Z axis.

All axis words are optional.

Being in incremental distance mode (G91) has no effect on *G10 L2*.

Important Concepts:

- `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.
- When a rotation is in effect jogging an axis will only move that axis in a positive or negative direction and not along the rotated axis.
- If a *G52* local offset or *G92* origin offset was in effect before *G10 L2*, it will continue to be in effect afterwards.
- When programming a coordinate system with R, any *G52* or *G92* will be applied **after** the rotation.
- The coordinate system whose origin is set by a *G10* command may be active or inactive at the time the *G10* is executed. If it is currently active, the new coordinates take effect immediately.

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.

- An axis is programmed that is not defined in the configuration.

**G10 L2 Example Line**

```
G10 L2 P1 X3.5 Y17.2
```

In the above example the origin of the first coordinate system (the one selected by *G54*) is set to be X=3.5 and Y=17.2. Because only X and Y are specified, the origin point is only moved in X and Y; the other coordinates are not changed.

**G10 L2 Example Line**

```
G10 L2 P1 X0 Y0 Z0 (clear offsets for X,Y & Z axes in coordinate system 1)
```

The above example sets the XYZ coordinates of the coordinate system 1 to the machine origin.

The coordinate system is described in the Coordinate System section.

### 11.5.15 G10 L10 Таблица набора инструментов

```
G10 L10 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L10 changes the tool table entry for tool P so that if the tool offset is reloaded, with the machine in its current position and with the current G5x and G52/G92 offsets active, the current coordinates for the given axes will become the given values. The axes that are not specified in the G10 L10 command will not be changed. This could be useful with a probe move as described in the G38 section.

**G10 L10 Example**

```
T1 M6 G43 (load tool 1 and tool length offsets)
G10 L10 P1 Z1.5 (set the current position for Z to be 1.5)
G43 (reload the tool length offsets from the changed tool table)
M2 (end program)
```

- See T & M6, and G43/G43.1 sections for more information.

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

### 11.5.16 G10 L11 Таблица набора инструментов

```
G10 L11 P- axes <R- I- J- Q->
```

- *P* - tool number
- *R* - radius of tool
- *I* - front angle (lathe)
- *J* - back angle (lathe)
- *Q* - orientation (lathe)

G10 L11 is just like G10 L10 except that instead of setting the entry according to the current offsets, it is set so that the current coordinates would become the given value if the new tool offset is reloaded and the machine is placed in the G59.3 coordinate system without any G52/G92 offset active.

This allows the user to set the G59.3 coordinate system according to a fixed point on the machine, and then use that fixture to measure tools without regard to other currently-active offsets. It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

### 11.5.17 G10 L20 Set Coordinate System

```
G10 L20 P- axes
```

- *P* - coordinate system (0-9)

G10 L20 is similar to G10 L2 except that instead of setting the offset/entry to the given value, it is set to a calculated value that makes the current coordinates become the given value.

**G10 L20 Example Line**

```
G10 L20 P1 X1.5 (set the X axis current location in coordinate system 1 to 1.5)
```

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.
- An axis is programmed that is not defined in the configuration.

### 11.5.18 G17 - G19.1 Plane Select

These codes set the current plane as follows:

- *G17* - XY (default)
- *G18* - ZX
- *G19* - YZ
- *G17.1* - UV
- *G18.1* - WU
- *G19.1* - VW

The UV, WU and VW planes do not support arcs.

It is a good idea to include a plane selection in the preamble of each G-code file.

The effects of having a plane selected are discussed in section G2 G3 Arcs and section G81 G89.

### 11.5.19 G20, G21 Units

- *G20* - to use inches for length units.
- *G21* - to use millimeters for length units.

It is a good idea to include units in the preamble of each G-code file.

### 11.5.20 G28, G28.1 Go/Set Predefined Position

**Warning**
Only use G28 when your machine is homed to a repeatable position and the desired G28 position has been stored with G28.1.

G28 uses the values stored in parameters 5161-5169 as the X Y Z A B C U V W final point to move to. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the INI file. All axes defined in the INI file will be moved when a G28 is issued. If no positions are stored with G28.1 then all axes will go to the machine origin.

- *G28* - makes a rapid move from the current position to the *absolute* position of the values in parameters 5161-5166.
- *G28 axes* - makes a rapid move to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5161-5166 for all *axes* specified. Any *axis* not specified will not move.
- *G28.1* - stores the current *absolute* position into parameters 5161-5166.

**G28 Example Line**

```
G28 Z2.5 (rapid to Z2.5 then to Z location specified in #5163)
```

It is an error if :

- Cutter Compensation is turned on

### 11.5.21 G30, G30.1 Go/Set Predefined Position

**Warning**
Only use G30 when your machine is homed to a repeatable position and the desired G30 position has been stored with G30.1.

G30 functions the same as G28 but uses the values stored in parameters 5181-5189 as the X Y Z A B C U V W final point to move to. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the INI file. All axes defined in the INI file will be moved when a G30 is issued. If no positions are stored with G30.1 then all axes will go to the machine origin.

**Note**
G30 parameters will be used to move the tool when a M6 is programmed if TOOL_CHANGE_AT_G30=1 is in the [EMCIO] section of the INI file.

- *G30* - makes a rapid move from the current position to the *absolute* position of the values in parameters 5181-5189.
- *G30 axes* - makes a rapid move to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5181-5189 for all *axes* specified. Any *axis* not specified will not move.
- *G30.1* - stores the current absolute position into parameters 5181-5186.

#### G30 Example Line

```
G30 Z2.5 (rapid to Z2.5 then to the Z location specified in #5183)
```

It is an error if :

- Cutter Compensation is turned on

### 11.5.22 G33 Spindle Synchronized Motion

```
G33 X- Y- Z- K- $-
```

- *K* - distance per revolution

For spindle-synchronized motion in one direction, code *G33 X- Y- Z- K-* where K gives the distance moved in XYZ for each revolution of the spindle. For instance, if starting at *Z=0*, *G33 Z-1 K.0625* produces a 1 inch motion in Z over 16 revolutions of the spindle. This command might be part of a program to produce a 16TPI thread. Another example in metric, *G33 Z-15 K1.5* produces a movement of 15mm while the spindle rotates 10 times for a thread of 1.5mm.

The (optional) $ argument sets which spindle the motion is synchronised to (default is zero). For example G33 Z10 K1 $1 will move the spindle in synchrony with the spindle.N.revs HAL pin value.

Spindle-synchronized motion waits for the spindle index and spindle at speed pins, so multiple passes line up. *G33* moves end at the programmed endpoint. G33 could be used to cut tapered threads or a fusee.

All the axis words are optional, except that at least one must be used.

**Note**
K follows the drive line described by *X- Y- Z-*. K is not parallel to the Z axis if X or Y endpoints are used for example when cutting tapered threads.

**Technical Info**

At the beginning of each G33 pass, LinuxCNC uses the spindle speed and the machine acceleration limits to calculate how long it will take Z to accelerate after the index pulse, and determines how many degrees the spindle will rotate during that time. It then adds that angle to the index position and computes the Z position using the corrected spindle angle. That means that Z will reach the correct position just as it finishes accelerating to the proper speed, and can immediately begin cutting a good thread.

**HAL Connections** The pin *spindle.N.at-speed* must be set or driven true for the motion to start. Additionally spindle.N.revs must increase by 1 for each revolution of the spindle and the spindle.N.index-enable pin must be connected to an encoder (or resolver) counter which resets index-enable once per rev.

See the Integrators Manual for more information on spindle synchronized motion.

**G33 Example**

```
G90 (absolute distance mode)
G0 X1 Z0.1 (rapid to position)
S100 M3 (start spindle turning)
G33 Z-2 K0.125 (move Z axis to -2 at a rate to equal 0.125 per revolution)
G0 X1.25 (rapid move tool away from work)
Z0.1 (rapid move to starting Z position)
M2 (end program)
```

- See G90 & G0 & M2 sections for more information.

It is an error if:

- All axis words are omitted.
- The spindle is not turning when this command is executed.
- The requested linear motion exceeds machine velocity limits due to the spindle speed.

## 11.5.23 G33.1 Rigid Tapping

```
G33.1 X- Y- Z- K- I- $-
```

- *K* - distance per revolution
- *I* - optional spindle speed multiplier for faster return move
- *$* - optional spindle selector

**Warning**
For Z only tapping preposition the XY location prior to calling G33.1 and only use a Z word in the G33.1. If the coordinates specified are not the current coordinates when calling G33.1 for tapping the move will not be along the Z axis but will be a coordinated, spindle-synchronized move from the current location to the location specified and back.

For rigid tapping (spindle synchronized motion with return), code *G33.1 X- Y- Z- K-* where *K-* gives the distance moved for each revolution of the spindle.

A rigid tapping move consists of the following sequence:

- A move from the current coordinate to the specified coordinate, synchronized with the selected spindle at the given ratio and starting from the current coordinate with a spindle index pulse.
- When reaching the endpoint, a command to reverse the spindle, and speed up by a factor set by the multiplier (e.g., from clockwise to counterclockwise).
- Continued synchronized motion beyond the specified end coordinate until the spindle actually stops and reverses.
- Continued synchronized motion back to the original coordinate.
- When reaching the original coordinate, a command to reverse the spindle a second time (e.g., from counterclockwise to clockwise).
- Continued synchronized motion beyond the original coordinate until the spindle actually stops and reverses.
- An **unsynchronized** move back to the original coordinate.

Spindle-synchronized motions wait for spindle index, so multiple passes line up.*G33.1* moves end at the original coordinate.

All the axis words are optional, except that at least one must be used.

**G33.1 Example**

```
G90 (set absolute mode)
G0 X1.000 Y1.000 Z0.100 (rapid move to starting position)
S100 M3 (turn on the spindle, 100 RPM)
G33.1 Z-0.750 K0.05 (rigid tap a 20 TPI thread 0.750 deep)
M2 (end program)
```

- See G90 & G0 & M2 sections for more information.

It is an error if:

- All axis words are omitted.
- The spindle is not turning when this command is executed
- The requested linear motion exceeds machine velocity limits due to the spindle speed

### 11.5.24 G38.*n* Straight Probe

```
G38.n axes
```

- *G38.2* - probe toward workpiece, stop on contact, signal error if failure
- *G38.3* - probe toward workpiece, stop on contact
- *G38.4* - probe away from workpiece, stop on loss of contact, signal error if failure
- *G38.5* - probe away from workpiece, stop on loss of contact

**Important**
Вы не сможете использовать перемещение датчика, пока ваш станок не будет настроен на подачу входного сигнала датчика. Входной сигнал датчика должен быть подключен к параметру *motion.probe-input* в файле .hal. G38.*n* использует motion.probe-input для определения момента установления (или потери) контакта датчика. TRUE — контакт датчика замкнут (касание), FALSE — контакт датчика разомкнут.

Program *G38*.n *axes* to perform a straight probe operation. The axis words are optional, except that at least one of them must be used. The axis words together define the destination point that the probe will move towards, starting from the current location. If the probe is not tripped before the destination is reached G38.2 and G38.4 will signal an error.

The tool in the spindle must be a probe or contact a probe switch.

In response to this command, the machine moves the controlled point (which should be at the center of the probe ball) in a straight line at the current feed rate toward the programmed point. In inverse time feed mode, the feed rate is such that the whole motion from the current point to the programmed point would take the specified time. The move stops (within machine acceleration limits) when the programmed point is reached, or when the requested change in the probe input takes place, whichever occurs first.

Table 11.15: Probing G-Codes

| Code | Target State | Move orientation | Error Signal |
|---|---|---|---|
| G38.2 | Touched | Toward piece | Yes |
| G38.3 | Touched | Toward piece | No |
| G38.4 | Untouched | From piece | Yes |
| G38.5 | Untouched | From piece | No |

After successful probing, parameters #5061 to #5069 will be set to the X, Y, Z, A, B, C, U, V, W coordinates of the location of the controlled point at the time the probe changed state (in the current work coordinate system). After unsuccessful probing, they are set to the coordinates of the programmed point. Parameter 5070 is set to 1 if the probe succeeded and 0 if the probe failed. If the probing operation failed, G38.2 and G38.4 will signal an error by posting an message on screen if the selected GUI supports that. And by halting program execution.

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.

**Пример G38.2**

```
G49
G38.2 Z-100 F100
#<zworkoffset> = [#[5203 + #5220 * 20] + #5213 * #5210]
G10 L1 P#5400 Z#<zworkoffset> (set new tool offset)
G43
```

A comment of the form *(PROBEOPEN filename.txt)* will open *filename.txt* and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it. The file must be closed with *(PROBECLOSE)*. For more information see the Comments section.

An example file *smartprobe.ngc* is included (in the examples directory) to demonstrate using probe moves to log to a file the coordinates of a part. The program *smartprobe.ngc* could be used with *ngcgui* with minimal changes.

It is an error if:

- the current point is the same as the programmed point.
- no axis word is used
- cutter compensation is enabled
- the feed rate is zero
- the probe is already in the target state

### 11.5.25 G40 Compensation Off

- *G40* - turn cutter compensation off. If tool compensation was on the next move must be a linear move and longer than the tool diameter. It is OK to turn compensation off when it is already off.

#### G40 Example

```
; current location is X1 after finishing cutter compensated move
G40 (turn compensation off)
G0 X1.6 (linear move longer than current cutter diameter)
M2 (end program)
```

See G0 & M2 sections for more information.

It is an error if:

- A G2/G3 arc move is programmed next after a G40.
- The linear move after turning compensation off is less than the tool diameter.

### 11.5.26 G41, G42 компенсация режущего инструмента

```
G41 <D-> (left of programmed path)
G42 <D-> (right of programmed path)
```

- *D* - tool number

The D word is optional; if there is no D word the radius of the currently loaded tool will be used (if no tool is loaded and no D word is given, a radius of 0 will be used).

If supplied, the D word is the tool number to use. This would normally be the number of the tool in the spindle (in which case the D word is redundant and need not be supplied), but it may be any valid tool number.

---

**Note**
*G41/G42 D0* is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the Tool Change section). On nonrandom tool changer machines, *G41/G42 D0* applies the Tool Length Offset of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G41/G42 D0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

---

To start cutter compensation to the left of the part profile, use G41. G41 starts cutter compensation to the left of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

To start cutter compensation to the right of the part profile, use G42. G42 starts cutter compensation to the right of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

The lead in move must be at least as long as the tool radius. The lead in move can be a rapid move.

Cutter compensation may be performed if the XY-plane or XZ-plane is active.

User M100-M199 commands are allowed when Cutter Compensation is on.

The behavior of the machining center when cutter compensation is on is described in the Cutter Compensation section along with code examples.

It is an error if:

- The D number is not a valid tool number or 0.
- The YZ plane is active.
- Cutter compensation is commanded to turn on when it is already on.

### 11.5.27 G41.1, G42.1 Динамическая компенсация режущего инструмента

```
G41.1 D- <L-> (left of programmed path)
G42.1 D- <L-> (right of programmed path)
```

- *D* - cutter diameter
- *L* - tool orientation (see lathe tool orientation)

G41.1 & G42.1 function the same as G41 & G42 with the added scope of being able to program the tool diameter. The L word defaults to 0 if unspecified.

It is an error if:

- The YZ plane is active.
- The L number is not in the range from 0 to 9 inclusive.
- The L number is used when the XZ plane is not active.
- Cutter compensation is commanded to turn on when it is already on.

### 11.5.28 G43 смещение длины инструмента

```
G43 <H->
```

- *H* - tool number (optional)
- *G43* - enables tool length compensation. G43 changes subsequent motions by offsetting the axis coordinates by the length of the offset. G43 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

*G43* without an H word uses the currently loaded tool from the last *T*n *M6*.

*G43 H*n uses the offset for tool *n*.

The active tool length compensation values are stored in the numbered parameters *5401-5409*.

**Note**
*G43 H0* is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the Tool Changers section). On nonrandom tool changer machines, *G43 H0* applies the Tool Length Offset of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G43 H0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

#### G43 H- Example Line

```
G43 H1 (set tool offsets using the values from tool 1 in the tool table)
```

It is an error if:

- the H number is not an integer, or
- the H number is negative, or
- the H number is not a valid tool number (though note that 0 is a valid tool number on nonrandom tool changer machines, it means ”the tool currently in the spindle”)

### 11.5.29 G43.1 Динамическое смещение длины инструмента

```
G43.1 axes
```

- *G43.1 axes* - change subsequent motions by replacing the current offset(s) of axes. G43.1 does not cause any motion. The next time a compensated axis is moved, that axis’s endpoint is the compensated location.

#### G43.1 Example

```
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)
```

- See G90 & T & M6 sections for more information.

It is an error if:

- motion is commanded on the same line as *G43.1*

**Note**
G43.1 does not write to the tool table.

## 11.5.30 G43.2 Apply additional Tool Length Offset

```
G43.2 H- or axes-
```

- *H* - tool number
- *G43.2 H*n - 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 Пример

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1's offset)
G43.2 H10 (also add in T10's tool offset)
M2 (end program)
```

#### G43.2 Пример осей

```
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)
```

You can sum together an arbitrary number of offsets by calling G43.2 more times. There are no built-in assumptions about which numbers are geometry offsets and which are wear offsets, or that you should have only one of each.

Like the other G43 commands, G43.2 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

It is an error if:

- *H* is unspecified and no axis offsets are specified.
- *H* is specified and the given tool number does not exist in the tool table.
- *H* is specified and axes are also specified.

**Note**
G43.2 does not write to the tool table.

## 11.5.31 G49 отмена смещения длины инструмента

- *G49* - cancels tool length compensation

It is OK to program using the same offset already in use. It is also OK to program using no tool length offset if none is currently being used.

### 11.5.32 G52 Local Coordinate System Offset

```
G52 axes
```

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 axes
```

To move in the machine coordinate system, program *G53* on the same line as a linear move. *G53* is not modal and must be programmed on each line. *G0* or *G1* does not have to be programmed on the same line if one is currently active.

For example *G53 G0 X0 Y0 Z0* will move the axes to the home position even if the currently selected coordinate system has offsets in effect.

**G53 Example**

```
G53 G0 X0 Y0 Z0 (rapid linear move to the machine origin)
G53 X2 (rapid linear move to absolute coordinate X2)
```

See G0 section for more information.

It is an error if:

- G53 is used without G0 or G1 being active,
- or G53 is used while cutter compensation is on.

### 11.5.34 G54-G59.3 Select Coordinate System

- *G54* - select coordinate system 1
- *G55* - select coordinate system 2
- *G56* - select coordinate system 3
- *G57* - select coordinate system 4
- *G58* - select coordinate system 5
- *G59* - select coordinate system 6
- *G59.1* - select coordinate system 7
- *G59.2* - select coordinate system 8
- *G59.3* - select coordinate system 9

The coordinate systems store the axis values and the XY rotation angle around the Z axis in the parameters shown in the following table.

Table 11.16: Coordinate System Parameters

| Select | 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 |

It is an error if:

- selecting a coordinate system is used while cutter compensation is on.

See the Coordinate System section for an overview of coordinate systems.

### 11.5.35 G61 Exact Path Mode

- *G61* - Exact path mode, movement exactly as programmed. Moves will slow or stop as needed to reach every programmed point. If two sequential moves are exactly co-linear movement will not stop.

### 11.5.36 G61.1 Exact Stop Mode

- *G61.1* - Exact stop mode, movement will stop at the end of each programmed segment.

### 11.5.37 G64 Path Blending

```
G64 <P- <Q->>
```

- *P* - motion blending tolerance
- *Q* - naive cam tolerance
- *G64* - best possible speed. Without P (Or a default value in RS274NGC) means to keep the best speed possible, no matter how far away from the programmed point you end up.
- *G64 P-* - Blend between best speed and deviation tolerance
- *G64 P- <Q- >* blending with tolerance. It is a way to fine tune your system for best compromise between speed and accuracy. The P- tolerance means that the actual path will be no more than P- away from the programmed endpoint. The velocity will be reduced if needed to maintain the path. If you set Q to a non-zero value it turns on the *Naive CAM Detector*: when there are a series of linear XYZ feed moves at the same feed rate that are less than Q- away from being collinear, they are collapsed into a single linear move. On G2/G3 moves in the G17 (XY) plane when the maximum deviation of an arc from a straight line is less than the G64 P- 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. It is OK to program for the mode that is already active. See also the Trajectory Control section for more information on these modes. If Q is not specified then it will have the same behavior as before and use the value of P-. Set Q to zero to disable the *Naive CAM Detector*.

It is a good idea to include a path control specification in the preamble of each G-code file.

**G64 P- Q- Example Line**

```
G64 P0.015 Q0.015 (set path following to be within 0.015 of the actual path)
```

The P- and Q- values are chosen small. Usually smaller than the machine accuracy or the accuracy of commonly manufactured parts. Below are examples with extreme P- and Q- values to understand the G64 function.

**G64 Without values**

```
G64 (set without P- and Q- values)
```

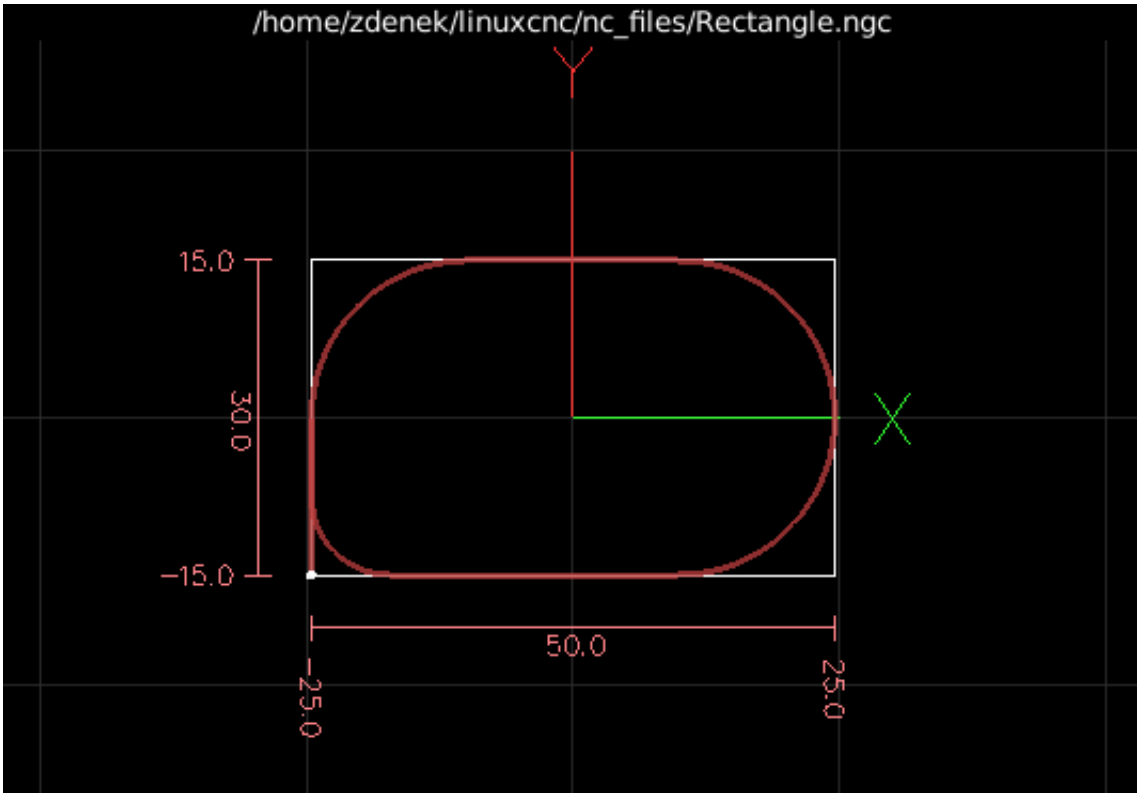



Figure 11.14: G64 Прямоугольник

**G64 With big Q- value**

```
G64 P0.015 Q6
```

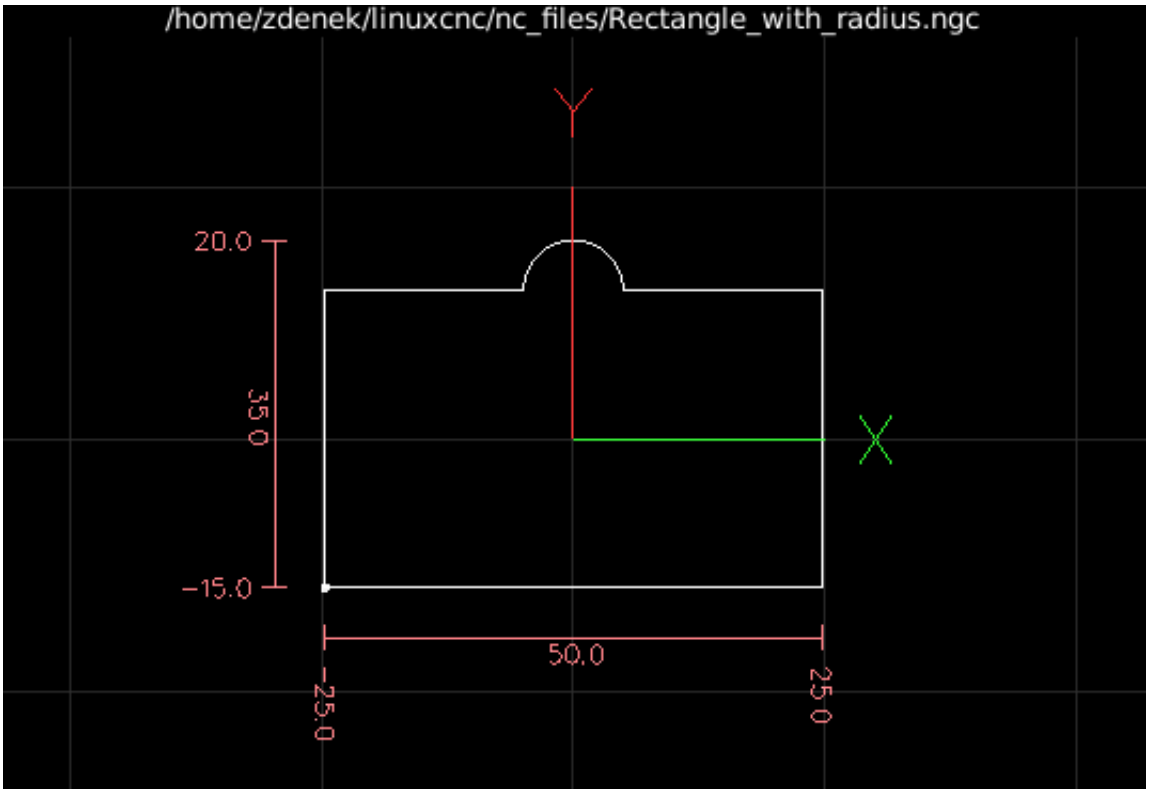



Figure 11.15: G64 Rectangle with radius before milling

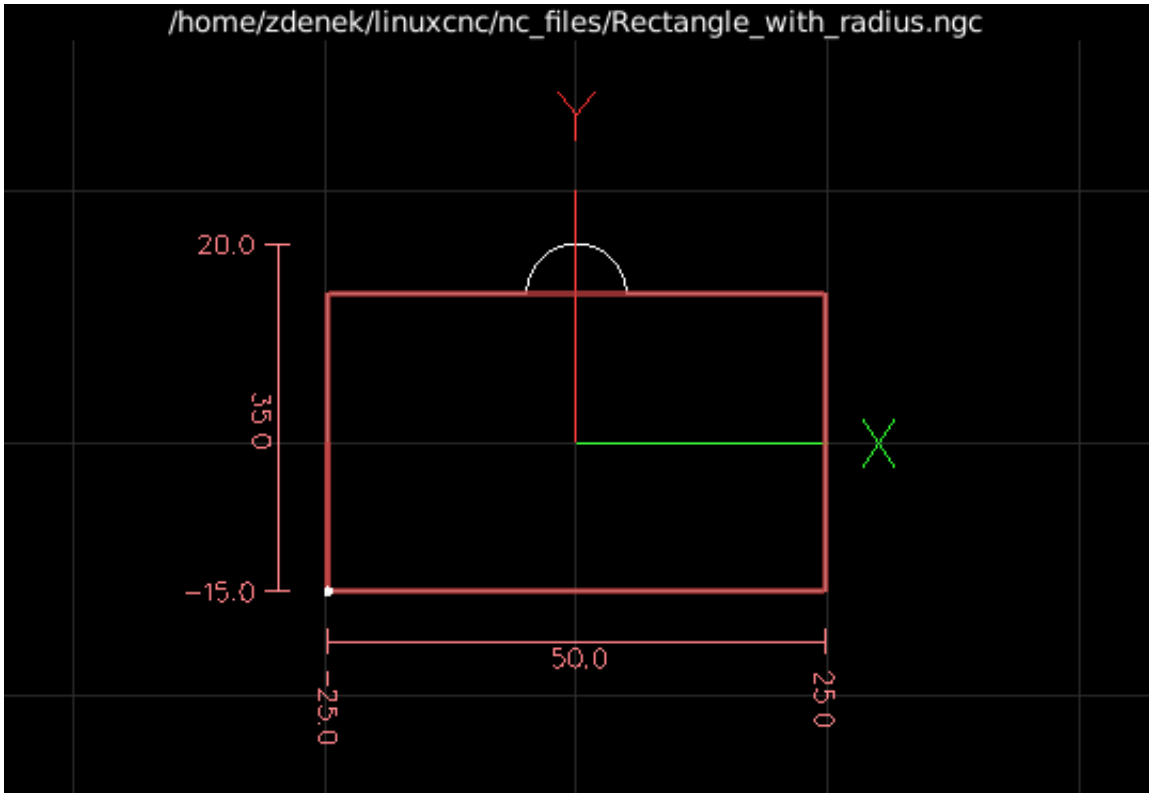



Figure 11.16: G64 Rectangle with radius after milling

```
G64 P0.015 Q2
```

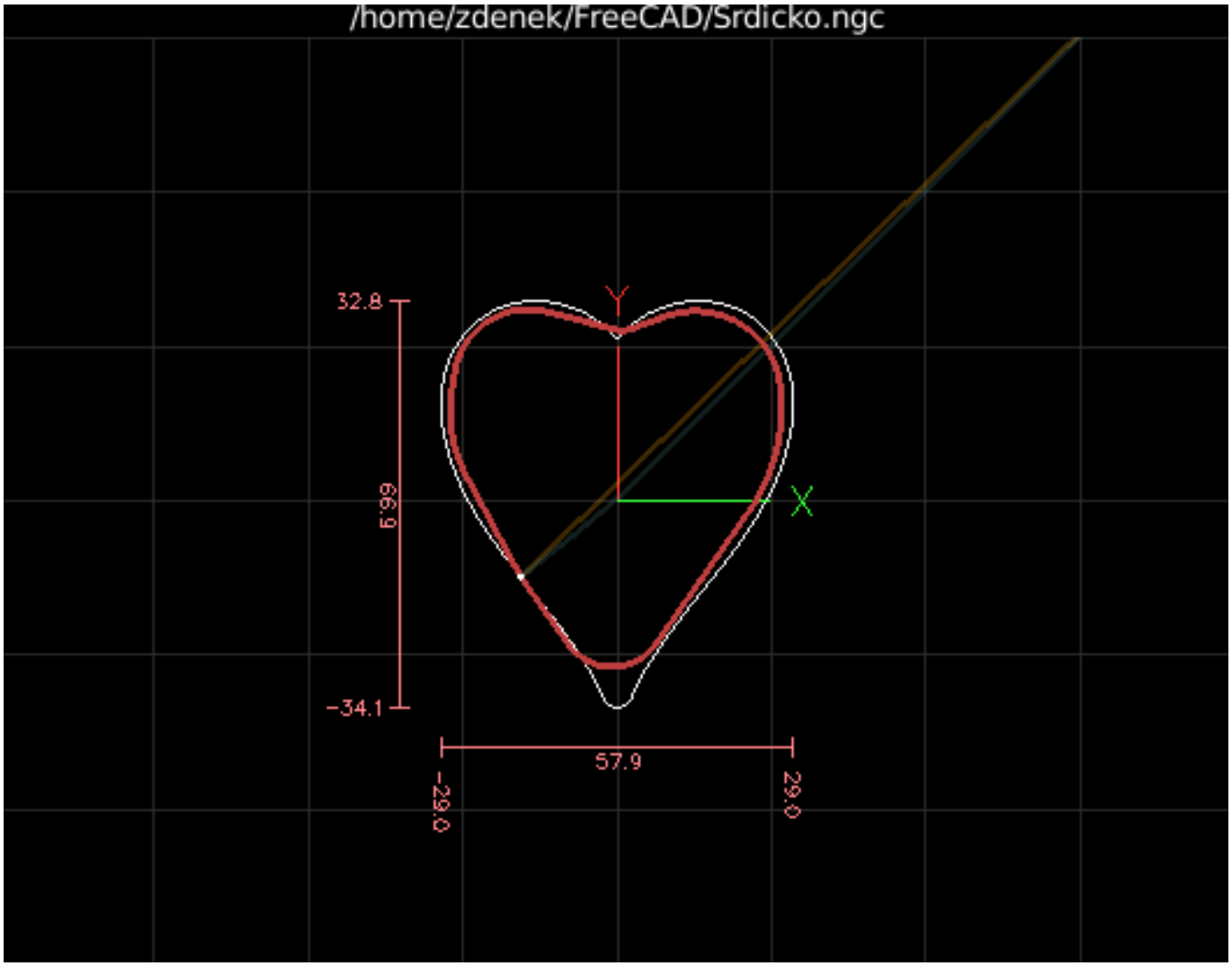



Figure 11.17: G64 Heart

### 11.5.38 G70 Lathe finishing cycle

```
G70 Q- <X-> <Z-> <D-> <E-> <P->
```

- *Q* - The subroutine number.
- *X* - The starting X position, defaults to the initial position.
- *Z* - The starting Z position, defaults to the initial position.
- *D* - The starting distance of the profile, defaults to 0.
- *E* - The ending distance of the profile, defaults to 0.
- *P* - The number of passes to use, defaults to 1.

The *G70* cycle is intended to be used after the shape of the profile given in the subroutine with number Q has been cut with G71 or G72.

- Preliminary motion.
  - If Z or X are used a rapid move to that position is done. This position is also used between each finishing pass.
  - Then a rapid move to the start of the profile is executed.
  - The path given in Q- is followed using the G1 and Section 11.5.5 commands.
  - If a next pass is required there is another rapid to the intermediate location, before a rapid is done to the start of the profile.
  - After the final pass, the tool is left at the end of the profile including E-.
- Multiple passes. The distance between the pass and the final profile is (pass-1)*(D-E)/P+E. Where pass the pass number and D,E and P are the D/E/P numbers.
- The distance is computed using the starting position of the cycle, with a positive distance towards this point.
- Fillet and chamfers in the profile. It is possible to add fillets or chamfers in the profile, see Section 11.5.39 for more details.

It is an error if:

- There is no subroutine defined with the number given in Q.
- The path given in the profile is not monotonic in Z or X.
- Section 11.5.18 has not been used to select the ZX plane.

### 11.5.39 G71 G72 Lathe roughing cycles

**Note**
The G71 and G72 cycles are currently somewhat fragile. See issue #2939 for example.

```
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* - The subroutine number.
- *X* - The starting X position, defaults to the initial position.
- *Z* - The starting Z position, defaults to the initial position.
- *D* - The remaining distance to the profile, defaults to 0.
- *I* - The cutting increment, defaults to 1.
- *R* - The retracting distance, defaults to 0.5.

The G71/G72 cycle is intended to rough cut a profile on a lathe. The G71 cycles remove layers of the material while traversing in the Z direction. The G72 cycles remove material while traversing the X axis, the so called facing cycle. The direction of travel is the same as in the path given in the subroutine. For the G71 cycle the Z coordinate must be monotonically changing, for the G72 this is required for the X axis.

The profile is given in a subroutine with number Q-. This subroutine may contain G0, G1, G2 and G3 motion commands. All other commands are ignored, including feed and speed settings. The Section 11.5.3 commands are interpreted as G1 commands. Each motion command may also include an optional A- or C- number. If the number A- is added a fillet with the radius given by A will be inserted at the endpoint of that motion, if this radius is too large the algorithm will fail with a non-monotonic path error. It is also possible to use the C- number, which allows a chamfer to be inserted. This chamfer has the same endpoints as a fillet of the same dimension would have but a straight line is inserted instead of an arc.

When in absolute mode the U (for X) and W (for Z) can be used as incremental displacements.

The G7x.1 cycles do not cut pockets. The G7x.2 cycles only cut after the first pocket and continue where G7x.1 stopped. It is advisible to leave some additional material to cut before the G7x.2 cycle, so if G7x.1 used a D1.0 the G7x.2 can use D0.5 and 0.5mm will be removed while moving from one pocket to the next.

The normal G7x cycles cut the entire profile in one cycle.

1. Preliminary motion.
   - If Z or X are used a rapid move to that position is done.
   - After the profile has been cut, the tool stops at the end of the profile, including the distance specified in D.
2. The D number is used to keep a distance from the final profile, to allow material to remain for finishing.

It is an error if:

- There is no subroutine defined with the number given in Q.
- The path given in the profile is not monotonic in Z or X.
- Section 11.5.18 has not been used to select the ZX plane.
- Section 11.5.26 is active.

### 11.5.40 G73 Drilling Cycle with Chip Breaking

```
G73 X- Y- Z- R- Q- D- <L->
```

- *R* - retract position along the Z axis.
- *Q* - delta increment along the Z axis.
- *L* - repeat

The *G73* cycle is drilling or milling with chip breaking. This cycle takes a Q number which represents a *delta* increment along the Z axis. Peck clearance can be specified by optional D number.

- Preliminary motion.

  - If the current Z position is below the R position, The Z axis does a rapid move to the R position.
  - Move to the X Y coordinates
- Move the Z-axis only at the current feed rate downward by delta or to the Z position, whichever is less deep.
- Rapid up either the D value, the G73_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Repeat steps 2 and 3 until the Z position is reached at step 2.
- The Z axis does a rapid move to the R position.

It is an error if:

- the Q number is negative or zero.
- the R number is not specified

### 11.5.41 G74 Left-hand Tapping Cycle with Dwell

```
G74 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Retract position along the Z axis.
- *L-* - Used in incremental mode; number of times to repeat the cycle. See G81 for examples.
- *P-* - Dwell time (seconds).
- *$-* - Selected spindle.
- *F-* - Feed rate (spindle speed multiplied by distance traveled per revolution (thread pitch)).

**Warning**
G74 does not use synchronized motion.

The *G74* cycle is intended for tapping with floating chuck and dwell at the bottom of the hole.

1. Preliminary motion, as described in the Preliminary and In-Between Motion section.
2. Disable Feed and Speed Overrides.
3. Move the Z-axis at the current feed rate to the Z position.
4. Stop the selected spindle (chosen by the $ parameter)
5. Start spindle rotation clockwise.
6. Dwell for the P number of seconds.
7. Move the Z-axis at the current feed rate to clear Z
8. Restore Feed and Speed override enables to previous state

The length of the dwell is specified by a *P-* word in the G74 block. The feed rate *F-* is spindle speed multiplied by distance per revolution (thread pitch). In example S100 with 1.25MM per revolution thread pitch gives a feed of F125.

### 11.5.42 G76 цикл нарезания резьбы

```
G76 P- Z- I- J- R- K- Q- H- E- L- $-
```

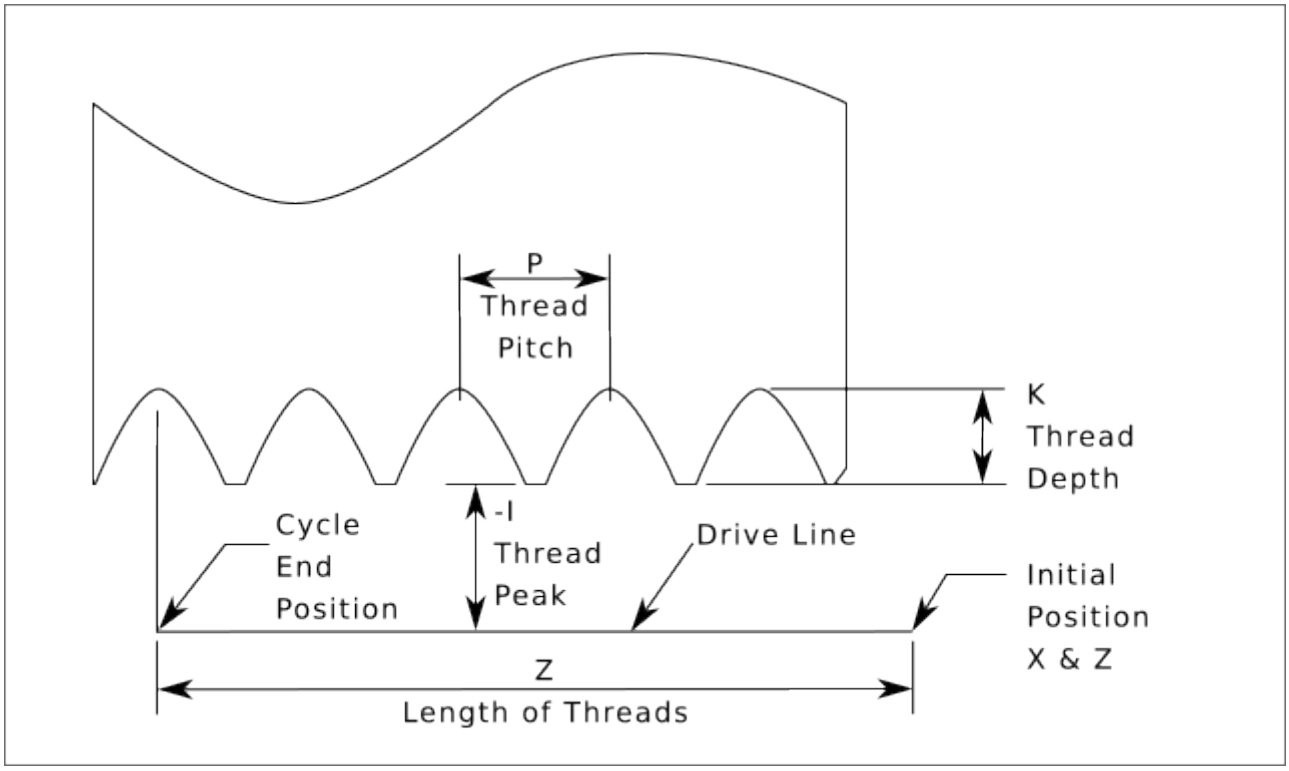



Figure 11.18: G76 Threading

- *Drive Line* - A line through the initial X position parallel to the Z.
- *P-* - The *thread pitch* in distance per revolution.
- *Z-* - The final position of threads. At the end of the cycle the tool will be at this Z position.

---

**Note**
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-* - The *thread peak* offset from the *drive line*. Negative *I* values are external threads, and positive *I* values are internal threads. Generally the material has been turned to this size before the *G76* cycle.
- *J-* - A positive value specifying the *initial cut depth*. The first threading cut will be *J* beyond the *thread peak* position.
- *K-* - A positive value specifying the *full thread depth*. The final threading cut will be *K* beyond the *thread peak* position.

Optional settings

- *$*- - 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.)

**Warning**
Unnecessarily high degression values will produce an unnecessarily high number of passes. (degressing = dive in stages)

- *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*- - The number of *spring passes*. Spring passes are additional passes at full thread depth. If no additional passes are desired, program *H0*.

Thread entries and exits can be programmed tapered with the *E* and *L* values.

- *E*- - Specifies the distance along the drive line used for the taper. The angle of the taper will be so the last pass tapers to the thread crest over the distance specified with E. *E0.2* will give a taper for the first/last 0.2 length units along the thread. For a 45 degree taper program E the same as K.
- *L*- - Specifies which ends of the thread get the taper. Program *L0* for no taper (the default), *L1* for entry taper, *L2* for exit taper, or *L3* for both entry and exit tapers. Entry tapers will pause at the drive line to synchronize with the index pulse then move at the feed rate in to the beginning of the taper. No entry taper and the tool will rapid to the cut depth then synchronize and begin the cut.

The tool is moved to the initial X and Z positions prior to issuing the G76. The X position is the *drive line* and the Z position is the start of the threads.

The tool will pause briefly for synchronization before each threading pass, so a relief groove will be required at the entry unless the beginning of the thread is past the end of the material or an entry taper is used.

Unless using an exit taper, the exit move is not synchronized to the spindle speed and will be a rapid move. With a slow spindle, the exit move might take only a small fraction of a revolution. If the spindle speed is increased after several passes are complete, subsequent exit moves will require a larger portion of a revolution, resulting in a very heavy cut during the exit move. This can be avoided by providing a relief groove at the exit, or by not changing the spindle speed while threading.

The final position of the tool will be at the end of the *drive line*. A safe Z move will be needed with an internal thread to remove the tool from the hole.

It is an error if:

- The active plane is not the ZX plane.
- Other axis words, such as X- or Y-, are specified.
- The *R*- degression value is less than 1.0.
- All the required words are not specified.

- *P-*, *J-*, *K-* or *H-* is negative.
- *E-* is greater than half the drive line length.

**HAL Connections** The pins *spindle.N.at-speed* and the *encoder*.n.*phase-Z* for the spindle must be connected in your HAL file before G76 will work. See the spindle pins in the Motion section for more information.

**Technical Info** The G76 canned cycle is based on the G33 Spindle Synchronized Motion. For more information see the G33 Technical Info.

The sample program *g76.ngc* shows the use of the G76 canned cycle, and can be previewed and executed on any machine using the *sim/lathe.ini* configuration.

**G76 Example Code**

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

In the figure the tool is in the final position after the G76 cycle is completed. You can see the entry path on the right from the Q29.5 and the exit path on the left from the L2 E0.045. The white lines are the cutting moves.

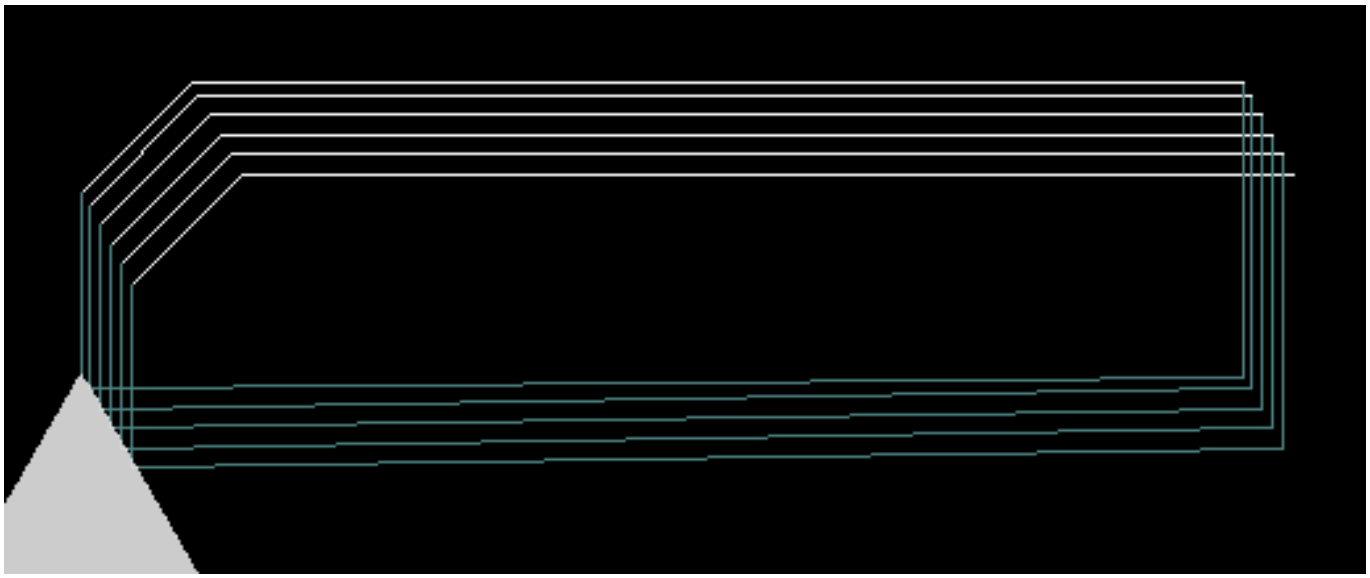


Figure 11.19: G76 Example

### 11.5.43 G80-G89 Canned Cycles

The canned cycles *G81* through *G89* and the canned cycle stop *G80* are described in this section.

All canned cycles are performed with respect to the currently-selected plane. Any of the nine planes may be selected. Throughout this section, most of the descriptions assume the XY-plane has been selected. The behavior is analogous if another plane is selected, and the correct words must be used. For instance, in the *G17.1* plane, the action of the canned cycle is along W, and the locations or increments are given with U and V. In this case substitute U,V,W for X,Y,Z in the instructions below.

Rotary axis words are not allowed in canned cycles. When the active plane is one of the XYZ family, the UVW axis words are not allowed. Likewise, when the active plane is one of the UVW family, the XYZ axis words are not allowed.

#### 11.5.43.1 Common Words

All canned cycles use X, Y, Z, or U, V, W groups depending on the plane selected and R words. The R (usually meaning retract) position is along the axis perpendicular to the currently selected plane (Z-axis for XY-plane, etc.) Some canned cycles use additional arguments.

#### 11.5.43.2 Sticky Words

For canned cycles, we will call a number *sticky* if, when the same cycle is used on several lines of code in a row, the number must be used the first time, but is optional on the rest of the lines. Sticky numbers keep their value on the rest of the lines if they are not explicitly programmed to be different. The R number is always sticky.

In incremental distance mode X, Y, and R numbers are treated as increments from the current position and Z as an increment from the Z-axis position before the move involving Z takes place. In absolute distance mode, the X, Y, R, and Z numbers are absolute positions in the current coordinate system.

#### 11.5.43.3 Repeat Cycle

The L number is optional and represents the number of repeats. L=0 is not allowed. If the repeat feature is used, it is normally used in incremental distance mode, so that the same sequence of motions is repeated in several equally spaced places along a straight line. When L- is greater than 1 in incremental mode with the XY-plane selected, the X and Y positions are determined by adding the given X and Y numbers either to the current X and Y positions (on the first go-around) or to the X and Y positions at the end of the previous go-around (on the repetitions). Thus, if you program *L10* , you will get 10 cycles. The first cycle will be distance X,Y from the original location. The R and Z positions do not change during the repeats. The L number is not sticky. In absolute distance mode, L>1 means *do the same cycle in the same place several times*, Omitting the L word is equivalent to specifying L=1.

#### 11.5.43.4 Retract Mode

The height of the retract move at the end of each repeat (called *clear Z* in the descriptions below) is determined by the setting of the retract mode, either to the original Z position (if that is above the R position and the retract mode is *G98*, OLD_Z), or otherwise to the R position. See the G98 G99 section.

#### 11.5.43.5 Canned Cycle Errors

It is an error if:

- axis words are all missing during a canned cycle,
- axis words from different groups (XYZ) (UVW) are used together,
- a P number is required and a negative P number is used,
- an L number is used that does not evaluate to a positive integer,
- rotary axis motion is used during a canned cycle,
- inverse time feed rate is active during a canned cycle,
- or cutter compensation is active during a canned cycle.

If the XY plane is active, the Z number is sticky, and it is an error if:

- the Z number is missing and the same canned cycle was not already active,
- or the R number is less than the Z number.

If other planes are active, the error conditions are analogous to the XY conditions above.

#### 11.5.43.6 Preliminary and In-Between Motion

Preliminary motion is a set of motions that is common to all of the milling canned cycles. If the current Z position is below the R position, the Z axis does a rapid move to the R position. This happens only once, regardless of the value of L.

In addition, at the beginning of the first cycle and each repeat, the following one or two moves are made:

- A rapid move parallel to the XY-plane to the given XY-position.
- The Z-axis make a rapid move to the R position, if it is not already at the R position.

If another plane is active, the preliminary and in-between motions are analogous.

#### 11.5.43.7 Why use a canned cycle?

There are at least two reasons for using canned cycles. The first is the economy of code. A single bore would take several lines of code to execute.

The G81 Example 1 demonstrates how a canned cycle could be used to produce 8 holes with ten lines of G-code within the canned cycle mode. The program below will produce the same set of 8 holes using five lines for the canned cycle. It does not follow exactly the same path nor does it drill in the same order as the earlier example. But the program writing economy of a good canned cycle should be obvious.

---

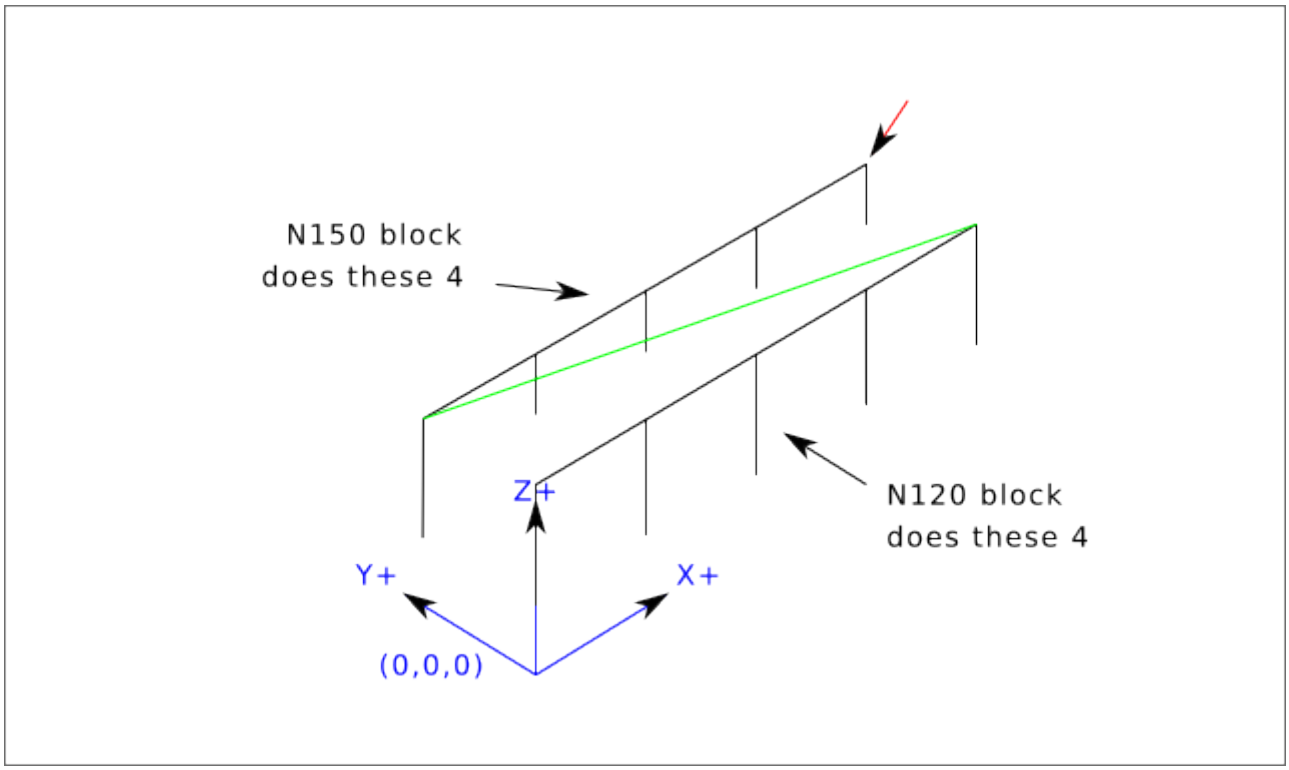
**Note**
Line numbers are not needed but help clarify these examples.

---

**Eight Holes**

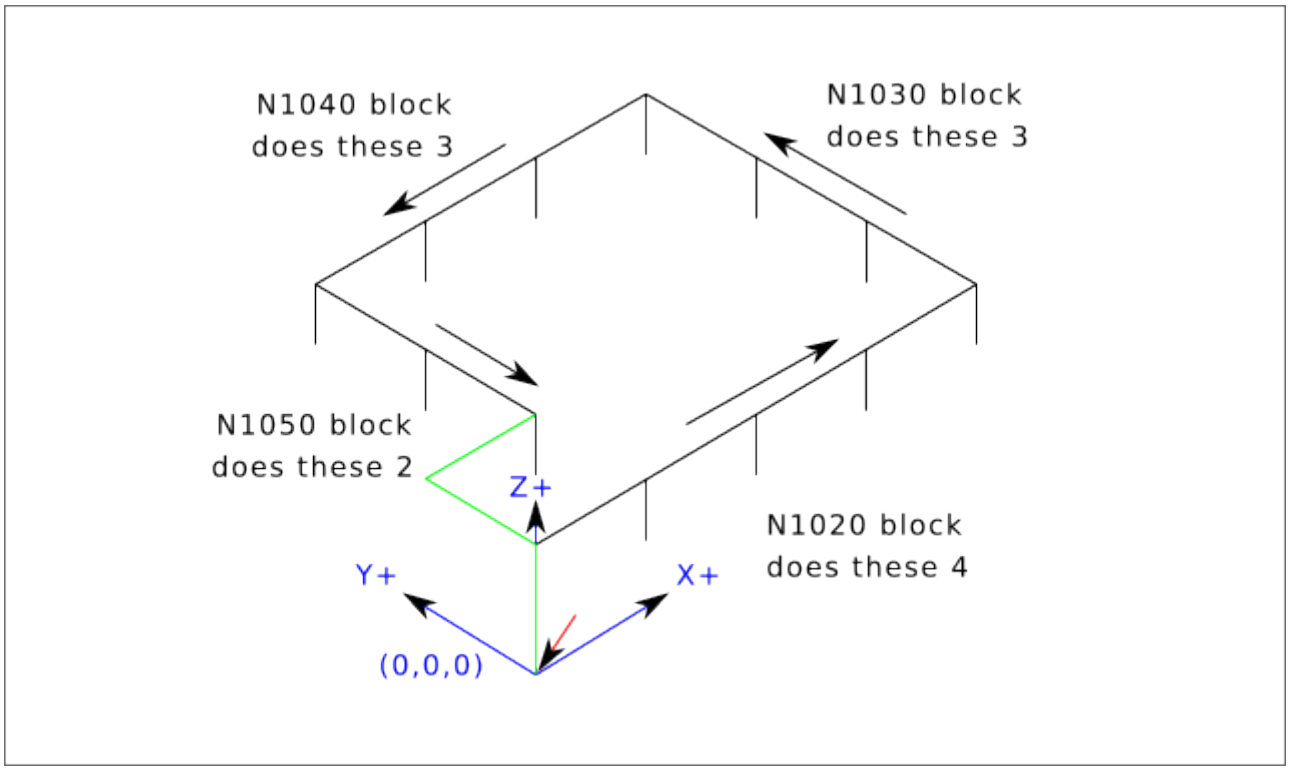
```
N100 G90 G0 X0 Y0 Z0 (move coordinate home)
N110 G1 F10 X0 G4 P0.1
N120 G91 G81 X1 Y0 Z-1 R1 L4(canned drill cycle)
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0.5 R1 L4(canned drill cycle)
N160 G80 (turn off canned cycle)
N170 M2 (program end)
```

The G98 on the second line above means that the return move will be to the Z value on the first line since it is higher than the specified R value.




**Twelve Holes in a Square** This example demonstrates the use of the L word to repeat a set of incremental drill cycles for successive blocks of code within the same G81 motion mode. Here we produce 12 holes using five lines of code in the canned motion mode.

```
N1000 G90 G0 X0 Y0 Z0 (move coordinate home)
N1010 G1 F50 X0 G4 P0.1
N1020 G91 G81 X1 Y0 Z-0.5 R1 L4 (canned drill cycle)
N1030 X0 Y1 R0 L3 (repeat)
N1040 X-1 Y0 L3 (repeat)
N1050 X0 Y-1 L2 (repeat)
N1060 G80 (turn off canned cycle)
N1070 G90 G0 X0 (rapid move home)
N1080 Y0
N1090 Z0
N1100 M2 (program end)
```




The second reason to use a canned cycle is that they all produce preliminary moves and returns that you can anticipate and control regardless of the start point of the canned cycle.

### 11.5.44 G80 Cancel Canned Cycle

- *G80* - cancel canned cycle modal motion. *G80* is part of modal group 1, so programming any other G-code from modal group 1 will also cancel the canned cycle.

It is an error if:

- Axis words are programmed when G80 is active.

**G80 Example**

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G80 (turn off canned cycle motion)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The following code produces the same final position and machine state as the previous code.

**G0 Example**

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The advantage of the first set is that, the G80 line clearly turns off the G81 canned cycle. With the first set of blocks, the programmer must turn motion back on with G0, as is done in the next line, or any other motion mode G word.

If a canned cycle is not turned off with G80 or another motion word, the canned cycle will attempt to repeat itself using the next block of code that contains an X, Y, or Z word. The following file drills (G81) a set of eight holes as shown in the following caption.

**G80 Example 1**

```
N100 G90 G0 X0 Y0 Z0 (coordinate home)
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (canned drill cycle)
N130 X2
N140 X3
N150 X4
N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (turn off canned cycle)
N210 G0 X0 (rapid move home)
N220 Y0
N230 Z0
N240 M2 (program end)
```

**Note**
Notice the Z position change after the first four holes. Also, this is one of the few places where line numbers have some value, being able to point a reader to a specific line of code.

The use of G80 in line N200 is optional because the G0 on the next line will turn off the G81 cycle. But using the G80 as shown in Example 1, will provide for easier to read canned cycle. Without it, it is not so obvious that all of the blocks between N120 and N200 belong to the canned cycle.

### 11.5.45 G81 Drilling Cycle

```
G81 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G81* cycle is intended for drilling.

The cycle functions as follows:

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis at the current feed rate to the Z position.
- The Z-axis does a rapid move to clear Z.

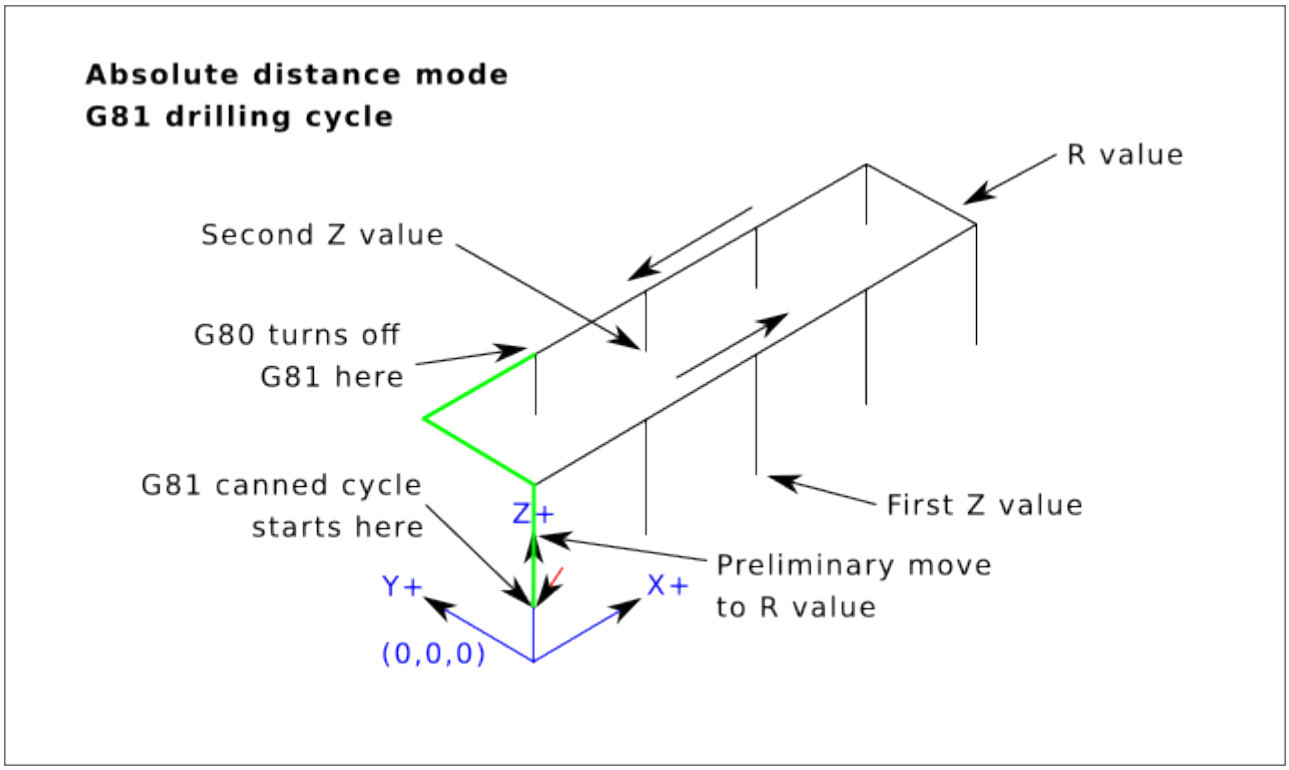



Figure 11.20: G81 Cycle

**Example 1 - Absolute Position G81**

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

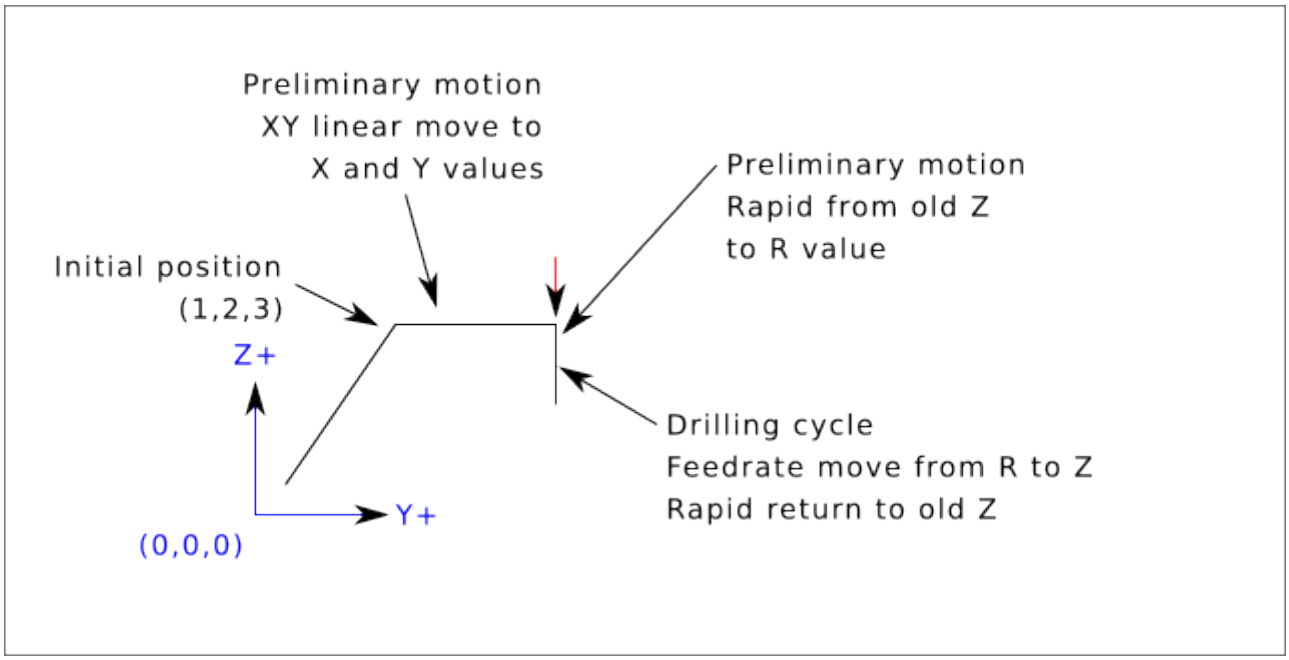
Suppose the current position is (X1, Y2, Z3) and the preceding line of NC code is interpreted.

This calls for absolute distance mode (G90) and OLD_Z retract mode (G98) and calls for the G81 drilling cycle to be performed once.

- The X value and X position are 4.
- The Y value and Y position are 5.
- The Z value and Z position are 1.5.
- The R value and clear Z are 2.8. OLD_Z is 3.

The following moves take place:

- A rapid move parallel to the XY plane to (X4, Y5)
- A rapid move move parallel to the Z-axis to (Z2.8).
- Move parallel to the Z-axis at the feed rate to (Z1.5)
- A rapid move parallel to the Z-axis to (Z3)




**Example 2 - Relative Position G81**

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Suppose the current position is (X1, Y2, Z3) and the preceding line of NC code is interpreted.

This calls for incremental distance mode (G91) and OLD_Z retract mode (G98). It also calls for the G81 drilling cycle to be repeated three times. The X value is 4, the Y value is 5, the Z value is -0.6 and the R value is 1.8. The initial X position is 5 (=1+4), the initial Y position is 7 (=2+5), the clear Z position is 4.8 (=1.8+3), and the Z position is 4.2 (=4.8-0.6). OLD_Z is 3.

The first preliminary move is a maximum rapid move along the Z axis to (X1,Y2,Z4.8), since OLD_Z < clear Z.

The first repeat consists of 3 moves.

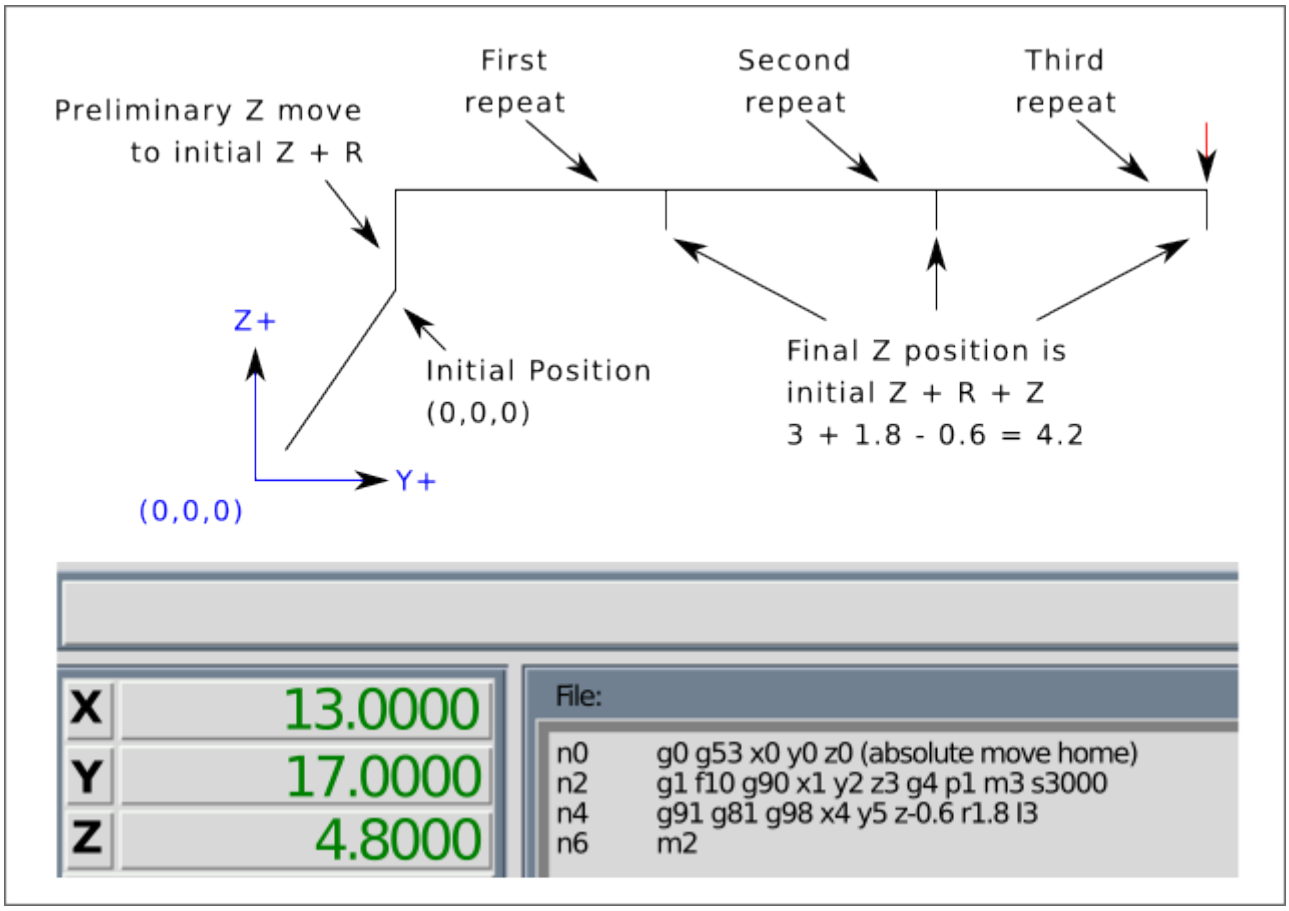
- A rapid move parallel to the XY-plane to (X5, Y7)
- Move parallel to the Z-axis at the feed rate to (Z4.2)
- A rapid move parallel to the Z-axis to (X5, Y7, Z4.8)

The second repeat consists of 3 moves. The X position is reset to 9 (=5+4) and the Y position to 12 (=7+5).

- A rapid move parallel to the XY-plane to (X9, Y12, Z4.8)
- Move parallel to the Z-axis at the feed rate to (X9, Y12, Z4.2)
- A rapid move parallel to the Z-axis to (X9, Y12, Z4.8)

The third repeat consists of 3 moves. The X position is reset to 13 (=9+4) and the Y position to 17 (=12+5).

- A rapid move parallel to the XY-plane to (X13, Y17, Z4.8)
- Move parallel to the Z-axis at the feed rate to (X13, Y17, Z4.2)
- A rapid move parallel to the Z-axis to (X13, Y17, Z4.8)

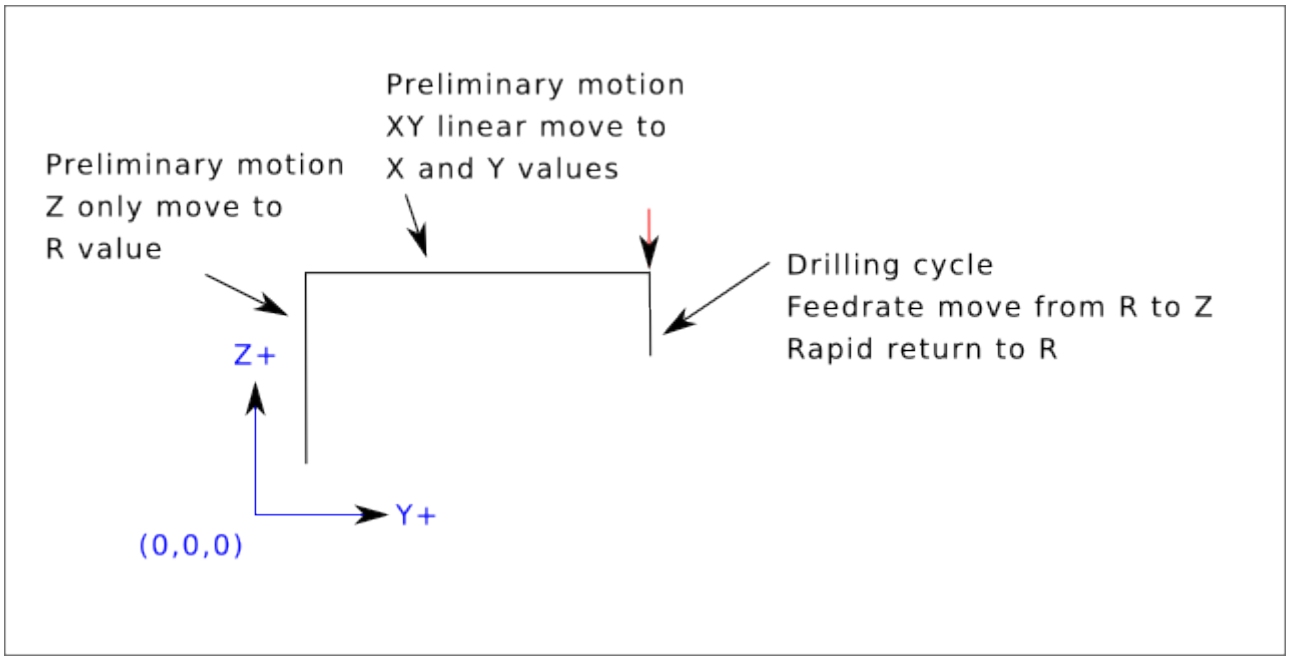



**Example 3 - Relative Position G81**

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

Now suppose that you execute the first G81 block of code but from (X0, Y0, Z0) rather than from (X1, Y2, Z3).

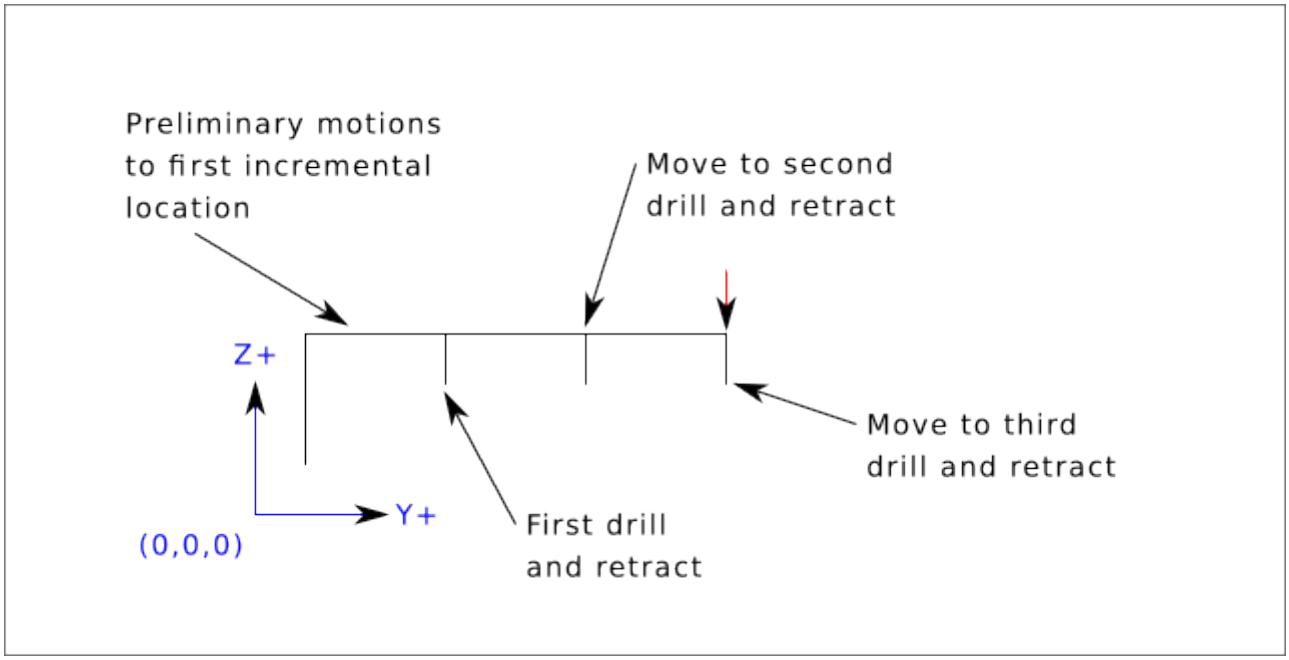
Since OLD_Z is below the R value, it adds nothing for the motion but since the initial value of Z is less than the value specified in R, there will be an initial Z move during the preliminary moves.




**Example 4 - Absolute G81 R > Z** This is a plot of the path of motion for the second g81 block of code.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location. After that initial Z move, the repeat feature works the same as it did in example 3 with the final Z depth being 0.6 below the R value.




**Example 5 - Relative position R > Z**

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location as in *Example 4*. After that initial Z move, the rapid move to X4 Y5 is done. Then the

final Z depth being 0.6 below the R value. The repeat function would make the Z move in the same location again.

### 11.5.46 G82 цикл сверления,задержка

```
G82 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G82* cycle is intended for drilling with a dwell at the bottom of the hole.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis at the current feed rate to the Z position.
- Dwell for the P number of seconds.
- The Z-axis does a rapid move to clear Z.

The motion of a G82 canned cycle looks just like G81 with the addition of a dwell at the bottom of the Z move. The length of the dwell is specified by a *P-* word in the G82 block.

```
G90 G82 G98 X4 Y5 Z1.5 R2.8 P2
```

This will be similar to example 3 above, just with an added dwell of 2 seconds at the bottom of the hole.

### 11.5.47 G83 Цикл сверления с частичным повторением импульсов

```
G83 (X- Y- Z-) or (U- V- W-) R- L- Q- D-
```

The *G83* cycle (often called peck drilling) is intended for deep drilling or milling with chip breaking. The retracts in this cycle clear the hole of chips and cut off any long stringers (which are common when drilling in aluminum). This cycle takes a Q number which represents a *delta* increment along the Z-axis. The retract before final depth will always be to the *retract* plane even if G98 is in effect. The final retract will honor the G98/99 in effect. G83 functions the same as G81 with the addition of retracts during the drilling operation. Peck clearance can be specified by optional D number.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis at the current feed rate downward by delta or to the Z position, whichever is less deep.
- Rapid move back out to the retract plane specified by the R word.
- Rapid up either the D value, the G83_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Repeat steps 2, 3, and 4 until the Z position is reached at step 2.
- The Z-axis does a rapid move to clear Z.

It is an error if:

- the Q number is negative or zero.

### 11.5.48 G84 Right-hand Tapping Cycle, Dwell

```
G84 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Retract position along the Z axis.
- *L-* - Used in incremental mode; number of times to repeat the cycle. See G81 for examples.
- *P-* - Dwell time (seconds).
- *$-* - Selected spindle.
- *F-* - Feed rate (spindle speed multiplied by distance traveled per revolution (thread pitch)).

**Warning**
G84 does not use synchronized motion.

The *G84* cycle is intended for tapping with floating chuck and dwell at the bottom of the hole.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Disable Feed and Speed Overrides.
- Move the Z-axis at the current feed rate to the Z position.
- Stop the selected spindle (chosen by the $ parameter)
- Start spindle rotation counterclockwise.
- Dwell for the P number of seconds.
- Move the Z-axis at the current feed rate to clear Z
- Restore Feed and Speed override enables to previous state

The length of the dwell is specified by a *P-* word in the G84 block. The feed rate *F-* is spindle speed multiplied by distance per revolution (thread pitch). In example S100 with 1.25MM per revolution thread pitch gives a feed of F125.

### 11.5.49 G85 Цикл растачивания, подача

```
G85 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G85* cycle is intended for boring or reaming, but could be used for drilling or milling.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis only at the current feed rate to the Z position.
- Retract the Z-axis at the current feed rate to the R plane if it is lower than the initial Z.
- Retract at the traverse rate to clear Z.

### 11.5.50 G86 цикл растачивания, остановка шпинделя, быстрый выход

```
G86 (X- Y- Z-) or (U- V- W-) R- L- P- $-
```

The *G86* cycle is intended for boring. This cycle uses a P number for the number of seconds to dwell.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis only at the current feed rate to the Z position.
- Dwell for the P number of seconds.
- Stop the selected spindle turning. (Chosen by the $ parameter)
- The Z-axis does a rapid move to clear Z.
- Restart the spindle in the direction it was going.

It is an error if:

- the spindle is not turning before this cycle is executed.

### 11.5.51 G87 цикл обратного растачивания

This code is currently unimplemented in LinuxCNC. It is accepted, but the behavior is undefined.

### 11.5.52 G88 цикл растачивания, остановка шпинделя, ручной отвод

This code is currently unimplemented in LinuxCNC. It is accepted, but the behavior is undefined.

### 11.5.53 G89 цикл развертывания, задержка, вывод на рабочей подаче

```
G89 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G89* cycle is intended for boring. This cycle uses a P number, where P specifies the number of seconds to dwell.

- Preliminary motion, as described in the Preliminary and In-Between Motion section.
- Move the Z-axis only at the current feed rate to the Z position.
- Dwell for the P number of seconds.
- Retract the Z-axis at the current feed rate to clear Z.

### 11.5.54 G90, G91 Distance Mode

- *G90* - absolute distance mode In absolute distance mode, axis numbers (X, Y, Z, A, B, C, U, V, W) usually represent positions in terms of the currently active coordinate system. Any exceptions to that rule are described explicitly in the G80 G89 section.
- *G91* - incremental distance mode In incremental distance mode, axis numbers usually represent increments from the current coordinate.

#### G90 Example

```
G90 (set absolute distance mode)
G0 X2.5 (rapid move to coordinate X2.5 including any offsets in effect)
```

#### G91 Example

```
G91 (set incremental distance mode)
G0 X2.5 (rapid move 2.5 from current position along the X axis)
```

- See G0 section for more information.

### 11.5.55 G90.1, G91.1 Arc Distance Mode

- *G90.1* - absolute distance mode for I, J & K offsets. When G90.1 is in effect I and J both must be specified with G2/3 for the XY plane or J and K for the XZ plane or it is an error.
- *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 axes
```

**Warning**
Only use *G92* after your machine has been positioned to the desired point.

*G92* makes the current point have the coordinates you want (without motion), where the axis words contain the axis numbers you want. All axis words are optional, except that at least one must be used. If an axis word is not used for a given axis, the offset for that axis will be zero.

When *G92* is executed, the origins of all coordinate systems move. They move such that the value of the current controlled point, in the currently active coordinate system, becomes the specified value. All of the coordinate system's origins (G53-G59.3) are offset this same distance.

*G92* uses the values stored in parameters 5211-5219 as the X Y Z A B C U V W offset values for each axis. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the INI file. All axes defined in the INI file will be offset when G92 is active. If an axis was not entered following the G92, that axis' offset will be zero.

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.

It is an error if all axis words are omitted.

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).

**Note**
The *G52* command can also be used to change this offset; see the Local and Global Offsets section for more details about G92 and G52 and how they interact.

See the Coordinate System section for an overview of coordinate systems.

See the Parameters section for more information.

### 11.5.57 G92.1, G92.2 Reset G92 Offsets

- *G92.1* - turn off G92 offsets and reset parameters 5211 - 5219 to zero.
- *G92.2* - turn off G92 offsets but keep parameters 5211 - 5219 available.

**Note**
G92.1 only clears G92 offsets, to change G53-G59.3 coordinate system offsets in G-code use either G10 L2 or 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* - is Inverse Time Mode. In inverse time feed rate mode, an F word means the move should be completed in [one divided by the F number] minutes. For example, if the F number is 2.0, the move should be completed in half a minute.

  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* - is Units per Minute Mode. In units per minute feed mode, an F word is interpreted to mean the controlled point should move at a certain number of inches per minute, millimeters per minute, or degrees per minute, depending upon what length units are being used and which axis or axes are moving.
- *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.

It is an error if:

- 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- <$-> (Constant Surface Speed Mode)
G97 S- <$-> (RPM Mode)
```

1. *D* - maximum rotation speed (RPM), optional
2. *S* - spindle speed
3. *$* - the spindle of which the speed will be varied, optional.
   - *G96 S- <D->* - selects constant surface speed of *S*:
     - In feet per minute if `G20` is in effect,
     - or meters per minute if `G21` is in effect.

When using G96, ensure that X0 in the current coordinate system (including offsets and tool lengths) is the center of rotation or LinuxCNC will not give the desired surface speed. G96 is not affected by radius or diameter mode.

To achieve CSS mode on selected spindles programme successive G96 commands for each spindle prior to issuing M3.

- *G97* selects RPM mode.

**G96 Example Line**

```
G96 D2500 S250 (set CSS with a max rpm of 2500 and a surface speed of 250)
```

It is an error if:

- S is not specified with G96
- A feed move is specified in G96 mode while the spindle is not turning

### 11.5.61 G98, G99 уровень возврата стандартного цикла

When spindle retracts during canned cycles, there are two options to choose from for the way it does it:

- *G98* - retract to the position that axis was in just before this series of one or more contiguous canned cycles was started.
- *G99* - retract to the position specified by the R word of the canned cycle.

Program a *G98* and the canned cycle will use the Z position prior to the canned cycle as the Z return position if it is higher than the R value specified in the cycle. If it is lower, the R value will be used. The R word has different meanings in absolute distance mode and incremental distance mode.

**G98 Retract to Origin**

```
G0 X1 Y2 Z3
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

The G98 to the second line above means that the return move will be to the value of Z in the first line since it is higher that the R value specified.

The *initial* (G98) plane is reset any time cycle motion mode is abandoned, whether explicitly (G80) or implicitly (any motion code that is not a cycle). Switching among cycle modes (say G81 to G83) does NOT reset the *initial* plane. It is possible to switch between G98 and G99 during a series of cycles.

# 11.6 M-Codes

## 11.6.1 M-Code Quick Reference Table

| Code | Description |
|---|---|
| M0 M1 | Program Pause |
| M2 M30 | Program End |
| M60 | Pallet Change Pause |
| M3 M4 M5 | Управление шпинделем |
| M6 | Tool Change |
| M7 M8 M9 | Coolant Control |
| M19 | Orient Spindle |
| M48 M49 | Feed & Spindle Overrides Enable/Disable |
| M50 | Feed Override Control |
| M51 | Spindle Override Control |
| M52 | Adaptive Feed Control |
| M53 | Feed Stop Control |
| M61 | Set Current Tool Number |
| m62-m65 | Output Control |
| M66 | Input Control |
| M67 | Analog Output Control |
| M68 | Analog Output Control |
| M70 | Save Modal State |
| M71 | Invalidate Stored Modal State |
| M72 | Restore Modal State |
| M73 | Save Autorestore Modal State |
| M98 M99 | Call and Return From Subprogram |
| M100-M199 | User Defined M-Codes |

## 11.6.2 M0, M1 Program Pause

* *M0* - pause a running program temporarily. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line. * *M1* - pause a running program temporarily if the optional stop switch is on. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the resume button will restart the program at the following line.

**Note**
It is OK to program *M0* and *M1* in MDI mode, but the effect will probably not be noticeable, because normal behavior in MDI mode is to stop after each line of input anyway.

## 11.6.3 M2, M30 конец программы

* *M2* - end the program. Pressing `Cycle Start` (”R” in the Axis GUI) will restart the program at the beginning of the file. * *M30* - exchange pallet shuttles and end the program. Pressing `Cycle Start` will start the program at the beginning of the file.

Both of these commands have the following effects:

- Change from Auto mode to MDI mode.

- Origin offsets are set to the default (like *G54*).
- Selected plane is set to XY plane (like *G17*).
- Distance mode is set to absolute mode (like *G90*).
- Feed rate mode is set to units per minute (like *G94*).
- Feed and speed overrides are set to ON (like *M48*).
- Cutter compensation is turned off (like *G40*).
- The spindle is stopped (like *M5*).
- The current motion mode is set to feed (like *G1*).
- Coolant is turned off (like *M9*).

**Note**
Lines of code after M2/M30 will not be executed. Pressing `Cycle Start` will start the program at the beginning of the file.

**Warning**
Using % to wrap the G-code does not do the same thing as a *Program End*. See the section on File Requirements for more information on what using % does not do.

### 11.6.4 M60 Pallet Change Pause

* *M60* - exchange pallet shuttles and then pause a running program temporarily (regardless of the setting of the optional stop switch). Pressing the cycle start button will restart the program at the following line.

### 11.6.5 M3, M4, M5 Spindle Control

* *M3 [$n]* - start the selected spindle clockwise at the *S* speed. * *M4 [$n]* - start the selected spindle counterclockwise at the *S* speed. * *M5 [$n]* - stop the selected spindle.

Use $ to operate on specific spindles. If $ is omitted then the commands default to operating on spindle 0. Use $-1 to operate on all active spindles.

This example will start spindles 0, 1, and 2 simultaneously at different speeds:

```
S100 $0
S200 $1
S300 $2
M3 $-1
```

This example will then reverse spindle 1 but leave the other spindles rotating forwards:

```
M4 $1
```

And this will stop spindle 2 and leave the other spindles rotating:

```
M5 $2
```

If the $ is omitted then behaviour is exactly as normal for a single spindle machine.

It is OK to use *M3* or *M4* if the S spindle speed is set to zero. If this is done (or if the speed override switch is enabled and set to zero), the spindle will not start turning. If, later, the spindle speed is set above zero (or the override switch is turned up), the spindle will start turning. It is OK to use *M3* or *M4* when the spindle is already turning or to use *M5* when the spindle is already stopped.

## 11.6.6 M6 Смена инструмента

### 11.6.6.1 Manual Tool Change

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 Tool Changer

To change a tool in the spindle from the tool currently in the spindle to the tool most recently selected (using a T word - see section Select Tool), program *M6*. When the tool change is complete:

- The spindle will be stopped.
- The tool that was selected (by a T word on the same line or on any line after the previous tool change) will be in the spindle.
- If the selected tool was not in the spindle before the tool change, the tool that was in the spindle (if there was one) will be placed back into the tool changer magazine.
- If configured in the INI file some axis positions may move when a M6 is issued. See the EMCIO section for more information on tool change options.
- No other changes will be made. For example, coolant will continue to flow during the tool change unless it has been turned off by an *M9*.

**Note**
The *T*- word is an integer number designating the tool pocket number in the carousel (not its index).

**Warning**
The tool length offset is not changed by *M6*, use *G43* after the *M6* to change the tool length offset.

The tool change may include axis motion. It is OK (but not useful) to program a change to the tool already in the spindle. It is OK if there is no tool in the selected slot; in that case, the spindle will be empty after the tool change. If slot zero was last selected, there will definitely be no tool in the spindle after a tool change. The tool changer will have to be setup to perform the tool change in HAL and possibly ClassicLadder.

## 11.6.7 M7, M8, M9 Coolant Control

* *M7* - turn mist coolant on. M7 controls iocontrol.0.coolant-mist pin. * *M8* - turn flood coolant on. M8 controls iocontrol.0.coolant-flood pin. * *M9* - turn both M7 and M8 off.

Connect one or both of the coolant control pins in HAL before M7 or M8 will control an output. M7 and M8 can be used to turn on any output via G-code.

It is OK to use any of these commands, regardless of the current coolant state.

### 11.6.8 M19 Orient Spindle

```
M19 R- Q- [P-] [$-]
```

- *R* Position to rotate to from 0, valid range is 0-360 degrees
- *Q* Number of seconds to wait until orient completes. If spindle.N.is-oriented does not become true within Q timeout an error occurs.
- *P* Direction to rotate to position.
  - *0* rotate for smallest angular movement (default)
  - *1* always rotate clockwise (same as M3 direction)
  - *2* always rotate counterclockwise (same as M4 direction)
- *$* The spindle to orient (actually only determines which HAL pins carry the spindle position commands)

M19 is a command of modal group 7, like M3, M4 and M5. M19 is cleared by any of M3,M4,M5.

Spindle orientation requires a quadrature encoder with an index to sense the spindle shaft position and direction of rotation.

INI Settings in the [RS274NGC] section:

- ORIENT_OFFSET = 0-360 (fixed offset in degrees added to M19 R word)
- HAL Контакты
  - *spindle.N.orient-angle* (out float) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET INI parameter.
  - *spindle.N.orient-mode* (out s32) Desired spindle rotation mode. Reflects M19 P parameter word, default = 0.
  - *spindle.N.orient* (out bit) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
  - *spindle.N.is-oriented* (in bit) Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.
  - *spindle.N.orient-fault* (in s32) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
  - *spindle.N.locked* (out bit) Spindle orient complete pin. Cleared by any of M3,M4,M5.

### 11.6.9 M48, M49 Speed and Feed Override Control

* *M48* - enable the spindle speed and feed rate override controls. * *M49* - disable both controls.

These commands also take an optional $ parameter to determine which spindle they operate on.

It is OK to enable or disable the controls when they are already enabled or disabled. See the Feed Rate section for more details.

They also can be be toggled individually using *M50* and *M51*, see below.

### 11.6.10 M50 Управление переопределения подачи

* *M50 <P1>* - enable the feed rate override control. The P1 is optional. * *M50 P0* - disable the feed rate control.

While disabled the feed override will have no influence, and the motion will be executed at programmed feed rate. (unless there is an adaptive feed rate override active).

### 11.6.11 M51 Управление переопределения скорости шпинделя

* *M51 <P1> <$->*- enable the spindle speed override control for the selected spindle. The P1 is optional. * *M51 P0 <$->* - disable the spindle speed override control program.

While disabled the spindle speed override will have no influence, and the spindle speed will have the exact program specified value of the S-word (described in the Spindle Speed section).

### 11.6.12 M52 Адаптивное управление подачей

* *M52 <P1>* - use an adaptive feed. The P1 is optional. * *M52 P0* - stop using adaptive feed.

When adaptive feed is enabled, some external input value is used together with the user interface feed override value and the commanded feed rate to set the actual feed rate. In LinuxCNC, the HAL pin *motion.adaptive-feed* is used for this purpose. The range for *motion.adaptive-feed* is defined by the MAX_FEED_OVERRIDE value in the [DISPLAY] section of the ini file and will be limited to a range of -MAX_FEED_OVERRIDE to MAX_FEED_OVERRIDE. 0 is equivalent to feed-hold.

**Note**
The use of negative adaptive-feed for reverse run is a new feature and is not very well tested as yet. The intended use is for plasma cutters and wire spark eroders but it is not limited to such applications.

### 11.6.13 M53 Контроль скорости подачи

* *M53 <P1>* - enable the feed stop switch. The P1 is optional. Enabling the feed stop switch will allow motion to be interrupted by means of the feed stop control. In LinuxCNC, the HAL pin *motion.feed-hold* is used for this purpose. A *true* value will cause the motion to stop when *M53* is active. * *M53 P0* - disable the feed stop switch. The state of *motion.feed-hold* will have no effect on feed when M53 is not active.

### 11.6.14 M61 Set Current Tool

* *M61 Q-* - change the current tool number while in MDI or Manual mode without a tool change. One use is when you power up LinuxCNC with a tool currently in the spindle you can set that tool number without doing a tool change.

**Warning**
The tool length offset is not changed by *M61*, use *G43* after the *M61* to change the tool length offset.

It is an error if:

- Q- is not 0 or greater

### 11.6.15 M62 - M65 Управление цифрового вывода

* *M62 P-* - turn on digital output synchronized with motion. * *M63 P-* - turn off digital output synchronized with motion. * *M64 P-* - turn on digital output immediately. * *M65 P-* - turn off digital output immediately.

The P- word specifies the digital output number. The P-word ranges from 0 to a default value of 3. If needed the the number of I/O can be increased by using the num_dio parameter when loading the motion controller. See the Motion section for more information.

The M62 & M63 commands will be queued. Subsequent commands referring to the same output number will overwrite the older settings. More than one output change can be specified by issuing more than one M62/M63 command.

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G-code (G0, G1, etc) right after the M62/63.

M64 & M65 happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending.

---

**Note**
M62-65 will not function unless the appropriate motion.digital-out-*nn* pins are connected in your HAL file to outputs.

---

### 11.6.16 M66 Wait on Input

```
M66 P- | E- <L->
```

- *P-* - specifies the digital input number from 0 to 3. (Adjustable from motmod argument num_dio)
- *E-* - specifies the analog input number from 0 to 3. (Adjustable from motmod argument num_aio)
- *L-* - specifies the wait mode type.
  - *Mode 0: IMMEDIATE* - no waiting, returns immediately. The current value of the input is stored in parameter #5399
  - *Mode 1: RISE* - waits for the selected input to perform a rise event.
  - *Mode 2: FALL* - waits for the selected input to perform a fall event.
  - *Mode 3: HIGH* - waits for the selected input to go to the HIGH state.
  - *Mode 4: LOW* - waits for the selected input to go to the LOW state.
- *Q-* - specifies the timeout in seconds for waiting. If the timeout is exceeded, the wait is interrupt, and the variable #5399 will be holding the value -1. The Q value is ignored if the L-word is zero (IMMEDIATE). A Q value of zero is an error if the L-word is non-zero.
- Mode 0 is the only one permitted for an analog input.

#### M66 Example Lines

```
M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)
```

M66 wait on an input stops further execution of the program, until the selected event (or the programmed timeout) occurs.

It is an error to program M66 with both a P-word and an E-word (thus selecting both an analog and a digital input). In LinuxCNC these inputs are not monitored in real time and thus should not be used for timing-critical applications.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Motion section for more information.

**Note**
M66 will not function unless the appropriate motion.digital-in-*nn* pins or motion.analog-in-*nn* pins are connected in your HAL file to an input.

**Example HAL Connection**

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

### 11.6.17 M67 Analog Output, Synchronized

```
M67 E- Q-
```

- *M67* - set an analog output synchronized with motion.
- *E-* - output number ranging from 0 to 3 (Adjustable from motmod argument num_aio)
- *Q-* - is the value to set (set to 0 to turn off).

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G-code (G0, G1, etc) right after the M67. M67 functions the same as M62-63.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Motion section for more information.

**Note**
M67 will not function unless the appropriate motion.analog-out-*nn* pins are connected in your HAL file to outputs.

### 11.6.18 M68 Analog Output, Immediate

```
M68 E- Q-
```

- *M68* - set an analog output immediately.
- *E-* - output number ranging from 0 to 3. (Adjustable from motmod argument num_aio)
- *Q-* - is the value to set (set to 0 to turn off).

M68 output happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending. M68 functions the same as M64-65.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Motion section for more information.

**Note**
M68 will not function unless the appropriate motion.analog-out-*nn* pins are connected in your HAL file to outputs.

### 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*.

A pair of *M70* and *M72* instructions will typically be used to protect a program against inadvertent modal changes within subroutines.

**M70 Saved state**

The state saved consists of:

- current G20/G21 settings (imperial/metric)
- selected plane (G17/G18/G19 G17.1,G18.1,G19.1)
- status of cutter compensation (G40,G41,G42,G41.1,G42,1)
- distance mode - relative/absolute (G90/G91)
- feed mode (G93/G94,G95)
- current coordinate system (G54-G59.3)
- tool length compensation status (G43,G43.1,G49)
- retract mode (G98,G99)
- spindle mode (G96-css or G97-RPM)
- arc distance mode (G90.1, G91.1)
- lathe radius/diameter mode (G7,G8)
- path control mode (G61, G61.1, G64)
- current feed and speed (*F* and *S* values)
- spindle status (M3,M4,M5) - on/off and direction
- mist (M7) and flood (M8) status
- speed override (M51) and feed override (M50) settings
- adaptive feed setting (M52)
- feed hold setting (M53)

Note that in particular, the motion mode (G1 etc) is NOT restored.

*current call level* means either:

- executing in the main program. There is a single storage location for state at the main program level; if several *M70* instructions are executed in turn, only the most recently saved state is restored when an *M72* is executed.
- executing within a G-code subroutine. The state saved with *M70* within a subroutine behaves exactly like a local named parameter - it can be referred to only within this subroutine invocation with an *M72* and when the subroutine exits, the parameter goes away.

A recursive invocation of a subroutine introduces a new call level.

### 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).

A subsequent *M72* at the same call level will fail.

If executed in a subroutine which protects modal state by an *M73*, a subsequent return or endsub will **not** restore modal state.

The usefulness of this feature is dubious. It should not be relied upon as it might go away.

### 11.6.21 M72 Restore Modal State

Modal state saved with an M70 code can be restored by executing an *M72*.

The handling of G20/G21 is specially treated as feeds are interpreted differently depending on G20/G21: if length units (mm/in) are about to be changed by the restore operation, 'M72 'will restore the distance mode first, and then all other state including feed to make sure the feed value is interpreted in the correct unit setting.

It is an error to execute an *M72* with no previous *M70* save operation at that level.

The following example demonstrates saving and explicitly restoring modal state around a subroutine call using *M70* and *M72*. Note that the *imperialsub* subroutine is not "aware" of the M7x features and can be used unmodified:

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call
O<imperialsub> endsub

; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)

(debug, in main, state now:)
o<showstate> call

M70 (save caller state in at global level)
```

```
O<imperialsub> call
M72 (explicitly restore state)

(debug, back in main, state now:)
o<showstate> call
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*.

Aborting a running program in a subroutine which has an *M73* operation will **not** restore state .

Also, the normal end (*M2*) of a main program which contains an *M73* will **not** restore state.

The suggested use is at the beginning of a O-word subroutine as in the following example. Using *M73* this way enables designing subroutines which need to modify modal state but will protect the calling program against inadvertent modal changes. Note the use of predefined named parameters in the *showstate* subroutine.

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
M73 (save caller state in current call context, restore on return or endsub)
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call

; note - no M72 is needed here - the following endsub or an
; explicit 'return' will restore caller state
O<imperialsub> endsub

; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)
(debug, in main, state now:)
o<showstate> call
o<imperialsub> call
(debug, back in main, state now:)
o<showstate> call
m2
```

### 11.6.23 M98 and M99

The interpreter supports Fanuc-style main- and sub-programs with the *M98* and *M99* M-codes. See Fanuc-Style Programs.

#### 11.6.23.1 Selectively Restoring Modal State

Executing an *M72* or returning from a subroutine which contains an *M73* will restore all modal state saved.

If only some aspects of modal state should be preserved, an alternative is the usage of predefined named parameters, local parameters and conditional statements. The idea is to remember the modes to be restored at the beginning of the subroutine, and restore these before exiting. Here is an example, based on snippet of *nc_files/tool-length-probe.ngc*:

```
O<measure> sub   (measure reference tool)
;
#<absolute> = #<_absolute>  (remember in local variable if G90 was set)
;
g30 (above switch)
g38.2 z0 f15 (measure)
g91 g0z.2 (off the switch)
#1000=#5063 (save reference tool length)
(print,reference length is #1000)
;
O<restore_abs> if [#<absolute>]
    g90 (restore G90 only if it was set on entry:)
O<restore_abs> endif
;
O<measure> endsub
```

### 11.6.24 M100-M199 определяемые пользователем команды

```
M1-- <P- Q->
```

- *M1--* - an integer in the range of 100 - 199.
- *P-* - a number passed to the file as the first parameter.
- *Q-* - a number passed to the file as the second parameter.

**Note**
After creating a new *M1nn* file you must restart the GUI so it is aware of the new file, otherwise you will get an *Unknown m code* error.

The external program named *M100* through *M199* (no extension, a capital M, found in directory pointed by *[DISPLAY] PROGRAM_PREFIX* parameter of the INI file) is executed with the optional P and Q values as its two arguments.

Execution of the G-code file pauses until the external program exits. If the external program exits with exit code other than *0* G-code 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.

After creating a new M1*nn* program, the GUI should be restarted so that the new program is taken into account, otherwise a *Unknown M-code* error will occur.

**Warning**
Do not use a word processor to create or edit the files. A word processor will leave unseen codes that will cause problems and may prevent a bash or python file from working. Use a text editor like Geany in Debian or Notepad++ in other operating systems to create or edit the files.

The error *Unknown M-code used* denotes one of the following:

- Имя сигнала не существует.
- The file is not an executable file.
- Имя файла имеет расширение.
- The file name does not follow this format Mnnn where nnn = 100 through 199.
- The file name used a lower case M.

For example to open and close a collet closer that is controlled by a parallel port pin using a bash script file using M101 and M102. Create two files named M101 and M102. Set them as executable files (typically right click/properties/permissions) before running LinuxCNC. Make sure the parallel port pin is not connected to anything in a HAL file.

**M101 Example File**

```
#!/bin/bash
# file to turn on parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out True
exit 0
```

**M102 Example File**

```
#!/bin/bash
# file to turn off parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out False
exit 0
```

To pass a variable to a M1nn file you use the P and Q option like this:

```
M100 P123.456 Q321.654
```

**M100 Example file**

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

To display a graphic message and stop until the message window is closed use a graphic display program like Eye of Gnome to display the graphic file. When you close it the program will resume.

**M110 Example file**

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

To display a graphic message and continue processing the G-code file suffix an ampersand to the command.

**M110 Example display and keep going**

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

# 11.7 O Коды

## 11.7.1 Use of O-codes

O-codes provide for flow control in NC programs. Each block has an associated number, which is the number used after O. Care must be taken to properly match the O-numbers. O-codes use the letter *O* not the number zero as the first character in the number like O100 or o100. O-codes are also sometimes called o-words and these terms are interchangeable.

---

**Note**
Using the lower case *o* makes it easier to distinguish from a 0 that might have been mistyped. For example *o100* is easier to see than *O100* that it is not a *0*.

---

## 11.7.2 Numbering

There are two categories of o-codes with different scoping rules:

**Subroutine definitions** (`sub`/`endsub`, `call`) are **global**. Each subroutine must have a unique number or name across the entire program and all called files.

**Control flow** (`if`/`endif`, `while`/`endwhile`, `do`/`while`, `repeat`/`endrepeat`, `break`, `continue`) are **local** to the subroutine or main program where they appear. The interpreter automatically scopes them, so `o100 if` inside `o<helper>` does not conflict with `o100 if` in the main program or in any other subroutine. You can safely reuse the same o-numbers for control flow in different subroutines.

Within a single subroutine body (or the main program), each o-number should still be used for only one control flow block.

**Numbering Example**

```
(the start of o100)
o100 sub
(notice that the if-endif block uses a different number within this sub)
  (the start of o110)
  o110 if [#2 GT 5]
    (some code here)
  (the end of o110)
  o110 endif
  (some more code here)
(the end of o100)
o100 endsub
```

**Reusing Control Flow Numbers Across Subroutines (valid)**

```
(o100 if in helper.ngc does not conflict with o100 if in another.ngc)

(file: helper.ngc)
o<helper> sub
  o100 if [#1 GT 5]
    (do something)
  o100 endif
o<helper> endsub

(file: another.ngc)
o<another> sub
  o100 if [#1 LT 0]
    (do something else)
  o100 endif
o<another> endsub
```

### 11.7.3 Comments

Comments on the same line as the o-word should not be used as the behavior can change in the future.

The behavior is undefined if:

- The same number is used for more than one block within the same scope (see [?] for scoping rules).
- Other words are used on a line with an o-word.
- Comments are used on a line with an o-word.

### 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.

**Subroutine Example**

```
o100 sub
  G53 G0 X0 Y0 Z0 (rapid move to machine home)
o100 endsub

(the subroutine is called)
o100 call
M2
```

See G53, G0 and M2 sections for more information.

**O- Return** Inside a subroutine, *o- return* can be executed. This immediately returns to the calling code, just as though *o- endsub* was encountered.

**O- Return Example**

```
o100 sub
  (test if parameter #2 is greater than 5)
  o110 if [#2 GT 5]
    (return to top of subroutine if test is true)
    o100 return
  o110 endif
    (this only gets executed if parameter #2 is not greater than 5)
    (DEBUG, parameter 1 is [#1])
o100 endsub
```

See the Binary Operators and Parameters sections for more information.

**O- Call** *o- Call* takes up to 30 optional arguments, which are passed to the subroutine as *#1*, *#2* , ..., #N. Parameters from #N+1 to #30 have the same value as in the calling context. On return from the subroutine, the values of parameters #1 through #30 (regardless of the number of arguments) will be restored to the values they had before the call. Parameters #1 - #30 are local to the subroutine.

Because *1 2 3* is parsed as the number 123, the parameters must be enclosed in square brackets. The following calls a subroutine with 3 arguments:

**O- Call Example**

```
o100 sub
  (test if parameter #2 is greater than 5)
  o110 if [#2 GT 5]
    (return to top of subroutine if test is true)
    o100 return
```

```
  o110 endif
    (this only gets executed if parameter #2 is not greater than 5)
    (DEBUG, parameter 1 is [#1])
    (DEBUG, parameter 3 is [#3])
o100 endsub

o100 call [100] [2] [325]
```

Subroutine **definitions** may not be nested. Defining a subroutine inside another subroutine is not allowed and will produce an error. For example, the following is **invalid**:

**Invalid Nested Subroutine Definition (produces an error)**

```
o<outer> sub
  o100 sub       (INVALID: nested subroutine definition)
    (code here)
  o100 endsub
o<outer> endsub
```

However, **calling** a subroutine from within another subroutine is perfectly valid. It is the `sub`/`endsub` definition that cannot be nested, not the `call`:

**Valid Nested Subroutine Call**

```
o<outer> sub
  (code here)
  o<helper> call [1] [2]  (OK: calling another sub)
o<outer> endsub
```

Other O-code control flow such as `if`/`endif`, `while`/`endwhile`, `do`/`while`, and `repeat`/`endrepeat` may be used freely inside subroutines.

Subroutines may only be called after they are defined. They may be called from other functions, and may call themselves recursively if it makes sense to do so. The maximum subroutine nesting level is 10.

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.

Numbered programs are enabled by default, and may be disabled by placing `DISABLE_FANUC_STYLE_SUB = 1` in the `[RS274NGC]` section of the `.ini` file.

---

**Note**
Numbered main and subprogram definitions and calls differ from traditional rs274ngc both in syntax and execution. To reduce the possibility of confusion, the interpreter will raise an error if definitions of one style are mixed with calls of another.

---

**Numbered Subprogram Simple Example**

```
o1 (Example 1)    ; Main program 1, "Example 1"
M98 P100          ; Call subprogram 100
M30               ; End main program
```

```
o100              ; Beginning of subprogram 100
  G53 G0 X0 Y0 Z0 ; Rapid move to machine home
M99               ; Return from subprogram 100
```

**o1 (Title)** The optional main program beginning block gives the main program the number `1`. Some controllers treat an optional following parenthesized comment as a program title, `Example 1` in this example, but this has no special meaning in the rs274ngc interpreter.

**M98 P- <L->** Call a numbered subprogram. The block `M98 P100` is analogous to the traditional `o100 call` syntax, but may only be used to call a following numbered subprogram defined with `o100`...`M99`. An optional *L*-word specifies a loop count.

**M30** The main program must be terminated with `M02` or `M30` (or `M99`; see below).

**O- subprogram definition start** Marks the start of a numbered subprogram definition. The block `o100` is similar to `o100 sub`, except that it must be placed later in the file than the `M98 P100` calling block.

**M99 return from numbered subroutine** The block `M99` is analogous to the traditional `o100 endsub` syntax, but may only terminate a numbered program (`o100` in this example), and may not terminate a subroutine beginning with the `o100 sub` syntax.

The `M98` subprogram call differs from rs274ngc `o call` in the following ways:

- The numbered subprogram must follow the `M98` call in the program file. The interpreter will throw an error if the subprogram precedes the call block.
- Parameters `#1`, `#2`, ..., `#30` are global and accessible in numbered subprograms, similar to higher-numbered parameters in traditional style calls. Modifications to these parameters within a subprogram are global modifications, and will be persist after subprogram return.
- `M98` subprogram calls have no return value.
- `M98` subprogram call blocks may contain an optional L-word specifying a loop repeat count. Without the L-word, the subprogram will execute once only (equivalent to `M98 L1`). An `M98 L0` block will not execute the subprogram.

In rare cases, the `M99` M-code may be used to terminate the main program, where it indicates an *endless program*. When the interpreter reaches an `M99` in the main program, it will skip back to the beginning of the file and resume execution at the first line. An example use of an endless program is in a machine warm-up cycle; a block delete program end `/M30` block might be used to stop the cycle at a tidy point when the operator is ready.

**Numbered Subprogram Full Example**

```
o1                            ; Main program 1
  #1 = 0
  (PRINT,X MAIN BEGIN:  1=#1)
  M98 P100 L5                 ; Call subprogram 100
  (PRINT,X MAIN END:  1=#1)
M30                           ; End main program

o100                          ; Subprogram 100
  #1 = [#1 + 1]
  M98 P200 L5                 ; Call subprogram 200
  (PRINT,>> o100:  #1)
M99                           ; Return from Subprogram 100

o200                          ; Subprogram 200
  #1 = [#1 + 0.01]
  (PRINT,>>>> o200:  #1)
M99                           ; Return from Subprogram 200
```

In this example, parameter #1 is initialized to 0. Subprogram o100 is called five times in a loop. Nested within each call to o100, subprogram o200 is called five times in a loop, for 25 times total.

Note that parameter #1 is global. At the end of the main program, after updates within o100 and o200, its value will equal 5.25.

### 11.7.5 Цикл

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 Example**

```
(draw a sawtooth shape)
G0 X1 Y0 (move to start position)
#1 = 0 (assign parameter #1 the value of 0)
F25 (set a feed rate)
o101 while [#1 LT 10]
  G1 X0
  G1 Y[#1/10] X1
  #1 = [#1+1] (increment the test counter)
o101 endwhile
M2 (end program)
```

**Do While Example**

```
#1 = 0 (assign parameter #1 the value of 0)
o100 do
  (debug, parameter 1 = #1)
  o110 if [#1 EQ 2]
    #1 = 3 (assign the value of 3 to parameter #1)
    (msg, #1 has been assigned the value of 3)
    o100 continue (skip to start of loop)
  o110 endif
  (some code here)
  #1 = [#1 + 1] (increment the test counter)
o100 while [#1 LT 3]
(msg, Loop Done!)
M2
```

Inside a while loop, *o- break* immediately exits the loop, and *o- continue* immediately skips to the next evaluation of the *while* condition. If it is still true, the loop begins again at the top. If it is false, it exits the loop.

### 11.7.6 Условный

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*.

If the *if* conditional evaluates to true then the group of statements following the *if* up to the next conditional line are executed.

If the *if* conditional evaluates to false then the *elseif* conditions are evaluated in order until one evaluates to true. If the *elseif* condition is true then the statements following the *elseif* up to the next conditional line are executed. If none of the *if* or *elseif* conditions evaluate to true then the statements following the *else* are executed. When a condition is evaluated to true no more conditions are evaluated in the group.

**If Endif Example**

```
(if parameter #31 is equal to 3 set S2000)
o101 if [#31 EQ 3]
  S2000
o101 endif
```

**If ElseIf Else EndIf Example**

```
(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
```

Several conditions may be tested for by *elseif* statements until the *else* path is finally executed if all preceding conditions are false:

**If Elseif Else Endif Example**

```
(if parameter #2 is greater than 5 set F100)
o102 if [#2 GT 5]
  F100
(else if parameter #2 less than 2 set F200)
o102 elseif [#2 LT 2]
  F20
(parameter #2 is between 2 and 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.

**Example with repeat**

```
(Mill 5 diagonal shapes)
G91 (Incremental mode)
o103 repeat [5]
... (insert milling code here)
G0 X1 Y1 (diagonal move to next position)
o103 endrepeat
G90 (Absolute mode)
```

### 11.7.8 Indirection

The o-number may be given by a parameter and/or calculation.

**Indirection Example**

```
o[#101+2] call
```

**Computing values in O-words** For more information on computing values see the following sections:

- Parameters
- Expressions
- Binary Operators
- Functions

### 11.7.9 Вызов файлов

To call a separate file with a subroutine name the file the same as your call and include a sub and endsub in the file. The file must be in the directory pointed to by *PROGRAM_PREFIX* or *SUBROUTINE_PATH* in the INI file. The file name can include **lowercase** letters, numbers, dash, and underscore only. A named subroutine file can contain only a single subroutine definition. Do not place additional numbered subroutine definitions (e.g. `o100 sub`) in the same file as a named subroutine. Numbered subroutines share a global namespace and can silently conflict across files. If you need helper subroutines, place each one in its own file.

**Named File Example**

```
o<myfile> call
```

**Numbered File Example**

```
o123 call
```

In the called file you must include the oxxx sub and endsub and the file must be a valid file.

**Called File Example**

```
(filename myfile.ngc)
o<myfile> sub
  (code here)
o<myfile> endsub
M2
```

---

**Note**
The file names are lowercase letters only so *o<MyFile>* is converted to *o<myfile>* by the interpreter. More information about the search path and options for the search path are in the INI configuration section.

---

### 11.7.10 Subroutine return values

Subroutines may optionally return a value by an optional expression at an *endsub* or *return* statement.

**Return value example**

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

A subroutine return value is stored in the *<_value>* predefined named parameter , and the *<_value_returned>* predefined parameter is set to 1, to indicate a value was returned. Both parameters are global, and are cleared just before the next subroutine call.

### 11.7.11 Ошибки

The following statements cause an error message and abort the interpreter:

- a `return` or `endsub` not within a sub definition
- a label on `repeat` which is defined elsewhere
- a label on `while` which is defińed elsewhere and not referring to a `do`
- a label on `if` defined elsewhere
- a undefined label on `else` or `elseif`
- a label on `else`, `elseif` or `endif` not pointing to a matching `if`
- a label on `break` or `continue` which does not point to a matching `while` or `do`
- a label on `endrepeat` or `endwhile` no referring to a corresponding `while` or `repeat`
- a subroutine definition (`sub`) inside another subroutine body (nested definitions)
- a numbered subroutine called from within a named subroutine file but not found in the offset table or as a separate file

To make these errors non-fatal warnings on stderr, set bit 0x20 in the `[RS274NGC]FEATURE=` mask ini option.

## 11.8 Другие коды

### 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.

The application of the feed rate is as described in the Feed Rate Section, unless *inverse time feed rate mode* or *feed per revolution mode* are in effect, in which case the feed rate is as described in the G93 G94 G95 section.

### 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.

The spindle(s) or selected spindle will turn at that speed when a *M3* or *M4* is in effect. It is OK to program an S word whether the spindle is turning or not. If the speed override switch is enabled and not set at 100%, the speed will be different from what is programmed.

It is OK to program S0, the spindle will not turn if that is done.

It is an error if:

- the S number is negative.

As described in the section Right-hand Tapping Cycle with Dwell, if a *G84* (tapping) drilling cycle is active and the speed and feed potentiometers are enabled, the one with the lowest setting will be used. The rotational speed and feed rate will remain synchronized. In this case, the speed may differ from the one programmed, even if the speed correction potentiometer is set to 100%.

### 11.8.3 T: Выбрать инструмент

*Tx* - prepare to change to tool *x*.

The tool is not changed until an *M6* is programmed (see Section M6). The T word may appear on the same line as the *M6* or on a previous line. It is OK if T words appear on two or more lines with no tool change. Only the the most recent T word will take effect at the next tool change.

**Note**
When LinuxCNC is configured for a nonrandom toolchanger (see the entry for RANDOM_TOOLCHANGER in the EMCIO Section), *T0* gets special handling: no tool will be selected. This is useful if you want the spindle to be empty after a tool change.

**Note**
When LinuxCNC is configured for a random toolchanger (see the entry for RANDOM_TOOLCHANGER in the EMCIO Section), *T0* does not get any special treatment: T0 is a valid tool like any other. It is customary to use T0 on a random toolchanger machine to track an empty pocket, so that it behaves like a nonrandom toolchanger machine and unloads the spindle.

It is an error if:

- a negative T number is used,
- T number is used that does not appear in the tool table file (with the exception that T0 on nonrandom toolchangers **is** accepted, as noted above).

On some machines, the carousel will move when a T word is programmed, at the same time machining is occurring. On such machines, programming the T word several lines before a tool change will save time. A common programming practice for such machines is to put the T word for the next tool to be used on the line after a tool change. This maximizes the time available for the carousel to move.

Rapid moves after a *T<n>* will not show on the AXIS preview until after a feed move. This is for machines that travel long distances to change the tool like a lathe. This can be very confusing at first. To turn this feature off for the current tool program a G1 without any move after the *T<n>*.

## 11.9 G-Code Examples

After you install LinuxCNC several sample files are placed in the /nc_files folder. Make sure the sample file is appropriate for your machine before running.

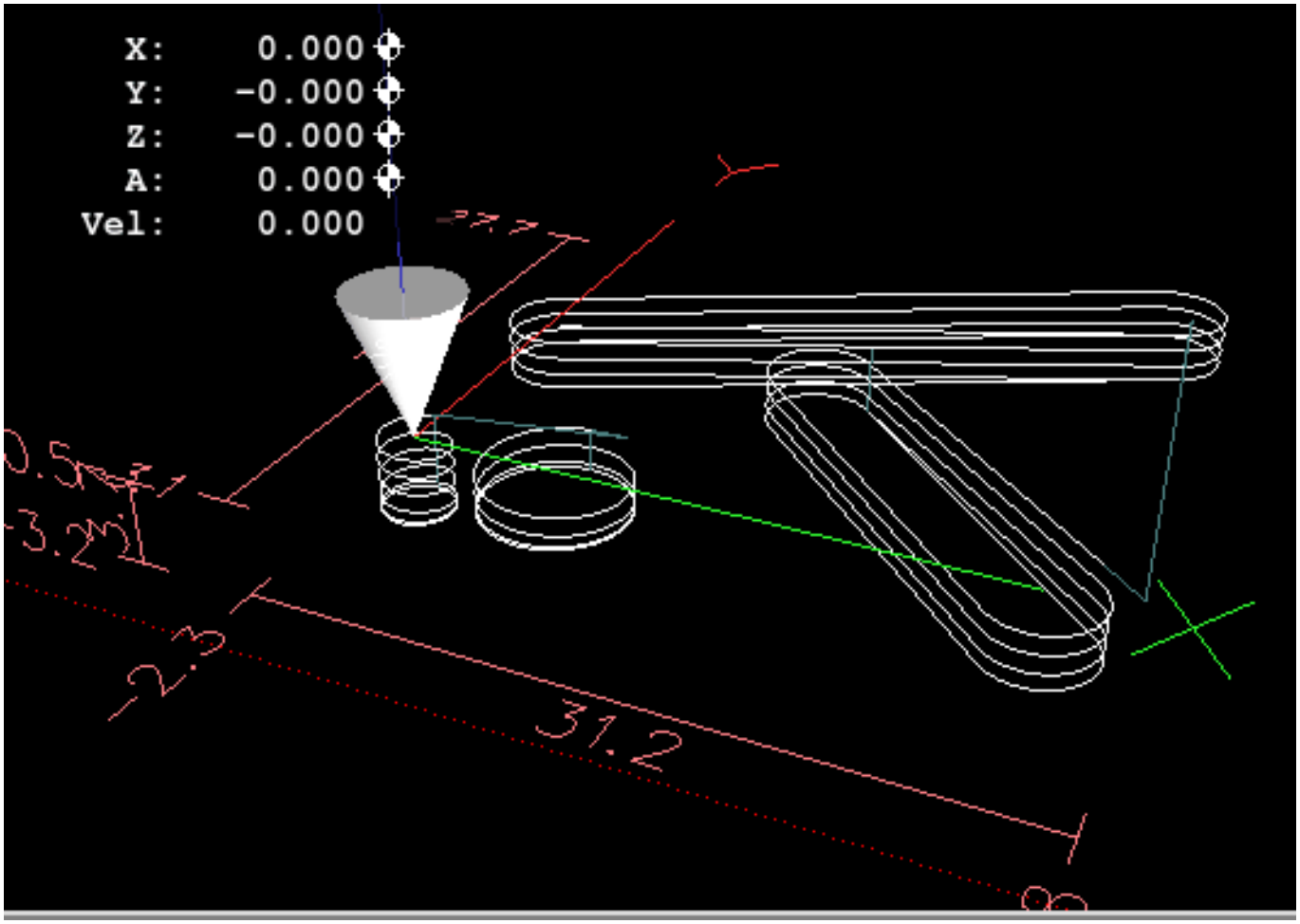
### 11.9.1 Mill Examples

#### 11.9.1.1 Helical Hole Milling

- File Name: useful-subroutines.ngc
- Description: Subroutine for milling a hole using parameters.

#### 11.9.1.2 Slotting

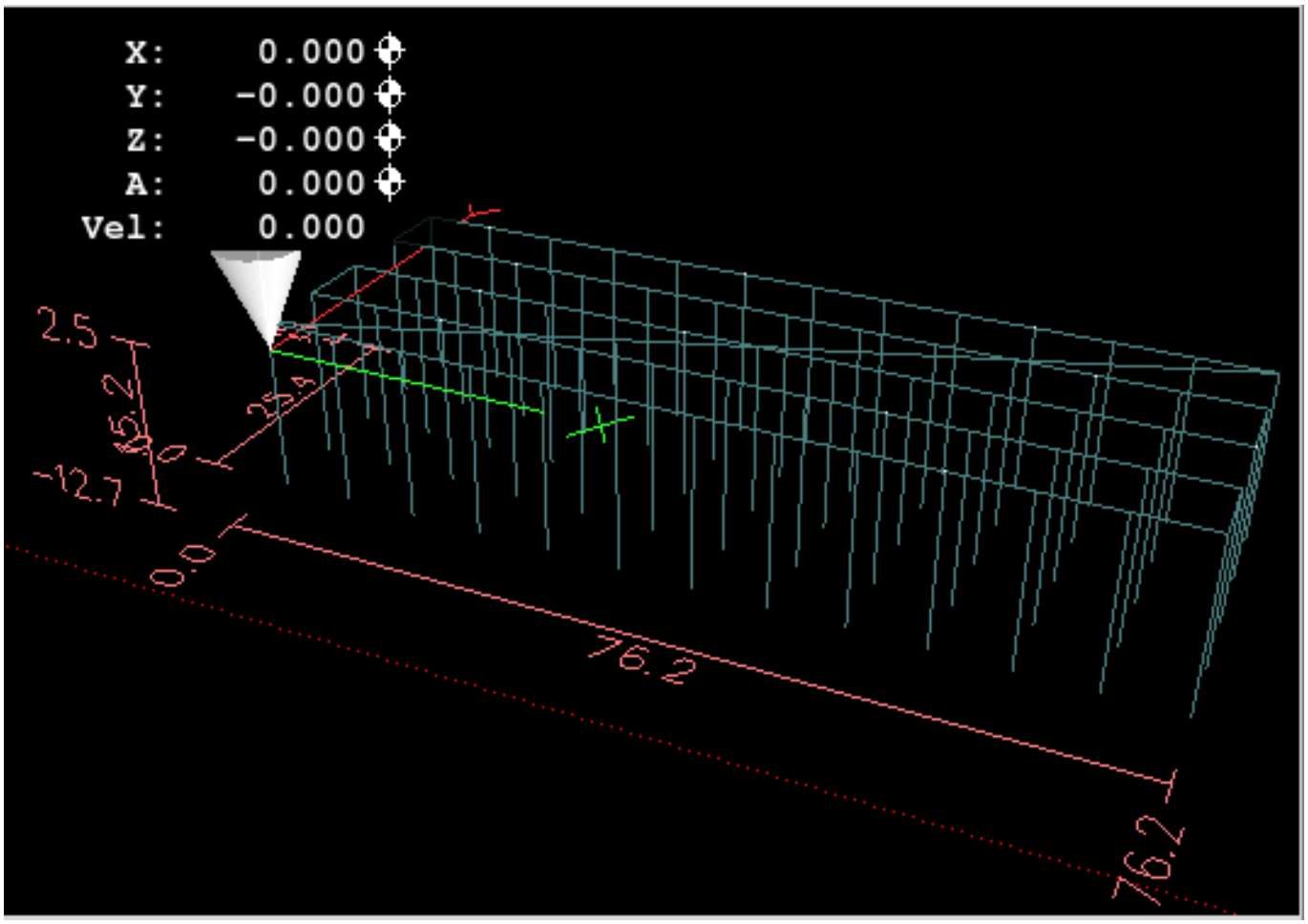
- File Name: useful-subroutines.ngc
- Description: Subroutine for milling a slot using parameters.




#### 11.9.1.3 Grid Probe

- File Name: gridprobe.ngc
- Description: Rectangular Probing

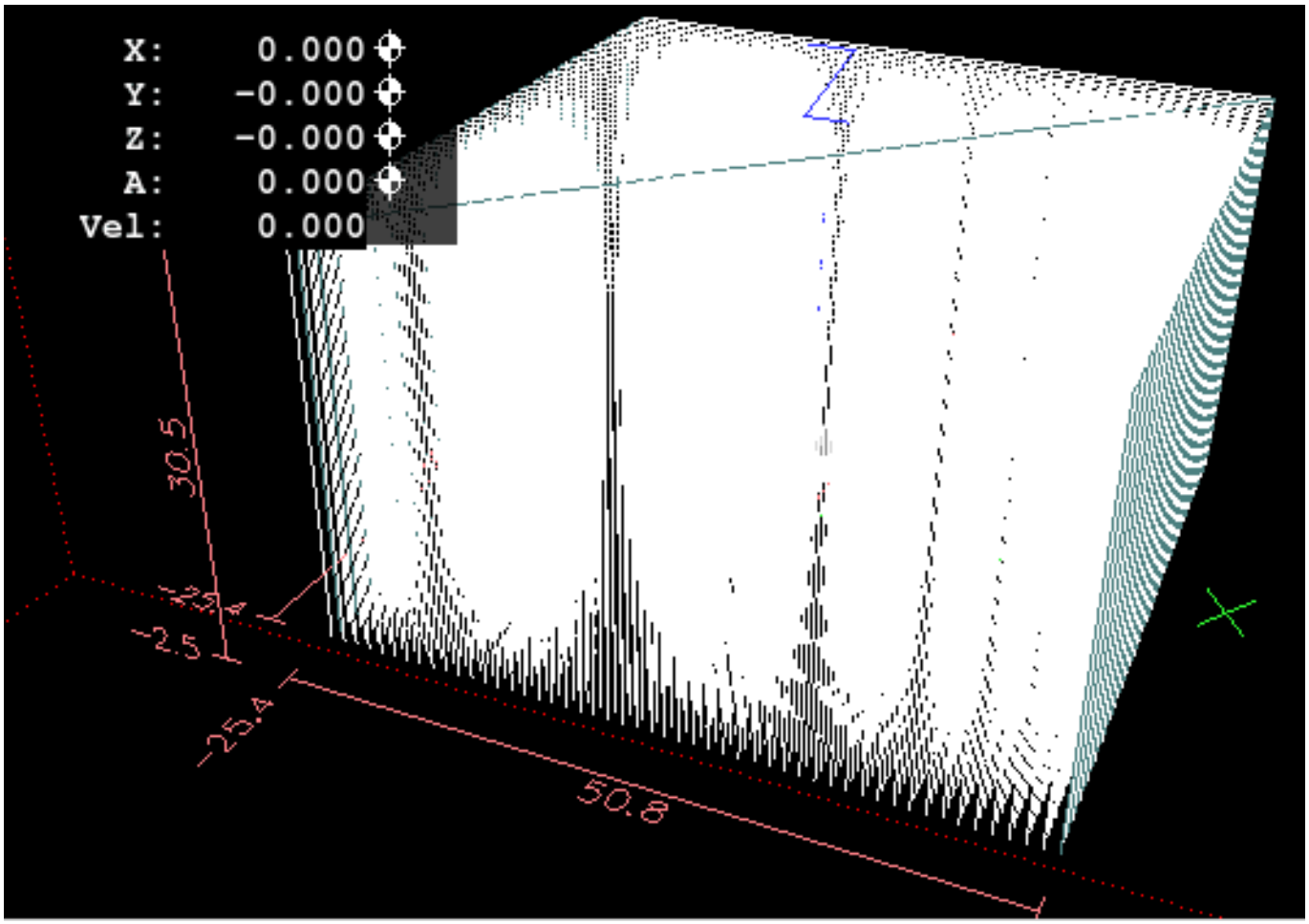
This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file.




#### 11.9.1.4 Smart Probe

- File Name: smartprobe.ngc
- Description: Rectangular Probing

This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file. This is improved from the grid probe file.




#### 11.9.1.5 Tool Length Probe

- File Name: tool-length-probe.ngc
- Description: Tool Length Probing

This program shows an example of how to measure tool lengths automatically using a switch hooked to the probe input. This is useful for machines without tool holders, where the length of a tool is different every time it is inserted.

#### 11.9.1.6 Hole Probe

- File Name: probe-hole.ngc
- Description: Finding the Center and Diameter of a hole.

The program demonstrates how to find the center of a hole, measure the hole diameter and record the results.

#### 11.9.1.7 Cutter Compensation

- File Name: comp-g1.ngc

- Description: Entry and exit movements with compensation of tool radius.

This program demonstrates the peculiarity of the toolpath without and with tool radius compensation. The tool radius is taken from the tool table.

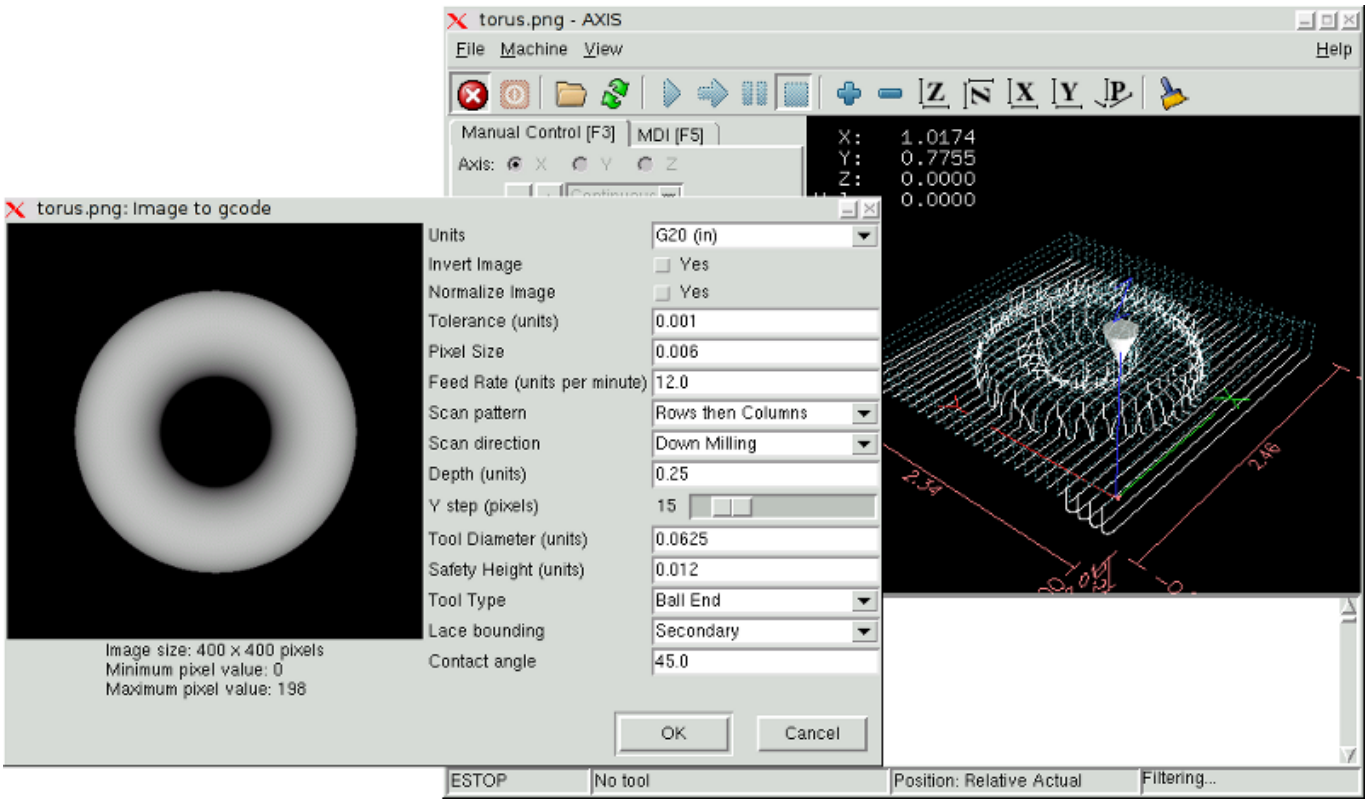
### 11.9.2 Lathe Examples

#### 11.9.2.1 Threading

- File Name lathe-g76.ngc
- Description: Facing, threading and parting off.

This file shows an example of threading on a lathe using parameters.

## 11.10 Image to G-Code




### 11.10.1 What is a depth map?

A depth map is a greyscale image where the brightness of each pixel corresponds to the depth (or height) of the object at each point.

### 11.10.2 Integrating image-to-gcode with the AXIS user interface

Add the following lines to the *[FILTER]* section of your INI file to make AXIS automatically invoke image-to-gcode when you open a PNG, GIF, or JPEG image:

```
PROGRAM_EXTENSION = .png,.gif,.jpg Grayscale Depth Image
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
```

The standard *sim/axis.ini* configuration file is already prepared this way.

### 11.10.3 Using `image-to-gcode`

Start image-to-gcode either by opening an image file in AXIS, or by invoking image-to-gcode from the terminal, as follows:

```
image-to-gcode torus.png > torus.ngc
```

Verify all the settings in the right-hand column, then press OK to create the G-code. Depending on the image size and options chosen, this may take from a few seconds to a few minutes. If you are loading the image in AXIS, the G-code will automatically be loaded and previewed once image-to-gcode completes. In AXIS, hitting reload will show the image-to-gcode option screen again, allowing you to tweak them.

### 11.10.4 Option Reference

#### 11.10.4.1 Units

Specifies whether to use G20 (inches) or G21 (mm) in the generated G-code and as the units for each option labeled *(units)*.

#### 11.10.4.2 Invert Image

If ”no”, the black pixel is the lowest point and the white pixel is the highest point. If ”yes”, the black pixel is the highest point and the white pixel is the lowest point.

#### 11.10.4.3 Normalize Image

If *yes*, the darkest pixel is remapped to black, the lightest pixel is remapped to white.

#### 11.10.4.4 Expand Image Border

If *None*, the input image is used as-is, and details which are at the very edges of the image may be cut off. If *White* or *Black*, then a border of pixels equal to the tool diameter is added on all sides, and details which are at the very edges of the images will not be cut off.

#### 11.10.4.5 Tolerance (units)

When a series of points are within *tolerance* of being a straight line, they are output as a straight line. Increasing tolerance can lead to better contouring performance in LinuxCNC, but can also remove or blur small details in the image.

#### 11.10.4.6 Pixel Size (units)

One pixel in the input image will be this many units—usually this number is much smaller than 1.0. For instance, to mill a 2.5x2.5-inch object from a 400x400 image file, use a pixel size of .00625, because 2.5 / 400 = .00625.

#### 11.10.4.7 Plunge Feed Rate (units per minute)

The feed rate for the initial plunge movement.

#### 11.10.4.8 Feed Rate (units per minute)

The feed rate for other parts of the path.

#### 11.10.4.9 Spindle Speed (RPM)

The spindle speed S-code that should be put into the G-code file.

#### 11.10.4.10 Scan Pattern

Possible scan patterns are:

- Rows
- Columns
- Rows, then Columns
- Columns, then Rows

#### 11.10.4.11 Scan Direction

Possible scan directions are:

- Positive: Start milling at a low X or Y axis value, and move towards a high X or Y axis value.
- Negative: Start milling at a high X or Y axis value, and move towards a low X or Y axis value.
- Alternating: Start on the same end of the X or Y axis travel that the last move ended on. This reduces the amount of traverse movements.
- Up Milling: Start milling at low points, moving towards high points.
- Down Milling: Start milling at high points, moving towards low points.

#### 11.10.4.12 Depth (units)

The top of material is always at $Z=0$. The deepest cut into the material is at $Z=-depth$.

#### 11.10.4.13 Step Over (pixels)

The distance between adjacent rows or columns. To find the number of pixels for a given units distance, compute *distance/pixel size* and round to the nearest whole number. For example, if *pixel size=.006* and the desired step over *distance=.015*, then use a Step Over of 2 or 3 pixels, because *.015/.006=2.5*.

#### 11.10.4.14 Tool Diameter

The diameter of the cutting part of the tool.

#### 11.10.4.15 Safety Height

The height to move to for traverse movements. image-to-gcode always assumes the top of material is at *Z=0*.

#### 11.10.4.16 Tool Type

The shape of the cutting part of the tool. Possible tool shapes are:

- Ball End
- Flat End
- 45 degree ”vee”
- 60 degree ”vee”

#### 11.10.4.17 Lace bounding

This controls whether areas that are relatively flat along a row or column are skipped. This option only makes sense when both rows and columns are being milled. Possible bounding options are:

- None: Rows and columns are both fully milled.
- Secondary: When milling in the second direction, areas that do not strongly slope in that direction are skipped.
- Full: When milling in the first direction, areas that strongly slope in the second direction are skipped. When milling in the second direction, areas that do not strongly slope in that direction are skipped.

#### 11.10.4.18 Contact angle

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 Roughing offset and depth per pass

Image-to-gcode can optionally perform roughing passes. The depth of successive roughing passes is given by *Roughing depth per pass*. For instance, entering 0.2 will perform the first roughing pass with a depth of 0.2, the second roughing pass with a depth of 0.4, and so on until the full Depth of the image is reached. No part of any roughing pass will cut closer than Roughing Offset to the final part. The following figure shows a tall vertical feature being milled. In this image, Roughing depth per pass is 0.2 inches and roughing offset is 0.1 inches.

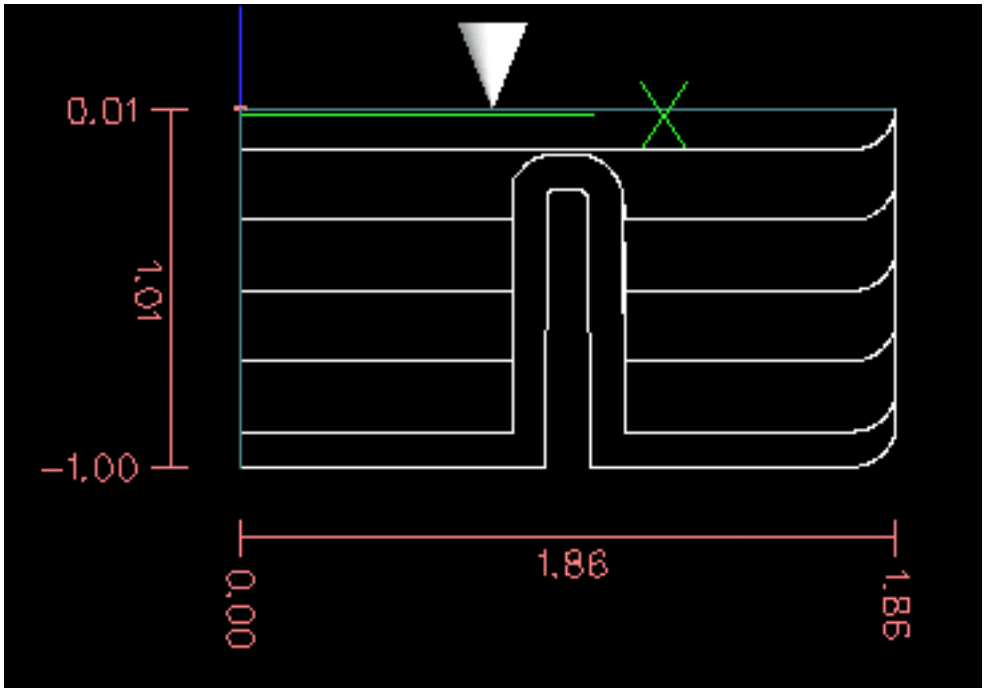



Figure 11.21: Roughing passes and final pass

## 11.11 RS274/NGC Differences

### 11.11.1 Changes from RS274/NGC

Differences that change the meaning of RS274/NGC programs

**Location after a tool change**
In LinuxCNC, the machine does not return to its original position after a tool change. This change was made because the new tool might be longer than the old tool, and the move to the original machine position could therefore leave the tool tip too low.

**Offset parameters are INI file units**
In LinuxCNC, the values stored in parameters for the G28 and G30 home locations, the P1...P9 coordinate systems, and the G92 offset are in "INI file units". This change was made because otherwise the meaning of a location changed depending on whether G20 or G21 was active when G28, G30, G10 L2, or G92.3 is programmed.

**Tool table lengths/diameters are in INI file units**
In LinuxCNC, the tool lengths (offsets) and diameters in the tool table are specified in INI file units only. This change was made because otherwise the length of a tool and its diameter would

change based on whether G20 or G21 was active when initiating G43, G41, G42 modes. This made it impossible to run G-code in the machine's non-native units, even when the G-code was simple and well-formed (starting with G20 or G21, and didn't change units throughout the program), without changing the tool table.

**G84, G87 not implemented**
G84 and G87 are not currently implemented, but may be added to a future release of LinuxCNC.

**G28, G30 with axis words**
When G28 or G30 is programmed with only some axis words present, LinuxCNC only moves the named axes. This is common on other machine controls. To move some axes to an intermediate point and then move all axes to the predefined point, write two lines of G code:

```
G0 X- Y- (axes to move to intermediate point)
G28 (move all axes to predefined point)
```

### 11.11.2 Additions to RS274/NGC

Differences that do not change the meaning of RS274/NGC programs

**G33, G76 threading codes**
These codes are not defined in RS274/NGC.

**G38.2**
The probe tip is not retracted after a G38.2 movement. This retraction move may be added in a future release of LinuxCNC.

**G38.3...G38.5**
These codes are not defined in RS274/NGC

**O-codes**
These codes are not defined in RS274/NGC

**M50...M53 overrides**
These codes are not defined in RS274/NGC

**M61..M66**
These codes are not defined in RS274/NGC

**G43, G43.1**
*Negative Tool Lengths*

The RS274/NGC spec says "it is expected that" all tool lengths will be positive. However, G43 works for negative tool lengths.

*Lathe tools*

G43 tool length compensation can offset the tool in both the X and Z dimensions. This feature is primarily useful on lathes.

*Dynamic tool lengths*

LinuxCNC allows specification of a computed tool length through G43.1 I K.

**G41.1, G42.1**
LinuxCNC allows specification of a tool diameter and, if in lathe mode, orientation in the G-code. The format is G41.1/G42.1 D L, where D is diameter and L (if specified) is the lathe tool orientation.

**G43 without H word**

In NGC this is not allowed. In LinuxCNC, it sets length offsets for the currently loaded tool. If no tool is currently loaded, it is an error. This change was made so the user doesn't have to specify the tool number in two places for each tool change, and because it's consistent with the way G41/G42 work when the D word is not specified.

**U, V, and W axes**

LinuxCNC allows machines with up to 9 axes by defining an additional set of 3 linear axes known as U, V and W

# Chapter 12

# Виртуальные панели управления

## 12.1 PyVCP

### 12.1.1 Введение

PyVCP, **Py**thon **V**irtual **C**ontrol **P**anel, разработан для того, чтобы дать интегратору возможность настраивать интерфейс AXIS с помощью кнопок и индикаторов для выполнения специальных задач.

Панели управления оборудованием могут использовать много контактов ввода-вывода и могут быть дорогими. Именно здесь Virtual Control Panels (виртуальные панели управления) имеют преимущество, а создание PyVCP ничего не стоит.

Виртуальные панели управления можно использовать для тестирования или мониторинга, чтобы временно заменить реальные устройства ввода-вывода при отладке релейной логики или для моделирования физической панели перед ее сборкой и подключением к плате ввода-вывода.

На следующем рисунке показаны многие виджеты PyVCP.

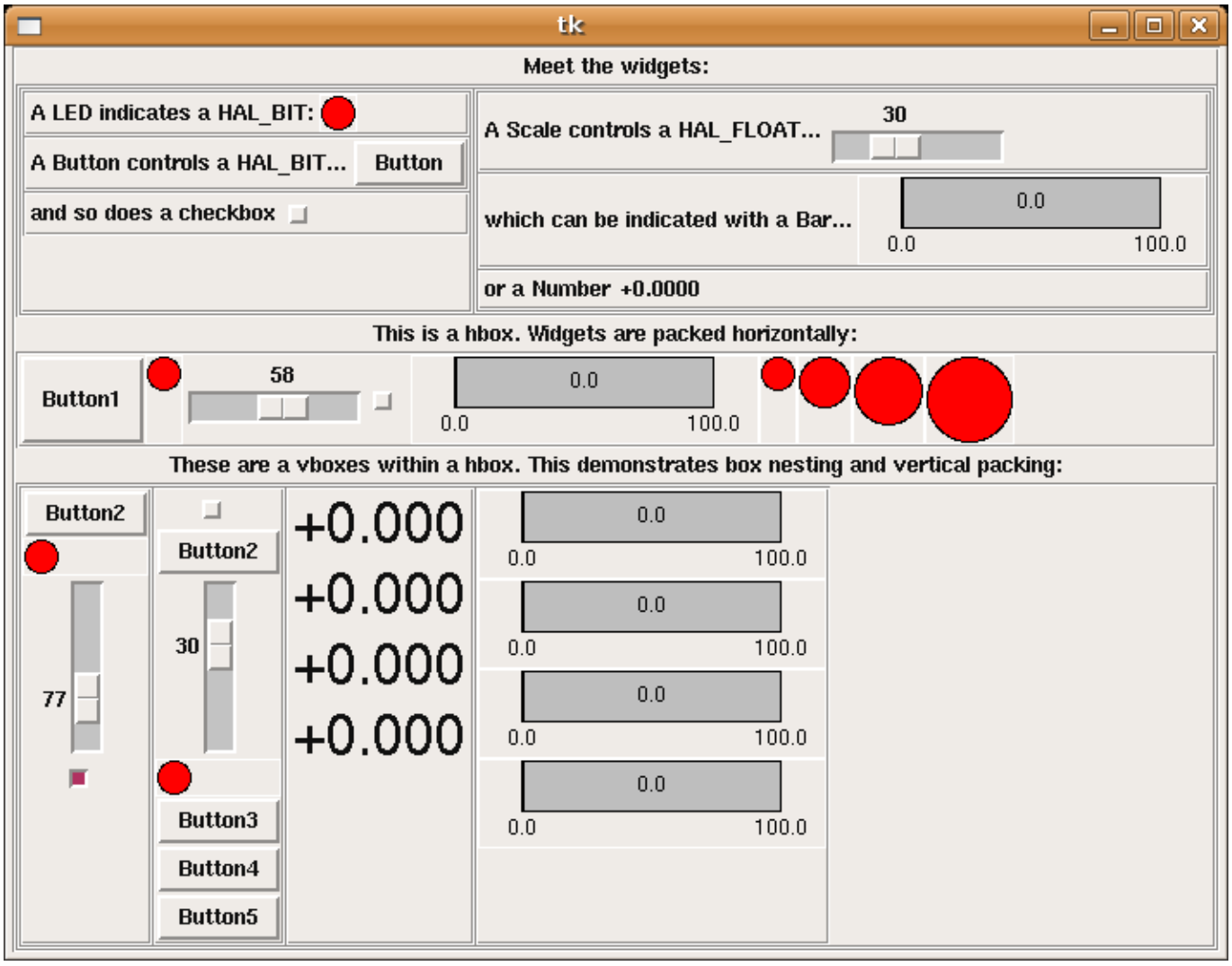



Figure 12.1: PyVCP Widgets Showcase

### 12.1.2 Устройство панели

Макет панели PyVCP определяется XML-файлом, содержащим теги виджетов между <pyvcp> и </pyvcp>. Например:

```
<pyvcp>
    <label text="This is a LED indicator"/>
    <led/>
</pyvcp>
```

Figure 12.2: Простой пример светодиодной панели PyVCP

Если вы поместите этот текст в файл tiny.xml и запустите

```
halcmd loadusr pyvcp -c mypanel tiny.xml
```

PyVCP создаст для вас панель, которая включает в себя два виджета: метку с текстом «Это светодиодный индикатор» и светодиод, используемый для отображения состояния сигнала HAL BIT. Он также создаст компонент HAL с именем *mypanel* (все виджеты на этой панели подключены к контактам, начинающимся с *mypanel*.). Поскольку внутри тега <led> не было тега <halpin>, PyVCP автоматически назовет контакт HAL для светодиодного виджета mypanel.led.0

Список виджетов, их тегов и параметров см. в разделе widgets reference ниже.

После создания панели подключение сигналов HAL к контактам PyVCP и обратно выполняется с помощью halcmd:

```
net <signal-name> <pin-name> <opt-direction> <opt-pin-name>signal-name
```

Если вы новичок в HAL, хорошим началом будет глава «Основы HAL» в Руководстве интегратора.

### 12.1.3 Безопасность

Части файлов PyVCP оцениваются как код Python и могут выполнять любые действия, доступные программам Python. Используйте XML-файлы PyVCP только из источника, которому вы доверяете.

### 12.1.4 AXIS

Поскольку AXIS использует тот же набор инструментов ГИП (Tkinter), что и PyVCP, можно включить панель PyVCP либо в правую, либо в нижнюю часть пользовательского интерфейса AXIS. Невозможно отобразить панель в обоих этих положениях одновременно. Типичный пример описан ниже.

В дополнение к или вместо отображения панели PyVCP, как описано выше, можно отображать одну или несколько панелей PyVCP в виде встроенных вкладок в графическом интерфейсе AXIS. Это достигается с помощью следующего в разделе `[DISPLAY]` INI-файла:

```
EMBED_TAB_NAME    = Spindle
EMBED_TAB_COMMAND = pyvcp spindle.xml
```

В текстовой метке вкладки AXIS будет отображаться слово «Spindle».

#### 12.1.4.1 Панель для примера

Поместите XML-файл PyVCP, описывающий панель, в тот же каталог, где находится ваш INI-файл. Допустим, мы хотим отобразить текущую скорость шпинделя с помощью виджета Bar. Поместите следующее в файл spindle.xml:

```
<pyvcp>
    <label>
        <text>”Spindle speed:”</text>
    </label>
    <bar>
        <halpin>”spindle-speed”</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Здесь мы создали панель с виджетом Label и Bar, указали, что контакт HAL, подключенный к панели, должен называться *spindle-speed*, и установили максимальное значение панели равным 5000 (см. widget reference ниже для всех вариантов). Чтобы AXIS знала об этом файле и вызывала его при запуске, нам нужно указать следующее в разделе `[DISPLAY]` INI-файла:

```
PYVCP = spindle.xml
```

Если панель должна появиться в нижней части пользовательского интерфейса AXIS, нам необходимо указать следующее в разделе [DISPLAY] INI-файла:

```
PYVCP_POSITION = BOTTOM
```

Любое значение, кроме BOTTOM, или отсутствие этой переменной приведет к размещению панели PYVCP справа.

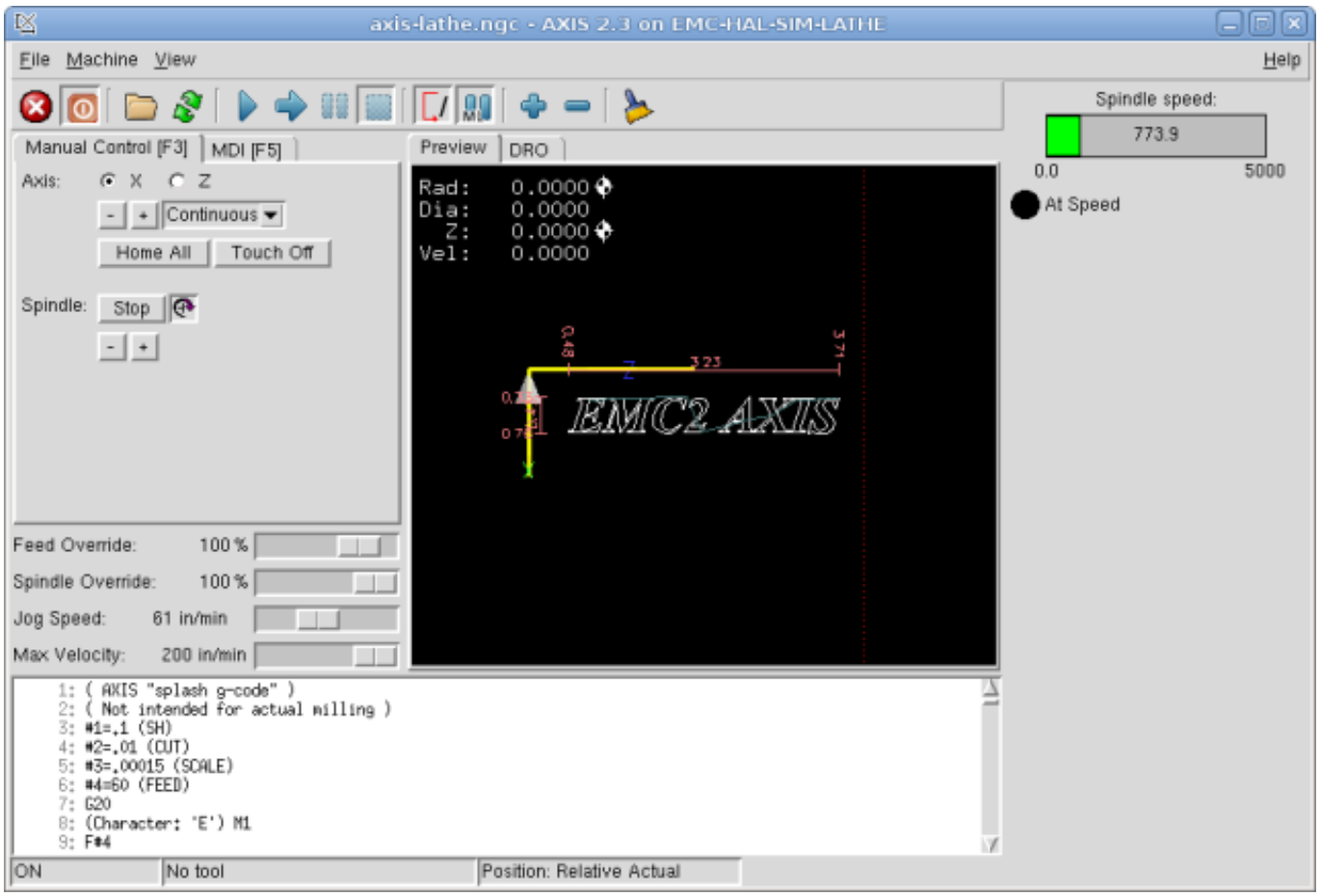
Чтобы наш виджет действительно отображал spindle-speed, его необходимо подключить к соответствующему сигналу HAL. Файл HAL, который будет запущен после запуска AXIS и PyVCP, можно указать в разделе [HAL] INI-файла:

```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

Это изменение запустит команды HAL, указанные в *spindle_to_pyvcp.hal*. В нашем примере содержимое может выглядеть так:

```
net spindle-rpm-filtered => pyvcp.spindle-speed
```

предполагается, что сигнал под названием *spindle-rpm-filtered* уже существует. Обратите внимание, что при совместной работе с AXIS все контакты HAL виджетов панели PyVCP имеют имена, начинающиеся с *pyvcp.*, все контакты HAL встроенных вкладок PyVCP начинаются с имени, указанного как EMBED_TAB_NAME, преобразованного в нижний регистр.




Вот как должна выглядеть вновь созданная панель PyVCP в AXIS. Конфигурация *sim/lathe* уже настроена таким образом.

### 12.1.5 Автономность

В этом разделе описывается, как панели PyVCP могут отображаться самостоятельно с контроллером станка LinuxCNC или без него.

Чтобы загрузить автономную панель PyVCP с LinuxCNC, используйте следующие команды:

```
loadusr -Wn mypanel pyvcp -g WxH+X+Y -c mypanel <path/>panel_file.xml
```

Вы можете использовать это, если вам нужна плавающая панель или панель с ГИП, отличным от AXIS.

- *-Wn panelname* — заставляет HAL ждать завершения загрузки компонента *panelname* (*become ready* на языке HAL) перед обработкой дополнительных команд HAL. Это важно, поскольку панели PyVCP экспортируют контакты HAL, и другим компонентам HAL они потребуются для подключения к ним. Обратите внимание на заглавную букву W и строчную букву n. Если вы используете опцию -Wn, вы должны использовать опцию -c для присвоения имени панели.
- *pyvcp < -g> < -c> Panel.xml* — создает панель с необязательной геометрией и/или именем панели из XML-файла панели. Panel.xml может иметь любое имя, оканчивающееся на .xml. Файл .xml — это файл, описывающий порядок создания панели. Вы должны добавить имя пути, если панель находится не в каталоге, в котором находится сценарий HAL.
- *-g <WxH><+X+Y>* — указывает геометрию, которая будет использоваться при построении панели. Синтаксис: *Width x Height + X Anchor + Y Anchor*. Вы можете установить размер или положение или и то и другое. Точкой привязки является левый верхний угол панели. Пример: -g 250x500+800+0. Это устанавливает ширину панели 250 пикселей, высоту 500 пикселей и привязывает ее к X800 Y0.
- *-c panelname* — сообщает PyVCP, как вызывать компонент, а также заголовок окна. Имя панели может быть любым именем без пробелов.

Чтобы загрузить *stand alone* панель PyVCP без LinuxCNC, используйте следующую команду:

```
loadusr -Wn mypanel pyvcp -g 250x500+800+0 -c mypanel mypanel.xml
```

Минимальная команда для загрузки панели PyVCP:

```
loadusr pyvcp mypanel.xml
```

Вы можете использовать это, если вам нужна панель без контроллера станка LinuxCNC, например, для тестирования или автономного УЦИ.

Команда loadusr используется, когда вы также загружаете компонент, который не позволит HAL закрыться до тех пор, пока он не будет завершен. Если вы загрузили панель, а затем загрузили Classic Ladder с помощью команды *loadusr -w classicladder*, CL будет удерживать HAL открытым (и панель), пока вы не закроете CL. Приведенный выше *-Wn* означает ожидание готовности компонента *-Wn "name"*. (*name* может быть любым именем. Обратите внимание на заглавную букву W и строчную букву n.) Параметр -c сообщает PyVCP о создании панели с именем *panelname*, используя информацию из *panel_file_name.xml*. Имя *panel_file_name.xml* может быть любым, но должно заканчиваться на .xml — это файл, описывающий, как создать панель. Вы должны добавить имя пути, если панель находится не в каталоге, в котором находится сценарий HAL.

Необязательная команда, которую можно использовать, если вы хотите, чтобы панель не позволяла HAL продолжать команды/выключаться. После загрузки любых других компонентов последняя команда HAL должна быть:

```
waitusr panelname
```

Это указывает HAL дождаться закрытия компонента *panelname*, прежде чем продолжать команды HAL. Обычно это последняя команда, чтобы HAL отключался при закрытии панели.

### 12.1.6 Widgets

Сигналы HAL бывают двух вариантов: биты и числа. Биты — это сигналы выключения/включения. Числа могут быть типа *s32*, *u32*, *s64* или *u64*. Дополнительную информацию о типах данных HAL см. в разделе HAL Data. Виджет PyVCP может либо отображать значение сигнала с помощью виджета-индикатора, либо изменять значение сигнала с помощью виджета управления. Таким образом, существует четыре класса виджетов PyVCP, которые вы можете подключить к сигналу HAL. Пятый класс вспомогательных виджетов позволяет вам упорядочивать и маркировать панель.

- Виджеты для индикации *битовых* сигналов: `led`, `rectled`.
- Виджеты для управления `битовыми` сигналами: `button`, `checkbutton`, `radiobutton`.
- Виджеты для индикации *number* сигналов: `number`, `s32`, `u32`, `bar`, `meter`.
- Виджеты для управления *number* сигналами: `spinbox`, `scale`, `jogwheel`.
- Вспомогательные виджеты: `hbox`, `vbox`, `table`, `label`, `labelframe`.

#### 12.1.6.1 Syntax

Каждый виджет описан кратко с последующей используемой разметкой и снимком экрана. Все теги внутри основного тега виджета являются необязательными.

#### 12.1.6.2 Главные примечания

В настоящее время поддерживаются синтаксис как на основе тегов, так и на основе атрибутов. Например, следующие фрагменты XML обрабатываются одинаково:

```
<led halpin=”my-led”/>
```

и

```
<led><halpin>”my-led”</halpin></led>
```

Когда используются синтаксис на основе атрибутов, для преобразования значения атрибута в значение Python используются следующие правила:

1. Если первый символ атрибута является одним из следующих, он оценивается как выражение Python: `{([”’` .
2. Если строка принимается функцией int(), значение рассматривается как целое число.
3. Если строка принимается функцией float(), значение рассматривается как число с плавающей запятой.
4. В противном случае строка принимается как строка.

При использовании синтаксиса на основе тегов текст внутри тега всегда оценивается как выражение Python.

В примерах ниже показано сочетание форматов.

**Comments** Чтобы добавить комментарий, используйте синтаксис комментария xml.

```
<!-- My Comment -->
```

**Редактирование XML-файла** Отредактируйте XML-файл с помощью текстового редактора. В большинстве случаев вы можете щелкнуть файл правой кнопкой мыши и выбрать *open with text editor* или что-то подобное.

**Цвета**

Цвета можно указать с помощью цветов RGB X11 по имени *gray75* или шестнадцатеричному имени *#0000ff*. Полный список находится здесь https://sedition.com/perl/rgb.html.

Общие цвета (цвета с цифрами обозначают оттенки этого цвета)

- белый
- черный
- синий и синий1 - 4
- голубой и голубой1–4
- зеленый и зеленый1 - 4
- желтый и желтый1 - 4
- красный и красный1 - 4
- фиолетовый и фиолетовый1 - 4
- серый и серый0 - 100

**HAL Контакты** Контакты HAL предоставляют возможность *подключить* виджет к чему-либо. После того как вы создадите контакт HAL для своего виджета, вы можете *подключить* его к другому контакту HAL с помощью команды *net* в файле .hal. Дополнительную информацию о команде net см. в разделе HAL-команды.

#### 12.1.6.3 Label

Метка — это способ добавить текст на панель.

- *<label></label>* - создает метку.
- *<text>”text”</text>* - текст, который нужно поместить на метку, пустую метку можно использовать в качестве разделителя для выравнивания других объектов.
- *<font>(”Helvetica”,20)</font>* - укажите шрифт и размер текста.
- *<relief>FLAT</relief>* - укажите рамку вокруг метки (*FLAT*, *RAISED*, *SUNKEN*), по умолчанию — *FLAT*.
- *<bd>_n_</bd>* - где *n* — ширина границы при использовании границ *RAISED* или *SUNKEN*.
- *<padx>_n_</padx>* - где *n* — количество дополнительного горизонтального пространства.
- *<pady>_n_</pady>* - где *n* — количество дополнительного вертикального пространства.

Метка имеет дополнительный контакт запрещения, который создается при добавлении *<disable_pin>True</disable_pin>*.

```
<label>
    <text>”This is a Label:”</text>
    <font>(”Helvetica”,20)</font>
</label>
```

Приведенный выше код создал этот пример:

Figure 12.3: Простой пример этикетки

#### 12.1.6.4 Мульти_метка

Расширение текстовой метки.

Выбираемая текстовая метка, может отображать до 6 легенд меток, когда соответствующий битовый контакт активирован.

Прикрепите каждый контакт легенды к сигналу и получите описательную метку, когда сигнал будет TRUE.

Если более чем один контакт легенды имеет значение TRUE, будет отображаться легенда *TRUE* с наибольшим номером.

Если контакт запрещения создан с параметром *<disable_pin>True</disable_pin>* и для этого контакта установлено значение true, метка становится серой.

```
<multilabel>
    <legends>["Label1", "Label2", "Label3", "Label4", "Label5", "Label6"]</legends>
    <font>("Helvetica",20)</font>
    <disable_pin>True</disable_pin>
</multilabel>
```

В приведенном выше примере будут созданы следующие контакты.

```
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
```

Если у вас более одной мультиметки, созданные контакты будут увеличивать число, например: *pyvcp.multilabel.1.legend1*.

#### 12.1.6.5 СИДы

СИД используется для индикации состояния *bit* контакта HAL. Цвет светодиода будет on_color, если halpin имеет значение true, и off_color в противном случае.

- *<led></led>* - создает круглый СИД
- *<rectled></rectled>* - создает прямоугольный СИД
- *<halpin>*name*</halpin>* - *имя* контакта, по умолчанию — *led*.n, где *n* — целое число, которое увеличивается для каждого светодиода.
- *<size>*n*</size>* - *n* — размер светодиода в пикселях, по умолчанию — 20.

- *<on_color>*color*</on_color>* - устанавливает цвет светодиода в *color*, когда контакт true. По умолчанию — *зеленый*. Дополнительную информацию см. в разделе colors.
- *<off_color>*color*</off_color>* - устанавливает цвет светодиода в *color*, когда контакт false. По умолчанию — *красный*.
- *<height>*n*</height>* - задает высоту светодиода в пикселях.
- *<width>*n*</width>* - устанавливает ширину СИД в пикселях.
- *<disable_pin>false</disable_pin>* - когда true добавляет контакт запрещения к светодиоду.
- *<disabled_color>*color*</disabled_color>* - устанавливает цвет СИД в *color*, когда контакт запрещен.

**Круглый СИД**

```
<led>
    <halpin>”my-led”</halpin>
    <size>50</size>
    <on_color>”green”</on_color>
    <off_color>”red”</off_color>
</led>
```

Приведенный выше код создал этот пример:

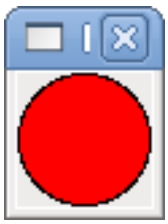


Figure 12.4: Пример круглого СИД

**Прямоугольный СИД** Это вариант виджета *led*.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <rectled>
        <halpin>”my-led”</halpin>
        <height>”50”</height>
        <width>”100”</width>
        <on_color>”green”</on_color>
        <off_color>”red”</off_color>
    </rectled>
</vbox>
```

Этот пример создал приведенный выше код . Также показано горизонтальное поле с объемным периметром.

Figure 12.5: Пример простого прямоугольного СИД

#### 12.1.6.6 Кнопки

Кнопка используется для управления контактом BIT. Контакт будет установлен в True, когда кнопка нажата и удерживается, и будет установлен как False, когда кнопка отпущена. Кнопки могут использовать следующие дополнительные параметры.

- *<padx>*n*</padx>* - где *n* — количество дополнительного горизонтального пространства.
- *<pady>*n*</pady>* - где *n* — количество дополнительного вертикального пространства.
- *<activebackground>”*color*”</activebackground>* - курсор над цветом установлен на *color*.
- *<fg>”*color*”</fg>* - цвет переднего плана установлен на *color*.
- *<bg>”*color*”</bg>* - цвет фона установлен на *color*.
- *<disable_pin>True</disable_pin>* - запретить контакт.

**Текстовая кнопка** Текстовая кнопка управляет *bit* контактом HAL. Контакт HAL false до тех пор, пока не будет нажата кнопка, тогда это true. Кнопка является кнопкой мгновенного действия.

Текстовая кнопка имеет дополнительный контакт запрещения, который создается при добавлении <disable_pin>True</disable_pin>.

```
<button>
    <halpin>”ok-button”</halpin>
    <text>”OK”</text>
</button>
<button>
    <halpin>”abort-button”</halpin>
    <text>”Abort”</text>
</button>
```

Приведенный выше код создал этот пример:

Figure 12.6: Пример простых кнопок

**Фиксирующая кнопка** Фиксирующая кнопка управляет *bit* контактом HAL. Контакт HAL будет установлен в True, когда кнопка отмечена, и false, когда кнопка не отмечена. Фиксирующая кнопка представляет собой кнопку типа переключатель. Фиксирующие кнопки могут быть первоначал установлены как TRUE или FALSE в поле initval. Также автоматически создается контакт, называемый changepin, который может переключать фиксирующую кнопку через HAL, если связанное значение изменяется, чтобы обновлять дисплей.

Figure 12.7: Не нажатая кнопка

Figure 12.8: Нажатая кнопка

**Пример кода фиксирующей кнопки**

```
<checkbutton>
    <halpin>”coolant-chkbtn”</halpin>
    <text>”Coolant”</text>
    <initval>1</initval>
</checkbutton>
<checkbutton>
    <halpin>”chip-chkbtn”</halpin>
    <text>”Chips    ”</text>
    <initval>0</initval>
</checkbutton>
```

Приведенный выше код создал этот пример:

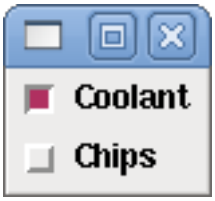



Figure 12.9: Простой пример фиксирующей кнопки

Фиксирующая кнопка охлаждающей жидкости нажата.
Обратите внимание на дополнительные пробелы в тексте ”Chips”, чтобы фиксирующие кнопки были выровнены.

**Кнопка с зависимой фиксацией** Кнопка с зависимой фиксацией установит один из контактов HAL в true. Остальные контакты установлены в false. Поле initval может быть настроено на выбор значения по умолчанию при отображении панели. Только одна кнопка с зависимой фиксацией может быть установлена в TRUE (1), или только конаткт с наибольшим номером, установленный в TRUE, будет иметь это значение.

```
<radiobutton>
    <choices>[”one”,”two”,”three”]</choices>
    <halpin>”my-radio”</halpin>
    <initval>0</initval>
</radiobutton>
```

Приведенный выше код создал этот пример:

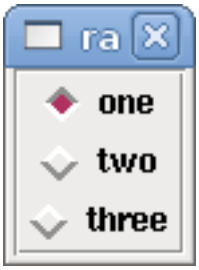



Figure 12.10: Простой пример кнопки с зависимой фиксацией

Обратите внимание, что контакты HAL в приведенном выше примере будут называться my-radio.one, my-radio.two и my-radio.three. На изображении выше выбрано значение *one*. Используйте тег `<orient>HORIZONTAL</orient>` для горизонтального отображения.

#### 12.1.6.7 Числовые индикаторы

Числовые индикаторы могут использовать следующие параметры форматирования

- `<font>("Font Name",n)</font>`, где *n* — размер шрифта.
- `<width>_n_</width>`, где *n* — общая ширина используемого пространства.
- `<justify>_pos_</justify>`, где *pos* — LEFT, CENTER или RIGHT (не работает).
- `<padx>_n_</padx>`, где *n* — количество дополнительного горизонтального пространства.
- `<pady>_n_</pady>`, где *n* — количество дополнительного вертикального пространства.

**Число** Числовой виджет отображает значение сигнала с плавающей запятой.

```
<number>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

Приведенный выше код создал этот пример:

Figure 12.11: Простой числовой пример

- *<font>* - это спецификация типа и размера шрифта Tkinter. Один шрифт, который будет отображаться размером как минимум до 200, — это *courier 10 pitch*, поэтому для действительно большого числового виджета вы можете указать:

```
<font>("courier 10 pitch",100)</font>
```

- *<format>* - указанный формат *в стиле C*, определяющий способ отображения числа.

**s32 число** Виджет числа s32 отображает значение чисоа s32. Синтаксис такой же, как у *числа*, за исключением имени, которое <s32>. Убедитесь, что ширина поля достаточна, чтобы соответствова наибольшему числу, которое вы планируете использовать.

```
<s32>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"6d"</format>
    <width>6</width>
</s32>
```

Приведенный выше код создал этот пример:

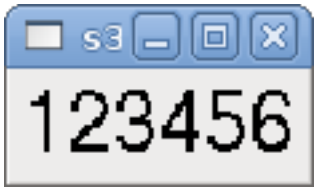



Figure 12.12: Просто пример числа s32

**u32 Число** Виджет числа u32 отображает значение числа u32. Синтаксис такой же, как и у *number*, за исключением имени, которое <u32>.

**Шкала** Виджет шкала отображает значение сигнала FLOAT как в графическом виде, используя шкалу, так и в виде числа. Цвет полосы может быть установлен как один цвет во всем диапазоне (по умолчанию используется цвет заливки) или установлен для изменения цвета, в зависимости от значения halpin (должны быть установлены все диапазоны: range1, range2 range3; если вам нужны только два диапазона, установите 2 из них одного цвета).

- `<halpin>"my-bar"</halpin>` (текст), извлекает и устанавливает имя контакта: `pyvcp.my-bar`.
- `<min_>0</min_>` (число), устанавливает минимальную шкалу.
- `<max_>140</max_>` (число), устанавливает максимальную шкалу.
- `<format>"3.1f"</format>` (текст), устанавливает числовой формат с использованием форматирован чисел Python.
- `<bgcolor>"grey"</bgcolor>` (текст), устанавливает цвет фона.
- `<fillcolor>"red"</fillcolor>` (текст), устанавливает цвет заливки.
- `<range1>0,100,"green"</range1>` (число, число, текст) устанавливает первый диапазон и цвет.
- `<range2>101,135,"orange"</range2>` (число, число, текст) устанавливает первый диапазон и цвет.
- `<range3>136, 150,"red"</range3>` (число, число, текст) задает первый диапазон и цвет.
- `<canvas_width>200</canvas_width>` (число), устанавливает общую ширину.
- `<canvas_height>50</canvas_height>` (число), устанавливает общую высоту.

- <bar_height>30</bar_height> (число), задает высоту панели, должно быть меньше canvas_height.
- <bar_width>150</bar_width> (число), задает ширину полосы, должна быть меньше canvas_width.

```
<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>
```

Приведенный выше код создал этот пример:

Figure 12.13: Простой пример шкалы

**Meter** Индикатор часового типа отображает значение сигнала FLOAT.

```
<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>
```

Приведенный выше код создал этот пример:

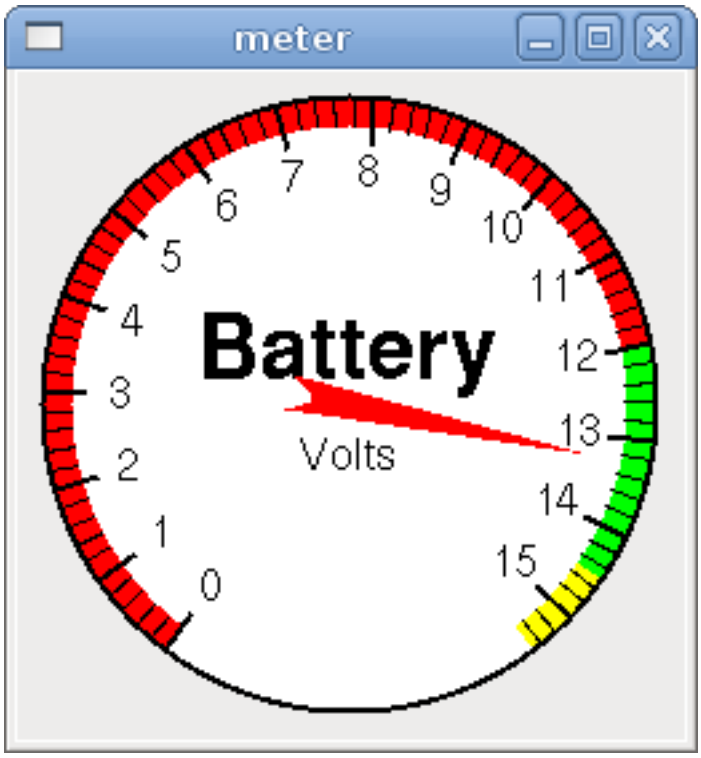



Figure 12.14: Простой пример индикатора часового типа

#### 12.1.6.8 Ввод чисел

**Окошко счетчика** Счетчик управляет контактом FLOAT. Вы увеличиваете или уменьшаете значение контакта, нажимая на стрелки или выделяя поле прокрутки и вращая колесо мыши. Если в поле param_pin установлено значение TRUE(1), будет создан контакт, который можно использовать для установки исходного значения счетчика и для удаленного изменения его значения без ввода HID.

```
<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>
```

Приведенный выше код создал этот пример:

Figure 12.15: Простой пример окошка счетчика

**Шкала с ползунком** Шкала управляет float или s32 контактом. Вы увеличиваете или уменьшаете значение шкалы, перетаскивая ползунок или выделяя шкалу и вращая колесо мыши. К *halpin* будут добавлены *-f* и *-i*, чтобы сформировать контакты float и s32. Width — это ширина ползунка в вертикальной ориентации и высота ползунка в горизонтальной ориентации. Если в поле param_pin установлено значение TRUE(1), будет создан контакт, который можно использовать для установки исходного значения счетчика и для удаленного изменения его значения без входа HID.

```
<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>
```

Приведенный выше код создал этот пример:

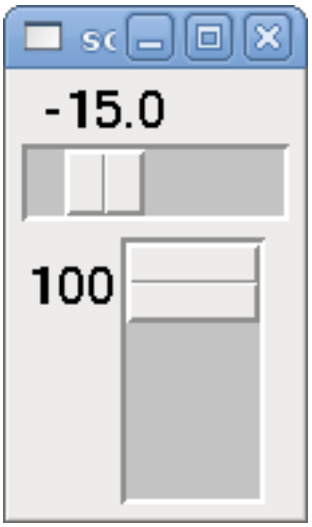



Figure 12.16: Простой пример шкалы с ползунком

**Note**
Обратите внимание, что по умолчанию отображается "min", даже если оно больше "max", пока "min" не станет отрицательным.

**Циферблат** Dial выводит значение HAL с плавающей запятой (float) и реагирует как на колесико мыши, так и на перетаскивание. Дважды щелкните левой кнопкой мыши, чтобы увеличить

разрешение, и дважды щелкните правой кнопкой мыши, чтобы уменьшить разрешение на одну цифру. Вывод ограничен минимальными и максимальными значениями. <cpr> — это количество делений на наружном кольце (остерегайтесь больших чисел). Если в поле param_pin установлено значение TRUE(1), будет создан контакт, который можно использовать для установки исходного значения счетчика и для удаленного изменения его значения без HID входа.

```
<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>
```

Приведенный выше код создал этот пример:

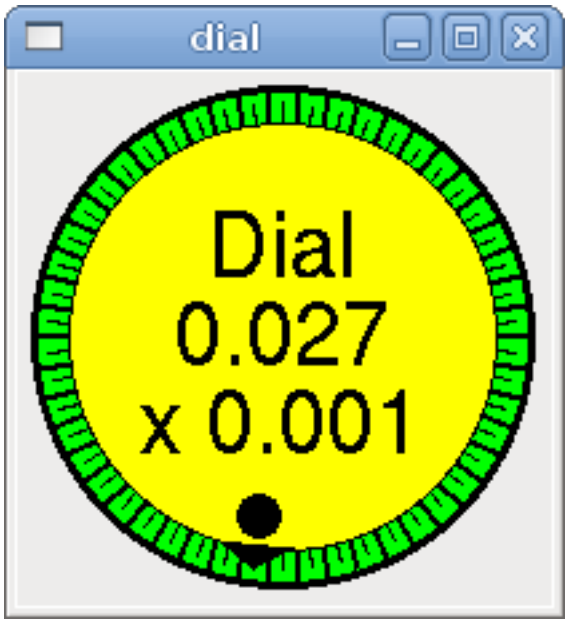



Figure 12.17: Простой пример циферблата

**Безупорный регулятор** Jogwheel имитирует настоящий безупорный регулятор, выводя контакт FLOAT, который ведет отсчет вверх или вниз при вращении колеса, либо путем перетаскивания кругового движения, либо путем вращения колеса мыши.

Опциональные тэги: * *<text>”My Text”</text>* отображает текст * *<bgcolor>”grey”</bgcolor>* *<fillcolor>”green”</fillcolor>* фон и активные цвета * *<scale_pin>1</scale_pin>* создает текст шкалы и контакт `FLOAT.scale` для отображения шкалы регулятора * *<clear_pin>1</clear_pin>* создает `DRO` и контакт `BIT.reset` для сброса DRO. Требуется Scale_pin для масштабируемого DRO. Shift+клик также сбрасывает DRO

```
<jogwheel>
    <halpin>”my-wheel”</halpin>
    <cpr>45</cpr>
```

```
    <size>250</size>
</jogwheel>
```

Приведенный выше код создал этот пример:

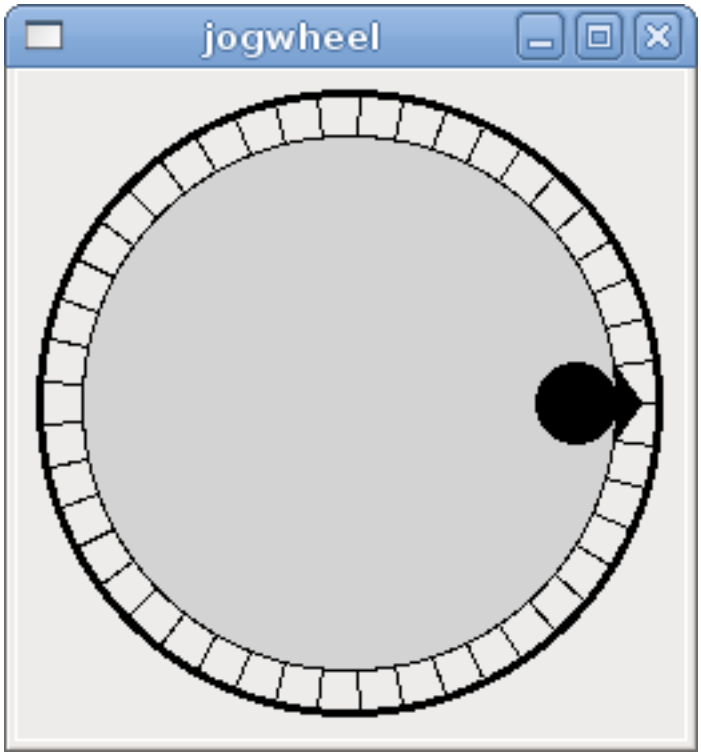



Figure 12.18: Простой пример безупорного регулятора

#### 12.1.6.9 Изображения

Image отображает только изображения формат .gif. Все изображения должны быть одинакового размера. Изображения должны находиться в том же каталоге, что и ваш INI-файл (или в текущем каталоге, если он запускается из командной строки с помощью halrun/halcmd).

**Бит изображения** *image_bit* осуществляет переключение между двумя изображениями, устанавлива halpin в true или false.

```
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_bit halpin='selectimage' images='fwd rev'/>
</vbox>
```

Этот пример был создан вышеприведенным кодом. Используются два файла изображений fwd.gif и rev.gif. FWD отображается, когда *selectimage* в false, а REV отображается, когда false в true.

Figure 12.19: Пример выбора изображения в False

Figure 12.20: Пример вывода изображения в True

**Изображение u32** *image_u32* аналогичен *image_bit*, за исключением того, что у вас по существу неограниченное количество изображений, и вы *выбираете* изображение, устанавливая для halpin целочисленное значение 0 для первого изображения в списке изображений, 1 для второго изображени и т. д.

```
<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>
```

Приведенный выше код создал следующий пример добавлением изображения stb.gif.

Figure 12.21: Простой пример image_u32 с halpin=0

Figure 12.22: Простой пример image_u32 с halpin=1

Figure 12.23: Простой пример image_u32 с halpin=2

Обратите внимание, что по умолчанию установлено минимальное значение, даже если оно установлен
выше максимального, если только не существует отрицательного минимального значения.

#### 12.1.6.10 Контейнеры

Containers — это виджеты, содержащие другие виджеты. Контейнеры используются для группировки других виджетов.

**Границы** Границы контейнера задаются двумя тегами, используемыми вместе. Тег <relief> указывает тип границы, а <bd> определяет ширину границы.

***<relief>_type_</relief>***
: Где *type* — FLAT, SUNKEN, RAISED, GROOVE, или RIDGE.

***<bd>_n_</bd>***
: Где *n* — ширина границы.

```
<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>
```

Приведенный выше код создал этот пример:

Figure 12.24: Пример границ контейнеров

**Заполнение** Заполнение контейнера указывается с помощью тега <*boxfill fill=""*/>. Допустимые значения: none, x, y и оба. Заливка X — это горизонтальная заливка, а заливка Y — вертикальная заливка

**<boxfill fill ="style"/>**
Где *style* — none, x, y или оба. По умолчанию — x для Vbox и y для Hbox.

**Якорь** Якоря контейнера указываются с помощью тега <boxanchor anchor=""/> . Якорь указывает, где разместить каждого ведомого в его пакете. Допустимые значения: center, n, s, e, w для центра, севера, юга, востока и запада. Также допустимы такие комбинации, как sw, se, nw и ne.

**<boxanchor anchor="position"/>**
Где *position* — center, n, s, e, w, ne, nw, se или sw. По умолчанию — center.

**Расширение** Расширение контейнера указывается с помощью логического тега <boxexpand expand=""/>. Допустимые значения: "yes", "no".

**<boxexpand expand="boolean"/>**
Где *boolean* означает "yes" или "no". По умолчанию "yes".

**Hbox** Используйте Hbox, если хотите расположить виджеты горизонтально рядом друг с другом.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a hbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

Приведенный выше код создал этот пример:

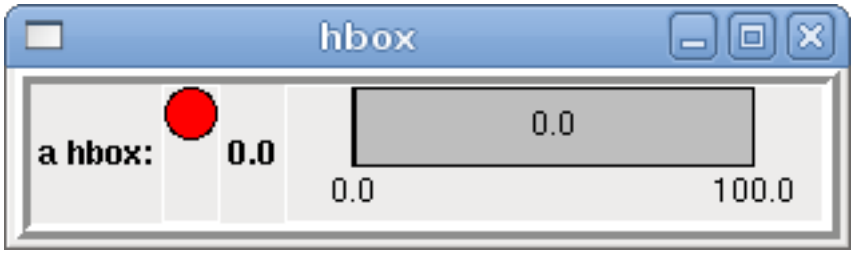



Figure 12.25: Простой пример hbox

Внутри Hbox вы можете использовать теги <*boxfill fill=””/>*, <*boxanchor anchor=””/>*, and <*box-expand expand=””/>*, чтобы выбрать, как будут вести себя элементы в поле при изменении размера окна. По умолчанию для Hbox используется *fill=”y”*, *anchor=”center”*, *expand=”yes”*.

**Vbox** Используйте Vbox, если хотите расположить виджеты вертикально друг над другом.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>”a vbox:”</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```

Приведенный выше код создал этот пример:

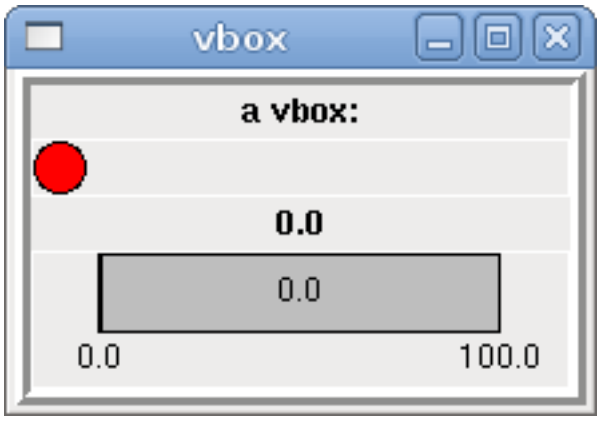



Figure 12.26: Простой пример vbox

Внутри Vbox вы можете использовать теги <*boxfill fill=””/>*, <*boxanchor anchor=””/>*, and <*box-expand expand=””/>*, чтобы выбрать, как будут вести себя элементы в поле. при изменении размера окна. По умолчанию для Vbox используется *fill=”x”*, *anchor=”center”*, *expand=”yes”*.

**Этикетка для рамки** Этикетка для рамки — это рамка с бороздкой и этикеткой в верхнем левом углу.

```
<labelframe text=”Label: Leds groupées”>
```

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

Приведенный выше код создал этот пример:

Figure 12.27: Простой пример этикетки для рамки

**Таблица** Таблица — это контейнер, который позволяет размещать данные в сетке строк и столбцов. Каждая строка начинается с тега *<tablerow/>*. Содержащийся виджет может охватывать строки или столбцы с помощью тега *<tablespan rows= cols=/>*. Стороны ячеек, к которым "stick" (прикрепляю содержащиеся в них виджеты, могут быть установлены с помощью тега *<tablesticky sticky=/>*. Таблица расширяется за счет гибких строк и столбцов.

**Пример кода таблицы**

```
<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>
```

Приведенный выше код создал этот пример:

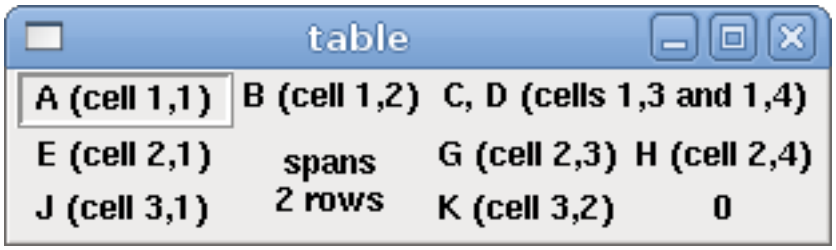
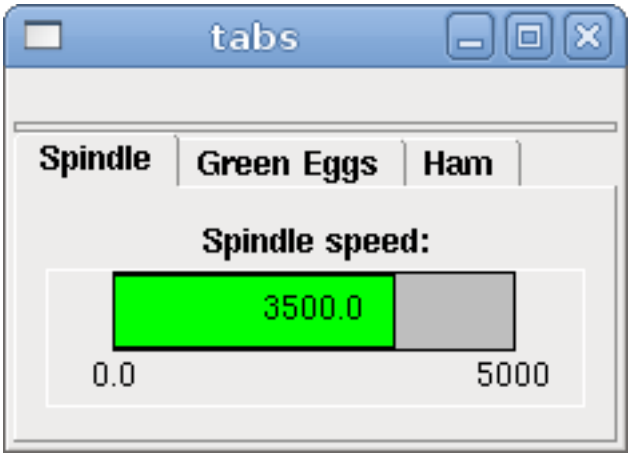



Figure 12.28: Пример таблицы

**Закладки** Интерфейс с вкладками может сэкономить немало места.

```
<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>
```

Приведенный выше код создал этот пример, показывающий каждую выбранную вкладку.



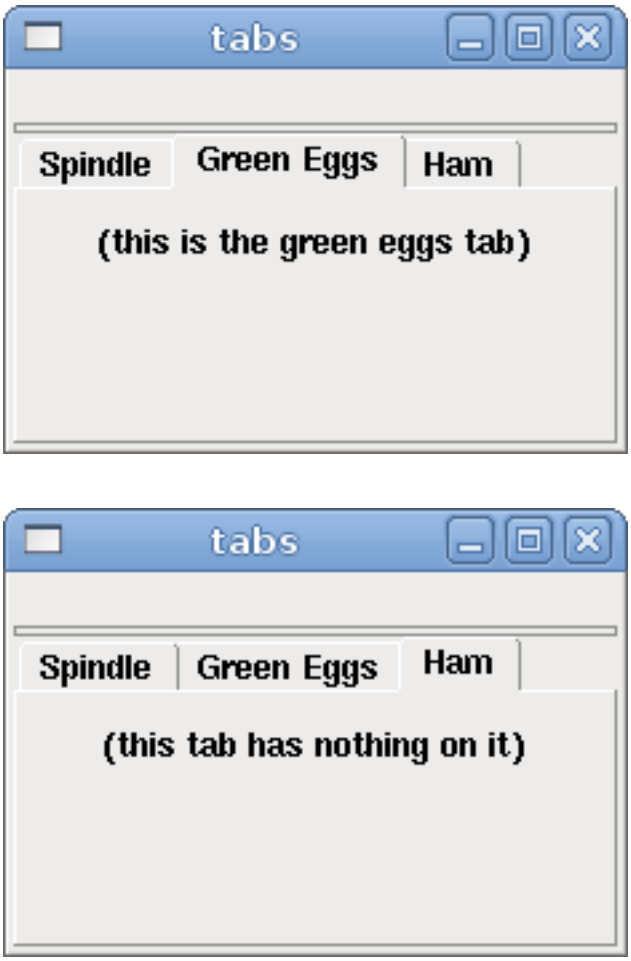



Figure 12.29: Простой пример вкладок

## 12.2 Примеры PyVCP

### 12.2.1 AXIS

Чтобы создать панель PyVCP для использования с интерфейсом AXIS, прикрепленным справа от AXIS, вам необходимо выполнить следующие основные действия.

- Создайте XML-файл, содержащий описание вашей панели, и поместите его в каталог конфигурации.
- Добавьте запись PyVCP в раздел [DISPLAY] INI-файла с именем вашего XML-файла.
- Добавьте запись POSTGUI_HALFILE в раздел [HAL] INI-файла с именем вашего файла postgui HAL.
- Добавьте ссылки на контакты HAL для вашей панели в файл postgui.hal, чтобы *подключить* вашу панель PyVCP к LinuxCNC.

### 12.2.2 Плавающие панели

Чтобы создать плавающие панели PyVCP, которые можно использовать с любым интерфейсом, вам необходимо выполнить следующие основные действия.

- Создайте XML-файл, содержащий описание вашей панели, и поместите его в каталог конфигурации.
- Добавьте строку loadusr в ваш файл HAL, чтобы загрузить каждую панель.
- Добавьте ссылки на контакты HAL для вашей панели в файл postgui.hal, чтобы *подключить* вашу панель PyVCP к LinuxCNC.

Ниже приведен пример команды loadusr для загрузки двух панелей PyVCP и присвоения каждой из них имени, чтобы имена соединений в HAL были известны.

```
loadusr -Wn btnpanel pyvcp -c btnpanel panel1.xml
loadusr -Wn sppanel pyvcp -c sppanel panel2.xml
```

-Wn заставляет HAL *Wait for name* перед загрузкой.

”pyvcp -c” заставляет PyVCP именовать панель.

Контакты HAL из файла panel1.xml будут называться btnpanel.<_pin name_>.

Контакты HAL из файла panel2.xml будут называться sppanel.<_pin name_>.

Убедитесь, что линия loadusr находится перед любыми цепями, использующими контакты PyVCP.

### 12.2.3 Пример кнопок подачи

В этом примере мы создадим панель PyVCP с кнопками перемещения для X, Y и Z. Эта конфигурация будет построена на основе конфигурации, созданной мастером StepConf. Сначала мы запускаем мастер StepConf и настраиваем наш станок, затем на странице Advanced Configuration Options мы делаем несколько вариантов, чтобы добавить пустую панель PyVCP, как показано на следующем рисунке. В этом примере мы назвали конфигурацию *pyvcp_xyz* на странице базовой информации о компьютере мастера StepConf.

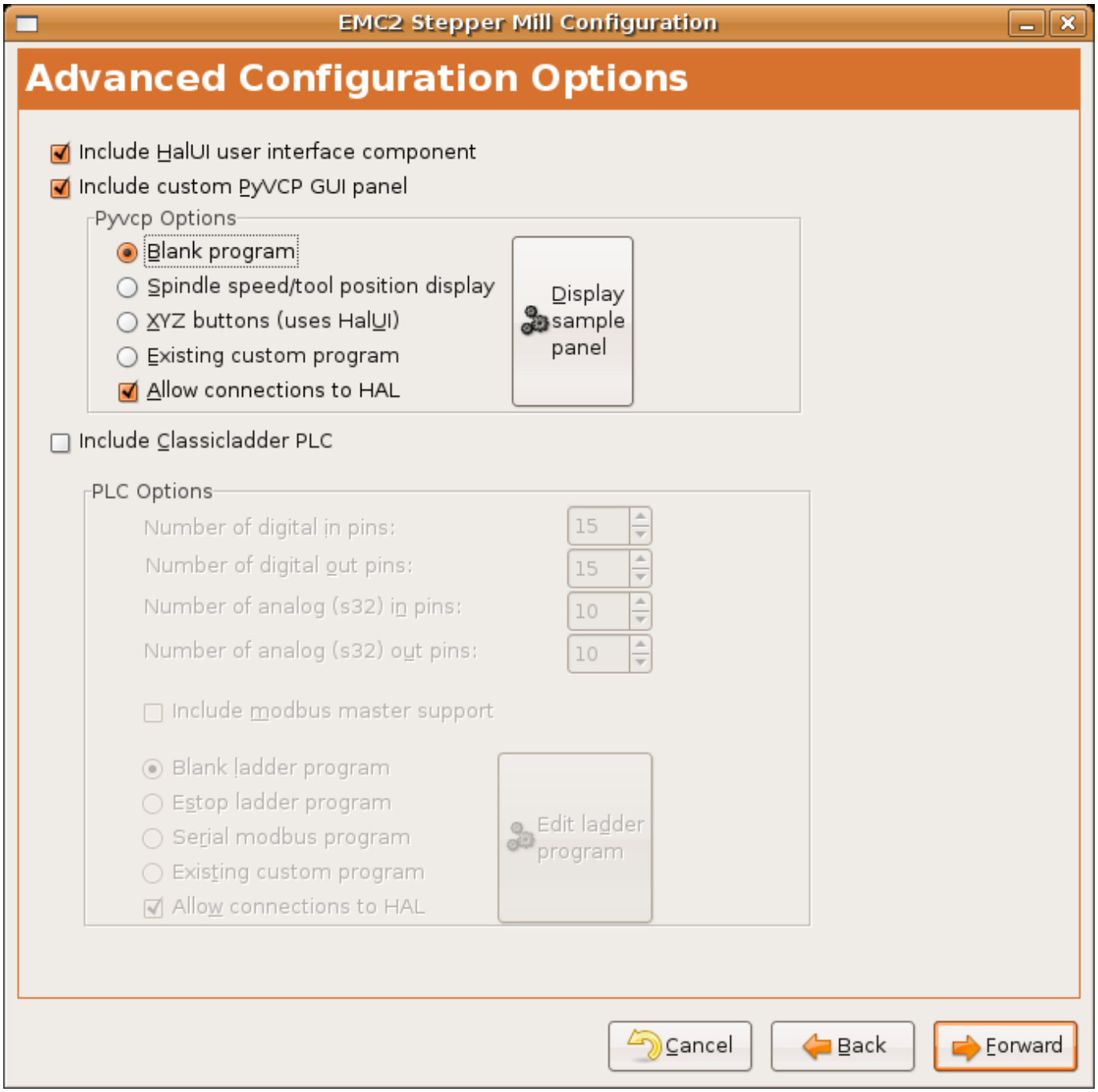



Figure 12.30: Конфигурация мастера XYZ

Мастер StepConf создаст несколько файлов и поместит их в каталог `linuxcnc/configs/pyvcp_xyz`. Если вы отметили галочкой создать ссылку, у вас будет ссылка на эти файлы на рабочем столе.

#### 12.2.3.1 Создайте виджеты

Откройте файл custompanel.xml, щелкнув на нем правой кнопкой мыши и выбрав *open with text editor*. Между тегами <pyvcp></pyvcp> мы добавим виджеты для нашей панели.

Более подробную информацию о каждом виджете можно найти в разделе PyVCP Widgets Reference руководства documentation des widgets.

В ваш файл custompanel.xml мы добавим описание виджетов.

```
<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>

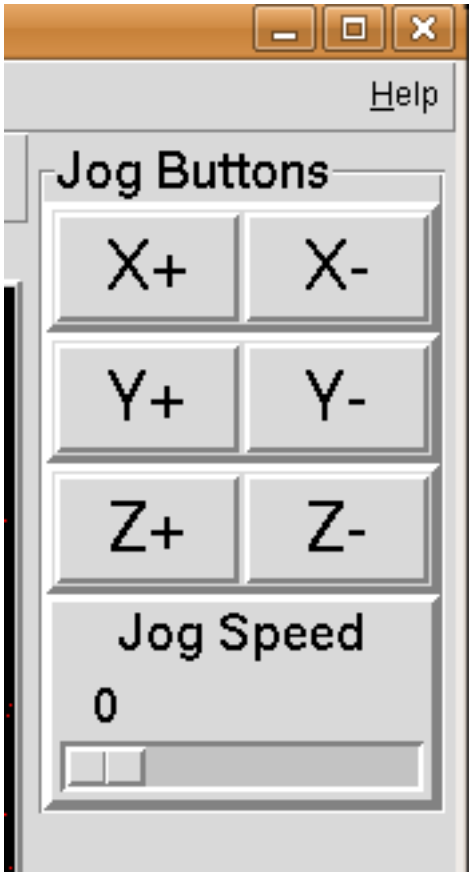
      <!-- the Y jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"y-plus"</halpin>
        <text>"Y+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <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>
```

После добавления вышеприведенного у вас теперь будет панель PyVCP, прикрепленная к правой стороне AXIS, которая выглядит следующим образом. Выглядит красиво, но ничего не делает, пока вы не *подключите* кнопки к halui. Если при попытке запуска вы получаете сообщение об ошибке, прокрутите вниз до нижней части всплывающего окна. Обычно ошибка представляет собой орфографическую или синтаксическую ошибку, и она будет там.




#### 12.2.3.2 Сделайте соединения

Чтобы установить необходимые соединения, откройте файл custom_postgui.hal и добавьте следующее.

```
# connect the X PyVCP buttons
```

```
net my-jogxminus halui.axis.x.minus <= pyvcp.x-minus
net my-jogxplus halui.axis.x.plus <= pyvcp.x-plus

# connect the Y PyVCP buttons
net my-jogyminus halui.axis.y.minus <= pyvcp.y-minus
net my-jogyplus halui.axis.y.plus <= pyvcp.y-plus

# connect the Z PyVCP buttons
net my-jogzminus halui.axis.z.minus <= pyvcp.z-minus
net my-jogzplus halui.axis.z.plus <= pyvcp.z-plus

# connect the PyVCP jog speed slider
net my-jogspeed halui.axis.jog-speed <= pyvcp.jog-speed-f
```

После сброса аварийного останова, перевода его в режим медленной подачи и перемещения ползунка скорости медленной подачи на панели PyVCP до значения большего нуля, кнопки медленной подачи PyVCP должны работать. Вы не можете использовать медленную подачу во время работы файла G-кода, во время паузы или когда выбрана вкладка MDI.

### 12.2.4 Тестер порта

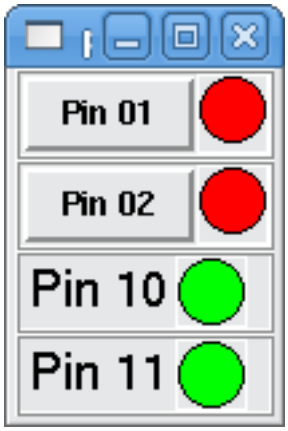
В этом примере показано, как создать простой тестер параллельного порта, используя PyVCP и HAL.

Сначала создайте файл ptest.xml со следующим кодом, чтобы создать описание панели.

```
<!-- 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>
```

Этот код создаст следующую плавающую панель, содержащую пару входных и пару выходных контактов.



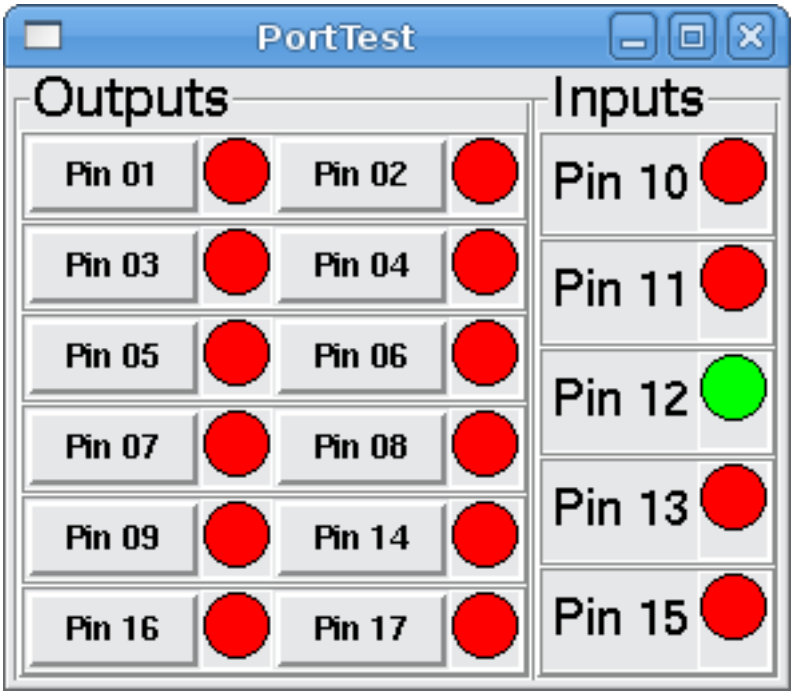

Чтобы запустить команды HAL, которые нам нужны для запуска всего, мы помещаем следующее в наш файл ptest.hal.

```
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
```

Чтобы запустить файл HAL, мы используем следующую команду из окна терминала.

```
~$ halrun -I -f ptest.hal
```

На следующем рисунке показано, как может выглядеть полная панель.




Чтобы добавить остальные контакты параллельного порта, просто измените файлы XML и HAL.

Чтобы отобразить контакты после запуска сценария HAL, используйте следующую команду в командной строке halcmd:

```
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
```

Это покажет вам, какие контакты являются входными, а какие — выходными, а также любые соединения.

## 12.2.5 GS2 измеритель оборотов

В следующем примере используется VDF-драйвер Automation Direct GS2 и отображается RPM и другая информация на панели PyVCP. Этот пример основан на примере GS2 в разделе «Примеры оборудования» данного руководства.

### 12.2.5.1 Панель

Чтобы создать панель, мы добавляем следующее в XML-файл.

```
<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>

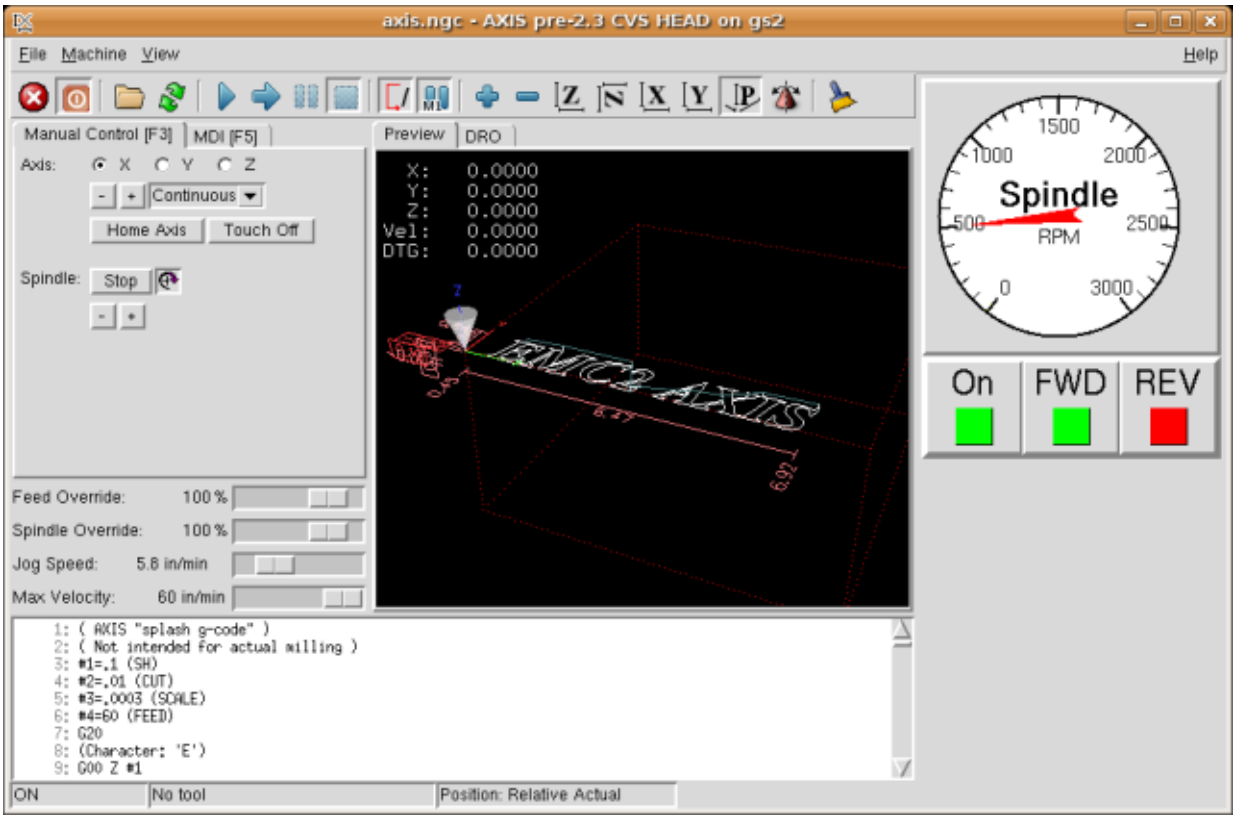
  <!-- the On Led -->
  <hbox>
  <relief>RAISED</relief>
  <bd>3</bd>
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
  <label>
  <text>”On”</text>
  <font>(”Helvetica”,18)</font>
  </label>
  <width>5</width>
   <hbox>
  <label width=”2”/> <!-- used to center the led -->
  <rectled>
  <halpin>”on-led”</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>
```

Вышеприведенное дает нам панель PyVCP, которая выглядит следующим образом.




#### 12.2.5.2 Соединения

Чтобы это заработало, мы добавим следующий код в файл custom_postgui.hal.

```
# display the rpm based on freq * rpm per hz
loadrt mult2
addf mult2.0 servo-thread
setp mult2.0.in1 28.75
net cypher_speed mult2.0.in0 <= spindle-vfd.frequency-out
net speed_out pyvcp.spindle_rpm <= mult2.0.out

# run led
net gs2-run => pyvcp.on-led

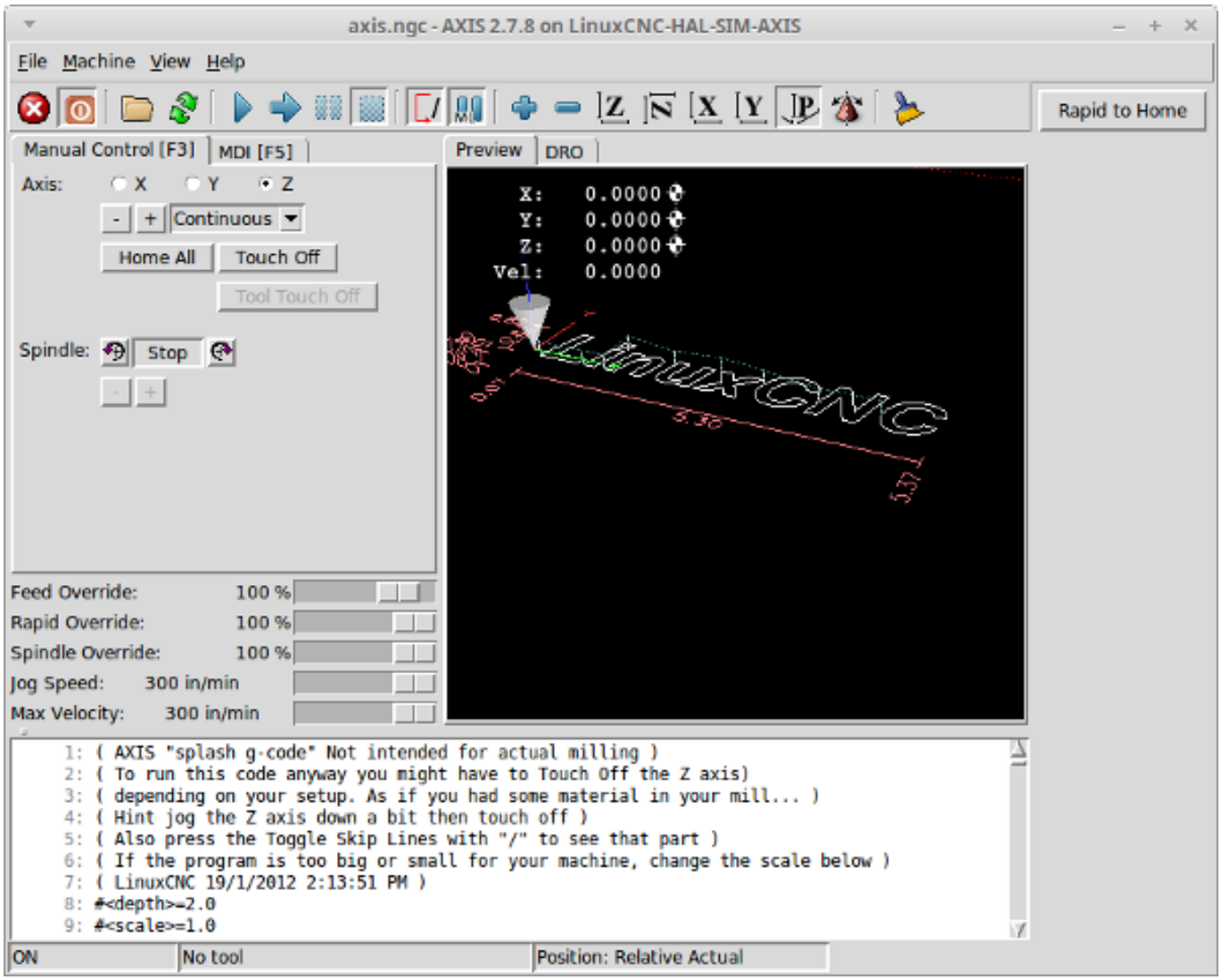
# fwd led
net gs2-fwd => pyvcp.fwd-led

# rev led
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Некоторые строки, возможно, потребуют пояснений. Линия индикатора вперед использует сигнал, созданный в файле custom.hal, тогда как индикатор обратно должен использовать бит шпинделя spindle-rev. Вы не можете связать бит шпинделя spindle-fwd, поэтому вы используете сигнал, с которым он был связан.

### 12.2.6 Кнопка быстрого приведения в исходное положение

В этом примере создается кнопка на боковой панели PyVCP, при нажатии которой все оси возвращаются в исходное положение. В этом примере предполагается, что у вас нет панели PyVCP.




В каталоге конфигурации создайте XML-файл. В этом примере он называется *rth.xml*. В файле *rth.xml* добавьте следующий код для создания кнопки.

```
<pyvcp>
<!-- rapid to home button example -->
<button>
<halpin>”rth-button”</halpin>
<text>”Rapid to Home”</text>
</button>
</pyvcp>
```

Откройте INI-файл в текстовом редакторе и в разделе [DISPLAY] добавьте следующую строку. Это то, что загружает панель PyVCP.

```
PYVCP = rth.xml
```

Если у вас нет раздела [HALUI] в INI-файле, создайте его и добавьте следующую команду MDI.

```
MDI_COMMAND = G53 G0 X0 Y0 Z0
```

**Note**
Информация о G-кодах G53 и G0.

Если у вас нет файла post gui, в разделе [HAL] добавьте следующее и создайте файл с именем *postgui.hal*.

```
POSTGUI_HALFILE = postgui.hal
```

В файле *postgui.hal* добавьте следующий код, чтобы связать кнопку PyVCP с командой MDI.

```
net rth halui.mdi-command-00 <= pyvcp.rth-button
```

**Note**
Информация о команде net

## 12.3 GladeVCP: Виртуальная панель управления

### 12.3.1 Что такое GladeVCP?

GladeVCP — это компонент LinuxCNC, который добавляет возможность добавлять новую панель пользовательского интерфейса к пользовательским интерфейсам LinuxCNC, например:

- AXIS
- Touchy
- Gscreen
- GMOCCAPY

В отличие от PyVCP, GladeVCP не ограничивается отображением и настройкой контактов HAL, поскольку произвольные действия могут выполняться в коде Python — фактически, с помощью GladeVCP и Python можно создать полный пользовательский интерфейс LinuxCNC.

GladeVCP использует Glade редактор пользовательского интерфейса WYSIWYG, который позволяет легко создавать визуально привлекательные панели. Он опирается на привязки PyGObject к богатому набору виджетов GTK3, и фактически все эти виджеты может использоваться в приложении GladeVCP, а не только в специализированных виджетах для взаимодействия с HAL и LinuxCNC, которые описаны здесь.

#### 12.3.1.1 Краткий обзор PyVCP и GladeVCP

Оба поддерживают создание панелей с *HAL widgets* — элементами пользовательского интерфейса, такими как светодиоды, кнопки, ползунки и т. д., значения которых связаны с контактом HAL, который, в свою очередь, взаимодействует с остальной частью LinuxCNC.

**PyVCP:**

- Набор виджетов: использует виджеты TkInter.
- Создание пользовательского интерфейса: цикл ”редактировать XML-файл/выполнить результат/оце внешний вид”.
- Нет поддержки внедрения пользовательской обработки событий.
- Никакого взаимодействия LinuxCNC за пределами контактов ввода-вывода HAL не поддерживается.

**GladeVCP:**

- Набор виджетов: зависит от набора виджетов GTK3.
- Создание пользовательского интерфейса: используется Glade редактор пользовательского интерфей WYSIWYG.
- Любое изменение контакта HAL может быть направлено на обратный вызов определяемого пользователем обработчика событий Python.
- Любой сигнал GTK (нажатие клавиши/кнопки, окно, ввод-вывод, таймер, сетевые события) может быть связан с определяемыми пользователем обработчиками в Python.
- Прямое взаимодействие с LinuxCNC: выполнение произвольных команд, например, запуск команд MDI для вызова подпрограммы G-кода, а также поддержка операций изменения статуса с помощью виджетов действий.
- Несколько независимых панелей GladeVCP можно запускать на разных вкладках.
- Разделение внешнего вида и функциональности пользовательского интерфейса: меняйте внешний вид, не трогая код.

### 12.3.2 Краткий тур с примером панели

Окна панели GladeVCP можно запускать в трех различных конфигурациях:

- всегда видимый интегрированный в AXIS с правой стороны, точно так же как панели PyVCP,
- как вкладка в AXIS,Touchy, Gscreen или GMOCCAPY; в AXIS это создаст третью вкладку помимо вкладок Preview и DRO, которые необходимо вызвать явно,
- как отдельное окно верхнего уровня, которое можно иконизировать/деиконифицировать независимо от главного окна.

**Установленный LinuxCNC** Если вы используете установленную версию LinuxCNC, примеры, показанные ниже, находятся в configuration picker в ветке *Sample Configurations* > *apps* > *Glade-VCP*.

**Git Checkout** Следующие инструкции применимы только в том случае, если вы используете git checkout. Откройте терминал и перейдите в каталог, созданный git, затем введите команды, как показано.

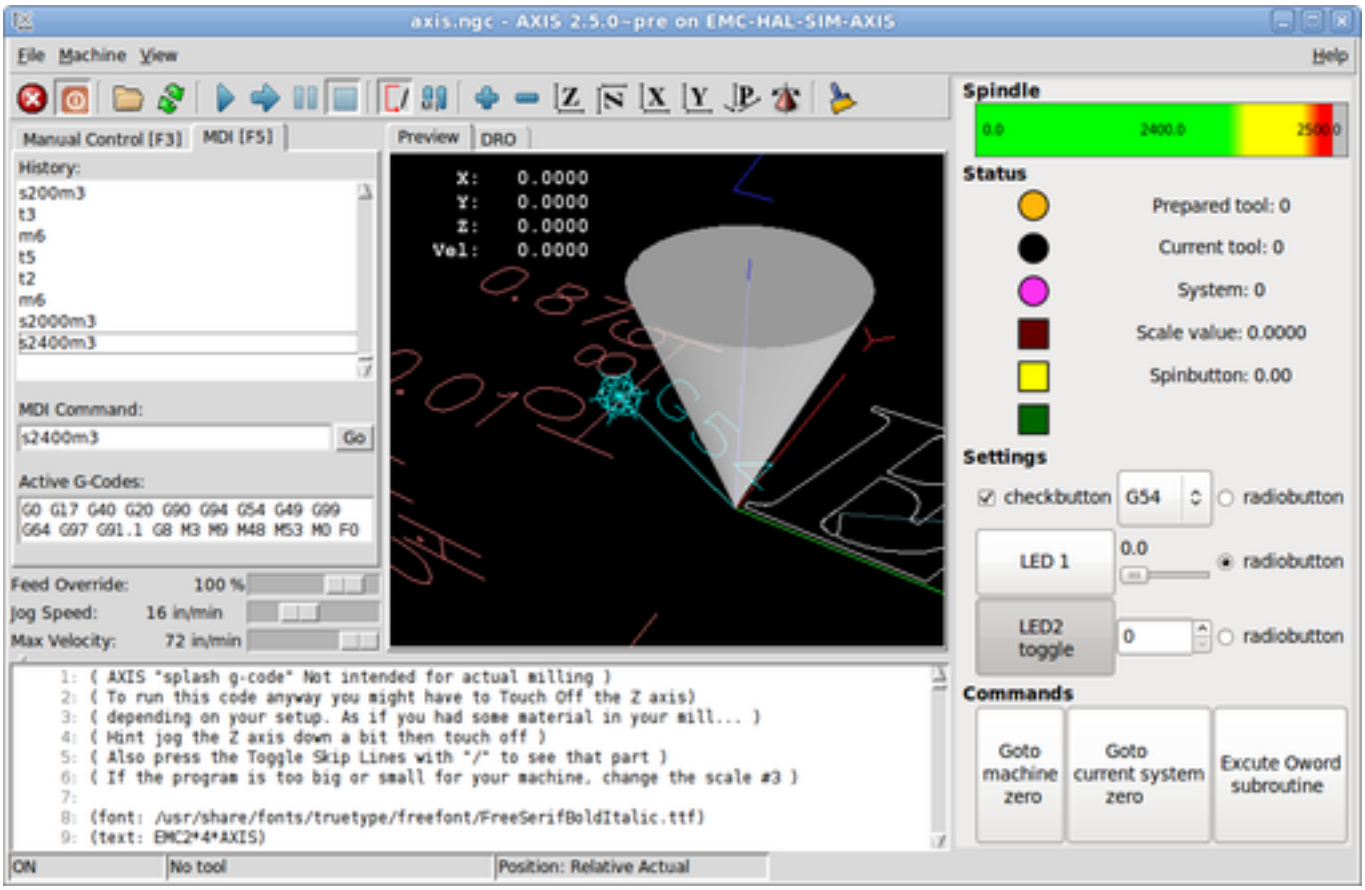
---

**Note**
Чтобы следующие команды работали с вашей git checkout, вы должны сначала запустить *make*, затем *sudo make setuid* then run', а затем запустить . *./scripts/rip-environment*. Дополнительную информацию о проверке git можно найти на вики-странице LinuxCNC.

---

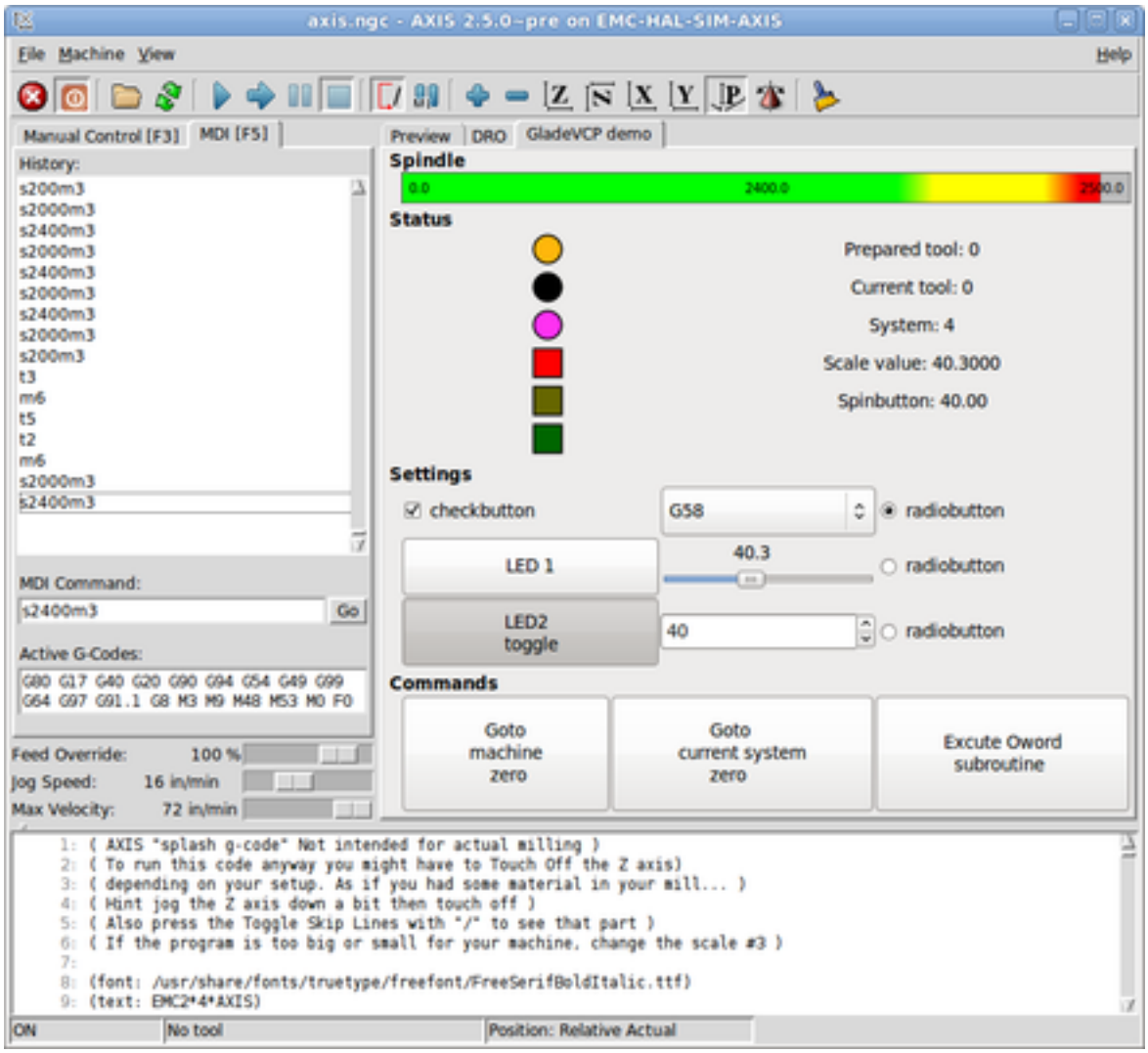
Запустите образец панели GladeVCP, интегрированной в AXIS, например PyVCP, следующим образом:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



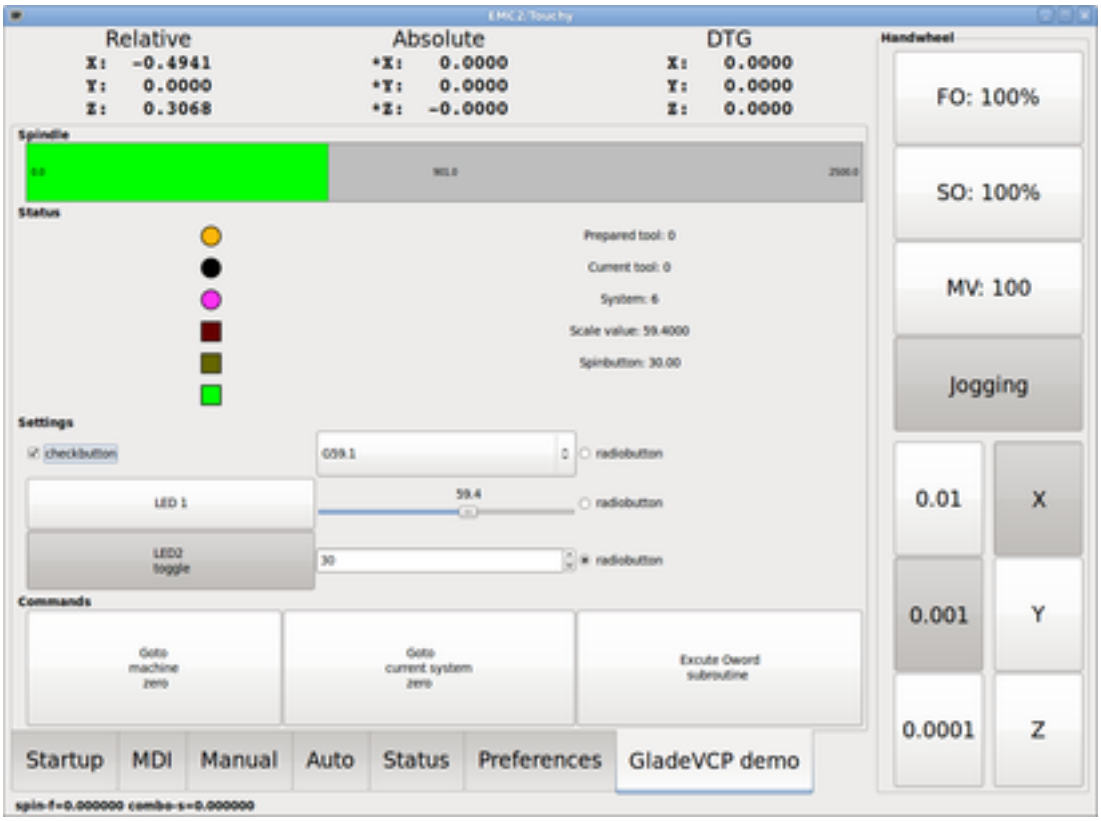

Запустите ту же панель, но как вкладку внутри AXIS:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_tab.ini
```




Чтобы запустить эту панель внутри *Touchy*:

```
$ cd configs/sim/touchy/gladevcp
$ linuxcnc gladevcp_touchy.ini
```




Функционально эти настройки идентичны — они отличаются только требованиями к площади экрана и видимости. Поскольку можно запускать несколько компонентов GladeVCP параллельно (с разными именами компонентов HAL), возможны и смешанные настройки — например, панель с правой стороны и одна или несколько вкладок для менее часто используемых частей интерфейса.

#### 12.3.2.1 Изучение примера панели

При запуске configs/sim/axis/gladevcp_panel.ini или configs/sim/axis/gladevcp_tab.ini изучите *Show HAL Configuration* — вы найдете компонент HAL `gladevcp` и сможете наблюдать значения их контактов при взаимодействии с виджетами в панель. Настройки HAL можно найти в файле *configs/axis/gladevcp/manual-example.hal*.

Панель примера имеет две рамки внизу. Панель настроена таким образом, что сброс ESTOP активирует рамку *Settings*, а включение машины активирует рамку *Commands* внизу. Виджеты HAL в рамке *Settings* связаны со светодиодами и метками в рамке *Status*, а также с текущим и подготовленным номером инструмента — поиграйте с ними, чтобы увидеть эффект. Выполнение команд *T<toolnumber>* и *M6* в окне MDI изменит поля текущего и подготовленного номера инструмента.

Кнопки в рамке *Commands* представляют собой *MDI Action widgets* — нажатие на них приведет к выполнению команды MDI в интерпретаторе. Третья кнопка *Execute Oword subroutine* представляет собой расширенный пример — она берет несколько значений контактов HAL из поля *Settings* и передает их в качестве параметров в подпрограмму Oword. Фактические параметры, полученные подпрограммой, отображаются с помощью команд *(DEBUG, )* — тело подпрограммы см. в *../../nc_files/ov*

Чтобы узнать, как панель интегрирована в AXIS, см. оператор *[DISPLAY]GLADEVCP* в configs/sim/axis/gladevcp/gladevcp_panel.ini, оператор *[DISPLAY]EMBED** в configs/sim/axis/gladevcp/gladevcp_tab.ini и *[HAL]POSTGUI_HALFILE* как в configs/sim/axis/gladevcp/gladevcp_tab.ini, так и в configs/sim/axis/gladevcp/gladevcp_panel.ini.

#### 12.3.2.2 Изучение описания пользовательского интерфейса

Пользовательский интерфейс создан с помощью редактора Glade UI — для его изучения вам необходимо его установить Glade installed. Чтобы отредактировать пользовательский интерфейс, выполните команду

```
$ glade configs/axis/gladevcp/manual-example.ui
```

Требуемая программа Glade может называться Glade-Gtk2 в более поздних системах.

В центральном окне показан внешний вид пользовательского интерфейса. Все объекты пользовательс интерфейса и объекты поддержки находятся в правом верхнем окне, где вы можете выбрать конкретный виджет (или кликнув на нем центральном окне). Свойства выбранного виджета отображаются и могут быть изменены в правом нижнем окне.

Чтобы увидеть, как команды MDI передаются из виджетов MDI Action, изучите виджеты, перечисленны в разделе *Actions* в правом верхнем углу окна, а в правом нижнем окне на вкладке *General* выберите свойство *MDI command*.

#### 12.3.2.3 Изучение обратного вызова Python

Посмотрите, как обратный вызов Python интегрирован в пример:

- В Glade см. виджет метки `hits` (простой виджет GTK+).
- В виджете `button1` перейдите на вкладку *Signals* и найдите сигнал *pressed*, связанный с обработчико *on_button_press*.
- В hitcounter.py посмотрите метод *on_button_press* и посмотрите, как он устанавливает свойство label в объекте *hits*.

Это лишь затрагивает концепцию — механизм обратного вызова будет рассмотрен более подробно в разделе GladeVCP Programming.

### 12.3.3 Создание и интеграция пользовательского интерфейса Glade

#### 12.3.3.1 Обязательное условие: установка Glade

Для просмотра или изменения файлов пользовательского интерфейса Glade вам необходимо установить Glade 3.38.2 или более позднюю версию — она не нужна только для запуска панели GladeVCP. Если команда *glade* отсутствует, установите ее командой:

```
$ sudo apt install glade
```

Затем проверьте установленную версию, которая должна быть равна или выше 3.6.7:

```
$ glade --version
```

Glade содержит внутренний интерпретатор Python, поддерживается только Python 3. Это справедливо для Debian Bullseye, Ubuntu 21 и Mint 21 или более поздних версий. Старые версии не будут работать, вы получите ошибку Python.

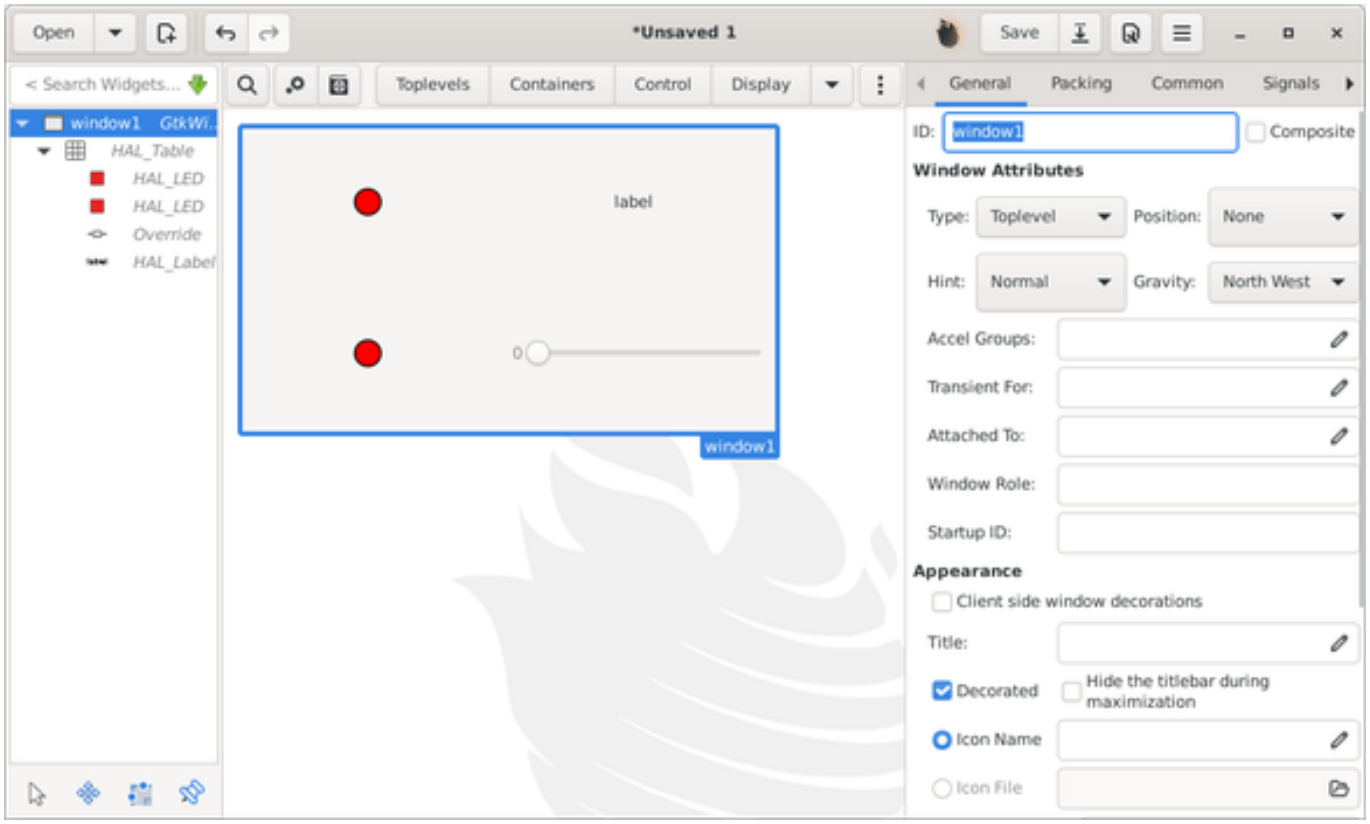
#### 12.3.3.2 Запуск Glade для создания нового пользовательского интерфейса

В этом разделе лишь описаны начальные шаги, специфичные для LinuxCNC. Дополнительную информацию и руководство по Glade можно найти на сайте http://glade.gnome.org. Некоторые советы и рекомендации Glade также можно найти на YouTube.

Либо измените существующий компонент пользовательского интерфейса, запустив `glade <file>.ui`, либо запустите новый, просто запустив команду `glade` из оболочки.

- Если LinuxCNC не был установлен из пакета, среду оболочки LinuxCNC необходимо настроить с помощью `source <linuxcncdir>/scripts/rip-environment`, иначе Glade не найдет виджеты, специфичные для LinuxCNC.
- Когда вас спросят о несохраненных настройках, просто примите значения по умолчанию и нажмите *Close*.
- В *Toplevels* (панель инструментов) выберите *GtkWindow* (первая запись) в качестве окна верхнего уровня. Установите *window1* в качестве идентификатора на правой панели на вкладке *General*. Это именование важно, поскольку GladeVCP полагается на него.
- С помощью кнопки с тремя точками вы можете найти виджеты, специфичные для LinuxCNC.
- Добавьте в фрейм контейнер, например HAL_Box или HAL_Table из *HAL Python*.
- Выберите и поместите некоторые элементы, такие как светодиод, кнопка и т. д., в контейнер.

Это будет выглядеть так:




Glade имеет тенденцию писать много сообщений в окно оболочки, которые в большинстве случаев можно игнорировать. Выберите *File→Save as*, дайте ему имя, например *myui.ui*, и убедитесь, что он сохранен как файл *GtkBuilder* (переключатель в левом нижнем углу диалогового окна «Save »). GladeVCP также корректно обрабатывает старый формат *libglade*, но использовать его нет смысла. Соглашение о расширении файла GtkBuilder — *.ui*.

#### 12.3.3.3 Тестирование панели

Теперь вы готовы попробовать (пока работает LinuxCNC, например, AXIS) с помощью:

```
gladevcp myui.ui
```

GladeVCP создает компонент HAL, названный так же, как базовое имя файла пользовательского интерфейса - в данном случае *myui* - если он не переопределен опцией `-c <component name>`. Если вы используете AXIS, просто попробуйте *Show HAL configuration* и проверьте его контакты.

Вы можете задаться вопросом, почему виджеты, содержащие *HAL_Hbox* или *HAL_Table*, отображаются серым цветом (неактивными). Контейнеры HAL имеют связанный с ними контакт HAL, который по умолчанию отключен, что приводит к тому, что все содержащиеся в нем виджеты становятся неактивными. Обычным вариантом использования было бы связывание этих контактов HAL контейнера с сигналом `halui.machine.is-on` или одним из сигналов `halui.mode`, чтобы гарантировать, что некоторые виджеты будут активными только в определенном состоянии.

Чтобы просто активировать контейнер, выполните команду HAL `setp Gladevcp.<имя-контейнера> 1`.

#### 12.3.3.4 Подготовка командного файла HAL

Предлагаемый способ связывания выводов HAL на панели GladeVCP — собрать их в отдельный файл с расширением *.hal*. Этот файл передается с помощью опции `POSTGUI_HALFILE=` в разделе `HAL` вашего INI-файла.

**Caution**
Не добавляйте командный файл GladeVCP HAL в раздел AXIS `[HAL]HALFILE=` ini, это не даст желаемого эффекта — см. следующие разделы.

#### 12.3.3.5 Интеграция в AXIS, например PyVCP

Поместите панель GladeVCP на правую боковую панель, указав в INI-файле следующее:

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -u ./hitcounter.py ./manual-example.ui

[HAL]
# HAL commands for GladeVCP components in a tab must be executed via POSTGUI_HALFILE
POSTGUI_HALFILE =  ./manual-example.hal

[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Имя компонента HAL по умолчанию для приложения GladeVCP, запускаемого с опцией GLADEVCP: `gladevcp`.

Командная строка, фактически запускаемая AXIS в приведенной выше конфигурации, выглядит следующим образом:

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./hitcounter.py ./manual- ↩
    example.ui
```

Вы можете добавить сюда произвольные параметры `gladevcp`, если они не конфликтуют с указанными выше параметрами командной строки.

Можно создать собственное имя компонента HAL, добавив параметр `-c`:

```
[DISPLAY]
# add GladeVCP panel where PyVCP used to live:
GLADEVCP= -c example -u ./hitcounter.py ./manual-example.ui
```

Командная строка, которую фактически запускает AXIS для вышеприведенного:

```
halcmd loadusr -Wn example gladevcp -c example -x {XID} -u ./hitcounter.py ./manual-example ↩
    .ui
```

---

**Note**
Спецификаторы файлов, такие как ./hitcounter.py, ./manual-example.ui и т. д., указывают, что файлы расположены в том же каталоге, что и файл INI. Возможно, вам придется скопировать их в свой каталог (в качестве альтернативы укажите правильный абсолютный или относительный путь к файлу(ам)).

---

---

**Note**
Параметр `[RS274NGC]SUBROUTINE_PATH=` установлен только для того, чтобы панель примера нашла подпрограмму Oword (oword.ngc) для виджета MDI Command. Возможно, в вашей настройке это не понадобится. Спецификатор относительного пути ../../nc_files/gladevcp_lib создан для работы с каталогами, скопированными средством выбора конфигурации, а также при использовании установки запуска на месте.

---

#### 12.3.3.6 Встраивание в виде вкладки

Для этого отредактируйте свой INI-файл и добавьте в разделы DISPLAY и HAL INI-файла следующим образом:

```
[DISPLAY]
# add GladeVCP panel as a tab next to Preview/DRO:
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 commands for GladeVCP components in a tab must be executed via POSTGUI_HALFILE
POSTGUI_HALFILE =  ./gladevcp/manual-example.hal

[RS274NGC]
# gladevcp Demo specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Обратите внимание на способ запуска команды вкладки `halcmd loadusr` — это гарантирует, что *POSTGUI_HALFILE* будет запущен только после того, как компонент HAL будет готов. В редких случаях вы можете запустить здесь команду, которая использует вкладку, но не имеет связанного компонента HAL. Такую команду можно запустить без *halcmd loadusr*, и это означает для AXIS, что ей не нужно ждать компонента HAL, поскольку его нет.

При изменении имени компонента в приведенном выше примере обратите внимание, что имена, используемые в `-Wn <component>` и `-c <component>`, должны быть идентичными.

Попробуйте это, запустив AXIS — рядом с вкладкой DRO должна появиться новая вкладка под названием *GladeVCP demo*. Выберите эту вкладку, и вы увидите, что пример панели хорошо вписывается в AXIS.

---

**Note**
Убедитесь, что файл пользовательского интерфейса является последним параметром, передаваемым GladeVCP в операторах GLADEVCP= и EMBED_TAB_COMMAND=.

#### 12.3.3.7 Интеграция в Touchy

Чтобы добавить вкладку GladeVCP в *Touchy*, отредактируйте свой INI-файл следующим образом:

```
[DISPLAY]
# add GladeVCP panel as a tab
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 specific Oword subs live here
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

**Note**
Спецификаторы файлов, такие как ./hitcounter.py, ./manual-example.ui и т. д., указывают, что файлы расположены в том же каталоге, что и файл INI. Возможно, вам придется скопировать их в свой каталог (в качестве альтернативы укажите правильный абсолютный или относительный путь к файлу(ам)).

Обратите внимание на следующие отличия от настройки вкладки AXIS:

- Командный файл HAL немного изменен, поскольку *Touchy* не использует компоненты *halui*, поэтому его сигналы недоступны, а некоторые ярлыки были использованы.
- Параметр INI *POSTGUI_HALFILE*= отсутствует, но передача командного файла HAL в строке *EMBED_TAB_COMMAND*= допускается.
- Заклинание *halcmd loaduser -Wn ...* не требуется.

#### 12.3.3.8 Loading Builtin Panels

There are builtin panels available on the system, you load them in a slightly different way.
You do not add any filename extension to the panel name.
If the panel requires a user file (-u option) it will automatically be loaded.
Builtin panel names can be shown by running the gladevcp command alone in a terminal.

Loading the builtin verser probe panel:

```
gladevcp gtk_verser_probe
```

Embedding is the same, no filename extension, but other options are fine:

```
[DISPLAY]
# add GladeVCP panel as a tab next to Preview/DRO:
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID}  ↩
    gtk_verser_probe
```

### 12.3.4 Параметры командной строки GladeVCP

These are the GladeVCP command line options:
(See also `man gladevcp`.)
If you enter gladevcp in a terminal this is what you will see:

```
b''Ив''b''cb''b''пв''b''ob''b''лв''b''ьв''b''зв''b''ob''b''вв''b''ab''b''нв''b''ив''b' ↩
    'eb'': gladevcp [options] myfile.ui
b''ив''b''cb''b''пв''b''ob''b''лв''b''ьв''b''зв''b''ob''b''вв''b''ab''b''нв''b''ив''b' ↩
    'eb'': gladevcp [options] built_in_panel_name

Options:
  -h, --help      show this help message and exit
  -c NAME         Set component name to NAME. Default is basename of UI file
  -d              Enable debug output
  -g GEOMETRY     Set geometry WIDTHxHEIGHT+XOFFSET+YOFFSET. Values are in
                  pixel units, XOFFSET/YOFFSET is referenced from top left of
                  screen use -g WIDTHxHEIGHT for just setting size or -g
                  +XOFFSET+YOFFSET for just position
  -H FILE         execute hal statements from FILE with halcmd after the
                  component is set up and ready
  -i              Enable info output
  -m MAXIMUM      Force panel window to maxumize
  -q              Enable only error debug output
  -r GTK_RC       read custom GTK rc file to set widget style
  -t THEME        Set gtk theme. Default is system theme
  -x XID          Reparent gladevcp into an existing window XID instead of
                  creating a new top level window
  --xid           reparent window into a plug add push the plug xid number to
                  standardout
  -u FILE         Use FILEs as additional user defined modules with handlers
  -U USEROPT      pass USEROPTs to Python modules
  -v              Enable verbose debug output
  --always_above  Request the window To always be above other windows
  --ini=INI_PATH  ini path

[GladeVCP-][CRITICAL]  Available built-in VCP Panels: (gladevcp:205)
['gtk_verser_probe', 'gtk_little_probe']
```

### 12.3.5 Понимание процесса запуска GladeVCP

Шаги интеграции, описанные выше, кажутся немного сложными, и это так. Таким образом, это помогает понять процесс запуска LinuxCNC и то, как он связан с GladeVCP.

Обычный процесс запуска LinuxCNC выполняет следующее:

- Запускается среда реального времени.
- Загружаются все компоненты HAL.
- Компоненты HAL связаны друг с другом посредством сценариев .hal cmd.
- `task`, `iocontrol` и, в конечном итоге, запускается пользовательский интерфейс.
- До GladeVCP предполагалось, что к моменту запуска пользовательского интерфейса весь HAL загружен, подключен и готов к работе.

Внедрение GladeVCP привело к следующей проблеме:

- Панели GladeVCP должны быть встроены в основную настройку окна графического интерфейса.
- Панели GladeVCP должны быть встроены в основные настройки окна графического пользовательско интерфейса, например, AXIS или Touchy, Gscreen или GMOCCAPY (встроенное окно или в виде встроенной вкладки).
- Для этого необходимо, чтобы главный ГИП был запущен до того, как окно GladeVCP можно будет подключить к главному ГИП.
- Однако GladeVCP также является компонентом HAL и создает собственные контакты HAL.
- Как следствие, все компоненты HAL, связанные с контактами HAL GladeVCP в качестве источника или приемника, должны запускаться **после** настройки ГИП.

Это задача `POSTGUI_HALFILE`. Эта опция INI проверяется ГИП. Если ГИП обнаруживает эту опцию, он запускает соответствующий файл HAL после настройки любой встроенной панели GladeVCP. Однако, он не проверяет, действительно ли используется панель GladeVCP, и в этом случае cmd-файл HAL запускается нормально. Поэтому, если вы НЕ запускаете GladeVCP через `GLADEVCP` или `EMBED_TAB` и т. д., но позже в отдельном окне оболочки или через какой-либо другой механизм, командный файл HAL в `POSTGUI_HALFILE` будет выполнен слишком рано. Предполагая, что здесь упоминаются контакты GladeVCP, это завершится неудачно с сообщением об ошибке, указывающим, что компонент GladeVCP HAL недоступен.

Итак, если вы запускаете GladeVCP из отдельного окна оболочки (т. е. не запускается встроенным ГИП):

- Вы не можете полагаться на параметр INI `POSTGUI_HALFILE`, вызывающий запуск команд HAL *в нужный момент времени*, поэтому закомментируйте это в INI-файле.
- Явно передайте командный файл HAL, который ссылается на контакты GladeVCP, в GladeVCP с опцией `-H <halcmd file>` (см. предыдущий раздел).

### 12.3.6 Ссылка на виджет HAL

GladeVCP включает в себя коллекцию виджетов Gtk с прикрепленными контактами HAL, называемыми виджетами HAL, предназначенными для управления, отображения или иного взаимодействия со слоем HAL LinuxCNC. Они предназначены для использования с редактором пользовательского интерфейса Glade. При правильной установке виджеты HAL должны появиться в группе виджетов Glade *HAL Python*. Многие поля, специфичные для HAL, в разделе *General* Glade имеют соответствующ всплывающую подсказку при наведении курсора мыши.

Сигналы HAL бывают двух вариантов: биты и числа. Биты — это сигналы выключения/включения. Числа могут быть типа ”float”, ”s32” или ”u32”. Дополнительную информацию о типах данных HAL см. в HAL manual. Виджеты GladeVCP могут либо отображать значение сигнала с помощью виджета индикатора, либо изменять значение сигнала с помощью виджета управления. Таким образом, существует четыре класса виджетов GladeVCP, которые можно подключить к сигналу HAL. Другой класс вспомогательных виджетов позволяет вам организовывать и маркировать панель.

- Виджеты для индикации ”битовых” сигналов: HAL_LED
- Виджеты для управления ”битовыми” сигналами: HAL_Button HAL_RadioButton HAL_CheckButton
- Виджеты для индикации ”числовых” сигналов: HAL_Label, HAL_ProgressBar, HAL_HBar and HAL_VBar, HAL_Meter
- Виджеты для управления ”числовыми” сигналами: HAL_SpinButton, HAL_HScale и HAL_VScale, Jog Wheel, Speed Control
- Виджеты управления чувствительностью : State_Sensitive_Table HAL_Table и HAL_HBox

- Предварительный просмотр пути инструмента: HAL_Gremlin
- Виджеты для отображения положения осей: DRO Widget, Combi DRO Widget
- Виджеты для работы с файлами: IconView File Selection
- Виджеты для отображения/редактирования смещений всех осей: OffsetPage
- Виджеты для отображения/редактирования всех коррекций инструмента: Tooloffset editor
- Виджет для отображения и редактирования G-кода: HAL_Sourceview
- Виджет для ввода MDI и отображения истории: MDI History

#### 12.3.6.1 Именование виджетов и контактов HAL

Большинство виджетов HAL имеют один связанный контакт HAL с тем же именем HAL, что и виджет (glade: General→Name).

Исключениями из этого правила на данный момент являются:

- *HAL_Spinbutton* и *HAL_ComboBox*, которые имеют два контакта: `<widgetname>-f` (float) и `<widgetnam` (s32)
- *HAL_ProgressBar*, который имеет входной контакт `<widgetname>-value` и входной контакт `<widgetna`

#### 12.3.6.2 Атрибуты Python и методы HAL Widgets

Виджеты HAL являются экземплярами GtKWidgets и, следовательно, наследуют методы, свойства и сигналы применимого класса GtkWidget. Например, чтобы выяснить, какие методы, свойства и сигналы, связанные с GtkWidget, имеет *HAL_Button*, найдите описание https://lazka.github.io/pgi-docs/Gtk-3.0/classes/Button.html. Gtk.Button[GtkButton] в Справочник по API PyGObject.

Простой способ узнать отношение наследования данного виджета HAL заключается в следующем: запустите Glade, поместите виджет в окно и выберите его; затем выберите вкладку *Signals* в окне *Properties*. Например, если выбрать виджет «HAL_LED», это покажет, что *HAL_LED* является производным от *GtkWidget*, который, в свою очередь, является производным от *GtkObject* и, в конечном итоге, от *GObject*.

Полную иерархию классов можно увидеть, вызвав GtkInspector в графическом интерфейсе Glade, выбрав виджет и нажав Control-Shift-I. Если Инспектор не открывается, он может быть разрешен из терминала, введя:

```
gsettings set org.gtk.Settings.Debug enable-inspector-keybinding true
```

Инспектор также удобен для тестирования изменений стиля CSS ”на лету”, а также для определения всех свойств и сигналов, доступных для виджета.

Виджеты HAL также имеют несколько атрибутов Python, специфичных для HAL:

**hal_pin**
: Базовый объект Python контакта HAL в случае, если виджет имеет один тип контакта

**hal_pin_s, hal_pin_f**
: Контакты s32 и float виджетов *HAL_Spinbutton* и *HAL_ComboBox* — обратите внимание, что эти виджеты не имеют атрибута *hal_pin*!

**hal_pin_scale**
: Входной контакт с плавающей запятой виджета *HAL_ProgressBar*, представляющий максимально абсолютное значение ввода.

Существует несколько методов HAL Widgets, специфичных для HAL, но единственный подходящий метод:

**<halpin>.get()**
Получите значение текущего контакта HAL, где <*halpin*> — это соответствующее имя контакта HAL, указанное выше.

#### 12.3.6.3 Установка значений контактов и виджетов

Как правило, если вам нужно установить значение выходного виджета HAL из кода Python, сделайте это, вызвав базовый *setter* Gtk (например, `set_active()`, `set_value()`). Не пытайтесь установить значение связанного контакта напрямую с помощью `halcomp[pinname] = value`, потому что виджет не заметит это изменение!

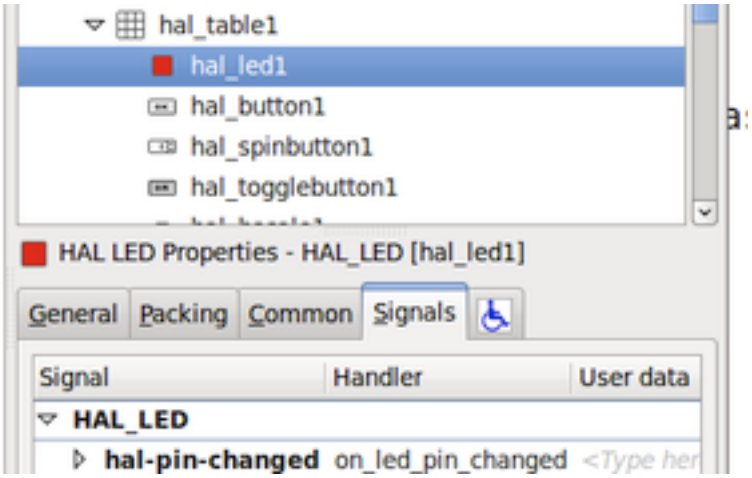
Может возникнуть соблазн программно *установить входные контакты виджета HAL*. Обратите внимание, что это в первую очередь противоречит назначению входного контакта — он должен быть связан с сигналами, генерируемыми другими компонентами HAL, и реагировать на них. Хотя в настоящее время в HAL Python нет защиты от записи на входные контакты, это не имеет смысла. Однако вы можете использовать `setp _pinname_ _value_` в связанном файле HAL для тестирования.

Совершенно нормально установить значение выходного контакта HAL с помощью `halcomp[pinname] = value` при условии, что этот контакт HAL не связан с виджетом, то есть был создан `hal_glib.GPin(ha` (пример см. в GladeVCP Programming).

#### 12.3.6.4 Сигнал hal-pin-changed

Программирование, управляемое событиями, означает, что пользовательский интерфейс сообщает вашему коду, когда ”что-то происходит” — посредством обратного вызова, например, при нажатии кнопки. Выходные виджеты HAL (те, которые отображают значение контакта HAL), такие как LED, Bar, VBar, Meter и т. д., поддерживают сигнал `hal-pin-changed`, который может вызвать обратный вызов вашего кода Python, когда - ну, контакт HAL меняет свое значение. Это означает, что больше нет необходимости постоянно опрашивать изменения контактов HAL в вашем коде: виджеты делают это в фоновом режиме и сообщают вам об этом.

Вот пример установки сигнала `hal-pin-changed` для HAL_LED в редакторе Glade UI:




Пример в `configs/apps/gladevcp/complex` показывает, как это обрабатывается в Python.

#### 12.3.6.5 Кнопки

Эта группа виджетов является производной от различных кнопок Gtk и состоит из виджетов HAL_Button, HAL_ToggleButton, HAL_RadioButton и CheckButton. Все они имеют один выходной BIT контакт, имя которого идентично имени виджета. Кнопки не имеют дополнительных свойств по сравнению с их базовыми классами Gtk.

- HAL_Button: мгновенное действие, не сохраняет состояние. Важный сигнал: `pressed`
- HAL_ToggleButton, HAL_CheckButton: сохраняет состояние вкл/выкл. Важный сигнал: `toggled`
- HAL_RadioButton: группа одна из многих. Важный сигнал: `toggled` (для каждой кнопки).
- Важные общие методы: `set_active()`, `get_active()`
- Важные свойства: `label`, `image`

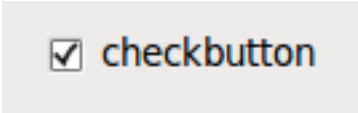



Figure 12.31: Кнопка независимой фиксации

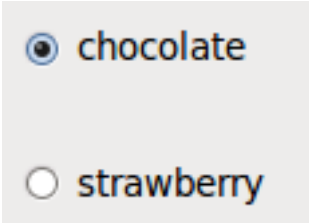



Figure 12.32: Кнопки зависимой фиксации

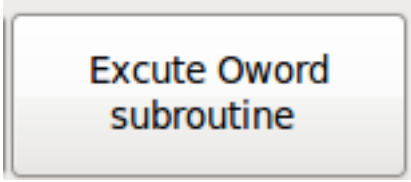



Figure 12.33: Кнопка-переключатель

---

**Tip**
Определение группы кнопок зависимой фиксации в Glade:

- Выберите активную кнопку по умолчанию.
- В разделе *General→Group* другой кнопки выберите имя активной кнопки по умолчанию в диалоговом окне *Choose a Radio Button in this project*.

---

См. `configs/apps/gladevcp/by-widget/` для приложений GladeVCP и файла пользовательского интерфейса для работы с кнопками зависимой фиксации.

#### 12.3.6.6 Шкалы

HAL_HScale и HAL_VScale являются производными от GtkHScale и GtkVScale соответственно.

**<widgetname>**
выходной FLOAT контакт

**<widgetname>-s**
выходной s32 контакт

Чтобы сделать шкалу полезной в Glade, добавьте *Adjustment* (General → Adjustment → New or existing adjustment) и отредактируйте объект настройки. Он определяет значения default/min/-max/increment. Кроме того, установите настройки *Page size* и *Page increment* на ноль, чтобы избежать предупреждений.

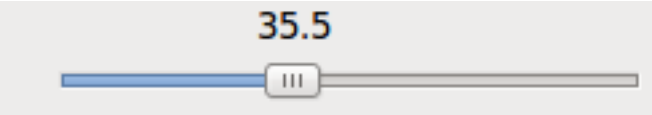



Figure 12.34: Пример HAL_HScale

#### 12.3.6.7 Окошко счетчика

HAL SpinButton является производным от GtkSpinButton и содержит два контакта:

**<widgetname>-f**
выходной FLOAT контакт

**<widgetname>-s**
выходной s32 контакт

Чтобы быть полезными, Spinbuttons необходимо значение настройки, такое как шкала, см. выше.

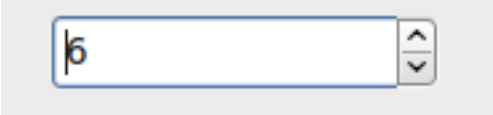



Figure 12.35: Пример SpinButton

#### 12.3.6.8 Hal_Dial

Виджет hal_dial имитирует маховичок или регулировочный циферблат.
Им можно управлять с помощью мыши. Вы можете просто использовать колесо мыши, пока курсор мыши находится над виджетом Hal_Dial, или удерживать левую кнопку мыши и перемещать курсор по кругу, чтобы увеличить или уменьшить отсчеты.
Двойным щелчком левой или правой кнопки масштабный коэффициент можно увеличить или уменьшить.

- Против часовой стрелки = уменьшить отсчеты
- По часовой стрелке = увеличить отсчеты

- Колесико вверх = увеличить отсчеты
- Колесико вниз = уменьшить отсчеты
- Двойной щелчок левой кнопкой мыши = масштаб x10
- Двойной щелчок правой кнопкой мыши = масштаб /10

`hal_dial` экспортирует свое значение отсчетов в виде контактов HAL:

**<widgetname>**
выходной s32 контакт

**<widgetname>-scaled**
выходной FLOAT контакт

**<widgetname>-delta-scaled**
выходной FLOAT контакт

`hal_dial` имеет следующие свойства:

**cpr**
Устанавливает количество отсчетов на оборот, допустимые значения находятся в диапазоне от 25 до 360.
по умолчанию = 100

**show_counts**
Установите в False, если вы хотите скрыть отображение отсчетов в середине виджета.
по умолчанию = True

**label**
Установите содержимое метки, которая может отображаться поверх значения отсчетов. Если длина метки превышает 15 символов, она будет сокращена до 15 символов.
по умолчанию = пусто

**center_color**
Это позволяет изменить цвет колеса. Он использует цветовую строку GDK.
по умолчанию = #bdefbdefbdef (серый)

**count_type_shown**
Доступны три вида отсчетов: 0) Необработанные отсчеты CPR 1) Масштабированные отсчеты 2) Дельта-масштабированные отсчеты.
по умолчанию = 1

- отсчет основан на выбранном CPR – он будет считаться положительным и отрицательным. Он доступен в виде контакта s32.
- Масштабированный отсчет – это отсчет CPR , умноженный на масштаб – он может быть положительным и отрицательным.
  Если вы измените масштаб, выходные данные немедленно отразят это изменение. Он доступен в виде контакта FLOAT.
- Дельта-масштабированный-отсчет — это ИЗМЕНЕНИЕ количества CPR, с коэффициентом масштаба.
  Если вы измените масштаб, только отсчеты после этого изменения будут масштабированы, а затем добавлены к текущему значению.
  Он доступен в виде контакта FLOAT.

**scale_adjustable**
Установите для этого параметра значение False, если вы хотите запретить изменение масштаба двойным щелчком по виджету.
Если это значение неверно, коэффициент масштаба не будет отображаться в виджете.
по умолчанию = True

**scale**
Установите это для масштабирования отсчетов.
по умолчанию = 1.0

Существуют способы прямого управления виджетом с помощью Python.

Использование goobject для установки перечисленных выше свойств:

```
[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)
```

Существуют методы Python:

- `[widget name].get_value()`
  Вернет значение счетчика как целое число s32
- `[widget name].get_scaled_value()`
  Вернет отсчеты в виде числа с плавающей запятой
- `[widget name].get_delta_scaled_value()`
  Вернет отсчеты в виде числа с плавающей запятой
- `[widget name].set_label(”string”)`
  Устанавливает содержимое метки с помощью ”string”

Выдается два сигнала GObject:

- `count_changed`
  Генерируется, когда изменяются отсчеты виджета, например, от прокрутки колеса.
- `scale_changed`
  Генерируется, когда изменяется масштаб виджета, например, от двойного щелчка.

Подключитесь к ним следующим образом:

```
[widget name].connect(’count_changed’, [count function name])
[widget name].connect(’scale_changed’, [scale function name])
```

Функции обратного вызова будут использовать этот шаблон:

```
def [count function name](widget, count,scale,delta_scale):
```

Это вернет: widget, current count, scale и delta scale этого виджета.

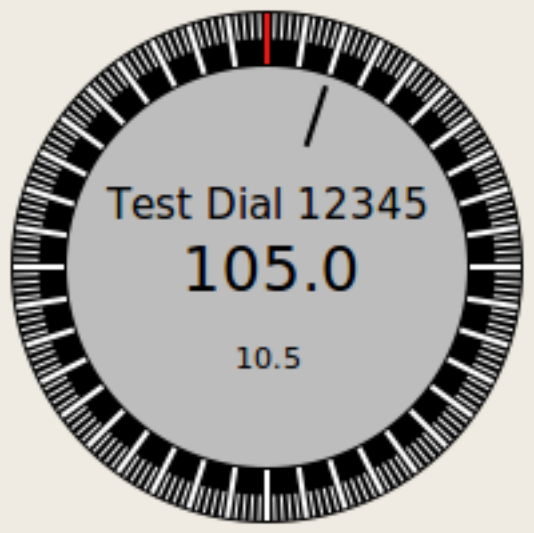



Figure 12.36: Пример Hal_Dial

#### 12.3.6.9 Маховичок подачи

Виджет `jogwheel` имитирует настоящий маховичок. Им можно управлять с помощью мыши. Вы можете просто использовать колесо мыши, пока курсор мыши находится над виджетом Jog-Wheel, или нажать левую кнопку мыши и перемещать курсор по кругу, чтобы увеличить или уменьшить отсчеты.

- Против часовой стрелки = уменьшить отсчеты
- По часовой стрелке = увеличить отсчеты
- Колесико вверх = увеличить отсчеты
- Колесико вниз = уменьшить отсчеты

Поскольку перемещение мыши при перетаскивании может происходить быстрее, чем виджет может обновиться, вы можете потерять отсчеты и крутя слишком быстро. Рекомендуется использовать колесико мыши и только при очень грубых изменениях — способ перетаскивания.

`jogwheel` экспортирует свое значение счетчика как контакт HAL:

**<widgetname>-s**
выходной s32 контакт

`jogwheel` имеет следующие свойства:

**size**
Устанавливает размер виджета в пикселях, допустимые значения находятся в диапазоне от 100 до 500, по умолчанию = 200

**cpr**
Устанавливает количество отсчетов на оборот, допустимые значения находятся в диапазоне от 25 до 100, по умолчанию = 40

**show_counts**
Установите в False, если вы хотите скрыть отображение отсчетов в середине виджета.

**label**
Установите содержимое метки, которая может отображаться поверх значения отсчетов. Цель состоит в том, чтобы дать пользователю представление об использовании этого маховичка. Если длина метки превышает 12 символов, она будет сокращена до 12 символов.

Существует несколько способов прямого управления виджетом с помощью Python.

Использование GObject для установки перечисленных выше свойств:

```
[widget name].set_property(”size”,int(value))
[widget name].set_property(”cpr”,int(value))
[widget name].set_property(”show_counts, True)
```

Существует два метода Python:

- `[widget name].get_value()`
  Вернет значение отсчетов как целое число
- `[widget name].set_label(”string”)`
  Устанавливает содержимое метки с помощью ”string”

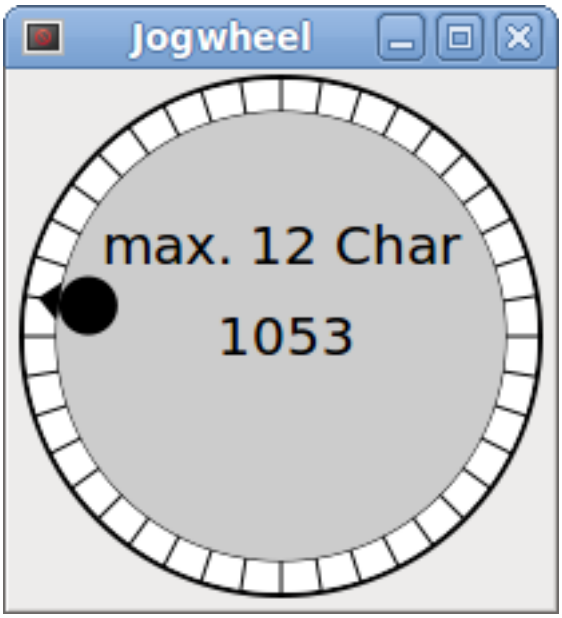



Figure 12.37: Пример JogWheel

#### 12.3.6.10 Контроль скорости

`speedcontrol` — это виджет, специально созданный для управления регулировкой с помощью сенсорного экрана. Это замена виджета обычного масштаба, который сложно перемещать по сенсорному экрану.

Значение контролируется двумя кнопками для увеличения или уменьшения значения. Приращение будет меняться, пока нажата кнопка. Значение каждого приращения, а также время между двумя изменениями можно установить с помощью свойств виджета.

`speedcontrol` предлагает некоторые контакты HAL:

**<widgetname>-value**
выходной float контакт
Отображаемое значение виджета.

**<widgetname>-scaled-value**
выходной float контакт
Отображаемое значение делится на значение масштаба. Это очень полезно, если скорость отображается в единицах/мин, но LinuxCNC ожидает, что она будет в единицах/секунду.

**<widgetname>-scale**
входной float контакт
Применяемый масштаб.
По умолчанию — 60.

**<widgetname>-increase**
входной bit контакт
Пока контакт true, значение будет увеличиваться.
Очень удобно с подключенным переключателем мгновенного действия.

**<widgetname>-decrease**
входной bit контакт
Пока контакт true, значение будет уменьшаться.
Очень удобно с подключенным переключателем мгновенного действия.

`speedcontrol` имеет следующие свойства:

**height**
Integer
Высота виджета в пикселях.
Допустимые значения: от 24 до 96.
По умолчанию — 36.

**value**
Float
Начальное значение для установки.
Допустимые значения находятся в диапазоне от 0,001 до 99999,0.
По умолчанию — 10.0.

**min**
Float
Минимально допустимое значение.
Допустимые значения: от 0,0 до 99999,0.
По умолчанию — 0,0.
Если вы измените это значение, приращение будет сброшено до значения по умолчанию, поэтому может потребоваться впоследствии установить новое приращение.

**max**
Float
Максимально допустимое значение.
Допустимые значения: от 0,001 до 99999,0.
По умолчанию — 100,0.
Если вы измените это значение, приращение будет сброшено до значения по умолчанию, поэтому может потребоваться впоследствии установить новое приращение.

**increment**
Float
Устанавливает применяемое приращение за клик мышки.
Допустимые значения: от 0,001 до 99999,0 и -1.
По умолчанию установлено значение -1, что приводит к увеличению значения на 100 от минимального до максимального.

**inc_speed**
Integer
Устанавливает задержку таймера для увеличения скорости, удерживая нажатыми кнопки.
Допустимые значения: от 20 до 300.
По умолчанию — 100.

**unit**
String
Устанавливает единицу измерения, отображаемую на панели после значения.
Допускается любая строка.
По умолчанию — ””.

**color**
Color
Устанавливает цвет полосы.
Допускается любой шестнадцатеричный цвет.
По умолчанию — ”#FF8116”.

**template**
String
Текстовый шаблон для отображения значения. Используется форматирование Python.
Любой разрешенный формат.
По умолчанию – ”%.1f”.

**do_hide_button**
Boolean
Показывать или скрывать увеличение кнопки уменьшения.
True или False.
По умолчанию = False.

Существует несколько способов прямого управления виджетом с помощью Python.

Использование GObject для установки перечисленных выше свойств:

```
[widget name].set_property(”do_hide_button”,bool(value))
[widget name].set_property(”color”,”#FF00FF”)
[widget name].set_property(”unit”, ”mm/min”)
etc.
```

Существуют также методы Python для изменения виджета:

```
[widget name].set_adjustment(gtk-adjustment)
```

Вы можете назначить существующую настройку элементу управления, чтобы можно было легко заменить существующие ползунки без большого количества изменений кода. Имейте в виду, что после изменения настройки вам может потребоваться установить новый шаг, поскольку он будет сброшен до значения по умолчанию (100 шагов от MIN до MAX):

- `[widget name].get_value()`
  Вернет значение отсчетов как число с плавающей запятой
- `[widget name].set_value(float(value))`
  Устанавливает виджет на заданное значение
- `[widget name].set_digits(int(value))`
  Устанавливает цифры значения, которое будет использоваться
- `[widget name].hide_button(bool(value))`
  Скрыть или показать кнопку

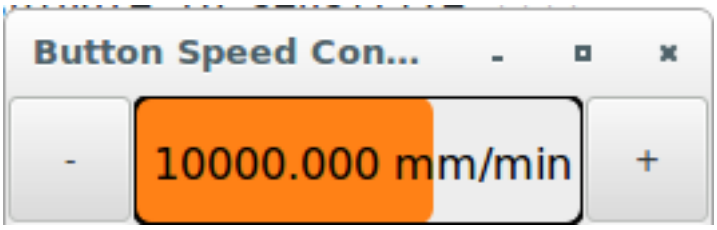



Figure 12.38: Пример Speedcontrol

#### 12.3.6.11 Label

`hal_label` — это простой виджет, основанный на GtkLabel, который представляет значение контакта HAL в определяемом пользователем формате.

**label_pin_type**
Тип HAL вывода (0:s32, 1:float, 2:u32), см. также всплывающую подсказку *General→HAL pin type* (обратите внимание, что это отличается от PyVCP, который имеет три виджета меток, по одному для каждого типа).

**text_template**
Определяет отображаемый текст - строку формата Python для преобразования значения контакта в текст. По умолчанию `%s` (значения преобразуются функцией str()), но может содержать любой допустимый аргумент в качестве аргумента метода Python format().
Пример: `Distance: %.03f` отобразит текст и значение контакта с 3 дробными цифрами, дополненными нулями для контакта FLOAT.

#### 12.3.6.12 Контейнеры

- HAL_HideTable
- HAL_Table
- State_Sensitive_Table
- HAL_HBox (устарело)

Эти контейнеры предназначены для десенсибилизации (серого цвета) или сокрытия их детей. Нечувствительные потомки не будут реагировать на вводимые данные.

**HAL_HideTable**
Имеет один входной контакт HAL BIT, который контролирует, скрыты ли его дочерние виджеты или нет.

**Pin: , <Panel_basename>.<widgetname>**
in bit pin
Если контакт имеет низкий логический уровень, то видны дочерние виджеты, что является состоянием по умолчанию.

**HAL_Table и HAL_Hbox**
Имеют один входной контакт HAL BIT, который контролирует, являются ли их дочерние виджеты чувствительными или нет.

**Pin: , <Panel_basename>.<widgetname>**
in bit pin
Если на контакте низкий логический уровень, дочерние виджеты неактивны, что является состоянием по умолчанию.

**State_Sensitive_Table**
Отвечает на состояние интерпретатора LinuxCNC.
При необходимости можно выбрать ответ на *must-be-all-homed*, *must-be-on* and *must-be-idle*.
Вы можете объединить их. В Estop всегда будет нечувствительно.
(Не имеет контакта).

---

**Warning**
**HAL_Hbox устарел — используйте HAL_Table.**
Если текущие панели используют его, он не подведет. Вы просто больше не найдете его в редакторе GLADE.
В будущих версиях GladeVCP этот виджет может быть полностью удален, и тогда вам придется обновить панель.

---

**Tip**
Если вы обнаружите, что какая-то часть вашего приложения GladeVCP *grayed out* (нечувствительна), посмотрите, не сброшен ли или не подключен контакт HAL_Table.

---

#### 12.3.6.13 LED

`hal_led` имитирует настоящий светодиодный индикатор.
Он имеет один входной контакт BIT, который контролирует его состояние: ВКЛ или ВЫКЛ.

СИДы имеют несколько свойств, которые определяют их внешний вид:

**on_color**
Строка, определяющая цвет ВКЛ состояния СИД.
Может быть любым допустимым именем gdk.Color.
Не работает на Ubuntu 8.04.

**off_color**
Строка, определяющая цвет ВЫКЛ состояния СИД.
Может быть любым допустимым именем gdk.Color или специальным значением `dark` . `dark` означает, что цвет ВЫКЛ будет установлен на значение 0,4 цвета ВКЛ.
Не работает на Ubuntu 8.04.

**pick_color_on, pick_color_off**
Цвета для состояний ВКЛ и ВЫКЛ.
Они могут быть представлены как строки `#RRRRGGGGBBBB` и являются необязательными свойствами, имеющими приоритет над `on_color` и `off_color`.

**led_size**
Радиус СИД (для квадратного - половина стороны СИД)

**led_shape**
Форма СИД.
Допустимые значения: 0 для круглой формы, 1 для овала и 2 для квадратной формы.

**led_blink_rate**
Если установлено и СИД в состоянии ВКЛ, тогда он мигает.
Период мигания равен значению ”led_blink_rate”, указанному в миллисекундах.

**create_hal_pin**
Выбор/отмена создания контакта HAL для управления СИД.
Без созданного контакта HAL СИД'ом можно управлять с помощью функции Python.

В качестве виджета ввода LED также поддерживает сигнал `hal-pin-changed`. Если вы хотите получать в своем коде уведомление при изменении контакт HAL LED, подключите этот сигнал к обработчику, например `on_led_pin_changed`, и укажите обработчик следующим образом:

```
def on_led_pin_changed(self,hal_led,data=None):
    print("on_led_pin_changed() - HAL pin value:",hal_led.hal_pin.get())
```

Он будет вызываться при любом фронте сигнала, а также во время запуска программы, чтобы сообщить текущее значение.

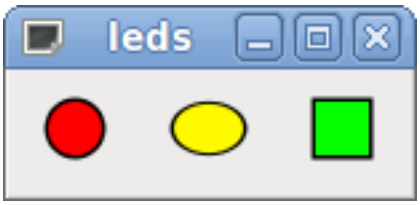



Figure 12.39: Пример СИД'ов

#### 12.3.6.14 ProgressBar

---

**Note**
Этот виджет может исчезнуть.
Вместо этого используйте виджеты HAL_HBar и HAL_VBar.

---

`HAL_ProgressBar` является производным от gtk.ProgressBar и имеет два входных float контакта HAL:

**<widgetname>**
текущее значение, которое будет отображаться

**<widgetname>-scale**
максимальное абсолютное значение ввода

`HAL_ProgressBar` имеет следующие свойства:

**scale**
Шкала значений.
Устанавливает максимальное абсолютное значение входа. То же, что и установка контакта <widgetname>.scale.
Число с плавающей запятой, диапазон от $-2^{24}$ до $+2^{24}$.

**green_limit**
Нижний предел зеленой зоны

**yellow_limit**
Нижний предел желтой зоны

**red_limit**
Нижний предел красной зоны

**text_template**
Текстовый шаблон для отображения текущего значения контакта `<widgetname>`.
Форматирование Python может использоваться для dict `{"value":value}`.

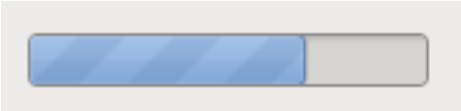


Figure 12.40: Пример HAL_ProgressBar

#### 12.3.6.15 ComboBox

`HAL_ComboBox` является производным от gtk.ComboBox. Позволяет выбрать значение из раскрывающег списка.

`HAL_ComboBox` экспортирует два контакта HAL:

**<widgetname>-f**
Текущее значение, тип FLOAT

**<widgetname>-s**
Текущее значение, тип s32

`HAL_ComboBox` имеет следующее свойство, которое можно установить в Glade:

**column**
Индекс столбца.
Тип s32.
Допустимый диапазон: -1..100.
Значение по умолчанию -1.

В режиме по умолчанию этот виджет устанавливает контакты в индекс выбранной записи списка. Таким образом, если ваш виджет имеет три метки, он может принимать только значения 0,1 и 2.

В режиме столбца (column > -1) сообщаемое значение выбирается из массива ListStore, как определено в Glade. Поэтому обычно определение вашего виджета будет иметь два столбца в ListStore, один с текстом, отображаемым в раскрывающемся списке, и значение int или float, используемое для этого выбора.

В `configs/apps/by-widget/combobox.{py,ui}` есть пример, который использует режим столбца для выбора значения float из ListStore.

Если вы, как и я, не понимаете, как редактировать ComboBox ListStore и CellRenderer, см. https://youtu.
Z5_F-rW2cL8.

#### 12.3.6.16 Шкалы

Виджеты `HAL_Bar` и `HAL_VBar` для горизонтальных и вертикальных шкал, представляющих значения с плавающей запятой.

`HAL_Bar` и `HAL_VBar` имеют по одному входному конаткту FLOAT HAL.

`HAL_Bar` и `HAL_VBar` обе шкалы имеют следующие свойства:

**invert**
Меняет местами min и max направления.
Инвертированный HBar растет справа налево, инвертированный VBar — сверху вниз.

**min, max**
Минимальное и максимальное значение желаемого диапазона. Если текущее значение находится за пределами этого диапазона, это не является ошибкой.

**show limits**
Используется для выбора/отмены выбора ограничений текста на панели.

**zero**
Нулевая точка диапазона.
Если она находится в пределах min/max диапазона, полоса будет расти от этого значения, а не от левой (или правой) стороны виджета.
Полезно для представления значений, которые могут быть как положительными, так и отрицательными.

**force_width, force_height**
Принудительная ширина или высота виджета.
Если не установлено, размер будет определяться на основе поля или фиксированного размера виджета, а полоса заполнит всю область.

**text_template**
Как и в Label, устанавливает текстовый формат для min/max/current значений.
Может использоваться для отключения отображения значений.

**value**
Устанавливает отображение шклы в соответствии с введенным значением.
Используется только для тестирования в редакторе GLADE.
Значение будет установлено с контакта HAL.

**target value**
Устанавливает целевую линию на введенное значение.
Используется только для тестирования в редакторе GLADE.
Значение можно установить в функции Python.

**target_width**
Ширина линии, обозначающей целевое значение.

**bg_color**
Цвет фона (неактивного) шкалы.

**target_color**
Цвет целевой линии.

**z0_color, z1_color, z2_color**
Цвета зон разных значений.
По умолчанию: `green`, `yellow` and `red`.
Описание зон смотрите в свойствах `z*_border`.

**z0_border, z1_border**
Определите границы цветовых зон.
По умолчанию включена только одна зона. Если вам нужно больше одной зоны, установите для `z0_border` и `z1_border` нужные значения, чтобы зона 0 заполнялась от 0 до первой границы, зона 1 заполнялась от первой до второй границы, а зона 2 — от последней границы до 1.
Границы задаются в виде дробей.
Допустимые значения находятся в диапазоне от 0 до 1.

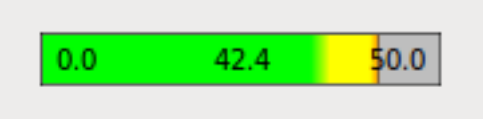



Figure 12.41: Горизонтальная шкала

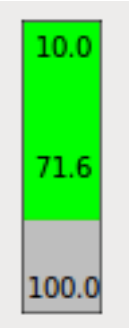



Figure 12.42: Вертикальная шкала

#### 12.3.6.17 Meter

`HAL_Meter` — это виджет, похожий на счетчик PyVCP, он представляет собой значение с плавающей запятой.

`HAL_Meter` имеет один входной FLOAT контакт HAL.

HAL Meter имеет следующие свойства:

**min, max**
: Минимальное и максимальное значение желаемого диапазона.
Если текущее значение находится за пределами этого диапазона, это не является ошибкой.

**force_size**
: Принудительный диаметр виджета.
Если не установлен, то размер будет определяться на основе поля или фиксированного размера виджета, а счетчик заполнит все доступное пространство с учетом соотношения сторон.

**text_template**
: Как и в Label, устанавливает текстовый формат для текущего значения.
Может использоваться для отключения отображения значений.

**label**
: Большая этикетка над центром циферблата.

**sublabel**
: Маленькая этикетка под центром циферблата.

**bg_color**
: Цвет фона циферблата.

**z0_color, z1_color, z2_color**
: Цвета зон разных значений.
По умолчанию: `green`, `yellow` and `red`.
Описание зон смотрите в свойствах `z*_border`.

**z0_border, z1_border**
Определите границы цветовых зон.
По умолчанию включена только одна зона. Если вам нужно более одной зоны, установите для `z0_border` и `z1_border` нужные значения, чтобы зона 0 заполнялась от минимальной до первой границы, зона 1 заполнялась от первой до второй границы, а зона 2 — от последней границы до максимальной.
Границы задаются как значения в диапазоне min-max.

Figure 12.43: Пример HAL Циферблатов

#### 12.3.6.18 HAL_Graph

Этот виджет предназначен для построения временного графика значений.

#### 12.3.6.19 Предварительный просмотр пути инструмента Gremlin для файлов NGC

Gremlin — это виджет предварительного просмотра сюжета, похожий на окно предварительного просмотра AXIS. Предполагается работающая среда LinuxCNC, такая как AXIS или Touchy. Чтобы подключиться к нему, проверяет переменную среды INI_FILE_NAME. Gremlin отображает текущий файл NGC — он отслеживает изменения и перезагружает файл ngc, если имя файла в AXIS/-Touchy меняется. Если вы запустите его в приложении GladeVCP, когда LinuxCNC не запущен, вы можете получить обратную трассировку, поскольку виджет Gremlin не может найти статус LinuxCNC, например текущее имя файла.

Gremlin не экспортирует контакты HAL.

Гремлин имеет следующие свойства:

**enable_dro**
Это отобразит УЦИ на графике.
По умолчанию = true.

**show_velocity**
Здесь отображается скорость инструмента.
По умолчанию = true.

**use_commanded**
Это выбирает режим УЦИ, который будет использоваться: заданные или фактические значения.
По умолчанию = true.

**metric_units**
Этот параметр выбирает режим УЦИ, который будет использоваться: метрические или имперские единицы.
По умолчанию = true.

**show_rapids**
Это говорит плоттеру показывать быстрые движения.
По умолчанию = true.

**show_dtg_**
Это выбирает режим УЦИ для отображения значения оставшегося пути.
По умолчанию = true.

**use_relative**
Это выбирает режим УЦИ для отображения значений относительно пользовательской системы или координат станка.
По умолчанию = true.

**show_live_plot**
Это говорит плоттеру, рисовать или нет.
По умолчанию = true.

**show_limits**
Это говорит плоттеру показать пределы станка.
По умолчанию = true.

**show_lathe_radius**
Выбирает режим УЦИ для отображения оси X в радиусе или диаметре, если в режиме токарного станка (выбирается в INI-файле с LATHE = 1).
По умолчанию = true.

**show_extents_option**
Это указывает плоттеру показать возможности станка.
По умолчанию = true.

**show_tool**
Это указывает плоттеру нарисовать инструмент.
По умолчанию = true.

**show_program**
Показывает программу G-кода.
По умолчанию = True

**use_joints_mode**
Используется со станками с non trivialkins кинематикой (например, роботах).
По умолчанию = false.

**grid_size**
Устанавливает размер сетки (виден только в видах X, Y и Z).
По умолчанию 0

**use_default_controls**
Это запрещает элементы управления мышью по умолчанию.
Это наиболее полезно при использовании сенсорного экрана, поскольку элементы управления по умолчанию работают некорректно. Вы можете программно добавлять элементы управления, используя Python и метод файла-обработчика.
По умолчанию = true.

**view**
Может быть любым из x, y, y2 , z, z2 , p (перспектива).
По умолчанию используется вид z.

**enable_dro**
Тип = boolean.
Отрисовывать УЦИ на выводе или нет.
По умолчанию = true.

**mouse_btn_mode**
Тип = integer.
Обработка кнопок мыши: приводит к различным функциям кнопки:

- 0 = по умолчанию: левая вращение, средняя перемещение, правая масштабирование
- 1 = левая масштабирование , средняя перемещение, правая вращение
- 2 = левая перемещение, средняя вращение, правая масштабирование
- 3 = левая масштабирование, средняя вращение, правая перемещение
- 4 = левая перемещение, средняя масштабирование, правая вращение
- 5 = левая вращение, средняя масштабирование, правая перемещение
- 6 = левая перемещение, средняя масштабирование, правая масштабирование

Режим 6 рекомендуется для плазменных и токарных станков, поскольку для таких станков вращение не требуется.

Существует несколько способов прямого управления виджетом с помощью Python.

Использование GObject для установки перечисленных выше свойств:

```
[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)
```

Существуют методы Python:

```
[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)
```

**Подсказки**

- Если вы установите для всех параметров построения графиков значение false, а для show_offsets значение true, вы получите страницу смещений вместо отрисовки графика.
- Если вы получите расстояние масштабирования перед изменением вида, а затем сбросите расстояние масштабирования, это будет намного удобнее для пользователя.
- если вы выберете элемент в предварительном просмотре, выбранный элемент будет использова в качестве центральной точки вращения

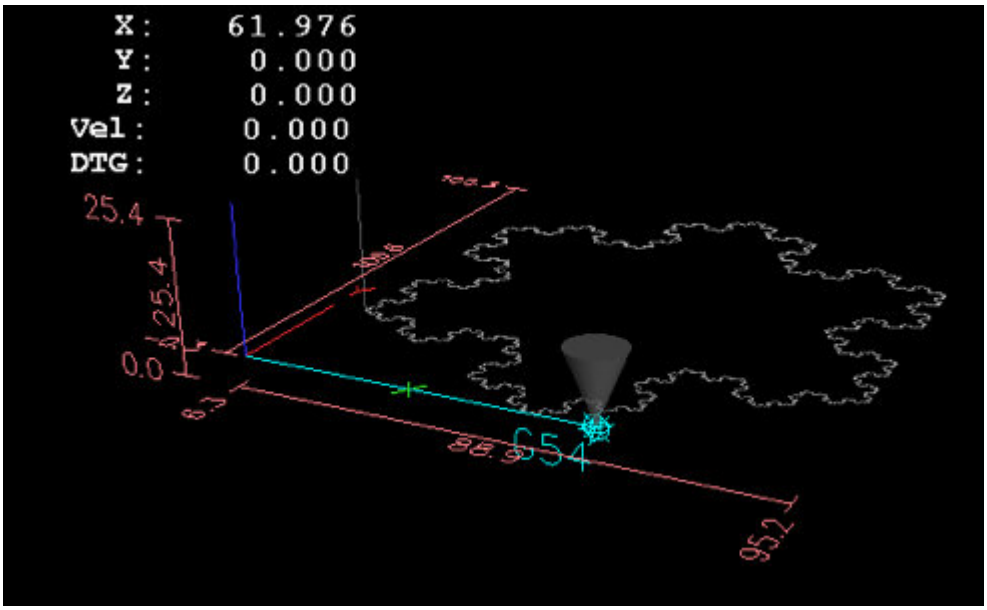



Figure 12.44: Пример Gremlin

#### 12.3.6.20 HAL_Offset

Виджет `HAL_Offset` используется для отображения смещения одной оси.

`HAL_Offset` имеет следующие свойства:

**display_units_mm**
Отображение в метрических единицах.

**joint_number**
Используется для выбора отображаемой оси (технически какое сочленение).
На станке с trivialkins (фрезерный, токарный) ось vs. номер сочленения имеют следующие комбинации:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

**mm_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью.

**imperial_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью.

**reference_type**

```
0:G5x 1:tool 2:G92 3:Rotation around Z
```

#### 12.3.6.21 Виджет УЦИ

Виджет УЦИ используется для отображения текущего положения оси.

Он имеет следующие свойства:

**display_units_mm**
Используется для переключения единиц отображения между метрическими и имперскими. По умолчанию — False.

**actual**
Выбирает фактическое положение (обратная связь) или заданное положение. По умолчанию — True.

**diameter**
Отображает диаметр для токарного станка. По умолчанию — False.

**mm_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью. По умолчанию — ”%10.3f”.

**imperial_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью. По умолчанию — ”%9.4f”.

**joint_number**
Используется для выбора отображаемой оси (технически какое сочленение). По умолчанию — 0.
На станке с кинематикой trivialkins (фрезерный, токарный) сочетание ось и номер сочленения следующие:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W +
```

**reference_type**

- 0 = `absolute` (исходное положение станка).
- 1 = `relative` (к началу координат текущего пользователя - G5x).
- 2 = `distance-to-go` (относительно начала координат текущего пользователя). По умолчанию — 0.

**font_family**
Укажите семейство шрифтов, например. mono. По умолчанию sans. Если шрифт не существует, будет использоваться текущий системный шрифт. По умолчанию — sans.

**font_size**
Укажите размер шрифта от 8 до 96. По умолчанию — 26.

**font_weight**
Укажите вес шрифта. Выберите lighter, normal, bold, or bolder. По умолчанию - bold.

**unhomed_color**
Цвет текста, когда вне исходной позиции, указан как цвет Gdk.RGBA. По умолчанию — красный, Gdk.RGBA(красный=1.000000, зеленый=0.000000, синий=0.000000, альфа=1.000000)

**homed_color**
Цвет текста когда в исходной позиции указан как цвет Gdk.RGBA. По умолчанию — зеленый, Gdk.RGBA(красный=0,000000, зеленый=0,501961, синий=0,000000, альфа=1,000000)

**Подсказки**

- Если вы хотите, чтобы изображение было выровнено по правому краю, установите для параметра Horizontal Alignment значение End.
- Фон виджета на самом деле прозрачный, поэтому, если вы поместите его поверх изображения, числа УЦИ будут отображаться поверх него без фона. Для этого существует специальная техника. См. анимированные функциональные схемы ниже.
- Виджет DRO — это модифицированный виджет меток GTK. Таким образом, многое из того, что можно сделать с меткой gtk, можно сделать и с виджетом DRO.
- Свойства шрифта также могут быть установлены из таблицы стилей CSS, которая имеет наивысший приоритет и переопределяет значения, установленные свойствами GObject.

Существует несколько способов прямого управления виджетом с помощью Python.

**Использование GObject для установки перечисленных выше свойств**

```
[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”))
```

**Использование таблицы стилей CSS для установки свойств шрифта**

Цвета могут быть указаны в одном из нескольких форматов, все они будут указывать один и тот же цвет: красный, *#ff0000, *rgb(255,0,0) или rgba(255,0,0,255).

Цвета могут упоминаться либо вместе:

```
.dro_unhomed {color: magenta}
.dro_homed {color: cyan}
```

или индивидуально по названию виджета:

```
#[widget name].dro_unhomed {color: magenta}
#[widget name].dro_homed {color: cyan}
```

На другие свойства стиля необходимо ссылаться по имени виджета:

```
#[widget name], #[widget name], #[widget name] {
    font-family: mono;
    font-size: 60px;
    font-weight: lighter;
}
```

**Существует два метода Python**

```
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

#### 12.3.6.22 Виджет Combi_DRO

Виджет Combi_DRO используется для отображения текущего положения, положения относительной оси и расстояния, которое необходимо пройти в одном УЦИ.
При нажатии на УЦИ порядок УЦИ будет переключаться.
В Relative Mode будет отображаться фактическая система координат.

`Combi_DRO` имеет следующие свойства:

**joint_number**
Используется для выбора отображаемой оси (технически какое сочленение).
На станке с кинематикой trivialkins (фрезерный, токарный) номера осей/сочленений следующие:

`0:X 1:Y 2:Z etc.`

**actual**
Выберите фактическое (обратная связь) или заданное положение.

**metric_units**
Используется для переключения между отображением метрических и имперских единиц.

**auto_units**
Единицы измерения будут переключаться между метрическими и имперскими единицами в зависимости от того, какой активный G-код — G20 или G21.
По умолчанию — TRUE.

**diameter**
Отображать положение в виде диаметра или радиуса.
В режиме диаметра УЦИ отобразит значение сочленения, умноженное на 2.

**mm_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью.
По умолчанию — ”%10.3f”.

**imperial_text_template**
Вы можете использовать форматирование Python для отображения позиции с разной точностью.
По умолчанию — ”%9.4f”.

**homed_color**
Цвет переднего плана чисел УЦИ, если сочленение находится в исходной позиции.
По умолчанию - зеленый.

**unhomed_color**
Цвет переднего плана чисел УЦИ, если сочленение не находится в исходной позиции.
По умолчанию красный.

**abs_color**
Цвет фона УЦИ, если основной УЦИ показывает абсолютные координаты.
По умолчанию синий.

**rel_color**
Цвет фона УЦИ, если основной УЦИ показывает относительные координаты.
По умолчанию черный.

**dtg_color**
Цвет фона УЦИ, если основной УЦИ показывает оставшееся расстояние.
По умолчанию желтый.

**font_size**
Размер шрифта больших цифр, у маленьких будет в 2,5 раза меньше.
Значение должно быть целым числом в диапазоне от 8 до 96.
По умолчанию 25.

**toggle_readout**
Щелчок левой кнопкой мыши переключает показания УЦИ в различные режимы [”Rel”, ”Abs”, ”DTG”].
Сняв флажок, вы можете запретить это поведение. Переключение по-прежнему можно выполнить с помощью `[widget name].toggle_readout()`.
Значение должно быть boolean.
По умолчанию — TRUE.

**cycle_time**
Время ожидания DRO между двумя опросами.
Эту настройку следует изменять только в том случае, если вы используете более 5 УЦИ одновременно, т. е. в конфигурации с 6 осями, чтобы избежать слишком сильного замедления основного приложения УЦИ.
Значение должно быть целым числом в диапазоне от 100мс до 1000мс.
По умолчанию — 150.

Использование GObject для установки перечисленных выше свойств:

```
[widget name].set_property(property, value)
```

Существует несколько методов Python для управления виджетом:

- `[widget name].set_to_inch(state)`
  Устанавливает УЦИ для отображения имперских единиц.
  `state` = boolean (True or False)
  По умолчанию FIXME.
- `[widget name].set_auto_units(state)`
  Если в True, УЦИ изменит единицы измерения в соответствии с активным G-кодом (G20 / G21).
  `state` = boolean (True or False)
  По умолчанию True.
- `[widget name].set_to_diameter(state)`
  Если True, УЦИ будет показывать диаметр, а не радиус, т. е. значение оси, умноженное на 2 (особенно необходимо для токарных станков).
  `state` = boolean (True or False)
  По умолчанию False.
- `[widget name].toggle_readout()`
  Переключает порядок УЦИ в виджете.
- `[widget name].change_axisletter(letter)`
  Изменяет автоматически заданную букву оси.
  Очень полезно изменить УЦИ токарного станка с *X* to *R* или ’D.
  `letter` = строка
- `[widget name].get_order()`
  Возвращает порядок УЦИ в виджете, который в основном используется для обеспечения их согласованности.
  Порядок также будет передан вместе с сигналом нажатия.
  Возвращает список, содержащий порядок.

- `[widget name].set_order(order)`
  Устанавливает порядок УЦИ, в основном используется для поддержания их согласованности.
  `order` = объект списка, должен быть одним из:
  - `["Rel", "Abs", "DTG"]` (по умолчанию)
  - `["DTG", "Rel", "Abs"]`
  - `["Abs", "DTG", "Rel"]`
- `[widget name].get_position()`
  Возвращает позицию УЦИ в виде списка чисел с плавающей запятой.
  Порядок не зависит от отображаемого на УЦИ порядка, и будет задаваться следующим образом: `[Absolute , relative , DTG]`.
  - `Absolute` = координаты станка , зависят от фактических свойств, дают фактическое или заданное положение.
  - `Relative` = будут координаты фактической системы координат.
  - `DTG` = расстояние, которое нужно пройти.
    Чаще всего будет 0, так как эту функцию не следует использовать во время движения станка из-за временных задержек..

Виджет будет выдавать следующие сигналы:

- `clicked`
  Этот сигнал подается, когда пользователь нажимает на виджет Combi_DRO.
  Он отправит следующие данные:
  - `widget` = объект виджета
    Объект виджета, который выдает сигнал.
  - `joint_number` = integer
    Номер сочленения УЦИ, где *0:X 1:Y 2:Z и т.д.*
  - `order` = объект списка
    Порядок УЦИ в этом виджете.
    Порядок можно использовать для установки других виджетов Combi_DRO в тот же порядок с помощью `[widget name].set_order(order)`.
- `units_changed`
  Этот сигнал выдается при смене единиц УЦИ.
  Он отправит следующие данные:
  - `widget` = объект виджета
    Объект виджета, который выдает сигнал.
  - `metric_units` = boolean
    True, если УЦИ отображает метрические единицы измерения, False в случае имперских..
- `system_changed`
  Этот сигнал выдается при смене единиц УЦИ.
  Он отправит следующие данные:
  - `widget` = объект виджета
    Объект виджета, который выдает сигнал.
  - `system` = string
    Реальная система координат. Будет один из G54 G55 G56 G57 G58 G59 G59.1 G59.2 G59.3 или Rel, если ничего не выбрано вообще, что произойдет только в Glade без запуска LinuxCNC..

С помощью команд можно получить некоторую информацию, которая может вас заинтересовать:

- `[widget name].system`
  Фактическая система, как указано в сигнале system_changed.
- `[widget name].homed`
  True , если сочленение в исходной позиции.
- `[widget name].machine_units`
  0, если имперская система мер, 1, если метрическая система мер.

Figure 12.45: Пример: три Combi_DRO в окне

```
X = Relative Mode
Y = Absolute Mode
Z = DTG Mode
```

#### 12.3.6.23 IconView (Выбор файла)

Это удобный виджет с сенсорным экраном для выбора файла и изменения каталогов.

`IconView` виджет имеет следующие свойства:

**icon_size**
Устанавливает размер отображаемого значка.
Допустимые значения — целые числа в диапазоне от 12 до 96.
По умолчанию — 48.

**start_dir**
Устанавливает каталог для запуска, когда виджет отобразится первый раз.
Должно быть строкой, содержащей действительный путь к каталогу.
По умолчанию — ”/”.

**jump_to_dir**
Устанавливает каталог ”перехода”, который выбирается соответствующей кнопкой в нижнем списке кнопок (5-я кнопка, считая слева).
Должно быть строкой, содержащей действительный путь к каталогу.
По умолчанию — ”\~”.

**filetypes**
Устанавливает файловый фильтр для отображаемых объектов.
Должно быть строкой, содержащей список отображаемых расширений, разделенных запятыми.
По умолчанию — ”ngc,py”.

**sortorder**
Устанавливает порядок сортировки отображаемого значка.
Должно быть целым числом от 0 до 3, где:

- 0 = ASCENDING (отсортировано по именам файлов)
- 1 = DESCENDING (отсортировано по именам файлов)
- 2 = FOLDERFIRST (сначала показывать папки, потом файлы), по умолчанию
- 3 = FILEFIRST (сначала показывать файлы, потом папки)

Использование GObject для установки перечисленных выше свойств:

```
[widget name].set_property(property,Value)
```

Существуют методы Python для управления виджетом:

- `[widget name].show_buttonbox(state)`
  Если в False, нижняя панель кнопок будет скрыта.
  Это полезно для пользовательских экранов со специальным расположением кнопок, позволяющим не изменять структуру ГИП. Хорошим примером является GMOCCAPY.
  `state` = boolean (True or False).
  По умолчанию — True.
- `[widget name].show_filelabel(state)`
  Если принимает значение True, будет отображаться метка файла (между окном IconView и нижней панелью кнопок).
  Скрытие этой метки может сэкономить место, но ее отображение очень полезно для отладки.
  `state` = boolean (True or False).
  По умолчанию — True.
- `[widget name].set_icon_size(iconsize)`
  Устанавливает размер иконки.
  Должно быть целое число в диапазоне от 12 до 96.
  По умолчанию = 48.
- `[widget name].set_directory(directory)`
  Позволяет установить каталог для отображения.
  `directory` = string (a valid file path).
- `[widget name].set_filetypes(filetypes)`
  Устанавливает используемый файловый фильтр.
  Будут показаны только файлы с указанными расширениями.
  `filetypes` = строка, содержащая список расширений, разделенных запятыми.
  По умолчанию = ”ngc,py”.
- `[widget name].get_selected()`
  Возвращает путь к выбранному файлу или «None», если был выбран каталог.
- `[widget name].refresh_filelist()`
  Обновляет список файлов.
  Нужен, если вы добавляете файл без изменения каталога.

Если блок кнопки скрыт, вы можете получить доступ к функциям этой кнопки через сигналы ее нажатия следующим образом:

```
[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")
```

Виджет будет выдавать следующие сигналы:

- `selected`
  Этот сигнал выдается, когда пользователь выбирает значок.
  Он вернет строку, содержащую путь к файлу, если был выбран файл или «None», если был выбран каталог.
- `sensitive`
  Этот сигнал подается, когда кнопки меняют свое состояние с чувствительного на нечувствительное или наоборот.
  Этот сигнал полезен для синхронизации окружающего ГИП с кнопкой виджета. См. пример GMOCCAPY.
  Он вернет **buttonname** и новое **state**:

  - `buttonname` одно из `btn_home`, `btn_dir_up`, `btn_sel_prev`, `btn_sel_next`, `btn_jump_to` или `btn_select`.
  - `state` является логическим значением и будет иметь значение True или False.
- `exit`
  Этот сигнал выдается при нажатии кнопки выхода для закрытия IconView.
  Чаще всего требуется, если приложение запускается автономно.

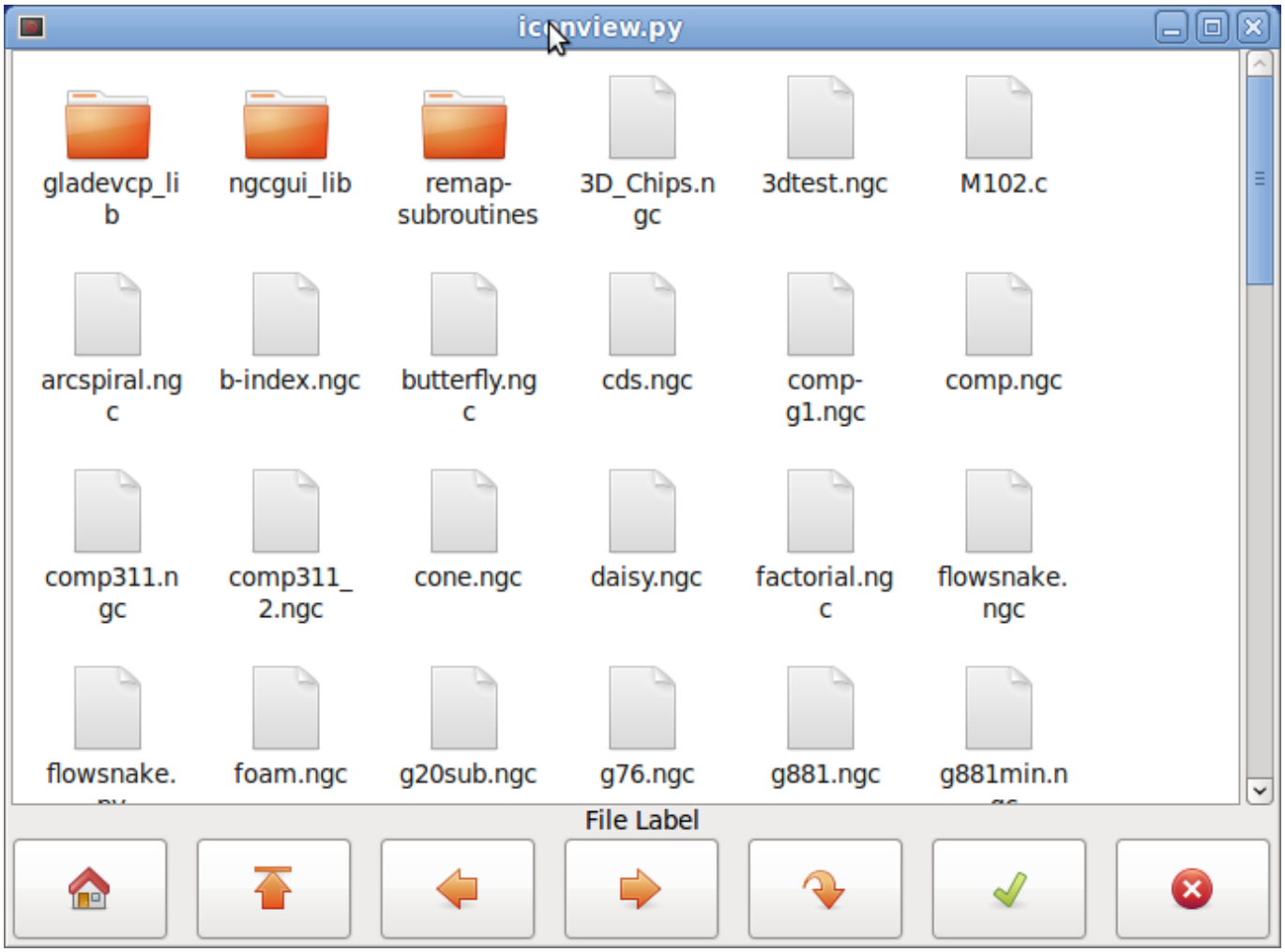



Figure 12.46: Пример Iconview

#### 12.3.6.24 Виджет калькулятора

Это простой виджет калькулятора, который можно использовать для числового ввода.
Вы можете предварительно настроить отображение и получить результат или заданное значение.

`calculator` имеет следующие свойства:

**is_editable**
: Это позволяет вводить ввод с клавиатуры.

**font**
: Это позволяет вам установить шрифт дисплея.

Существует несколько способов прямого управления виджетом с помощью Python.

Использование goobject для установки перечисленных выше свойств:

```
[widget name].set_property("is_editable",True)
[widget name].set_property("font","sans 25")
```

Существуют методы Python:

- `[widget name].set_value(2.5)`
  Предустанавливает дисплей и записывается.
- `[widget name].set_font(”sans 25”)`
- `[widget name].set_editable(True)`
- `[widget name].get_value()`
  Возвращает вычисленное значение — число с плавающей запятой.
- `[widget name].set_editable(True)`
- `[widget name].get_preset_value()`
  Возвращает записанное значение: число с плавающей запятой.

#### 12.3.6.25 Виджет редактора инструментов

Это виджет `редактор инструментов` для отображения и изменения файла инструмента.
В режиме токарного станка корректоры износа и коррекции инструмента будут отображаться отдельно.
Коррекции износа обозначаются номером инструмента выше 10000 (стиль Fanuc).
Он проверяет текущий файл раз в секунду, чтобы увидеть, обновил ли его LinuxCNC.

---

**Note**
LinuxCNC требует переназначения вызовов инструментов для фактического использования смещений износа.

---

`tooleditor` имеет следующие свойства:

**font**
Отобразить шрифт для использования

**hide_columns**
Это скроет данные столбцы.
Столбцы обозначаются (по порядку) следующим образом: `s,t,p,x,y,z,a,b,c,u,v,w,d,i,j,q`.
Вы можете скрыть любое количество столбцов, включая выделение и комментарии.

**lathe_display_type**
Показать формат токарного станка

Существует несколько способов прямого управления виджетом с помощью Python.

Использование goobject для установки перечисленных выше свойств:

```
[widget name].set_properties('hide_columns','uvwijq')
```

Это скроет столбцы uvwij и q и отобразит все остальные.

Существуют методы Python:

- `[widget name].set_visible(”ijq”,False)`
  Скрыло столбцы ij и Q, а остальные оставило как есть.
- `[widget name].set_filename(path_to_file)`
  Устанавливает и загружает файл инструмента.

- `[widget name].reload(None)`
  Перезагружает текущий файл инструмента.
- `[widget name].set_font('sans 16,tab='1')`
  Устанавливает шрифт (Pango) на вкладке, заголовке столбца и данных инструмента.
  `all_offsets`, `wear_offsets`, `tool_offsets` можно установить одновременно, добавив 1, 2 и/или 3 к строке табуляции.
  По умолчанию установлены все вкладки.
- `[widget name].set_title_font('sans 16,tab='1')`
  Устанавливает шрифт (Pango) только для заголовков столбцов.
  `all_offsets`, `wear_offsets`, `tool_offsets` можно установить одновременно, добавив 1, 2 и/или 3 к строке табуляции.
  По умолчанию установлены все вкладки.
- `[widget name].set_tab_font('sans 16,tab='1')`
  Устанавливает шрифт (Pango) только на вкладках.
  `all_offsets`, `wear_offsets`, `tool_offsets` можно установить одновременно, добавив 1, 2 и/или 3 к строке табуляции.
  По умолчанию установлены все вкладки.
- `[widget name].set_col_visible("abcUVW", False, tab='1')`
  Это скроет (False) столбцы abcuvw на вкладке 1 (all_offsets)
- `[widget name].set_lathe_display(value)`
  Скрывает или показывает вкладки износа и коррекции инструмента, используемые на токарных станках.
- `[widget name].get_toolinfo(toolnum)`
  Возвращает массив информации об инструменте с запрошенным номером инструмента или текущего инструмента, если номер инструмента не указан.
  Возвращает None, если инструмент не найден в таблице или текущий инструмент отсутствует.
- `[widget name].hide_buttonbox(self, True)`
  *Удобный* метод скрытия кнопок.
  Вы должны вызвать это после show_all().
- `[widget name].get_selected_tool()`
  Возвращает выбранный пользователем (выделенный) номер инструмента.
- `[widget name].set_selected_tool(toolnumber)`
  Выбирает (подсвечивает) требуемый инструмент.

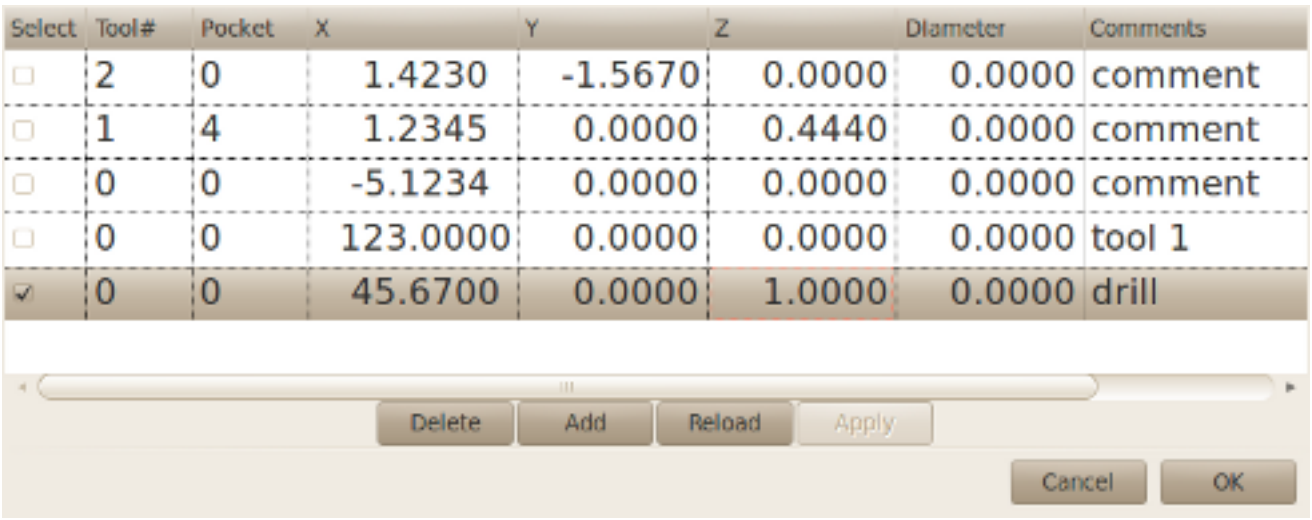



Figure 12.47: Пример Tooleditor

#### 12.3.6.26 Offsetpage

Виджет `Offsetpage` используется для отображения/редактирования смещений всех осей.
Он имеет удобные кнопки для обнуления смещений G92 и вращения вокруг оси Z.
Это позволит вам выбирать режим редактирования только тогда, когда станок включен и находится в режиме ожидания.
В это время вы можете напрямую редактировать смещения в таблице. Снимите флажок с кнопки редактирования, чтобы позволить OffsetPage отражать изменения.

Он имеет следующие свойства:

**display_units_mm**
: Отображение в метрических единицах

**hide_columns**
: Список столбцов без пробелов, которые нужно скрыть. Столбцы обозначаются (по порядку) следующим образом: `xyzabcuvwt`.
Вы можете скрыть любой из столбцов.

**hide_rows**
: Список строк без пробелов, которые нужно скрыть.
Строки обозначаются (по порядку) так: `0123456789abc`.
Вы можете скрыть любую из строк.

**font**
: Устанавливает тип и размер шрифта текста.

**highlight_color**
: При редактировании это цвет выделения.

**foreground_color**
: Когда `OffsetPage` обнаруживает активную пользовательскую систему координат, он будет использовать этот цвет для текста.

**mm_text_template**
: Вы можете использовать форматирование Python для отображения позиции с разной точностью.

**imperial_text_template**
: Вы можете использовать форматирование Python для отображения позиции с разной точностью.

Существует несколько способов прямого управления виджетом с помощью Python.

Использование goobject для установки перечисленных выше свойств:

```
[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”)
```

Существуют методы Python для управления виджетом:

- `[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”)`
- `[widget name].mark_active(”G55”)`
  Позволяет напрямую указать строку для выделения, например, если вы хотите использовать собственные элементы управления навигацией. См. главу GMOCCAPY.
- `[widget name].selection_mask = (”Tool”,”Rot”,”G5x”)`
  Эти строки НЕ доступны для выбора в режиме редактирования.
- `[widget name].set_names([[’G54’,’Default’],[”G55”,”Vice1”],[’Rot’,’Rotational’]])`
  Это позволяет вам установить текст столбца *T* каждой/любой строки.
  Это список пар offset-name/user-name.
  Текст по умолчанию тот же, что и имя смещения.
- `[widget name].get_names()`
  Это возвращает список пар row-keyword/user-name.
  Столбец имени пользователя доступен для редактирования, поэтому сохранение этого списка удобно для пользователя.
  См. `set_names` выше.

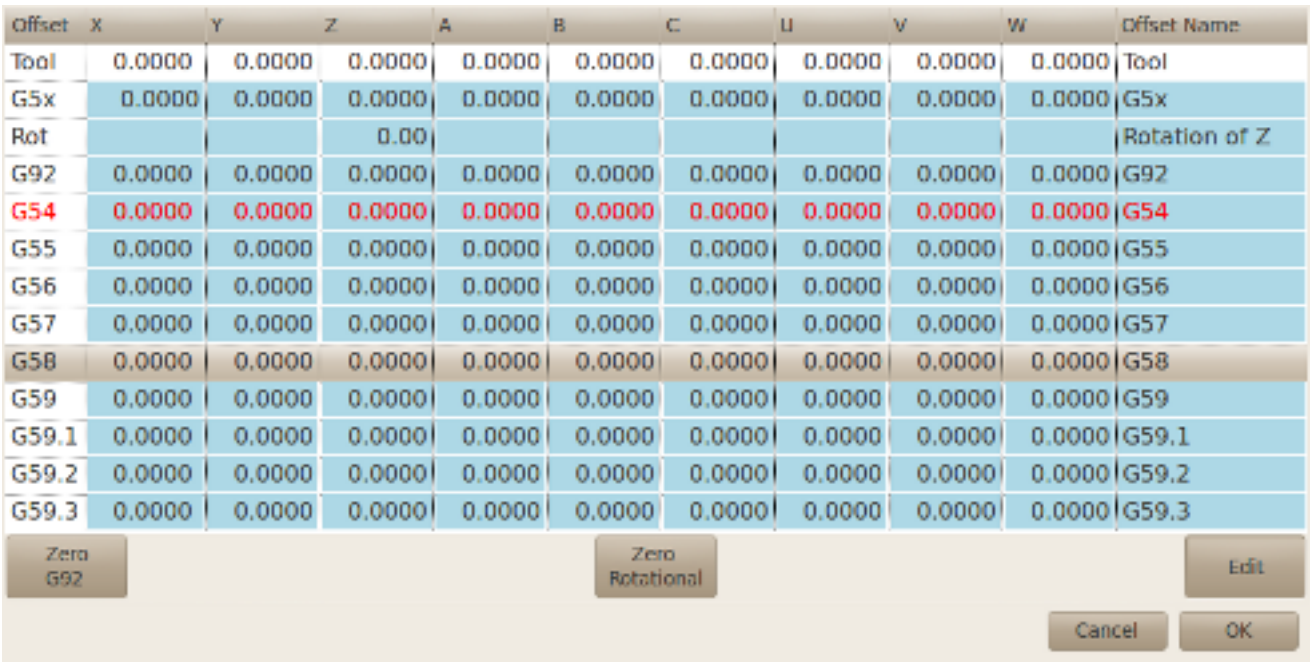



Figure 12.48: Пример Offsetpage

#### 12.3.6.27 Виджет HAL_sourceview

Это для отображения и простого редактирования G-кода. Он ищет спецификации подсветки `.ngc` в `~/share/gtksourceview-4/language-specs/`. Текущая бегущая строка будет выделена.

С помощью внешнего связующего кода Python он может:

- Поиск текста, отмена и повтор изменений.
- Используется для выбора строки программы.

Существуют методы Python для управления виджетом:

- `[widget name].redo()`
  Повторить один уровень изменений.
- `[widget name].undo()`
  Отменить один уровень изменений
- `[widget name].text_search(direction=True,mixed_case=True,text='G92')`
  Поиск вперед (направление = True) или назад,
  Поиск со смешанным регистром (mixed_case = True) или с точным совпадением
- `[widget name].set_line_number(linenumber)`
  Устанавливает линию для выделения.
  Использует номера строк исходного кода.
- `[widget name].get_line_number()`
  Возвращает текущую выделенную строку.
- `[widget name].line_up()`
  Перемещает выделенную строку на одну строку вверх.

- `[widget name].line_down()`
  Перемещает выделенную строку на одну строку вниз.
- `[widget name].load_file('filename')`
  Загружает файл.
  Использование None (не строки имени файла) приведет к перезагрузке той же программы.
- `[widget name].get_filename()`
  FIXME описание

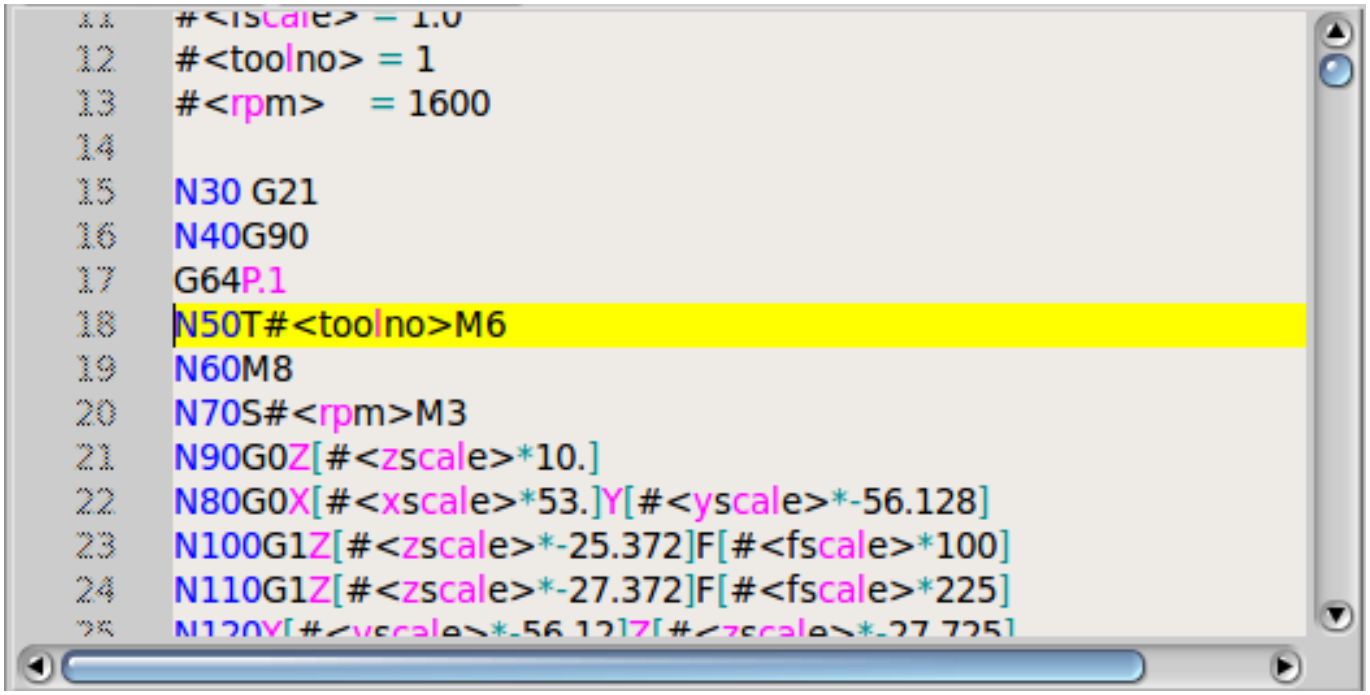



Figure 12.49: Пример Sourceview

#### 12.3.6.28 История MDI

Этот виджет служит для отображения и ввода кодов MDI.
Он будет автоматически затемнен, если MDI недоступен, например, во время аварийного останова и работы программы.

**font_size_tree**
Целое значение от 8 до 96.
Изменит размер шрифта по умолчанию для древовидного представления на выбранное значение.

**font_size_entry**
Целое значение от 8 до 96.
Измените размер шрифта записи по умолчанию на выбранное значение.

**use_double_click**
Boolean, True включает функцию двойного щелчка мышью, а двойной щелчок по записи отправляет эту команду.
Не рекомендуется использовать эту функцию на реальных станках, поскольку двойной щелчок по неправильной записи может привести к опасным ситуациям.

Использование goobject для установки перечисленных выше свойств:

```
[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 Анимированные функциональные диаграммы: виджеты HAL в растровом изображ

Для некоторых приложений может быть желательно иметь фоновое изображение (например, функциональную диаграмму) и размещать виджеты в соответствующих местах этого изображения. Хорошей комбинацией является установка растрового фонового изображения, например, из файла .png, создание фиксированного размера окна GladeVCP и использование виджета Glade фиксированного для размещения виджетов на этом изображении. Код приведенного ниже примера можно найти в `configs/apps/gladevcp/animated-backdrop`:

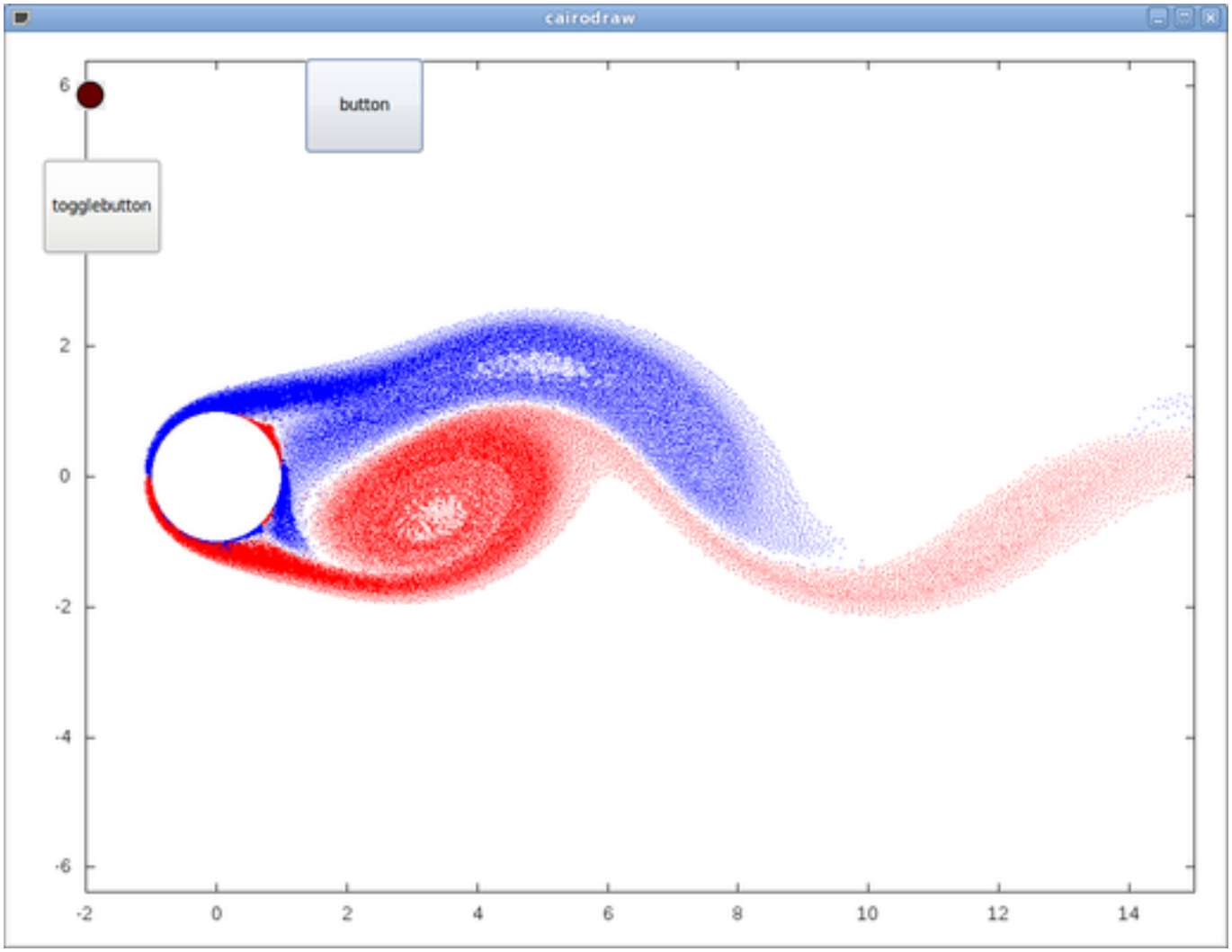



Figure 12.50: Пример виджеты HAL в растровом изображении

### 12.3.7 Справочник по виджетам действий

GladeVCP включает в себя набор ”стандартных действий” под названием **VCP Action Widgets** для редактора пользовательского интерфейса Glade.

---

**Note**
Помимо виджетов HAL, которые взаимодействуют с контактами HAL, действия VCP взаимодействуют с LinuxCNC и интерпретатором G-кода.

---

Виджеты действий VCP являются производными от виджета Gtk.Action.

Коротко о виджете Action:

- Это объект, доступный в Glade
- Сам по себе он не имеет внешнего вида
- Его цель: связать видимый, чувствительный компонент пользовательского интерфейса, такой как меню, кнопка инструмента или кнопка, с командой. См. свойство *General→Related→Action* этого виджета.
- ”Стандартное действие” будет выполняться при запуске соответствующего компонента пользовател интерфейса (нажатие кнопки, щелчок меню...).
- Он обеспечивает простой способ выполнения команд, не прибегая к программированию на Python.

Внешний вид VCP Actions в Glade примерно следующий:

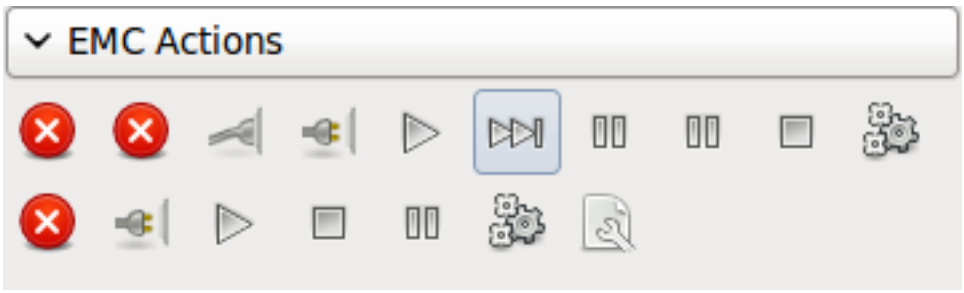



Figure 12.51: Виджеты Action

При наведении на подсказку предоставляется описание.

#### 12.3.7.1 Виджеты VCP Action

Виджеты действий VCP — это виджеты одноразового типа. Они реализуют одно действие и предназначены для использования в простых кнопках, пунктах меню или группах переключателей/пр

#### 12.3.7.2 VCP Action Python

Этот виджет используется для выполнения небольшого произвольного кода Python.

В командной строке могут использоваться специальные ключевые слова для доступа к важным функциям.

- *ACTION* для доступа к библиотеке команд ACTION.
- `GSTAT` для доступа к библиотеке статусных сообщений Gstat.
- *INFO* для доступа к собранным данным из INI-файла.
- *HAL* для доступа к модулю HAL linuxcnc Python.
- `STAT` для доступа к исходному статусу LinuxCNC через модуль LinuxCNC Python.
- `CMD` для доступа к командам LinuxCNC через модуль LinuxCNC Python.
- `EXT` для доступа к функциям файла-обработчика, если они доступны.
- `linuxcnc` для доступа к модулю LinuxCNC Python.
- `self` для доступа к экземпляру виджета.
- *dir* для доступа к списку атрибутов обработчиков.

Есть варианты

- выбрать, когда виджет будет активен,
- установить режим перед выполнением команды.

Пример команды для печати сообщения на терминале:

```
print('action activated')
```

Пример команды для перевода станка в выключенное состояние:

```
CMD.state(linuxcnc.STATE_OFF)
```

Пример команды для вызова функции-обработчика, передающей данные:

```
EXT.on_button_press(self, 100)
```

Вы можете использовать точку с запятой для разделения нескольких команд;

```
print('Set Machine Off');CMD.state(linuxcnc.STATE_OFF)
```

Более подробную информацию о INFO и ACTION можно найти здесь: GladeVCP Libraries modules.

Более подробную информацию о GStat можно найти здесь.: GStat.

#### 12.3.7.3 Виджеты VCP ToggleAction

Это **бимодальные** виджеты. Они реализуют два действия или используют второе (обычно `нажатое`) состояние, чтобы указать, что в данный момент выполняется действие. Действия переключения предназначены для использования в `ToggleButtons`, `ToggleToolButtons` или для переключения пунктов меню. Симплексным примером является кнопка переключения `ESTOP`.

На данный момент доступны следующие виджеты:

- Переключатель `ESTOP` отправляет команды `ESTOP` или `ESTOP_RESET` в LinuxCNC в зависимости от его состояния.
- Переключатель `ON/OFF` отправляет команды `STATE_ON` и `STATE_OFF`.
- `Pause/Resume` отправляет команды `AUTO_PAUSE` или `AUTO_RESUME`.

Следующие действия переключения имеют только одну связанную команду и используют состояние `pressed`, чтобы указать, что запрошенная операция выполняется:

- Переключатель `Run` отправляет команду `AUTO_RUN` и ждет в `pressed` состоянии, пока интерпретатор снова не перейдет в режим ожидания.
- Переключение `Stop` неактивно до тех пор, пока интерпретатор не перейдет в активное состояние (выполняет G-код), а затем позволит пользователю отправить команду `AUTO_ABORT`.
- Переключение `MDI` отправляет заданную команду MDI и ожидает ее завершения в `pressed` неактивном состоянии.

#### 12.3.7.4 Виджеты Action_MDI Toggle и Action_MDI

Эти виджеты предоставляют средства для выполнения произвольных команд MDI.
Виджет Action_MDI не ожидает завершения команды, как это делает переключатель Action_MDI, который остается отключенным до завершения команды.

#### 12.3.7.5 Простой пример: выполнение команды MDI при нажатии кнопки

`configs/apps/gladevcp/mdi-command-example/whoareyou.ui` — это файл пользовательского интерфей Glade, который передает основы:

1. Откройте его в Glade и изучите, как это делается.
2. Запустите AXIS, а затем запустите его из окна терминала с помощью `gladevcp whoareyou.ui`.
3. См. действие `hal_action_mdi1` и его свойство `MDI command` — оно просто выполняет `(MSG, "Hi, I'm an VCP_Action_MDI")`, поэтому в AXIS должно появиться всплывающее окно с сообщени например:

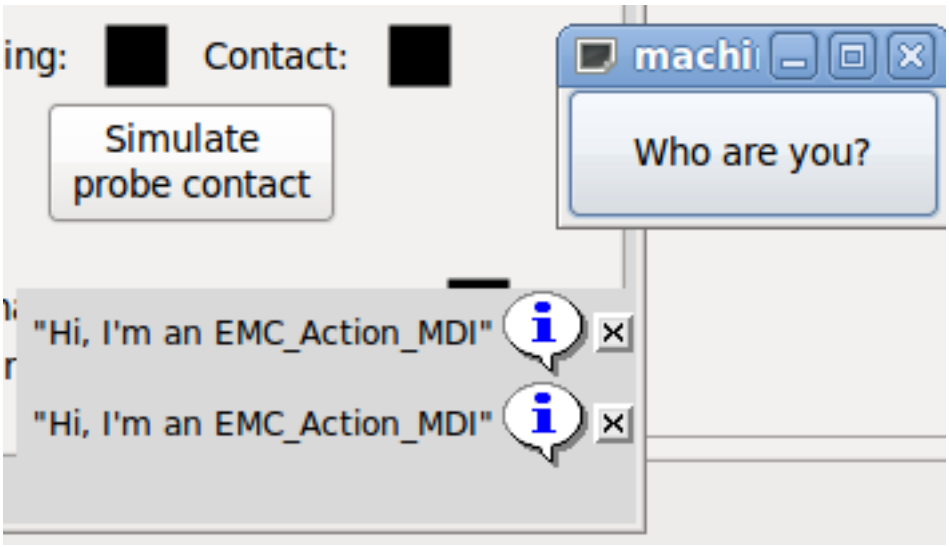



Figure 12.52: Простой пример Action_MDI

Вы заметите, что кнопка, связанная с действием Action_MDI, неактивна, если машина выключена, находится в режиме аварийного останова или если интерпретатор работает. Он автоматически становится активным, когда станок включается и выходит из режима аварийного останова, а программа находится в режиме ожидания.

#### 12.3.7.6 Передача параметров с помощью виджетов Action_MDI и ToggleAction_MDI

При желании в строки `MDI-команды` могут быть подставлены параметры перед их передачей интерпретатору. В настоящее время параметрами могут быть имена контактов HAL в компоненте GladeVCP. Вот как это работает:

- Предположим, у вас есть HAL SpinBox с именем `speed`, и вы хотите передать его текущее значение в качестве параметра в команде MDI.
- HAL SpinBox будет иметь контакт HAL плавающего типа с именем Speed-f (см. описание Hal-Widgets).
- Чтобы заменить это значение в команде MDI, вставьте имя контакта HAL, заключенное следующим образом: `${pin-name}`
- для приведенного выше HAL SpinBox мы могли бы использовать `(MSG, "The speed is: ${speed-f}"` просто для того, чтобы показать, что происходит.

Пример файла пользовательского интерфейса — `configs/apps/gladevcp/mdi-command-example/speed` Вот что вы получите при его запуске:

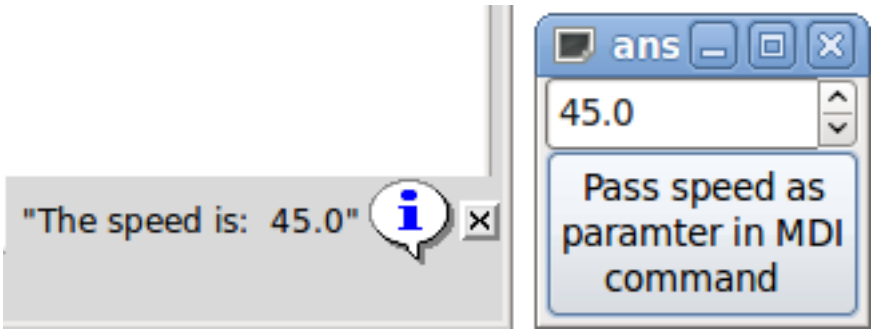



Figure 12.53: Пример передачи параметра Action_MDI

#### 12.3.7.7 Расширенный пример: передача параметров в подпрограмму с O-словом

Совершенно нормально вызывать подпрограмму с O-словом в команде MDI и передавать значения контактов HAL в качестве фактических параметров. Пример файла пользовательского интерфейса находится в `configs/apps/gladevcp/mdi-command-example/owordsub.ui`.

Поместите `nc_files/gladevcp_lib/oword.ngc`, чтобы AXIS могла его найти, и запустите `gladevcp owordsub.ui` из окна терминала. Это выглядит так:

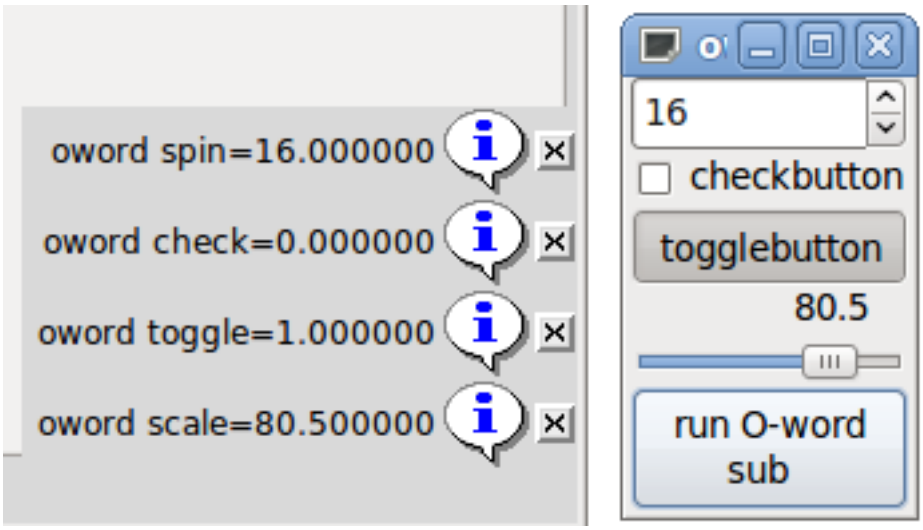



Figure 12.54: Расширенный пример Action_MDI

#### 12.3.7.8 Подготовка к действию MDI и последующая очистка

Интерпретатор G-кода LinuxCNC имеет единый глобальный набор переменных, таких как подача, скорость шпинделя, относительный/абсолютный режим и другие. Если вы используете команды G-кода или подпрограммы O-слов, некоторые из этих переменных могут быть изменены командой или подпрограммой - например, процедура проверки, скорее всего, установит значение подачи довольно низким. Без каких-либо дополнительных мер предосторожности ваша предыдущая настройка подачи будет перезаписана значением подпрограммы зондирования.

Чтобы справиться с этим удивительным и нежелательным побочным эффектом данной подпрограммы O-слова или оператора G-кода, выполняемого с помощью LinuxCNC ToggleAction_MDI, вы можете связать обработчики до и после MDI с данным LinuxCNC ToggleAction_MDI. Эти обработчики являются необязательными и предоставляют возможность сохранить любое состояние перед выполнением действия MDI, а затем восстановить его до предыдущих значений. Имена сигналов: `mdi-command-start` и `mdi-command-stop`; имена обработчиков могут быть установлены в Glade, как и в любом другом обработчике.

Вот пример того, как значение канала может быть сохранено и восстановлено такими обработчиками (обратите внимание, что каналы команд и состояния LinuxCNC доступны как `self.linuxcnc` и `self.stat` через класс VCP_ActionBase):

```
    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
```

Только виджет `Action_MDI` Toggle поддерживает эти сигналы.

---

**Note**
В более поздней версии LinuxCNC будут доступны новые M-коды M70-M72. Они значительно упростят сохранение состояния перед вызовом подпрограммы и восстановление состояния при возврате.

---

#### 12.3.7.9 Использование объекта LinuxCNC Stat для обработки изменений статуса

Многие действия зависят от статуса LinuxCNC — в ручном, MDI или автоматическом режиме? Программа запущена, приостановлена или простаивает? Вы не можете запустить команду MDI во время работы программы G-кода, поэтому об этом необходимо позаботиться. Многие действия LinuxCNC заботятся об этом сами, а соответствующие кнопки и пункты меню деактивируются, когда операция в данный момент невозможна.

При использовании обработчиков событий Python, которые находятся на более низком уровне, чем Actions, необходимо самому позаботиться о работе с зависимостями состояния. Для этой цели существует виджет LinuxCNC Stat: для связи изменений состояния LinuxCNC с обработчиками событий.

LinuxCNC Stat не имеет видимого компонента — вы просто добавляете его в свой пользовательский интерфейс с помощью Glade. После добавления вы можете связать обработчики со следующими сигналами:

- связанные с состоянием:
  - `state-estop`: выдается при возникновении состояния аварийного останова,
  - `state-estop-reset`: выдается при сбросе станка,
  - `state-on`: выдается при включении станка,
  - `state-off`: выдается, когда станок выключен.
- связанные с режимом:
  - `mode-manual`: выдается, когда LinuxCNC переходит в ручной режим,
  - `mode-mdi`: выдается, когда LinuxCNC входит в режим MDI,
  - `mode-auto`: выдается, когда LinuxCNC переходит в автоматический режим,
- связанный с интерпретатором: генерируется, когда интерпретатор G-кода переходит в этот режим
  - `interp-run`
  - `interp-idle`
  - `interp-paused`
  - `interp-reading`
  - `interp-waiting`
  - `file-loaded`
  - `line-changed`
- связанный с исходным положением: генерируется, когда LinuxCNC в исходном состоянии или нет
  - `all-homed`
  - `not-all-homed`

### 12.3.8 Программирование GladeVCP

#### 12.3.8.1 Действия, определяемые пользователем

Большинство наборов виджетов и связанные с ними редакторы пользовательского интерфейса поддерживают концепцию обратных вызовов, то есть функций в написанном пользователем коде, которые выполняются, когда ч*то-то происходит* в пользовательском интерфейсе — такие события,

как щелчки мыши, ввод символов, движение мыши, события таймера и т. д. скрытие и экспозиция окон и так далее.

Виджеты вывода HAL обычно сопоставляют события типа ввода, такие как нажатие кнопки, с изменением значения связанного контакта HAL посредством такого заранее определенного обратного вызова. В PyVCP это действительно единственный поддерживаемый тип обработки событий — выполнение чего-то более сложного, например выполнение команд MDI для вызова подпрограммы G-кода, не поддерживается.

В GladeVCP изменения контактов HAL — это всего лишь один тип общего класса событий (называемых сигналами) в GTK+. Большинство виджетов могут генерировать такие сигналы, и редактор Glade поддерживает связывание такого сигнала с именем метода или функции Python.

Если вы решите использовать пользовательские действия, ваша задача — написать модуль Python, методы класса которого — или, в простом случае, просто функции — могут называться в Glade обработчиками событий. GladeVCP предоставляет возможность импортировать ваши модули при запуске и автоматически связывает ваши обработчики событий с сигналами виджетов, как указано в описании пользовательского интерфейса Glade.

#### 12.3.8.2 Основная библиотека

Существует три библиотеки функций, которые можно использовать для программирования GladeVCP.

- *Info*: собирает данные из INI-файла.
- *Action*: Набор функций для изменения состояний LinuxCNC.
- *Status*: Сообщает о состоянии LinuxCNC. Он окружает *Gstat*.

Импорт и создание экземпляров библиотек:

```
from gladevcp.core import Info, Action

ACTION = Action()
INFO = Info()
```

Использование функций библиотеки:

```
print(INFO.MACHINE_IS_METRIC)
ACTION.SET_ERROR_MESSAGE('Something went wrong')
```

Дополнительную информацию можно найти здесь: GladeVCP Libraries modules. Существует пример конфигурации, демонстрирующий использование базовой библиотеки с виджетами Python действий GladeVCP и файлом-обработчиком Python. Попробуйте загрузить *sim/axis/gladevcp/gladevcp_panel_tester*.

#### 12.3.8.3 Пример: добавление пользовательских обратных вызовов в Python

Это всего лишь минимальный пример, передающий идею — подробности изложены в оставшейся части этого раздела.

GladeVCP может не только манипулировать или отображать контакты HAL, вы также можете писать обычные обработчики событий на Python. Это можно использовать, среди прочего, для выполнения команд MDI. Вот как это сделать:

Напишите такой модуль Python и сохраните его, например: handlers.py:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

В Glade определите кнопку или HAL кнопку, выберите вкладку *Signals*, а в свойствах GtkButton выберите строку *pressed*. Введите туда *on_button_press* и сохраните файл Glade.

Затем добавьте опцию *-u handlers.py* в командную строку GladeVCP. Если ваши обработчики событий распределены по нескольким файлам, просто добавьте несколько опций *-u <pyfilename>*.

Теперь нажатие кнопки должно изменить ее метку, поскольку она установлена в функции обратного вызова.

Что делает флаг `+-u+`: все функции Python в этом файле собираются и настраиваются как потенциальны обработчики обратного вызова для ваших виджетов Gtk — на них можно ссылаться на вкладках Glade *Signals*. Обработчики обратного вызова вызываются с конкретным экземпляром объекта в качестве параметра, как и экземпляр GtkButton выше, поэтому вы можете применить оттуда любой метод GtkButton.

Или сделайте еще что-нибудь полезное, например вызов команды MDI!

#### 12.3.8.4 События изменения значения HAL

Виджеты ввода HAL, такие как светодиод, автоматически связывают состояние своего контакта HAL (включено/выключено) с оптическим внешним видом виджета (светодиод горит/погашен).

Помимо этой встроенной функциональности, можно связать обратный вызов изменения с любым контактом HAL, включая контакты предопределенных виджетов HAL. Это хорошо вписывается в управляемую событиями структуру типичного приложения-виджета: каждое действие, будь то щелчок мыши, клавиша, истечение таймера или изменение значения контакта HAL, генерирует обратный вызов и обрабатывается одним и тем же ортогональным механизмом.

Для определяемых пользователем контактов HAL, не связанных с конкретным виджетом HAL, имя сигнала имеет значение *value-changed*. Подробности см. в разделе Добавление контактов HAL ниже.

Виджеты HAL идут с заранее определенным сигналом, который называется *hal-pin-changed*. Подробности смотрите в разделе HAL Widgets.

#### 12.3.8.5 Модель программирования

Общий подход заключается в следующем:

- Создайте свой пользовательский интерфейс с помощью Glade и установите обработчики сигналов там, где вы хотите, чтобы действия были связаны с виджетом.
- Напишите модуль Python, содержащий вызываемые объекты (см. *handler models* ниже).
- Передайте путь к вашему модулю в GladeVCP с помощью опции *-u <module>*.
- GladeVCP импортирует модуль, проверяет его на наличие обработчиков сигналов и подключает их к дереву виджетов.
- Запускается основной цикл событий.

Для простых задач достаточно определить функции, названные в честь обработчиков сигналов Glade. Они будут вызываться, когда в дереве виджетов произойдет соответствующее событие. Вот тривиальный пример: предполагается, что сигнал *pressed* кнопки Gtk или кнопки HAL связан с обратным вызовом под названием *on_button_press*:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

Добавьте эту функцию в файл Python и запустите следующим образом:

```
gladevcp -u <myhandler>.py mygui.ui
```

Обратите внимание, что связь между обработчиками должна осуществляться через глобальные переменные, что плохо масштабируется и явно не питонично. Вот почему мы придумали модель обработчика на основе классов.

Идея здесь такова: обработчики связаны с методами класса. Базовые классы создаются и проверяются во время запуска GladeVCP и связываются с деревом виджетов в качестве обработчиков сигналов. Итак, задача теперь состоит в том, чтобы написать:

- Одно или несколько определений классов с одним или несколькими методами, в одном модуле или разделенные на несколько модулей,
- функция *get_handlers* в каждом модуле, которая будет возвращать список экземпляров классов в GladeVCP — имена их методов будут связаны с обработчиками сигналов.

Вот минимальный пример модуля определяемого пользователем обработчика:

```
class MyCallbacks :
    def on_this_signal(self,obj,data=None):
        print("this_signal happened, obj=",obj)

def get_handlers(halcomp,builder,useropts):
    return [MyCallbacks ()]
```

Теперь *on_this_signal* будет доступен в качестве обработчика сигнала для вашего дерева виджетов.

Для панели GladeVCP, которая реагирует на входы HAL, может быть важно, чтобы код обработчика мог сообщать, что панель GladeVCP в данный момент активна и отображается. Например, панели внутри интерфейса Touchy вполне может потребоваться выполнить действие при срабатывании переключателя, подключенного к touchy.cycle-start (точно так же, как собственные вкладки по-разному реагируют на одну и ту же кнопку).
Чтобы это стало возможным, из ГИП (на момент написания статьи только Touchy) отправляется сигнал на встроенную вкладку. Сигнал имеет тип "Gladevcp", и отправляются два сообщения: "Visible" и "Hidden". (Обратите внимание, что сигналы имеют фиксированную длину в 20 символов, поэтому при любом сравнении следует использовать только первые символы, отсюда и [:7] ниже.)
Пример обработчика для этих сигналов:

```
    # 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)
```

Если во время проверки модуля GladeVCP обнаруживает функцию `get_handlers`, она вызывает ее следующим образом:

```
get_handlers(halcomp,builder,useropts)
```

Аргументы:

- `halcomp` - относится к разрабатываемому компоненту HAL,
- `builder` - дерево виджетов — результат чтения определения пользовательского интерфейса (либо ссылка на объект типа GtkBuilder, либо на объект типа libglade),
- `useropts` - список строк, собранных из опции командной строки GladeVCP `-U <useropts>`.

Затем GladeVCP проверяет список экземпляров классов и извлекает имена их методов. Имена уточняющих методов подключаются к дереву виджетов как обработчики сигналов. Учитываются только имена методов, которые не начинаются с символа _ (подчеркивание).

Обратите внимание: независимо от того, используете ли вы libglade или новый формат Gtk-Builder для своего пользовательского интерфейса Glade, виджеты всегда можно называть `builder.get`
Кроме того, полный список виджетов доступен как `builder.get_objects()` независимо от формата пользовательского интерфейса.

#### 12.3.8.6 Последовательность инициализации

Важно знать, в каком случае вызывается ваша функция get_handlers(), чтобы вы знали, что там делать безопасно, а что нет. Сначала модули импортируются и инициализируются в порядке командной строки. После успешного импорта get_handlers() вызывается в следующем состоянии:

- Дерево виджетов создано, но еще не реализовано (еще не выполнено `window.show()` верхнего уровня).
- Компонент halcomp HAL настроен, и к нему уже добавлены контакты всех виджетов HAL.
- Добавлять больше контаков HAL безопасно, поскольку на этом этапе `halcomp.ready()` еще не вызывался, поэтому вы можете добавить свои собственные выводы, например, в методе класса *init*()`.

После того как все модули импортированы и имена методов извлечены, выполняются следующие шаги:

- Все имена соответствующих методов будут подключены к дереву виджетов с помощью `connect_sign`
(в зависимости от типа импортируемого пользовательского интерфейса — GtkBuilder или старый формат libglade).
- Компонент HAL завершается с помощью halcomp.ready().
- Если ID окна был передан в качестве аргумента, дерево виджетов перестраивается для запуска в этом окне, а window1 верхнего уровня Glade1 закрывается (см. FAQ).
- Если командный файл HAL был передан с параметром `-H halfile`, он выполняется с помощью halcmd.
- Запускается основной цикл Gtk.

Поэтому, когда ваш класс-обработчик инициализирован, все виджеты существуют, но еще не реализованы (отображаются на экране). Компонент HAL также не готов, поэтому небезопасно получать доступ к значениям контактов в вашем методе `__init__()`.

Если вы хотите, чтобы обратный вызов выполнялся при запуске программы после того, как доступ к выводам HAL станет безопасным, тогда подключите обработчик к сигналу реализации window1 верхнего уровня (что может быть его единственной реальной целью). На этом этапе GladeVCP завершил все задачи настройки, файл HAL запущен, и GladeVCP собирается войти в основной цикл Gtk.

#### 12.3.8.7 Несколько обратных вызовов с одним и тем же именем

Внутри класса имена методов должны быть уникальными. Однако вполне нормально иметь несколько экземпляров класса, передаваемых в GladeVCP с помощью get_handlers() с одинаковыми именами методов. При возникновении соответствующего сигнала эти методы будут вызываться в порядке определения — модуль за модулем, а внутри модуля экземпляры класса возвращаются методом `get_handlers()`.

#### 12.3.8.8 Флаг GladeVCP `-U <useropts>`

Вместо расширения GladeVCP для любой мыслимой опции, которая потенциально может быть полезна для класса-обработчика, вы можете использовать флаг -U <useroption> (при желании несколько раз). Этот флаг собирает список строк <useroption>. Этот список передается в функцию get_handlers() (аргумент useropts). Ваш код может интерпретировать эти строки по своему усмотрению. Возможное использование — передать их функции exec Python в вашем `get_handlers()` следующим образом:

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
    ...
    global debug # assuming there's a global var
    for cmd in useropts:
        exec cmd in globals()
```

Таким образом, вы можете передавать в модуль произвольные операторы Python с помощью опции `gladevcp -U`, например:

```
gladevcp -U debug=42 -U "print 'debug=%d' % debug" ...
```

Это должно установить debug на 2 и подтвердить, что ваш модуль действительно это сделал.

#### 12.3.8.9 Постоянные переменные в GladeVCP

Раздражающим аспектом GladeVCP в его более ранней форме и PyVCP является тот факт, что вы можете изменять значения и контакты HAL с помощью ввода текста, ползунков, полей прокрутки, кнопок переключения и т. д., но их настройки не сохраняются и не восстанавливаются при следующем запуске LinuxCNC — они начинаются со значения по умолчанию, установленного в определении панели или виджета.

GladeVCP имеет простой в использовании механизм для сохранения и восстановления состояния виджетов HAL и программных переменных (фактически любого атрибута экземпляра типа int, float, bool или string).

Этот механизм использует популярный формат файлов INI для сохранения и перезагрузки постоянных атрибутов.

**Сохраняемость, версии программ и проверка подписи** Представьте себе переименование, добавление или удаление виджетов в Glade: файл .INI, хранящийся в предыдущей версии программы, или совершенно другой пользовательский интерфейс не смогут правильно восстановить состояние, поскольку переменные и типы могут измениться.

GladeVCP обнаруживает эту ситуацию по сигнатуре, которая зависит от всех имен и типов объектов, которые сохраняются и подлежат восстановлению. В случае несовпадения подписи создается новый INI-файл с настройками по умолчанию.

#### 12.3.8.10 Использование постоянных переменных

Если вы хотите, чтобы какое-либо состояние виджета Gtk, выходные значения контактов виджетов HAL и/или атрибуты класса вашего класса-обработчика сохранялись при вызовах, действуйте следующим образом:

- Импортируйте модуль Gladevcp.persistence.
- Решите, какие атрибуты экземпляра и их значения по умолчанию вы хотите сохранить, если таковые имеются.
- Решите, состояние каких виджетов должно сохраняться.
- Опишите эти решения в методе `\__init()__` вашего класса-обработчика через вложенный словарь следующим образом:

```
def __init__(self, halcomp,builder,useropts):
    self.halcomp = halcomp
    self.builder = builder
    self.useropts = useropts
    self.defaults = {
        # the following names will be saved/restored as method attributes
        # the save/restore mechanism is strongly typed - the variables type will be derived ↩
            from the type of the
        # initialization value. Currently supported types are: int, float, bool, string
        IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "a string"},
        # to save/restore all widget's state which might remotely make sense, add this:
        IniFile.widgets : widget_defaults(builder.get_objects())
        # a sensible alternative might be to retain only all HAL output widgets' state:
        # IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(),  ↩
            hal_only=True,output_only = True)),
    }
```

Затем свяжите INI-файл с этим дескриптором:

```
self.ini_filename = __name__ + '.ini'
self.ini = IniFile(self.ini_filename,self.defaults,self.builder)
self.ini.restore_state(self)
```

После `restore_state()` у self будут установлены атрибуты, если выполняется следующее:

```
self.nhits = 0
self.a = 1.67
self.d = True
self.c = "a string"
```

Обратите внимание, что типы сохраняются и сохраняются при восстановлении. В этом примере предполагается, что файл INI не существует или имеет значения по умолчанию из self.defaults.

После этого заклинания вы можете использовать следующие методы IniFile:

**ini.save_state(obj)**
: Сохраняет атрибуты objs согласно словарю `IniFile.vars` и состояние виджета, как описано в `IniFile.widgets` в `self.defaults`.

**ini.create_default_ini()**
: Создает INI-файл со значениями по умолчанию.

**ini.restore_state(obj)**
: Восстановите выходные контакты HAL и атрибуты объекта как сохраненные/инициализированные по умолчанию, как указано выше.

#### 12.3.8.11 Сохранение состояния при завершении работы GladeVCP

Чтобы сохранить состояние виджета и/или переменной при выходе, выполните следующие действия:

- Выберите какой-нибудь внутренний виджет (тип не важен, например стол).
- На вкладке *Signals* выберите *GtkObject*. В первом столбце должен появиться сигнал *destroy*.
- Добавьте имя обработчика, например. *on_destroy* во второй столбец.
- Добавьте обработчик Python, как показано ниже:

```
import gtk
...
def on_destroy(self,obj,data=None):
    self.ini.save_state(self)
```

Это позволит сохранить состояние и правильно завершить работу GladeVCP, независимо от того, встроена ли панель в AXIS или является отдельным окном.

---

**Caution**
Не используйте `window1` (окно верхнего уровня) для подключения события *destroy*. Из-за того, как панель GladeVCP взаимодействует с AXIS, если панель встроена в AXIS, **window1 не будет правильно получать события уничтожения**. Однако, поскольку при выключении все виджеты уничтожаются, подойдет любой. Рекомендуется: используйте виджет второго уровня — например, если у вас на панели есть контейнер таблицы, используйте его.

---

При следующем запуске приложения GladeVCP виджеты должны появиться в том состоянии, в котором оно было закрыто.

---

**Caution**
Строка *GtkWidget* имеет похожее звучание *destroy-event* — **не используйте его для подключения к обработчику *on_destroy*, это не сработает** — убедитесь, что вы используете событие «*destroy* из строки *GtkObject*.

---

#### 12.3.8.12 Сохранение состояния при нажатии Ctrl-C

По умолчанию реакция GladeVCP на событие Ctrl-C заключается в простом выходе - *без* сохранения состояния. Чтобы убедиться, что этот случай рассмотрен, добавьте вызов обработчика `on_unix_signal` который будет автоматически вызываться по Ctrl-C (фактически по сигналам SIGINT и SIGTERM). Пример:

```
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 Ручное редактирование файлов INI (.ini)

Вы можете это сделать, но учтите, что значения в self.defaults переопределяют ваши изменения, если в вашем редактировании есть синтаксическая или типическая ошибка. Ошибка обнаружена, консольное сообщение уведомит об этом, а плохой ini-файл будет переименован и получит суффикс .BAD. Последующие неверные файлы INI перезаписывают более ранние файлы .BAD.

---

#### 12.3.8.14 Добавление контактов HAL

Если вам нужны контакты HAL, которые не связаны с конкретным виджетом HAL, добавьте их следующим образом:

```
import hal_glib
...
# in your handler class __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, hal. ↩
    HAL_IN))
```

Чтобы получить обратный вызов при изменении значения этого контакта, свяжите обратный вызов `value-change` с этим контактом, добавьте:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

и определите метод обратного вызова (или функцию, в данном случае оставьте параметр `self`):

```
# note '_' - this method will not be visible to the widget tree
def _on_example_trigger_change(self,pin,userdata=None):
    print("pin value changed to:" % (pin.get()))
```

#### 12.3.8.15 Добавление таймеров

Поскольку GladeVCP использует виджеты Gtk, основанные на базовом классе PyGObject, функциональ доступен полный функционал GLib. Вот пример обратного вызова таймера:

```
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 Программная настройка свойств виджета HAL

В Glade свойства виджетов обычно устанавливаются фиксированными во время редактирования. Однако вы можете установить свойства виджета во время выполнения, например, из значений INI-файла, что обычно делается в коде инициализации обработчика. Также возможна установка свойств из значений контактов HAL.

В следующем примере (предполагается, что виджет HAL Meter называется `meter`) минимальное значение счетчика устанавливается из параметра INI-файла при запуске, а максимальное значение устанавливается через контакт HAL, что приводит к динамической перенастройке масштаба виджета:

```
import linuxcnc
import os
import hal
import hal_glib

class HandlerClass:

    def _on_max_value_change(self,hal_pin,data=None):
        self.meter.max = float(hal_pin.get())
        self.meter.queue_draw() # force a widget redraw
```

```
    def __init__(self, halcomp,builder,useropts):
        self.builder = builder

        # HAL pin with change callback.
        # When the pin's value changes the callback is executed.
        self.max_value = hal_glib.GPin(halcomp.newpin('max-value',  hal.HAL_FLOAT, hal. ↩
            HAL_IN))
        self.max_value.connect('value-changed', self._on_max_value_change)

        inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
        mmin = inifile.getreal("METER", "MIN", fallback=0.0)
        self.meter = self.builder.get_object('meter')
        self.meter.min = mmin


def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

#### 12.3.8.17 Обратный вызов с изменением значения с помощью hal_glib

GladeVCP использует библиотеку hal_glib, которую можно использовать для подключения сигнала "watcher" к входному контакту HAL.
Этот сигнал можно использовать для регистрации функции, которая будет вызываться, когда вывод HAL меняет состояние.

Необходимо импортировать модули `hal_glib` и `hal`:

```
import hal_glib
import hal
```

Затем сделайте вывод и подключите сигнал *value-changed* (watcher) к вызову функции:

```
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)
```

И есть функция, которую нужно вызвать:

```
    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()))

        # this can be called outside the function
        self.example_trigger.get()
```

#### 12.3.8.18 Примеры и создание собственного приложения GladeVCP

Посетите `linuxcnc_root_directory/configs/apps/gladevcp` для запуска примеров и начальных версий ваших собственных проектов.

### 12.3.9 FAQ

1. *Я получаю неожиданное событие unmap в моей функции-обработчике сразу после запуска. Что это?*

   Это является следствием того, что в вашем файле пользовательского интерфейса Glade для свойства window1 Visible установлено значение True, а также переназначение родительского окна GladeVCP на AXIS или Touchy. Создается дерево виджетов GladeVCP, включая окно верхнего уровня, а затем *перероднится в AXIS*, в результате чего это окно верхнего уровня остается бесхозным. Чтобы избежать зависания этого бесполезного пустого окна, оно не отображается (сделано невидимым), что и является причиной получаемого вами сигнала отмены отображения. Предлагаемое исправление: установите для window1.visible значение False и игнорируйте начальное событие unmap.

2. *Моя программа GladeVCP запускается, но не появляется ожидаемое окно?*

   Окно, которое AXIS выделяет для GladeVCP, получит *естественный размер* всех своих дочерних виджетов вместе взятых. Задача дочернего виджета — запросить размер (ширину и/или высоту). Однако не все виджеты запрашивают ширину больше 0, например виджет Graph в его текущей форме. Если в вашем файле Glade есть такой виджет, и именно он определяет макет, возможно, вы захотите явно установить его ширину. Обратите внимание, что установка свойств ширины и высоты window1 в Glade не имеет смысла, поскольку это окно станет осиротевшим во время повторного родительского контроля, и, следовательно, его геометрия не будет влиять на макет (см. выше). Общее правило таково: если вы вручную запускаете файл пользовательского интерфейса с помощью *gladevcp <uifile>* и его окно имеет разумную геометрию, оно также должно корректно отображаться в AXIS.

3. *Хочу мигающий светодиод, но он не мигает*

   Я установил галочку, чтобы он мигал с интервалом 100 мс. Он не мигает, и я получаю предупреждение при запуске: Внимание: значение ”0” типа `gint` недопустимо или выходит за пределы диапазона свойства `led-blink-rate` типа `gint`? Похоже, это ошибка Glade. Просто введите в поле частоту моргания и снова сохраните — у меня это работает.

4. *Моя панель GladeVCP в AXIS не сохраняет состояние, когда я закрываю AXIS, хотя я определил обработчик on_destroy, связанный с сигналом уничтожения окна*

   Скорее всего, этот обработчик связан с окном window1, которое из-за переоформления невозможно использовать для этой цели. Пожалуйста, свяжите обработчик on_destroy с сигналом уничтожения внутреннего окна. Например, у меня есть блокнот внутри window1, и я связал on_destroy с сигналом уничтожения блокнотов, и это работает нормально. Это не работает для window1.

5. *Я хочу установить цвет фона или текст виджета HAL_Label в зависимости от его значения контакта HAL*

   См. пример в configs/apps/gladevcp/colored-label. Установить цвет фона виджета GtkLabel (a HAL_Label является производным от GtkLabel) немного сложно. Виджет GtkLabel не имеет собственного объекта окна по соображениям производительности, и только объекты окна могут иметь цвет фона. Решение состоит в том, чтобы поместить метку в контейнер Event-Box, который имеет окно, но в противном случае невидим — см. файл colorlabel.ui.

**Я определил виджет `hal_spinbutton` в Glade и установил свойство value по умолчанию в со**

Это связано с ошибкой в старой версии Gtk, поставляемой с Ubuntu 8.04 и 10.04, и, скорее всего, это относится ко всем виджетам, использующим настройку. Обходной путь, упомянутый, например, в http://osdir.com/ml/gtk-app-devel-list/2010-04/msg00129.html, не позволяет надежно установить значение контакта HAL, лучше установить его явно в ` on_realize` обработчик сигнала во время создания виджета. См. пример в `configs/apps/gladevcp/by-widget/spinbutto`

### 12.3.10 Решение проблем

- Убедитесь, что у вас установлена версия LinuxCNC для разработки. Вам больше не нужен файл axisrc, об этом упоминалось на старой вики-странице GladeVCP.
- Запустите GladeVCP или AXIS из окна терминала. Если вы получаете ошибки Python, проверьте, существует ли еще файл `/usr/lib/python2.6/dist-packages/hal.so`, помимо более нового `/usr/lib/python2.6/dist-packages/_hal. .so` (обратите внимание на подчеркивание); если да, удалите файл `hal.so`. Он был заменен файлом hal.py в том же каталоге и сбивает с толку механизм импорта.
- Если вы используете запуск на месте, выполните команду *make clean*, чтобы удалить случайно оставшийся файл hal.so, а затем выполните команду *make*.
- Если вы используете виджеты *HAL_table* или *HAL_HBox*, имейте в виду, что с ними связан контакт HAL, который по умолчанию отключен. Этот контакт контролирует, активны или нет дочерние элементы этого контейнера.

### 12.3.11 Примечание по реализации: обработка ключей в AXIS

Мы считаем, что обработка ключей работает нормально, но, поскольку это новый код, мы рассказываем вам о нем, чтобы вы могли следить за проблемами; пожалуйста, сообщите нам об ошибках или странном поведении. Это история:

AXIS использует набор виджетов TkInter. Приложения GladeVCP используют виджеты Gtk и запускаются в контексте отдельного процесса. Они подключены к AXIS с помощью протокола Xembed. Это позволяет дочернему приложению, такому как GladeVCP, правильно разместиться в родительском окне и, теоретически, иметь встроенную обработку событий.

Однако это предполагает, что и родительское, и дочернее приложение должным образом поддерживают протокол Xembed, что Gtk поддерживает, а TkInter — нет. Следствием этого является то, что некоторые ключи не будут правильно пересылаться с панели GladeVCP в AXIS ни при каких обстоятельствах. Одной из таких ситуаций был случай, когда виджет Entry или SpinButton имел фокус: в этом случае, например, клавиша Escape не была бы перенаправлена в AXIS и не вызвала бы аварийное завершение работы, как должно, с потенциально катастрофическими последствиями.

Таким образом, ключевые события в GladeVCP обрабатываются явно и выборочно пересылаются в AXIS, чтобы гарантировать, что такие ситуации не могут возникнуть. Подробности см. в функции `keyboard_forward()` в `lib/python/gladevcp/xembed.py`.

### 12.3.12 Добавление пользовательских виджетов

В Wiki LinuxCNC есть информация о добавлении пользовательских виджетов в GladeVCP. Пользовательские виджеты GladeVCP

### 12.3.13 Вспомогательные приложения GladeVCP

Поддержка предоставляется для независимо установленных приложений GladeVCP, которые соответствуют размещению в системном каталоге, как определено элементами LINUXCNC_AUX_GLADEVCP и LINUXCNC_AUX_EXAMPLES, сообщаемыми скриптом linuxcnc_var:

```
$ linuxcnc_var LINUXCNC_AUX_GLADEVCP
/usr/share/linuxcnc/aux_gladevcp
$ linuxcnc_var LINUXCNC_AUX_EXAMPLES
/usr/share/linuxcnc/aux_examples
```

Системный каталог, определенный LINUXCNC_AUX_GLADEVCP (/usr/share/linuxcnc/aux_gladevcp), указывает расположение файлов Python, совместимых с GladeVCP, и связанных с ним подкаталогов. Файл Python импортируется при запуске GladeVCP и становится доступным для последующих приложений GladeVCP, включая встроенное использование в поддержке ГИП'ов.

Системный каталог, определенный LINUXCNC_AUX_EXAMPLES (/usr/share/linuxcnc/aux_examples), определяет расположение примерных подкаталогов конфигурации, используемых для вспомогательны приложений. См. раздел Начало работы/Запуск LinuxCNC на предмет *Adding Configuration Selection Items*.

Для тестирования можно указать спецификацию времени выполнения вспомогательных приложений с помощью экспортированной переменной среды: GLADEVCP_EXTRAS. Эта переменная должна представлять собой список путей из одного или нескольких каталогов конфигурации, разделенных знаком (:). Обычно эта переменная задается в оболочке, запускающей `linuxcnc`, или в сценарии запуска `~/.profile` пользователя. Пример:

```
export GLADEVCP_EXTRAS=~/mygladevcp:/opt/othergladevcp
```

Файлы, найденные в каталогах, указанных с помощью переменной среды GLADEVCP_EXTRAS, заменяют файлы с одинаковыми именами в подкаталогах системного каталога, указанных LINUXNC_AUX_GLADEVCP (например, /usr/share/linuxcnc/aux_gladevcp). Это положение позволяет разработчику тестировать приложение, экспортируя GLADEVCP_EXTRAS, чтобы указать частный каталог приложения, не удаляя каталог приложения, установленный в системе. Сообщения, указывающие на отклоненные дубликаты, выводятся на стандартный вывод.

---

**Note**
Для поддержки вспомогательных приложений GladeVCP требуется модуль Python с именем *importlib*. Этот модуль может быть недоступен в старых версиях, таких как Ubuntu-Lucid.

---

## 12.4 Модули библиотеки GladeVCP

Библиотеки — это готовые модули Python, которые предоставляют GladeVCP дополнительные возможности. Таким образом, вы можете выбрать, какие функции вам нужны, но при этом вам не придется самостоятельно создавать общие.

### 12.4.1 Информация

Info — это библиотека для сбора и фильтрации данных из INI-файла.

Доступные данные и значения по умолчанию:

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")
```

```
IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, ”share”,”qtvcp”)

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint(”KINS”,”JOINTS”, fallback=0)
AVAILABLE_AXES = [’X’,’Y’,’Z’]
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:’X’,1:’Y’,2:’Z’}
GET_JOG_FROM_NAME = {’X’:0,’Y’:1,’Z’:2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, ”TYPE”, fallback=”LINEAR”)
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 units per minute
MIN_LINEAR_JOG_VEL = 60 units per minute
MAX_LINEAR_JOG_VEL = 300 units per minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint(”TRAJ”, ”SPINDLES”, fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY

# 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”)

# 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)
```

Есть несколько *вспомогательных функций*, которые в основном используются для поддержки виджетов

```
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)
```

Чтобы импортировать эти модули, добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Info
```

Чтобы создать экземпляр модуля и использовать его в файле-обработчике, добавьте этот код Python в раздел создания экземпляра:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

INFO = Info()
```

Для доступа к данным INFO используйте следующий общий синтаксис:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

### 12.4.2 Действие

Эта библиотека используется для управления контроллером движения LinuxCNC. Она пытается скрыть второстепенные детали и добавить удобные методы для разработчиков.

Чтобы импортировать эти модули, добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from gladevcp.core import Action
```

Чтобы создать экземпляр модуля и использовать его, добавьте этот код Python в раздел создания экземпляра:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

ACTION = Action()
```

Для доступа к командам действий используйте общий синтаксис, например:

```
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)
```

Есть несколько *вспомогательных функций,* которые в основном используются для поддержки этой библиотеки.

```
get_jog_info (num)
jnum_check(num)
ensure_mode(modes)
open_filter_program(filename, filter)
```

## 12.5 QtVCP

QtVCP is an **infrastructure to build custom CNC screens or control panels for LinuxCNC**.

It displays a *`.ui` file built with Qt Designer* screen editor and combines this with *Python programming* to create a GUI screen for running a CNC machine.

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 Showcase

Few examples of QtVCP built screens and virtual control panels:

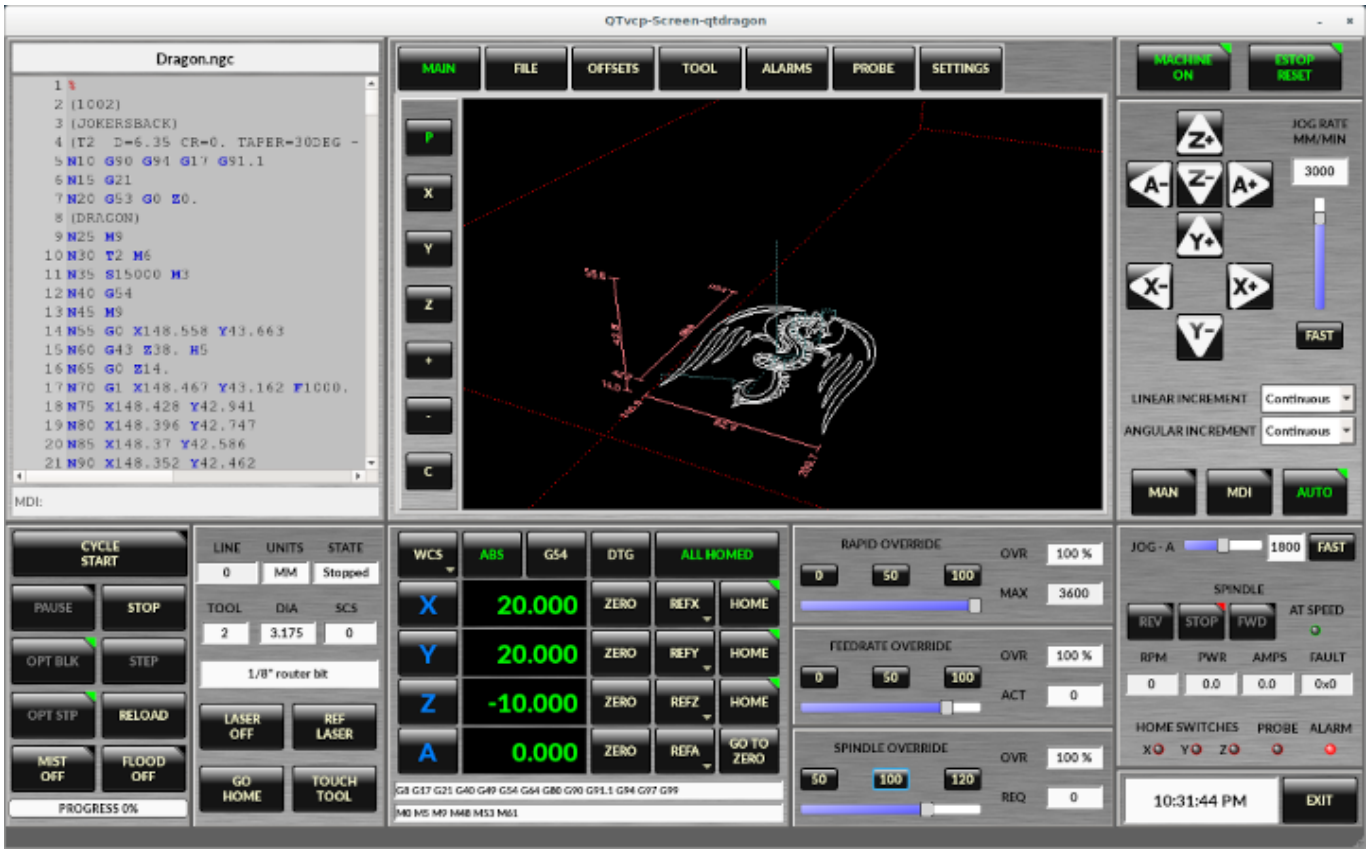



Figure 12.55: QtDragon - 3/4-Axis Sample

Figure 12.56: QtDefault - 3-Axis Sample

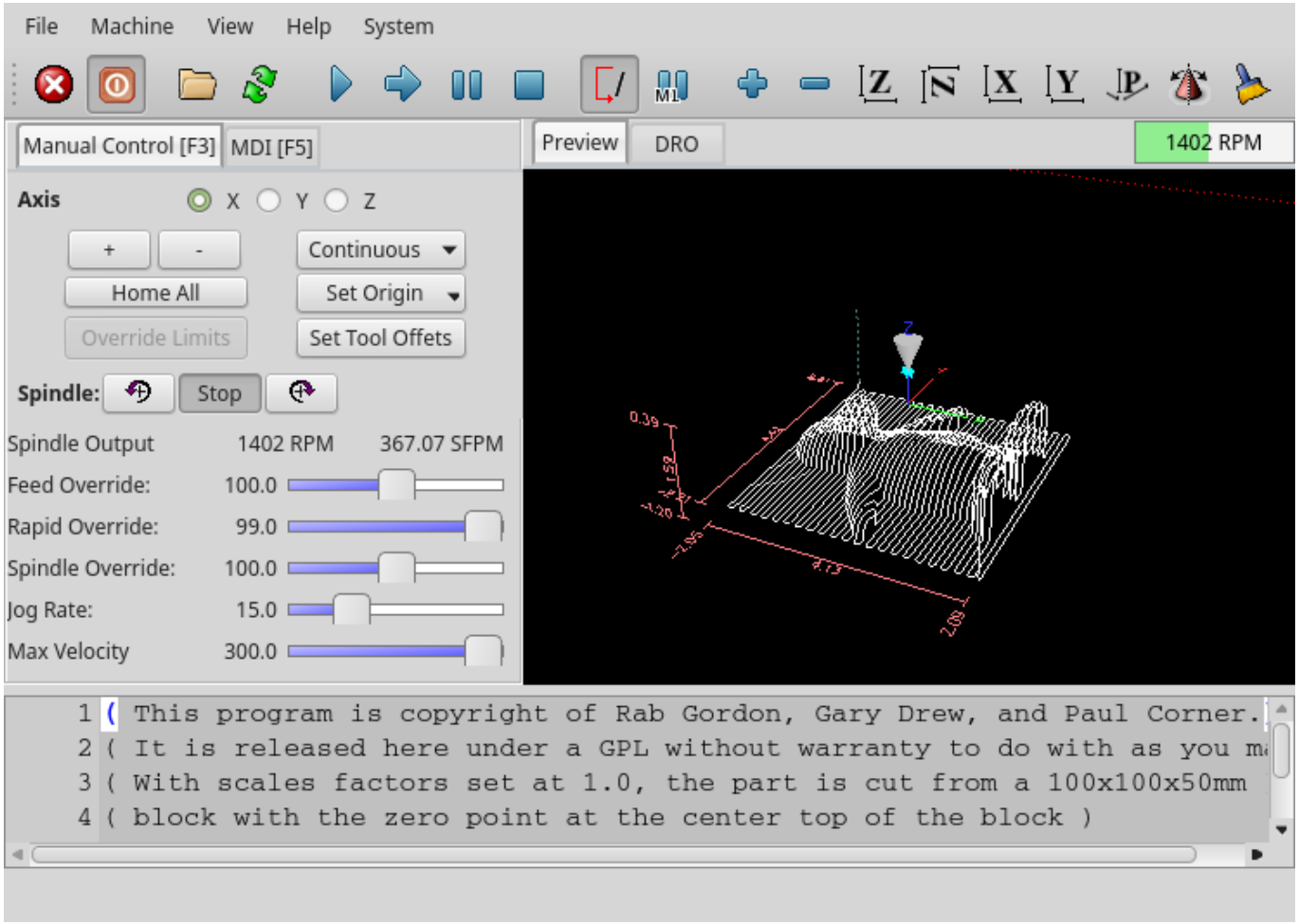



Figure 12.57: QtAxis - Self Adjusting Axis Sample

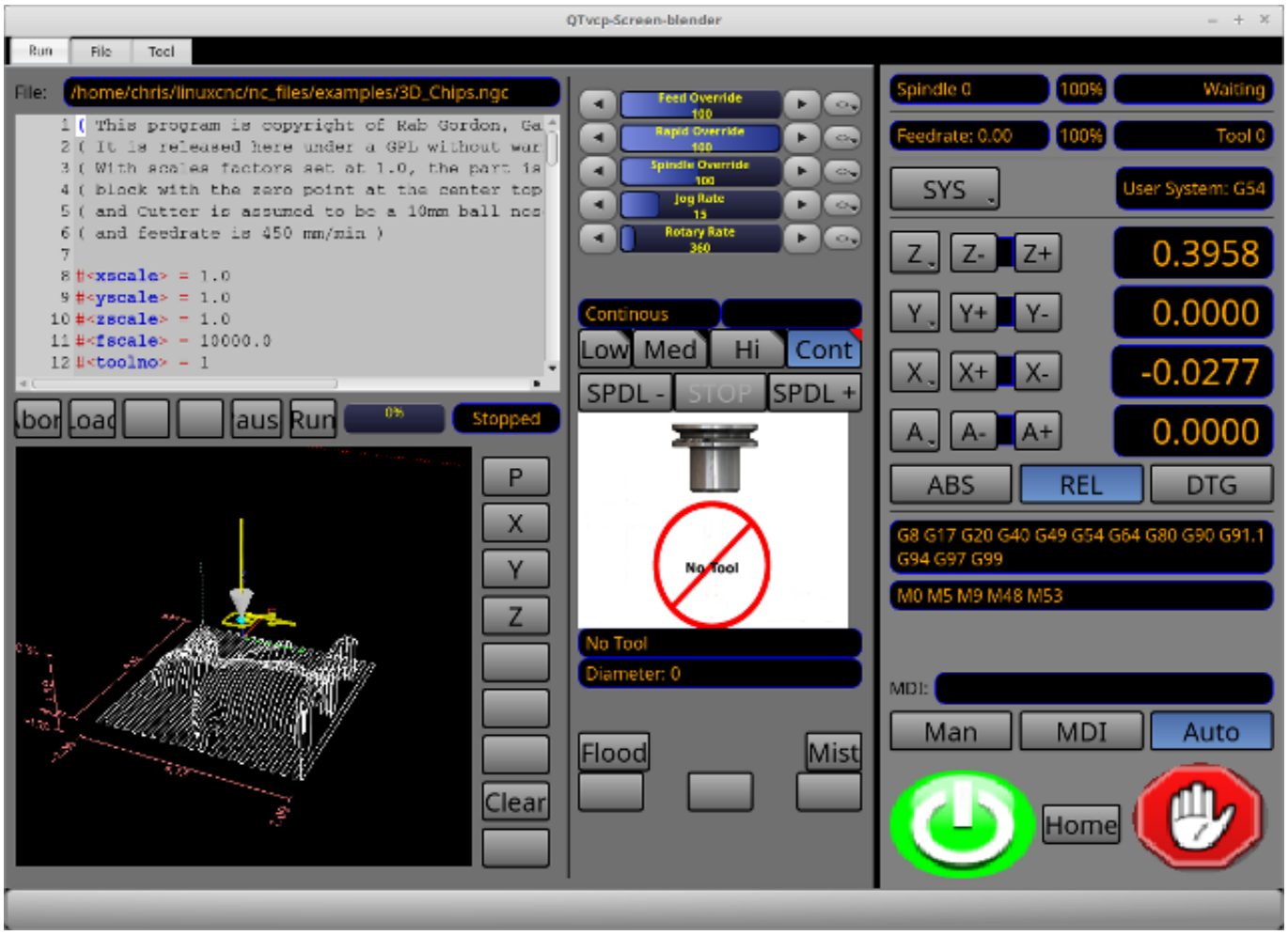



Figure 12.58: Blender - 4-Axis Sample

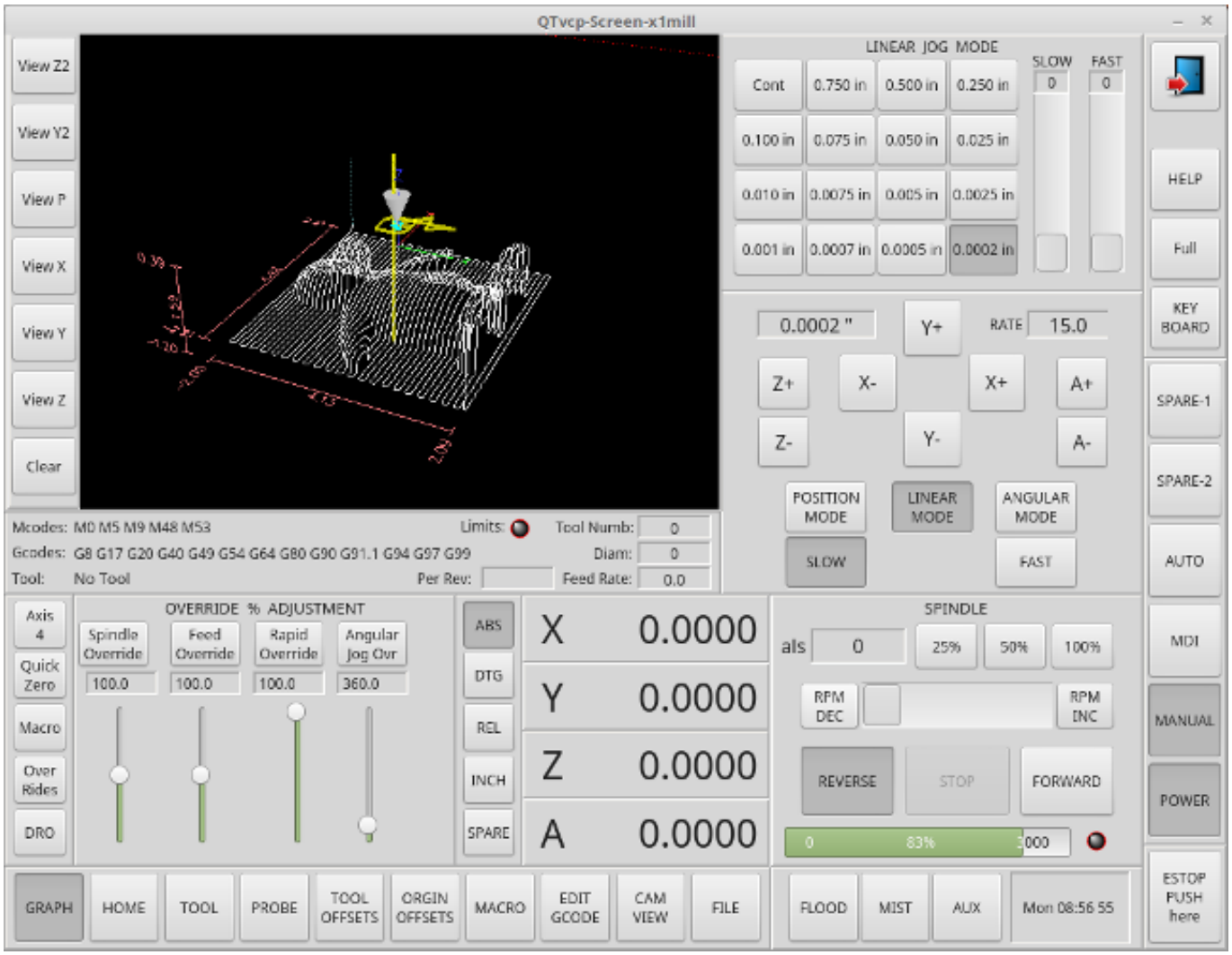



Figure 12.59: X1mill - 4-Axis Sample

Figure 12.60: cam_align - Camera Alignment VCP

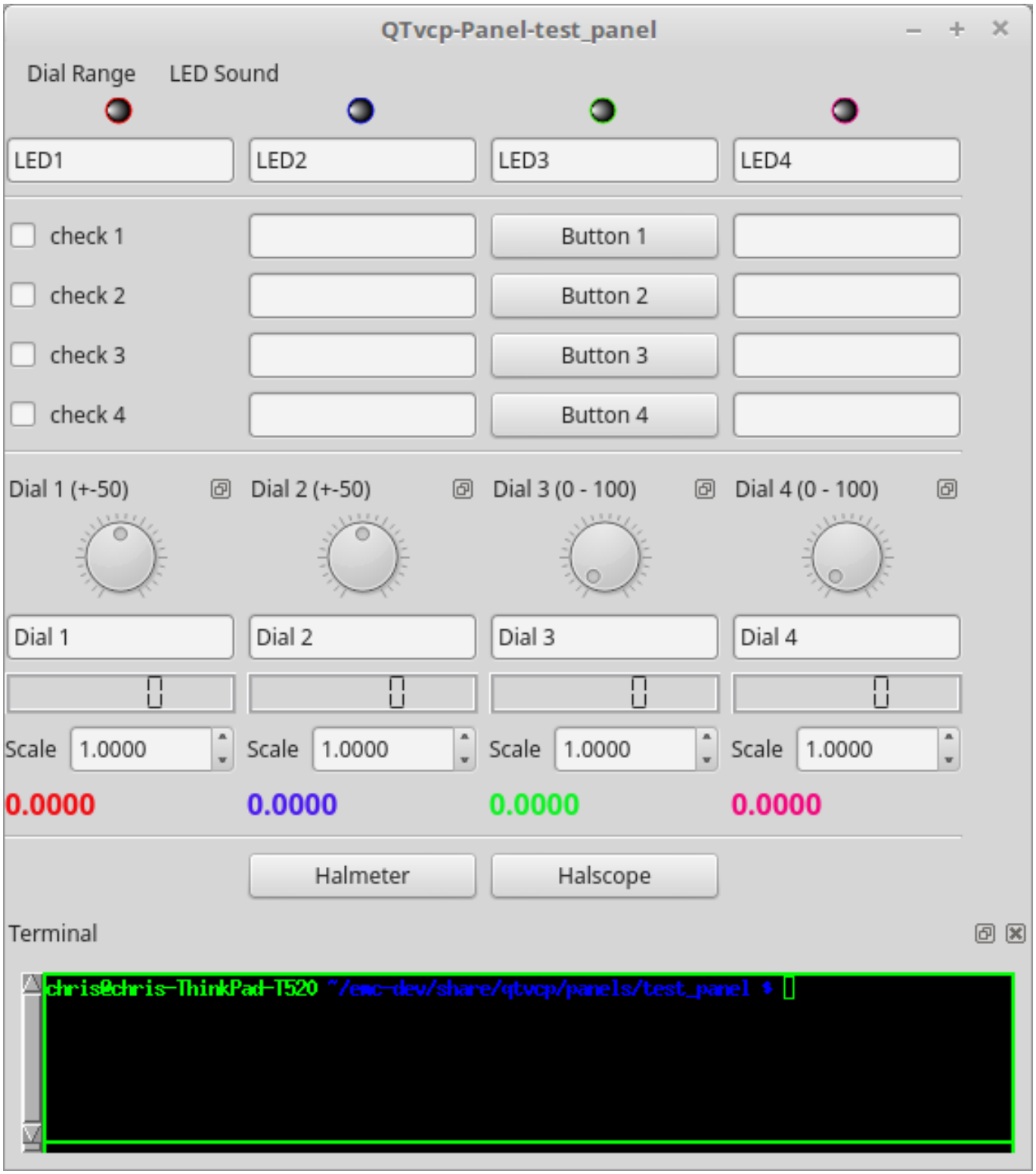



Figure 12.61: test_panel - Test Panel VCP

### 12.5.2 Обзор

*Two files are used, individually or in combination*, to add customizations:

- A **UI file** that is a *XML* file made with *Qt Designer* graphical editor.
- A **handler file** which is a *Python* code text file.

Normally QtVCP uses the stock UI and handler file, but you can specify QtVCP to use *local* UI and handler files.
A ***local* file** is one that is in the *configuration folder* that defines the rest of the machine's requirements.

One is not restricted to adding a custom panel on the right or a custom tab as QtVCP leverages *Qt Designer* (the editor) and *PyQt5* (the widget toolkit).

QtVCP has some added **special LinuxCNC widgets and actions**.
There are special widgets to bridge third party widgets to HAL pins.
It's possible to create widget responses by connecting signals to Python code in the handler file.

#### 12.5.2.1 QtVCP Widgets

QtVCP uses the **PyQt5 toolkit's widgets** for LinuxCNC integration.

**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 Settings

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>
```

> **Note**
> All `<options>` must appear before `<screen_name>`.

Варианты

- `-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` Send QtVCP's X11 window id number to standard output; for embedding.
- `-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.

**<screen_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.

**Cycle Times**

```
[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.

The widgets, graphics and HAL pin update can be set separately.

If the update time is not set right the screen can become unresponsive or very jerky.

#### 12.5.2.3 Qt Designer UI File

A Qt Designer file is a text file organized in the *XML* standard that describes the **layout and widgets** of the screen.

*PyQt5* uses this file to build the display and react to those widgets.

The Qt Designer editor makes it relatively easy to build and edit this file.

#### 12.5.2.4 Handler Files

A handler file is a file containing *Python* code, which **adds to QtVCP default routines**.

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 Libraries Modules

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).

**Библиотеки** — это готовые *модули Python*, которые **добавляют возможности** GladeVCP. Таким образом, вы можете выбрать, какие функции вам нужны, но при этом вам не придется самостоятельно создавать общие.

Such libraries include:

- `audio_player`
- `aux_program_loader`
- `keybindings`
- `message`
- `preferences`
- `notify`
- `virtual_keyboard`
- `machine_log`

#### 12.5.2.6 Themes

Themes are a way to modify the **look and feel** of the widgets on the screen.

For instance the *color* or *size* of buttons and sliders can be changed using themes.

The *Windows theme* is default for screens.
The *System theme* is default for panels.

To see available themes, they can be loaded by running the following command in a terminal:

```
qtvcp -d -t <theme_name>
```

QtVCP can also be customized with *Qt stylesheets (QSS)* using CSS.

#### 12.5.2.7 Local Files

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 will look for a folder named <screen_name> (in the launched configuration folder that holds the INI file).

In that folder, QtVCP will load any of the available following files:

- `<screen_name>.ui`,
- `<screen_name>_handler.py`, and
- `<screen_name>.qss`.

#### 12.5.2.8 Modifying Stock Screens

There are *three ways* to customize a screen/panel.

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 - effectively 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,
e.g., 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.

---

**Note**
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, e.g.:

```
[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_`**

This file will be read and executed as Python code in the **handler file context**.

**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.

What can be used can vary by screen and development cycle.

**A simple example** Reference the main window to change the title (won't show if using INI entries for title change).

```
self.w.setWindowTitle('My Title 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*.

There is a QtVCP panel to help with this:

- Open a terminal and run the following command:

```
qtvcp copy
```

- Select the screen and destination folder in the dialog
- If you wish to **name your screen** differently than the builtin screen's default name, change the *basename* in the edit box.
- 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.
- Validate to copy all the files
- Delete the files you don't wish to modify so that the original files will be used.

## 12.5.3 VCP Panels

QtVCP can be used to create control panels that interface with **HAL**.

### 12.5.3.1 Builtin Panels

There are several **builtin HAL panels** available.

In a terminal type `qtvcp <return>` to see a list:

**`test_panel`**
: Collection of useful widgets for testing HAL components, including speech of LED state.

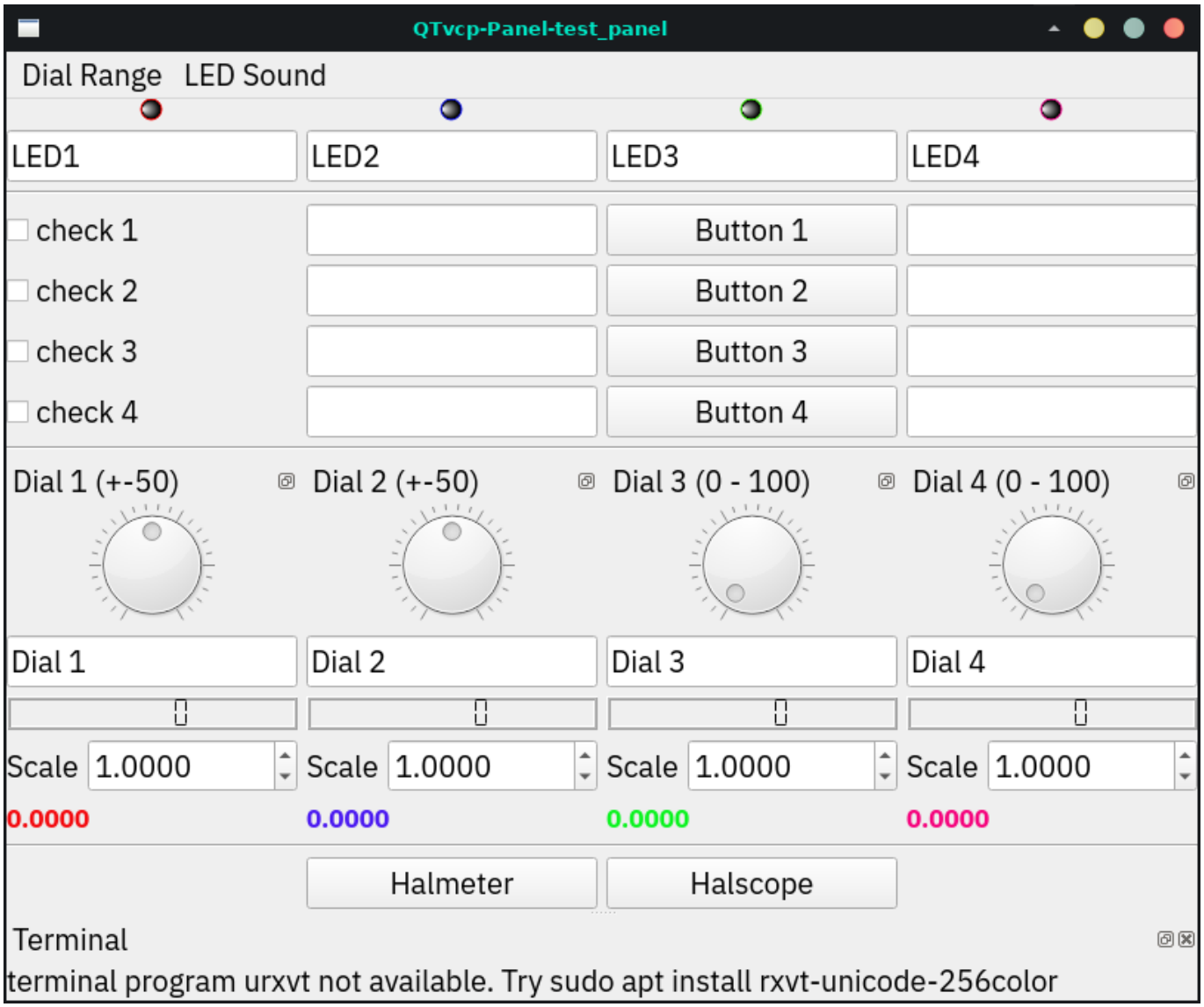



Figure 12.62: QtVCP HAL Test Builtin Panel

**cam_align**
: A camera display widget for rotational alignment.

Figure 12.63: cam_align - Camera Alignment VCP

**sim_panel**
A small control panel to simulate MPG jogging controls etc.
For simulated configurations.

Figure 12.64: QtVCP Sim Builtin Panel

**vismach_mill_xyz**
3D OpenGL view of a 3-axis milling machine.

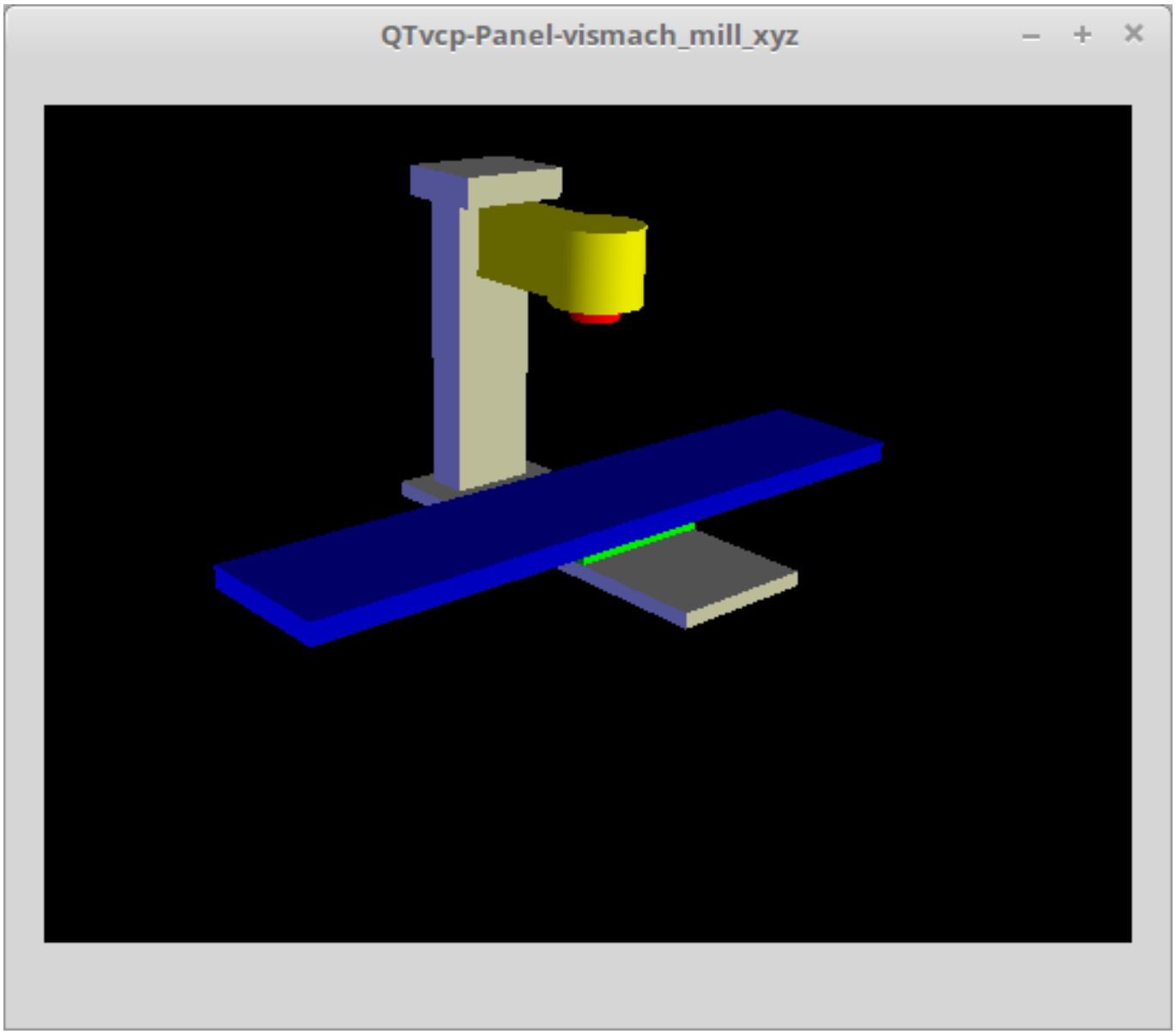



Figure 12.65: QtVismach - 3-Axis Mill Builtin Panel

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 Custom Panels

You can of course **make your own panel and load it**.

If you made a UI file named `my_panel.ui` and a HAL file named `my_panel.hal`, you would then load this from a terminal with:

```
halrun -I -f my_panel.hal
```

### Example HAL file loading a QtVCP panel

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime programs
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # (1)

# add components to thread
addf classicladder.0.refresh thread1


# connect pins
net bit-input1     test_panel.checkbox_1        classicladder.0.in-00
net bit-hide       test_panel.checkbox_4        classicladder.0.hide_gui

net bit-output1    test_panel.led_1             classicladder.0.out-00

net s32-in1        test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

(1) In this case we load `qtvcp` using **`-Wn`** which waits for the panel to finish loading before continuing to run the next HAL command.
This is to *ensure that the panel created HAL pins are actually done* in case they are used in the rest of the file.

### 12.5.4 Build A Simple Clean-sheet Custom Screen

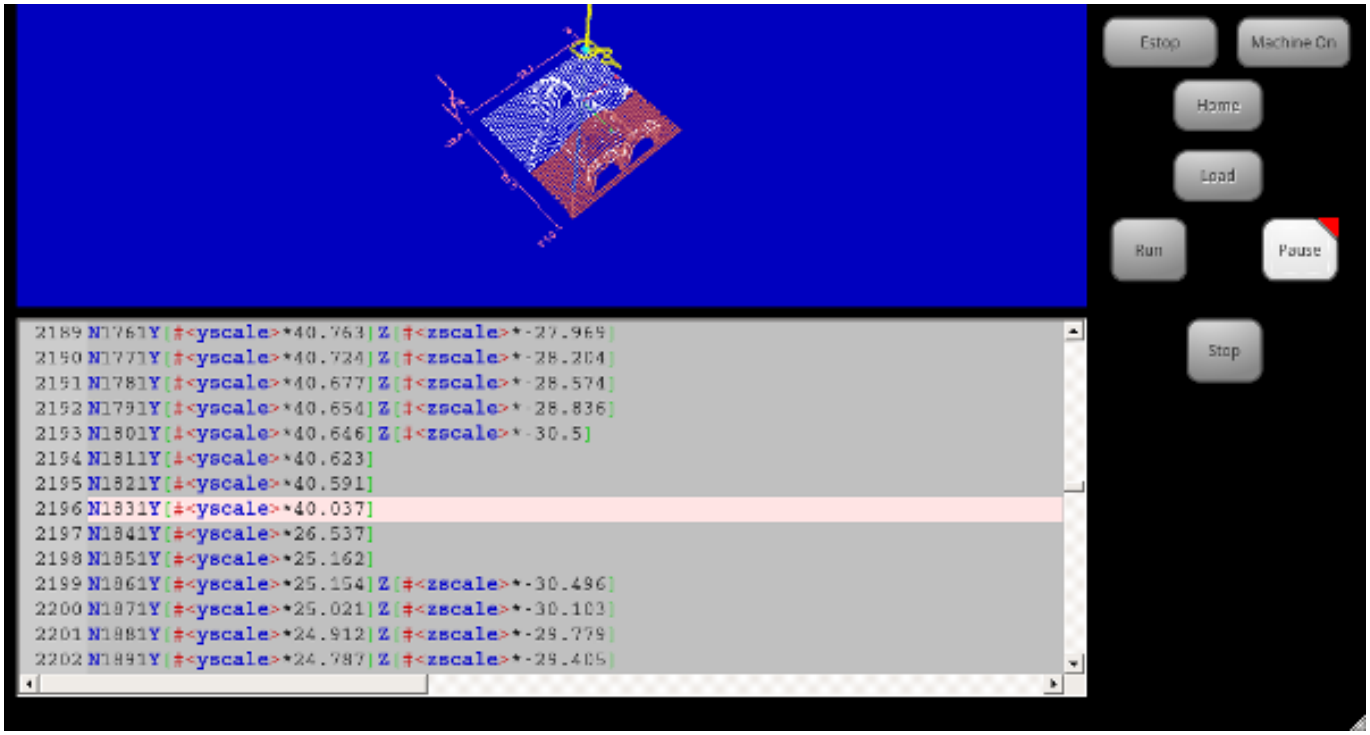



Figure 12.66: QtVCP Ugly custom screen

#### 12.5.4.1 Обзор

To build a panel or screen:

- Use Qt Designer to build a design you like and save it to your configuration folder with a name of your choice, ending with `.ui`
- Modify the configuration INI file to load QtVCP using your new `.ui` file.
- Then connect any required HAL pins in a HAL file.

#### 12.5.4.2 Get Qt Designer To Include LinuxCNC Widgets

**Install Qt Designer** First you must have the **Qt Designer installed**.
The following commands should add it to your system, or use your package manager to do the same:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

**Add qtvcp_plugin.py link to the Qt Designer Search Path** Then you must add a link to the `qtvcp_plugin.py` in one of the folders that Qt Designer will search into.

In a *RIP* version of LinuxCNC `qtvcp_plugin.py` will be:

`'~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py'`

For a *Package installed* version it should be:

```
'usr/lib/python2.7/qtvcp/plugins/qtvcp_plugin.py' or
'usr/lib/python2.7/dist-packages/qtvcp/plugins/qtvcp_plugin.py'
```

Make a symbolic link to the above file and move it to one of the places Qt Designer searches in.

Qt Designer searches in these two place for links (pick one):

```
'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' or
'~/.designer/plugins/python'
```

You may need to create the `plugins/python` folder.

Start Qt Designer:

- For a *RIP install*:
  Open a terminal, set the environment for LinuxCNC <1>, then load Qt Designer <2> with :

```
. scripts/rip-environment    (1)
designer -qt=5               (2)
```

- For a *package install*:
  Open a terminal and type:

```
designer -qt=5
```

If all goes right, Qt Designer will launch and you will see the selectable LinuxCNC widgets on the left hand side.

#### 12.5.4.3 Build The Screen `.ui` File

**Create MainWindow Widget** When Qt Designer is first started there is a *'New Form' dialog* displayed.
Pick *'Main Window'* and press the *'Create'* button.
A *`MainWindow` widget* is displayed.

We are going to make this window a specific non resizeable size:

Set `MainWindow` Minimum and Maximum Size

- Grab the corner of the window and resize to an appropriate size, say 1000x600.

- Right click on the window and click set *minimum size*.
- Do it again and set *maximum size*.

Our sample widget will now not be resizable.

**Add the ScreenOptions Widget** Drag and drop the `ScreenOptions` widget anywhere onto the main window.

This widget doesn't add anything visually but sets up some **common options**.

It's recommended to always *add this widget before any other*.

Right click on the main window, not the `ScreenOptions` widget, and set the *layout* as vertical to make the `ScreenOptions` fullsized.

**Add Panel Content** On the right hand side there is a panel with tabs for a *Property editor* and an *Object inspector*.

On the Object inspector click on the *ScreenOptions*.
Then switch to the Property Editor and, under the *ScreenOptions* heading, toggle **`filedialog_option`**.

Drag and drop a **`GCodeGraphics`** *widget* and a **`GcodeEditor`** *widget*.
Place and resize them as you see fit leaving some room for buttons.

**Add Action Buttons** Add 7 action buttons on to the main window.

If you double click the button, you can add text.
Edit the button labels for *Estop*, *Machine On*, *Home*, *Load*, *Run*, *Pause* and *stop*.

Action buttons *default to no action* so we must change the properties for defined functions. You can edit the properties:

- directly in the *property editor* on the right side of Qt Designer, or
- conveniently, left double clicking on the button to launch a *properties dialog* that allows selecting actions while only displaying relevant data to the action.

We will describe the convenient way first:

- Right click the *Machine On* button and select *Set Actions*.
- When the dialog displays, use the combobox to navigate to `MACHINE CONTROLS - Machine On`.
- In this case there is no option for this action so select *OK*.

Now the button will turn the machine on when pressed.

And now the direct way with Qt Designer's property editor:

- Select the *Machine On* button.
- Go to the Property Editor on the right side of Qt Designer.
- Scroll down until you find the *ActionButton* heading.
- Click the `machine_on` action checkbox you will see in the list of properties and values.

The button will now control machine on/off.

Do the same for all the other button with the addition of:

- With the *Home* button we must also change the `joint_number` property to `-1`.
  This tells the controller to *home all the axes* rather then a specific axis.
- With the *Pause* button:
  - Under the `Indicated_PushButton` heading check the `indicator_option`.
  - Under the `QAbstactButton` heading check `checkable`.

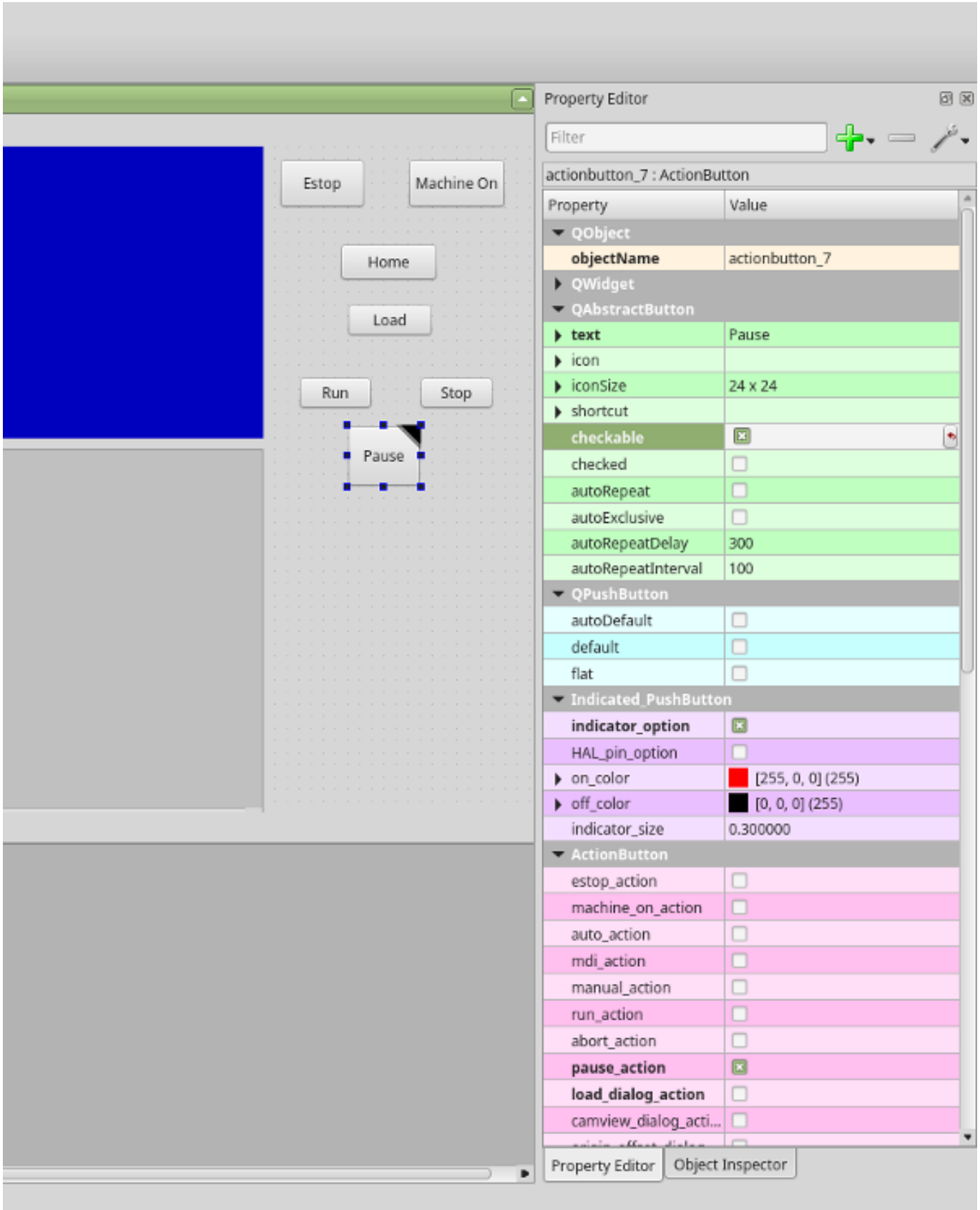



Figure 12.67: Qt Designer: Selecting Pause Button's Properties

**Save The .ui File** We then need to save this design as `tester.ui` in the `sim/qtvcp` folder.

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 file

A handler file is **required**.

It allows customizations to be written in Python.

For instance, *keyboard controls* are usually written in the handler file.

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** If you are using QtVCP to make a CNC control screen, under the *INI file* `[DISPLAY]` heading, set:

```
DISPLAY = qtvcp <screen_name>
```

`<screen_name>` is the *base name* of the `.ui` and `_handler.py` files.

In our example there is already a sim configuration called tester, that we will use to display our test screen.

**[HAL] Section** If your screen used *widgets with HAL pins*, then you must **connect them in a HAL file**.

QtVCP looks in the *INI file*, under the `[HAL]` heading for the entries below:

**`POSTGUI_HALFILE=<filename>`**
: Typically `<filename>` would be `+<screen_name>_postgui.hal+`, but can be any legal filename.
You can have *multiple `POSTGUI_HALFILE` lines* in the INI: each will be run one after the other in the order they appear.
These commands are *executed after the screen is built*, guaranteeing the widget HAL pins are available.

**`POSTGUI_HALCMD=<command>`**
: `<command>` would be *any valid HAL command*.
You can have *multiple `POSTGUI_HALCMD` lines* in the INI: each will be run one after the other in the order they appear.
To guaranty the widget HAL pins are available, these commands are executed:

- *after the screen is built*,
- *after all the POSTGUI_HALFILEs are run*.

In our example there are no HAL pins to connect.

### 12.5.5 Handler File In Detail

Handler files are used to *create custom controls using Python*.

#### 12.5.5.1 Обзор

Here is a sample handler file.

It's broken up in sections for ease of discussion.

```
############################
# **** 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

##########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
##########################################

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

This section is for **importing required library modules** for your screen.

It would be typical to import QtVCP's *keybinding*, *Status* and *Action* libraries.

#### 12.5.5.3 INSTANTIATE LIBRARIES Section

By instantiating the libraries here we **create global reference**.

You can note this by the commands that don't have `self.` in front of them.

By convention we *capitalize the names of globally referenced libraries*.

#### 12.5.5.4 HANDLER CLASS Section

The **custom code** is placed *in a class so QtVCP can utilize it*.

This is the definitions of the handler class.

#### 12.5.5.5 INITIALIZE Section

Like all Python libraries the **+`__init__`+ function** is called when the library is *first instantiated*.

This is where you would set up *defaults*, as well as *reference variables* and *global variables*.

The widget references are not available at this point.

The variables `halcomp`, `widgets` and `paths` give access to QtVCP's HAL component, widgets, and path info respectively.

#### 12.5.5.6 SPECIAL FUNCTIONS Section

There are several *special functions* that QtVCP looks for in the handler file. If QtVCP finds these it will call them, if not it will silently ignore them.

**`class_patch__(self):`**
*Class patching*, also known as *monkey patching*, allows to **override function calls in an imported module**.
Class patching must be done *before the module is instantiated*, and it *modifies all instances* made after that.
An example might be patching button calls from the G-code editor to call functions in the handler file instead.
Class patching function redefined here, will be called with the HandlerClass instance as *self* rather than the patched class instance. This can make access to the patched class function/variables more difficult.
When class patching outside of the HandlerClass class, the function call will use the patched class instance as *self*.

**`initialized__(self):`**
This function is *called after the widgets and HAL pins are built*.
You can manipulate the widgets and HAL pins or add more HAL pins here.
Typically there can be

- preferences checked and set,
- styles applied to widgets,
- статус LinuxCNC, подключенного к функциям.
- keybindings would be added.

**`pre_hal_init__(self):`**
This function is called before the HAL-ified widgets have their hal_init_ function called.
Some property changes need to be done before HAL_init is called on the widget.

**`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):`**
This function gives **raw key press events**.
It takes *precedence over* the `processed_key_event`.

**`keyrelease_event__(receiver, event):`**
This function gives **raw key release events**.
It takes *precedence over* the `processed_key_event`.

**`before_loop__(self):`**
This function is *called just before the Qt event loop is entered*. At that point, all widgets/libraries/initialization code has completed and the screen is already displayed.

**`system_shutdown_request__(self):`**
If present, this function **overrides the normal function called for total system shutdown**.
It could be used to do *pre-shutdown housekeeping*.

**Bibliography**

[1] + 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):`**
This function is *called just before the screen closes*. It can be used to do cleanup before closing.

#### 12.5.5.7 STATUS CALLBACKS Section

By convention this is where you would put functions that are **callbacks from STATUS definitions**.

#### 12.5.5.8 CALLBACKS FROM FORM Section

By convention this is where you would put functions that are **callbacks from the widgets connected to the MainWindow** in the Qt Designer editor.

#### 12.5.5.9 GENERAL FUNCTIONS Section

By convention this is where you put your **general functions**.

#### 12.5.5.10 KEY BINDING Section

If you are *using the `keybinding` library* this is where you place your **custom key call routines**.

The function signature is:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

KEY being the code (from the keybindings library) for the desired key.

#### 12.5.5.11 CLOSING EVENT Section

Putting the **`closeEvent` function here will catch closing events**.

This *replaces any predefined `closeEvent`* function from QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

---

**Note**
It is usually better to use the special `closing_cleanup__` function.

---

## 12.5.6 Connecting Widgets to Python Code

It is possible to connect widgets to Python code using **signals and slots**.

In this way you can:

- *Give new functions to LinuxCNC widgets*, or
- *Utilize standard Qt widgets to control LinuxCNC*.

### 12.5.6.1 Обзор

**In the Qt Designer editor**:

- You *create user function slots*
- You *connect the slots to widgets using signals*.

**In the handler file**:

- 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()`.
- Press the *OK* button.
- 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.

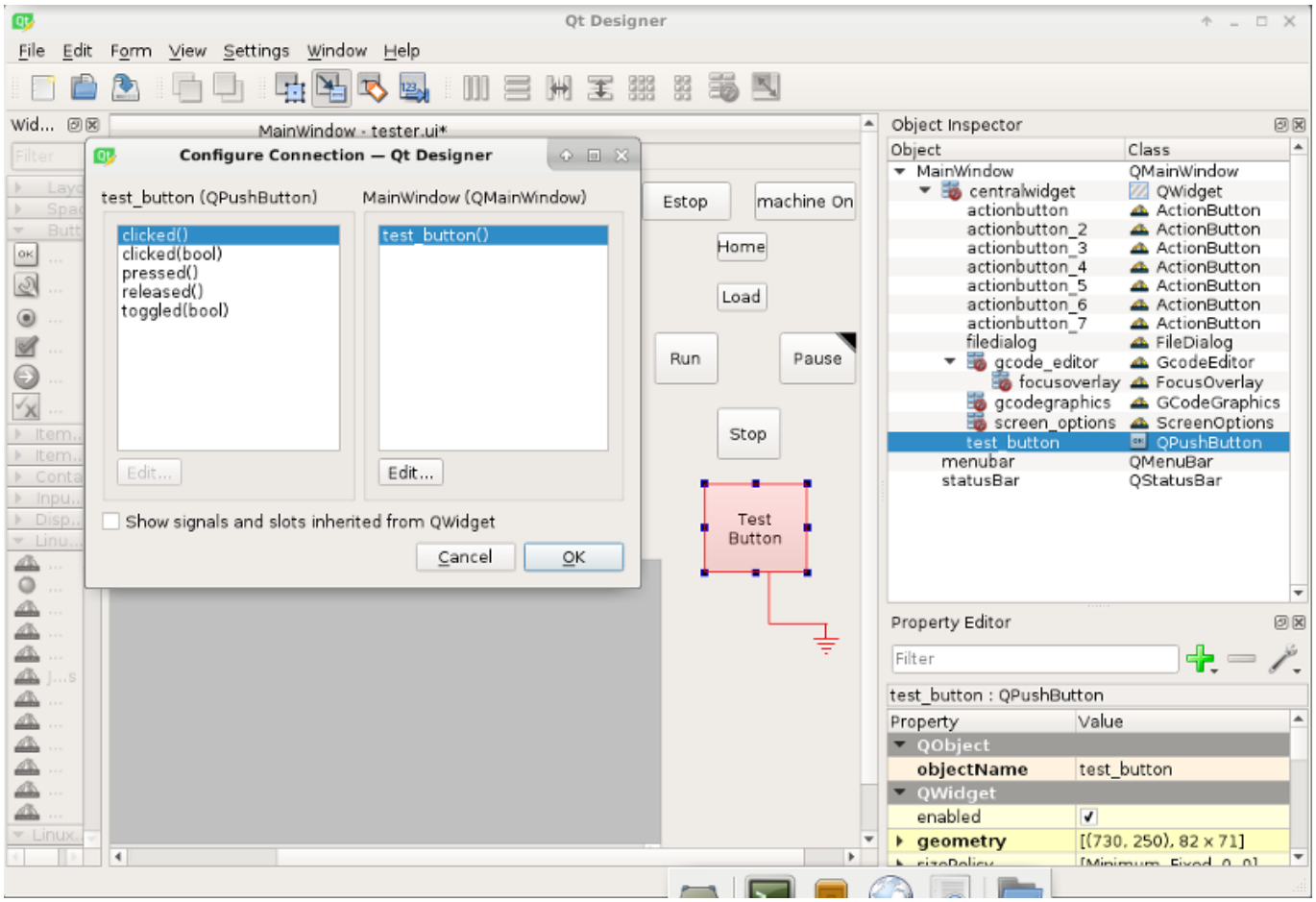



Figure 12.68: Qt Designer Signal/Slot Selection

#### 12.5.6.3 Python Handler Changes

Now you must **add the function to the handler file**.

The function signature is **`def slot_name(self):`**.

For our example, we will add some code to print the widget name:

```
def test_button(self):
    name = self.w.sender().text()
    print(name)
```

Add this code under the section named:

```
########################
# callbacks from form #
########################
```

In fact it doesn't matter where in the handler class you put the commands but by convention this is where to put it.

Save the handler file.

Now when you load your screen and press the button it should print the name of the button in the terminal.

### 12.5.7 More Information

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 Virtual Control Panels

QtVCP can be used to **create control panels** that interface with *HAL*.

### 12.6.1 Builtin Virtual Control Panels

There are several **builtin HAL panels** available.

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
```

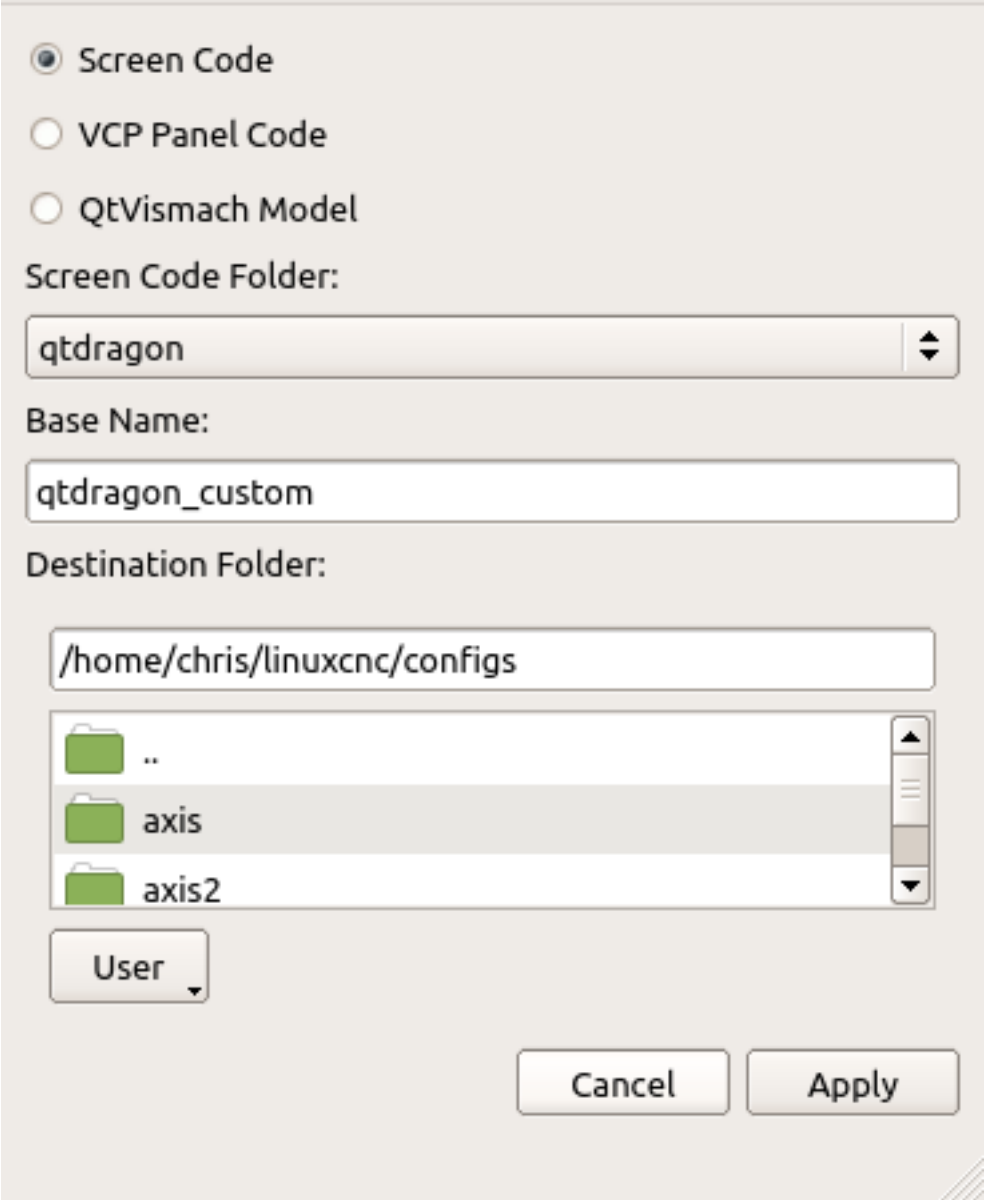
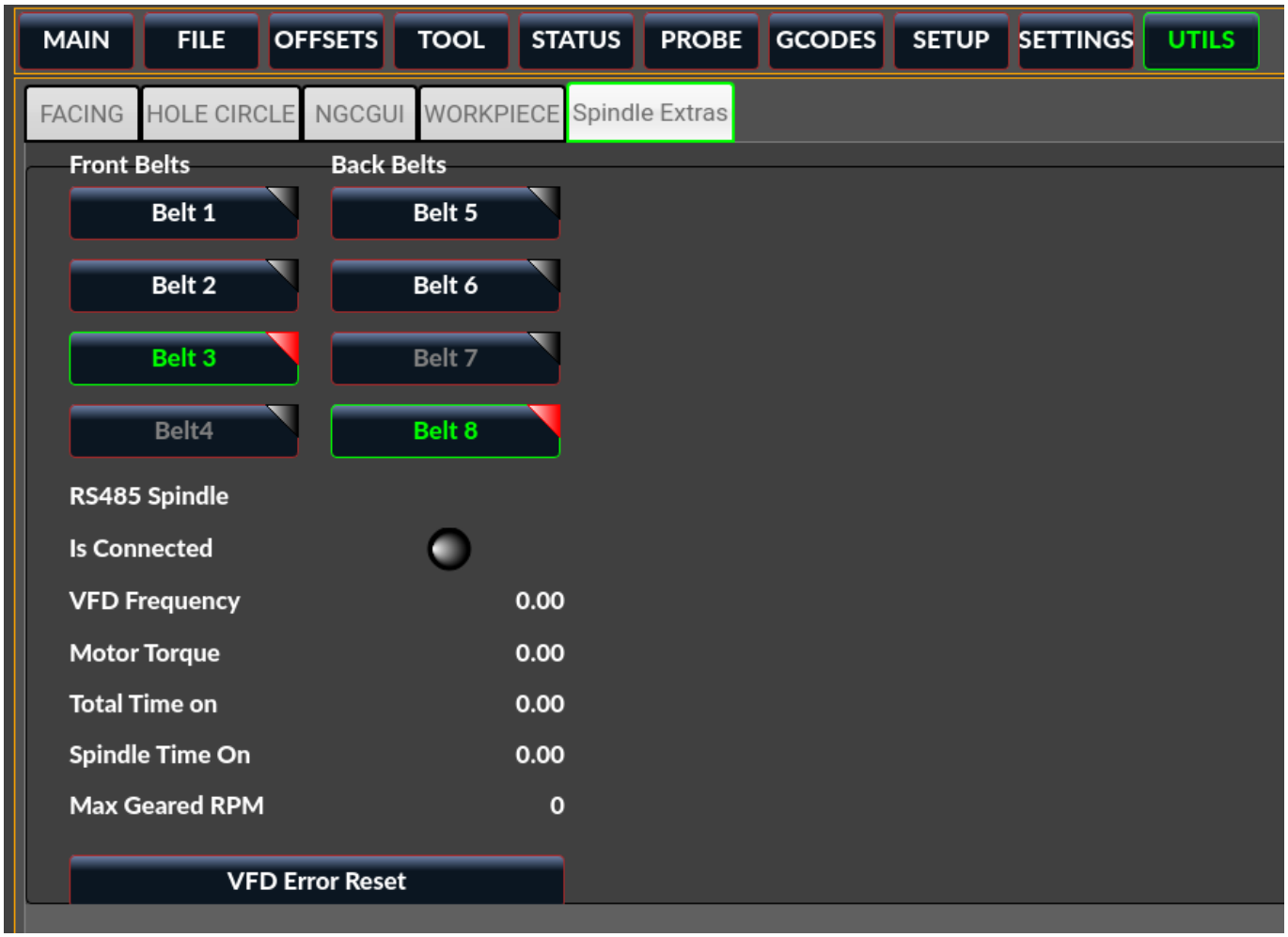



Figure 12.69: QtVCP `copy` Dialog - Screen, VCP Panel or QtVismach Code Copy Panel

#### 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`

- This panel has a **dial that adjusts S32 and Float HAL output pins**.
- The dial's range can be adjusted from a drop down menu.
- The output can be scaled with the `spinbox`.
- A `combobox` can be used to automatically select and connect to a signal.

```
loadusr qtvcp test_dial
```

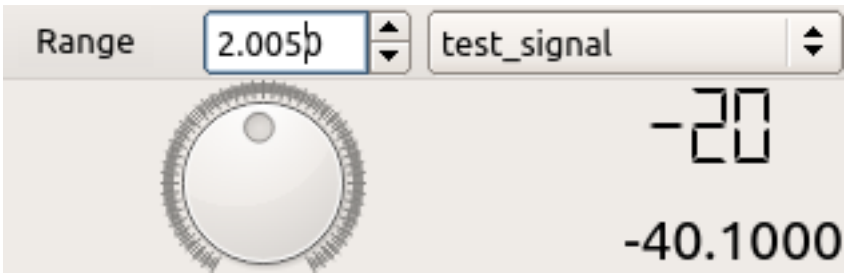



Figure 12.70: QtVCP `test_dial` Panel - Test Dial VCP

#### 12.6.1.4 test_button

- This panel has a **button that will set a HAL pin**.
- The button can be selected as a *momentary* or a *toggle* button.
- A HAL pin will be created that follows the button state.
- The button's *indicator color* can be adjusted from a drop down menu.
- A HAL pin or signal can be selected to follow the button state.
- 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`.

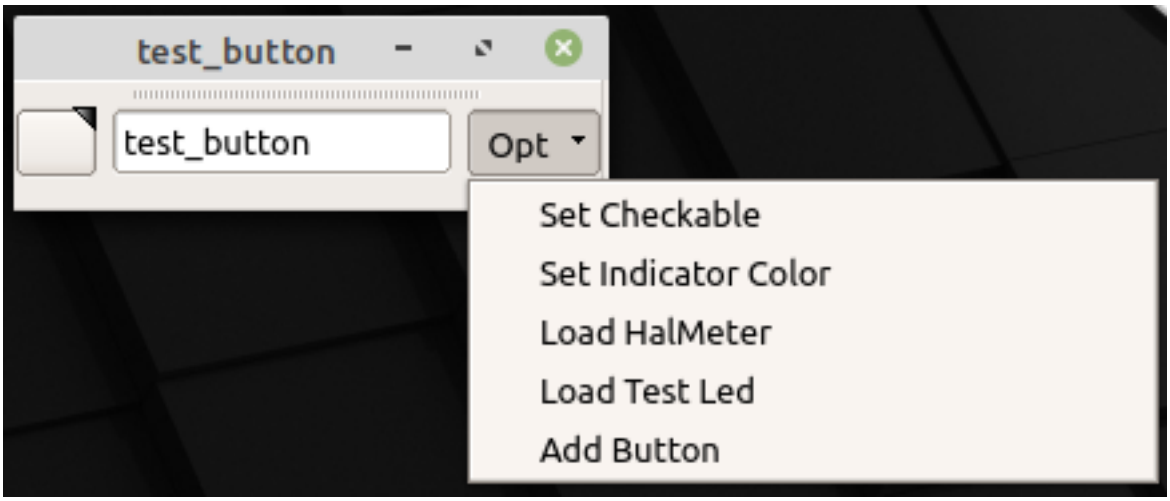



Figure 12.71: QtVCP `test_button` - Test Button VCP

#### 12.6.1.5 test_led

- This panel has an **LED that can selected to watch HAL bit pins/signals**.
- The LED's color can be adjusted from a drop down menu.
- The text box and state can be output as speech if sound is selected.
- A `combobox` can be used to automatically select and connect to a pin/signal.
- You can add more LEDs from the drop down menu.
- The LED can be detached from the main windows.

Here is how to load `test_led` from a HAL script:

```
loadusr qtvcp test_led
loadusr qtvcp -o 4 test_led
```

The `-o` switch sets how many LEDs the panel starts with.
If loading directly from a terminal omit the *loadusr*.

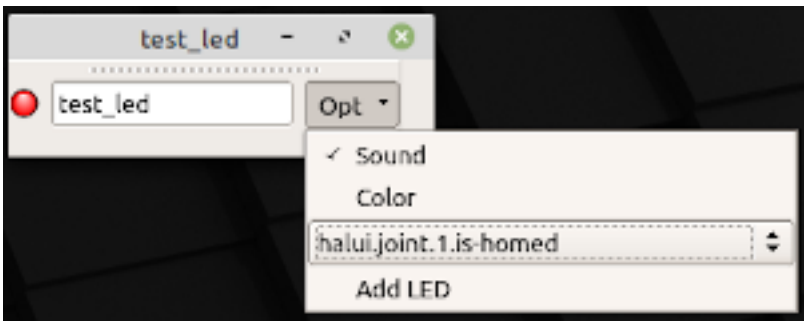



Figure 12.72: QtVCP `test_dial` Panel - Test LED VCP

#### 12.6.1.6 `test_panel`

**Collection of useful widgets for testing HAL component**, including speech of LED state.

```
loadusr qtvcp test_panel
```

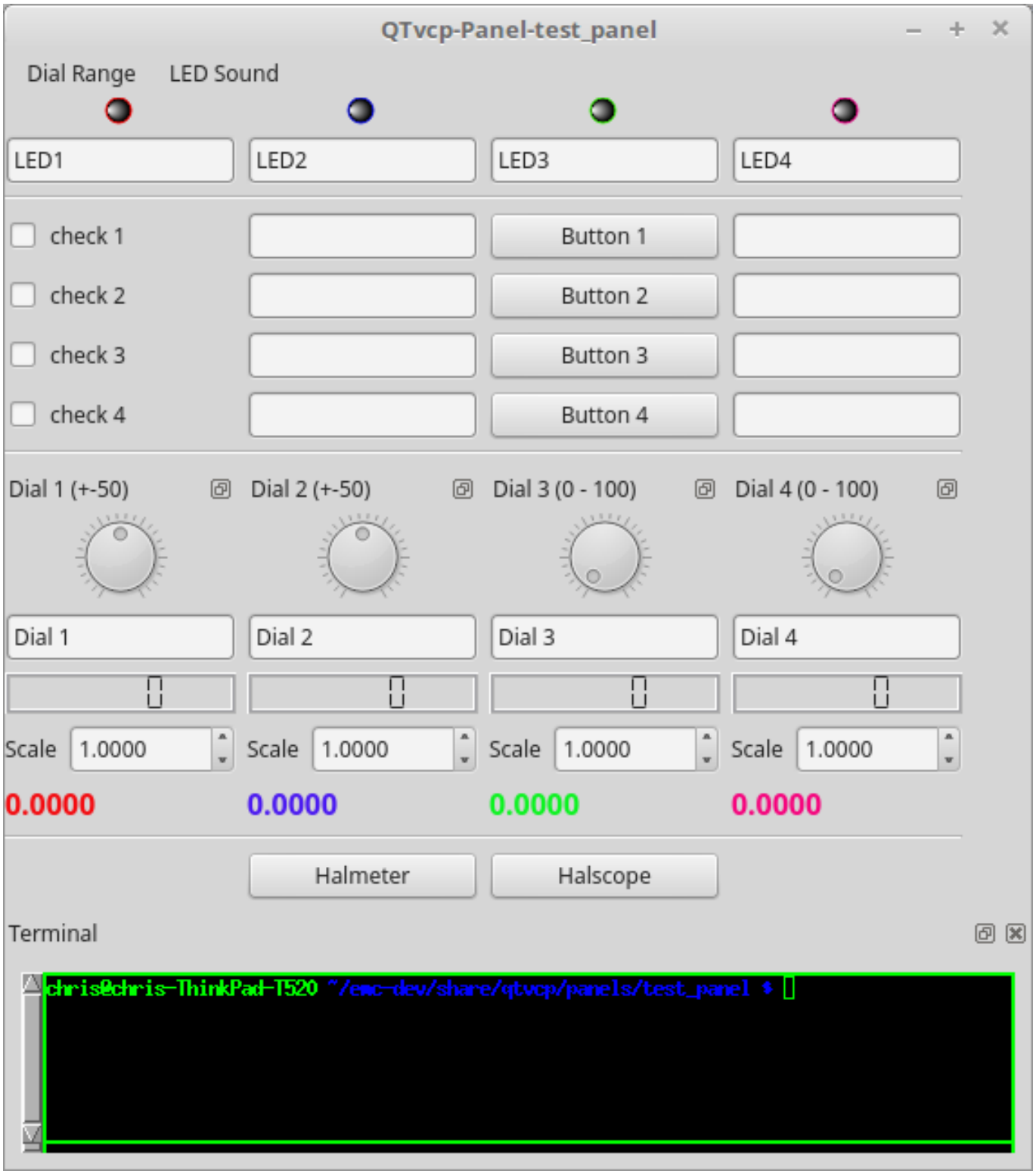



Figure 12.73: QtVCP `test_panel` - HAL Component Testing Panel

#### 12.6.1.7 `cam_align`

A **camera display widget for rotational alignment**.

Figure 12.74: QtVCP `cam_align` Panel - Camera Based Alignment Panel

**Использование** 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
```

**Note**
All <options> must appear before the `cam_align` panel name.

Qtvcp варианты

- `-c NAME` HAL component name. Default is to use the UI file name.

- `-d` Debugging on. or remove for no minimum output
- `-v` Verbose debugging on. You can find all available resolutions.
- `-x {XID}` used for embedding into AXIS or Gmoccapy
- `-o <option>` Options passed to cam_align. can use multiple -o entries

**Cam_align Options** Доступны следующие опции -o:

- `size=400,400` Size of the embedded window (width,height)
- `imagesize=300,300` Size of the image inside the window (width,height)
- `rotincr=5` Sets the increment of the crosshair rotation. (degrees)
- `xscale=100` Scales the image in X. A negative value will flip the image in X (percent)
- `yscale=100` Scales the image in Y. A negative value will flip the image in Y (percent)
- `camnumber=1` Sets what system camera to use
- `api=V4L2` Sets the opencv backend library for the camera
- `res=1280,720` Sets the requested resolution (width,height)

For instance 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_CAM_ALIGN1=G10 L20 P1 X0 Y0,Set XY\nOrigin
MDI_COMMAND_CAM_ALIGN2=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
```

Figure 12.75: QtVCP `sim_panel` - Simulated Controls Panel For Screen Testing.

#### 12.6.1.9 `tool_dialog`

**Manual tool change dialog** that gives tool description.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Options:

- `-o notify_on` - *use desktop notify* dialogs instead of QtVCP native ones.
- `-o audio_on` - *play sound on tool change*
- `-o speak_on` - *speak announcement of tool change*

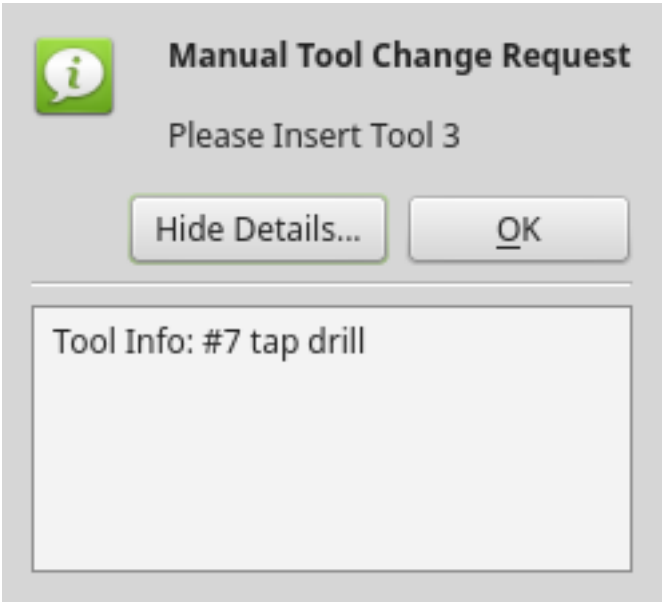



Figure 12.76: QtVCP `tool_dialog` - Manual Tool Change Dialog

### 12.6.2 `vismach` 3D Simulation Panels

These panels are prebuilt simulation of common machine types.
These are also embed-able in other screens such as AXIS or GMOCCAPY.

#### 12.6.2.1 QtVCP `vismach_mill_xyz`

3D OpenGL view of a *3-Axis milling machine*.

```
loadusr qtvcp vismach_mill_xyz
```

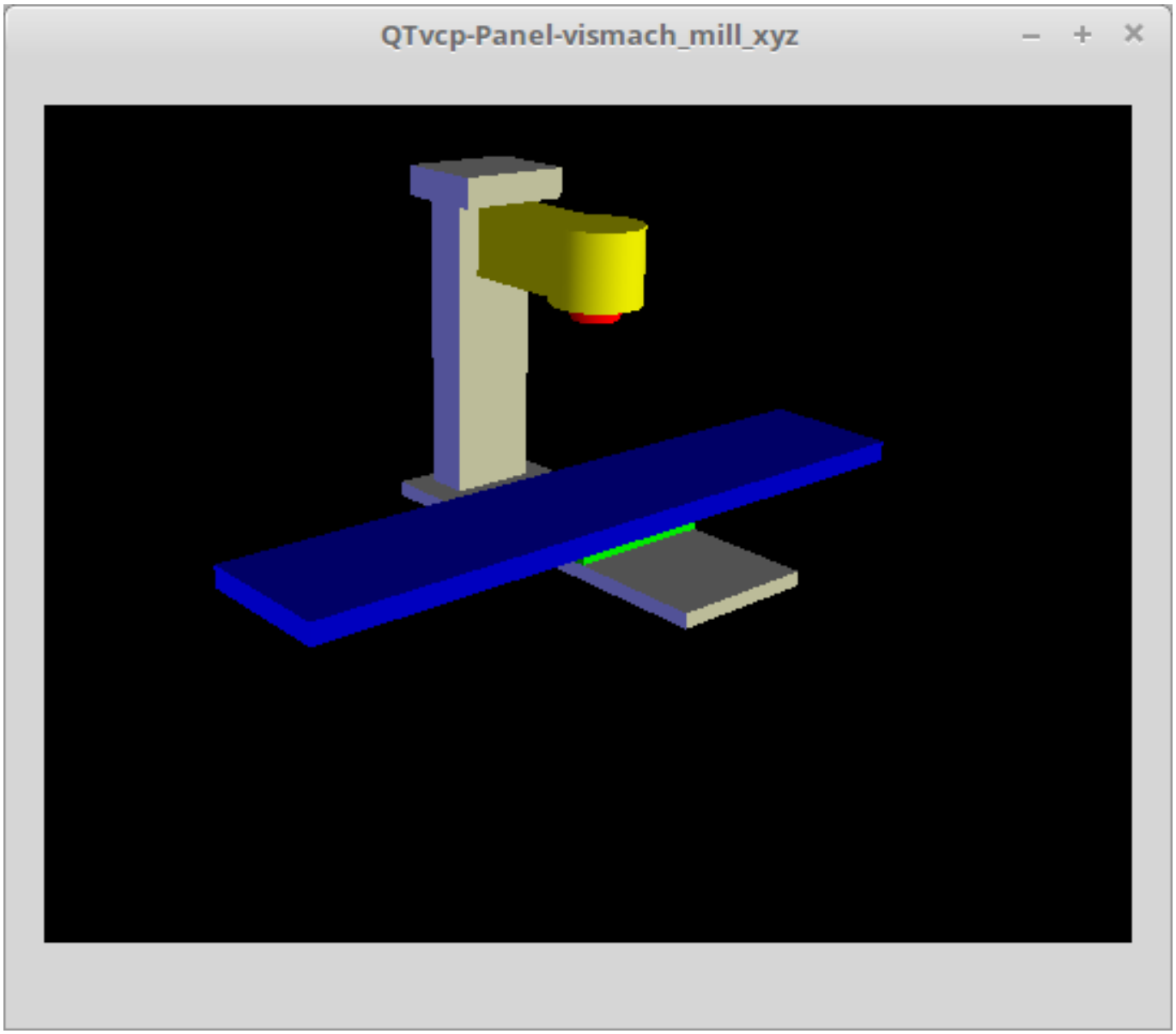



Figure 12.77: QtVCP `vismach_mill_xyz` - 3-Axis Mill 3D View Panel

#### 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
```

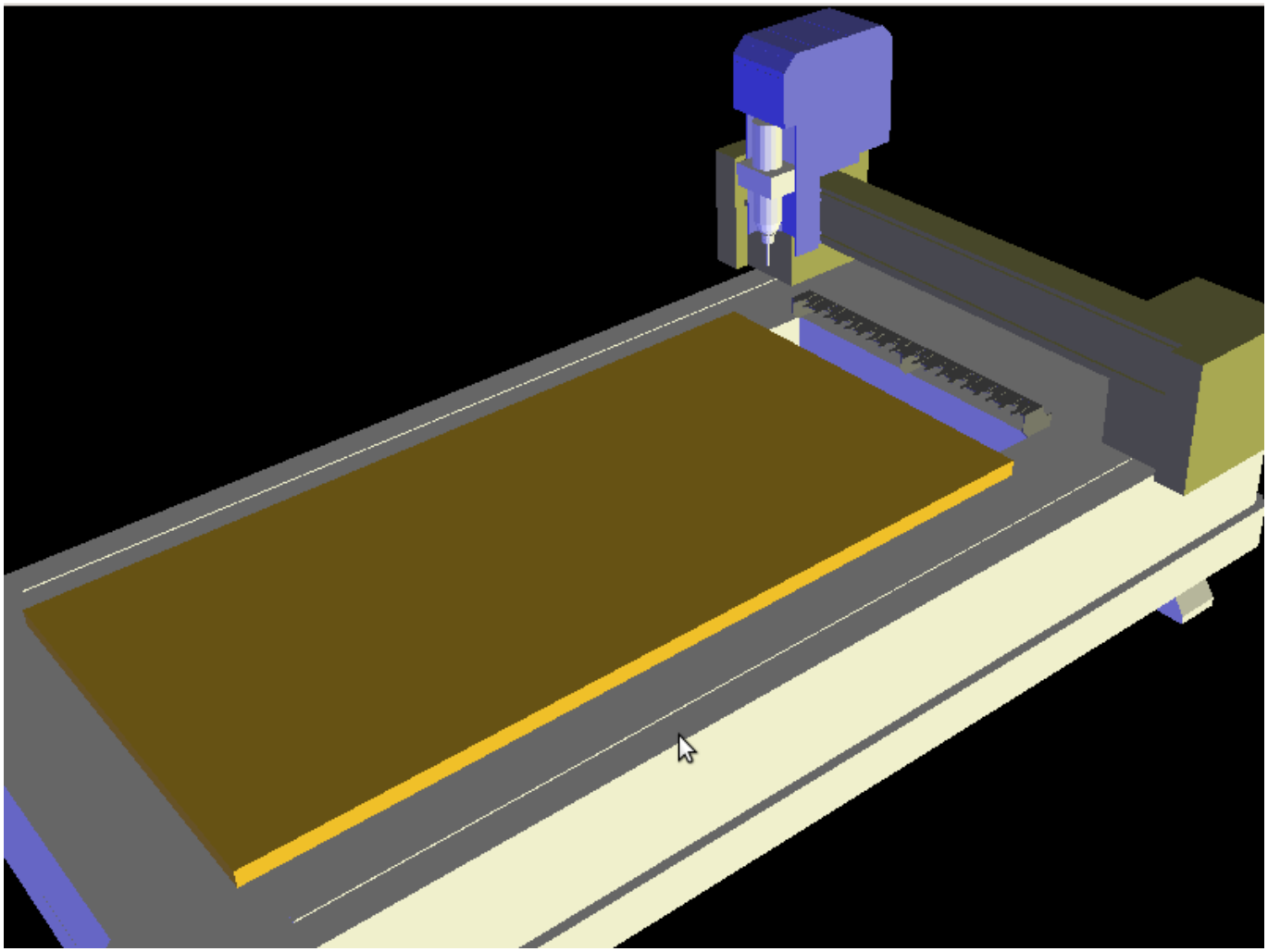


Figure 12.78: QtVCP `vismach_router_atc` - 3-Axis Gantry Bed Mill 3D View Panel

#### 12.6.2.3 QtVCP `vismach_scara`

3D OpenGL view of a *SCARA based milling machine*.

```
loadusr qtvcp vismach_scara
```

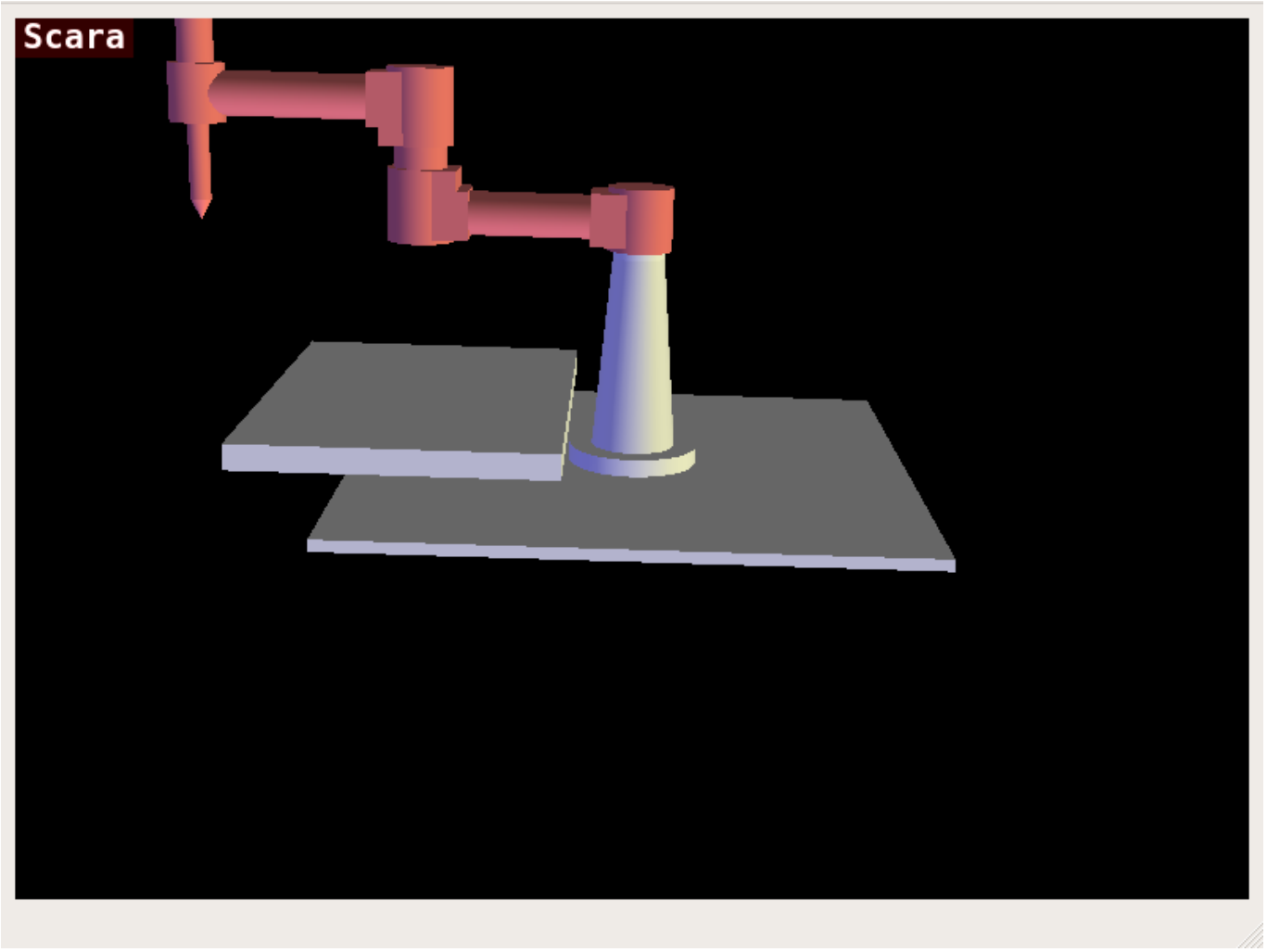



Figure 12.79: QtVCP `vismach_scara` - SCARA Mill 3D View Panel

#### 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
```

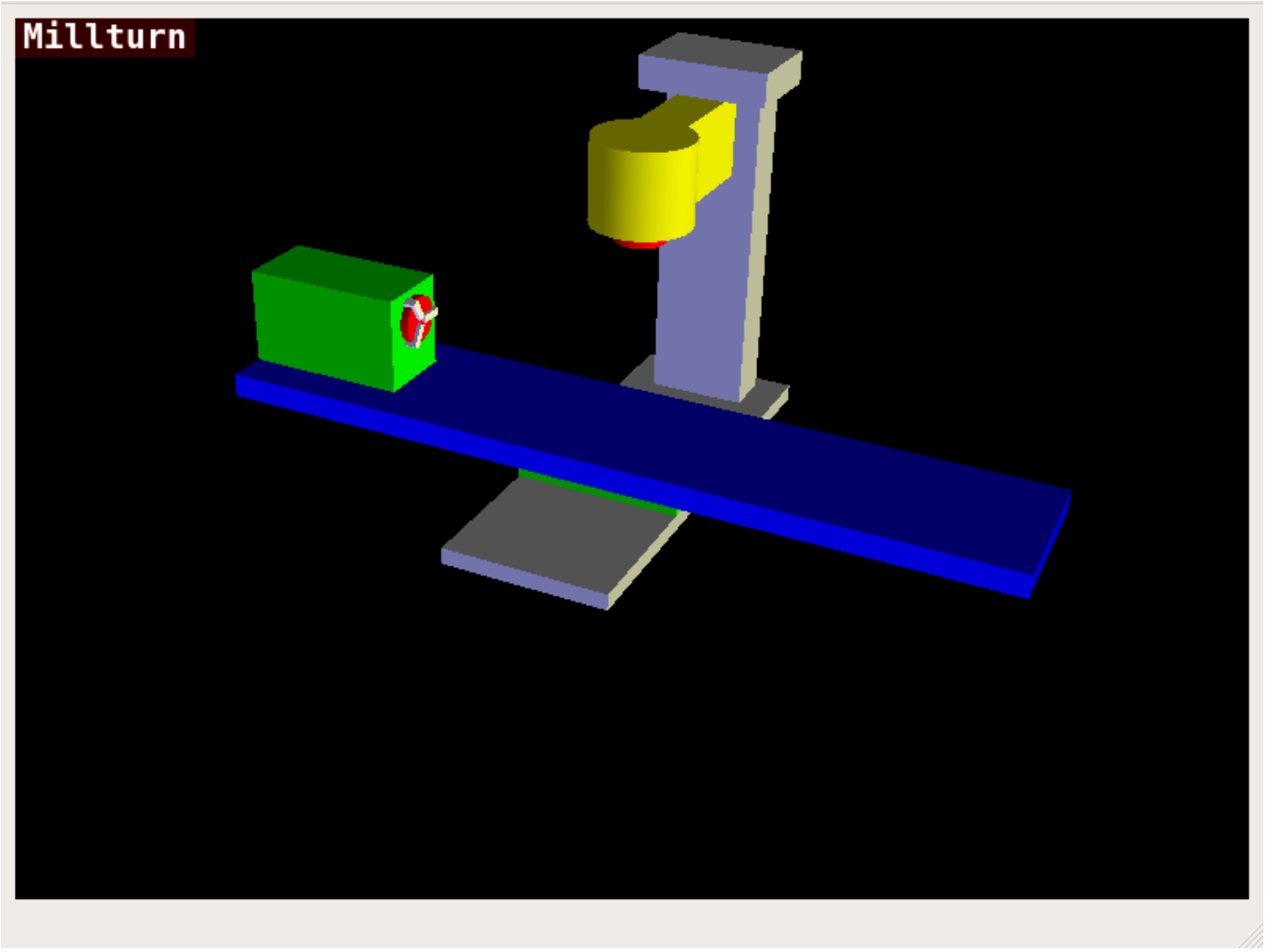



Figure 12.80: QtVCP `vismach_millturn` - 4 Axis MillTurn 3D View Panel

#### 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
```

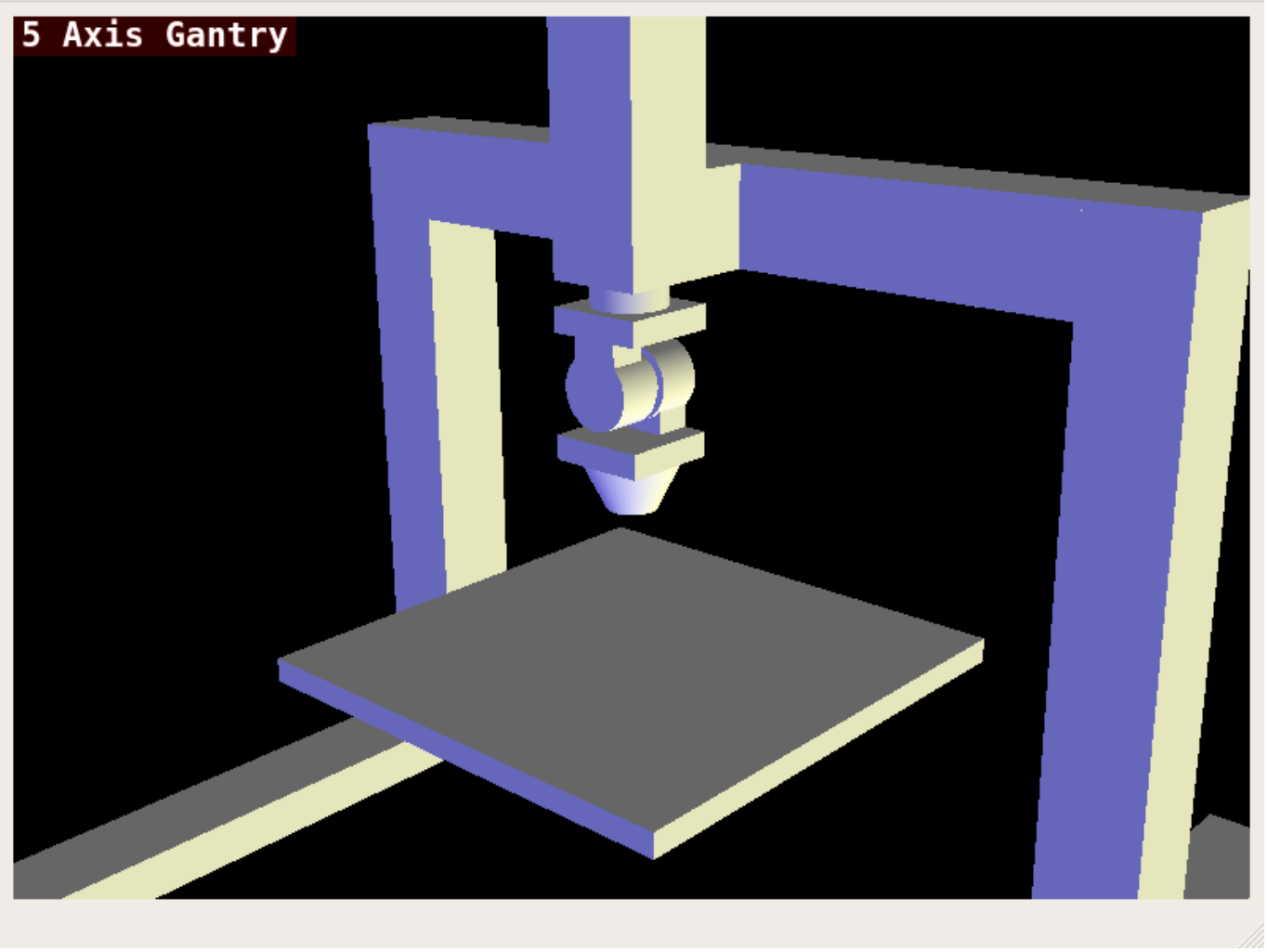



Figure 12.81: QtVCP `vismach_mill_5axis_gantry` - 5-AxIs Gantry Mill 3D View Panel

#### 12.6.2.6 QtVCP `vismach_fanuc_200f`

3D openGL view of a *6 joint robotic arm*.

```
loadusr qtvcp vismach_fanuc_200f
```

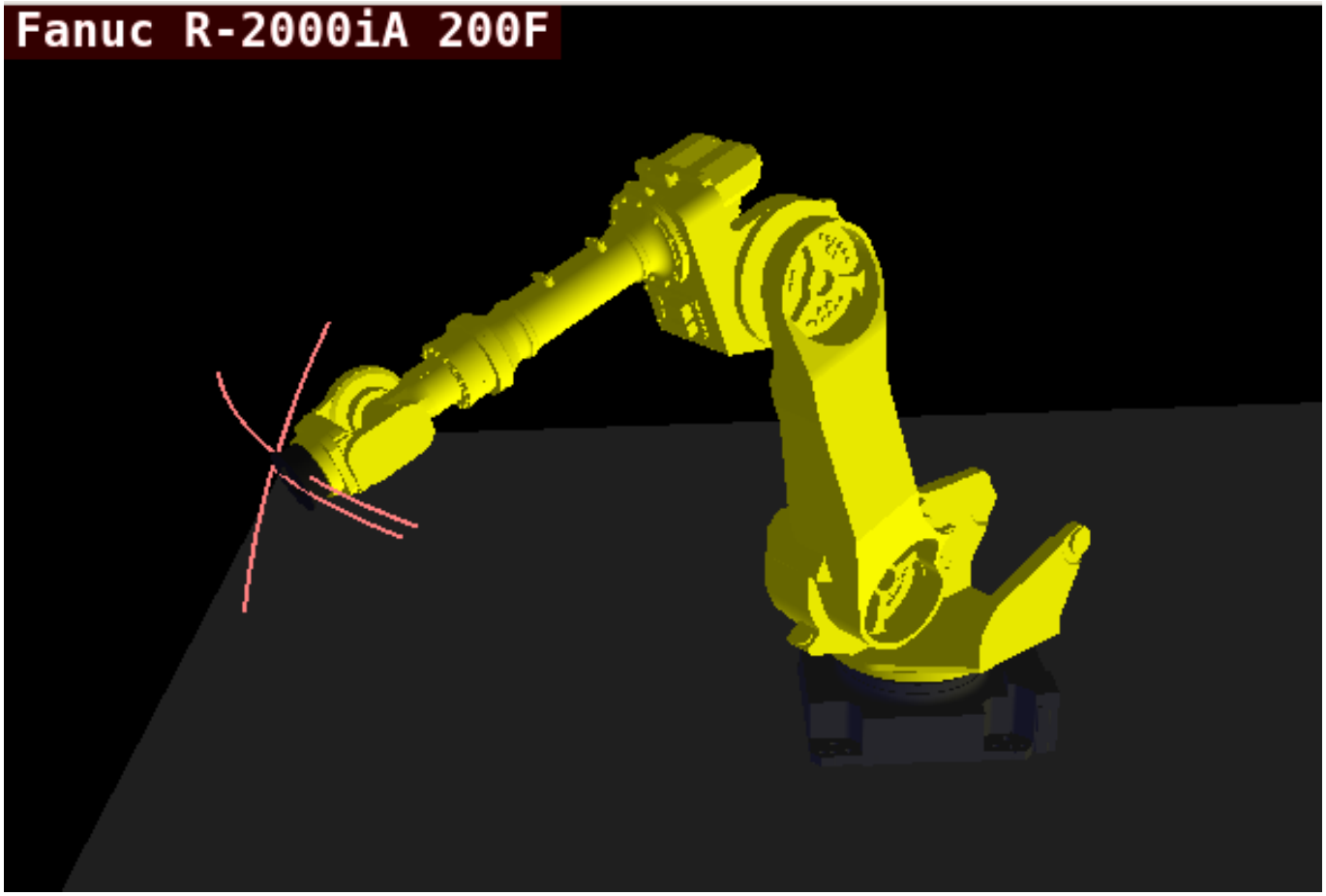



Figure 12.82: QtVCP `vismach_fanuc_200f` - 6 Joint Robotic Arm

### 12.6.3 Custom Virtual Control Panels

You can of course **make your own panel and load it**.

If you made a UI file named `my_panel.ui` and a HAL file named `my_panel.hal`, you would then load this from a terminal with:

```
halrun -I -f my_panel.hal
```

**Example HAL file loading a QtVCP panel**

```
# load realtime components
loadrt threads
loadrt classicladder_rt

# load non-realtime programs
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # (1)

# add components to thread
addf classicladder.0.refresh thread1


# connect pins
net bit-input1     test_panel.checkbox_1        classicladder.0.in-00
```

```
net bit-hide       test_panel.checkbox_4        classicladder.0.hide_gui

net bit-output1    test_panel.led_1             classicladder.0.out-00

net s32-in1        test_panel.doublescale_1-s   classicladder.0.s32in-00

# start thread
start
```

(1), (1) In this case we load `qtvcp` using **`-Wn`** which waits for the panel to finish loading before continuing to run the next HAL command.
This is to *ensure that the panel created HAL pins are actually done* in case they are used in the rest of the file.

## 12.6.4 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 Встраивание команд

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 Советы по программированию обработчиков

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(Tr`

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 - effectively 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,
i.e. 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** uses *QtVCP widgets* for LinuxCNC integration.

**Widget** is the general name for the *UI objects* such as buttons and labels in PyQt.

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 two, heading on the right side of the editor:

- One is for **HAL only widgets**.
- The other is for **CNC control widgets**.

You are free to mix them in any way on your panel.

**Note**
This description of widget properties can easily be out of date due to further development and lack of people to write docs (a good way to give back to the project). The definitive descriptions are found by looking in the source code.

## 12.7.1 HAL Only Widgets

These widgets usually have *HAL pins* and **don't react to the machine controller**.

### 12.7.1.1 `CheckBox` Widget

This widget allows the user to **check a box to set a HAL pin true or false**.

It is based on PyQt's *QCheckButton*.

### 12.7.1.2 *DetachTabWidget* - Container Widget With User Detachable Panels

This container widget works just like a QTabWidget -it displays multiple panels one at a time with tabs to select.
If you double click the tab or drag the tab, the tab will detach from the main window.
When a tab is detached, the contents are placed into a QDialog.
The tab can be re-attached by closing the dialog or by double clicking on its window frame.

Он основан на PyQt's QTabWidget.

### 12.7.1.3 `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*.

### 12.7.1.4 `FocusOverlay` - Focus Overlay Widget

This widget places a **colored overlay over the screen**, usually while a dialog is showing.

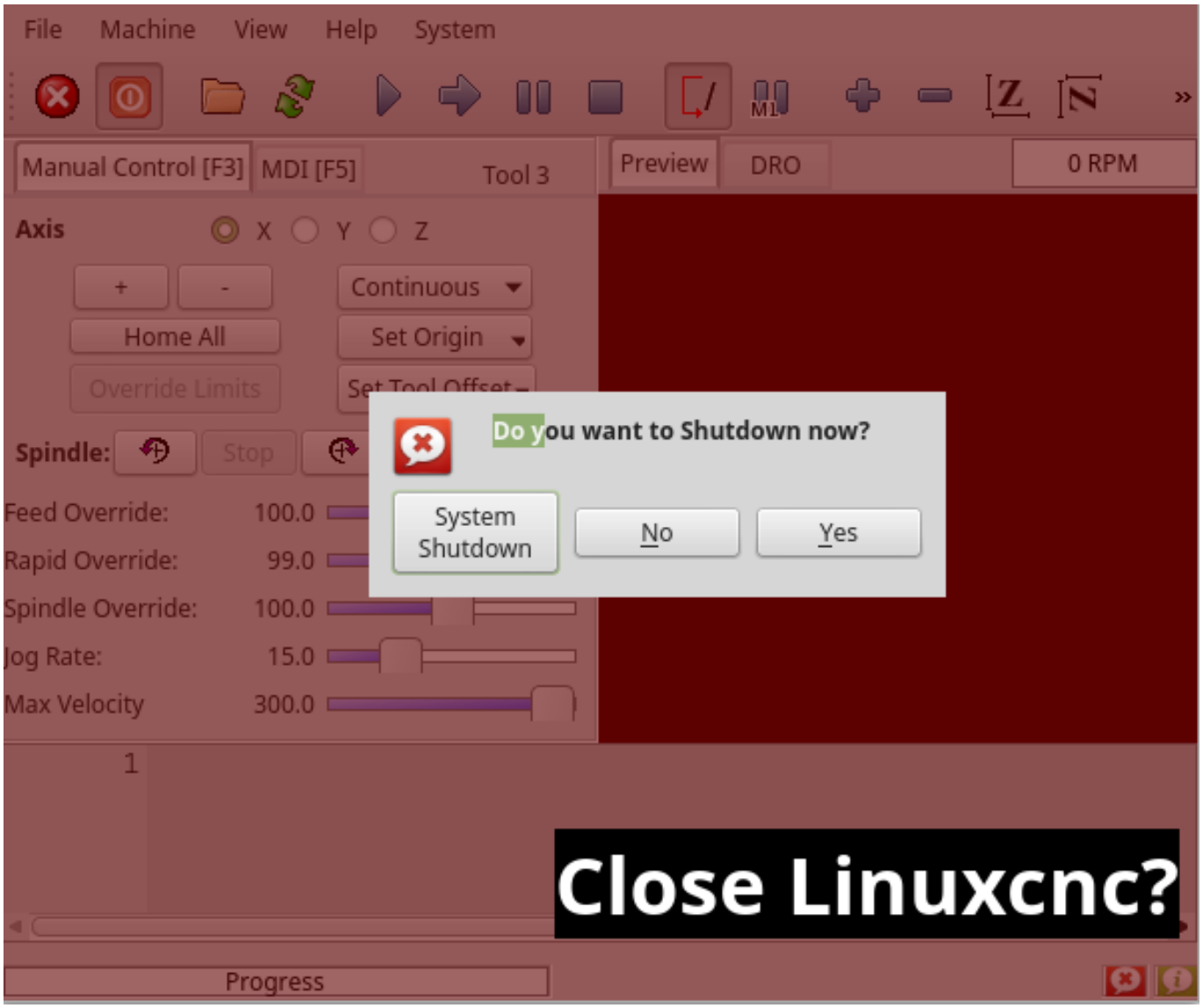



Figure 12.83: Focus overlay example for confirm close prompt

Used to create a *focused* feel and to draw attention to critical information.

It can also show a translucent image.
It can also display message text and buttons.

This widget *can be controlled with STATUS messages*.

#### 12.7.1.5 `Gauge` - Round Dial Gauge Widget

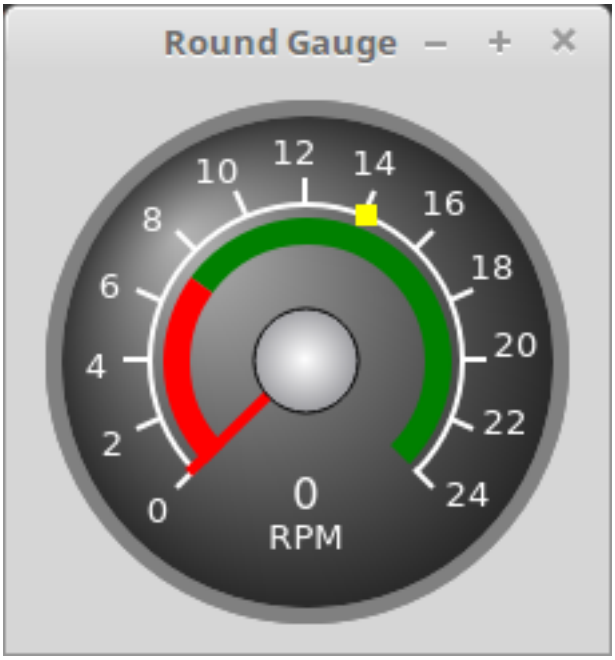



Figure 12.84: QtVCP `Gauge`: Round Dial Gauge Widget

Round Gauge can be used in a LinuxCNC GUI to **display an input parameter** on the dial face.

**Customizable Parameters** There are several properties that are user settable in order to customize the *appearance of the gauge*.

The following parameters can be set either programmatically or via the Qt Designer property editor.

**`halpin_option`**
Setting this to `True` will *create 2 HAL pins*:

- 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`**
This is the *maximum expected value of the value input signal*.
In other words, it is the full scale input.

**`num_ticks`**
This is the *number of ticks/gauge readings* on the gauge face.
It should be set to a number that ensures the text readings around the gauge face are readable.
The minimum allowed value is 2.

**`zone1_color`**
Zone1 extends *from the maximum reading to the threshold point*.
It can be set to any RGB color.

**zone2_color**
Zone2 extends *from the threshold point to the minimum reading,* which is 0.
It can be set to any RGB color.

**bezel_color**
This is the color of the *outer ring of the gauge*.

**bezel_width**
This is the width of the *outer ring of the gauge*.

**threshold**
The threshold is the *transition point between the zones*.
It should be set to a value between 0 and the maximum value.
The maximum allowed value is set to the gauge's `max_value` and minimum value is 0.

**gauge_label**
This is the *text below the value readout,* near the bottom of the gauge.
The function of the gauge is then easily visible.

**base_color**
Цвет шкалы.

**base_gradient_color**
The highlight color of the gauge.

**center_color**
Цвет центра шкалы.

**center_gradient_color**
The highlight color of the center of the gauge.

**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**
This is the *actual input value* that will be displayed with the gauge needle and in the digital readout.
It must be set to a value between 0 and `max_value` maximum value.

**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.

#### 12.7.1.6 `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*.

#### 12.7.1.7 `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*.

#### 12.7.1.8 `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`.

#### 12.7.1.9 *HalBar* - HAL Bar Level Indicator

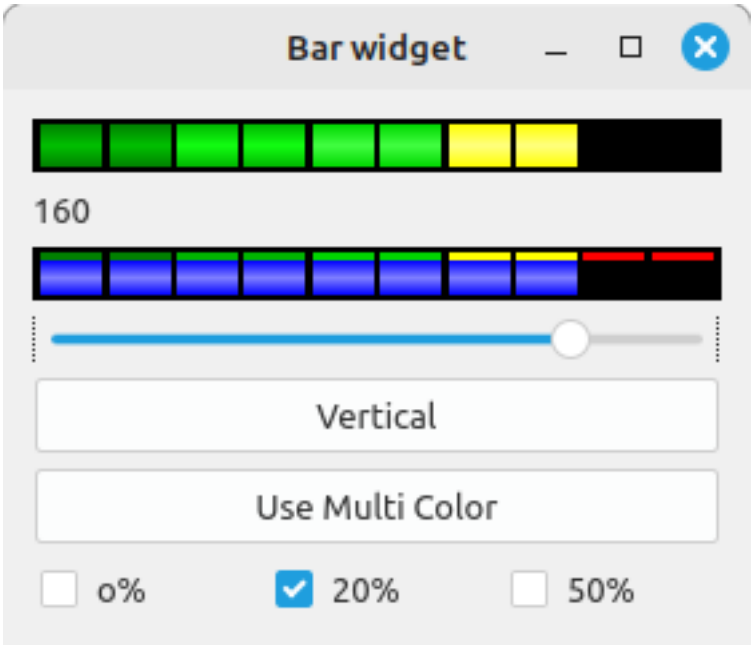



Figure 12.85: QtVCP `HalBar`: Panel demonstrating the HAL Bar Level Indicator

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.
- *indicatorColor*: a QColor definition of the optional single color current value bar.
- *useMultiColorIndicator*: bool switch for choosing the option of single or multicolor value bar.
- *split*: the integer percentage split of max value bar versus current value bar (0 to 50%).
- *setVertical*: bool switch for choosing vertical or horizontal indicator.
- *setInverted*: bool switch for choosing inverted direction.
- *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.

**Note**
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';
}
```

#### 12.7.1.10 `HALPad` - HAL Buttons Joypad

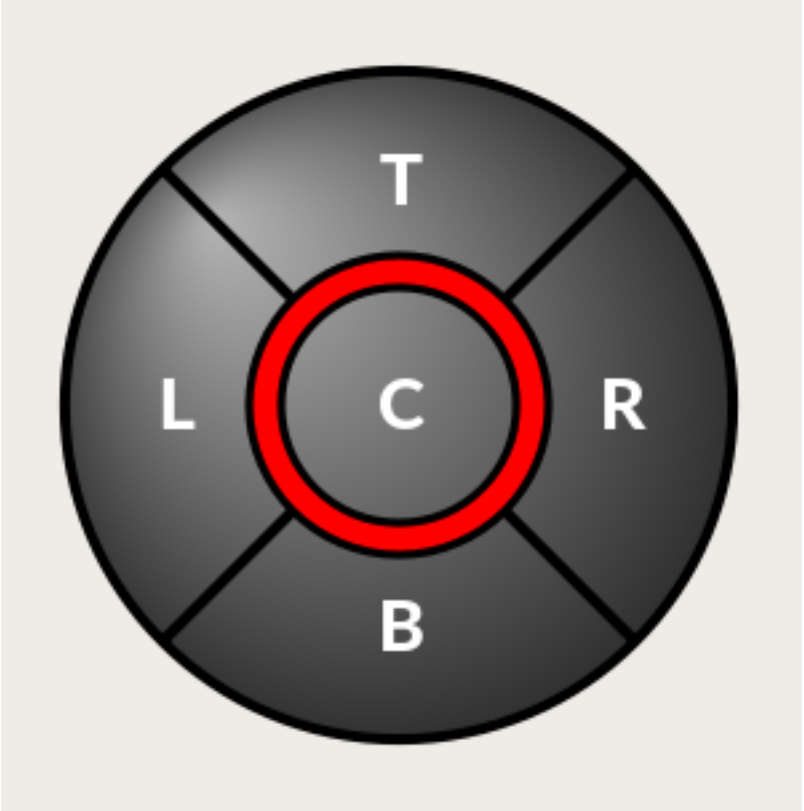



Figure 12.86: QtVCP `HALPad`: HAL Buttons Joypad

This widget looks and acts like a **5 buttons D-pad**, with an LED ring.

Each button has an selectable type (Bit, S32 or Float) output HAL pin.

The LED center ring has selectable colors for off and on and is controlled by a bit HAL pin.

**HALPad ENUMS** There are *enumerated constants* used:

- To reference **indicator positions**:
  - NONE
  - LEFT
  - RIGHT
  - CENTER
  - TOP
  - BOTTOM
  - LEFTRIGHT
  - TOPBOTTOM
- For **HAL pins type**:
  - NONE
  - BIT
  - S32
  - FLOAT

You use the widget name in Qt Designer plus the reference constant:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad Properties

**pin_name**
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**
Color selection for the center LED ring to be displayed when the `<BASENAME>.light.center` HAL pin is `True` or `False`.

**text_color**
Color selection for the button text.

**text_font**
Font selection for the button text.

**HALPad Styles** The above properties could be set in *styles sheets*.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

#### 12.7.1.11 `HALLabel` - HAL Label Widget

This widget **displays values sent to it**.

Values can be sent from:

- *HAL pins*
  The input pin can be selected as Bit, S32, Float or no pin selected
- *Programmatically*
- *A `QtSignal`*

There is a `textTemplate` property to set the rich text and/or to format the text.
Basic formatting might be:

- `%r` for booleans
- `%d` for integers
- `%0.4f` for 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.

There are function calls to display values:

**[HALLabelName].setDisplay(some_value)**
Can be used to set the display if no HAL pin is selected.

**[HALLabelName].setProperty(textTemplate,"%d")**
Sets the template of the display.

It is based on PyQt's *QLabel*.

#### 12.7.1.12 `LCDNumber` - LCD Style Number Readout Widget

This widget *displays HAL float/s32/bit values in a LCD looking way*.

It can display numbers in decimal, hexadecimal, binary and octal formats by setting the **`mode`** property.

When using floats you can set a formatting string.

You must set the **`digitCount`** property to an appropriate setting to display the largest number.

Свойства

**`pin_name`**
: Option string to be used as the HAL pin name.
If set to an empty string the widget name will be used.

**`bit_pin_type`**
: Selects the input pin as type BIT.

**`s32_pin_type`**
: Selects the input pin as type S32.

**`float_pin_type`**
: Select the input pin as type `FLOAT`.

**`floatTemplate`**
: A string that will be used as a Python3 format template to tailor the LCD display.
Only used when a `FLOAT` pin is selected, 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.

It is based on PyQt's *QLCDNumber*.

#### 12.7.1.13 `LED` - Indicator Widget

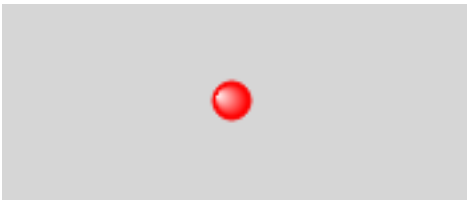


Figure 12.87: QtVCP `LED`: LED Indicator Widget

A **LED like indicator** that optionally follows a HAL pin's logic.

**`halpin_option`**
: Selects if the LED follows an input HAL pin or program state.

**`diameter`**
: Diameter of the LED (defaults to 15).

**`color`**
: Color of the LED when on (defaults to green).

**`off_color`**
: Color of the LED when off (defaults to black).

**gradient**
: turns the gradient high light on or off (defaults to on).

**on_gradient_color**
: Color highlight of the LED when on (defaults to white).

**off_gradient_color**
: Color highlight of the LED when off (defaults to white).

**alignment**
: Qt alignment hint.

**state**
: Current state of LED

**flashing**
: Turns flashing option on and off.

**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;
}
```

#### 12.7.1.14 `PushButton` - HAL Pin Toggle Widget

This widget allows a user to **set a HAL pin true or false** with the push of a button.

As an option it can be a *toggle button*.

For a *LED Indicator Option*, see Section 12.7.5.1[IndicatedPushButton] below for more info.

It also has other options.

It is based on PyQt's *QPushButton*.

#### 12.7.1.15 `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*.

#### 12.7.1.16 `Slider` - HAL Pin Value Adjusting Widget

Allows one to **adjust a HAL pin value using a sliding pointer**.

#### 12.7.1.17 `TabWidget` - Tab Widget

This widget allows the tab height to be adjusted with stylesheets.

The `TabWidget` properties can be defined in a *stylesheet* with the following code added to the `.qss` file.
`name_of_tab` being the widget name defined in Qt Designer's editor.
If you omit the *#name_of_tab* text, all TabWidgets tab height will be set.
This shows how to set a particular widget's tab height:

```
TabWidget #name_of_tab{
  qproperty-tabsize: 1.5;
}
```

Он основан на PyQt's QTabWidget.

#### 12.7.1.18 `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(`**

This shows the `page 0` layout, and puts all other widgets back to where they were as initially built in Qt Designer.

**`[_WidgetSwitcherName_].show_next()`**
Show next widget.

It is based on the *QStack* widget.

#### 12.7.1.19 `XEmbed` - Program Embedding Widget

Allows one to **embed a program into the widget**.

Only programs that utilize the `xembed` protocol will work such as:

- GladeVCP virtual control panels
- Onboard virtual keyboard
- QtVCP virtual control panels
- mplayer video player

### 12.7.2 Machine Controller Widgets

These widgets **interact with the Machine Controller state**.

#### 12.7.2.1 `ActionButton` - Machine Controller Action Control Widget

These buttons are used for **control actions on the machine controller**.

They are built on top of `IndicatedPushButton` so can have LEDs overlaid.

---

**Note**
If you left double click on this widget you can launch a dialog to set any of these actions. The dialogs will help to set the right related data to the selected action. You can also change these properties directly in the property editor.

---

**Actions** You can select one of these:

**`Estop` , `Machine On` , `Auto` , `mdi` , `manual` , `run` , `run_from_line status`**
: Gets line number from `STATUS` message `gcode-line-selected`.

**`run_from_line slot`**
: Gets line number from Qt Designer int/str slot `setRunFromLine`.

**`abort` , `pause` , `load dialog`**
: Requires a dialog widget present.

**`Camview dialog`**
: Requires `camview` dialog widget present.

**`origin offset dialog`**
: Requires origin offset dialog widget present.

**`macro dialog`**
: Requires macro dialog widget present.

**`Launch Halmeter` , `Launch Status` , `Launch Halshow` , `Home`**
: Set the joint number to -1 for `all-home`.

**`Unhome`**
: Set the joint number to -1 for `all-unhome`.

**`Home Selected`**
: Homes the joint/axis selected by `STATUS`.

**Unhome Selected**
Unhomes the joint/axis selected by STATUS.

**zero axis , zero G5X**
Zeros the current user coordinate system offsets.

**zero G92**
Zeros the optional G92 offsets.

**zero Z rotational**
Zeros the rotation offset.

**jog joint positive**
Set the joint number.

**jog joint negative**
Set the joint number.

**jog selected positive**
Selected with a different widget or STATUS.

**jog selected negative**
Selected with a different widget or STATUS.

**jog increment**
Set metric/imperial/angular numbers.

**jog rate**
Set the float/alt float number.

**feed override**
Set the float/alt float number.

**rapid override**
Set the float/alt float number.

**spindle override**
Set the float/alt float number.

**spindle fwd , spindle backward , spindle stop , spindle up , spindle down , view change**
Set view_type_string.

**limits override , flood , mist , block delete , optional stop , mdi command**
Set command_string, 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** These set *attributes* of the selected action (availability depends on the widget):

**toggle float option**
Allows jog rate and overrides to toggle between two rates.

**joint number**
Selects the joint/axis that the button controls.

**incr imperial number**
Sets the imperial jog increment (set negative to ignore).

**incr mm number**
Sets the metric jog increment (set negative to ignore).

**incr angular number**
Sets the angular jog increment (set negative to ignore).

**float number**
Used for `jograte` and overrides.

**float alternate number**
For `jograte` and overrides that can toggle between two float numbers.

**view type string**
Can be:

- `p`,
- `x`, `y`, `y2`, `z`, `z2`,
- `zoom-in`, `zoom-out`,
- `pan-up`, `pan-down`, `pan-left`, `pan-right`,
- `rotate-up`, `rotate-down`, `rotate-cw`, `rotate-ccw`
- `clear`.

**command string**
MDI command string that will be invoked if the MDI command action is selected.

**ini_mdi_number**
(Legacy way)
A reference to the *INI file* `[MDI_COMMAND_LIST]` section.
Set an integer of select one line under the INI's `[MDI_COMMAND]` line starting at 0.
Then in the INI file, under the heading `[MDI_COMMAND_LIST]` add appropriate lines.
Commands separated by the `;` will be run one after another
The button label text can be set with any text after a comma, the `\n` symbol adds a line break.

**ini_mdi_key**
(preferred way)
A reference to the *INI file* `[MDI_COMMAND_LIST]` section.
This string will be added to *MDI_COMMAND_* to form an entry to look for
in the INI file, under the heading `[MDI_COMMAND_LIST]`.
Commands separated by the `;` will be run one after another
The button label text can be set with any text after a comma, the `\n` symbol adds a line break.

```
[MDI_COMMAND_LIST]
MDI_COMMAND_MACRO0 = G0 Z25;X0 Y0;Z0, Goto\nUser\nZero
MDI_COMMAND_MACRO1 = G53 G0 Z0;G53 G0 X0 Y0, Goto\nMachn\nZero
```

Action buttons are subclassed from Section 12.7.5.1[IndicatedPushButton]. See the following sections for more information about:

- LED Indicator option

- Enabled on State
- Text Changes On State
- Call Python Command On State

#### 12.7.2.2 `ActionToolButton` - Optional Actions Menu Button Widget

`ActionToolButton` buttons are similar in concept to action buttons, but they use *QToolButtons* to allow for **optional actions** to be selected by pushing and holding the button till the option menu pops up. Currently there is only one option: `userView`.

It is based on PyQt's *QToolButton*. **userView Record and Set User View Widget**

User View tool button allows to **record and return to an arbitrary graphics view**.

Press and hold the button to have the menu pop up and press *record view* to record the currently displayed graphics view.

Click the button normally to return to the last recorded position.

The recorded position will be remembered at shutdown if a preference file option is set up.

---

**Note**
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 `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:

- Zero the axis
- Divide the axis by 2
- Set the axis arbitrarily
- Reset the axis to the last number recorded

You must have selected an entry dialog widget that corresponds to the dialog_code_string, usually this is selected from the screenOptions widget.

**`halpin_option`**
: Установит контакт HAL в true при выборе оси.

**`joint_number`**
: Should be set to the appropriate joint number.

**`axis_letter`**
: Необходимо установить соответствующую букву оси.

Вот свойства меню, вызываемые щелчком и удерживанием:

**showLast**
Show the *Set to last* action.

**showDivide**
Show the *Divide by 2* action.

**showGotoOrigin**
Show the *Go to G53/G5x origin* action.

**showZeroOrigin**
Show the *Zero Origin* action.

**showSetOrigin**
Show the *Set Origin* action.

**dialog_code_string**
Sets which dialog will pop up with numerical entry, i.e. *ENTRY* or *CALCULATOR* to call a typing only entry dialog or a touch/typing calculator type entry dialog.

Вот пример записи таблицы стилей:

```
AxisToolButton {
    /* Modify all the menu options */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

It is based on PyQt's *QToolButton*.

#### 12.7.2.4 `BasicProbe` - Simple Mill Probing Widget

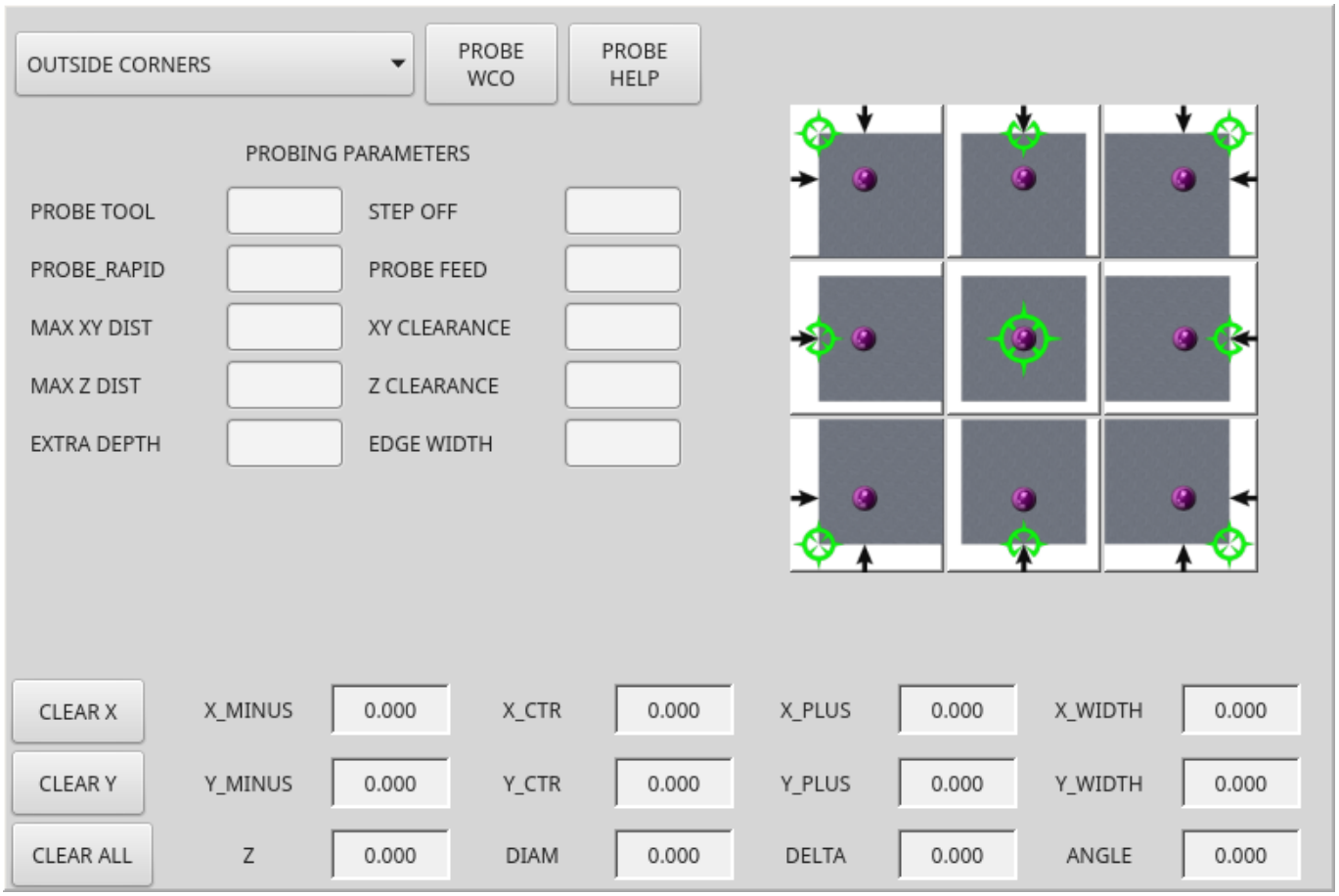



Figure 12.88: QtVCP `BasicProbe`: Simple Mill Probing Widget

Widget for **probing on a mill**. Used by the *QtDragon* screen.

#### 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.

This is used to **align the work piece or zero part features using a webcam**.

It uses *OpenCV* vision library.

#### 12.7.2.6 `DROLabel` - Axis Position Display Widget

This will **display the current position of an axis**.
You can also click on the label and see a list of actions.

**Qjoint_number**
Joint index number (X=0 Y=1) of offset to display (10 will specify rotational offset).

**Qreference_type**
Actual, relative or distance to go (0,1,2).

**metric_template**
Format of display, e.g. `%10.3f`.

**imperial_template**
format of display, e.g. `%9.4f`.

**angular_template**
Format of display, e.g. `%Rotational: 10.1f`.

**always_display_diameter**
Toggles display option

**always_display_radius**
Toggles display option

**display_as_per_m7m8**
Toggles display option. Will follow the current M7/8 mode.

**follow_reference_changes**
Toggles display option. Will follow the STATUS message reference mode, i.e. you can use Action buttons to set how it is currently displayed.

Ниже приведены опции меню, доступные по щелчку:

**showLast**
Show the *Set to last* action.

**showDivide**
Show the *Divide by 2* action.

**showGotoOrigin**
Show the *Go to G53/G5x origin* action.

**showZeroOrigin**
Show the *Zero Origin* action.

**showSetOrigin**
Show the *Set Origin* action.

**dialogName**
Sets which dialog window will pop up with numerical entry, i.e. ENTRY or CALCULATOR.

The `DROLabel` widget holds a property **`isHomed`** that can be used with a stylesheet to change the *color of the `DRO_Label` based on homing state of the joint* number in LinuxCNC.

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.
- показать все пункты меню.

```
DROLabel {
    font: 25pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.4f’;
    qproperty-metric_template: ’%10.3f’;
    qproperty-angular_template: ’%11.2f’;

    /* Modify all the menu options */
    qproperty-showLast: true;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: true;
    qproperty-dialogName: CALCULATOR;
}

DROLabel[isHomed=false] {
    color: red;
}

DROLabel[isHomed=true] {
    color: green;
}
```

Here is how you specify a particular widget by its `objectName` in Qt Designer:

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

It is based on PyQt’s *QLabel*.

#### 12.7.2.7 `FileManager` - File Loading Selector Widget

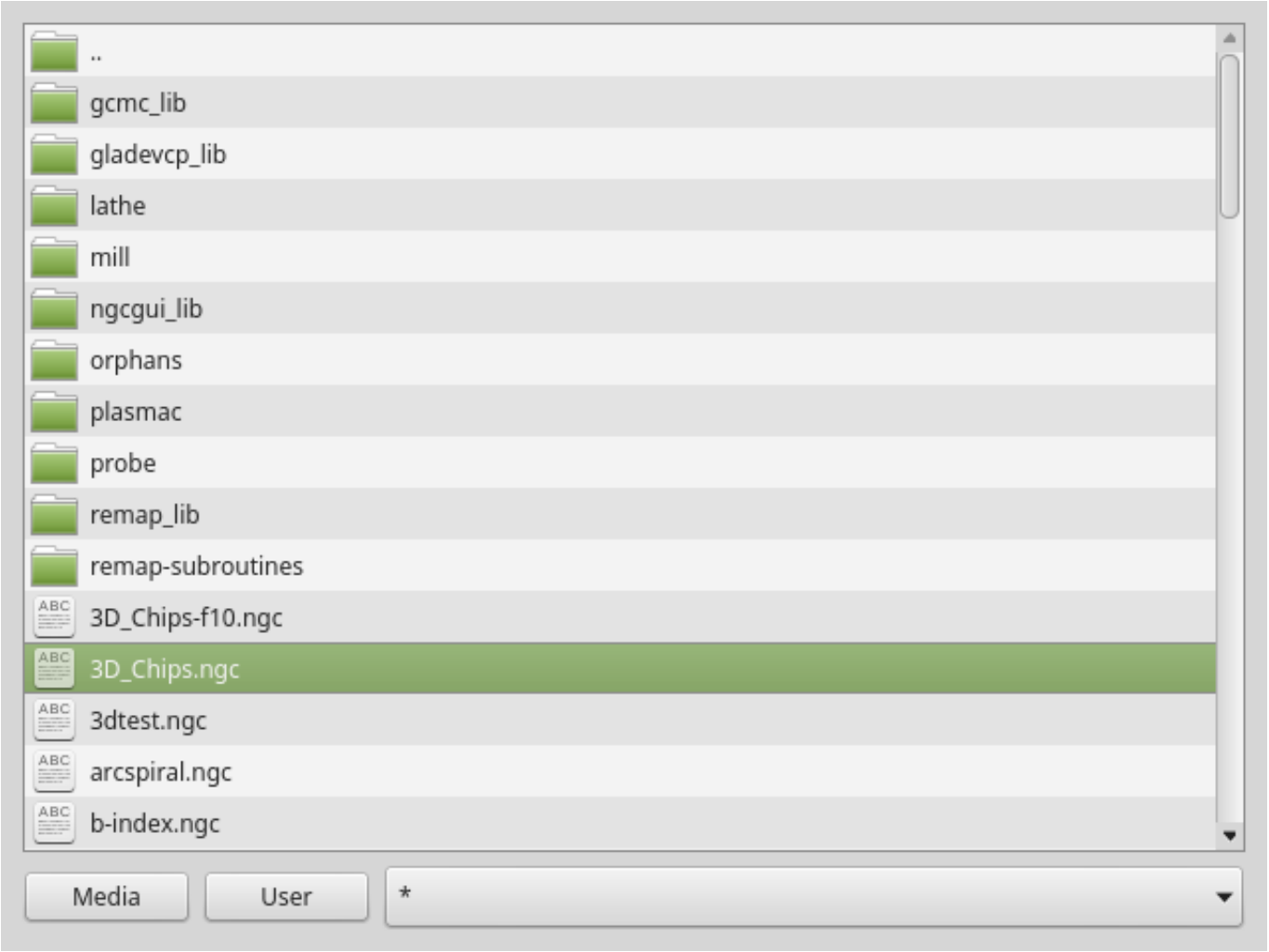



Figure 12.89: QtVCP `FileManager`: File Loading Selector Widget

This widget is used to **select files to load**.

It has a the ability to scroll the names with hardware such as a MPG.

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

#### 12.7.2.8 `GcodeDisplay` - G-code Text Display Widget

This **displays G-code in text form**, highlighting the currently running line.

This can also display:

- **MDI history** when LinuxCNC is in `MDI` mode.
- **Log entries** when LinuxCNC is in `MANUAL` mode.
- **Preference file entries** if you enter `PREFERENCE` in capitals into the `MDILine` widget.

It has a *signal* **`percentDone(int)`** that can be connected to a slot (such as a `progressBar` to display percent run).

**`auto_show_mdi_status`**
Set true to have the widget switch to MDI history when in MDI mode.

**`auto_show_manual_status`**
Set true to have the widget switch to machine log when in Manual mode.

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-styleColorCursor:white;
    qproperty-styleColor0: black;
    qproperty-styleColor1: #000000; /* black */
    qproperty-styleColor2: blue;
    qproperty-styleColor3: red;
    qproperty-styleColor4: green;
    qproperty-styleColor5: darkgreen;
    qproperty-styleColor6: darkred;
    qproperty-styleColor7: deeppink;
    qproperty-styleColorMarginText: White;
    qproperty-styleColorMarginBackground: blue;
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont1: ”Times,18,-1,0,90,1,0,0,0,0”;
    qproperty-styleFont2: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont3: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont4: ”Times,12,-1,0,90,0,0,0,0,0”;
```

```
    qproperty-styleFont5: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont6: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFont7: ”Times,12,-1,0,90,0,0,0,0,0”;
    qproperty-styleFontMargin: ”Times,14,-1,0,90,0,0,0,0,0”;
}
```

For `GcodeDisplay` widget’s *default G-code lexer*:

- **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$

---

**Note**
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.

---

*Font definitions*:

```
”style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0”
```

It is based on PyQt’s *QsciScintilla*.

#### 12.7.2.9 `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.

#### 12.7.2.10 `GCodeGraphics` - G-code Graphic Backplot Widget

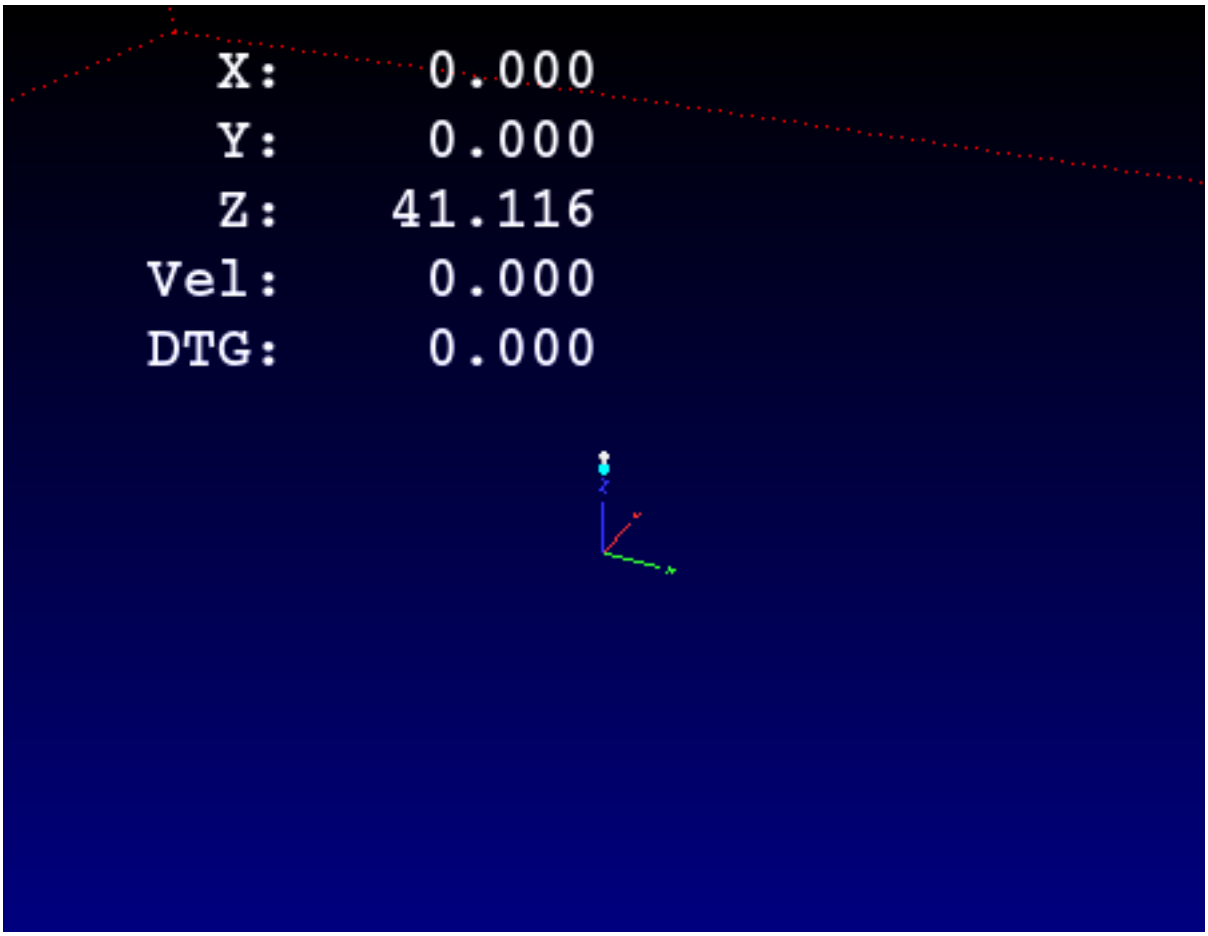



Figure 12.90: QtVCP GcodeGraphics: G-code Graphic Backplot Widget

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)***
Determines whether or not to *show the DRO*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

**_dtg *(bool)***
Determine whether or not to *show the Distance To Go*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

**_metric *(bool)***
Determines whether or not to *show the units in metric by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

**_overlay *(bool)***
Determines whether or not to *show the overlay by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

**_offsets *(bool)***
Determines whether or not to *show the offsets by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

**_small_origin *(bool)***
Determines whether or not to *show the small origin by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

**overlay_color *(primary, secondary, or RGBA formatted color)***
Sets the *default overlay color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

**overlay_alpha** ***(float)***
Sets the *default overlay alpha value*. This affects the opacity of the overlay when set between 0.0 and 1.0.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-overlay_alpha: 0.15;
}
```

**background_color** ***(primary, secondary, or RGBA formatted color)***
Sets the *default background color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

**+_use_gradient_background+** ***(bool)***
Determines whether or not *use a gradient background by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
}
```

**jog_color** ***(primary, secondary, or RGBA formatted color)***
Sets the *default jog color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

**Feed_color** ***(primary, secondary, or RGBA formatted color)***
Sets the *default feed color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

**Rapid_color** ***(primary, secondary, or RGBA formatted color)***
Sets the *default rapid color*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

**InhibitControls** ***(bool)***
Determines whether or not to *inhibit external controls by default*.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

**MouseButtonMode** ***(int)***
Changes the *mouse button behavior* to rotate, move or zoom within the preview.
The following shows an example of how to set this property:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

There are 12 valid modes:

| Mode | Move | Zoom | Rotate |
|---|---|---|---|
| 0 | Left | Middle | Right |
| 1 | Middle | Right | Left |
| 2 | Middle | Left | Right |
| 3 | Left | Right | Middle |
| 4 | Right | Left | Middle |
| 5 | Right | Middle | Left |

Modes 6-11 are intended for machines that only require a 2D preview such as plasma or some lathes and have no rotate button assigned.

| Mode | Move | Zoom |
|---|---|---|
| 6 | Left | Middle |
| 7 | Middle | Left |
| 8 | Right | Left |
| 9 | Left | Right |
| 10 | Middle | Right |
| 11 | Right | Middle |

**`MouseWheelInvertZoom` *(bool)***
Determines 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_)`**
Directly set the relative pan of view in x and y direction.

**`ACTION.ADJUST_ROTATE(_X,Y_)`**
Directly set the relative rotation of view in x and y direction.

Он основан на PyQt *OpenGL* виджете.

#### 12.7.2.11 `JointEnableWidget` - FIXME

FIXME JointEnableWidget documentation

#### 12.7.2.12 `JogIncrements` - Jog Increments Value Selection Widget

This widget allows the user to **select jog increment values for jogging**.

The jogging values come from the *INI file* under:

- `[DISPLAY]INCREMENTS`, or
- `[DISPLAY]ANGULAR_INCREMENTS`

This will be *available to all widgets* through `STATUS`.

You can select linear or angular increments by the property **`linear_option`** in Qt Designer property editor.

It is based on PyQt's *ComboBox*.

#### 12.7.2.13 `MacroTab` - Special Macros Widget

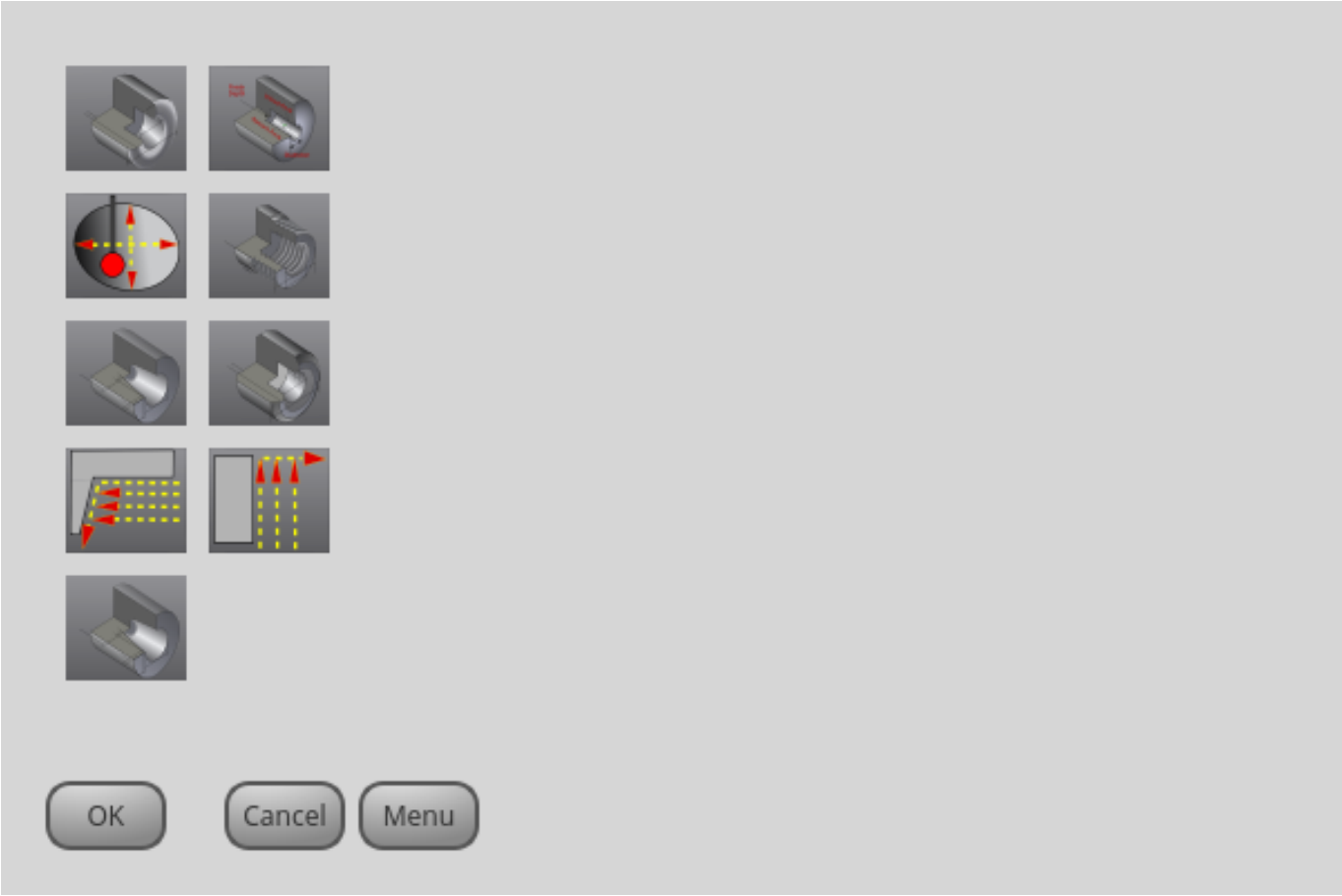



Figure 12.91: QtVCP `MacroTab`: Special Macros Widget

This widget allows a user to **select and adjust special macro programs** for doing small jobs.

It uses *images for visual representation* of the macro and for an icon.

It searches for special macros using the *INI definition*:

```
[RS274NGC]
SUBROUTINE_PATH =
```

The macros are **`O-word` subroutines with special comments** to work with the launcher. The first three lines *must* have the keywords below, the fourth is optional.

Here is a sample for the first four lines in an *O-word file*:

```
; 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** This is the *first line* in the O-word file.

It is a **comma separated list of text to display above an entry**.
There will be **one for every variable required** in the O-word function.

If the macro does not require variables, leave it empty:

```
; MACROCOMMAND=
```

**MACRODEFAULTS** This must be the *second line* in the O-word file.

It is a **comma separated list of the default values for each variable** in the O-word function.

If you use the word `radiotrue`, `radiofalse`, `true` or `false` in the list, a *`radiobutton`* will be shown.
If you use the word `checktrue` or `checkfalse` in the list, a *`checkbox`* will be shown.
If you use the word `buttontrue` or `buttonfalse` in the list, a *`Checkable Pushbutton`* will be shown.
If the default has a decimal, macroTab assumes you want a float value otherwise an integer.

---

**Note**
When using radiobuttons, only set one radiobutton as true. Radio button are used for exclusive choices.

---

**MACROIMAGE** This must be the *third line* in the O-word file.

- **SVG Images**
  If using SVG image files, they must end with the `.svg` extension.

  The images must be added to *SVG layers* which are used to define the different images for macro and icon.

  Value is comma separated list of three ordered fields:

  ```
  ; MACROIMAGE=filename.svg,macro_layer_name[,icon_layer_name]
  ```

  With:

  **`_filename_.svg`**
  : SVG image file name as first field.
    It is assumed to be in the same folder as the O-word file.

  **`*macro_layer_name`**
  : Macro image layer name as second field.

  **`icon_layer_name`**
  : Icon image layer name as optional third field. If the third entry is missing, the same image will be used for macro and icon.

- **PNG/JPG Images**:
  Value remains a comma separated list:

  ```
  ; MACROIMAGE=macro_image.(png|jpg)[,icon_image.(png|jpg)]
  ```

  With:

  **`_macro_image_.(png|jpg)`**
  : Macro image file name as first field.
    It is assumed that the image file are in the same folder than the macro.

  **`_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.

If the keyword is present but the entries are missing, no images will be used.

**MACROOPTIONS** This *optional line must be the fourth* line in the O-word file.

It is a comma separated list of keyword and data, all of which are optional:

**LOAD:yes**
Shows a load button.

**SAVE:yes**
Shows a save button.

**DEFAULT:ThisMacroData.txt**
Sets the default preselected filename when loading/saving data for this macro.
It can be any valid filename but must end in *.txt*.

**PATH:~/linuxcnc/nc_files/mySavedMacrosData**
Sets the default directory folder to preselect when loading/saving data for this macro.

Here are stylesheet hints for adjusting the MacroTab widget.

```
MacroTab CustomButton{
    width: 20px;
    height: 40px;
}

MacroTab QPushButton {
    width: 80px;
    height: 40px;
}

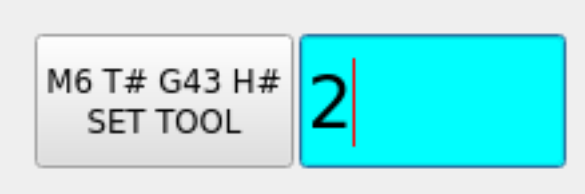
MacroTab QLabel {
    font: 24pt ”Lato Heavy”;
}

TouchSpinBox  LineEdit {
    font: 12pt ”Lato Heavy”;
}

TouchSpinBox  QPushButton {
    width: 60px;
    height: 100px;
}
```

#### 12.7.2.14 `OperatorValueLine` - Operator Value Line Entry Widget

The operator enters values into this widget, which will be applied to a template and then optionally issued to the MDI either immediately or applied at a later time. The widget supports the optional popup calculator, keyboard, or tool chooser for touchscreen-friendly entry by setting the `dialog_keyboard_option`. To change which type of dialog is presented, edit the `dialog_code_option`.




The widget supports a formatting option which is passed to Python’s string `format()` to produce the final output for the MDI command. The special token `{value}` can be inserted anywhere in this format string where the value should appear. The formatting property is called `mdi_command_format_option`, e.g.:

- `M3 S{value}` to start the spindle at the speed entered by the operator.

- `M6 T{value} G43 H{value}` to issue a tool change and tool length offset change from the tool number entered

The widget may be configured to automatically issue the MDI command upon submit when `issue_mdi_on_su` is set to `True`. If `False` issuing the command may be done at a later time via a signal or function call from another widget.

In cases where `issue_mdi_on_submit_option` is `False`, calling the `issue_mdi()` function will issue the command. Slots attached to widgets such as PushButtons can trigger the MDI command when pressed, e.g.:

```
    def setSpindleSpeed(self, event):
        self.w.lineSpindleSpeed.issue_mdi()
        ACTION.SET_MANUAL_MODE()

    def setToolNumber(self, event):
        self.w.lineToolNumber.issue_mdi()
        ACTION.SET_MANUAL_MODE()
```

The widget tracks whether a value entered is pending and has not yet been issued via the property `isPendingValue`. This may be used to style the widget via the stylesheet. This can be used to alert the operator that they entered a value but another action must be taken to apply it.

The following style sheet excerpt will highlight the entry widget with a cyan background when values are pending and have not been applied.

```
#lineSpindleSpeed[isPendingValue=true],
#lineToolNumber[isPendingValue=true] {
    background: cyan;
}

#lineSpindleSpeed[isPendingValue=false],
#lineToolNumber[isPendingValue=false] {
    background: none;
}
```

#### 12.7.2.15 `MDILine` - MDI Commands Line Entry Widget

One can **enter MDI commands** here.

A popup keyboard is available.

**Embedded Commands** There are also **embedded commands** available from this widget.

Enter any of these *case sensitive* commands to load the respective program or access the feature:

**`HALMETER`**
: Starts LinuxCNC `halmeter` utility.

**`HALSHOW`**
: Starts LinuxCNC `halshow` utility.

**`HALSCOPE`**
: Starts LinuxCNC `halscope` utility.

**`STATUS`**
: Starts LinuxCNC `status` utility.

**`CALIBRATION`**
: Starts LinuxCNC Calibration

**CLASSICLADDER**
Starts the ClassicLadder GUI if the *ClassicLadder realtime HAL component* was loaded by the machine's config files.

**PREFERENCE**
*Loads the preference file* into the `GcodeEditor`.

**CLEAR HISTORY**
*Clears the MDI History*.

**net**
See `halcmd net` commands.
An error will result if the command is unsuccessful.

- *Syntax*: `net <signal name> <pin name>`
- *Example*: `net plasmac:jog-inhibit motion.jog-stop`

**setp**
*Sets the value of a pin or a parameter*.
Valid values depend on the object type of the pin or parameter.
It results in an error if the data types do not match or the pin is connected to a signal.

- *Syntax*: `setp <pin/parameter-name> <value>`
- *Example*: `setp plasmac.resolution 100`

**unlinkp**
*Disconnects a pin from a signal*.
An error will result if the pin does not exist.
Running LinuxCNC from terminal may help determine the root cause as error messages from `hal_lib.c` will be displayed there.

- *Syntax*: `unlinkp <pin name>`
- *Example*: `unlinkp motion.jog-stop`

---

**Note**
The `MDILine` function **`spindle_inhibit`** can be used by a GUI's handler file to inhibit `M3`, `M4`, and `M5` spindle commands if necessary.

---

It is based on PyQt's *QLineEdit*.

#### 12.7.2.16 `MDIHistory` - MDI Commands History Widget

Displays a **scrollable list of past MDI command**.

An edit line is embedded for MDI commands. The same MDILine embedded commands may be accessed from this widget.

The history is *recorded on a file defined in the INI* under the heading `[DISPLAY]` (this shows the default):

```
MDI_HISTORY_FILE = '~/.axis_mdi_history'
```

#### 12.7.2.17 `MDITouchy` - Touch Screen MDI Entry Widget

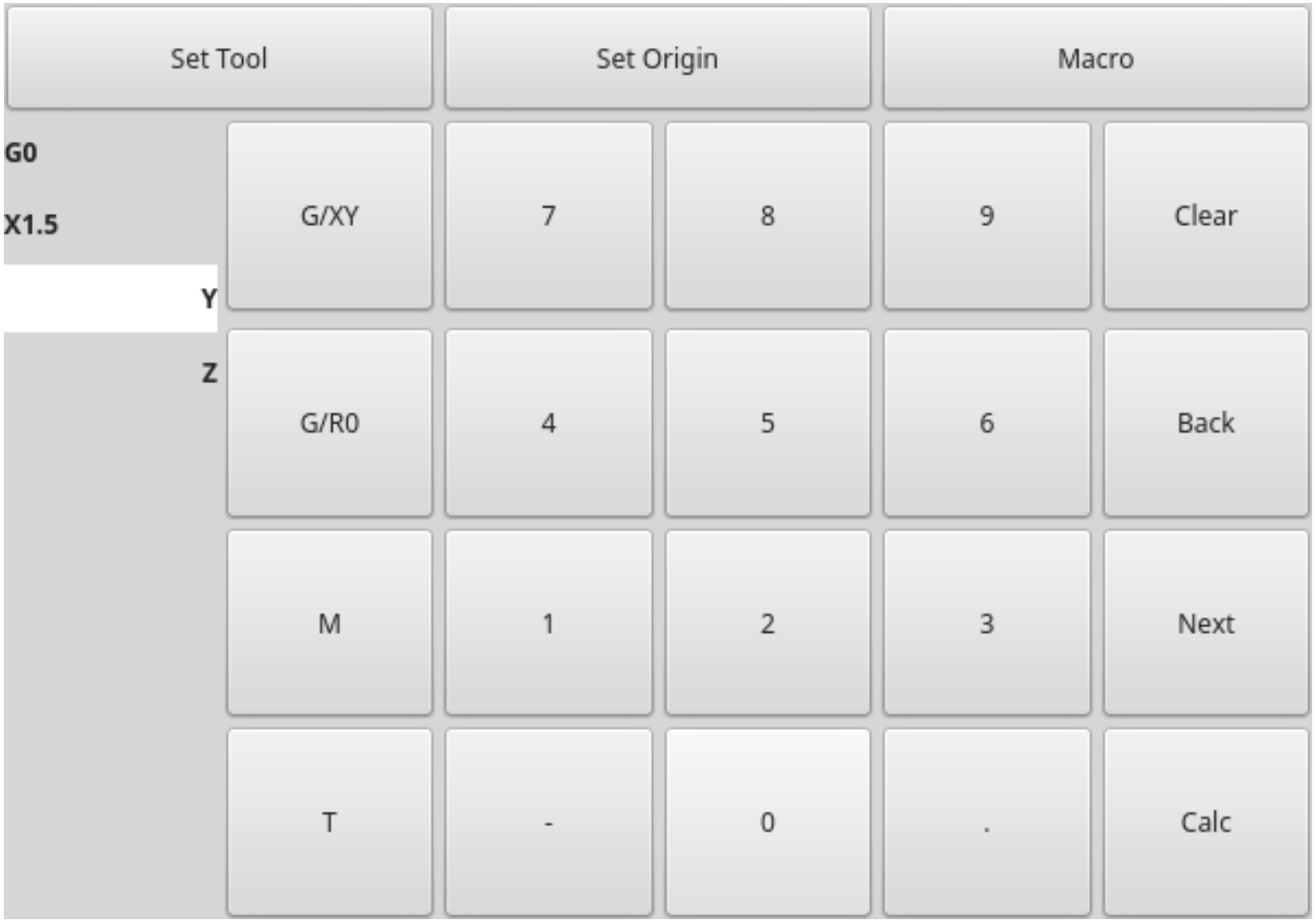



Figure 12.92: QtVCP `MDITouchy`: Touch Screen MDI Entry Widget

This widget displays **buttons and entry lines to use for entering MDI commands**.

Based on LinuxCNC's Touchy screen's MDI entry process, its large buttons are most useful for touch screens.

To use `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.
- Then press `Next` and `Back` to navigate between fields.
- `Calc` will pop up a calculator dialog.
- `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()
```

The macro button *cycles though macros defined in the INI [DISPLAY] heading*.

Add one or more `MACRO` lines of the following format:

```
MACRO = macro_name [param1] [... paramN]
```

In the example below, `increment` is the name of the macro, and it accepts two parameters, named `xinc` and `yinc`.

```
MACRO = incerment 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.

Keeping on with the example above, it would be named `increment.ngc` and its content could look like:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Notice the *name of the sub matches the file name and macro name exactly*, including case.

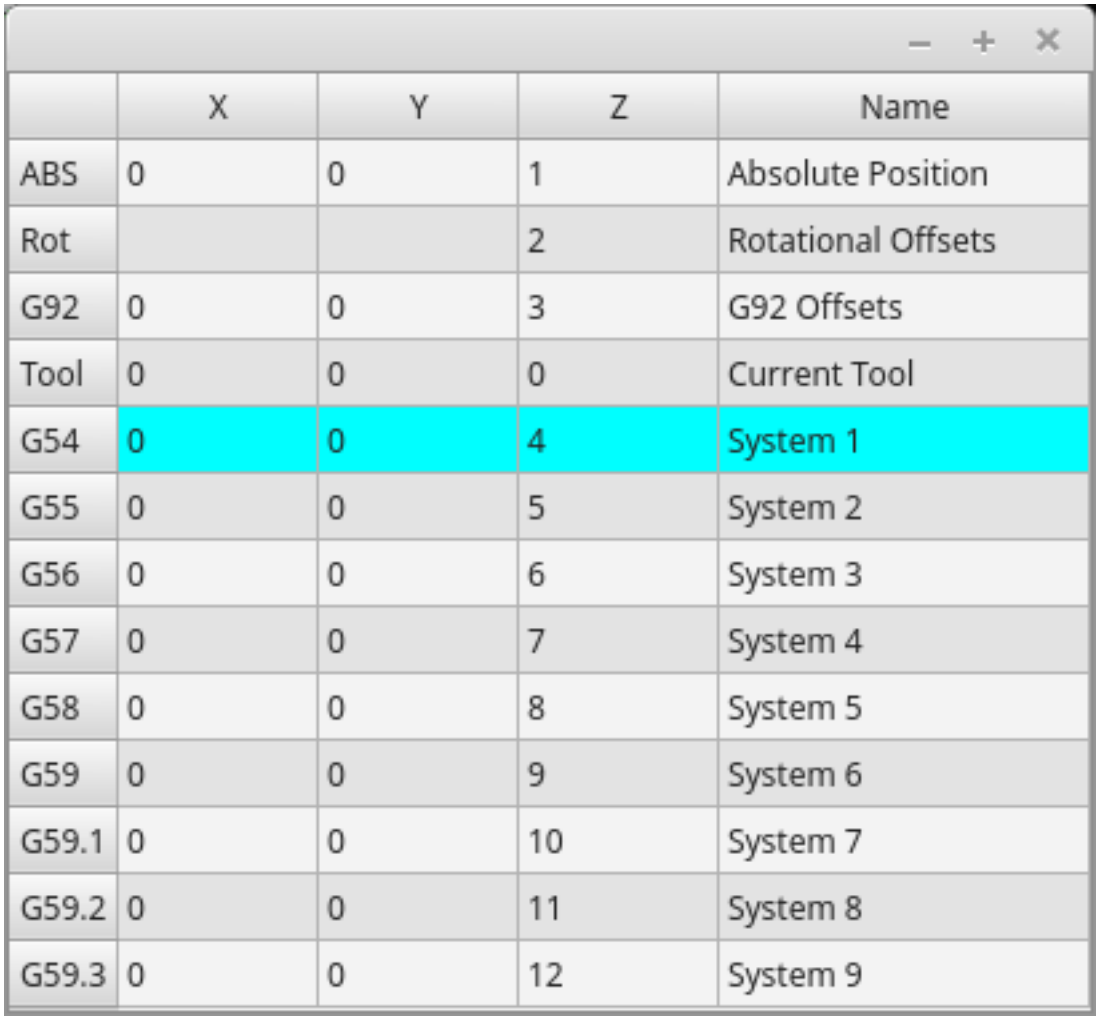
When you invoke the macro by pressing the Macro button you can enter values for parameters (`xinc` and `yinc` in our example).
These are passed to the macro as positional parameters: `#1`, `#2`... `#N` respectively.
Parameters you leave empty are passed as value `0`.

If there are several different macros, press the Macro button repeatedly to cycle through them.

In this simple example, if you enter -1 for xinc and invoke the running of the MDI cycle, a rapid *G0* move will be invoked, moving one unit to the left.

This macro capability is useful for edge/hole probing and other setup tasks, as well as perhaps hole milling or other simple operations that can be done from the panel without requiring specially-written G-code programs.

#### 12.7.2.18 `OriginOffsetView` - Origins View and Setting Widget



| | X | Y | Z | Name |
|---|---|---|---|---|
| ABS | 0 | 0 | 1 | Absolute Position |
| Rot | | | 2 | Rotational Offsets |
| G92 | 0 | 0 | 3 | G92 Offsets |
| Tool | 0 | 0 | 0 | Current Tool |
| G54 | 0 | 0 | 4 | System 1 |
| G55 | 0 | 0 | 5 | System 2 |
| G56 | 0 | 0 | 6 | System 3 |
| G57 | 0 | 0 | 7 | System 4 |
| G58 | 0 | 0 | 8 | System 5 |
| G59 | 0 | 0 | 9 | System 6 |
| G59.1 | 0 | 0 | 10 | System 7 |
| G59.2 | 0 | 0 | 11 | System 8 |
| G59.3 | 0 | 0 | 12 | System 9 |

Figure 12.93: QtVCP `OriginOffsetsView`: Origins View and Setting Widget

This widget allows one to **visualize and modify User System Origin offsets** directly.

It will *update LinuxCNC's Parameter file* for changes made or found.

The 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 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.

It is based on PyQt's *QTableView*, *QAbstractTableModel*, and *ItemEditorFactory*.
Properties, functions and styles of the PyQt base objects are always available.

**Свойства** `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.

These can be set 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;
}
```

#### 12.7.2.19 `RadioAxisSelector` - FIXME

FIXME RadioAxisSelector documentation

#### 12.7.2.20 `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**.

#### 12.7.2.21 `StateLabel` - Controller Modes State Label Display Widget

This will **display a label based on the machine controller modes true/false states**.

You can select between different texts based on true or false.

**States Selection Properties** Состояния выбираются с помощью следующих свойств:

**`css_mode_status`**
`True` when machine is in `G96` *Constant Surface Speed Mode*.

**diameter_mode_status**
`True` when machine is in `G7` *Lathe Diameter Mode*.

**fpr_mode_status**
`True` when machine is in `G95` *Feed per revolution Mode*.

**metric_mode_status**
`True` when machine is in `G21` *Metric Mode*.

Text templates properties

**true_textTemplate**
This will be the text set when the option is `True`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in true state, might be: *Metric Mode*

**false_textTemplate**
This will be the text set when the option is `False`.
You can use *Qt rich text* code for different fonts/colors etc.
Typical template for metric mode in false state, might be: *Imperial Mode*.

It is based on PyQt's *QLabel*.

#### 12.7.2.22 `StatusLabel` - Controller Variables State Label Display Widget

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**
Used to display the actual spindle speed as *reported from the HAL pin* `spindle.0.speed-i`.
It's converted to *RPM*.
A `textTemplate` of `%d` would typically be used.

**actual_surface_speed_status**
Used to display the actual cutting surface speed on a lathe based on X axis and spindle speed.
It's converted to distance per minute.
A `textTemplate` of `%4.1f` (feet per minute) and `altTextTemplate` of `%d` (meters per minute) would typically be used.

**blendcode_status**
Shows the current `G64` setting.

**current_feedrate_status**
Shows the current actual feedrate.

**current_FPU_status**
Shows the current actual feed per unit.

**fcode_status**
Shows the current programmed `F` code setting.

**feed_override_status**
Shows the current feed override setting in percent.

**filename_status**
Показывает имя последнего загруженного файла.

**filepath_status**
Показывает путь к последнему загруженному файлу.

**gcode_status**
Показывает все активные G-коды.

**gcode_selected_status**
Show the current selected G-code line.

**halpin_status**
Shows the HAL pin output of a selected HAL pin.

**jograte_status**
Shows the current QtVCP based Jog Rate.

**jograte_angular_status**
Shows the current QtVCP based Angular Jog Rate.

**jogincr_status**
Shows the current QtVCP based Jog increment.

**jogincr_angular_status**
Shows the current QtVCP based Angular Jog increment.

**machine_state_status**
Shows the current *machine interpreter state* using the text described from the 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**
Показывает запрошенную скорость шпинделя - фактическая может отличаться.

**rapid_override_status**
Shows the current rapid override setting in (0-100) percent.

**spindle_override_status**
Shows the current spindle override setting in percent.

**timestamp_status**
Shows the time based on the system settings.
An example of a useful `textTemplate` setting: `%I:%M:%S %p`.
See the Python time module for more info.

**tool comment_status**
Returns the comment text from the current loaded tool.

**tool diameter_status**
Returns the diameter from the current loaded tool.

**tool_number_status**
Returns the tool number of the current loaded tool.

**tool_offset_status**
Returns the offset of the current loaded tool, indexed by `index_number` to select axis (0=x,1=y,etc.).

**user_system_status**
Shows the *active user coordinate system* (`G5x` setting).

Прочие Настройки

**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 base-name).

**textTemplate**
This is usually used for **imperial (G20) or angular numerical settings**, though not every option has imperial/metric conversion.
This uses *Python formatting rules* to set the text output.
One can use `%s` for no conversion, `%d` for integer conversion, `%f` for float conversion, etc.
You can also use *Qt rich text* code.
Typical template used for formatting imperial float numbers to text would be `%9.4f` or `%9.4f inch`.

**alt_textTemplate**
This is usually used for **metric (G21) numerical settings**.
This uses *Python formatting rules* to set the text output.
Typical template used for formatting metric float to text would be `%10.3f` or `%10.3f mm`.

It is based on PyQt's *QLabel*.

#### 12.7.2.23 `StatusImageSwitcher` - Controller Status Image Switcher

Status image switcher will **switch between images based on LinuxCNC states**.

***watch_spindle**
Toggles between *3 images*: `stop`, `fwd`, `revs`.

***watch_axis_homed**
Toggles between *2 images*: `axis not homed`, `axis homed`.

***watch_all_homed**
Would toggle between *2 images*: `not all homed`, `all homed`.

***watch_hard_limits**
Would toggle between *2 images or one per joint*.

Here is an example of using it to display an icon of Z axis homing state:

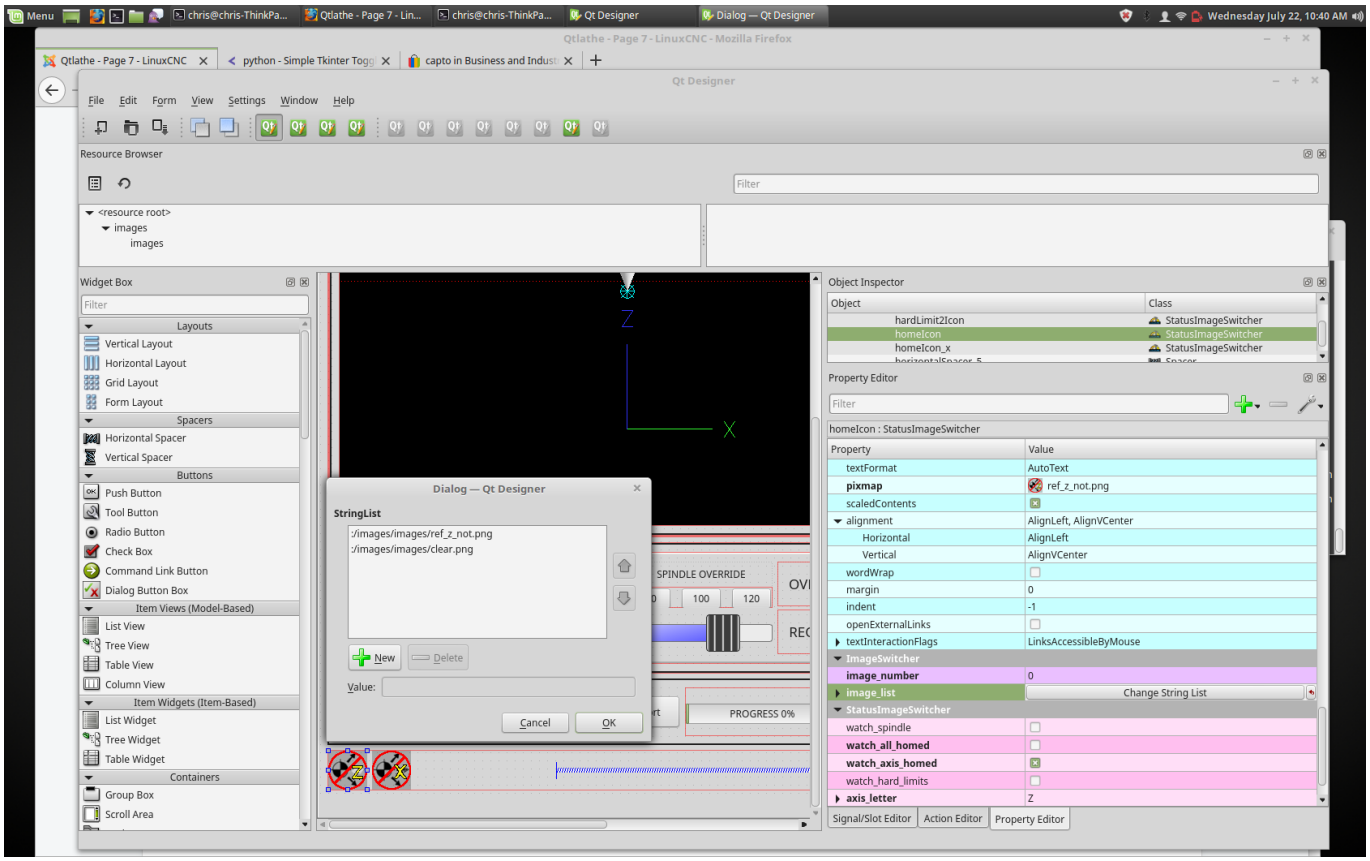



Figure 12.94: QtVCP `StatusImageSwitcher`: Controller Status Image Switcher

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.

**Note**
The `pixmap` setting is for test display only and will be ignored outside of Qt Designer.

- Right click the image name and you should see *Copy path*.
- Click *Copy path*.
- Now double click the *image list* property so the dialog shows.
- Click the *New* button.
- Paste the image path in the entry box.

Do that again for the next image.
*Use a clear image to represent a hidden icon.*

You can *test the images display* from the image list by changing the `image number`. In this case `0` is unhomed and `1` would be homed.
This is for test display only and will be ignored outside of Qt Designer.

#### 12.7.2.24 `StatusStacked` - Mode Status Display Switching Widget

This widget **displays one of three panels based on LinuxCNC's mode**.

This allows you to automatically display different widgets on *Manual*, *MDI* and *Auto* modes. **TODO** It is based on PyQt's *QStacked* widget.

#### 12.7.2.25 `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*.

**Свойства** 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`**
: Color of transparent layer shown when quitting.

**`catch_error_option`**
: *Monitoring of the LinuxCNC error channel*.
This also sends the message through `STATUS` to anything that registers.

**play_sounds_option**
Playing sounds using `beep`, `espeak` and the system sound.

**use_pref_file_option**
Setting up a *preferences file path*.
Using the magic word `WORKINGFOLDER` in the preference file path will be replaced with the launched configuration path, e.g. `WORKINFOLDER/my_preferences`.

**use_send_zmq_option**
Used to initiate *ZMQ based outgoing messages*.

**use_receive_zmq_messages**
Used to initiate *ZMQ based in coming messages*.
These messages *can be used to call functions in the handler file*, allowing **external programs to integrate tightly with QtVCP** based screens.

**embedded_program_option**
Embed programs defined in the *INI*.

**default_embed_tab**
This is the property for a *default location to embed external programs*.
It should be set to name of a tab page widget in Qt Designer.

**focusOverlay_option**
Focus_overlay will put a transparent image or colored panel over the main screen to emphasize focus to an external event - typically a dialog.

**messageDialog_option**
Sets up the message dialog - used for general messages.

**message_overlay_color**
Color of transparent layer shown when the message dialog is shown.

**closeDialog_option**
Sets up the standard close screen prompt dialog.

**entryDialog_option**
Sets up the numerical entry dialog.

**entryDialogSoftKey_option**
Sets up a floating software keyboard when entry dialog is focused.

**entry_overlay_color**
Color of transparent layer shown when the entry dialog is shown.

**toolDialog_option**
Sets up the manual tool change dialog, including HAL pin.

**tool_overlay_color**
Color of transparent layer shown when the tool dialog is shown.

**ToolUseDesktopNotify**
Option to use desktop notify dialogs for manual tool change dialog.

**ToolFrameless**
Frameless dialogs can not be easily moved by users.

**fileDialog_option**
Sets up the file choosing dialog.

**file_overlay_color**
Color of transparent layer shown when the file dialog is shown.

**keyboardDialog_option**
Sets up a keyboard entry widget.

**keyboard_overlay_color**
Color of transparent layer shown when the keyboard dialog is shown.

**vesaProbe_option**
Sets up the Versa style probe dialog.

**versaProbe_overlay_color**
Color of transparent layer shown when the `versaProbe` dialog is shown.

**macroTabDialog_option**
Sets up the macro selection dialog.

**macroTab_overlay_color**
Color of transparent layer shown when the `macroTab` dialog is shown.

**camViewDialog_option**
Sets up the camera alignment dialog.

**camView_overlay_color**
Color of transparent layer shown when the `camView` dialog is shown.

**toolOffset_option**
Sets up the tool offset display/editor dialog.

**toolOffset_overlay_color**
Color of transparent layer shown when the `toolOffset` dialog is shown.

**originOffset_option**
Sets up the origin display/editor dialog.

**originOffset_overlay_color**
Color of transparent layer shown when the `originOffset` dialog is shown.

**calculatorDialog_option**
Sets up the calculator entry dialog.

**calculator_overlay_color**
Color of transparent layer shown when the calculator dialog is shown.

**machineLogDialog_option**
Sets up a dialog to display logs from the machine and QtVCP.

**machineLog_overlay_color**
Color of transparent layer shown when the `machineLog` dialog is shown.

**runFromLineDialog_option**
Sets up a dialog to display starting options when starting machine execution from a arbitrary line.

**runFromLine_overlay_color**
Color of transparent layer shown when the `runFromLine` dialog is shown.

**user1Color**
Optional color the screen designer can use in their design.

**user2Color**
Optional color the screen designer can use in their design.

**user3Color**
Optional color the screen designer can use in their design.

**user4Color**
Optional color the screen designer can use in their design.

**user5Color**
Optional color the screen designer can use in their design.

**user6Color**
Optional color the screen designer can use in their design.

**user7Color**
Optional color the screen designer can use in their design.

**user8Color**
Optional color the screen designer can use in their design.

**user9Color**
Optional color the screen designer can use in their design.

**user10Color**
Optional color the screen designer can use in their design.

**Setting Properties Programmatically** The screen designer chooses the **default settings of the `screenOptions` widget**.

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 the handler file**:
  Here we reference the widget by the Qt Designer user defined name:

```
# 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)
}
```

**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*.

**Preference File Entries** If the *preference file option is selected*, `screenOption` widget will make an **INI based preference file**.

While *other QtVCP widgets will add to this list*, the `screenOptions` widget will add these entries under the following headings:

**[SCREEN_OPTIONS]**

**catch_errors *(bool)* , desktop_notify *(bool)***
Whether to display errors/messages in the system's notification mechanism.

**notify_max_msgs *(int)***
Number of displayed errors at one time.

**shutdown_check** ***(bool)***
Whether to pop a confirmation dialog.

**sound_player_on** ***(bool)***
Turns all sounds on or off.

**[MCH_MSG_OPTIONS]**

**mchnMsg_play_sound** ***(bool)***
To play alert sound when dialog pops.

**mchnMsg_speak_errors** ***(bool)***
To use Espeak to speak error messages.

**mchnMsg_speak_text** ***(bool)***
To use Espeak to speak all other messages.

**mchnMsg_sound_type** ***(str)***
Sound to play when messages displayed. See notes below.

**[USER_MSG_OPTIONS]**

**usermsg_play_sound** ***(bool)***
To play alert sound when dialog pops.

**userMsg_sound_type** ***(str)***
Sound to play when user messages displayed. See notes below.

**userMsg_use_focusOverlay** ***(bool)***

**[SHUTDOWN_OPTIONS]**

**shutdown_play_sound** ***(bool)*** **, shutdown_alert_sound_type** ***(str)***
Sound to play when messages displayed. See notes below.

**shutdown_exit_sound_type** ***(str)***
Sound to play when messages displayed. See notes below.

**shutdown_msg_title** ***(str)***
Short title string to display in dialog.

**shutdown_msg_focus_text** ***(str)***
Large text string to superimpose in focus layer.

**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)***
Short Title string.
If the speak option is also selected it will be spoken with Espeak.

**notify_start_detail** ***(str)***
Longer description string.

**notify_start_timeout** ***(int)***
Time in seconds to display before closing.

*_sound_type entries

- **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

These Sound options require `python3-gst1.0` installed.

- **Audio Files**
  You can also specify a *file path to an arbitrary audio file*.
  You can use ~ in path to substitute for the user home file path.
- **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_'`**

#### 12.7.2.26 `StatusSlider` - Controller Setting Adjustment Slider Widget

This widget allow the user to **adjust a LinuxCNC setting via a slider**.

The widget can adjust:

- Jog rate
- Angular jog rate
- Feed rate
- Spindle override rate
- Rapid override rate

**Свойства** `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**
Selects a linear jograte slider.

**jograte_angular_rate**
Selects a angular jograte slider.

**max_velocity_rate**
Selects a maximum velocity rate slider.

**alertState**
String to define style change: `read-only`, `under`, `over` and `normal`.

**alertUnder**
Sets the float value that signals the stylesheet for *under* warning.

**alertOver**
Sets the float value that signals the stylesheet for *over* warning.

These can be set 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)
```

- Or (if appropriate) in stylesheets.

```
/* warning colors for overrides if out of normal range*/
/* widget object name is slider_spindle_ovr */

#slider_spindle_ovr[alertState='over'] {
    background: red;
}
#slider_spindle_ovr[alertState='under'] {
    background: yellow;
}
```

It is based on PyQt's *QSlider*.

#### 12.7.2.27 `StateLED` - Controller State LED Widget

This widget gives **status on the selected LinuxCNC state**.

**States** The state options are:

**is_paused_status , is_estopped_status , is_on_status , is_idle_status_ , is_homed_status , is_fl**

**Свойства** There are properties that can be changed:

**halpin_option**
Adds an output pin that reflects selected state.

**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 Alignment hint.

**state**
Current state of LED (for testing in Qt Designer).

**flashing**
Turns flashing option on and off.

**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` would be the name defined in Qt Designer's editor.

It is based on the *LED* widget.

#### 12.7.2.28 `StatusAdjustmentBar` - Controller Value Setting Widget

This widget allows **setting values using buttons while displaying a bar**.

It also has an *optional hi/low toggle button* that can be held down to set the **levels**.

The widget can adjust:

- Jog rate
- Angular jog rate
- Feed rate
- Spindle override rate
- Rapid override rate

It is based on PyQt's *QProgressBar*.

#### 12.7.2.29 `SystemToolButton` - User System Selection Widget

This widget allows you to **manually select a G5x user system by pressing and holding**.
If you don’t set the button text it will automatically update to the current system.
It is based on PyQt’s *QToolButton*.

#### 12.7.2.30 `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*.

#### 12.7.2.31 `StatusImageSwitcher` - Controller Status Image Switching Widget

This widget will **display images based on LinuxCNC status**.
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

#### 12.7.2.32 `ToolOffsetView` - Tools Offsets View And Edit Widget

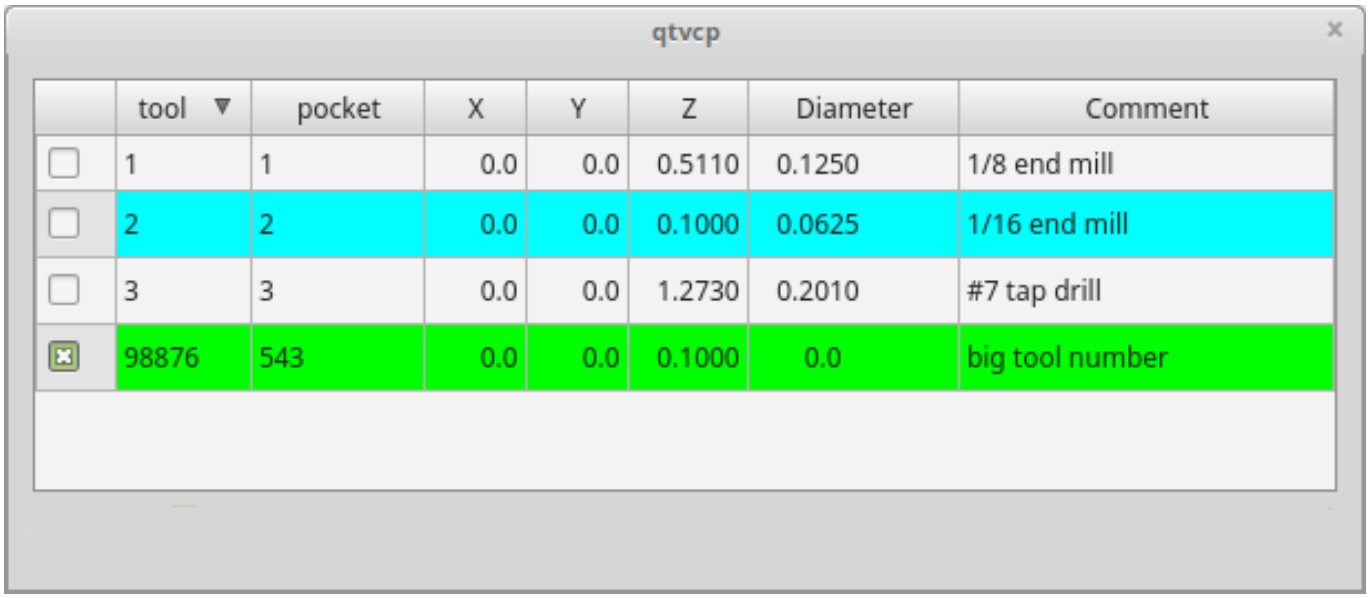



Figure 12.95: QtVCP `ToolOffsetView`: Tools Offsets View And Edit Widget

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.

It is based on PyQt's *QTableView*, *QAbstractTableModel*, and *ItemEditorFactory*.
Properties, functions and styles of the PyQt base objects are always available.

**Свойства** `ToolOffsetView` has properties that can be set in Qt Designer, in Python handler code or (if appropriate) in stylesheets:

**`dialog_code_string`**
Sets which dialog will pop up with numerical entry.

**`text_dialog_code_string`**
Sets which dialog will pop up with text entry.

**metric_template**
Metric numerical data format.

**imperial_template**
Imperial numerical data format.

**styleCodeHighlight**
Current tool-in-use highlight color.

**styleCodeSelected**
Selected highlight color.

In a handler file:

```
self.w.tooloffsetview.setProperty('dialog_code','CALCULATOR')
self.w.tooloffsetview.setProperty('metric_template','%10.3f')
```

and in style sheets:

```
ToolOffsetView{
  qproperty-styleColorHighlist: lightblue;
  qproperty-styleColorSelected: #444;
}
```

**Функции** `ToolOffsetView` has some functions useful for screen builders to add actions:

**add_tool()**
Adds a blank dummy tool (99) that the user can edit to suit.

**delete_tools()**
Deletes the currently checkbox selected tools.

**get_checked_list()**
Returns a list of tools selected by checkboxs.

**set_all_unchecked()**
Uncheck all selected tools.

**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()
```

#### 12.7.2.33 `VersaProbe` - Mill Probing Widget

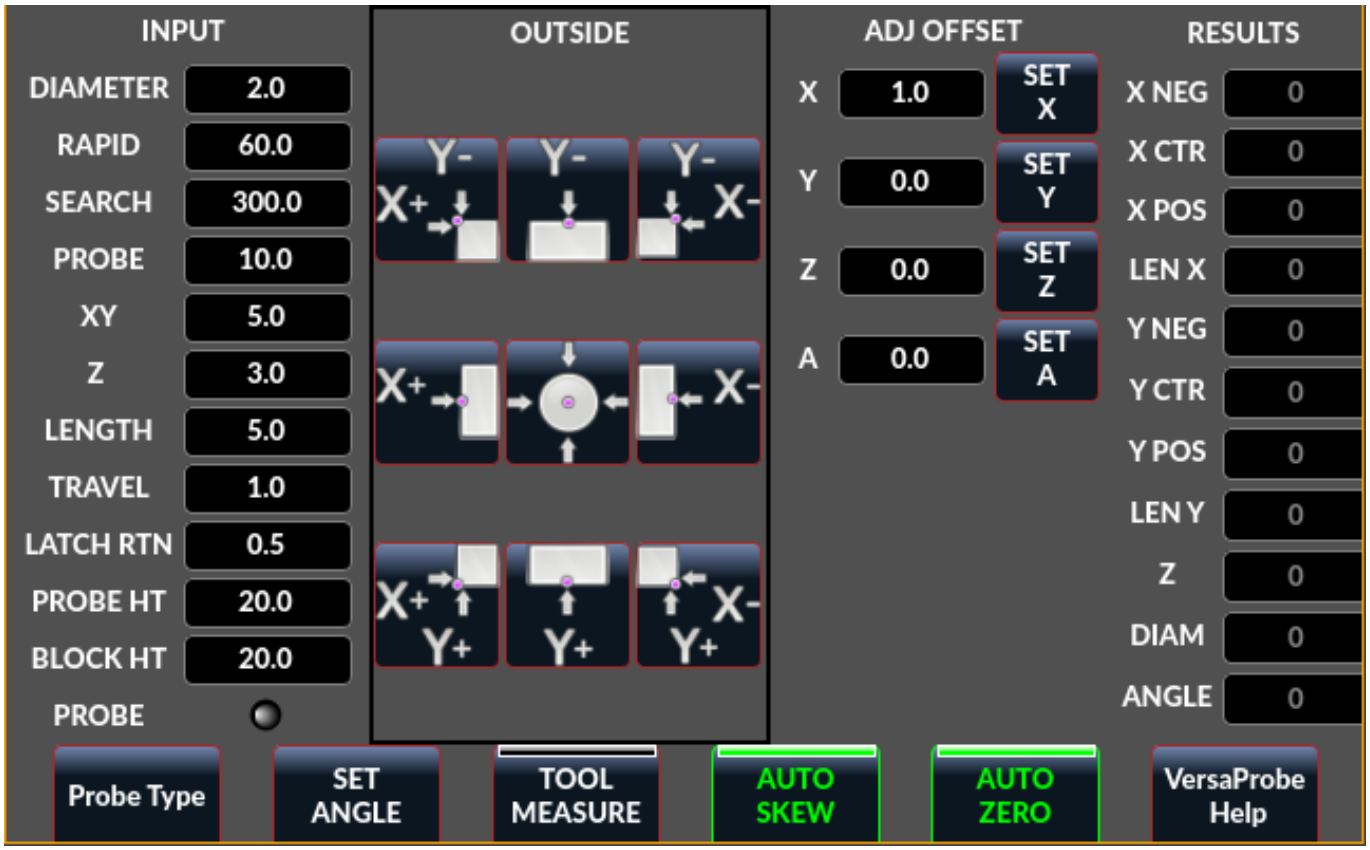



Figure 12.96: QtVCP `VersaProbe`: Mill Probing Widget

Widget for **probing on a mill**. Used by the *QtDragon* screen.

### 12.7.3 Dialog Widgets

Dialogs are used to **present or request immediately required information** in a focused way.

The typical used dialogs can be loaded using the `ScreenOptions` *widget*.

You can also add them directly to the *UI* - but each dialog must have a unique launch name or you will see multiple dialogs displayed, one after another.

**Use dialogs from Python Code** 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)
```

- **Add a callback function that processes the general message**:
  Keep in mind this function will *get all general messages* so the `dict` keynames are not guaranteed to be there. Using the `.get()` function and/or using `try`/`except` is advisable. This function should:
  - check the name and id is the same as we sent,
  - then extract the return value and any user variables.

```
# process the 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` - General Message Dialog Widget

This is a **general message dialog widget**.

If there is a Focus Overlay widget present, it can signal it to display.

If the sound library is set up it can *play sounds*.

There are *options* that can be set when requesting a dialog, these would be added to the message `dict`.

**TITLE**
Title of the dialog window.

**MESSAGE**
Title message text in bold.

**MORE**
Standard text under the heading.

**DETAILS**
Initial hidden text.

**TYPE *(OK|YESNO|OKCANCEL)* , ICON (QUESTION|INFO|CRITICAL|WARNING) , PINNAME**
Not implemented yet.

**FOCUSTEXT *(overlay text|None)***
Text to display if focus overlay is used. Use `None` for no text.

**FOCUSCOLOR (QColor(_R, G, B, A_))**
Color to use if focus overlay is used.

**PLAYALERT**
Sound to play if sound is available, i.e., `SPEAK` <*spoken_message*> .

When using `STATUS` 's `request-dialog` function, the *default launch name* is **`MESSAGE`**.

It is based on PyQt's *QMessagebox*.

#### 12.7.3.2 `ToolDialog` - Manual Tool Change Dialog Widget

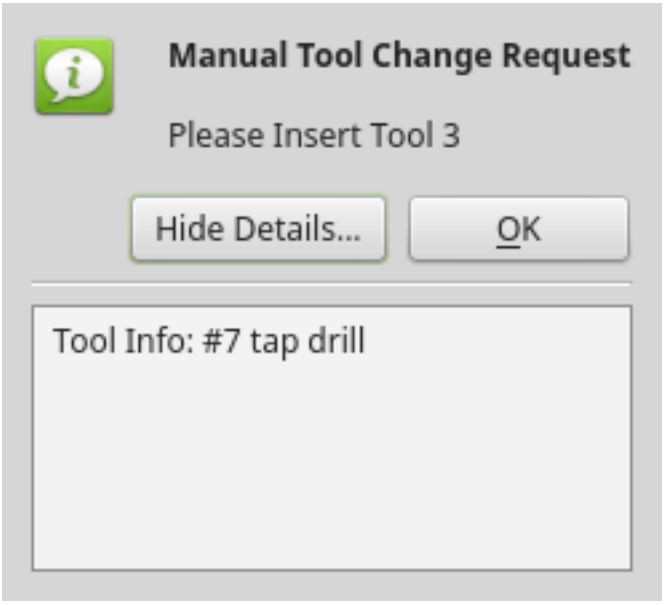



Figure 12.97: QtVCP `ToolDialog`: Manual Tool Change Dialog

This is used as a **manual tool change prompt**.

It has *HAL pins to connect to the machine controller*. The pins are named the same as the original AXIS manual tool prompt and works the same.

The tool change dialog *can only be launched by HAL pins*.

If there is a Focus Overlay widget present, it will signal it to display.

It is based on PyQt's *QMessagebox*.

#### 12.7.3.3 `FileDialog` - Load and Save File Chooser Dialog Widget

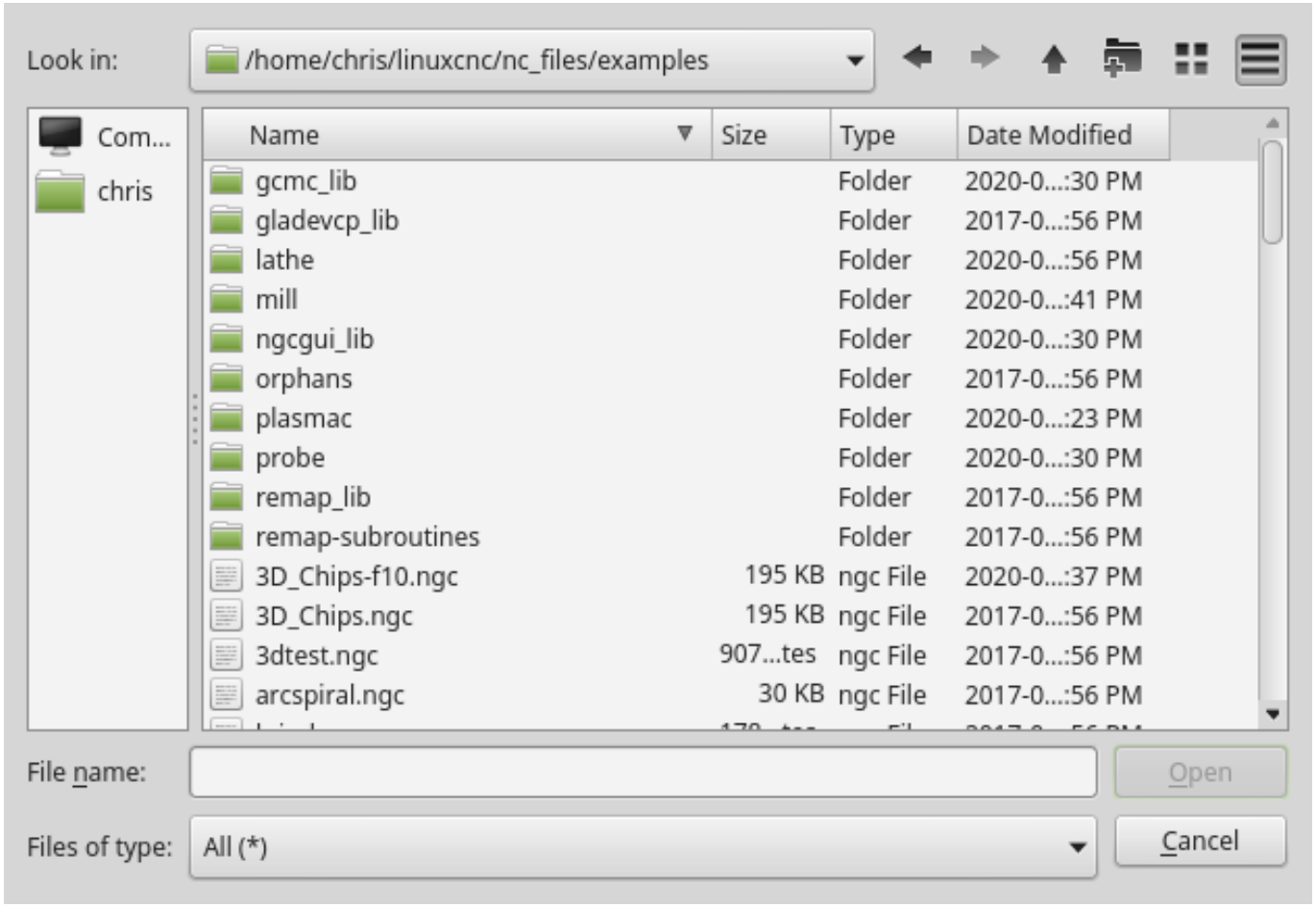



Figure 12.98: QtVCP `FileDialog`: Load and Save File Chooser Widget

This is used to **load G-code files**.

If there is a Focus Overlay widget present, it will signal it to display.

When using `STATUS` 's `request-dialog` function, the default launch names are **`LOAD`** or **`SAVE`**.

There are *options* that can be set when requesting a dialog, these would be added to the message dict:

**`EXTENSIONS`**, **`FILENAME`**, **`DIRECTORY`**

An example Python call, for a *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)
```

And for a *save dialog*

```
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)
```

It is based on PyQt's *QMessagebox*.

#### 12.7.3.4 `OriginOffsetDialog` - Origin Offset Setting Dialog Widget

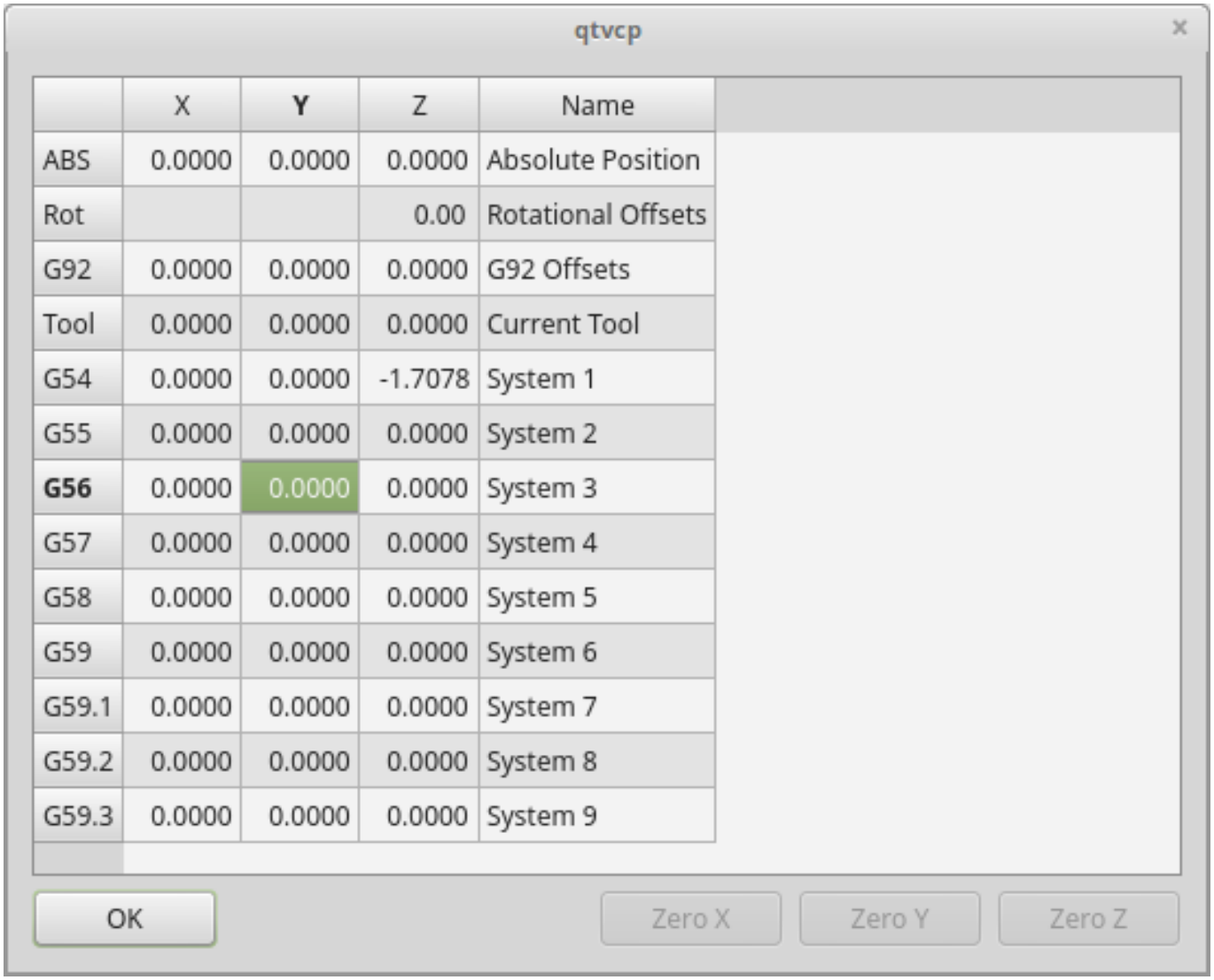



Figure 12.99: QtVCP `OriginOffsetDialog`: Origin Offset Setting Widget

This widget allows one to **modify User System origin offsets directly** in a dialog form.
If there is an Focus Overlay widget present, it will signal it to display.
When using `STATUS` 's `request-dialog` function, the default launch name is **`ORIGINOFFSET`**.
It is based on PyQt's *QDialog*.

#### 12.7.3.5 `ToolOffsetDialog` - Tool Offset Setting Dialog Widget

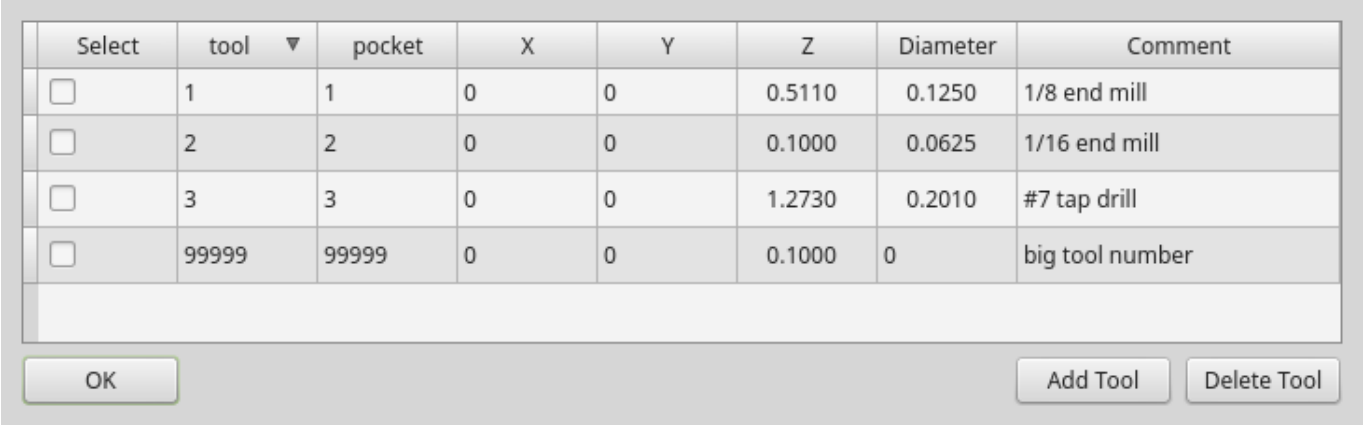



Figure 12.100: QtVCP `ToolOffsetDialog`: Tool Offset Setting Dialog Widget

This widget allows one to **modify Tool offsets directly** in a dialog form.
If there is an Focus Overlay widget present, it will signal it to display.
When using `STATUS` 's `request-dialog` function, the default launch name is **`TOOLOFFSET`**.
It is based on PyQt's *QDialog*.

#### 12.7.3.6 `ToolChooserDialog` - Tool Chooser Dialog Widget

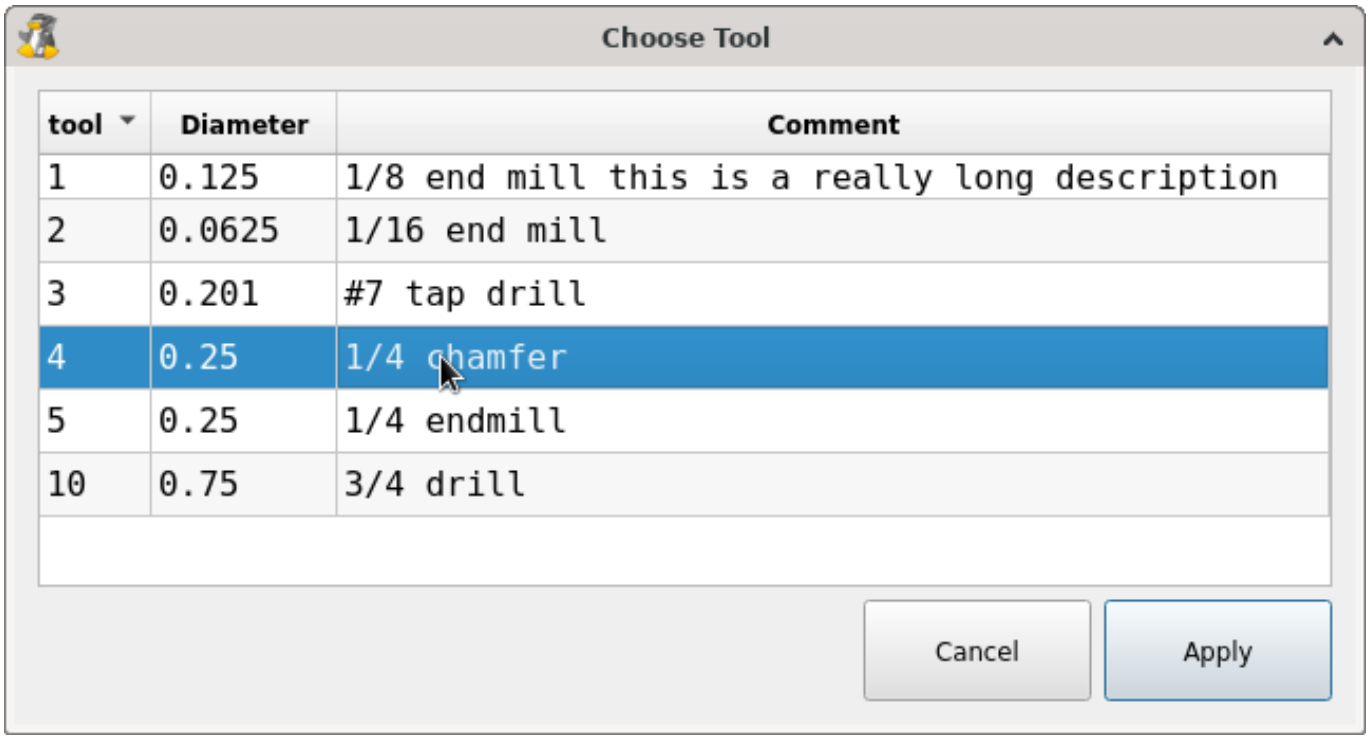



Figure 12.101: QtVCP `ToolChooserDialog`: Tool Chooser Dialog Widget

This widget allows the operator to select one of the tools defined in the tool table. If a tool is selected and *Apply* is pressed or the tool is double-clicked, the dialog will return the tool number selected. This can be used in conjunction with the `OperatorValueLine` widget to create a tool change widget, for example.

If there is an Focus Overlay widget present, it will signal it to display.

When using `STATUS` 's `request-dialog` function, the default launch name is **`TOOLCHOOSER`**.

It is based on PyQt's *QDialog*.

#### 12.7.3.7 `MachineLog` - Machine Events Journal Display Widget

This widget displays various **log event messages** that have been output by the system during the current session. This includes informational messages as well as errors.

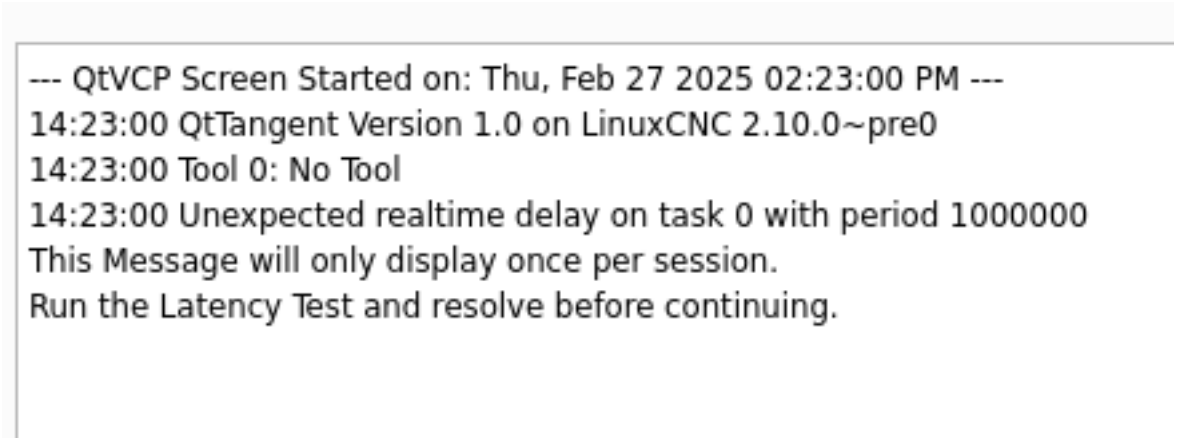



Figure 12.102: QtVCP `MachineLog`: Machine Events Log in `machine_log` (plain) mode

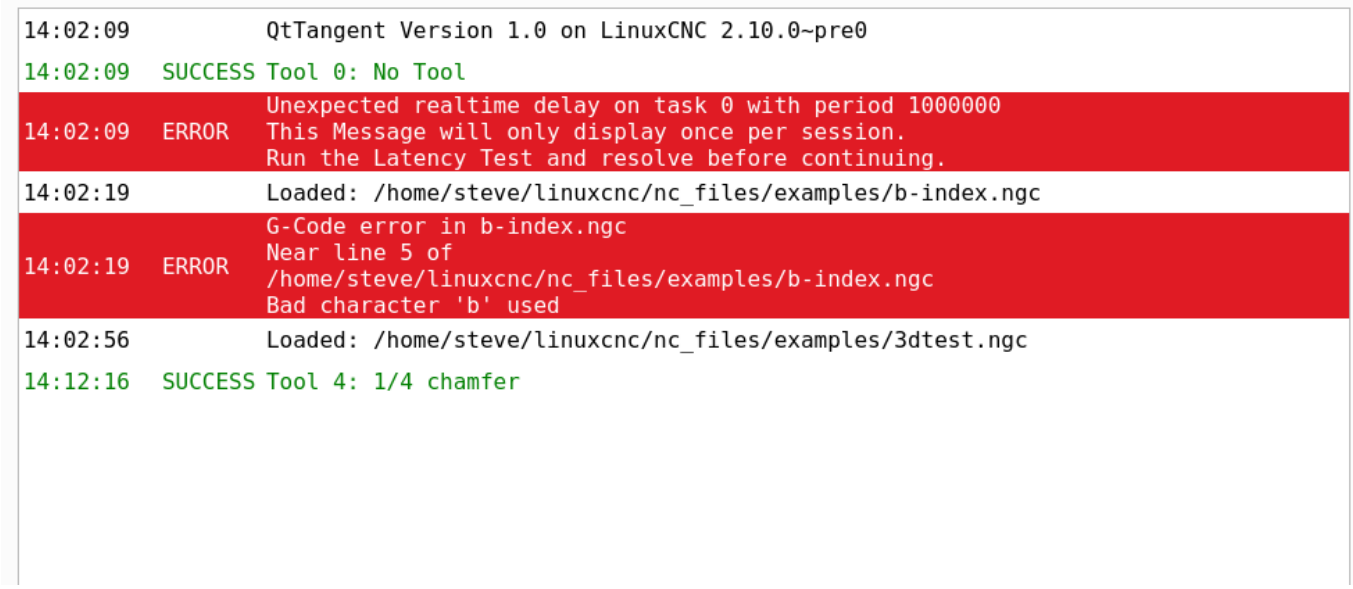



Figure 12.103: QtVCP `MachineLog`: Machine Events Log in `machine_log_severity` mode

Two distinct types of logs may be displayed:

- machine log (plain text or severity highlighted)
- integrator log (plain text only)

The type of log shown by the widget is controlled by the option properties of the widget. By selecting `machine_log_option` or `integrator_log_option` the appropriate log will be displayed. These options will display plain styled logs in a Qt `QTextEdit` widget.

Additionally, there is a `machine_log_severity_option` property that may be chosen that will display the machine log in a variety of colors depending on the severity of the message, by using a `QTableWidget`. The colors may be configured with the properties of the widget.

Severity is conveyed via the `option` value sent along with the `STATUS` signal called `update-machine-log`. The `option` parameter is a comma-delimited list, containing typically

``` text = *an error has occurred.* STATUS.emit(*update-machine-log*, text, *TIME,ERROR*) ```

The log may be cleared by calling the `clear()` method of the widget.

#### 12.7.3.8 `MacroTabDialog` - Macro Launch Dialog Widget

This is a dialog to **display the macrotab widget**.

`MacroTab` displays a *choice of macro programs to run using icons*.

If there is a Focus Overlay widget present, it will signal it to display.

When using ``STATUS``'s `request-dialog` function, the default launch name is **`MACROTAB`**.

It is based on PyQt's *QDialog*.

#### 12.7.3.9 `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`**.

It is based on PyQt's *QDialog*.

#### 12.7.3.10 `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`**.

It is based on PyQt's *QDialog*.

#### 12.7.3.11 `CalculatorDialog` - Calculator Dialog Widget

This is a dialog to **display a calculator for numeric entry**, such as origin offset, spindle RPM, etc. It is primarily intended for touchscreen use, but it has support for physical keyboard input as well.
```

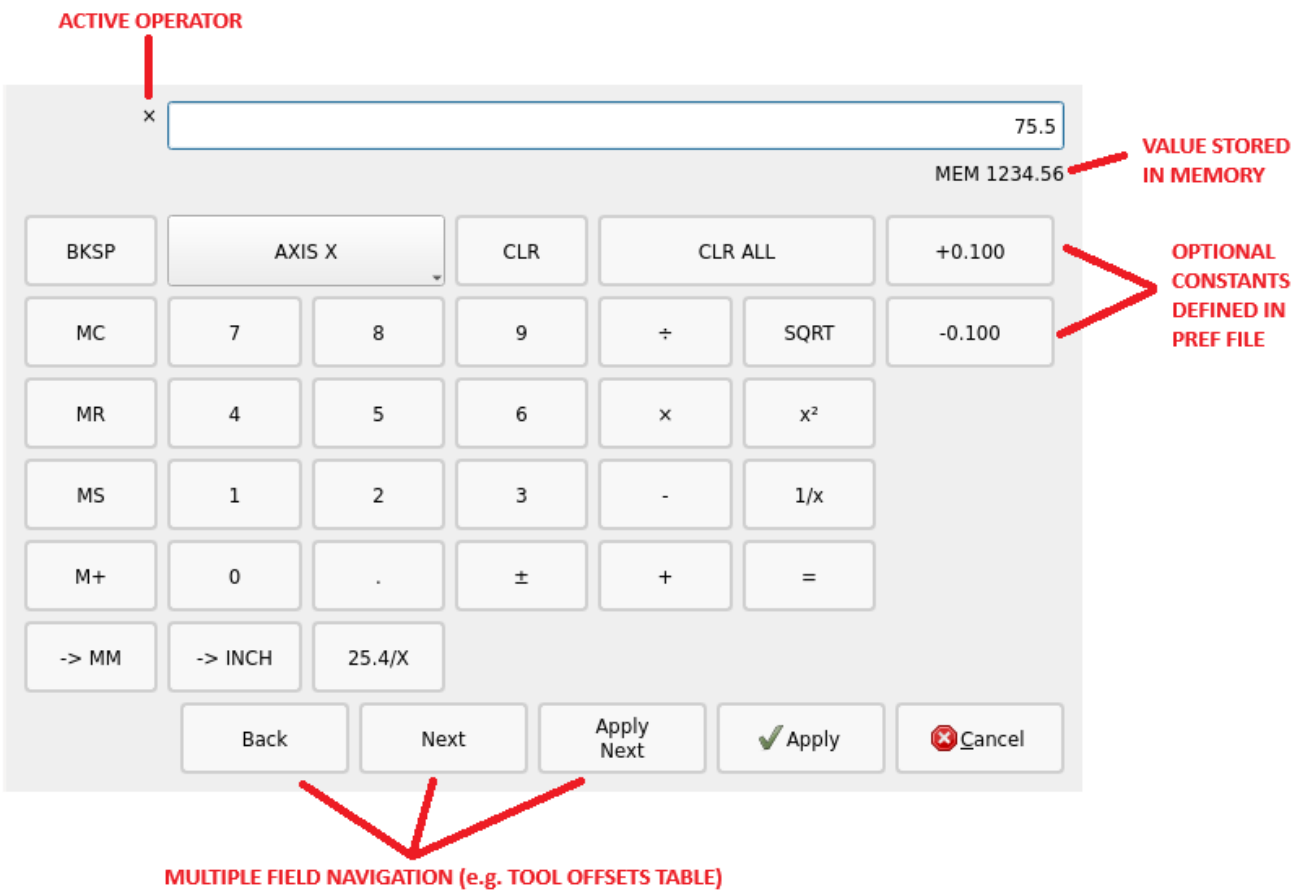



Figure 12.104: QtVCP `CalculatorDialog`: Calculator Dialog Widget

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`**.

It is based on PyQt's *QDialog*.

In the `CALCULATOR` section of the preferences file the following options may be set:

- `constValuesList` - A comma-delimited list of common values you might enter, that will appear on a dedicated row of buttons at the bottom of the calculator. e.g. setting to `0.100, -0.100` would provide two buttons for +0.100 and -0.100 which are commonly used when edge-finding on inch mills. Up to six (6) values may be entered, beyond that the list will be truncated. Values must be valid floating point or integer.
- `onShowBehavior` - A list of optional behaviors that will be triggered when the calculator dialog is shown. Each option must be separated by a comma.
  - `CLEAR_ALL` to issue a **Clear All** each time the calculator is shown. This will clear any previously entered values from the last time the calculator was used and open with the display value set to `0`
  - `FORCE_FOCUS` will force the focus to the calculator input field when the widget is shown. This will allow a physical keyboard to provide input to the widget properly without additional clicks. Also, it has the side-effect of selecting the current value, such that typing from a physical keyboard will replace the existing value unless the text selection is changed.

- `acceptOnReturnKey` - If set to `True`, the calculator will accept the current value and close the dialog when the keyboard return/enter key is pressed. If set to `False`, the return key will be ignored and the **Apply** button must be clicked. For cases where the **Apply Next** button is available, the return key will perform this action instead, and the dialog will remain open.

While the intent of this widget is to provide a touchscreen-friendly interface, it is possible to use a physical keyboard to enter values, typically via a numeric keypad. The keys perform mostly as one would expect, but a few special key functions also exist:

- the **Enter** or **Return** key is equivalent to the **Equal** (=) operator in the cases where a calculation is pending. Otherwise it will perform the **Apply** function if it is enabled via the `acceptOnReturnKey` preference.
- the **minus key** (-) will toggle the sign of the current number in the calculator display when it is hit twice in a row. Otherwise when hit only once it will perform a subtraction operator as expected.
- **Alt+Left Arrow** will perform **Back** function, moving to the previous field for cases where this is supported.
- **Alt+Right Arrow** will perform a **Next** function, moving to the next field for cases where this is supported.
- **Alt+Backspace** will cancel out of the calculator and not provide a return value to the calling widget.

#### 12.7.3.12 `RunFromLine` - Run-From-Line Dialog Widget

Figure 12.105: QtVCP `RunFromLine`: Run-From-Line Dialog Widget

Dialog to **preset spindle settings before running a program from a specific line**.

It is based on PyQt's *QDialog*.

#### 12.7.3.13 `VersaProbeDialog` - Part Touch Probing Dialog Widget

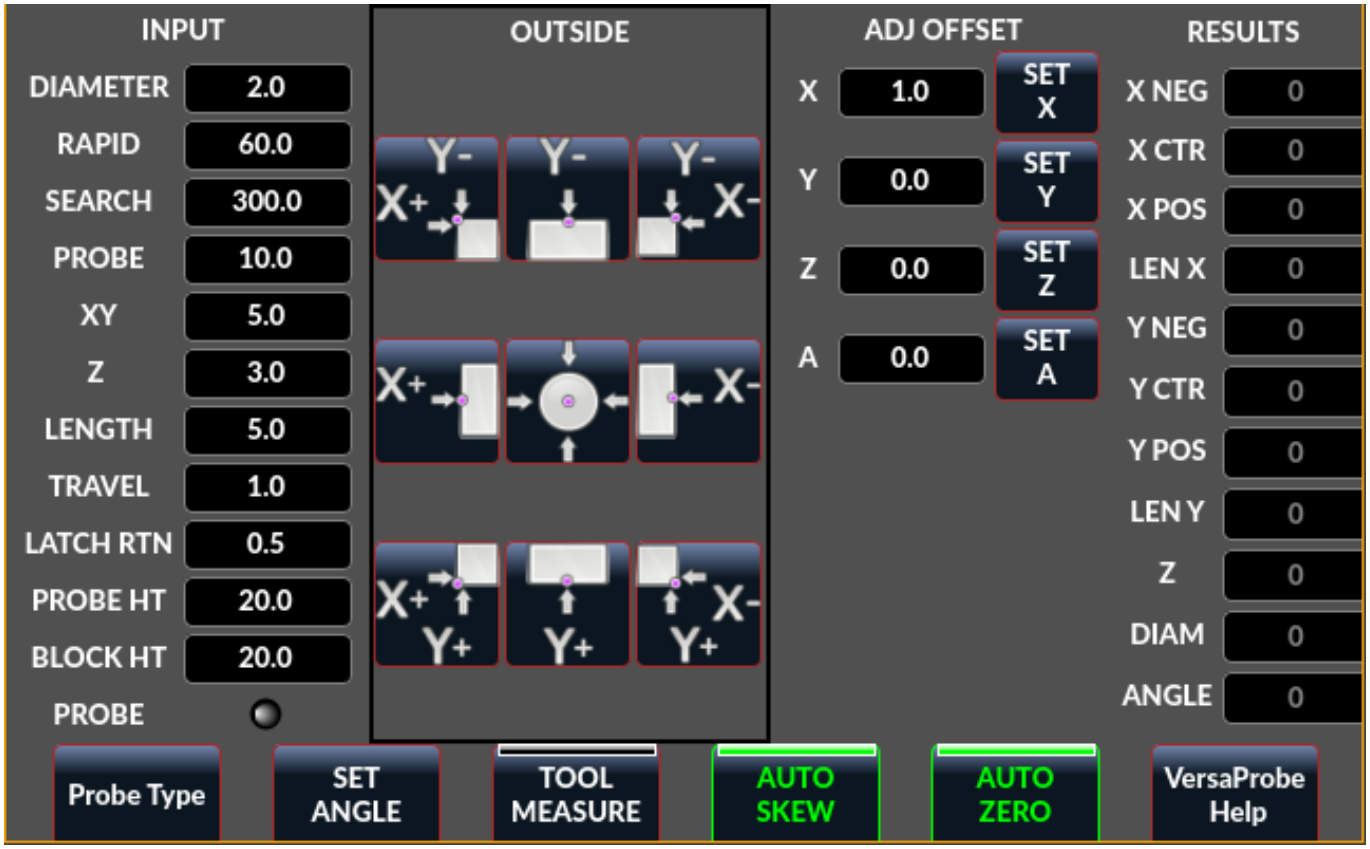



Figure 12.106: QtVCP `VersaProbeDialog`: Part Touch Probing Dialog Widget

This is a dialog to display a **part probing screen based on Verser Probe v2**.
It is based on PyQt's *QDialog*.

#### 12.7.3.14 `MachineLogDialog` - Machine and Debugging Logs Dialog Widget

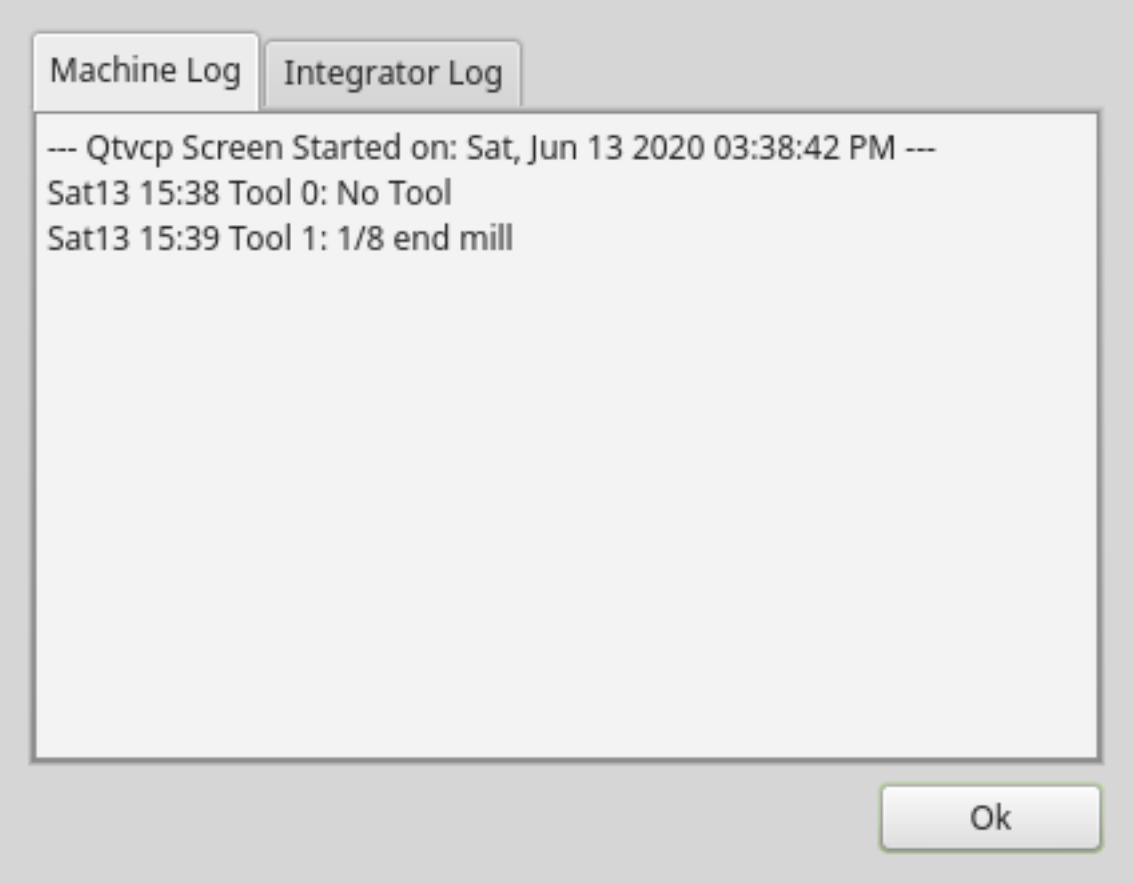



Figure 12.107: QtVCP `MachineLogDialog`: Machine and Debugging Logs Dialog Widget

This is a dialog to **display the machine log and QtVCP's debugging log**.
It is based on PyQt's *QDialog*.

### 12.7.4 Other Widgets

Other available widgets:

#### 12.7.4.1 `NurbsEditor` - NURBS Editing Widget

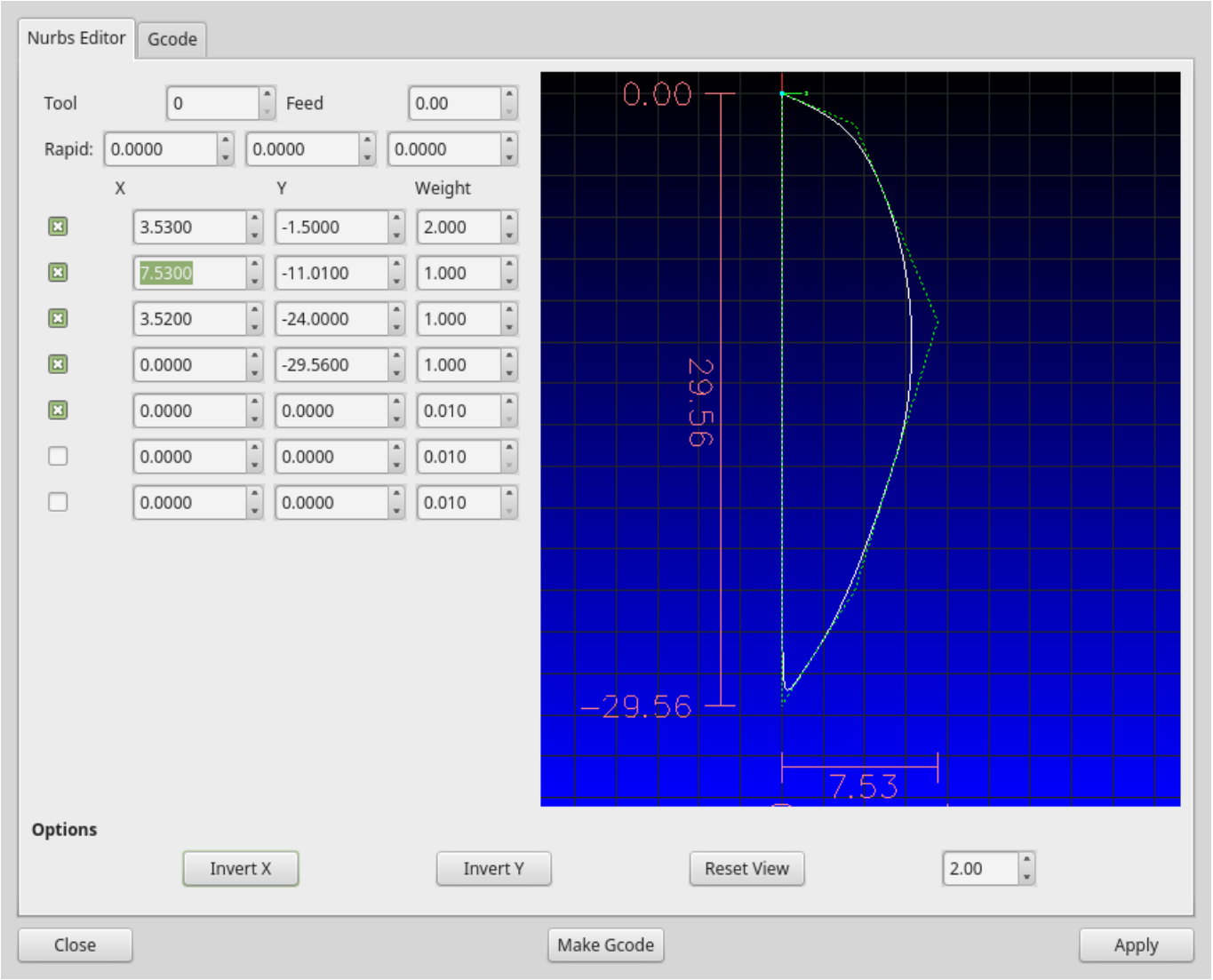



Figure 12.108: QtVCP `NurbsEditor`: NURBS Editing Widget

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.

It is based on PyQt's *QDialog*.

#### 12.7.4.2 `JoyPad` - 5 button D-pad Widget

It is the base class for the `HALPad` widget.

This widget looks and acts like a **5 button D-pad, with a LED like indicators in a ring**.

You can put text or icons in each of the button positions.

You can *connect to output signals* when the buttons are pressed.

There are also *input slots* to change the color of the indicator(s).

**ENUMS** There are **enumerated constants used to reference indicator positions**.
They are used in Qt Designer editor's property editor or in Python code.

**NONE , LEFT, L , RIGHT, R , CENTER, C , TOP, T , BOTTOM, B , LEFTRIGHT, X , TOPBOTTOM, A**
For Python handler code, you use the widget name in Qt Designer plus the reference constant:

```
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:

Callable Functions

**reset_highlight()**
Clears the highlight indicator.

**set_highlight(_button_, state=_True_)**
Set the highlight indicator in position `button` to state `state`.
You can use *strings letters* (`LRCTBXA`) or *position* `ENUMS` for the button argument.

**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_)**
Sets the buttons pop-up tooltip descriptive text.

**setLight(_state_)**
Sets the highlight indicator to the `True` color or `False` color.
The `set_highlight()` function must be used prior to set the indicator to use.

**Сигналы** 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***

They are based on PyQt's *Signal* (`QtCore.pyqtSignal()`)

**Slots** Slots can be connected to in Qt Designer editor or Python code:

**`set_colorStateTrue()` , `set_colorStateFalse()` , `set_colorState(_bool_)` , `set_true_color(_str_)`**

**Свойства** These can be set in stylesheets or Python code:

**`highlightPosition`**
Set the indicator position.

**`setColorState`**
Select the color state of the indicator.

**`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`**
Color selection for the button text.

**`button_font`**
Font selection for the button text.

The above properties could be set 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;
}
```

- **In 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**.

You will see them as a collapsible header in the Qt Designer properties column.

#### 12.7.5.1 `IndicatedPushButtons`

This class **modifies `QPushButton` behaviour**.

**`indicator_option`** *puts a* LED *on the top of the button*.

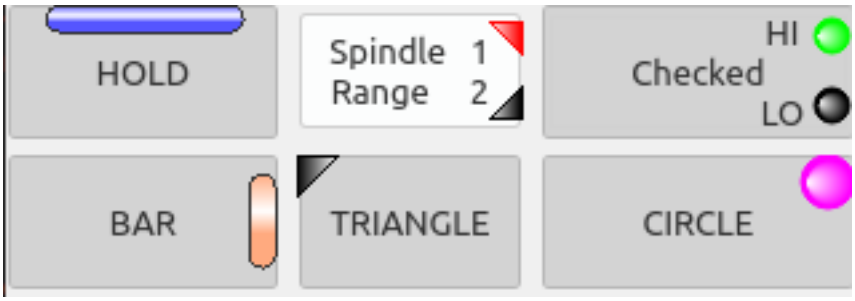



Figure 12.109: QtVCP `PushButton`: IndicatedPushButton Button, LED Indicator Option

It can be a *triangle*, *circle*, *top bar*, or *side bar*.
The *triangle* and *circle* LEDs can display double vertical LEDs optionally each with it's own HAL pin.
+ The *size* and *position* can be adjusted.

It will indicate:

- the **current state of the button**, or
- the **state of a HAL pin**, or
- **LinuxCNC status**.

**Свойства** These properties are available to customize the indicator (not all are applicable to every LED shape):

**`doubleIndicator`**
: With triangle or round LEDs add a second vertical LED.

**`on_color`** , **`off_color`** , **`flashIndicator`**
: When the indicator is true, flash the LED on and off.

**`flashRate`**
: Скорость мигания

**`indicator_size`**
: Размер треугольного СИД

**`circle_diameter`**
: Diameter of round LED

**`shape_option`**
: 0-4 LED shape type

**right_edge_offset**
Space from right edge

**top_edge_offset**
Space from top edge

**height_fraction**
Used for bar Leds

**width_fraction**
Used for bar Leds

**corner_radius**
Indicator corner radius.

The LED indicator color can be defined in a *stylesheet* with the following code added to the `.qss` file:

```
Indicated_PushButton{
  qproperty-on_color: #000;
  qproperty-off_color: #444;
}
```

Or for a particular button:

```
Indicated_PushButton #button_estop{
  qproperty-on_color: black;
  qproperty-off_color: yellow;
}
```

**Варианты** `IndicatedPushButton` have **exclusive options**:

**indicator_HAL_pin_option**
Adds a `halpin`, named `<buttonname>-led` that controls the button indicator state.

**indicator_status_option**
Makes the LED indicate the state of these selectable LinuxCNC status:

- *Is Estopped*
- *Is On*
- *All Homed*
- *Is Joint Homed*
- *Idle*
- *Paused*
- *Flood*
- *Mist*
- *Block Delete*
- *Optional Stop*
- *Manual*
- *MDI*
- *Auto*
- *Spindle Stopped*
- *Spindle Forward*
- *Spindle Reverse*
- *On Limits*

Some `indicator_status_options` holds a property that can be used with a *stylesheet* to change the color of the button based on the state of the property in LinuxCNC.
Currently these status properties can be used to auto style buttons:

- `is_estopped_status` will toggle the `isEstop` property
- `is_on_status` will toggle the `isStateOn` property
- `is_manual_status`, `is_mdi_status`, `is_auto_status` will toggle the `isManual`, `isMDI`, `isAuto` properties.
- `is_homed_status` will toggle the *isAllHomed* property

Here is a sample stylesheet entry setting the background of mode button widgets when LinuxCNC is in that mode:

```
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;
}
```

Often, having the button disabled and enabled based on the state of LinuxCNC's motion controller is necessary.

There are several properties that can be selected to aid with this:

**`isAllHomedSentive`**, **`isOnSensitive`**, **`isIdleSensitive`**, **`isRunSensitive`**, **`isRunPausedSensitive`**, **`is`**

You can select multiple properties for combined requirements.

Choosing the **`checked_state_text_option`** allows a *checkable* button to *change the text based on its checked state*.

It uses the following properties to specify the text for each state:

**`true_state_string`**, **`false_state_string`**
: \\n will be converted to a newline.

You can set/change these in stylesheets:

```
ActionButton #action_aux{
  qproperty-true_state_string: "Air\\nOn";
  qproperty-false_state_string: "Air\\nOff";
}
```

The **`python_command_option`** allow small snippets of Python code to be run from the push of a button, without having to edit the handler file. Though, it can call functions in the handler file.

When using the `command_string` properties.

**true_python_cmd_string**
A Python command that will be called when the button is toggled `True`.

**false_python_cmd_string**
A Python command that will be called when the button is toggled `False`.

*Special capitalized words* will give access to the following:

**INSTANCE**
Will give access to the widgets instances and handler functions.
E.g., `INSTANCE.my_handler_function_call(True)`

**ACTION**
Will give access to QtVCP's `ACTION` library.
E.g., `ACTION.TOGGLE_FLOOD()`

**PROGRAM_LOADER**
Will give access to QtVCP's `PROGRAM_LOADER` library.
E.g., `PROGRAM_LOADER.load_halshow()`

**HAL**
Will give access to HAL's Python module.
E.g., `HAL.set_p('motion.probe-input','1')`

### 12.7.6 Import-Only Widgets

These widgets are usually the **base class widget for other QtVCP widgets**.

They are *not available directly from the Qt Designer editor* but could be **imported and manually inserted**.

They could also be **subclassed** to make a similar widget with new features.

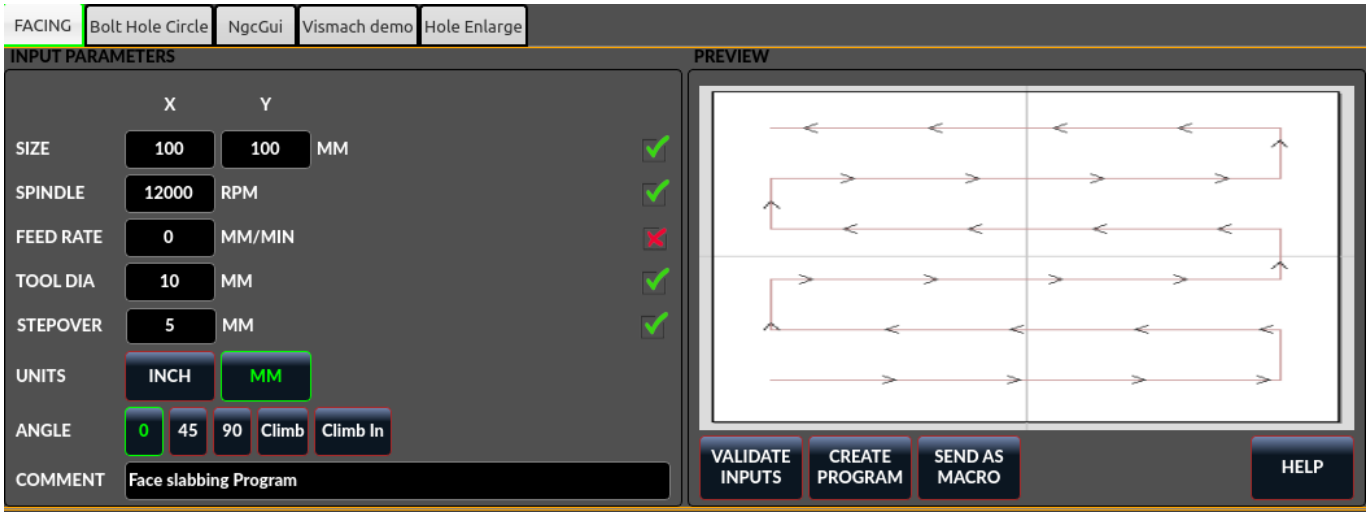
#### 12.7.6.1 Auto Height

Widget for measuring two heights with a probe.
For setup.

#### 12.7.6.2 G-code Utility

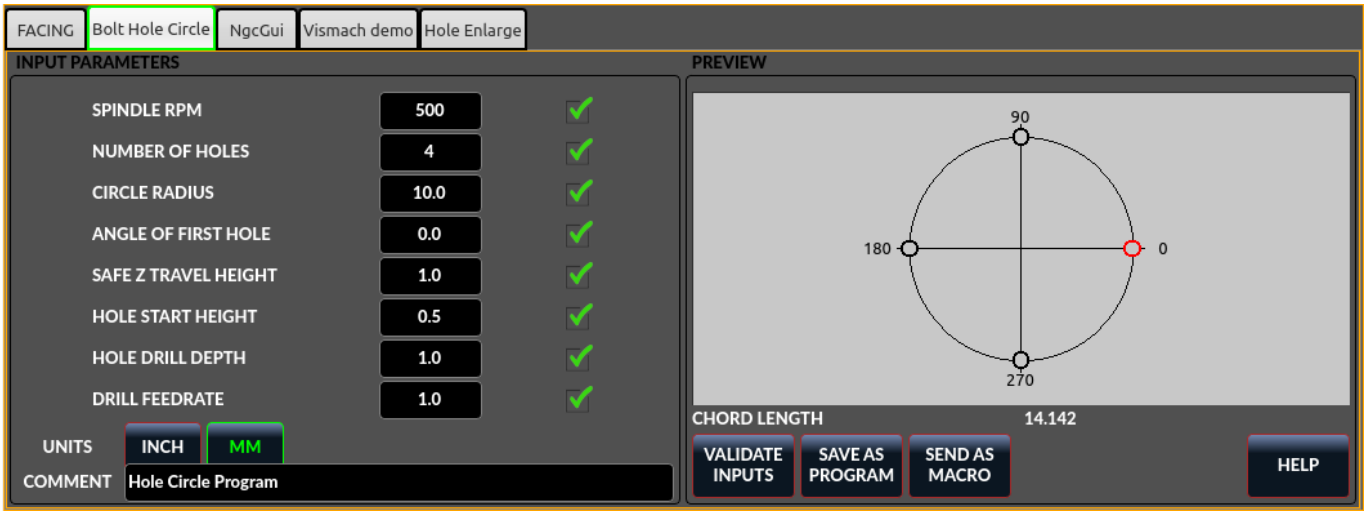
Widgets for performing common machining processes.

### 12.7.6.3 Facing



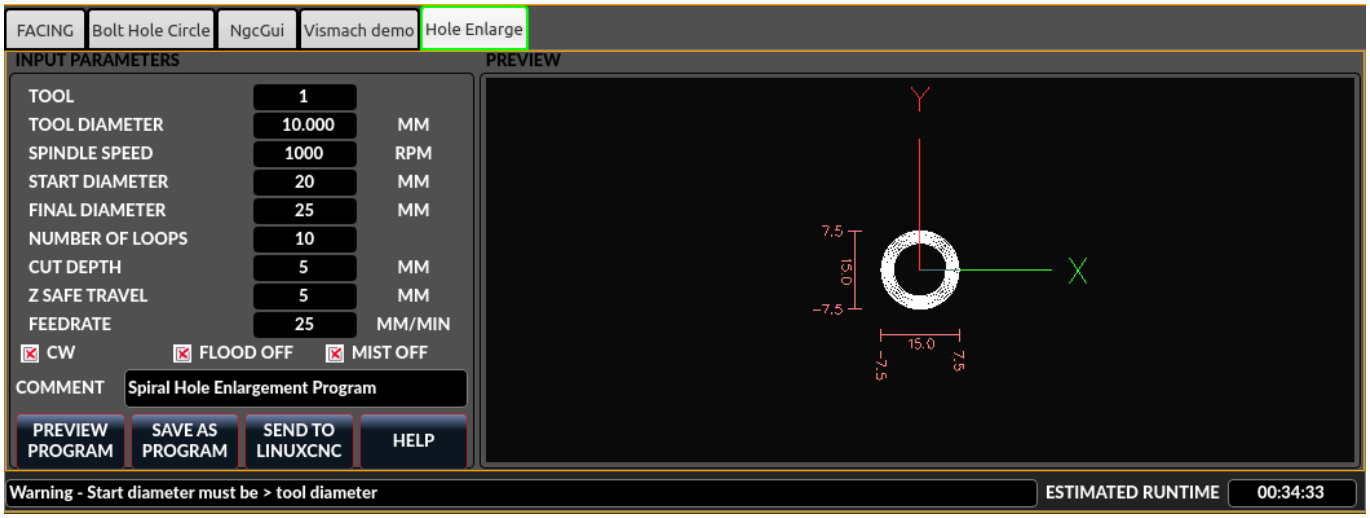

Slab or face a definable area with different strategies.

### 12.7.6.4 Hole Circle



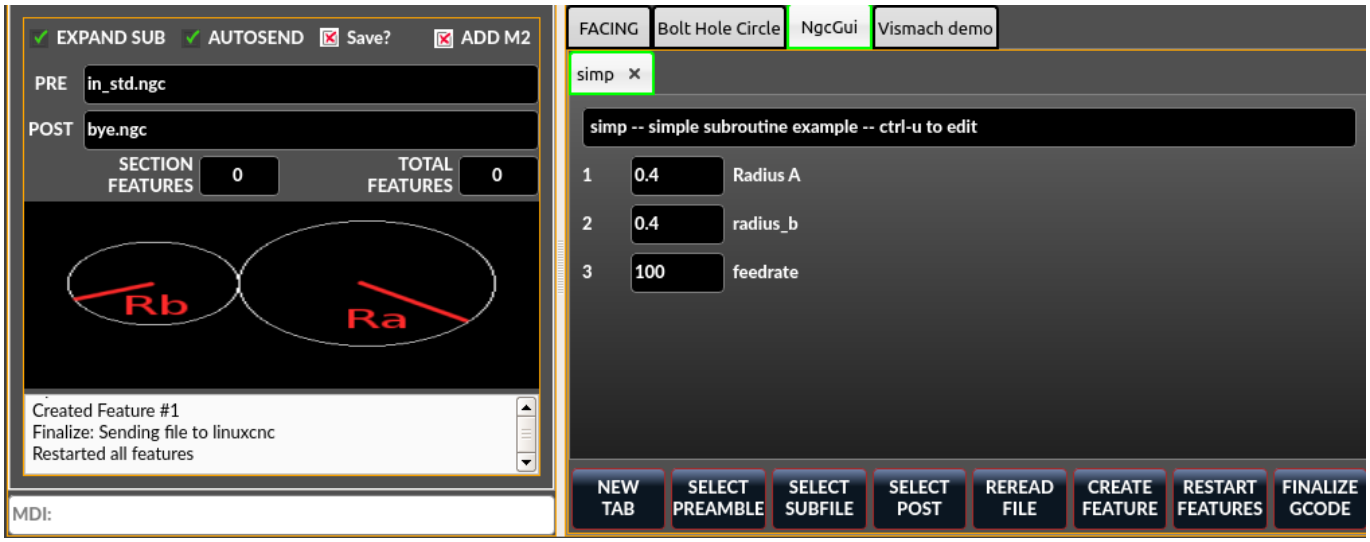

Drill multiple holes on a bolt hole circle.

#### 12.7.6.5 Hole Enlarge




Use an end mill to enlarge a drilled hole.

#### 12.7.6.6 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
```

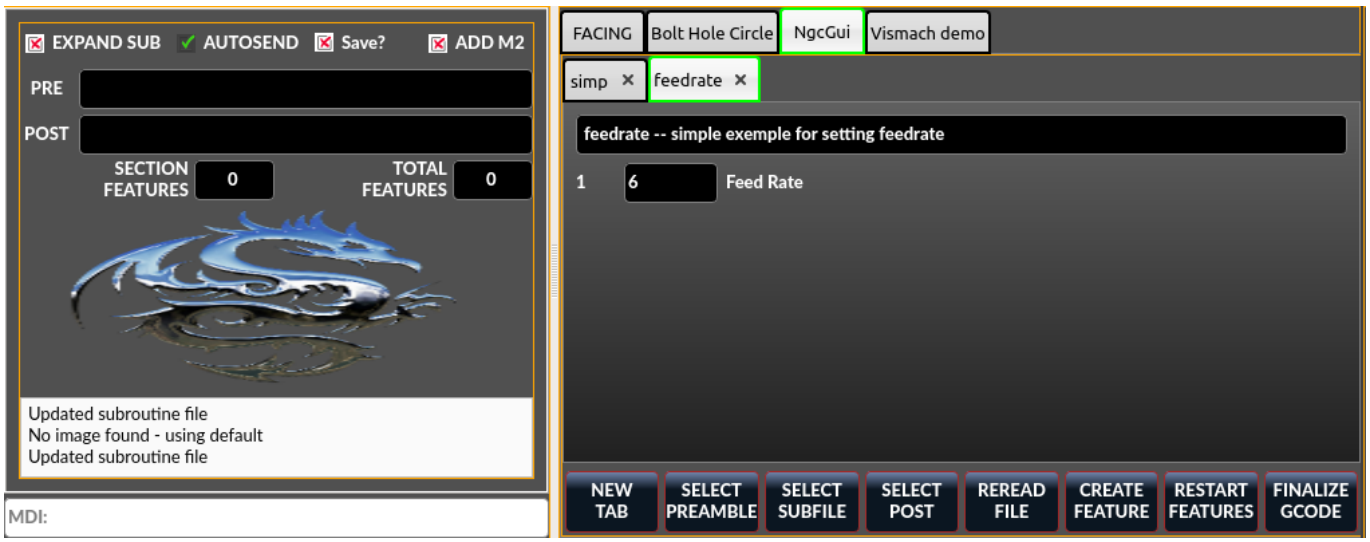
```
# pre selected programs tabs
# specify filenames only, files must be in the NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

- *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:

- For creating a subroutine for use with NGCGUI, the filename and the subroutine name must be the same.
- 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.
- The variables used must be numbered variables and must not skip number.
- Comments and presets may be included.
- 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
```



#### 12.7.6.7 Qt PDF

Allows adding loadable PDFs to a screen.

#### 12.7.6.8 Qt Vismach

Use this to build/add OpenGl simulated machines.

#### 12.7.6.9 Окно выбора Hal

This widget is combobox that will allows selection of a pin or signal on the system.

```
from qtvcp.widgets.hal_selectionbox import HALSelectionBox

    def buildComboBox(self):
        # combo box for HAL pin selection
        combobox = HALSelectionBox()
        combobox.setShowTypes([combobox.PINS,combobox.SIGNALS])
        combobox.setPinTypes([combobox.HAL_BIT], direction = [combobox.HAL_IN])
        combobox.setSignalTypes([combobox.HAL_BIT], driven = [False,True])
        combobox.hal_init()
        combobox.selectionUpdated.connect(lambda w: self.signalSelected(w))

    def signalSelected(self, sig):
        print('Watching:',sig)
```

Есть вызовы функций

```
# set the list of types to show from: PINS SIGNALS
combobox.setShowTypes([combobox.PINS])

# set the pin types to show: HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# and a list of directions: HAL_IN HAL_OUT
combobox.setPinTypes(types=[combobox.HAL_BIT], direction = [HAL_IN])

# set the signal types to show: HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# and a list of driven/undriven (by a connected pin) to show
combobox.setSignalTypes( types=[combobox.HAL_BIT], driven = [True,True])
```

## 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 Использование

- **Import Status modules**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Status
```

- **Instantiate Status module**
  Добавьте этот код Python в раздел создания экземпляра:

```
STATUS = Status()
```

- **Connect to STATUS messages**
  Use *GObject syntax*.

#### 12.8.1.2 Пример

For example, you can catch machine on and off messages.

---

**Note**
The example below shows the *two common ways of connecting signals*, one of them *using lambda*. **lambda** is *used to strip off or manipulate arguments* from the status message before calling the function. You can see the difference in the called function signature: The one that uses lambda does not accept the status object - lambda did not pass it to the function.

---

- Place these commands into the `[INITIALIZE]` section of the Python handler file:

```
STATUS.connect('state-on', self.on_state_on)
STATUS.connect('state-off', lambda: w, self.on_state_off())
```

  In this example code, when LinuxCNC is in *"machine on" state* the function `self.on_state_on` will be called.
  When LinuxCNC is in *"machine off" state* the function `self.on_state_off` will be called.

- These would call functions that looks like these:

```
def on_state_on(self, status_object):
    print('LinuxCNC machine is on')
def on_state_off(self):
    print('LinuxCNC machine is off')
```

### 12.8.2 Info

**Info**— это библиотека для **сбора и фильтрации данных из INI-файла**.

#### 12.8.2.1 Available data and defaults

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = self.INI.getint("KINS","JOINTS", fallback=0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.getstring(section, "TYPE", fallback="LINEAR")
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 units per minute
MIN_LINEAR_JOG_VEL = 60 units per minute
MAX_LINEAR_JOG_VEL = 300 units per minute

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = self.INI.getint("TRAJ", "SPINDLES", fallback=1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY
```

#### 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")
```

#### 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 Использование

- **Import `Info` module**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Info
```

- **Instantiate `Info` module**.
  добавьте этот код Python в раздел создания экземпляра:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

INFO = Info()
```

- **Для доступа к информации `INFO`** используйте следующий общий синтаксис:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

### 12.8.3 Action

**Action** библиотека используется для **управления контроллером движения LinuxCNC**.
It tries to hide incidental details and add convenience methods for developers.

#### 12.8.3.1 Helpers

There are some **helper functions**, mostly used for this library's support:

**get_jog_info (_num_) , jnum_check(_num_) , ensure_mode(_modes_) , open_filter_program(_filena**

Open G-code filter program.

#### 12.8.3.2 Использование

- **Import Action module**
  добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Action
```

- **Instantiate Action module**
  добавьте этот код Python в раздел создания экземпляра:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

ACTION = Action()
```

- **Access ACTION commands**
  Используйте общий синтаксис, например:

```
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()          # Toggles pause/resume
ACTION.PAUSE_MACHINE()
ACTION.RESUME()

ACTION.SET_MAX_VELOCITY_RATE(rate)
ACTION.SET_RAPID_RATE(rate)
ACTION.SET_FEED_RATE(rate)
ACTION.SET_SPINDLE_RATE(rate)

ACTION.SET_JOG_RATE(rate)
ACTION.SET_JOG_INCR(incr)
ACTION.SET_JOG_RATE_ANGULAR(rate)
ACTION.SET_JOG_INCR_ANGULAR(incr, text)

ACTION.SET_SPINDLE_ROTATION(direction = 1, rpm = 100, number = 0)
ACTION.SET_SPINDLE_FASTER(number = 0)
ACTION.SET_SPINDLE_SLOWER(number = 0)
ACTION.SET_SPINDLE_STOP(number = 0)

ACTION.SET_USER_SYSTEM(system)

ACTION.ZERO_G92_OFFSET()
ACTION.ZERO_ROTATIONAL_OFFSET()
ACTION.ZERO_G5X_OFFSET(num)

ACTION.RECORD_CURRENT_MODE()
ACTION.RESTORE_RECORDED_MODE()

ACTION.SET_SELECTED_AXIS(jointnum)

ACTION.DO_JOG(jointnum, direction)
ACTION.JOG(jointnum, direction, rate, distance=0)

ACTION.TOGGLE_FLOOD()
ACTION.SET_FLOOD_ON()
ACTION.SET_FLOOD_OFF()

ACTION.TOGGLE_MIST()
ACTION.SET_MIST_ON()
ACTION.SET_MIST_OFF()

ACTION.RELOAD_TOOLTABLE()
ACTION.UPDATE_VAR_FILE()

ACTION.TOGGLE_OPTIONAL_STOP()
ACTION.SET_OPTIONAL_STOP_ON()
ACTION.SET_OPTIONAL_STOP_OFF()

ACTION.TOGGLE_BLOCK_DELETE()
ACTION.SET_BLOCK_DELETE_ON()
ACTION.SET_BLOCK_DELETE_OFF()

ACTION.RELOAD_DISPLAY()
ACTION.SET_GRAPHICS_VIEW(view)
```

```
ACTION.UPDATE_MACHINE_LOG(text, option=None):

ACTION.CALL_DIALOG(command):

ACTION.HIDE_POINTER(state):

ACTION.PLAY_SOUND(path):
ACTION.PLAY_ERROR():
ACTION.PLAY_DONE():
ACTION.PLAY_READY():
ACTION.PLAY_ATTENTION():
ACTION.PLAY_LOGIN():
ACTION.PLAY_LOGOUT():
ACTION.SPEAK(speech):

ACTION.BEEP():
ACTION.BEEP_RING():
ACTION.BEEP_START():

ACTION.SET_DISPLAY_MESSAGE(string)
ACTION.SET_ERROR_MESSAGE(string)

ACTION.TOUCHPLATE_TOUCHOFF(search_vel, probe_vel, max_probe,
       z_offset, retract_distance, z_safe_travel, rtn_method=None, error_rtn = None)
```

## 12.8.4 Qhal

A library for HAL component/system interactions.

### 12.8.4.1 Attributes

Вот функции, которые можно вызвать для объекта Qhal:

**setUpdateRate(cyclerate)**
: Set cycle rate in ms

**newPin(name, pin type constant, pin direction constant)**
: returns a new QPin object

**getPinObject(name)**
: returns an existing named QPin object

**getValue(name)**
: returns the named pin, signal, or parameter’s value, use the full component.pin name.

**setPin(name,value)**
: sets the named pin’s value, use the full component.pin name.

**setSignal(name,value)**
: sets the named signal’s value, use the full component.pin name.

**makeUniqueName(name)**
: returns an unique HAL pin name string by adding *-x* (a number) to the given pin name string

**exit()**
: выгрузка компонента

#### 12.8.4.2 Constants

Вот доступные константы:

- **HAL_BIT**
- **HAL_FLOAT**
- **HAL_S32**
- **HAL_U32**
- **HAL_IN**
- **HAL_OUT**
- **HAL_IO**
- **HAL_RO**
- **HAL_RW**

#### 12.8.4.3 Рекомендации

Available object references:

- **comp** компонентный объект
- **hal** the hal library object

### 12.8.5 QPin

A wrapper class around HAL pins

#### 12.8.5.1 Сигналы

There are 3 Qt signals that the QPin pin can be connect to:

- **value_changed** will call a named function with an argument of the current value (depreciated)
- **pinValueChanged** will call a named function with arguments of the pin object and the current value
- **isDrivenChanged** will call a named function with arguments of the pin object and current state when the pin is (un)connected to a driving pin

#### 12.8.5.2 Attributes

Вот функции, которые можно вызвать для объекта QPin:

- **<Pin object>.get()** returns the current value of the pin object
- **<Pin object>.set(X)** sets the value of the pin object to the value X
- **<Pin object>.text()** returns the pin name string

#### 12.8.5.3 Рекомендации

Available object references:

- **hal** the hal library object

#### 12.8.5.4 Пример

**Add a function that gets called when the pin state changes**

```
from qtvcp.core import Qhal
QHAL = Qhal()

########################################
# Special Functions called from QtVCP
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.pin_button_in = QHAL.newpin('cycle-start-in',QHAL.HAL_BIT, QHAL.HAL_IN)
    self.pin_button_in.pinValuechanged.connect(self.buttonChanged)
    self.pin_button_in.isDrivenChanged.connect(lambda p,s: self.buttonDriven(p,s))

def buttonChanged(self, pinObject, value):
    print('Pin name:{} changed value to {}'.format(pinObject.text(), value))

def buttonDriven(self, pinObject, state):
    message = 'not driven by an output pin'
    if state:
        message = 'is driven by an output pin'
    print('Pin name:{} is {}'.format(pinObject.text(), message))
```

### 12.8.6 Tool

This library **handles tool offset file changes**.

**Warning**
**LinuxCNC doesn't handle third party manipulation of the tool file well.**

#### 12.8.6.1 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**.

---

**Note**
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**.

---

**Note**
This also **requires remap code to add the wear offsets t tool change time**.

---

## 12.8.7 Path

**Path** module gives **reference to important files paths**.

### 12.8.7.1 Referenced Paths

**PATH.PREFS_FILENAME**
The preference file path.

**PATH.WORKINGDIR**
The directory QtVCP was launched from.

**PATH.IS_SCREEN**
Is this a screen or a VCP?

**PATH.CONFIGPATH**
Launched configuration folder.

**PATH.RIPCONFIGDIR**
The Run-in-place config folder for QtVCP screens.

**PATH.BASEDIR**
Base folder for LinuxCNC.

**PATH.BASENAME**
The Qt Designer files name (no ending).

**PATH.IMAGEDIR**
The QtVCP image folder.

**PATH.SCREENDIR**
The QtVCP builtin Screen folder.

**PATH.PANELDIR**
The QtVCP builtin VCP folder.

**PATH.HANDLER**
Handler file Path.

**PATH.HANDLERDIR**
Directory where the Python handler file was found.

**PATH.XML**
QtVCP UI file path.

**PATH.HANDLERDIR**
Directory where the UI file was found.

**PATH.QSS**
QtVCP QSS file path.

**PATH.PYDIR**
LinuxCNC’s Python library.

**PATH.LIBDIR**
The QtVCP library folder.

**PATH.WIDGET**
The QtVCP widget folder.

**PATH.PLUGIN**
The QtVCP widget plugin folder.

**PATH.VISMACHDIR**
Directory where prebuilt Vismach files are found.

Not currently used:

**PATH.LOCALEDIR**
Locale translation folder.

**PATH.DOMAIN**
Translation domain.

#### 12.8.7.2 Helpers

There are some helper functions available:

```
file_list = PATH.find_vismach_files()
directory_list = PATH.find_screen_dirs()
directory_list = PATH.find_panel_dirs()
```

#### 12.8.7.3 Использование

- **Import `Path` module**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.core import Path
```

- **Instantiate `Path` module**
  Добавьте этот код Python в раздел создания экземпляра:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

PATH = Path()
```

### 12.8.8 `VCPWindow`

**`VCPWindow`** module gives **reference to the `MainWindow` and 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.

#### 12.8.8.1 Использование

- **Import `VCPWindow` module**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **Чтобы создать экземпляр `VCPWindow` модуля**
  добавьте этот код Python в раздел создания экземпляра:

```
###########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
###########################################

WIDGETS = VCPWindow()
```

## 12.8.9 `Aux_program_loader`

**`Aux_program_loader`** module allows an easy way to **load auxiliary programs LinuxCNC often uses**.

### 12.8.9.1 Helpers

**`load_halmeter()`**
*Halmeter* is used to **display one HAL pin data**.
Load a `halmeter` with:

```
AUX_PRGM.load_halmeter()
```

**`load_ladder()`**
Load *ClassicLadder* PLC program:

```
AUX_PRGM.load_ladder()
```

**`load_status()`**
Load LinuxCNC `status` program:

```
AUX_PRGM.load_status()
```

**`load_halshow()`**
Load *HALshow,* configure display program:

```
AUX_PRGM.load_halshow()
```

**`load_halscope()`**
Load *HALscope* program:

```
AUX_PRGM.load_halscope()
```

**`load_tooledit()`**
Load *Tooledit* program:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

**`load_calibration()`**
Load *Calibration* program:

```
AUX_PRGM.load_calibration()
```

**`keyboard_onboard()`**
Load *onboard/Matchbox keyboard*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

### 12.8.9.2 Использование

- **Import `Aux_program_loader` module**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Instantiate `Aux_program_loader` module**
  Добавьте этот код Python в раздел создания экземпляра:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

AUX_PRGM = Aux_program_loader()
```

### 12.8.10 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.

#### 12.8.10.1 Использование

**Import Keylookup module** Чтобы импортировать эти модули, добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.keybindings import Keylookup
```

**Instantiate Keylookup module** To instantiate `Keylookup` module* so you can use it, add this Python code to your instantiate section:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

KEYBIND = Keylookup()
```

---

**Note**
`Keylookup` requires code under the `processed_key_event` function to call `KEYBIND.call()`.
Most handler files already have this code.

---

In the handler file, under the *initialized function* use this general syntax to **create keybindings**:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Here we add a keybinding for `F10`, `F11` and `F12`:

```
########################################
# Special Functions called from QtVCP
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    KEYBIND.add_call('Key_F10','on_keycall_F10',None)
    KEYBIND.add_call('Key_F11','on_keycall_override',10)
    KEYBIND.add_call('Key_F12','on_keycall_override',20)
```

And then we need to **add the functions that get called**.
In the handler file, under the `KEY BINDING CALLS` section, add this:

```
####################
# KEY BINDING CALLS #
####################

def on_keycall_F12(self,event,state,shift,cntrl,value):
    if state:
        print('F12 pressed')

def on_keycall_override(self,event,state,shift,cntrl,value):
    if state:
        print('value = {}'.format(value))
```

#### 12.8.10.2 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",

}
```

### 12.8.11 Messages

**Messages** module is used to **display pop up dialog messages on the screen**.

These messages are:

- *defined in the INI file under the `[DISPLAY]` heading,* and
- *controlled by HAL pins*.

Use this style if you need independent HAL pins for each dialog message.

#### 12.8.11.1 Свойства

**BOLDTEXT**
: Generally is a title.

**TEXT**
: Text below title, and usually longer.

**DETAIL**
: Text hidden unless clicked on.

**PINNAME**
: Basename of the HAL pin(s).

**TYPE**
Specifies whether it is a (can have dialog and status options together): **status** - shown in the *status bar and the notify dialog*.
Requires no user intervention.

- **nonedialog** - specifically does not show a dialog.
  **okdialog** - *requiring the user to click OK to close the dialog*.
  OK messages have *two HAL pins*:
  - One HAL pin to launch the dialog, and
  - one to signify it is waiting for a response. **yesnodialog** - *requiring the user to select yes or no buttons to close the dialog*.
    Yes/No messages have *three HAL pins*:
  - Один для показа диалога,
  - One for waiting, and
  - one for the answer. ***okcanceldialog*** - *requiring the user to select ok or cancel*
    *Ok/Cancel messages have _three HAL pins*:
  - Один для показа диалога,
  - One for waiting, and
  - one for the answer.
- ***closepromptdialog*** - *requiring the user to select*

By default, `STATUS` messages for `focus_overlay` and alert sound will be sent when the dialog shows. This allows screen *focus* dimming/blurring and sounds to be added to alerts.

#### 12.8.11.2 HAL Контакты

Названия контактов HAL будут использовать эти шаблоны:

***<SCREEN BASENAME>.<PINNAME>***
вызывая s32 контакт

***<SCREEN BASENAME>.<PINNAME>-waiting***
*Waiting for the user's response* output bit pin

***<SCREEN BASENAME>.<PINNAME>-response***
Контакт выходного бита *Ответ пользователя*

***<SCREEN BASENAME>.<PINNAME>-response-s32***
*The user response* output s32 pin

#### 12.8.11.3 Examples

Here are sample INI message definition code blocks that would be found under the `[DISPLAY]` heading:

- Status bar and desktop notify pop up message:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

- Pop up dialog asking a Yes/No question:

```
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 asking an OK answer + Status bar and desktop notification:

```
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
```

The `ScreenOptions` widget can automatically set up the message system.

## 12.8.12 multimessages

**`Messages`** module is used to **display pop up dialog messages on the screen**.

These messages are:

- *defined in the INI file under the `[DISPLAY]` heading,* and
- *controlled by one s32 HAL pin per defined id*.
- *each message is called by a corresponding number on the s32 pin.*

Use this style of user messages for instance when a VFD sends error messages encoded as numbers. It uses common invoke/response/wait HAL pins for all (per ID name) multimessage dialogs. The HAL pin names would use these patterns:

***<SCREEN BASENAME>.<ID NAME>***
: вызывая s32 контакт

***<SCREEN BASENAME>.<ID NAME>-waiting***
: *Waiting for the user's response* output bit pin

***<SCREEN BASENAME>.<ID NAME>-response***
: Контакт выходного бита *Ответ пользователя*

***<SCREEN BASENAME>.<ID NAME>-response-s32***
: *The user response* output s32 pin

### 12.8.12.1 Свойства

**`TITLE`**
: This is the title shown on the dialog window.

**`TEXT`**
: Text below title, and usually longer.

**`DETAIL`**
: Text hidden unless clicked on.

**TYPE**
Specifies type of message the user sees (can have dialog and status options together): **status** - shown in the *status bar and the notify dialog*.
Requires no user intervention.

- **nonedialog** - specifically does not show a dialog.
  **okdialog** - *requiring the user to click OK to close the dialog*.
  OK messages use *two HAL pins*:
  - One HAL pin to launch the dialog, and
  - one to signify it's waiting for response. **yesnodialog** - *requiring the user to select yes or no buttons to close the dialog*.
    Yes/No messages use *three HAL pins*:
  - Один для показа диалога,
  - One for waiting, and
  - one for the answer.

By default, `STATUS` messages for `focus_overlay` and alert sound will be sent when the dialog shows. This allows screen *focus* dimming/blurring and sounds to be added to alerts.

#### 12.8.12.2 Examples

Here are sample INI message definition code blocks that would be found under the `[DISPLAY]` heading:

```
[DISPLAY]
MULTIMESSAGE_ID = VFD

MULTIMESSAGE_VFD_NUMBER = 1
MULTIMESSAGE_VFD_TYPE = okdialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 1
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 1
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 1
MULTIMESSAGE_VFD_ICON = WARNING

MULTIMESSAGE_VFD_NUMBER = 2
MULTIMESSAGE_VFD_TYPE = nonedialog status
MULTIMESSAGE_VFD_TITLE = VFD Error: 2
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR MESSAGE NUMBER 2
MULTIMESSAGE_VFD_DETAILS = DETAILS for VFD error 2
MULTIMESSAGE_VFD_ICON = INFO
```

### 12.8.13 Notify

**`Notify`** module is used to **send messages that are integrated into the desktop**.

It uses the `pynotify` library.

Ubuntu/Mint does not follow the standard so you can't set how long the message stays up for.
I suggest fixing this with the `notify-osd` package available from this PPA (DISCONTINUED due to move of Ubuntu to Gnome).

Notify *keeps a list of all the alarm messages since starting* in **`self.alarmpage`**.
If you click *'Show all messages'* in the notify popup, it will print them to the terminal.

The `ScreenOptions` widget can automatically set up the notify system.

Typically `STATUS` *messages* are used to sent notify messages.

#### 12.8.13.1 Свойства

You can set the:

**title**
: Notification message title text.

**message**
: Notification message content text.

**icon**
: Notification message icon.

**timeout**
: How long the message stays up for.

### 12.8.14 Preferences

**Preferences** module allows one to **load and save preference data permanently to storage media**.

The ScreenOptions widget can automatically set up the preference system.

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_**. Also all widgets can have access to a specific preferences file at initialization time. The ScreenOptions widget can automatically set up the preference file.

### 12.8.15 Player

This module **allows playing sounds using Gstreamer, beep and Espeak**.

It can:

- **play sound/music files** using *Gstreamer* (non blocking),
- **play sounds** using the beep library (currently blocks while beeping),
- **speak words** using the espeak library (non blocking while speaking).

There are *default alert sounds* using Mint or FreeDesktop default sounds.

You can play arbitrary sounds or even songs by specifying the path.

STATUS has *messages to control Player module*.

The ScreenOptions widget can automatically set up the audio system.

#### 12.8.15.1 Sounds

**Alerts** There are default **alerts** to choose from:

- ERROR
- READY
- ATTENTION

- RING
- DONE
- LOGIN
- LOGOUT

**Beeps** There are three **beeps**:

- BEEP_RING
- BEEP_START
- BEEP

#### 12.8.15.2 Использование

- **Import `Player` module**
  Добавьте этот код Python в раздел импорта:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.audio_player import Player
```

- **Instantiate `Player` module**
  Добавьте этот код Python в раздел создания экземпляра:

```
########################################
# **** INSTANTIATE LIBRARIES SECTION **** #
########################################

SOUND = Player()
SOUND._register_messages()
```

  The **`_register_messages()`** function connects the audio player to the STATUS library so sounds can be played with the STATUS message system.

#### 12.8.15.3 Пример

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')
```

### 12.8.16 Virtual Keyboard

This library allows you to **use STATUS messages to launch a virtual keyboard**.

It uses Onboard or Matchbox libraries for the keyboard.

### 12.8.17 Toolbar Actions

This library supplies **prebuilt submenus and actions for toolbar menus and toolbar buttons**. Toolbuttons, menu and toolbar menus are:

- *built in Qt Designer*, and
- *assigned actions/submenus in the handler file*.

#### 12.8.17.1 Actions

**estop , power , load , reload , gcode_properties , run , pause , abort , block_delete , optional_stop**

Toggles dimensions display.

#### 12.8.17.2 Submenus

**recent_submenu , home_submenu , unhome_submenu , zero_systems_submenu , grid_size_submenu**

Menu to set graphic grid size

#### 12.8.17.3 Использование

Here is the typical code to add to the relevant *handler file* sections:

```
############################
# **** IMPORT SECTION **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

############################################
# **** instantiate libraries section **** #
############################################

TOOLBAR = ToolBarActions()
```

#### 12.8.17.4 Examples

- Assigning Tool Actions To Toolbar Buttons

```
########################################
# Special Functions called from QtVCP
########################################

# At this point:
#   * the widgets are instantiated,
#   * the HAL pins are built but HAL is not set ready.
def initialized__(self):
    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)
```

- Add a custom toolbar function:

```
####################
# GENERAL FUNCTIONS #
####################

def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

### 12.8.18 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*.

#### 12.8.18.1 Builtin Samples

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
```

#### 12.8.18.2 Primitives Library

Provides the **basic building blocks of a simulated machine**.

**`Collection`**
: A `collection` is an **object of individual machine parts**.

  This holds a **hierarchical list** of primitive shapes or *STL objects* that operations can be applied to.

**`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
- *read a HAL system pin directly* if the component argument is set to `None`.

**`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
- *read a HAL system pin directly* if the component argument is set to `None`.

**`HalToolCylinder`**
This object will build a *CylinderZ object* that will **change size and length based on loaded tool dimension** (from the tool table)

It reads the `halui.tool.diameter` and `motion.tooloffset.z` *HAL pins*.

Example from mill_xyz sample:

```
toolshape = CylinderZ(0)
toolshape = Color([1, .5, .5, .5], [toolshape])
tool = Collection([
    Translate([HalTranslate([tooltip], None, "motion.tooloffset.z", 0, 0, - ↩
        MODEL_SCALING)], 0, 0, 0),
    HalToolCylinder(toolshape)
])
```

**`Track`**
**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`**
**Build a sphere** from *center* and *radius* coordinates.

**`TriangleXY`, `TriangleXZ`, `TriangleYZ`**
**Build a triangle** in the *specified plane* by giving the *corners Z coordinates* for each side.

**`ArcX`**
**Build an arc** by specifying

**`Box`**
**Build a box** specified by the *6 vertex coordinates*.

**`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**.

> **Note**
> 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`**
**Applies a color** to the *parts of a collection*.

**`AsciiSTL, AsciiOBJ`**
**Loads a STL or OBJ data file** as a *Vismach part*.

#### 12.8.18.3 Использование

**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** Instantiate the simulation widget and add it to the screen's main layout:

```
########################################
# Special Functions called from QtVCP
########################################

# At this point:
#   * the widgets are instantiated,
#   * the HAL pins are built but HAL is not set ready.
def initialized__(self):
    machine = MILL.Window()
    self.w.mainLayout.addWidget(machine)
```

#### 12.8.18.4 More Information

More information on how to build a custom machine simulation in the Qt Vismach chapter.

## 12.9 QtVismach

**Vismach** is a set of **Python functions that can be used to create and animate models of machines**.

This chapter is about the Qt embedded version of Vismach, also see: https://sa-cnc.com/linuxcnc-vismach/ .

### 12.9.1 Введение

Vismach displays the model in a **3D viewport** and the **model parts are animated as the values of associated HAL pins change**.

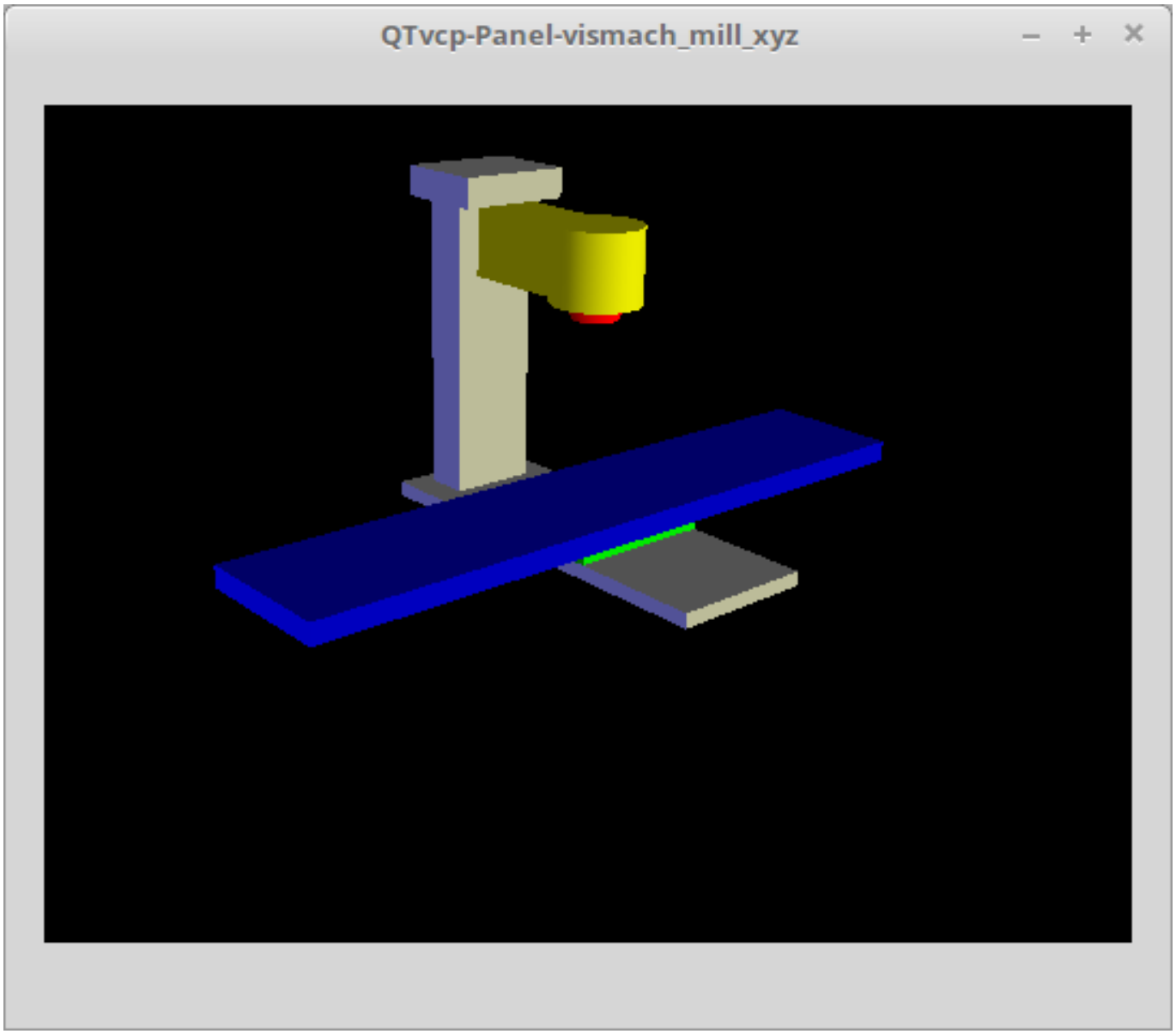



Figure 12.110: QtVismach 3D Viewport

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()
```

You can select between the two options in the functions that take these entries:

**hal_comp**
: The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

**hal_pin**
: The *name of the BIT HAL IN pin* that will change the color.
If hal_comp is *None* then this must be the full name of a system pin otherwise this is the pin name excluding the component name.

### 12.9.5 Creating Parts

#### 12.9.5.1 Import STL or OBJ Files

It is probably easiest to:

- *create geometry in a CAD package*
- *import into the model script using the `AsciiSTL()` or `AsciiOBJ()` functions.*

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.

---

**Note**
It is much easier to move while building if the origin of the model is at a rotational pivot point.

---

#### 12.9.5.2 Build from Geometric Primitives

Alternatively parts can be *created inside the model script from a range of shape primitives*.

**`assembly = collction([part1,part2,part3])`**
Collection is a general container of related parts

Many shapes are *created at the origin* and need to be *moved to the required location* after creation.

**`cylinder = CylinderX(x1, r1, x2, r2)`**, **`cylinder = CylinderY(y1, r1, y2, r2)`**, **`cylinder = Cyl`**

Creates a *(optionally tapered) cylinder on the given axis* with the given radii at the given points on the axis.

**`sphere = Sphere(x, y, z, r)`**
Creates a *sphere of radius r at (x,y,z)*.

**`triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2)`**, **`triangle = TriangleXZ(x1, z1, x2, z`**

Creates a *triangular plate between planes* defined by the last two values parallel to the specified plane, with vertices given by the three coordinate pairs.

**`arc = ArcX(x1, x2, r1, r2, a1, a2)`**
Create an *arc shape*.

**`box = Box(x1, y1, z1, x2, y2, z2)`**
Creates a *rectangular prism with opposite corners* at the specified positions and edges parallel to the XYZ axes.

**`box = BoxCentered(xw, yw, zw)`**
Creates an xw by yw by zw *box centred on the origin*.

**`box = BoxCenteredXY(xw, yw, z)`**
Creates a *box ground on WY plane* of width xw / yw and height z.

Composite parts may be created by assembling these primitives either at creation time or subsequently:

```
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.

### 12.9.6.1 Translating Model parts

**part1 = Translate([part1], x, y, z)**
Move part1 the specified distances in x, y and 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"
```

Otherwise the component instance would be specified and the pin name of that component would be specified. `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**
*Rotation angles* are in degrees, so for a rotary joint with a 0-1 scaling you would need to use an angle scale of 360.

#### 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()])
```

#### 12.9.7.5 HAL Adjustable Primitives

All shape primitives can have HAL pin names substituted for coordinates.
Either by adding the component object as the first variable and substituting the pinname string for a coordinate, or
by substituting the full component/pinname string for a coordinate.

This example creates a *rectangular prism with opposite corners* at the specified positions and edges parallel to the XYZ axes.
The Z start coordinate will be controlled by the HAL pin *Zstart*.

```
box = Box(component, x1, y1, 'Zstart', x2, y2, z2)
box = Box(x1, y1, 'componentName.Zstart', x2, y2, z2)
```

### 12.9.8 Assembling the model

In order for parts to move together they need to be assembled with the **`Collection()` command**.

It is important to **assemble the parts and define their motions in the correct sequence**.

For example to create a moving head milling machine with a rotating spindle and an animated draw bar you would:

- Create the head main body.
- Create the spindle at the origin.
- Define the rotation.
- Move the head to the spindle or spindle to the head.
- Create the draw bar.
- Define the motion of the draw bar.
- Assemble the three parts into a head assembly.
- Define the motion of the head assembly.

In this example the spindle rotation is indicated by rotation of a set of drive dogs:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Finally a **single collection of all the machine parts, floor and work** (if any) needs to be created.

For a *serial machine* each new part will be added to the collection of the previous part.

For a *parallel machine* there may be several "base" parts.

Thus, for example, in `scaragui.py` link3 is added to link2, link2 to link1 and link1 to link0, so the final model is created by:

```
model = Collection([link0, floor, table])
```

Whereas a VMC model with separate parts moving on the base might have

```
model = Collection([base, saddle, head, carousel])
```

## 12.9.9 Other functions

### 12.9.9.1 Цвет

Sets the *display color of the part*.

**part = Color([_colorspec_], [_part_])**
Note that unlike the other functions, the part definition comes second in this case.

**_colorspec_**
Three (0-1.0) RGB values and opacity. [R,G,B,A]
For example [1.0,0,0,0.5] for a 50% opacity red.

### 12.9.9.2 HALColorFlip

Sets the *display color of the part based on a designated HAL bit pin state*.

**part = HALColorFlip([_colorspec_], [_colorspec_], [_part_], hal_comp, hal_pin)**
Note that unlike the other functions, the part definition comes second in this case.

**_colorspec_**
Three (0-1.0) RGB values and opacity.
For example [1.0,0,0,0.5] for a 50% opacity red.

**hal_comp**
The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

**hal_pin**
The *name of the BIT HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

### 12.9.9.3 HALColorRGB

Sets the *display color of the part based on a designated HAL U32 pin value*.
The color is decoded from the U32 value. each color is a 0-255 decimal value (shown here in hex)
red = 0xXXXXXXRR
green = 0xXXXXGGXX
blue = 0xXXBBXXXX
combined as 0xXXBBGGRR

**part = HALColorRGB([_part_], hal_comp, hal_pin, alpha=1.0)**

**hal_comp**
The *HAL component Object* or None.
In QtVCP if you are reading *system pins* directly, then the component argument is set to `None`.

**hal_pin**
The *name of the U32 HAL IN pin* that will change the color.
if hal_comp is *None* then this must be the full name of a system pin other wise this is the pin name excluding the component name.

**alpha=**
Sets the opacity. (0-1.0)

#### 12.9.9.4 Heads Up Display

Creates a *heads-up display* in the Vismach GUI to display items such as axis positions, titles, or messages.

**myhud = Hud()**

```
myhud = Hud()
myhud.show(”Mill_XYZ”)‘
```

#### 12.9.9.5 HAL Heads Up Display

A more advanced version of the Hud that allows HAL pins to be displayed:

**myhud = HalHud()**

```
myhud = HalHud()
myhud.display_on_right()
myhud.set_background_color(0,.1,.2,0)
myHud.set_text_color(1,1,1)
myhud.show_top(”Mill_XYZ”)
myhud.show_top(”------------”)
myhud.add_pin(’axis-x: ’,”{:10.4f}”,”axis.x.pos-cmd”)
myhud.add_pin(’axis-y: ’,”{:10.4f}”,”axis.y.pos-cmd”)
myhud.add_pin(’axis-z: ’,”{:10.4f}”,”axis.z.pos-cmd”)
myhud.show(”-------------”)
```

Some of the available HalHUD function:

- set_background_color(red, green, blue, alpha)
- add_pin(text, format, pinname)
- set_text_color(red, green, blue)

#### 12.9.9.6 Скрыть коллекцию

```
 HideCollection is a container that uses a HAL pin to control display of the  ↩
    contained parts. +
A logic high on the HAL pin will hide the contained parts.
```

```
comp.newpin("hide-chuck", hal.HAL_BIT, hal.HAL_IN)
# <snip make a machine with an A axis chuck assembly>
chuckassembly = HideCollection([chuckassembly],comp,'hide-chuck')
# also can be used like this
chuckassembly = HideCollection([chuckassembly],None,'myvismach.hide-chuck')
```

#### 12.9.9.7 Plot Color Based on Motion Type

If you wish to plot different colors for different motions you need to add some more Python code.

Добавьте это в начало файла:

```
from qtvcp.core import Status
STATUS = Status()
```

и это в класс Window:

```
STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

# uncomment to change feed color and to see all colors printed to the terminal
#v.setColorsAttribute('FEED',(0,1,0))
#print(v.colors)
```

You can set DEFAULT, FEED, TRAVERSE, ARC, PROBE, ROTARYINDEX, TOOLCHANGE colors with setColorsAttribute().

#### 12.9.9.8 Capture

- `tooltip = Capture()`
  Think of this as an invisible part that needs to be attached to the tooltip to track the position and orientation of the tool coordinate system. It is actually a transformation matrix that is constantly updated as the model moves.
- `work = Capture()`
  Same as above but attached to the work table to track the work coordinate system.

#### 12.9.9.9 main

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.

**`main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)`**

**`_model_`**
: Should be a collection that contains all the machine parts.

**`_tooltip_ and _work_`**
: Need to be created by `Capture()` to draw the backplot which is basically the tooltip position drawn in the work coordinate system. See `mill_xyz.py` for an example of how to connect the tool tip to a tool and the tool to the model.

**`_size_`**
: Sets the extent of the volume visualized in the initial view.

**`_hud_`**
: refers to a head-up display.

**`_rotation_vectors_ or _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 Basic structure of a `QtVismach` script

```
# imports
import hal
from qtvcp.lib.qt_vismach.qt_vismach import *

# import Status for motion type messages
from qtvcp.core import Status
STATUS = Status()

# create HAL pins here if needed
#c = hal.component("samplegui")
#c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)

# create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()

# Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],None,"joint.0.pos-fb",360,0,0,1)
part1 = Translate([dogs],-1,49,0)

# create a top-level model
model = Collection([base, saddle, head, carousel])

# we want to either embed into qtvcp or display directly with PyQt5
# so build a window to display the model

class Window(QWidget):

    def __init__(self):
        super(Window, self).__init__()
        self.glWidget = GLWidget()
        v = self.glWidget
        v.set_latitudelimits(-180, 180)
```

```
        world = Capture()

        # uncomment if there is a HUD
        # HUD needs to know where to draw
        #v.hud = myhud
        #v.hud.app = v

        # update plot color based on motion type
        STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

        # uncomment to change feed color
        #v.setColorsAttribute('FEED',(0,1,0))
        # and to see all colors printed to the terminal
        #print(v.colors)

        v.model = Collection([model, world])
        size = 600
        v.distance = size * 3
        v.near = size * 0.01
        v.far = size * 10.0
        v.tool2view = tooltip
        v.world2view = world
        v.work2view = work

        mainLayout = QHBoxLayout()
        mainLayout.addWidget(self.glWidget)
        self.setLayout(mainLayout)

# if you call this file directly from python3, it will display a PyQt5 window
# good for confirming the parts of the assembly.

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 Обзор

Building custom widgets allows one to **use the Qt Designer editor to place a custom widget** *rather than doing it manually in a handler file*.

A useful custom widgets would be a great way to contribute back to LinuxCNC.

#### 12.10.1.1 Widgets

**Widget** is the *general name for the UI objects* such as buttons and labels in PyQt.

There are also **special widgets made for LinuxCNC** that make integration easier.

All these widgets can be *placed with Qt Designer editor* - allowing one to *see the result* before actually loading the panel in LinuxCNC.

#### 12.10.1.2 Qt Designer

**Qt Designer** is a *WYSIWYG (What You See is What You Get) editor for placing PyQt widgets*.
It's original intend was for building the graphic widgets for programs.
We leverage it to **build screens and panels for LinuxCNC**.
In Qt Designer, on the left side of the editor, you find **three categories of LinuxCNC widgets**:

- *HAL only widgets*.
- *LinuxCNC controller widgets*.
- *dialog widgets*.

For Qt Designer to *add custom widgets* to it's editor it must have a **plugin** added to the right folder.

#### 12.10.1.3 Initialization Process

QtVCP does *extra setup* for **widgets subclassed from `_HALWidgetBase`**, aka "HAL-ified" widgets.
This includes:

- Injecting *important variables*,
- Calling an *extra setup function*
- Calling a *closing cleanup function* at shutdown.

These functions are not called when the Qt Designer editor displays the widgets.
When QtVCP builds a screen from the `.ui` file:

1. It searches for all the HAL-ified widgets.
2. It finds the `ScreenOptions` widget, to collect information it needs to inject into the other widgets
3. It instantiates each widget and if it is a HAL-ified widget, calls the `hal_init()` function.
   **`hal_init()`** is defined in the base class and it:
   a. Adds variables such as the preference file to every HAL-ified widget.
   b. Call `+_hal_init()+` on the widget.
      `+_hal_init()+` allows the widget designer to do setup that requires access to the extra variables.

Here is a description of the extra variables injected into "HAL-ified" widgets:

**`self.HAL_GCOMP`**
The *HAL component instance*

**`self.HAL_NAME`**
This *widget's name* as a string

**`self.QT_OBJECT_`**
This *widget's object instance*

**`self.QTVCP_INSTANCE_`**
The *very top level parent* of the screen

**`self.PATHS_`**
The *QtVCP's path library instance*

**`self.PREFS_`**
The *optional preference file instance*

**`self.SETTINGS_`**
The `Qsettings` *object instance*

#### 12.10.1.4 cleanup process

When QtVCP closes, it calls the **+_hal_cleanup()+** function *on all HAL-ified widgets*.

The base class creates an empty +_hal_cleanup()+ function, which can be redefined in the custom widget subclass.

This can be used to do such things as record preferences, etc.

This function is not called when the Qt Designer editor displays the widgets.

### 12.10.2 Custom HAL Widgets

HAL widgets are the simplest to show example of.
qtvcp/widgets/simple_widgets.py holds many HAL only widgets.

Lets look at a snippet of simple_widgets.py:

**In the Imports section** This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5 import QtWidgets                     # ❶
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase    # ❷
import hal                                      # ❸
```

In this case we need access to:

❶ PyQt's QtWidgets library,

❷ LinuxCNC's HAL library, and

❸ 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.

**In the WIDGET section** Here is a *custom widget* based on PyQt's **QGridLayout** widget.

QGridLayout allows one to:

- *Place objects in a grid* fashion.
- *Enable/disable all widgets inside it* based on a **HAL pin state**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):    # ❶
    def __init__(self, parent = None):                          # ❷
        super(GridLayout, self).__init__(parent)                # ❸
```

Line by Line:

1. This defines the *class name* and the *libraries it inherits from*.
   This class, named `Lcnc_GridLayout`, inherits the functions of `QWidget` and `+_HalSensitiveBase+`.
   `+_HalSensitiveBase+` is *subclass* of **`+_HalWidgetBase+`**, the *base class of most QtVCP widgets*, meaning it has all the functions of `+_HalWidgetBase+` plus the functions of `+_HalSensitiveBase+`.
   It adds the function to make the widget be enabled or disabled based on a HAL input BIT pin.

2. This is the function *called when the widget is first made* (said instantiated) - this is pretty standard.

3. This function initializes our widget's **`Super` classes**.
   `Super` just means the *inherited baseclasses*, that is `QWidget` and `_HalSensitiveBase`.
   Pretty standard other than the widget name will change.

### 12.10.3 Custom Controller Widgets Using `STATUS`

Widget that interact with LinuxCNC's controller are only a little more complicated and they require some *extra libraries*.

In this cut down example we will add properties that can be changed in Qt Designer.

This LED indicator widget will respond to selectable LinuxCNC controller states.

```
#!/usr/bin/env python3

##############################
# Imports
##############################
from PyQt5.QtCore import pyqtProperty
from qtvcp.widgets.led_widget import LED
from qtvcp.core import Status

###########################################
# **** instantiate libraries section **** #
###########################################
STATUS = Status()

########################################
# custom widget class definition
########################################
class StateLED(LED):
    def __init__(self, parent=None):
        super(StateLED, self).__init__(parent)
        self.has_hal_pins = False
        self.setState(False)
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False

    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))

    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

```
    ########################################################################
    # Qt Designer properties setter/getters/resetters
    ########################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False

    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

#### 12.10.3.1 In The *Imports* Section

This is where we import libraries that our widget class needs.

```
#!/usr/bin/env python3

################################
# Imports
################################
from PyQt5.QtCore import pyqtProperty       # ❶
from qtvcp.widgets.led_widget import LED    # ❷
from qtvcp.core import Status               # ❸
```

We import

❶ `pyqtProperty` so we can interact with the Qt Designer editor,

❷ `LED` because our custom widget is based on it,

❸ `Status` because it gives us status messages from LinuxCNC.

#### 12.10.3.2 In The *Instantiate Libraries* Section

Here we create the `Status` library instance:

```
##########################################
# **** instantiate libraries section **** #
##########################################
STATUS = Status()
```

Typically we instantiated the library *outside of the widget class* so that the reference to it is **global** - meaning you don't need to use `self.` in front of it.

By convention we use *all capital* letters in the name for global references.

#### 12.10.3.3 In The *Custom Widget Class Definition* Section

This is the meat and potatoes of our custom widget.

**Class definition and instance initialization function**

```
class StateLed(LED):                              # ❶
    def __init__(self, parent=None):              # ❷
        super(StateLed, self).__init__(parent)    # ❸
        self.has_hal_pins = False                 # ❹
        self.setState(False)                      # ❺
        self.is_estopped = False
        self.is_on = False
        self.invert_state = False
```

❶ Defines the **name** *of our custom widget* and what other *class it inherits from*.
In this case we inherit `LED` - a QtVCP widget that represents a status light.

❷ Typical of most widgets - called when the widget is first made.

❸ Typical of most widgets - calls the parent (super) widget initialization code.
Then we set some attributes:

❹ Inherited from `Lcnc_Led` - we set it here so no HAL pin is made.

❺ Inherited from `Lcnc_led` - we set it to make sure the LED is off.

The other attributes are for the selectable options of our widget.

**Widget's HAL initialization function**

```
    def _hal_init(self):
        if self.is_estopped:
            STATUS.connect('state-estop', lambda w:self._flip_state(True))
            STATUS.connect('state-estop-reset', lambda w:self._flip_state(False))
        elif self.is_on:
            STATUS.connect('state-on', lambda w:self._flip_state(True))
            STATUS.connect('state-off', lambda w:self._flip_state(False))
```

This function connects `STATUS` (LinuxCNC status message library) to our widget, so that the LED will on or off based on the selected state of the controller.

We have two states we can choose from `is_estopped` or `is_on`.
Depending on which is active our widget get connected to the appropriate STATUS messages.

**+_hal_init()+** is *called on each widget that inherits* +_HalWidgetBase+, when QtVCP first builds the screen.
You might wonder why it's called on this widget since we didn't have +_HalWidgetBase+ in our class definition (`class Lcnc_State_Led(Lcnc_Led):`) - it's called because `Lcnc_Led` inherits +_HalWidgetBase+.

In this function you have access to some extra information (though we don't use them in this example):

**self.HAL_GCOMP**
: the *HAL component* instance

**self.HAL_NAME**
: This *widget's name* as a string

**self.QT_OBJECT_**
: This *widget's PyQt object instance*

**self.QTVCP_INSTANCE_**
: The *very top level parent* of the screen

**self.PATHS_**
: The *instance of QtVCP's path* library

**self.PREFS_**
: the *instance of an optional preference file*

**self.SETTINGS_**
: the `Qsettings` *object*

We could use this information to create HAL pins or look up image paths etc.

```
STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Lets look at this line more closely:

- `STATUS` is very common theme is widget building.
  `STATUS` uses `GObject` message system to send messages to widgets that register to it.
  This line is the registering process.
- `state-estop` is the message we wish to listen for and act on. There are many messages available.
- `lambda w:self._flip_state(True)` is what happens when the message is caught.
  The lambda function accepts the widget instance (`w`) that GObject sends it and then calls the function `self._flip_state(True)`.
  Lambda was used to strip the (`w`) object before calling the `self._flip_state` function.
  It also allowed use to send `self._flip_state()` the True state.

```
    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

This is the function that actually flips the state of the LED.
It is what gets called when the appropriate STATUS message is accepted.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

The function called looks like this:

```
    def _set_feedrate_text(self, widget, data):
```

in which the widget and any data must be accepted by the function.

```
    ########################################################################
    # Qt Designer properties setter/getters/resetters
    ########################################################################

    # invert status
    def set_invert_state(self, data):
        self.invert_state = data
    def get_invert_state(self):
        return self.invert_state
    def reset_invert_state(self):
        self.invert_state = False

    # machine is estopped status
    def set_is_estopped(self, data):
        self.is_estopped = data
    def get_is_estopped(self):
        return self.is_estopped
    def reset_is_estopped(self):
        self.is_estopped = False

    # machine is on status
    def set_is_on(self, data):
        self.is_on = data
    def get_is_on(self):
        return self.is_on
    def reset_is_on(self):
        self.is_on = False
```

This is *how Qt Designer sets the attributes of the widget*.
This *can also be called directly* in the widget.

```
    ######################################
    # Qt Designer properties
    ######################################
    invert_state_status = pyqtProperty(bool, get_invert_state, set_invert_state,  ↩
        reset_invert_state)
    is_estopped_status = pyqtProperty(bool, get_is_estopped, set_is_estopped,  ↩
        reset_is_estopped)
    is_on_status = pyqtProperty(bool, get_is_on, set_is_on, reset_is_on)
```

This is the **registering of properties in Qt Designer**.

The **property name**:

- is the *text used in Qt Designer,*
- *cannot be the same as the attributes* they represent.

These properties show in Qt Designer in the order they appear here.

### 12.10.4 Custom Controller Widgets with Actions

Here is an example of a widget that sets the user reference system.

It changes:

- the machine controller state using the `ACTION` library,
- whether the button can be clicked or not using the `STATUS` library.

```
import os
import hal

from PyQt5.QtWidgets import QWidget, QToolButton, QMenu, QAction
from PyQt5.QtCore import Qt, QEvent, pyqtProperty, QBasicTimer, pyqtSignal
from PyQt5.QtGui import QIcon

from qtvcp.widgets.widget_baseclass import _HalWidgetBase
from qtvcp.widgets.dialog_widget import EntryDialog
from qtvcp.core import Status, Action, Info

# Instantiate the libraries with global reference
# STATUS gives us status messages from LinuxCNC
# INFO holds INI details
# ACTION gives commands to LinuxCNC
STATUS = Status()
INFO = Info()
ACTION = Action()

class SystemToolButton(QToolButton, _HalWidgetBase):
    def __init__(self, parent=None):
        super(SystemToolButton, self).__init__(parent)
        self._joint = 0
        self._last = 0
        self._block_signal = False
        self._auto_label_flag = True
        SettingMenu = QMenu()
        for system in('G54', 'G55', 'G56', 'G57', 'G58', 'G59', 'G59.1', 'G59.2', 'G59.3'):

            Button = QAction(QIcon('exit24.png'), system, self)
            Button.triggered.connect(self[system.replace('.','_')])
            SettingMenu.addAction(Button)

        self.setMenu(SettingMenu)
        self.dialog = EntryDialog()

    def _hal_init(self):
        if not self.text() == '':
            self._auto_label_flag = False
        def homed_on_test():
            return (STATUS.machine_is_on()
                    and (STATUS.is_all_homed() or INFO.NO_HOME_REQUIRED))

        STATUS.connect('state-off', lambda w: self.setEnabled(False))
        STATUS.connect('state-estop', lambda w: self.setEnabled(False))
        STATUS.connect('interp-idle', lambda w: self.setEnabled(homed_on_test()))
        STATUS.connect('interp-run', lambda w: self.setEnabled(False))
        STATUS.connect('all-homed', lambda w: self.setEnabled(True))
        STATUS.connect('not-all-homed', lambda w, data: self.setEnabled(False))
        STATUS.connect('interp-paused', lambda w: self.setEnabled(True))
        STATUS.connect('user-system-changed', self._set_user_system_text)

    def G54(self):
        ACTION.SET_USER_SYSTEM('54')

    def G55(self):
        ACTION.SET_USER_SYSTEM('55')

    def G56(self):
        ACTION.SET_USER_SYSTEM('56')
```

```
    def G57(self):
        ACTION.SET_USER_SYSTEM('57')

    def G58(self):
        ACTION.SET_USER_SYSTEM('58')

    def G59(self):
        ACTION.SET_USER_SYSTEM('59')

    def G59_1(self):
        ACTION.SET_USER_SYSTEM('59.1')

    def G59_2(self):
        ACTION.SET_USER_SYSTEM('59.2')

    def G59_3(self):
        ACTION.SET_USER_SYSTEM('59.3')

    def _set_user_system_text(self, w, data):
        convert = { 1:"G54", 2:"G55", 3:"G56", 4:"G57", 5:"G58", 6:"G59", 7:"G59.1", 8:"G59 ↩
            .2", 9:"G59.3"}
        if self._auto_label_flag:
            self.setText(convert[int(data)])

    def ChangeState(self, joint):
        if int(joint) != self._joint:
            self._block_signal = True
            self.setChecked(False)
            self._block_signal = False
            self.hal_pin.set(False)

    ##############################
    # required class boiler code #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

### 12.10.5 Stylesheet Property Changes Based On Events

It's possible to **have widgets restyled when events change**. You must explicitly *"polish" the widget* to have PyQt redo the style.
This is a relatively expensive function so should be used sparingly.

This example sets an `isHomed` property based on LinuxCNC's homed state and in turn uses it to change stylesheet properties:

**This example will set the property isHomed based on LinuxCNC's homed state.**

```
class HomeLabel(QLabel, _HalWidgetBase):
    def __init__(self, parent=None):
        super(HomeLabel, self).__init__(parent)
        self.joint_number = 0
        # for stylesheet reading
        self._isHomed = False

    def _hal_init(self):
        super(HomeLabel, self)._hal_init()
```

```
        STATUS.connect('homed', lambda w,d: self._home_status_polish(int(d), True))
        STATUS.connect('unhomed', lambda w,d: self._home_status_polish(int(d), False))

    # update ishomed property
    # polish widget so stylesheet sees the property change
    # some stylesheets color the text on home/unhome
    def _home_status_polish(self, d, state):
        if self.joint_number = d:
            self.setProperty('isHomed', state)
            self.style().unpolish(self)
            self.style().polish(self)

    # Qproperty getter and setter
    def getisHomed(self):
        return self._isHomed
    def setisHomed(self, data):
        self._isHomed = data

    # Qproperty
    isHomed = QtCore.pyqtProperty(bool, getisHomed, setisHomed)
```

Here is a sample stylesheet to change text color based on home state.

In this case any widget based on the HomeLabel widget above will change text color.
You would usually pick specific widgets using `HomeLabel #specific_widget_name[homed=true]`:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

### 12.10.6 Use Stylesheets To Change Custom Widget Properties

```
class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        alternateFont0 = self.font

    # Qproperty getter and setter
    def getFont0(self):
        return self.aleternateFont0
    def setFont0(self, value):
        self.alternateFont0(value)
    # Qproperty
    styleFont0 = pyqtProperty(QFont, getFont0, setFont0)
```

Sample stylesheet that sets a custom widget property.

```
Label{
    qproperty-styleFont0: "Times,12,-1,0,90,0,0,0,0,0";
}
```

### 12.10.7 Widget Plugins

We must *register our custom widget* for Qt Designer to use them.

Here are a typical samples.
They would need to be added to `qtvcp/plugins/`
Then `qtvcp/plugins/qtvcp_plugin.py` would need to be adjusted to *import* them.

#### 12.10.7.1 Gridlayout Example

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.simple_widgets import Lcnc_GridLayout
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

######################################
# GridLayout
######################################
class LcncGridLayoutPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        QPyDesignerCustomWidgetPlugin.__init__(self)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return Lcnc_GridLayout(parent)
    def name(self):
        return "Lcnc_GridLayout"
    def group(self):
        return "LinuxCNC - HAL"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('lcnc_gridlayout')))
    def toolTip(self):
        return "HAL enable/disable GridLayout widget"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return True
    def domXml(self):
        return '<widget class="Lcnc_GridLayout" name="lcnc_gridlayout" />\n'
    def includeFile(self):
        return "qtvcp.widgets.simple_widgets"
```

#### 12.10.7.2 SystemToolbutton Example

```
#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin
from qtvcp.widgets.system_tool_button import SystemToolButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

######################################
# SystemToolButton
```

```
###################################
class SystemToolButtonPlugin(QPyDesignerCustomWidgetPlugin):
    def __init__(self, parent = None):
        super(SystemToolButtonPlugin, self).__init__(parent)
        self.initialized = False
    def initialize(self, formEditor):
        if self.initialized:
            return
        self.initialized = True
    def isInitialized(self):
        return self.initialized
    def createWidget(self, parent):
        return SystemToolButton(parent)
    def name(self):
        return "SystemToolButton"
    def group(self):
        return "LinuxCNC - Controller"
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('systemtoolbutton')))
    def toolTip(self):
        return "Button for selecting a User Coordinate System"
    def whatsThis(self):
        return ""
    def isContainer(self):
        return False
    def domXml(self):
        return '<widget class="SystemToolButton" name="systemtoolbutton" />\n'
    def includeFile(self):
        return "qtvcp.widgets.system_tool_button"
```

#### 12.10.7.3 Making a plugin with a MenuEntry dialog box

It possible to add an entry to the dialog that pops up when you right click the widget in the layout.

This can do things such as selecting options in a more convenient way.

This is the plugin used for *action buttons*.

```
#!/usr/bin/env python3

import sip
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtDesigner import QPyDesignerCustomWidgetPlugin, \
                QPyDesignerTaskMenuExtension, QExtensionFactory, \
                QDesignerFormWindowInterface, QPyDesignerMemberSheetExtension
from qtvcp.widgets.action_button import ActionButton
from qtvcp.widgets.qtvcp_icons import Icon
ICON = Icon()

Q_TYPEID = {
    'QDesignerContainerExtension':     'org.qt-project.Qt.Designer.Container',
    'QDesignerPropertySheetExtension': 'org.qt-project.Qt.Designer.PropertySheet',
    'QDesignerTaskMenuExtension': 'org.qt-project.Qt.Designer.TaskMenu',
    'QDesignerMemberSheetExtension': 'org.qt-project.Qt.Designer.MemberSheet'
}

###################################
# ActionBUTTON
###################################
class ActionButtonPlugin(QPyDesignerCustomWidgetPlugin):
```

```
    # The __init__() method is only used to set up the plugin and define its
    # initialized variable.
    def __init__(self, parent=None):
        super(ActionButtonPlugin, self).__init__(parent)
        self.initialized = False

    # The initialize() and isInitialized() methods allow the plugin to set up
    # any required resources, ensuring that this can only happen once for each
    # plugin.
    def initialize(self, formEditor):

        if self.initialized:
            return
        manager = formEditor.extensionManager()
        if manager:
            self.factory = ActionButtonTaskMenuFactory(manager)
            manager.registerExtensions(self.factory, Q_TYPEID['QDesignerTaskMenuExtension' ↩
                ])
        self.initialized = True

    def isInitialized(self):
        return self.initialized

    # This factory method creates new instances of our custom widget
    def createWidget(self, parent):
        return ActionButton(parent)

    # This method returns the name of the custom widget class
    def name(self):
        return "ActionButton"

    # Returns the name of the group in Qt Designer's widget box
    def group(self):
        return "LinuxCNC - Controller"

    # Returns the icon
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path('actionbutton')))

    # Returns a tool tip short description
    def toolTip(self):
        return "Action button widget"

    # Returns a short description of the custom widget for use in a "What's
    # This?" help message for the widget.
    def whatsThis(self):
        return ""

    # Returns True if the custom widget acts as a container for other widgets;
    def isContainer(self):
        return False

    # Returns an XML description of a custom widget instance that describes
    # default values for its properties.
    def domXml(self):
        return '<widget class="ActionButton" name="actionbutton" />\n'

    # Returns the module containing the custom widget class. It may include
    # a module path.
    def includeFile(self):
        return "qtvcp.widgets.action_button"
```

```
class ActionButtonDialog(QtWidgets.QDialog):

   def __init__(self, widget, parent = None):

      QtWidgets.QDialog.__init__(self, parent)

      self.widget = widget

      self.previewWidget = ActionButton()

      buttonBox = QtWidgets.QDialogButtonBox()
      okButton = buttonBox.addButton(buttonBox.Ok)
      cancelButton = buttonBox.addButton(buttonBox.Cancel)

      okButton.clicked.connect(self.updateWidget)
      cancelButton.clicked.connect(self.reject)

      layout = QtWidgets.QGridLayout()
      self.c_estop = QtWidgets.QCheckBox("Estop Action")
      self.c_estop.setChecked(widget.estop )
      layout.addWidget(self.c_estop)

      layout.addWidget(buttonBox, 5, 0, 1, 2)
      self.setLayout(layout)

      self.setWindowTitle(self.tr("Set Options"))

   def updateWidget(self):

      formWindow = QDesignerFormWindowInterface.findFormWindow(self.widget)
      if formWindow:
          formWindow.cursor().setProperty("estop_action",
              QtCore.QVariant(self.c_estop.isChecked()))
      self.accept()

class ActionButtonMenuEntry(QPyDesignerTaskMenuExtension):

    def __init__(self, widget, parent):
        super(QPyDesignerTaskMenuExtension, self).__init__(parent)
        self.widget = widget
        self.editStateAction = QtWidgets.QAction(
          self.tr("Set Options..."), self)
        self.editStateAction.triggered.connect(self.updateOptions)

    def preferredEditAction(self):
        return self.editStateAction

    def taskActions(self):
        return [self.editStateAction]

    def updateOptions(self):
        dialog = ActionButtonDialog(self.widget)
        dialog.exec_()

class ActionButtonTaskMenuFactory(QExtensionFactory):
    def __init__(self, parent = None):
        QExtensionFactory.__init__(self, parent)

    def createExtension(self, obj, iid, parent):

        if not isinstance(obj, ActionButton):
```

```
            return None
        if iid == Q_TYPEID['QDesignerTaskMenuExtension']:
            return ActionButtonMenuEntry(obj, parent)
        elif iid == Q_TYPEID['QDesignerMemberSheetExtension']:
            return ActionButtonMemberSheet(obj, parent)
        return None
```

## 12.11 QtVCP Handler File Code Snippets

### 12.11.1 Preference File Loading/Saving

Here is how to **load and save preferences at launch and closing time**.

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.

---

**Note**
The file is actually saved in `~/.config/QtVcp`

---

**Пример** In this example:

- 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:

```
# set recorded columns sort settings
self.SETTINGS_.beginGroup("MyGroupName")
self.int_value = self.SETTINGS_.value('myInteger', type = 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:

```
# save values with QSettings
self.SETTINGS_.beginGroup("MyGroupName")
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')
```

**Create the key bound function in the KEYBINDING SECTION:**

```
def on_keycall_F12(self,event,state,shift,cntrl):
    if state:
        STYLEEDITOR.load_dialog()
```

### 12.11.4 Request Dialog Entry

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 Speak a Startup Greeting

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** Under the `init.(self, halcomp, widgets, paths)` function:

```
STATUS.emit('play-alert','SPEAK Please remember to oil the ways.')
```

**`SPEAK`** is a keyword: *everything after it will be pronounced*.

### 12.11.6 ToolBar Functions

Toolbar buttons and submenus are added in Qt Designer but the code to make them do something is added in the handler file. To **add a submenus** in Qt Designer:

- Add a `Qaction` by typing in the toolbar column then clicking the + icon on the right.
- This will add a sub column that you need to type a name into.
- Now the original `Qaction` will be a `Qmenu` instead.
- Now erase the `Qaction` you added to that `Qmenu`, the menu will stay as a menu.

In this example we assume you added a toolbar with one submenu and three actions. These actions will be configured to create:

- a recent file selection menu,
- an about pop up dialog action,
- a quit program action, and
- a user defined function action.

The `objectName` of the toolbar button is used to identify the button when configuring it - *descriptive names help*.

Using the action editor menu, right click and select edit.
Edit the object name, text, and button type for an appropriate action.

In this example the:

- submenu name must be `menuRecent`,
- actions names must be `actionAbout`, `actionQuit`, `actionMyFunction`

**Loads the toolbar_actions library in the IMPORT SECTION**

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

**Instantiate ToolBarActions module in the INSTANTIATE LIBRARY SECTION**

```
TOOLBAR = ToolBarActions()
```

**Configure submenus and actions in the SPECIAL FUNCTIONS SECTION** Under the `def initialized_` function add:

```
TOOLBAR.configure_submenu(self.w.menuRecent, 'recent_submenu')
TOOLBAR.configure_action(self.w.actionAbout, 'about')
TOOLBAR.configure_action(self.w.actionQuit, 'Quit', lambda d:self.w.close())
TOOLBAR.configure_action(self.w.actionMyFunction, 'My Function', self.my_function)
```

**Define the user function in the GENERAL FUNCTIONS SECTION**

```
def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

The function to be called if the action "My Function" button is pressed.

### 12.11.7 Add HAL Pins That Call Functions

In this way you *don't need to poll the state of input pins*.

**Loads the Qhal library in the IMPORT SECTION**

```
from qtvcp.core import Qhal
```

This is to allow access to **QtVCP's HAL component**. For more info: Qhal
Qhal's newPin function returns a QPin object. For more info: QPin

**Instantiate Qhal in the INSTANTIATE LIBRARY SECTION**

```
QHAL = Qhal()
```

**Add a function that gets called when the pin state changes** Under the `initialized__` function, make sure there is an entry similar to this:

```
########################################
# Special Functions called from QtVCP
########################################

# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    self.pin_cycle_start_in = QHAL.newPin('cycle-start-in',QHAL.HAL_BIT, QHAL.HAL_IN)
    self.pin_cycle_start_in.pinValueChanged.connect(lambda o,s: self.cycleStart(s))
```

**Define the function called by pin state change in the GENERAL FUNCTIONS SECTION**

```
####################
# 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()
```

This function assumes there is a Tab widget, named `mainTab`, that has tabs with the names `tab_auto`, `tab_graphics`, `tab_filemanager` and `tab_mdi`.

In this way the cycle start button works differently depending on what tab is shown.

This is simplified - *checking state and error trapping might be helpful*.

### 12.11.8 Read/Write System HAL Pins Directly

Sometimes you need to read a system pin and creating a HAL pin and connecting to it, is more work then required. You can read it directly without connecting to it.

Here is how to read a pin, parameter, or signal:

```
self.h.hal.get_value('spindle.0.at-speed')

# or use the Qhal library like this:
QHAL.getValue('spindle.0.at-speed')
```

Here is how to write to an unconnected pin or a non driven signal:

```
self.h.hal.set_p('componentName.pinName','10')
self.h.hal.set_s('componentName.signalName','10')

# or use the Qhal library like this:
QHAL.setPin('componentName.pinName', '10')
QHAL.setSignal('componentName.signalName', '10')
```

Using *self.h.hal* or *QHAL.hal* allows access to the HAL python module functions: Python HAL Interface

### 12.11.9 Add A Special Max Velocity Slider Based On Percent

Some times you want to **build a widget to do something not built in**. The built in Max velocity slider acts on units per minute, here we show how to do on percent.

The **STATUS** command makes sure the slider adjusts if LinuxCNC changes the current max velocity.

**valueChanged.connect()** *calls a function when the slider is moved*.

In Qt Designer add a **QSlider** widget called `mvPercent`, then add the following code to the handler file:

```
############################
# 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))
```

### 12.11.10 Toggle Continuous Jog On and Off

Generally selecting continuous jogging is a momentary button, that requires you to select the previous jog increment after.

We will build a button that toggles between continuous jog and whatever increment that was already selected.

In Qt Designer:

- Add an `ActionButton` with no action
- Call it `btn_toggle_continuous`.
- Set the `AbstractButton` property `checkable` to `True`.
- Set the `ActionButton` properties `incr_imperial_number` and `incr_mm_number` to `0`.

- Use Qt Designer's slot editor to use the button signal `clicked(bool)` to call form's handler function `toggle_continuous_clicked()`.
  See Using Qt Designer To Add Slots section for more information.

Then add this code snippets to the handler file under the `initialized__` function:

```
# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    STATUS.connect('jogincrement-changed', \
        lambda w, d, t: self.record_jog_incr(d,t) \
        )
    # set a default increment to toggle back to
    self.L_incr = 0.01
    self.L_text = "0.01in"
```

In the `GENERAL FUNCTIONS SECTION` add:

```
######################
# GENERAL FUNCTIONS #
######################

# if it isn't continuous, record the latest jog increment
# and untoggle the continuous button
def record_jog_incr(self,d, t):
    if d != 0:
        self.L_incr = d
        self.L_text = t
        self.w.btn_toggle_continuous.safecheck(False)
```

In the `CALLBACKS FROM FORM SECTION` add:

```
########################
# CALLBACKS FROM FORM #
########################

def toggle_continuous_clicked(self, state):
    if state:
        # set continuous (call the actionbutton's function)
        self.w.btn_toggle_continuous.incr_action()
    else:
        # reset previously recorded increment
        ACTION.SET_JOG_INCR(self.L_incr, self.L_text)
```

### 12.11.11 Class Patch The File Manager Widget

**Note**
Class patching (monkey patching) is a little like *black magic* - so use it *only when needed*. The Major problem is if the widget library functions are changed during development, the functions may break.
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*.
You can do this class patch inside or outside the HandlerClass instance.
This will change what *self* represents in the function.

Outside the HanderClass, *self* will be the patched class instance.
Inside the HanderClass, *self* will be the HandlerClass instance.
This would change what functions/variables you can access in the function.
Here we show an inside the HandlerClass example:
In the `IMPORT SECTION` add:

```
from qtvcp.widgets.file_manager import FileManager as FM
```

Here we are going to:

1. *Keep a reference to the original function* (1) so we can still call it
2. *Redirect the class to call our custom function* (2) in the handler file instead.

```
########################################
# Special Functions called from QtVCP    #
########################################

# For changing functions in widgets we can 'class patch'.
# class patching must be done before the class is instantiated.
def class_patch__(self):
    self.old_load = FM.load # keep a reference of the old function (1)
    FM.load = self.our_load # redirect function to our handle file function (2)
```

3. *Write a custom function to replace the original*:
   This function must have the **same signature as the original function**.
   *self* is the HandlerClass instance *not* the patched class instance.
   In this example we are still going to call the original function by using the reference to it we recorded earlier.
   It *requires the first argument to be the widget instance*, which in this case is `self.w.filemanager` (the name given in the Qt Designer editor).

```
####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

Now our custom function will print the file path to the terminal before loading the file. Obviously boring but shows the principle.

**Note**
There is another slightly different way to do this that can have advantages: you can *store the reference to the original function in the original class*.
The trick here is to make sure the function name you use to store it is not already used in the class. `super__` added to the function name would be a good choice.
We won't use that in built in QtVCP widgets.

```
########################################
# 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):
    FM.super__load = FM.load # keep a reference of the old function in the original class
    FM.load = self.our_load # redirect function to our handle file function

####################
# GENERAL FUNCTIONS #
####################

def our_load(self,fname):
    print(fname)
    self.w.filemanager.super__load(fname)
```

### 12.11.12 Adding Widgets Programmatically

In some situation it is only possible to **add widgets with Python code** rather then using the Qt Designer editor.

When adding QtVCP widgets programmatically, sometimes there are *extra steps* to be taken.

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.

This is a cut down example from QtAxis screen's handler file.

**Import required libraries** First we must import the libraries we need, if they're not already imported in the handler file:

- `QtWidgets` gives us access to the `QProgressBar`,
- `QColor` is for the *LED color*,
- `StateLED` is the QtVCP library used to *create the spindle-at-speed LED*,
- `Status` is used to *catch LinuxCNC status information*,
- `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
```

**Instantiate Status and Info channels** `STATUS` and `INFO` are initialized outside the handler class so as to be *global references* (no self. in front):

```
########################################
# **** instantiate libraries section **** #
########################################

STATUS = Status()
INFO = Info()
```

**Register STATUS monitoring function** 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*
- *Call* the `self.update_spindle()` *function*

```
########################
# **** 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))
```

**Add the widgets to the tab** 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*.

We are assuming there is a tab widget built with Qt Designer called `rightTab`.

```
####################
# general functions #
####################

def make_corner_widgets(self):
    # make a spindle-at-speed green LED
    self.w.led = LED()                                          # ❶
    self.w.led.setProperty('is_spindle_at_speed_status',True) # ❷
    self.w.led.setProperty('color',QColor(0,255,0,255))        # ❸
    self.w.led.hal_init(HAL_NAME = 'spindle_is_at_speed')      # ❹
```

```
    # 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)

    # add the container to the corner of the right tab widget
    self.w.rightTab.setCornerWidget(w)                           # 11
```

1, 1 This initializes the basic StateLed widget and uses `self.w.led` as the reference from then on.

2, 2 Since the state LED can be used for many indications, we must set the property that designates it as a spindle-at-speed LED.

3 This sets it as green when on.

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_`**
: This *widget's PyQt object instance*

**`self.QTVCP_INSTANCE_`**
: The *very top level parent* of the screen

**`self.PATHS_`**
: The *instance of QtVCP's path* library

**`self.PREFS_`**
: the *instance of an optional preference file*

**`self.SETTINGS_`**
: the `Qsettings` *object*

5 Initializes a PyQt5 `QProgressBar`.

6 Sets the max range of the progress bar to the max specified in the `INI`.

7 We create a QWidget
Since you can only add one widget to the tab corner and we want two there, we must add both into a **container**.

8 add a QHBoxLayout to the QWidget.

9, 10 Then we add our QProgress bar and LED to the layout.

11 Finally we add the QWidget (with our QProgress bar and LED in it) to the tab widget's corner.

**Create the STATUS monitoring function** Now we build the function to actually update out the `QProgressBar` when `STATUS` updates the spindle speed:

```
########################
# callbacks from STATUS #
########################
def update_spindle(self, data):
    self.w.rpm_bar.setInvertedAppearance(bool(data<0))       # 1
    self.w.rpm_bar.setFormat('{0:d} RPM'.format(int(data)))  # 2
    self.w.rpm_bar.setValue(abs(data))                       # 3
```

1 In this case we chose to display left-to-right or right-to-left, depending if we are turning clockwise or anticlockwise.

2 This formats the writing in the bar.

3 This sets the length of the colored bar.

### 12.11.13 Update/Read Objects Periodically

Sometimes you need to **update a widget or read a value regularly** that isn't covered by normal libraries.

Here we update an LED based on a watched HAL pin every 100 ms.

We assume there is an LED named `led` in the Qt Designer UI file.

**Load the Qhal library for access to QtVCP's HAL component** In the `IMPORT SECTION` add:

```
from qtvcp.core import Qhal
```

**Instantiate Qhal** In the `INSTANTIATE LIBRARY SECTION` add:

```
QHAL = Qhal()
```

Now add/modify these sections to include code that is similar to this:

**Register a function to be called at CYCLE_TIME period** 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())
```

**Create the custom function to be called periodically**

```
####################
# general functions #
####################
def update_periodic(self):
    data = QHAL.getvalue('spindle.0.is-oriented')
    self.w.led.setState(data)
```

### 12.11.14 External Control With ZMQ

*QtVCP can automatically set up* **ZMQ messaging** *to send and/or receive remote messages from external programs.*

It uses ZMQ's **publish/subscribe messaging pattern**.

As always, consider **security** before letting programs interface though messaging.

#### 12.11.14.1 ZMQ Messages Reading

Sometimes you want to **control the screen with a separate program**.

**Enable reception of ZMQ messages** In the `ScreenOptions` widget, you can select the property **`use_receive_zmq_option`**.
You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

```
########################
# **** INITIALIZE **** #
########################
# widgets allows access to widgets from the QtVCP files
# at this point the widgets and hal pins are not instantiated
def __init__(self,halcomp,widgets,paths):
    # directly select ZMQ message receiving
    self.w.screen_options.setProperty('use_receive_zmq_option',True)
```

This **allows an external program to call functions in the handler file**.

**Add a function to be called on ZMQ message reception** 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)
```

**Create an external program sending ZMQ messages that will trigger function call** Here is a sample external program to call a function. It alternates between two data sets every second. Run this in a separate terminal from LinuxCNC to see the sent messages.

```
#!/usr/bin/env python3
from time import sleep

import zmq
import json

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://127.0.0.1:5690")
topic = b'QtVCP'

# prebuilt message 1
# makes a dict of function to call plus any arguments
x = {                                   # ❶
  "FUNCTION": "test_zmq_function",
  "ARGS": [True,200]
}
# convert to JSON object
m1 = json.dumps(x)
```

```
# prebuild message 2
x = {                                   # ❷
  "FUNCTION": "test_zmq_function",
  "ARGS": [False,0],
}
# convert to JSON object
m2 = json.dumps(x)

if __name__ == '__main__':
    while True:
        print('send message 1')
        socket.send_multipart([topic, bytes((m1).encode('utf-8'))])
        sleep(ms(1000))

        print('send message 2')
        socket.send_multipart([topic, bytes((m2).encode('utf-8'))])
        sleep(ms(1000))
```

❶, ❷ Set the **function to call** and the **arguments to send** to that function.

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.

#### 12.11.14.2 ZMQ Messages Writing

You may also want to **communicate with an external program from the screen**.

In the `ScreenOptions` widget, you can select the property **`use_send_zmq_message`**. You can also set this property directly *in the handler file*, as in this sample.

We assume the `ScreenOptions` widget is called `screen_options` in Qt Designer:

**Enable sending of ZMQ messages**

```
########################
# **** INITIALIZE **** #
########################
# 'widgets' 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 sending
    self.w.screen_options.setProperty('use_send_zmq_option',True)
```

This allows sending messages to a separate program.
The message sent will depend on what the external program is expecting.

**Create a function to send ZMQ messages** Let's add a specific function for testing.
You will need to run LinuxCNC from a terminal to see the printed text.
Also, something needs to be added to call this function, such as a button click.

```
####################
# general functions #
####################
def send_zmq_message(self):
    # This could be any Python object JSON can convert
    message = {"name": "John", "age": 30}
    self.w.screen_options.send_zmq_message(message)
```

**Use or create a program that will receive ZMQ messages** Here is a sample program that will receive the message and print it to the terminal:

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.SUB)

# Define subscription and messages with topic to accept.
topic = "" # all topics
sock.setsockopt_string(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

while True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

### 12.11.15 Sending Messages To Status Bar Or Desktop Notify Dialogs

There are several ways to **report information to the user**.

A **status bar** is used for *short information* to show the user.

---

**Note**
Not all screens have a status bar.

---

#### Status bar usage example

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` is in seconds and we assume `statusbar` is the Qt Designer set name of the widget.

You can also use the `Status` library to send a message to the `notify` library if it is enabled (usually set in `ScreenOptions` widget): This will send the message to the statusbar and the **desktop notify dialog**.

The messages are also recorded until the user erases them using controls. The users can recall any recorded messages.

There are several options:

**STATUS.TEMPORARY_MESSAGE**
Show the message for a short time only.

**STATUS.OPERATOR_ERROR , STATUS.OPERATOR_TEXT , STATUS.NML_ERROR , STATUS.NML_TEXT**

#### Example of sending an operator message:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

You can send messages thru LinuxCNC's operator message functions. These are usually caught by the notify system, so are equal to above. They would be printed to the terminal as well.

```
ACTION.SET_DISPLAY_MESSAGE('MESSAGE')
ACTION.SET_ERROR_MESSAGE('MESSAGE')
```

### 12.11.16 Catch Focus Changes

Focus is used to **direct user action** such as keyboard entry to the proper widget.

**Get currently focused widget**

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

**Get focused widget when focus changes**

```
# at this point:
# the widgets are instantiated.
# the HAL pins are built but HAL is not set ready
def initialized__(self):
    QtWidgets.QApplication.instance().event_filter.focusIn.connect(self.focusInChanged)

####################
# general functions #
####################

def focusInChanged(self, widget):
    if isinstance(widget.parent(),type(self.w.gcode_editor.editor)):
        print('G-code Editor')
    elif isinstance(widget,type(self.w.gcodegraphics)):
        print('G-code Display')
    elif isinstance(widget.parent(),type(self.w.mdihistory) ):
        print('MDI History')
```

Notice we sometimes compare to `widget`, sometimes to `widget.parent()`.

This is because *some QtVCP widgets are built from multiple **sub-widgets*** and the latter actually get the focus; so we need to **check the parent** of those sub-widgets.

Other times the main widget is what gets the focus, 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.

### 12.11.17 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
```

### 12.11.18 G-code to read Qt preferences

Here is how to create an O-word program to read a QtDragon preference file entry and add it as a G-code parameter.
Calling this O-word will update the param *toolToLoad*
This uses *Python hot comment* to communicate with the embedded python instance.
See the Remap section of the Documents for a description.

```
(filename myofile.ngc)
o<myofile> sub

;py,from interpreter import *
;py,import os
;py,from qtvcp.lib.preferences import Access

; find and print the preference file path
;py,CONFPATH = os.environ.get('CONFIG_DIR', '/dev/null')
; adjust for your preference file name
;py,PREFFILE = os.path.join(CONFPATH,'qtdragon.pref')
;py,print(PREFFILE)

; get an preference instance
;py,Pref = Access(PREFFILE)

; load a preference and print it
;py,this.params['toolToLoad']=Pref.getpref('Tool to load', 0, int,'CUSTOM_FORM_ENTRIES')
;py,print('Tool to load->:',this.params['toolToLoad'])
```

```
; return the value
o<myofile> endsub [#<toolToLoad>]
M2
```

## 12.12 QtVCP Development

### 12.12.1 Обзор

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*,
- the *handler file*, and
- possibly the `.qss` *theme file*.

### 12.12.3 QtVCP Startup To Shutdown

**QtVCP source** is located in `+src/emc/usr_intf/qtvcp+` folder of LinuxCNC source tree.

#### 12.12.3.1 QtVCP Startup

When QtVCP first starts:

1. It must decide if this object is a screen or a panel.
2. It searches for and collects information about paths of required files and useful folders.
3. It then:
    a. Builds the HAL component,
    b. Loads the window instance,
    c. Adds handler extensions,
    d. Installs an event filter.

Now the window/widgets are instantiated, the HAL pins are built. This also initiates the `+_init_hal()+` function of the widgets. . The `+initialized__()+` handler function is called . The `STATUS` library is forced to update. . HAL component is set ready at this point. . A variety of optional switch arguments are set, including calling a `POSTGUI` HAL file (if a screen). . Terminate signals are trapped and QtVCP now polls for events.

#### 12.12.3.2 QtVCP Shutdown

Finally when QtVCP is asked to shutdown:

1. It calls shutdown functions in the handler file,
2. `STATUS` monitoring is shut down
3. HAL component gets killed

### 12.12.4 Path Information

When QtVCP loads it collects paths 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**
Path of handler file

**XML**
Path of .ui file

**DOMAIN**
Путь перевода

**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**.

When read, it is ***consumed***, i.e. *no other object can read it*.

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

**Warning**
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_`**
: This *widget's PyQt object instance*

**`self.QTVCP_INSTANCE_`**
: The *very top level parent* of the screen

**`self.PATHS_`**
: The *instance of QtVCP's path* library

**`self.PREFS_`**
: The *instance of an optional preference file*

**`self.SETTINGS_`**
: The `Qsettings` *object*

When making a custom widget, _import and sub class _the **`+_HalWidgetBase+`** class for this behavior.

#### 12.12.5.6 Dialogs

Dialogs (AKA "pop up windows") are *best loaded with the* `ScreenOptions` *widget*, but they can be placed on the screen in Qt Designer.

It doesn't matter where on the layout but *to make them hidden*, cycle the `state` property to true then false.

By default, if there is a preference file, the dialogs will remember their last size/placement.
It is possible to override this so they open in the same location each time.

#### 12.12.5.7 Styles (Themes)

While it is possible to set styles *in Qt Designer*, it is more convenient to change them later if they are all set in a ***separate `.qss` file***. The file should be put in the *same location as the handler file*.

# Chapter 13

# Программирование интерфейса польз

## 13.1 Panelui

### 13.1.1 Введение

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.

While actual input buttons are required to be momentary, Panelui will use this input to make toggle, radio, or momentary button output.

### 13.1.2 Loading Commands

The command used to load panelui (with optional -d debug switch):

```
loadusr -W panelui -d
```

This will initialize panelui, which will look for the INI file panelui.ini in the config folder or user folder.

One can validate the INI file with this command:

```
loadusr pyui
```

This will read, try to correct, then save the panelui.ini file. It will print errors to the terminal if found.

A typical HAL file will have these commands added:

```
# commands needed for panelui loading
#
# sampler is needed for panelui
# cfg= must always be u for panelui. depth sets the available buffer
loadrt sampler cfg=u depth=1025
```

```
#uncomment to validate the panelui INI file
#loadusr pyui

# -d = debug, -v = verbose debug
# -d will show you keypress identification and commands called
# -v is for develeper info
loadusr -W panelui -d

# using simulated buttons instead of the MESA 7I73 card
# so we load the sim_matrix_kb component to convert HAL pins to keyscan codes
loadrt sim_matrix_kb

# connect the components together.
# sampler talks to panelui internally
net key-scan    sim-matrix-kb.0.out
net key-scan    sampler.0.pin.0

# add panelui components to a thread

addf sim-matrix-kb.0    servo-thread
addf sampler.0          servo-thread
```

### 13.1.3 panelui.ini file reference

**Key words**

- 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= This sets the Button's output type, eg S32, U32, FLOAT, BIT, NONE, COMMAND, ZMQ.
- DEFAULT= This sets the starting output of the group or button.
- GROUP= In radiobuttons, designates the group the button interacts with.
- GROUP_OUTPUT= sets the output the group pin will be, if this button is active.
- STATUS_PIN= If TRUE, a HAL pin will be added that reflects the current state of button.
- TRUE_STATE= sets the output the HAL pin will be, if button is TRUE.
- FALSE_STATE= sets the OUTPUT the HAL pin will be, if the button is FALSE.
- TRUE_COMMAND= sets the command and arguments to be called when the button is TRUE.
- FALSE_COMMAND= sets the command and arguments to be called when the button is FALSE.
- TRUE_FUNCTION= sets the ZMQ message function and arguments to be called when the button is TRUE.
- FALSE_FUNCTION= sets the ZMQ message function and arguments to be called when the button is FALSE.

**HAL Prefix**

```
[HAL_PREFIX]
    NAME= Yourname
```

This allows one to change the prefix of the HAL pins from *panelui* to an arbitrary name.

**ZMQ Messaging Setup**

```
[ZMQ_SETUP]
  TOPIC = 'QTVCP'
  SOCKET = 'tcp://127.0.0.1:5690'
  ENABLE = True
```

This sets up and enables ZMQ based messaging. TOPIC and SOCKET must match the listening program.

**Radio Buttons** Radiobutons allow only one button in the group to be active at a time. Each group has its own output pin, separate from each button in the group. Radio button definitions start with the text *RADIO_BUTTON* inside single brackets.

```
[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
```

**Toggle Buttons** Togglebuttons only change state on each press of the button. Toggle button definitions start with the text *TOGGLE_BUTTON* inside single brackets.

```
[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** Momentary buttons are true when pressed and false when released. Momentary button definitions start with the text *MOMENTARY_BUTTON* inside single brackets.

```
[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 Internal Command reference

There are a number of internal commands you may use.

**home_selected**

- required argument: axis number (int)

**unhome_selected**

- required argument: axis number (int)

**spindle_forward_adjust**

- optional argument: starting RPM (int) - default 100
- Description: If the spindle is stopped it will start in the forward direction. If it is already running it will increase or decrease the rpm depending on what direction the spindle is running in.

**spindle_forward**

- optional argument: starting RPM (int) - default 100

**spindle_reverse**

- optional argument: starting RPM (int) - default 100

**spindle_reverse_adjust**

- optional argument: starting RPM (int) - default 100
- Description: If the spindle is stopped it will start in the reverse direction. If it is already running it will increase or decrease the rpm depending on what direction the spindle is running in.

**spindle_faster**

- Description: increases spindle speed by 100 RPM

**spindle_slower**

- Description: decreases spindle speed by 100 RPM, until RPM is 100

**set_linear_jog_velocity**

- required argument: velocity in inches per minute (float)
- description: sets the jog velocity on axis 0,1,2,6,7,8 (X,Y,Z,U,V,W)

**set_angular_jog_velocity**

- required argument: velocity in degrees per minute (float)
- description: sets the jog velocity on axis 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)
- Description: Move Z axis to the given machine position

**feed_hold**

- required argument: state (bool 0 or 1)

**feed_override**

- required argument: rate (float)

**rapid_override**

- required argument: rate (float 0-1)

**spindle_override**

- required argument: rate (float)

**max_velocity**

- required argument: rate (float)

**optional_stop**

- required argument: state (bool 0 or 1)

**block_delete**

- required argument: state (bool 0 or 1)

**single_block**

- required argument: state (bool 0 or 1)

**smart_cycle_start**

- Description: If idle, starts G-code program, if paused runs one line.

**re_start line**

- required argument: line number (int)

**mdi_and_return**

- required argument: G-code command(s)
- Description: records the current mode, calls commands and then returns to mode.

**mdi**

- required argument: G-code command(s)
- Description: sets mode to MDI, calls commands.

### 13.1.5 ZMQ Messages

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
```

Here is a sample program that will receive the message and print it to the terminal.

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# Define the socket using the "Context"
sock = context.socket(zmq.SUB)

# Define subscription and messages with topic to accept.
topic = "" # all topics
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 File Extension

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 Python-модуль для LinuxCNC

This documentation describes the `linuxcnc` python module, which provides a Python API for talking to LinuxCNC.

### 13.2.1 Введение

User interfaces control LinuxCNC activity by sending NML messages to the LinuxCNC task controller, and monitor results by observing the LinuxCNC status structure, as well as the error reporting channel.

Programmatic access to NML is through a C++ API; however, the most important parts of the NML interface to LinuxCNC are also available to Python programs through the `linuxcnc` module.

Beyond the NML interface to the command, status and error channels, the `linuxcnc` module also contains:

- support for reading values from INI files

### 13.2.2 Usage Patterns for the LinuxCNC NML interface

The general pattern for `linuxcnc` usage is roughly like this:

- Импортируйте модуль `linuxcnc`.
- При необходимости установите соединения с каналами команд, состояний и ошибок NML.
- Poll the status channel, either periodically or as needed.
- Before sending a command, determine from status whether it is in fact OK to do so (for instance, there is no point in sending a *Run* command if task is in the ESTOP state, or the interpreter is not idle).
- Send the command by using one of the `linuxcnc` command channel methods.

To retrieve messages from the error channel, poll the error channel periodically, and process any messages retrieved.

- Poll the status channel, either periodically or as needed.
- Print any error message and explore the exception code.

The module `linuxcnc` also defines the `error` Python exception type to support error reporting.

### 13.2.3 Чтение статуса LinuxCNC с помощью модуля Python linuxcnc

Here is a Python fragment to explore the contents of the `linuxcnc.stat` object which contains some 80+ values (run while LinuxCNC is running for typical values):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
import linuxcnc
try:
    s = linuxcnc.stat() # create a connection to the status channel
    s.poll() # get current values
except linuxcnc.error, detail:
    print("error", detail)
    sys.exit(1)
for x in dir(s):
    if not x.startswith("_"):
        print(x, getattr(s,x))
```

The linuxcnc module uses the default compiled-in path to the NML configuration file unless overridden, see ReadingINI file values for an example.

#### 13.2.3.1 linuxcnc.stat attributes

**acceleration**
*(returns float)* - default acceleration, reflects the INI entry [TRAJ]DEFAULT_ACCELERATION.

**active_queue**
*(returns integer)* - number of motions blending.

**actual_position**
*(returns tuple of floats)* - current trajectory position, (x y z a b c u v w) in machine units.

**adaptive_feed_enabled**
*(returns boolean)* - status of adaptive feedrate override (0/1).

**ain**
*(returns tuple of floats)* - current value of the analog input pins.

**angular_units**
*(returns float)* - machine angular units per deg, reflects [TRAJ]ANGULAR_UNITS INI value.

**aout**
*(returns tuple of floats)* - current value of the analog output pins.

**axes (*Removed since version 2.9*)**
instead, use `axis_mask.bit_count()` to get the number of axes configured.

**axis**
*(returns tuple of dicts)* - reflecting current axis values. See The axis dictionary.

**axis_mask**
*(returns integer)* - mask of axis available as defined by `[TRAJ]COORDINATES` in the INI file. Returns the sum of the axes X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256.

**block_delete**
*(returns boolean)* - block delete current status.

**call_level**
*(returns integer)`* - current subroutine depth. - 0 if not in a subroutine, depth if not otherwise specified.

**command**
*(returns string)* - currently executing command.

**current_line**
*(returns integer)* - currently executing line.

**current_vel**
*(returns float)* - current velocity in user units per second.

**cycle_time**
*(returns float)* - thread period

**debug**
*(returns integer)* - debug flag from the INI file.

**delay_left**
*(returns float)* - remaining time on dwell (G4) command, seconds.

**din**
*(returns tuple of integers)* - current value of the digital input pins.

**distance_to_go**
*(returns float)* - remaining distance of current move, as reported by trajectory planner.

**dout**
*(returns tuple of integers)* - current value of the digital output pins.

**dtg**
*(returns tuple of floats)* - remaining distance of current move for each axis, as reported by trajectory planner.

**echo_serial_number**
*(returns integer)* - The serial number of the last completed command sent by a UI to task. All commands carry a serial number. Once the command has been executed, its serial number is reflected in `echo_serial_number`.

**enabled**
*(returns boolean)* - trajectory planner enabled flag.

**estop**
*(returns integer)* - Returns either STATE_ESTOP or not.

**exec_state**
*(returns integer)* - task execution state. One of EXEC_ERROR, EXEC_DONE, EXEC_WAITING_FOR_MO
EXEC_WAITING_FOR_MOTION_QUEUE, EXEC_WAITING_FOR_IO, EXEC_WAITING_FOR_MOTION_A
EXEC_WAITING_FOR_DELAY, EXEC_WAITING_FOR_SYSTEM_CMD, EXEC_WAITING_FOR_SPINDLE_

**feed_hold_enabled**
*(returns boolean)* - enable flag for feed hold.

**feed_override_enabled**
*(returns boolean)* - enable flag for feed override.

**feedrate**
*(returns float)* - current feedrate override, 1.0 = 100%.

**file**
*(returns string)* - currently loaded G-code filename with path.

**flood**
*(returns integer)* - Flood status, either FLOOD_OFF or FLOOD_ON.

**g5x_index**
*(returns integer)* - currently active coordinate system, G54=1, G55=2 etc.

**g5x_offset**
*(returns tuple of floats)* - offset of the currently active coordinate system.

**g92_offset**
*(returns tuple of floats)* - pose of the current g92 offset.

**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*)

**heartbeat**
*(returns integer)* - motion controller heartbeat counter. Increments every servo cycle. Rate is determined by `[EMCMOT]SERVO_PERIOD` in the INI file.

**homed**
*(returns tuple of integers)* - currently homed joints, with 0 = not homed, 1 = homed.

**id**
*(returns integer)* - currently executing motion ID.

**ini_filename**
*(returns string)* - path to the INI file passed to linuxcnc.

**inpos**
*(returns boolean)* - machine-in-position flag.

**input_timeout**
*(returns boolean)* - flag for M66 timer in progress.

**interp_state**
*(returns integer)* - current state of RS274NGC interpreter. One of INTERP_IDLE, INTERP_READING, INTERP_PAUSED, INTERP_WAITING.

**interpreter_errcode**
*(returns integer)* - current RS274NGC interpreter return code. One of INTERP_OK, INTERP_EXIT, INTERP_EXECUTE_FINISH, INTERP_ENDFILE, INTERP_FILE_NOT_OPEN, INTERP_ERROR. see src/emc/nml_intf/interp_return.hh

**joint**
*(returns tuple of dicts)* - reflecting current joint values. See The joint dictionary.

**joint_actual_position**
*(returns tuple of floats)* - actual joint positions.

**joint_position**
*(returns tuple of floats)* - Desired joint positions.

**joints**
*(returns integer)* - number of joints. Reflects [KINS]JOINTS INI value.

**kinematics_type**
*(returns integer)* - The type of kinematics. One of:

- KINEMATICS_IDENTITY
- KINEMATICS_FORWARD_ONLY
- KINEMATICS_INVERSE_ONLY
- KINEMATICS_BOTH

**limit**
*(returns tuple of integers)* - axis limit masks. minHardLimit=1, maxHardLimit=2, minSoftLimit=4, maxSoftLimit=8.

**linear_units**
*(returns float)* - machine linear units per mm, reflects [TRAJ]LINEAR_UNITS INI value.

**max_acceleration**
*(returns float)* - maximum acceleration. Reflects [TRAJ]MAX_ACCELERATION.

**max_velocity**
*(returns float)* - maximum velocity. Reflects the current maximum velocity. If not modified by halui.max-velocity or similar it should reflect [TRAJ]MAX_VELOCITY.

**mcodes**
*(returns tuple of 10 integers)* - currently active M-codes.

**mist**
*(returns integer)* - Mist status, either MIST_OFF or MIST_ON

**motion_line**
*(returns integer)* - source line number motion is currently executing. Relation to `id` unclear.

**motion_mode**
*(returns integer)* - This is the mode of the Motion controller. One of TRAJ_MODE_COORD, TRAJ_MODE_FREE, TRAJ_MODE_TELEOP.

**motion_type**
*(returns integer)* - The type of the currently executing motion. One of:

- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING

- MOTION_TYPE_INDEXROTARY
- Or 0 if no motion is currently taking place.

**optional_stop**
*(returns integer)* - option stop flag.

**paused**
*(returns boolean)* - `motion paused` flag.

**single_stepping**
*(returns boolean)* - `motion single_stepping` flag.

**pocket_prepped**
*(returns integer)* - A Tx command completed, and this pocket is prepared. -1 if no prepared pocket.

**poll()**
-*(built-in function)* method to update current status attributes.

**position**
*(returns tuple of floats)* - trajectory position.

**probe_tripped**
*(returns boolean)* - flag, True if probe has tripped (latch).

**probe_val**
*(returns integer)* - reflects value of the `motion.probe-input` pin.

**probed_position**
*(returns tuple of floats)* - position where probe tripped.

**probing**
*(returns boolean)* - flag, True if a probe operation is in progress.

**program_units**
*(returns integer)* - one of CANON_UNITS_INCHES=1, CANON_UNITS_MM=2, CANON_UNITS_CM=3

**queue**
*(returns integer)* - current size of the trajectory planner queue.

**queue_full**
*(returns boolean)* - the trajectory planner queue is full.

**rapidrate**
*(returns float)* - rapid override scale.

**read_line**
*(returns integer)* - line the RS274NGC interpreter is currently reading.

**rotation_xy**
*(returns float)* - current XY rotation angle around Z axis.

**settings**
*(returns tuple of floats)* - current interpreter settings:
settings[0] = sequence number,
settings[1] = feed rate,
settings[2] = speed,
settings[3] = `G64 P` blend tolerance,
settings[4] = `G64 Q` naive CAM tolerance.

**spindle**
’ (returns tuple of dicts) ’ - returns the current spindle status, see The spindle dictionary.

**spindles**
*(returns integer)* - number of spindles. Reflects `[TRAJ]SPINDLES` INI value.

**state**
*(returns integer)* - current command execution status. One of `RCS_DONE`, `RCS_EXEC`, `RCS_ERROR`.

**task_mode**
*(returns integer)* - current task mode. One of `MODE_MDI`, `MODE_AUTO`, `MODE_MANUAL`.

**task_paused**
*(returns integer)* - task paused flag.

**task_state**
*(returns integer)* - current task state. One of `STATE_ESTOP`, `STATE_ESTOP_RESET`, `STATE_ON`, `STATE_OFF`.

**tool_in_spindle**
*(returns integer)* - current tool number.

**taskbeat**
*(returns integer)* - task main loop heartbeat counter. Increments every task cycle. Rate is determined by `[TASK]CYCLE_TIME` in the INI file.

**tool_from_pocket**
*(returns integer)* - pocket number for the currently loaded tool (0 if no tool loaded).

**tool_offset**
*(returns tuple of floats)* - offset values of the current tool.

**tool_table**
*(returns tuple of tool_results)* - list of tool entries. Each entry is a sequence of the following fields: id, xoffset, yoffset, zoffset, aoffset, boffset, coffset, uoffset, voffset, woffset, diameter, frontangle, backangle, orientation. The id and orientation are integers and the rest are floats.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
# to find the loaded tool information it is in tool table index 0
if s.tool_table[0].id != 0: # a tool is loaded
    print(s.tool_table[0].zoffset)
else:
    print(”No tool loaded.”)
```

**toolinfo(*toolno*)**
*(returns dict of tooldata for toolno)* - An initial stat.poll() is required to initialize. *toolno* must be greater than zero and less than or equal to the highest tool number in use. Dictionary items include all tooldata items: *toolno*, *pocketno*, *diameter*,*frontangle*,*backangle*,*orientation*, *xoffset*,*yoffset*, ... *woffset*, *comment*.
As an example, the following script

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
toolno = 1
print(s.toolinfo(toolno))
```

produces the output:

```
': 0, 'xoffset': 0.0, 'yoffset': 0.0, 'zoffset': 0.18, 'aoffset': 0.0, 'boffset': 0.0, ' ↩
    coffset': 0.0, 'uoffset': 0.0, 'voffset': 0.0, 'woffset': 0.0, 'comment': 'Tool_18 28 ↩
    Jan23:18.53.25'}
```

**velocity**
*(returns float)* - This property is defined, but it does not have a useful interpretation.

#### 13.2.3.2 The `axis` dictionary

The axis configuration and status values are available through a list of per-axis dictionaries. Here's an example how to access an attribute of a particular axis: Note that many properties that were formerly in the `axis` dictionary are now in the `joint` dictionary, because on nontrivial kinematics machines these items (such as backlash) are not the properties of an axis.

**max_position_limit**
*(returns float)* - maximum limit (soft limit) for axis motion, in machine units.configuration parameter, reflects `[JOINT___n__]MAX_LIMIT`.

**min_position_limit**
*(returns float)* - minimum limit (soft limit) for axis motion, in machine units.configuration parameter, reflects `[JOINT___n__]MIN_LIMIT`.

**velocity**
*(returns float)* - current velocity.

#### 13.2.3.3 The `joint` dictionary

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
print("Joint 1 homed: ", s.joint[1]["homed"])
```

For each joint, the following dictionary keys are available:

**backlash**
*(returns float)* - Backlash in machine units. configuration parameter, reflects `[JOINT___n__]BACKLASH`.

**enabled**
*(returns integer)* - non-zero means enabled.

**fault**
*(returns integer)* - non-zero means axis amp fault.

**ferror_current**
*(returns float)* - current following error.

**ferror_highmark**
*(returns float)* - magnitude of max following error.

**homed**
*(returns integer)* - non-zero means has been homed.

**homing**
*(returns integer)* - non-zero means homing in progress.

**inpos**
*(returns integer)* - non-zero means in position.

**input**
*(returns float)* - current input position.

**jointType**
*(returns integer)* - type of axis configuration parameter, reflects `[JOINT___n__]TYPE` with `LINEAR=1`, `ANGULAR=2`. See Joint INI configuration for details.

**max_ferror**
*(returns float)* - maximum following error. configuration parameter, reflects `[JOINT___n__]FERROR`.

**max_hard_limit**
*(returns integer)* - non-zero means max hard limit exceeded.

**max_position_limit**
*(returns float)* - maximum limit (soft limit) for joint motion, in machine units. configuration parameter, reflects `[JOINT___n__]MAX_LIMIT`.

**max_soft_limit**
non-zero means `max_position_limit` was exceeded, int

**min_ferror**
*(returns float)* - configuration parameter, reflects `[JOINT___n__]MIN_FERROR`.

**min_hard_limit**
*(returns integer)* - non-zero means min hard limit exceeded.

**min_position_limit**
*(returns float)* - minimum limit (soft limit) for joint motion, in machine units. configuration parameter, reflects `[JOINT___n__]MIN_LIMIT`.

**min_soft_limit**
*(returns integer)* - non-zero means `min_position_limit` was exceeded.

**output**
*(returns float)* - commanded output position.

**override_limits**
*(returns integer)* - non-zero means limits are overridden.

**units**
*(returns float)* - joint units per mm, or per degree for angular joints.
(joint units are the same as machine units, unless set otherwise by the configuration parameter `[JOINT___n__]UNITS`)

**velocity**
*(returns float)* - current velocity.

#### 13.2.3.4 The `spindle` dictionary

**brake**
*(returns integer)* - value of the spindle brake flag.

**direction**
*(returns integer)* - rotational direction of the spindle with forward=1, reverse=-1.

**enabled**
*(returns integer)* - value of the spindle enabled flag.

**homed**
(not currently implemented)

**increasing**
*(returns integer)* - unclear.

**orient_fault**
*(returns integer)*

**orient_state**
*(returns integer)*

**override**
*(returns float)* - spindle speed override scale.

**override_enabled**
*(returns boolean)* - value of the spindle override enabled flag.

**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 Preparing to send commands

Some commands can always be sent, regardless of mode and state; for instance, the `linuxcnc.command.abo` method can always be called.

Other commands may be sent only in appropriate state, and those tests can be a bit tricky. For instance, an MDI command can be sent only if:

- ESTOP has not been triggered, and
- the machine is turned on and
- the axes are homed and
- the interpreter is not running and
- the mode is set to `MDI mode`

An appropriate test before sending an MDI command through `linuxcnc.command.mdi()` could be:

```
#!/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() # wait until mode switch executed (or default timeout of 5s occurs!)
    c.mdi("G0 X10 Y20 Z30")
```

**Warning**
Read important information on *wait_complete()* in the 'linuxcnc.command methods' section below.

### 13.2.5 Sending commands through `linuxcnc.command`

Before sending a command, initialize a command channel like so:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# Usage examples for some of the commands listed below:
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` attributes

**serial**
the current command serial number

#### 13.2.5.2 `linuxcnc.command` methods:

**abort()**
send EMC_TASK_ABORT message.

**auto(int[, int])**
run, step, pause or resume a program.

**brake(int)**
engage or release spindle brake.

**debug(int)**
set debug level via EMC_SET_DEBUG message.

**display_msg(string)**
sends a operator display message to the screen. (max 254 characters)

**error_msg(string)**
sends a operator error message to the screen. (max 254 characters)

**feedrate(float)**
set the feedrate override, 1.0 = 100%.

**flood(int)**
turn on/off flooding.

**Syntax**
flood(command)
flood(linuxcnc.FLOOD_ON)
flood(linuxcnc.FLOOD_OFF)

**Constants**
FLOOD_ON
FLOOD_OFF

**home(int)**
home a given joint.

**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)

**Command Constants**
linuxcnc.JOG_STOP
linuxcnc.JOG_CONTINUOUS
linuxcnc.JOG_INCREMENT

**jjogmode**

**True**
request individual joint jog (requires teleop_enable(0))

**False**
request axis Cartesian coordinate jog (requires teleop_enable(1))

**joint_num_or_axis_index**

**For joint jog (jjogmode=1)**
joint_number

**For axis Cartesian coordinate jog (jjogmode=0)**
zero-based index of the axis coordinate with respect to the known coordinate letters XYZABCUVW (x=>0,y=>1,z=>2,a=>3,b=>4,c=>5,u=>6,v=>7,w=>8)

**load_tool_table()**
reload the tool table.

**maxvel(float)**
set maximum velocity

**mdi(string)**
send an MDI command. Maximum 254 chars.

**mist(int)**
turn on/off mist.

**Syntax**
mist(command)
mist(linuxcnc.MIST_ON)
mist(linuxcnc.MIST_OFF)

**Constants**
MIST_ON
MIST_OFF

**mode(int)**
set mode (MODE_MDI, MODE_MANUAL, MODE_AUTO).

**override_limits()**
set the override axis limits flag.

**program_open(string)**
open an NGC file.

**rapidrate()**
set rapid override factor

**reset_interpreter()**
reset the RS274NGC interpreter

**set_adaptive_feed(int)**
set adaptive feed flag

**set_analog_output(int, float)**
set analog output pin to value

**set_block_delete(int)**
set block delete flag

**set_digital_output(int, int)**
set digital output pin to value

**set_feed_hold(int)**
set feed hold on/off

**set_feed_override(int)**
set feed override on/off

**set_max_limit(int, float)**
set max position limit for a given axis

**set_min_limit()**
set min position limit for a given axis

**set_optional_stop(int)**
set optional stop on/off

**set_spindle_override(int [, int])**
set spindle override enabled. Defaults to spindle 0.

**spindle(direction: int, speed: float=0, spindle: int=0, wait_for_speed: int=0)**

- Direction: [SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECREA or SPINDLE_CONSTANT]
- Speed: Speed in RPM, defaults to 0.
- Spindle: Spindle number to command defaults to 0.
- Wait_for_speed: if 1 motion will wait for speed before continuing, defaults to not.

**Warning**
MDI commands will ignore this. ”S1000” after this will turn the spindle off.

**text_msg(string)**
sends a operator text message to the screen (max 254 characters).

```
#!/usr/bin/env python3
import linuxcnc
c = linuxcnc.command()

# Increase speed of spindle 0 by 100rpm. Spindle must be on first.
c.spindle(linuxcnc.INCREASE)

# Increase speed of spindle 2 by 100rpm. Spindle must be on first.
c.spindle(linuxcnc.SPINDLE_INCREASE, 2)

# Set speed of spindle 0 to 1024 rpm.
c.spindle.(linuxcnc.SPINDLE_FORWARD, 1024)

# Set speed of spindle 1 to -666 rpm.
c.spindle.(linuxcnc.SPINDLE_REVERSE, 666, 1)

# Stop spindle 0.
c.spindle.(linuxcnc.SPINDLE_OFF)

# Stop spindle 0 explicitly.
c.spindle.(linuxcnc.SPINDLE_OFF, 0)
```

**spindleoverride(float [, int])**
Set spindle override factor. Defaults to spindle 0.

**state(int)**
Set the machine state. Machine state should be STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON, or STATE_OFF.

**task_plan_sync()**
On completion of this call, the VAR file on disk is updated with live values from the interpreter.

**teleop_enable(int)**
Enable/disable teleop mode (disable for joint jogging).

**tool_offset(int, float, float, float, float, float, int)**
Set the tool offset. See usage example above.

**traj_mode(int)**
Set trajectory mode. Mode is one of `MODE_FREE`, `MODE_COORD`, or `MODE_TELEOP`.

**unhome(int)**
Unhome a given joint.

**wait_complete([float])**
Wait for completion of the last command. Takes an optional timeout value in seconds.
Timeout defaults to 5 seconds if omitted.
Returns -1 if timed out.
Returns `RCS_DONE` or `RCS_ERROR` according to command execution status.
Note that python execution will be blocked until this function returns.

### 13.2.6 Reading the error channel

To handle error messages, connect to the error channel and periodically poll() it.

Note that the NML channel for error messages has a queue (other than the command and status channels), which means that the first consumer of an error message deletes that message from the queue; whether your another error message consumer (e.g. AXIS) will *see* the message is dependent on timing. It is recommended to have just one error channel reader task in a setup.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
e = linuxcnc.error_channel()

error = e.poll()

if error:
    kind, text = error
    if kind in (linuxcnc.NML_ERROR, linuxcnc.OPERATOR_ERROR):
        typus = "error"
    else:
        typus = "info"
    print(typus, text)
```

### 13.2.7 Reading INI file values

Here's an example for reading values from an INI file through the `linuxcnc.ini` object:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# run as:
# 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.getstring("EMC", "MACHINE", fallback="unknown")
print("machine name: ", machine_name)

# inifile.findall() returns a list of matches, or an empty list
# if the key wasn't found:

extensions = inifile.findall("FILTER", "PROGRAM_EXTENSION")
print("extensions: ", extensions)

# override default NML file by INI parameter if given
nmlfile = inifile.getstring("EMC", "NML_FILE", fallback="")
if nmlfile:
    linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)

# Other examples:
realval = inifile.getreal("AXIS_X", "MAX_VELOCITY", fallback=5.0)
boolval = inifile.getbool("JOINT_0", "HOME_USE_INDEX", fallback=False)

# None is returned without fallback= if the variable was not found
intval  = inifile.getint( "KINS", "JOINTS")
if None == intval:
    print("Error: [KINS]JOINTS not defined or an invalid integer"
```

Or for the same INI file as LinuxCNC:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# run as:
# python3 ini-example2.py

import linuxcnc

stat = linuxcnc.stat()
stat.poll()

inifile = linuxcnc.ini(stat.ini_filename)

# See example above for usage of 'inifile' object
```

### 13.2.8 The `linuxcnc.positionlogger` type

Some usage hints can be gleaned from `src/emc/usr_intf/gremlin/gremlin.py`.

#### 13.2.8.1 members

**npts**
number of points.

#### 13.2.8.2 methods

**start(float)**
start the position logger and run every ARG seconds

**clear()**
clear the position logger

**stop()**
stop the position logger

**call()**
Plot the backplot now.

**last([int])**
Return the most recent point on the plot or None

## 13.3 HAL Python модуль

This documentation describes the `hal` python module, which provides a Python API for creating and accessing HAL pins and signals.

### 13.3.1 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 Функции

**component**
+
Сам компонент создается вызовом конструктора *hal.component*. Аргументами являются имя компонента HAL и (опционально) префикс, используемый для имен выводов и параметров. Если префикс не указан, используется имя компонента.
.Пример

```
h = hal.component("passthrough")
```

**newpin**
+
Создать новый контакт.
Аргументы: суффикс имени контакта, тип контакта и направление контакта. Для параметров аргументами являются: суффикс имени параметра, тип параметра и направление параметра.
.Пример:

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

**ready**
Tells the HAL system the component is initialized. Locks out adding pins.

**unready**
Allows a component to add pins after *ready()* has been called. One should call *ready()* on the component after.

**component_exists**
Does the specified component exist at this time.

**Пример**

```
hal.component_exists(”testpanel”)
```

**component_is_ready**
Is the specified component ready at this time.

**Пример**

```
hal.component_is_ready(”testpanel”)
```

**get_msg_level**
Get the current Realtime msg level.

**set_msg_level**
Set the current Realtime msg level. used for debugging information.

**connect**
Connect a pin to a signal.

**Пример**

```
hal.connect(”pinname”,”signal_name”)
```

**disconnect**
Disconnect a pin from a signal.

**Пример**

```
hal.disconnect(”pinname”)
```

**get_value**
Read a pin, param, or signal directly.

**Пример**

```
value = hal.get_value(”iocontrol.0.emc-enable-in”)
```

**get_info_pins()**
Returns a list of dicts of all system pins.

```
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()**
Returns a list of dicts of all system signals.

```
listOfDicts = hal.get_info_signals()
signalName1 = listOfDicts[0].get('NAME')
signalValue1 = listOfDicts[0].get('VALUE')
driverPin1 = listOfDicts[0].get('DRIVER')
```

**get_info_params()**
Returns a list of dicts of all system parameters.

```
listOfDicts = hal.get_info_params()
paramName1 = listOfDicts[0].get('NAME')
paramValue1 = listOfDicts[0].get('VALUE')
paramDirection1 = listOfDicts[0].get('DIRECTION')
```

**new_sig**
Create a new signal of the type specified.

**Пример**

```
hal.new_sig("signalname",hal.HAL_BIT)
```

**pin_has_writer**
Does the specified pin have a driving pin connected.
Returns True or False.

```
h.in.pin_has_writer()
```

**get_name**
Get the HAL object name.
Return a string.

```
h.in.get_name()
```

**get_type**
Get the HAL object's type.
Returns an integer.

```
h.in.get_type()
```

**get_dir**
Get the HAL object direction type.
Returns an integer.

```
h.in.get_dir()
```

**get**
Get the HAL object value.

```
h.in.get()
```

**set**
Set the HAL object value.

```
h.out.set(10)
```

**is_pin**
Is the object a pin or parameter?
Returns True or False.

```
h.in.is_pin()
```

**sampler_base**
TODO

**stream_base**
TODO

**stream**
TODO

**set_p**
Set a pin value of any pin in the HAL system.

**Пример**

```
hal.set_p(”pinname”,”10”)
```

**set_s**
Set the value of any unconnected signal in the HAL system.

**Пример**

```
hal.set_s(”signalname”,”10”)
```

## 13.4 GStat Python Module

### 13.4.1 Intro

GStat is a Python class used to send messages from LinuxCNC to other Python programs. It uses GObject to deliver messages, making it easy to listen for specific information. This is referred to as event-driven programming, which is more efficient then every program polling LinuxCNC at the same time. GladeVCP, Gscreen, Gmoccapy and QtVCP use GStat extensively. GStat is in the `hal_glib` module.

Обзор

- First, a program imports the `hal_glib` module and instantiates GStat.
- Then it *connects* to the messages it wishes to monitor.
- 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.
- When GStat calls the registered function, it sends the GStat object plus any return codes from the message.

Typical code signatures:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', FUNCTION_TO_CALL)

def FUNCTION_TO_CALL(gstat_object, return_codes):
```

Often LAMBDA is used to strip the GSTAT object and manipulate the return codes:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', lambda o, return: FUNCTION_TO_CALL(not return))

def FUNCTION_TO_CALL(return_codes):
```

### 13.4.2 Sample GStat Code

There are some basic patterns for using GStat, depending on what library you are using them in. If using GStat with GladeVCP, Gscreen, or QtVCP, the GObject library is not needed as those toolkits already set up GObject.

#### 13.4.2.1 Sample HAL component code pattern

This program creates two HAL pins that output the status of G20/G21.

```
#!/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 to change HAL pin state
def mode_changed(obj, data):
        h['g20'] = not data
```

```
        h['g21'] = data

# Make a component and pins
h = hal.component("metric_status")
h.newpin("g20", hal.HAL_BIT, hal.HAL_OUT)
h.newpin("g21", hal.HAL_BIT, hal.HAL_OUT)
h.ready()

# connect a GSTAT message to a callback function
GSTAT.connect("metric-mode-changed",mode_changed)

# force GSTAT to initialize states
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 extension code pattern

This file assumes there are three GTK labels named:

- *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 extension code pattern

QtVCP extends GStat, so must be loaded differently but all the messages are available in QtVCP. This handler file assumes there are three QLabels named:

- *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 Messages

**periodic**
*(returns nothing)* - sent every 100 ms.

**state-estop**
*(returns nothing)* - Sent when LinuxCNC is goes into estop.

**state-estop-reset**
*(returns nothing)* - Sent when LinuxCNC comes out of estop.

**state-on**
*(returns nothing)* - Sent when LinuxCNC is in machine on state.

**state-off**
*(returns nothing)* - Sent when LinuxCNC is in machine off state.

**homed**
*(returns string)* - Sent as each joint is homed.

**all-homed**
*(returns nothing)* - Sent when all defined joints are homed.

**not-all-homed**
*(returns string)* - Sends a list of joints not currently homed.

**override_limits_changed**
*(returns string)* - Sent if LinuxCNC has been directed to override its limits.

**hard-limits-tripped**
*(returns bool, Python List)* - Sent when any hard limit is tripped. bool indicates if any limit is tripped, the list shows all available joint's current limit values.

**mode-manual**
*(returns nothing)* - Sent when LinuxCNC switches to manual mode.

**mode-mdi**
*(returns nothing)* - Sent when LinuxCNC switches to MDI mode.

**mode-auto**
*(returns nothing)* - Sent when LinuxCNC switches to auto mode.

**command-running**
*(returns nothing)* - Sent when running a program or MDI

**command-stopped**
*(returns nothing)* - Sent when a program or MDI stopped

**command-error**
*(returns nothing)* - Sent when there is a command error

**interp-run**
*(returns nothing)* - Sent when LinuxCNC's interpreter is running an MDI or program.

**interp-idle**
*(returns nothing)* - Sent when LinuxCNC's interpreter is idle.

**interp-paused**
*(returns nothing)* - Sent when LinuxCNC's interpreter is paused.

**interp-reading**
*(returns nothing)* - Sent when LinuxCNC's interpreter is reading.

**interp-waiting**
*(returns nothing)* - Sent when LinuxCNC's interpreter is waiting.

**jograte-changed**
*(returns float)* - Sent when jog rate has changed.
LinuxCNC does not have an internal jog rate.
This is GStat's internal jog rate.
It is expected to be in the machine's native units regardless of the current unit mode .

**jograte-angular-changed**
*(returns float)* - Sent when the angular jog rate has changed.
LinuxCNC does not have an internal angular jog rate.
This is GStat's internal jog rate.
It is expected to be in the machine's native units regardless of the current unit mode .

**jogincrement-changed**
*(returns float, text)* - Sent when jog increment has changed.
LinuxCNC does not have an internal jog increment.
This is GStat's internal jog increment.
It is expected to be in the machine's native units regardless of the current unit mode .

**jogincrement-angular-changed**
*(returns float, text)* - Sent when angular jog increment has changed.
LinuxCNC does not have an internal angular jog increment.
This is GStat's internal angular jog increment.
It is expected to be in the machine's native units regardless of the current unit mode .

**program-pause-changed**
*(returns bool)* - Sent when program is paused/unpaused.

**optional-stop-changed**
*(returns bool)* - Sent when optional stop is set/unset

**block-delete-changed**
*(returns bool)* - sent when block delete is set/unset.

**file-loaded**
*(returns string)* - Sent when LinuxCNC has loaded a file

**reload-display**
*(returns nothing)* - Sent when there is a request to reload the display

**line-changed**
*(returns integer)* - Sent when LinuxCNC has read a new line.
LinuxCNC does not update this for every type of line.

**tool-in-spindle-changed**
*(returns integer)* - Sent when the tool has changed.

**tool-info-changed**
*(returns Python object)* - Sent when current tool info changes.

**current-tool-offset**
*(returns Python object)* - Sent when the current tool offsets change.

**motion-mode-changed**
*(returns integer)* - Sent when motion's mode has changed

**spindle-control-changed**
*(returns integer, bool, integer, bool)* - (spindle num, spindle on state, requested spindle direction & rate, at-speed state)
Sent when spindle direction or running status changes or at-speed changes.

**current-feed-rate**
*(returns float)* - Sent when the current feed rate changes.

**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**
*(returns float)* - Sent when the current requested RPM changes

**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)* - Sent when the spindle override value changes
in percent

**feed-override-changed**
*(returns float)* - Sent when the feed override value changes
in percent

**rapid-override-changed**
*(returns float)* - Sent when the rapid override value changes
in percent (0-100)

**max-velocity-override-changed**
*(returns float)* - Sent when the maximum velocity override value changes
in units per minute

**feed-hold-enabled-changed**
*(returns bool)* - Sent when feed hold status changes

**itime-mode**
*(returns bool)* - Sent when G93 status changes
(inverse time mode)

**fpm-mode**
*(returns bool)* - Sent when G94 status changes
(feed per minute mode)

**fpr-mode**
*(returns bool)* - Sent when G95 status changes
(feed per revolution mode)

**css-mode**
*(returns bool)* - Sent when G96 status changes
(constant surface feed mode)

**rpm-mode**
*(returns bool)* - Sent when G97 status changes
(constant RPM mode)

**radius-mode**
*(returns bool)* - Sent when G8 status changes
display X in radius mode

**diameter-mode**
*(returns bool)* - Sent when G7 status changes
display X in Diameter mode

**flood-changed**
*(returns bool)* - Sent when flood coolant state changes.

**mist-changed**
*(returns bool )* - Sent when mist coolant state changes.

**m-code-changed**
*(returns string)* - Sent when active M-codes change

**g-code-changed**
*(returns string)* - Sent when active G-code change

**metric-mode-changed**
*(returns bool)* - Sent when G21 status changes

**user-system-changed**
*(returns string)* - Sent when the reference coordinate system (G5x) changes

**mdi-line-selected**
*(returns string, string)* - intended to be sent when an MDI line is selected by user.
This depends on the widget/libraries used.

**gcode-line-selected**
*(returns integer)* - intended to be sent when a G-code line is selected by user.
This depends on the widget/libraries used.

**graphics-line-selected**
*(returns integer)* - intended to be sent when graphics line is selected by user.
This depends on the widget/libraries used.

**graphics-loading-progress**
*(returns integer)* - intended to return percentage done of loading a program or running a program.
This depends on the widget/libraries used.

**graphics-gcode-error**
*(returns string)* - intended to be sent when a G-code error is found when loading.
This depends on the widget/libraries used.

**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*)

**Note**

1. See the images extends non-rotated and extends rotated 30 degrees for a better understanding.

**graphics-view-changed**
*(returns string, Python dict or None)* - intended to be sent when graphics view is changed.
This depends on the widget/libraries used.

**mdi-history-changed**
*(returns None)* - intended to be sent when an MDI history needs to be reloaded.
This depends on the widget/libraries used.

**machine-log-changed**
*(returns None)* - intended to be sent when machine log has changed.
This depends on the widget/libraries used.

**update-machine-log**
*(returns string, string)* - intended to be sent when updating the machine.
This depends on the widget/libraries used.

**move-text-lineup**
*(returns None)* - intended to be sent when moving the cursor one line up in G-code display.
This depends on the widget/libraries used.

**move-text-linedown**
*(returns None)* - intended to be sent when moving the cursor one line down in G-code display.
This depends on the widget/libraries used.

**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: *requested dialog name*
  The dict usually has several keyname pairs - it depends on the dialog.
  dialogs return information using a general message
  This depends on the widget/libraries used.

**focus-overlay-changed**
*(returns bool, string, Python object)* - intended to be sent when requesting an overlay to be put over the display.
This depends on the widget/libraries used.

**play-sound**
*(returns string)* - intended to be sent when requesting a specific sound file to be played.
This depends on the widget/libraries used.

**virtual-keyboard**
*(returns string)* - intended to be sent when requesting a on screen keyboard.
This depends on the widget/libraries used.

**dro-reference-change-request**
*(returns integer)* - intended to be sent when requesting a DRO widget to change its reference.
0 = machine, 1 = relative, 3 = distance-to-go
This depends on the widget/libraries used.

**show-preferences**
*(returns None)* - intended to be sent when requesting the screen preferences to be displayed.
This depends on the widget/libraries used.

**shutdown**
*(returns None)* - intended to be sent when requesting LinuxCNC to shutdown.
This depends on the widget/libraries used.

**status-message**
*returns python dict (message), python dict (options)* Intended for a screen/panel to get status/log messages from widgets, but can be used generally.
The listening object is expected to look for and handle at least these entries:

The message dict would include:

- TITLE: (string)
- SHORTTEXT: (string)
- DETAILS: (string)

The options dict would include:

- LEVEL: (integer)
- LOG: (bool)

The listening object could use this to display information on a text line or message dialog.
The LEVEL would indicate urgency 0 = DEFAULT 1 = WARNING 2 = CRITICAL
LOG indicates whether the message should be logged to a file/page if available.
LOG messages would be assumed to use the DETAILS entry.

An example of how to send a message:

```
mess = {'SHORTTEXT':'File Copy Failed',
        'TITLE':'FileManager',
        'DETAILS':'There is not enough room on disk to copy the file'}
opt = {'LOG':True,'LEVEL':2}
STATUS.emit('status-message',mess,opt)
```

An example of the listening object:

```
# tell STATUS we want to respond to any sent 'status-message' messages.
STATUS.connect('status-message', lambda w, d, o: self.add_external_status(m,o))

    def add_external_status(self, message, option):

        # extract and trap errors for expected entries
        level = option.get('LEVEL', STATUS.DEFAULT)
        log = option.get("LOG", True)
        title = message.get('TITLE', '')
        mess = message.get('SHORTTEXT', '')
        logtext = message.get('DETAILS', '')

        # call a function to print the message on a statusbar:
        self.add_status(mess, level, noLog=True)

        # request a log file update
        if log:
            STATUS.emit('update-machine-log', "{}\n{}".format(title, logtext), 'TIME')
```

**error**
*(returns integer, string)* - intended to be sent when an error has been reported .
integer represents the kind of error. ERROR, TEXT or DISPLAY
string is the actual error message.
This depends on the widget/libraries used.

**general**
*(returns Python dict)* - intended to be sent when message must be sent that is not covered by a more specific message.
General message should be used a sparsely as reasonable because all object connected to it will have to parse it.
It uses a Python dict for communication.
The dict should include and be checked for a unique id keyname pair:

- ID: *UNIQUE_ID_CODE*
  The dict usually has more keyname pair - it depends on implementation.

**forced-update**
*(returns None)* - intended to be sent when one wishes to initialize or arbitrarily update an object.
This depends on the widget/libraries used.

**progress**
*(returns integer, Python object)* - intended to be sent to indicate the progress of a filter program.
This depends on the widget/libraries used.

**following-error**
*(returns Python list)* - returns a list of all joints current following error.

### 13.4.4 Функции

These are convenience functions that are commonly used in programming.

**set_jograte**
*(float)* - LinuxCNC has no internal concept of jog rate -each GUI has its own. This is not always convenient.
This function allows one to set a jog rate for all objects connected to the signal `jograte-changed`.
It defaults to 15.
GSTAT.set_jog_rate(10) would set the jog rate to 10 machine-units-per-minute and emit the `jograte-cha`
signal.

**get_jograte()**
*(Nothing)* - x = GSTAT.get_jograte() would return GSTAT's current internal jograte (float).

**set_jograte_angular**
*(float)* -

**get_jograte_angular**
*(None)* -

**set_jog_increment_angular**
*(float, string)* -

**get_jog_increment_angular**
*(None)* -

**set_jog_increments**
*(float, string)* -

**get_jog_increments**
*(None)* -

**is_all_homed**
*(nothing)* - This will return the current state of all_homed (BOOL).

**machine_is_on**
*(nothing)* - This will return the current state of machine (BOOL).

**estop_is_clear**
*(nothing)* - This will return the state of Estop (BOOL)

**set_tool_touchoff**
*(tool,axis,value)* - This command will

1. record the current mode,
2. switch to MDI mode,
3. invoke the MDI command: G10 L10 P[TOOL] [AXIS] [VALUE],
4. wait for it to complete,
5. invoke G43,
6. wait for it to complete,
7. switch back to the original mode.

**set_axis_origin**
*(axis,value)* - This command will

1. record the current mode,
2. switch to MDI mode,
3. invoke the MDI command: G10 L20 P0 [AXIS] [VALUE],
4. wait for it to complete,
5. switch back to the original mode,
6. emit a *reload-display* signal.

**do_jog**
*(axis_number,direction, distance)* - This will jog an axis continuously or at a set distance.
You must be in the proper mode to jog.

**check_for_modes**
*(mode)* - This function checks for required LinuxCNC mode.
It returns a Python tuple (state, mode)
mode will be set the mode the system is in
state will set to:

- false if mode is 0
- false if machine is busy
- true if LinuxCNC is in the requested mode
- None if possible to change, but not in requested mode

**get_current_mode**
*(nothing)* - returns integer: the current LinuxCNC mode.

**set_selected_joint**
*(integer)* - records the selected joint number internally.
requests the joint to be selected by emitting the
*joint-selection-changed* message.

**get_selected_joint**
*(None)* - returns integer representing the internal selected joint number.

**set_selected_axis**
*(string)* - records the selected axis letter internally.
Requests the axis to be selected by emitting the `axis-selection-changed` message.

**get_selected_axis**
*(None)* - returns string representing the internal selected axis letter.

**is_man_mode**
*(None)* -

**is_mdi_mode**
*(None)* -

**is_auto_mode**
*(None)* -

**is_on_and_idle**
*(None)* -

**is_auto_running**
*(None)* -

**is_auto_paused**
*(None)* -

**is_file_loaded**
*(None)* -

**is_metric_mode**
*(None)* -

**is_spindle_on**
*(None)* -

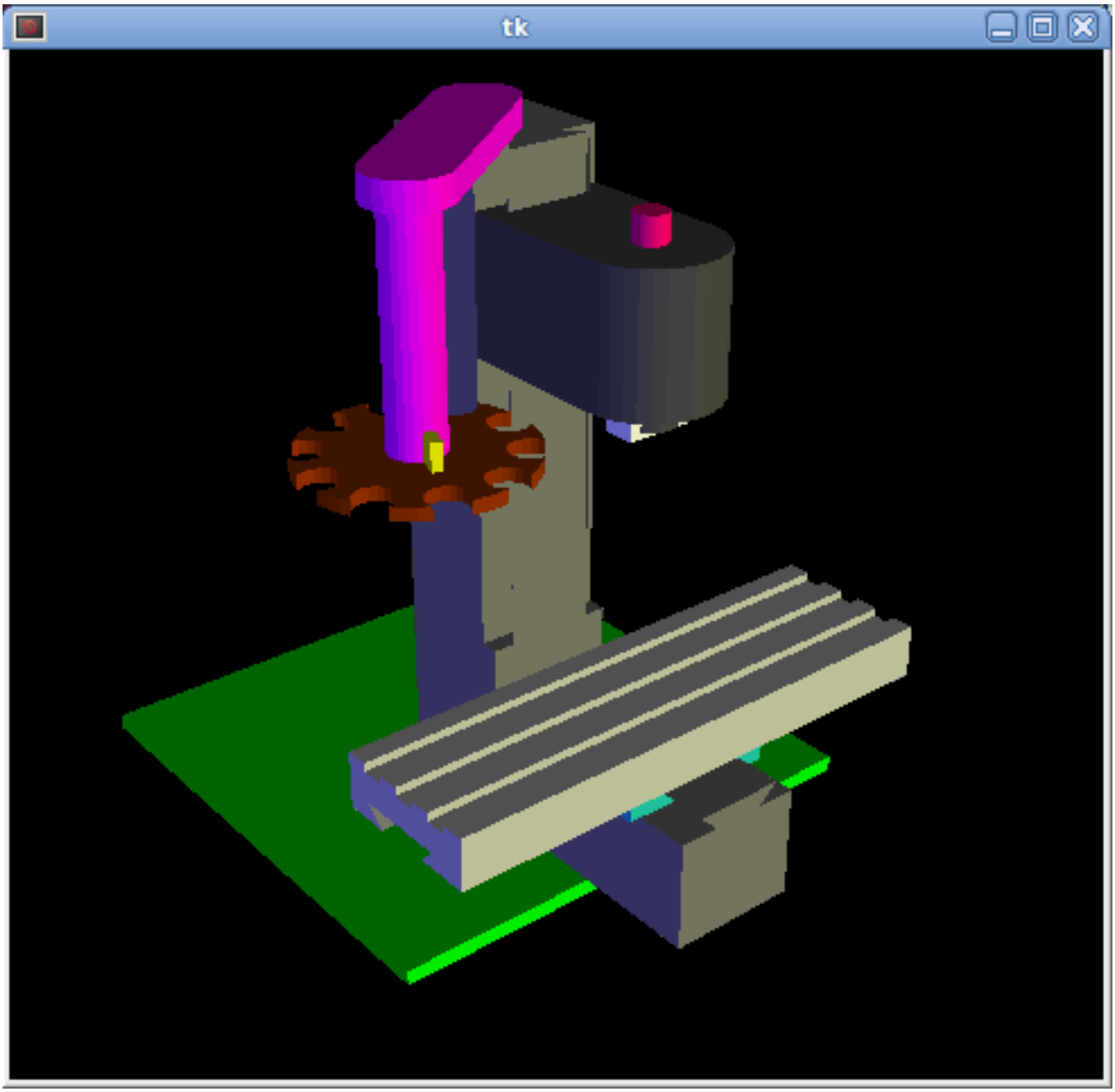
**shutdown**
*(None)* -

### 13.4.5 Known Issues

Some status points are reported wrongly during a running program because the interpreter runs ahead of the current position of a running program. This will hopefully be resolved with the merge of state-tags branch.

## 13.5 Vismach

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 and the model parts are animated as the values of associated HAL pins change.




The Vismach viewport view can be manipulated as follows:

- **zoom** by scroll wheel or right button drag,
- **pan** by left button drag,
- **rotate** by middle-button drag or shift-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 I will use the simplest and most basic of them.

The basic sequence in creating the Vismach model is

- Create the HAL pins that control the motion.
- Create the parts.

- Define how they move.
- Assemble into movement groups.

### 13.5.1 Start the script

It is useful for testing to include the *#!/usr/bin/env python3* to allow the file to be run as a script. The first thing to do is to import the required libraries.

```
#!/usr/bin/env python3

from vismach import *
import hal
import math
import sys
```

### 13.5.2 Create the 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()
```

Will create HAL pins *samplegui.joint0* and *samplegui.joint1*. When loading the Vismach model with *loadusr -W samplegui* the *c.ready()* function tells loadusr it’s ready.

### 13.5.3 Creating Parts

It is probably easiest to create geometry in a CAD package and **import** into the model script with the `AsciiSTL()` or `AsciiOBJ()` functions. Both functions can take one of two named arguments, either a filename or raw 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 ...”)`
  The parts will be created in the Vismach space in the same locations as they occupy in the STL or OBJ space. This means that it may be possible to assemble the model in the CAD package.

Alternatively parts can be created inside the model script from a range of **shape primitives**. Many shapes are created at the origin and need to be moved to the required location after creation:

- `cylinder = CylinderX(x1, r1, x2, r2) + cylinder = CylinderY(y1, r1, y2, r2) + cylinder = CylinderZ(z1, r1, z2, r2)`
  Creates a (optionally tapered) cylinder on the given axis with the given radii at the given points on the axis.
- `sphere = Sphere(x, y, z, r)`
  Creates a sphere of radius r at (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)`
  Creates a triangular plate between planes defined by the last two values parallel to the specified plane, with vertices given by the three coordinate pairs.
- `arc = ArcX(x1, x2, r1, r2, a1, a2)`
  Create an arc shape.
- `box = Box(x1, y1, z1, x2, y2, z2)`
  Creates a rectangular prism with opposite corners at the specified positions and edges parallel to the XYZ axes.
- `box = BoxCentered(xw, yw, zw)`
  Creates an xw by yw by zw box centred on the origin.
- `box = BoxCenteredXY(xw, yw, z)`
  Creates a box of width xw / yw and height z.

Composite parts may be created by **assembling** these primitives either at creation time or subsequently using `Collection()`:

```
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 Moving Parts

Parts may need to be moved in the Vismach space to assemble the model. They may also need to be moved to create the animation as the animation rotation axis is created at the origin (but moves with the Part):

- `part1 = Translate([part1], x, y, z)`
  Move part1 the specified distances in x, y and z.
- `part1 = Rotate([part1], theta, x, y, z)`
  Rotate the part by angle theta about an axis between the origin and x, y, z.

### 13.5.5 Animating Parts

To animate the model (controlled by the values of HAL pins) there are two functions *HalTranslate* and *HalRotate*. 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:

- `part = HalTranslate([part], comp, ”hal_pin”, xs, ys, zs)`
  The function arguments are:
  - first a *collection/part* which can be pre-created earlier in the script, or could be created at this point if preferred eg `part1 = HalTranslate([Box(....)], ...)`.
  - The *HAL component* is the next argument, i.e. the object returned by the `comp = hal.component(...)` command. After that is the name of the HAL in that will animate the motion, this needs to match an existing HAL pin that is part of the HAL component created earlier in the script.

  - Then follow the *X, Y, Z scales*.
    For a Cartesian machine created at 1:1 scale this would typically be 1,0,0 for a motion in the positive X direction.
    However if the STL file happened to be in cm and the machine was in inches, this could be fixed at this point by using 0.3937 (1cm /2.54in) as the scale.

- `part = HalRotate([part], 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.

  - The *axis of rotation* is from the origin point to the point defined by (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.
  - *Rotation angles* are in degrees, so for a rotary joint with a 0-1 scaling you would need to use an angle scale of 360.

### 13.5.6 Assembling the model.

In order for parts to move together they need to be assembled with the Collection() command. It is important to assemble the parts and define their motions in the correct sequence. For example to create a moving head milling machine with a rotating spindle and an animated draw bar you would:

- Create the head main body.
- Create the spindle at the origin.
- Define the rotation.
- Move the head to the spindle or spindle to the head.
- Create the draw bar.
- Define the motion of the draw bar.
- Assemble the three parts into a head assembly.
- Define the motion of the head assembly.

In this example the spindle rotation is indicated by rotation of a set of drive dogs:

```
#Drive dogs
dogs = Box(-6,-3,94,6,3,100)
dogs = Color([1,1,1,1],[dogs])
dogs = HalRotate([dogs],c,"spindle",360,0,0,1)
dogs = Translate([dogs],-1,49,0)

#Drawbar
draw = CylinderZ(120,3,125,3)
draw = Color([1,0,.5,1],[draw])
draw = Translate([draw],-1,49,0)
draw = HalTranslate([draw],c,"drawbar",0,0,1)

# head/spindle
head = AsciiSTL(filename="./head.stl")
head = Color([0.3,0.3,0.3,1],[head])
head = Translate([head],0,0,4)
head = Collection([head, tool, dogs, draw])
head = HalTranslate([head],c,"Z",0,0,0.1)

# base
```

```
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# mount head on it
base = Collection([head, base])
```

Finally a single collection of all the machine parts, floor and work (if any) needs to be created:

- For a *serial machine* each new part will be added to the collection of the previous part.
- For a *parallel machine* there may be several "base" parts.

Thus, for example, in scaragui.py link3 is added to link2, link2 to link1 and link1 to link0, so the final model is created by:

```
model = Collection([link0, floor, table])
```

Whereas a VMC model with separate parts moving on the base might have:

```
model = Collection([base, saddle, head, carousel])
```

### 13.5.7 Other functions

- `part = Color([colorspec], [part])`
  Sets the display color of the part. Note that unlike the other functions the part definition comes second in this case.
  The colorspec consists of the three RGB values and an opacity. For example [1,0,0,0.5] for a 50% opacity red.
- `myhud = Hud()`
  Creates a heads-up display in the Vismach GUI to display such items as axis positions.
- `tooltip = Capture()`
  Think of this as an invisible part that needs to be attached to the tooltip to track the position and orientation of the tool coordinate system. It is actually a transformation matrix that is constantly updated as the model moves.
- `work = Capture()`
  Same as above but attached to the work table to track the work coordinate system.
- `main(model, tooltip, work, size=10, hud=0, rotation_vectors=None, lat=0, lon=0)`
  This is the command that makes it all happen, creates the display etc.
  - *model* should be a collection that contains all the machine parts.
  - *tooltip* and *work* need to be created by `Capture()`. Vismach needs this information to draw the backplot which is basically the tooltip position drawn in the work coordinate system.
    See `scaragui.py` for an example of how to connect the tool tip to a tool and the tool to the model.
  - Either *rotation_vectors* or *latitude/longitude* can be used to set the original viewpoint and it is advisable to do as the default initial viewpoint is rather unhelpfully from immediately overhead.
  - *size* sets the extent of the volume visualized in the initial view.
  - *hud* refers to a head-up display of axis positions.

### 13.5.8 Basic structure of a Vismach script.

```
#imports
from vismach import *
import hal
#create the HAL component and pins
comp = hal.component("compname")
comp.newpin("pin_name", hal.HAL_FLOAT, hal.HAL_IN)
...
#create the floor, tool and work
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()
...
#Build and assemble the model
part1 = Collection([Box(-6,-3,94,6,3,100)])
part1 = Color([1,1,1,1],[part1])
part1 = HalRotate([part1],comp,"pin_name",360,0,0,1)
part1 = Translate([dogs],-1,49,0)
...
#create a top-level model
model = Collection([base, saddle, head, carousel])
#Start the visualization
main(model, tooltip, work, 100, lat=-75, lon=215)
```

# Part III

# Глоссарий, авторское право и история

# Chapter 14

# С обратной стороны

Работа над этим справочником продолжается. Если вы можете помочь с написанием, редактированием или подготовкой графики, пожалуйста, свяжитесь с любым членом команды авторов или присоединитесь и отправьте электронное письмо по адресу emc-users@lists.sourceforge.net.

Copyright © 2000-2025 LinuxCNC.org

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 no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

Если вы не найдете лицензию, вы можете заказать копию по адресу:

```
Free Software Foundation, Inc.
51 Franklin Street
Fifth Floor
Boston, MA 02110-1301 USA.
```

(Версия на английском языке является официальной)

LINUX® является зарегистрированным товарным знаком Линуса Торвальдса в США и других странах. Зарегистрированный товарный знак Linux® используется в соответствии с сублицензией от LMI, эксклюзивного лицензиата Линуса Торвальдса, владельца товарного знака во всем мире.

Проект LinuxCNC не связан с Debian®. *Debian* является зарегистрированным товарным знаком, принадлежащим Software in the Public Interest, Inc.

Проект LinuxCNC не связан с UBUNTU®. *UBUNTU* — зарегистрированная торговая марка, принадлежащая Canonical Limited.

# Chapter 15

# Glossary

A listing of terms and what they mean. Some terms have a general meaning and several additional meanings for users, installers, and developers.

**Acme Screw**
A type of lead-screw that uses an Acme thread form. Acme threads have somewhat lower friction and wear than simple triangular threads, but ball-screws are lower yet. Most manual machine tools use acme lead-screws.

**Axis**
One of the computer controlled movable parts of the machine. For a typical vertical mill, the table is the X axis, the saddle is the Y axis, and the quill or knee is the Z axis. Angular axes like rotary tables are referred to as A, B, and C. Additional linear axes relative to the tool are called U, V, and W respectively.

**AXIS(ГИП)**
One of the Graphical User Interfaces available to users of LinuxCNC. It features the modern use of menus and mouse buttons while automating and hiding some of the more traditional LinuxCNC controls. It is the only open-source interface that displays the entire tool path as soon as a file is opened.

**GMOCCAPY (GUI)**
A Graphical User Interfaces available to users of LinuxCNC. It features the use and feel of an industrial control and can be used with touch screen, mouse and keyboard. It support embedded tabs and hal driven user messages, it offers a lot of hal beens to be controlled with hardware. GMOCCAPY is highly customizable.

**Backlash**
The amount of ”play” or lost motion that occurs when direction is reversed in a lead screw. or other mechanical motion driving system. It can result from nuts that are loose on leadscrews, slippage in belts, cable slack, ”wind-up” in rotary couplings, and other places where the mechanical system is not ”tight”. Backlash will result in inaccurate motion, or in the case of motion caused by external forces (think cutting tool pulling on the work piece) the result can be broken cutting tools. This can happen because of the sudden increase in chip load on the cutter as the work piece is pulled across the backlash distance by the cutting tool.

**Backlash Compensation**
Any technique that attempts to reduce the effect of backlash without actually removing it from the mechanical system. This is typically done in software in the controller. This can correct the final resting place of the part in motion but fails to solve problems related to direction changes while in motion (think circular interpolation) and motion that is caused when external forces (think cutting tool pulling on the work piece) are the source of the motion.

**Ball Screw**
A type of lead-screw that uses small hardened steel balls between the nut and screw to reduce friction. Ball-screws have very low friction and backlash, but are usually quite expensive.

**Ball Nut**
A special nut designed for use with a ball-screw. It contains an internal passage to re-circulate the balls from one end of the screw to the other.

**CNC**
Computer Numerical Control. The general term used to refer to computer control of machinery. Instead of a human operator turning cranks to move a cutting tool, CNC uses a computer and motors to move the tool, based on a part program.

**Halcompile**
Инструмент, используемый для сборки, компиляции и установки компонентов LinuxCNC HAL.

**Configuration(n)**
A directory containing a set of configuration files. Custom configurations are normally saved in the users home/linuxcnc/configs directory. These files include LinuxCNC's traditional INI file and HAL files. A configuration may also contain several general files that describe tools, parameters, and NML connections.

**Configuration(v)**
The task of setting up LinuxCNC so that it matches the hardware on a machine tool.

**Coordinate Measuring Machine**
A Coordinate Measuring Machine is used to make many accurate measurements on parts. These machines can be used to create CAD data for parts where no drawings can be found, when a hand-made prototype needs to be digitized for moldmaking, or to check the accuracy of machined or molded parts.

**Display units**
The linear and angular units used for onscreen display.

**DRO**
A Digital Read Out is a system of position-measuring devices attached to the slides of a machine tool, which are connected to a numeric display showing the current location of the tool with respect to some reference position. DROs are very popular on hand-operated machine tools because they measure the true tool position without backlash, even if the machine has very loose Acme screws. Some DROs use linear quadrature encoders to pick up position information from the machine, and some use methods similar to a resolver which keeps rolling over.

**EDM**
EDM is a method of removing metal in hard or difficult to machine or tough metals, or where rotating tools would not be able to produce the desired shape in a cost-effective manner. An excellent example is rectangular punch dies, where sharp internal corners are desired. Milling operations can not give sharp internal corners with finite diameter tools. A *wire* EDM machine can make internal corners with a radius only slightly larger than the wire's radius. A *sinker* EDM can make internal corners with a radius only slightly larger than the radius on the corner of the sinking electrode.

**EMC**
The Enhanced Machine Controller. Initially a NIST project. Renamed to LinuxCNC in 2012.

**EMCIO**
The module within LinuxCNC that handles general purpose I/O, unrelated to the actual motion of the axes.

**EMCMOT**
The module within LinuxCNC that handles the actual motion of the cutting tool. It runs as a real-time program and directly controls the motors.

**Энкодер**
Устройство для измерения положения. Обычно это оптико-механическое устройство, выдающее квадратурный сигнал. Сигнал может считаться специальным оборудованием или напрямую parport с LinuxCNC.

**Feed**
Relatively slow, controlled motion of the tool used when making a cut.

**Feed rate**
The speed at which a cutting motion occurs. In auto or MDI mode, feed rate is commanded using an F word. F10 would mean ten machine units per minute.

**Feedback**
Метод (например, сигналы квадратурного энкодера), с помощью которого LinuxCNC получает информацию о положении двигателей.

**Feedrate Override**
A manual, operator controlled change in the rate at which the tool moves while cutting. Often used to allow the operator to adjust for tools that are a little dull, or anything else that requires the feed rate to be ”tweaked”.

**Floating Point Number**
A number that has a decimal point. (12.300) In HAL it is known as float.

**G-code**
The generic term used to refer to the most common part programming language. There are several dialects of G-code, LinuxCNC uses RS274/NGC.

**GUI**
Графический Интерфейс Пользователя.

**General**
A type of interface that allows communications between a computer and a human (in most cases) via the manipulation of icons and other elements (widgets) on a computer screen.

**LinuxCNC**
An application that presents a graphical screen to the machine operator allowing manipulation of the machine and the corresponding controlling program.

**HAL**
Уровень аппаратной абстракции. На самом высоком уровне это просто способ загрузки и соединения нескольких строительных блоков для сборки сложной системы. Многие из этих блоков представляют собой драйверы для аппаратных устройств. Однако HAL может делать больше, чем просто настраивать драйверы оборудования.

**Home**
A specific location in the machine’s work envelope that is used to make sure the computer and the actual machine both agree on the tool position.

**INI file**
A text file that contains most of the information that configures LinuxCNC for a particular machine.

**Instance**
One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the ”Lassie” object is an instance of the ”Dog” class.

**Joint Coordinates**
These specify the angles between the individual joints of the machine. See also Kinematics

**Jog**
Manually moving an axis of a machine. Jogging either moves the axis a fixed amount for each key-press, or moves the axis at a constant speed as long as you hold down the key. In manual mode, jog speed can be set from the graphical interface.

**kernel-space**
Код, выполняемый внутри ядра, в отличие от кода, выполняемого в пользовательском пространств
Некоторые системы реального времени (например, RTAI) выполняют код реального времени в ядре, а код не в реальном времени — в пользовательском пространстве, в то время как другие системы реального времени (например, Preempt-RT) выполняют код как в реальном времени, так и не в реальном времени в пользовательском пространстве.

**Kinematics**
The position relationship between world coordinates and joint coordinates of a machine. There are two types of kinematics. Forward kinematics is used to calculate world coordinates from joint coordinates. Inverse kinematics is used for exactly the opposite purpose. Note that kinematics does not take into account, the forces, moments etc. on the machine. It is for positioning only.

**Lead-screw**
An screw that is rotated by a motor to move a table or other part of a machine. Lead-screws are usually either ball-screws or acme screws, although conventional triangular threaded screws may be used where accuracy and long life are not as important as low cost.

**Machine units**
Линейные и угловые единицы, используемые для конфигурации станка. Эти единицы указаны и используются в INI-файле. Выводы и параметры HAL также обычно выражаются в единицах станка.

**MDI**
Manual Data Input. This is a mode of operation where the controller executes single lines of G-code as they are typed by the operator.

**NIST**
National Institute of Standards and Technology. An agency of the Department of Commerce in the United States.

**NML**
Neutral Message Language provides a mechanism for handling multiple types of messages in the same buffer as well as simplifying the interface for encoding and decoding buffers in neutral format and the configuration mechanism.

**Offsets**
An arbitrary amount, added to the value of something to make it equal to some desired value. For example, G-code programs are often written around some convenient point, such as X0, Y0. Fixture offsets can be used to shift the actual execution point of that G-code program to properly fit the true location of the vice and jaws. Tool offsets can be used to shift the ”uncorrected” length of a tool to equal that tool’s actual length.

**Part Program**
A description of a part, in a language that the controller can understand. For LinuxCNC, that language is RS-274/NGC, commonly known as G-code.

**Program Units**
The linear and angular units used in a part program. The linear program units do not have to be the same as the linear machine units. See G20 and G21 for more information. The angular program units are always measured in degrees.

**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.

**Rapid**
Fast, possibly less precise motion of the tool, commonly used to move between cuts. If the tool meets the workpiece or the fixturing during a rapid, it is probably a bad thing!

**Rapid rate**
The speed at which a rapid motion occurs. In auto or MDI mode, rapid rate is usually the maximum speed of the machine. It is often desirable to limit the rapid rate when testing a G-code program for the first time.

**Real-time**
Программное обеспечение, предназначенное для соблюдения очень строгих временных условий. В Linux для удовлетворения этих требований необходимо установить ядро реального времени, такое как RTAI или Preempt-RT, и собрать программное обеспечение LinuxCNC для работы в специальной среде реального времени. Программное обеспечение реального времени может работать в пространствах ядра или пользователя в зависимости от возможностей, предлагаемых системой.

**RTAI**
Real Time Application Interface, see https://www.rtai.org/, the real-time extensions for Linux that LinuxCNC can use to achieve real-time performance.

**RTLINUX**
See https://en.wikipedia.org/wiki/RTLinux, an older real-time extension for Linux that LinuxCNC used to use to achieve real-time performance. Obsolete, replaced by RTAI.

**RTAPI**
A portable interface to real-time operating systems including RTAI and POSIX pthreads with realtime extensions.

**RS-274/NGC**
The formal name for the language used by LinuxCNC part programs.

**Servo Motor**
Generally, any motor that is used with error-sensing feedback to correct the position of an actuator. Also, a motor which is specially-designed to provide improved performance in such applications.

**Servo Loop**
A control loop used to control position or velocity of an motor equipped with a feedback device.

**Signed Integer**
A whole number that can have a positive or negative sign. In HAL it is usually a s32, but could be also a s64.

**Spindle**
Часть станка, которая вращается для резания. На фрезе или дрели шпиндель удерживает режущий инструмент. На токарном станке шпиндель удерживает заготовку.

**Spindle Speed Override**
A manual, operator controlled change in the rate at which the tool rotates while cutting. Often used to allow the operator to adjust for chatter caused by the cutter's teeth. Spindle Speed Override assumes that the LinuxCNC software has been configured to control spindle speed.

**StepConf**
An LinuxCNC configuration wizard. It is able to handle many step-and-direction motion command based machines. It writes a full configuration after the user answers a few questions about the computer and machine that LinuxCNC is to run on.

**Stepper Motor**
A type of motor that turns in fixed steps. By counting steps, it is possible to determine how far the motor has turned. If the load exceeds the torque capability of the motor, it will skip one or more steps, causing position errors.

**TASK**
The module within LinuxCNC that coordinates the overall execution and interprets the part program.

**Tcl/Tk**
A scripting language and graphical widget toolkit with which several of LinuxCNCs GUIs and selection wizards were written.

**Traverse Move**
A move in a straight line from the start point to the end point.

**Units**
See ”Machine Units”, ”Display Units”, or ”Program Units”.

**Unsigned Integer**
A whole number that has no sign. In HAL it is usually a u32 but could be also a u64.

**World Coordinates**
This is the absolute frame of reference. It gives coordinates in terms of a fixed reference frame that is attached to some point (generally the base) of the machine tool.

# Chapter 16

# Авторское право

## 16.1 Legal Section

Translations of this file provided in the source tree are not legally binding.

### 16.1.1 Copyright Terms

**Copyright (c) 2000-2022 LinuxCNC.org**

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 no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

### 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. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**0. PREAMBLE**

Цель настоящей лицензии — сделать руководство, учебник или другой письменный документ ”свободным” в смысле свободы: гарантировать каждому эффективную свободу копировать и распространять его с модификацией или без нее, как в коммерческих, так и в некоммерческих целях. Во-вторых, настоящая лицензия оставляет за автором и издателем возможность получить признание за свою работу, не неся при этом ответственности за изменения, внесенные другими.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

**1. APPLICABILITY AND DEFINITIONS**

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

«Второй раздел» — это поименованное приложение или основной раздел Документа, который касается исключительно отношений издателей или авторов Документа к общей теме Документа (или к связанным с ней вопросам) и не содержит ничего, что могло бы напрямую относиться к в рамках этой общей темы. (Например, если Документ частично является учебником по математике, Второй раздел не может объяснять какую-либо математику.) Отношения могут быть вопросом исторической связи с предметом или связанными с ним вопросами, а также юридическими, коммерческими, философскими, этическая или политическая позиция по отношению к ним.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

”Прозрачная” копия Документа означает машиночитаемую копию, представленную в формате, спецификация которого доступна широкой публике, содержание которого может быть просмотрено и отредактировано прямо и понятно с помощью обычных текстовых редакторов или (для изображений, состоящих из пикселей) общедоступных программ рисования или (для рисунков) какой-либо широко доступный редактор рисунков, который подходит для ввода в форматировщики текста или для автоматического перевода в различные форматы, подходящие для ввода в форматировщики текста. Копия, сделанная в формате файла ”Прозрачный”, разметка которого была разработана для предотвращения или избегания последующего изменения читателями, не является ”Прозрачной” Копия, которая не является ”Прозрачной”, называется ”Непрозрачной”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

**2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

**3. COPYING IN QUANTITY**

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add

other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

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.

Просьба, но не обязательно, чтобы вы связались с авторами Документа задолго до распространения большого количества копий, чтобы дать им возможность предоставить вам обновленную версию Документа.

**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. Используйте на титульном листе (и на обложках, если таковые имеются) заголовок, отличный от названия Документа и предыдущих версий (которые должны, если таковые имеются, быть указаны в разделе «История» Документа) . Вы можете использовать то же название, что и в предыдущей версии, если первоначальный издатель этой версии дает разрешение. Б. Укажите на титульном листе в качестве авторов одно или несколько физических или юридических лиц, ответственных за авторство изменений в Модифицированной версии, а также не менее пяти основных авторов Документа (всех его основных авторов, если он имеет менее пяти). C. Укажите на титульном листе имя издателя Модифицированной версии в качестве издателя. D. Сохраните все уведомления об авторских правах на Документ. E. Добавьте соответствующее уведомление об авторских правах для ваших изменений рядом с другими уведомлениями об авторских правах. F. Включите сразу после уведомлений об авторских правах уведомление о лицензии, дающее публичное разрешение на использование Модифицированной версии в соответствии с условиями настоящей Лицензии, в форме, показанной в Приложении ниже. G. Сохраните в этом уведомлении о лицензии полные списки Неизменяемых разделов и обязательных сопроводительных текстов, приведенных в уведомлении о лицензии к Документу. H. Включите неизмененную копию настоящей Лицензии. I. Сохраните раздел, озаглавленный «История», и его название, и добавьте к нему пункт, указывающий как минимум название, год, новых авторов и издателя Модифицированной версии, как указано на титульном листе. Если в Документе нет раздела под названием ”История”, создайте его, указав название, год, авторов и издателя Документа, как указано на его титульной странице, затем добавьте элемент, описывающий Модифицированную версию, как указано в предыдущем предложении. J. Сохраните сетевое местоположение, если таковое имеется, указанное в Документе, для публичного доступа к прозрачной копии Документа, а также сетевые местоположени указанные в Документе для предыдущих версий, на которых он был основан. Их можно разместить в разделе ”История”. Вы можете опустить сетевое местоположение для произведения, которое было опубликовано не менее чем за четыре года до самого Документа, или если первоначальный издатель версии, на которую оно ссылается, дает разрешение. K. В любом разделе, озаглавленном ”Благодарности” или ”Посвящения”, сохраните название раздела и сохраните в нем всю суть и тон каждого из приведенных в нем благодарностей и/или посвящений участников. L. Сохраните все неизменяемые разделы Документа без изменений в их тексте и названиях. Номера разделов или их эквиваленты не считаются частью названий разделов. M. Удалить любой раздел, озаглавленны ”Подтверждения”. Такой раздел не может быть включен в Модифицированную версию. N. Не

переименовывайте существующие разделы как ”Подтверждения” и не конфликтет по названию с каким-либо неизменным разделом.

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.

# Chapter 17

# История LinuxCNC

## 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.

NIST first became interested in writing a motion control package as a test platform for concepts and standards. Early sponsorship from General Motors resulted in an adaptation of the fledgling version of EMC using PMAC intelligent control boards running under a ”real time” version of Windows NT and controlling a large milling machine.

As is required of all *work product* of US federal government employees, the resulting software and the report about it are required to be in the public domain and a report about it was duly published, including on the Internet. It was there that Matt Shaver discovered EMC. He contacted NIST and entered into discussions with Fred Proctor about adapting the code for use in controlling less expensive hardware to be used for upgrades and replacements of CNC controls that were obsolete or just plain dead. NIST was intrigued because they too wanted something less expensive. In order to launch a cooperative effort, a formal agreement was created which guaranteed that the resulting code and design would remain in the public domain.

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.

NIST set up a mailing list for people interested in EMC. As time went on, others outside NIST became interested in improving EMC. Many people requested or coded small improvements to the code. Ray Henry wanted to refine the user interface. Since Ray was reluctant to try tampering with the C code in which the user interface was written, a simpler method was sought. Fred Proctor of NIST suggested a scripting language and wrote code to interface the Tcl/Tk scripting language to the internal NML communications of EMC. With this tool Ray went on to write a Tcl/Tk program that became the predominant user interface for EMC at the time.

For NIST’s perspective, see this paper written by William Shackleford and Frederick Proctor, describing the history of EMC and its transition to open source.

К этому времени интерес к EMC начал существенно возрастать. По мере того как все больше и больше людей пытались установить EMC, сложность внесения в ядро Linux расширений реального времени и компиляции кода EMC становилась совершенно очевидной. Было предпринято множество попыток документировать процесс и написать сценарии, некоторые из них имели умеренный успех. Постоянно возникала проблема соответствия правильной версии патчей и компиляторов выбранной версии Linux. Пол Корнер пришел на помощь с BDI (brain dead install), который представлял собой компакт-диск, с которого можно было установить полную рабочую систему (Linux, патчи и EMC). Подход BDI открыл мир EMC гораздо более широкому сообществу пользователей Поскольку это сообщество продолжало расти, список рассылки EMC и архивы кода были перенесены на SourceForge и был создан веб-сайт LinuxCNC.

With a larger community of users participating, EMC became a major focus of interest at the on-going CNC exhibits at NAMES and NAMES became the annual meeting event for EMC. For the first couple of years, the meetings just happened because the interested parties were at NAMES. In 2003 the EMC user community had its first announced public meeting. It was held the Monday after NAMES in the lobby of the arena where the NAMES show was held. Organization was loose, but the idea of a hardware abstraction layer (HAL) was born and the movement to restructure the code for ease of development (EMC2) was proposed.

### 17.1.1 Name Change

In the spring of 2011, the LinuxCNC Board of Directors was contacted by a law firm representing EMC Corporation (www.emc.com) about the use of ”EMC” and ”EMC2” to identify the software offered on linuxcnc.org. EMC Corporation has registered various trademarks relating to EMC and EMC² (EMC with superscripted numeral two).

After a number of conversations with the representative of EMC Corporation, the final result is that, starting with the next major release of the software, linuxcnc.org will stop identifying the software using ”emc” or ”EMC”, or those terms followed by digits. To the extent that the LinuxCNC Board of Directors controls the names used to identify the software offered on linuxcnc.org, the board has agreed to this.

As a result, it was necessary to choose a new name for the software. Of the options the board considered, there was consensus that ”LinuxCNC” is the best option, as this has been our website’s name for years.

In preparation for the new name, we have received a sub-license of the LINUX® trademark from the Linux Foundation (www.linuxfoundation.org), protecting our use of the LinuxCNC name. (LINUX® is the registered trademark of Linus Torvalds in the U.S. and other countries.)

The rebranding effort included the linuxcnc.org website, the IRC channels, and versions of the software and documentation since version 2.5.0.

### 17.1.2 Additional Info

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