LinuxCNC V2.10.0-pre0-5839-g6cb44daad3

Contents

I Початок роботи і конфігурація 1

1 Початок роботи з LinuxCNC 2

1.1 Про LinuxCNC . . . 2
1.1.1 Architecture - Context diagram . . . 3
1.1.2 Операційна система . . . 4
1.1.3 Отримання допомоги . . . 4
1.1.3.1 Веб-форум . . . 4
1.1.3.2 IRC . . . 5
1.1.3.3 Список розсилки . . . 5
1.1.3.4 Веб-форум . . . 5
1.1.3.5 Вікі LinuxCNC . . . 5
1.1.3.6 Звіти про помилки . . . 6
1.2 Системні вимоги . . . 6
1.2.1 Мінімальні вимоги . . . 6
1.2.2 Вимоги до ядра та версії . . . 6
1.2.2.1 Preempt-RT з пакетом *linuxcnc-uspace* . . . 7
1.2.2.2 RTAI з пакетом «linuxcnc» . . . 7
1.2.2.3 Xenomai з пакетом *linuxcnc-uspace* . . . 7
1.2.2.4 RTAI з пакетом *linuxcnc-uspace* . . . 7
1.2.3 Проблемне обладнання . . . 7
1.2.3.1 Ноутбуки . . . 7
1.2.3.2 Відеокарти . . . 7
1.3 Отримання LinuxCNC . . . 8
1.3.1 Завантажте зображення . . . 8
1.3.1.1 Звичайне завантаження . . . 8
1.3.1.2 Завантажити за допомогою zsync . . . 9
1.3.1.3 Перевірте зображення . . . 9
1.3.2 Записати образ на завантажувальний пристрій . . . 9

1.3.2.1 Зображення Raspberry Pi . . . 10
1.3.2.2 Образ AMD-64 (x86-64, ПК) з використанням інструментів графічного інтерфейсу . . . 10
1.3.2.3 Командний рядок - Linux . . . 10
1.3.2.4 Командний рядок - MacOS . . . 10
1.3.3 Тестування LinuxCNC . . . 11
1.3.4 Встановлення LinuxCNC . . . 11
1.3.5 Оновлення для LinuxCNC . . . 12
1.3.6 Проблеми з встановленням . . . 12
1.3.7 Альтернативні методи встановлення . . . 12
1.3.7.1 Встановлення на Debian Trixie (з ядром Preempt-RT) . . . 13
1.3.7.2 Встановлення на Debian Trixie (з експериментальним ядром RTAI) . . . 14
1.3.7.3 Встановлення на Raspbian 12 . . . 14
1.4 Запуск LinuxCNC . . . 14
1.4.1 Виклик LinuxCNC . . . 14
1.4.2 Запуск конфігурації . . . 15
1.4.3 Наступні кроки в налаштуванні . . . 17
1.4.4 Конфігурації симулятора . . . 17
1.4.5 Ресурси конфігурації . . . 18
1.5 Оновлення LinuxCNC . . . 18
1.5.1 Оновіться до нової версії . . . 18
1.5.1.1 Конфігурація Apt Sources . . . 19
1.5.1.2 Оновлення до нової версії . . . 21
1.5.1.3 Ubuntu . . . 22
1.5.2 Оновлення без мережі . . . 22
1.5.3 Оновлення файлів конфігурації для версії 2.9 . . . 22
1.5.3.1 Суворіша обробка підключаємих інтерпретаторів . . . 22
1.5.3.2 Кантерп . . . 23
1.5.3.3 Обмеження шпинделя в INI . . . 23
1.5.4 Оновлення файлів конфігурації для версії 2.10.y . . . 23
1.5.5 Нові компоненти HAL . . . 23
1.5.5.1 Не в реальному часі . . . 23
1.5.5.2 У режимі реального часу . . . 23
1.5.6 Нові водії . . . 24
1.6 Найчастіші запитання щодо Linux . . . 24
1.6.1 Автоматичний вхід . . . 24
1.6.1.1 Debian . . . 24
1.6.1.2 Ubuntu . . . 24
1.6.2 Автоматичний запуск . . . 25

1.6.3 Термінал . . . 25
1.6.4 Сторінки користувача . . . 25
1.6.5 Список модулів . . . 25
1.6.6 Редагування кореневого файлу . . . 26
1.6.6.1 Спосіб командного рядка . . . 26
1.6.6.2 Шлях графічного інтерфейсу . . . 26
1.6.6.3 Root-доступ . . . 26
1.6.7 Команди терміналу . . . 26
1.6.7.1 Робочий каталог . . . 26
1.6.7.2 Зміна каталогів . . . 26
1.6.7.3 Перелік файлів у каталозі . . . 27
1.6.7.4 Пошук файлу . . . 27
1.6.7.5 Пошук тексту . . . 27
1.6.7.6 Діагностичні повідомлення . . . 28
1.6.8 Предмети зручності . . . 28
1.6.8.1 Запуск термінала . . . 28
1.6.9 Проблеми з обладнанням . . . 28
1.6.9.1 Інформація про обладнання . . . 28
1.6.9.2 Роздільна здатність монітора . . . 28
1.6.10 Шляхи . . . 29

2 Загальна користувацька інформація **30**
2.1 Передмова користувача . . . 30
2.2 Вступ для користувача LinuxCNC . . . 31
2.2.1 Вступ . . . 31
2.2.2 Як працює LinuxCNC . . . 31
2.2.3 Графічні інтерфейси користувача . . . 33
2.2.4 Інтерфейс користувача . . . 41
2.2.5 Віртуальна панель керування . . . 41
2.2.6 Мови . . . 44
2.2.7 Думай як оператор CNC . . . 44
2.2.8 Режими роботи . . . 45
2.3 Важливі концепції користувача . . . 45
2.3.1 Контроль траєкторії . . . 45
2.3.1.1 Планування траєкторії . . . 45
2.3.1.2 Слідування шляхом . . . 46
2.3.1.3 Програмування Планувальника . . . 46
2.3.1.4 Планування переїздів . . . 47
2.3.2 G-код . . . 48

2.3.2.1 Значення за замовчуванням . . . 48
2.3.2.2 Швидкість подачі . . . 48
2.3.2.3 Зміщення радіуса інструмента . . . 48
2.3.3 Самонаведення . . . 48
2.3.4 Зміни інструментів . . . 49
2.3.5 Системи координат . . . 49
2.3.5.1 G53 Координати машини . . . 49
2.3.5.2 G54-59.3 Координати користувача . . . 49
2.3.5.3 Коли ти загубився . . . 49
2.3.6 Конфігурації машини . . . 50
2.4 Запуск LinuxCNC . . . 51
2.4.1 Запуск LinuxCNC . . . 51
2.4.1.1 Вибір конфігурації . . . 53
2.5 Огляд верстата з CNC . . . 53
2.5.1 Механічні компоненти . . . 53
2.5.1.1 Сокири . . . 54
2.5.1.2 Шпиндель . . . 54
2.5.1.3 Охолоджувальна рідина . . . 54
2.5.1.4 Коригування подачі та швидкості . . . 54
2.5.1.5 Перемикач видалення блоку . . . 54
2.5.1.6 Додатковий перемикач зупинки програми . . . 55
2.5.2 Компоненти керування та даних . . . 55
2.5.2.1 Лінійні осі . . . 55
2.5.2.2 Осі обертання . . . 55
2.5.2.3 Контрольована точка . . . 55
2.5.2.4 Координований лінійний рух . . . 55
2.5.2.5 Швидкість подачі . . . 56
2.5.2.6 Охолодження . . . 56
2.5.2.7 Залишатися . . . 56
2.5.2.8 Одиниці . . . 56
2.5.2.9 Поточна позиція . . . 57
2.5.2.10 Вибрана площина . . . 57
2.5.2.11 Карусель інструментів . . . 57
2.5.2.12 Зміна інструменту . . . 57
2.5.2.13 Трансфер з піддонів . . . 57
2.5.2.14 Перевизначення швидкості . . . 57
2.5.2.15 Режим керування шляхом . . . 57
2.5.3 Взаємодія перекладача з перемикачами . . . 58
2.5.3.1 Перемикачі подачі та швидкості . . . 58

2.5.3.2 Перемикач видалення блоку . . . 58
2.5.3.3 Додатковий перемикач зупинки програми . . . 58
2.5.4 Таблиця інструментів . . . 58
2.5.5 Параметри . . . 59
2.6 Інформація для користувача токарного верстата . . . 59
2.6.1 Режим токарного верстата . . . 59
2.6.2 Стіл токарних інструментів . . . 60
2.6.3 Орієнтація токарного інструменту . . . 60
2.6.4 Інструмент дотику вимкнено . . . 62
2.6.4.1 X Touch Off . . . 62
2.6.4.2 Z Touch Off . . . 63
2.6.4.3 Зсув машини Z . . . 63
2.6.5 Синхронізований рух шпинделя . . . 64
2.6.6 Дуги . . . 64
2.6.6.1 Дуги та конструкція токарного верстата . . . 64
2.6.6.2 Режим радіуса та діаметра . . . 64
2.6.7 Шлях інструменту . . . 65
2.6.7.1 Контрольна точка . . . 65
2.6.7.2 Кути різання без різального компаса . . . 65
2.6.7.3 Вирізання радіуса . . . 67
2.6.7.4 Використання компенсації різака . . . 69
2.7 Грунтовка для плазмового різання для користувачів LinuxCNC . . . 69
2.7.1 Що таке плазма? . . . 69
2.7.2 Ініціалізація дуги . . . 70
2.7.2.1 Високочастотний пуск . . . 70
2.7.2.2 Початок віддачі . . . 71
2.7.3 Плазмовий CNC . . . 71
2.7.4 Вибір плазмового верстата для операцій з CNC . . . 73
2.7.5 Типи контролю висоти пальника . . . 73
2.7.6 Сигнал «Дуга в порядку» . . . 74
2.7.7 Початкове вимірювання висоти . . . 74
2.7.7.1 Поплавкові вимикачі . . . 75
2.7.7.2 Омічне зондування . . . 75
2.7.7.3 Гіперсенсування за допомогою MESA THCAD-5 . . . 76
2.7.7.4 Приклад HAL-коду для гіперсенсорики . . . 77
2.7.8 Затримка THC . . . 78
2.7.9 Вимірювання напруги пальника . . . 78
2.7.10 Відрив факела . . . 79
2.7.11 Кутовий замок / Захист від швидкості . . . 79

2.7.12 Перетин порожнечі/виїмки . . . 79
2.7.13 Різання отворів та малих форм . . . 80
2.7.14 Контакти вводу/виводу для плазмових контролерів . . . 80
2.7.14.1 Дуга в порядку (вхід) . . . 81
2.7.14.2 Паяльна лампа увімкнена (вихід) . . . 81
2.7.14.3 Поплавковий вимикач (вхід) . . . 81
2.7.14.4 Увімкнення омічного датчика (вихід) . . . 82
2.7.14.5 Омічний зонд (вхід) . . . 82
2.7.14.6 Датчик відриву пальника . . . 82
2.7.15 G-код для плазмових контролерів . . . 82
2.7.15.1 Увімкнути/вимкнути роботу THC: . . . 83
2.7.16 Зовнішнє зміщення та плазмове різання . . . 83
2.7.17 Зчитування напруги дуги за допомогою Mesa THCAD . . . 84
2.7.17.1 З'єднання THCAD . . . 85
2.7.17.2 Початкове тестування THCAD . . . 85
2.7.17.3 Яку модель THCAD використовувати? . . . 85
2.7.18 Постпроцесори та вкладення . . . 86
2.7.19 Проектування для шумних електричних середовищ . . . 86
2.7.20 Грунтові води . . . 87
2.7.21 Столи з нижнім відводом повітря . . . 87
2.7.22 Проектування для швидкості та прискорення . . . 88
2.7.23 Відстань, пройдена за оберт двигуна . . . 88
2.7.24 Конфігурація плазми QtPlasmaC LinuxCNC . . . 88
2.7.25 Контролер Hypertherm RS485 . . . 88
2.7.26 Постпроцесори для плазмового різання . . . 89

3 Майстри налаштування 90
3.1 Майстер налаштування крокового двигуна . . . 90
3.1.1 Вступ . . . 90
3.1.2 Початкова сторінка . . . 91
3.1.3 Основна інформація . . . 92
3.1.4 Налаштування паралельного порту . . . 94
3.1.5 Налаштування паралельного порту 2 . . . 96
3.1.6 Конфігурація осі . . . 97
3.1.6.1 Знаходження максимальної швидкості . . . 99
3.1.6.2 Знаходження максимального прискорення . . . 100
3.1.7 Конфігурація шпинделя . . . 101
3.1.7.1 Контроль швидкості шпинделя . . . 101
3.1.7.2 Синхронізований зі шпинделем рух . . . 102

3.1.7.3 Визначення калібрування шпинделя . . . 102
3.1.8 Опції . . . 103
3.1.9 Повна конфігурація машини . . . 104
3.1.10 Axis Подорожі та будинки . . . 104
3.1.10.1 Робота без кінцевих вимикачів . . . 105
3.1.10.2 Робота без домашніх перемикачів . . . 105
3.1.10.3 Варіанти підключення домашнього та кінцевого вимикача . . . 105
3.2 Майстер налаштування Mesa . . . 106
3.2.1 Покрокові інструкції . . . 108
3.2.2 Створити або редагувати . . . 108
3.2.3 Основна інформація про машину . . . 109
3.2.4 Зовнішня конфігурація . . . 111
3.2.5 Конфігурація графічного інтерфейсу . . . 113
3.2.6 Конфігурація Меси . . . 116
3.2.7 Налаштування вводу/виводу Mesa . . . 118
3.2.8 Конфігурація паралельного порту . . . 122
3.2.9 Конфігурація осі . . . 123
3.2.10 Конфігурація шпинделя . . . 132
3.2.11 Розширені параметри . . . 133
3.2.12 Компоненти HAL . . . 134
3.2.13 Розширене використання PnCconf . . . 135

4 Конфігурація 138
4.1 Концепції інтегратора . . . 138
4.1.1 Розташування файлів . . . 138
4.1.1.1 Встановлено . . . 138
4.1.1.2 Командний рядок . . . 139
4.1.2 Файли . . . 139
4.1.3 Крокові системи . . . 139
4.1.3.1 Базовий період . . . 139
4.1.3.2 Час кроку . . . 140
4.1.4 Сервосистеми . . . 140
4.1.4.1 Основні операції . . . 140
4.1.4.2 Пропорційний термін . . . 142
4.1.4.3 Інтегральний член . . . 142
4.1.4.4 Похідний термін . . . 142
4.1.4.5 Налаштування петлі . . . 143
4.1.4.6 Ручне налаштування . . . 143
4.1.5 Планування траєкторії S-подібної кривої . . . 143

4.1.5.1 Увімкнення . . . 143
4.1.5.2 Тюнінг . . . 144
4.1.6 RTAI . . . 144
4.1.6.1 ACPI . . . 144
4.1.7 Апаратні опції інтерфейсу комп’ютер/машина . . . 144
4.1.7.1 litehm2/rv901t . . . 144
4.2 Тестування затримки . . . 145
4.2.1 Що таке латентність? . . . 145
4.2.2 Тести на затримку . . . 145
4.2.2.1 Тест на затримку . . . 145
4.2.2.2 Графік затримки . . . 147
4.2.2.3 Гістограма затримки . . . 148
4.2.3 Налаштування затримки . . . 150
4.2.3.1 Налаштування BIOS для зменшення затримки . . . 150
4.2.3.2 Налаштування Preempt-RT на затримку . . . 151
4.3 Налаштування крокового двигуна . . . 151
4.3.1 Отримання максимальної користі від поетапного програмного забезпечення . 151
4.3.1.1 Виконайте тест затримки . . . 152
4.3.1.2 З’ясуйте, чого очікують ваші накопичувачі . . . 152
4.3.1.3 Виберіть свій БАЗОВИЙ_ПЕРІОД . . . 153
4.3.1.4 Використовуйте steplen, stepspace, dirsetup та/або dirhold . . . 155
4.3.1.5 Без здогадок! . . . 155
4.4 Конфігурація INI . . . 155
4.4.1 Компоненти INI-файлу . . . 155
4.4.1.1 Коментарі . . . 156
4.4.1.2 Розділи . . . 156
4.4.1.3 Змінні . . . 157
4.4.1.4 Розділи та змінні на замовлення . . . 157
4.4.1.5 Включити файли . . . 158
4.4.2 Розділи INI-файлу . . . 159
4.4.2.1 [EMC] Розділ . . . 159
4.4.2.2 [DISPLAY] Розділ . . . 159
4.4.2.3 [FILTER] Розділ . . . 164
4.4.2.4 [RS274NGC] Розділ . . . 165
4.4.2.5 [EMCMOT] Розділ . . . 167
4.4.2.6 [TASK] Розділ . . . 168
4.4.2.7 [HAL] розділ . . . 168
4.4.2.8 [HALUI] розділ . . . 170
4.4.2.9 [APPLICATIONS] Розділ . . . 170

4.4.2.10 [TRAJ] Розділ . . . 171
4.4.2.11 [KINS] Розділ . . . 174
4.4.2.12 [AXIS_<letter>] Розділ . . . 174
4.4.2.13 [JOINT_<num>] Розділи . . . 175
4.4.2.14 [SPINDLE_<num>] Розділ(и)) . . . 182
4.4.2.15 [EMCIO] Розділ . . . 183
4.5 Конфігурація самонаведення . . . 183
4.5.1 Огляд . . . 183
4.5.2 Передумова . . . 184
4.5.3 Приклад розташування окремого домашнього вимикача . . . 185
4.5.4 Приклад схеми спільного граничного/домашнього перемикача . . . 186
4.5.5 Послідовність самонаведення . . . 187
4.5.6 Конфігурація . . . 189
4.5.6.1 HOME_SEARCH_VEL . . . 189
4.5.6.2 HOME_LATCH_VEL . . . 189
4.5.6.3 HOME_FINAL_VEL . . . 190
4.5.6.4 HOME_IGNORE_LIMITS . . . 190
4.5.6.5 HOME_USE_INDEX . . . 190
4.5.6.6 HOME_INDEX_NO_ENCODER_RESET . . . 190
4.5.6.7 HOME_OFFSET . . . 190
4.5.6.8 HOME . . . 191
4.5.6.9 HOME_IS_SHARED . . . 191
4.5.6.10HOME_ABSOLUTE_ENCODER . . . 191
4.5.6.11HOME_SEQUENCE . . . 192
4.5.6.12VOLATILE_HOME . . . 193
4.5.6.13LOCKING_INDEXER . . . 193
4.5.6.14Негайне самонаведення . . . 193
4.5.6.15Запобігання самонаведенню . . . 193
4.6 Конфігурація токарного верстата . . . 195
4.6.1 Площина за замовчуванням . . . 195
4.6.2 Налаштування INI . . . 195
4.7 Швидкий старт для степпера . . . 196
4.7.1 Тест на затримку . . . 196
4.7.2 Шерлайн . . . 196
4.7.3 Ксилотекс . . . 196
4.7.4 Інформація про машину . . . 196
4.7.5 Інформація про розпіновку . . . 197
4.7.6 Механічна інформація . . . 197
4.8 Конфігурація крокового двигуна . . . 199

4.8.1 Вступ . . . 199
4.8.2 Максимальна швидкість кроку . . . 199
4.8.3 Розпіновка . . . 199
4.8.3.1 Стандартна розпіновка HAL . . . 200
4.8.3.2 Огляд . . . 203
4.8.3.3 Зміна standard_pinout.hal . . . 203
4.8.3.4 Зміна полярності сигналу . . . 203
4.8.3.5 Додавання PWM-регулювання швидкості шпинделя . . . 203
4.8.3.6 Додавання сигналу ввімкнення . . . 204
4.8.3.7 Зовнішня кнопка ESTOP . . . 204
4.9 Діагностика крокових двигунів . . . 204
4.9.1 Поширені проблеми . . . 204
4.9.1.1 Крековий рух на один крок . . . 204
4.9.1.2 Без руху степерів . . . 204
4.9.1.3 Відстань неправильна . . . 204
4.9.2 Повідомлення про помилки . . . 205
4.9.2.1 Помилка після . . . 205
4.9.2.2 Помилка RTAPI . . . 205
4.9.3 Тестування . . . 206
4.9.3.1 Час кроку . . . 206
4.10 Фільтрувати програми . . . 207
4.10.1 Вступ . . . 207
4.10.2 Налаштування INI для програмних фільтрів . . . 207
4.10.3 Створення програм фільтрації на базі Python . . . 208

5 HAL (Рівень абстракції апаратного забезпечення) 211
5.1 Вступ до HAL . . . 211
5.1.1 Огляд HAL . . . 211
5.1.2 Зв’язок . . . 213
5.1.3 Проектування системи HAL . . . 215
5.1.3.1 Вибір деталі . . . 216
5.1.3.2 Проектування взаємозв’язків . . . 216
5.1.3.3 Впровадження . . . 216
5.1.3.4 Тестування . . . 216
5.1.3.5 Короткий зміст . . . 217
5.1.4 Концепції HAL . . . 218
5.1.5 Компоненти HAL . . . 219
5.1.6 Проблеми з синхронізацією в HAL . . . 219
5.2 Основи HAL . . . 220

5.2.1 Команди HAL . . . 220
5.2.1.1 loadrt . . . 221
5.2.1.2 addf . . . 221
5.2.1.3 loadusr . . . 222
5.2.1.4 net . . . 223
5.2.1.5 setp . . . 224
5.2.1.6 sets . . . 225
5.2.1.7 unlinkp . . . 225
5.2.1.8 Застарілі команди . . . 225
5.2.2 Дані HAL . . . 226
5.2.2.1 Біт . . . 226
5.2.2.2 Float . . . 226
5.2.2.3 s32 . . . 226
5.2.2.4 u32 . . . 226
5.2.2.5 s64 . . . 226
5.2.2.6 u64 . . . 227
5.2.3 Файли HAL . . . 227
5.2.4 Параметр HAL . . . 227
5.2.5 Основні логічні компоненти . . . 227
5.2.5.1 and2 . . . 227
5.2.5.2 не . . . 228
5.2.5.3 or2 . . . 228
5.2.5.4 xor2 . . . 229
5.2.6 Логічні приклади . . . 229
5.2.7 Компоненти конверсії . . . 230
5.2.7.1 weighted_sum . . . 230
5.3 HAL TWOPASS . . . 230
5.3.1 TWOPASS . . . 230
5.3.2 Графічний інтерфейс користувача після публікації . . . 232
5.3.3 Виключення файлів .hal . . . 233
5.3.4 Приклади . . . 234
5.4 Підручник з HAL . . . 234
5.4.1 Вступ . . . 234
5.4.2 Halcmd . . . 234
5.4.2.1 Нотація . . . 234
5.4.2.2 Автозаповнення за допомогою клавіші Tab . . . 235
5.4.2.3 Середовище RTAPI . . . 235
5.4.3 Простий приклад . . . 235
5.4.3.1 Завантаження компонента . . . 235

5.4.3.2 Вивчення HAL . . . 236
5.4.3.3 Запуск коду в реальному часі . . . 237
5.4.3.4 Зміна параметрів . . . 239
5.4.3.5 Збереження конфігурації HAL . . . 239
5.4.3.6 Вихід з Халруна . . . 240
5.4.3.7 Відновлення конфігурації HAL . . . 240
5.4.3.8 Вилучення HAL з пам’яті . . . 240
5.4.4 Півметра . . . 241
5.4.5 Приклад Stepgen . . . 243
5.4.5.1 Встановлення компонентів . . . 243
5.4.5.2 З’єднання контактів із сигналами . . . 245
5.4.5.3 Налаштування виконання в реальному часі - потоки та функції . . . 245
5.4.5.4 Налаштування параметрів . . . 247
5.4.5.5 Запустіть це! . . . 247
5.4.6 Галскоп . . . 248
5.4.6.1 Підключення зондів осцилографа . . . 250
5.4.6.2 Захоплення наших перших хвильових форм . . . 253
5.4.6.3 Вертикальні коригування . . . 254
5.4.6.4 Запуск . . . 255
5.4.6.5 Горизонтальні коригування . . . 257
5.4.6.6 Більше каналів . . . 258
5.4.6.7 Більше зразків . . . 259
5.5 Приклади HAL . . . 259
5.5.1 Підключення двох виходів . . . 259
5.5.2 Ручна зміна інструменту . . . 260
5.5.3 Обчисліть швидкість . . . 261
5.5.4 Деталі плавного запуску . . . 262
5.5.5 Автономний HAL . . . 264
5.6 Основні компоненти . . . 266
5.6.1 Рух . . . 266
5.6.1.1 Опції . . . 267
5.6.1.2 Піни . . . 267
5.6.1.3 Параметри . . . 268
5.6.1.4 Функції . . . 269
5.6.2 Шпиндель . . . 269
5.6.2.1 Піни . . . 269
5.6.3 Осі та шарнірні штифти та параметри . . . 271
5.6.4 iControl . . . 271
5.6.4.1 Піни . . . 271

5.6.5 Налаштування INI . . . 271
5.6.5.1 Піни . . . 271
5.7 Список компонентів HAL . . . 273
5.7.1 Компоненти . . . 273
5.7.1.1 Користувацькі інтерфейси (не в режимі реального часу) . . . 273
5.7.1.2 Рух (не в реальному часі) . . . 274
5.7.1.3 Драйвери обладнання . . . 274
5.7.1.4 Mesa та інші плати вводу/виводу (реального часу) . . . 275
5.7.1.5 Утиліти (не в режимі реального часу) . . . 276
5.7.1.6 Обробка сигналів (реальний час) . . . 277
5.7.1.7 Генерація сигналів (реального часу) . . . 278
5.7.1.8 Кінематика (реальний час) . . . 279
5.7.1.9 Керування рухом (у режимі реального часу) . . . 280
5.7.1.10 Керування двигуном (у режимі реального часу) . . . 280
5.7.1.11 Моделювання/Тестування . . . 280
5.7.1.12 Інше (у реальному часі) . . . 281
5.7.2 Виклики HAL API . . . 282
5.7.3 Виклики RTAPI . . . 283
5.8 Описи компонентів HAL . . . 284
5.8.1 StepGen . . . 284
5.8.1.1 Піни . . . 285
5.8.1.2 Параметри . . . 285
5.8.1.3 Типи кроків . . . 286
5.8.1.4 Функції . . . 287
5.8.2 PWMgen . . . 287
5.8.2.1 Типи виводу . . . 288
5.8.2.2 Піни . . . 288
5.8.2.3 Параметри . . . 289
5.8.2.4 Функції . . . 289
5.8.3 Енкодер . . . 289
5.8.3.1 Піни . . . 290
5.8.3.2 Параметри . . . 291
5.8.3.3 Функції . . . 291
5.8.4 PID . . . 292
5.8.4.1 Піни . . . 292
5.8.4.2 Функції . . . 294
5.8.5 Імітований кодер . . . 294
5.8.5.1 Піни . . . 294
5.8.5.2 Параметри . . . 294

5.8.5.3 Функції . . . 295
5.8.6 Усунення дребезгу . . . 295
5.8.6.1 Піни . . . 295
5.8.6.2 Параметри . . . 296
5.8.6.3 Функції . . . 296
5.8.7 SigGen . . . 296
5.8.7.1 Піни . . . 296
5.8.7.2 Параметри . . . 297
5.8.7.3 Функції . . . 297
5.8.8 lut5 . . . 297
5.9 Генератор компонентів HAL . . . 299
5.9.1 Вступ . . . 299
5.9.2 Встановлення . . . 299
5.9.3 Компіляція . . . 300
5.9.3.1 Усередині дерева вихідних кодів . . . 300
5.9.3.2 Компоненти реального часу поза деревом вихідного коду . . . 300
5.9.3.3 Компоненти, що не працюють у реальному часі, поза межами дерева вихідного коду . . . 300
5.9.4 Використання компонента . . . 301
5.9.5 Визначення . . . 301
5.9.6 Створення екземпляра . . . 301
5.9.7 Неявні параметри . . . 302
5.9.8 Синтаксис . . . 302
5.9.8.1 Функції HAL . . . 304
5.9.8.2 Опції . . . 304
5.9.8.3 Ліцензія та авторство . . . 306
5.9.8.4 Зберігання даних для кожного екземпляра . . . 306
5.9.8.5 Коментарі . . . 307
5.9.9 Обмеження . . . 307
5.9.10 Зручні макроси . . . 307
5.9.11 Компоненти з однією функцією . . . 308
5.9.12 Компонент особистості . . . 308
5.9.13 Приклади . . . 308
5.9.13.1 постійний . . . 308
5.9.13.2 синхронізація . . . 309
5.9.13.3 out8 . . . 309
5.9.13.4 hal_loop . . . 310
5.9.13.5 демо-масиву . . . 311
5.9.13.6 ранд . . . 311

5.9.13.7 логіка (використовуючи особистість) . . . 311
5.9.13.8 Загальні функції . . . 313
5.9.14 Використання командного рядка . . . 313
5.10 Файли HALCL . . . 314
5.10.1 Сумісність . . . 314
5.10.2 Команди Haltcl . . . 314
5.10.3 змінні INI-файлу Haltcl . . . 314
5.10.4 Конвертування HAL-файлів у Tcl-файли . . . 316
5.10.5 Примітки Haltcl . . . 316
5.10.6 Приклади Haltcl . . . 316
5.10.7 Haltcl Інтерактивний . . . 317
5.10.8 Приклади розподілу Haltcl (simulator) . . . 317
5.11 Інтерфейс користувача HAL . . . 317
5.11.1 Вступ . . . 317
5.11.2 MDI . . . 318
5.11.3 Приклад конфігурації . . . 318
5.11.4 Довідка про піни Halui . . . 318
5.11.4.1 Перервати . . . 319
5.11.4.2 E-Стій . . . 319
5.11.4.3 Перевизначення каналу . . . 319
5.11.4.4 Туман . . . 319
5.11.4.5 Повінь . . . 319
5.11.4.6 Самонаведення . . . 319
5.11.4.7 Машина . . . 320
5.11.4.8 Максимальна швидкість . . . 320
5.11.4.9 MDI . . . 320
5.11.4.10 Суглоб . . . 320
5.11.4.11 Суглобовий біг . . . 321
5.11.4.12 Вісь . . . 322
5.11.4.13 Біг по осі . . . 322
5.11.4.14 Режим . . . 323
5.11.4.15 Програма . . . 324
5.11.4.16 Швидке перевизначення . . . 324
5.11.4.17 Корекції шпинделя . . . 325
5.11.4.18 Шпиндель . . . 325
5.11.4.19 Інструмент . . . 325
5.12 Приклади залів . . . 326
5.12.1 Дистанційний запуск . . . 326
5.12.2 Пауза та відновлення . . . 327

5.13 Створення компонентів Python, що не працюють у реальному часі . . . 328
5.13.1 Базовий приклад використання . . . 328
5.13.2 Компоненти та затримки, що не працюють у реальному часі . . . 329
5.13.3 Створення контактів та параметрів . . . 329
5.13.3.1 Зміна префікса . . . 330
5.13.4 Читання та запис виводів і параметрів . . . 330
5.13.4.1 Виводи керування виходом (HAL_OUT) . . . 330
5.13.4.2 Керування двонаправленими (HAL_IO) контактами . . . 330
5.13.5 Виходимо . . . 331
5.13.6 Корисні функції . . . 331
5.13.7 Константи . . . 331
5.13.8 Інформація про систему . . . 331
5.14 Канонічні інтерфейси пристроїв . . . 332
5.14.1 Вступ . . . 332
5.14.2 Цифровий вхід . . . 332
5.14.2.1 Піни . . . 332
5.14.2.2 Параметри . . . 332
5.14.2.3 Функції . . . 332
5.14.3 Цифровий вихід . . . 332
5.14.3.1 Піни . . . 332
5.14.3.2 Параметри . . . 332
5.14.3.3 Функції . . . 332
5.14.4 Аналоговий вхід . . . 333
5.14.4.1 Піни . . . 333
5.14.4.2 Параметри . . . 333
5.14.4.3 Функції . . . 333
5.14.5 Аналоговий вихід . . . 333
5.14.5.1 Піни . . . 333
5.14.5.2 Параметри . . . 333
5.14.5.3 Функції . . . 334
5.15 Інструменти HAL . . . 334
5.15.1 Halcmd . . . 334
5.15.2 Півметра . . . 334
5.15.3 Галшоу . . . 336
5.15.4 Галскоп . . . 338
5.15.5 Пін-код SIM-карти . . . 338
5.15.6 Моделювання зонда . . . 340
5.15.7 Гістограма HAL . . . 340
5.15.8 Halreport . . . 342

6 Драйвери обладнання 345
6.1 Драйвер паралельного порту . . . 345
6.1.1 Завантаження . . . 346
6.1.2 Адреса порту PCI . . . 348
6.1.3 Піни . . . 349
6.1.4 Параметри . . . 349
6.1.5 Функції . . . 350
6.1.6 Поширені проблеми . . . 350
6.1.7 Використання DoubleStep . . . 350
6.1.8 probe_parport . . . 351
6.1.8.1 Встановлення probe_parport . . . 351
6.2 Драйвер AX5214H . . . 351
6.2.1 Встановлення . . . 351
6.2.2 Піни . . . 352
6.2.3 Параметри . . . 352
6.2.4 Функції . . . 352
6.3 Загальний водій мехатроніки . . . 352
6.3.1 I/O роз’єми . . . 354
6.3.1.1 Піни . . . 355
6.3.1.2 Параметри . . . 355
6.3.2 Роз’єми осей . . . 356
6.3.2.1 Модулі інтерфейсу Axis . . . 356
6.3.2.2 Енкодер . . . 357
6.3.2.3 Модуль StepGen . . . 360
6.3.2.4 Сигнали ввімкнення та несправності . . . 363
6.3.2.5 Вісь DAC . . . 364
6.3.3 Сервопідсилювачі CAN-шини . . . 365
6.3.3.1 Піни . . . 365
6.3.3.2 Параметри . . . 365
6.3.4 Сторожовий таймер . . . 365
6.3.4.1 Піни . . . 366
6.3.4.2 Параметри . . . 366
6.3.5 Кінцеві, вихідні та аварійні вимикачі . . . 366
6.3.5.1 Піни . . . 367
6.3.5.2 Параметри . . . 367
6.3.6 Світлодіоди стану . . . 368
6.3.6.1 CAN . . . 368
6.3.6.2 RS485 . . . 368
6.3.6.3 EMC . . . 368

6.3.6.4 Boot . . . 368
6.3.6.5 Error . . . 369
6.3.7 Модулі розширення вводу/виводу RS485 . . . 369
6.3.7.1 Модуль релейних виходів . . . 370
6.3.7.2 Модуль цифрового входу . . . 370
6.3.7.3 Модуль DAC & ADC . . . 371
6.3.7.4 Модуль підвіски для навчання . . . 372
6.3.8 Виправлення . . . 374
6.3.8.1 Виправлення щодо карти GM6-PCI . . . 374
6.4 GS2 VFD Драйвер . . . 374
6.4.1 Параметри командного рядка . . . 374
6.4.2 Піни . . . 375
6.4.3 Параметри . . . 376
6.5 Драйвер HAL для контактів GPIO Raspberry Pi . . . 376
6.5.1 Мета . . . 376
6.5.2 Застосування . . . 376
6.5.3 Піни . . . 377
6.5.4 Параметри . . . 378
6.5.5 Функції . . . 378
6.5.6 Нумерація контактів . . . 378
6.5.7 Відомі помилки . . . 378
6.6 Загальний драйвер для будь-якого GPIO, що підтримується gpiod. . . . 379
6.6.1 Мета . . . 379
6.6.2 Застосування . . . 379
6.6.3 Піни . . . 380
6.6.4 Параметри . . . 380
6.6.5 Функції . . . 380
6.6.6 Ідентифікація PIN-коду . . . 381
6.6.7 Вирішення проблем із дозволами. . . . 381
6.6.8 Автор . . . 381
6.6.9 Відомі помилки . . . 381
6.7 Mesa Драйвер HostMot2 . . . 382
6.7.1 Вступ . . . 382
6.7.2 Бінарні файли прошивки . . . 382
6.7.3 Встановлення прошивки . . . 382
6.7.4 Завантаження HostMot2 . . . 383
6.7.5 Сторожовий пес . . . 383
6.7.5.1 Піни . . . 383
6.7.5.2 Параметри . . . 383

6.7.6 Функції HostMot2 . . . 383
6.7.7 Розпіновки . . . 384
6.7.8 PIN-файли . . . 385
6.7.9 Прошивка . . . 385
6.7.10 Піни HAL . . . 385
6.7.11 Конфігурації . . . 386
6.7.12 GPIO . . . 388
6.7.12.1 Піни . . . 388
6.7.12.2 Параметри . . . 389
6.7.13 StepGen . . . 389
6.7.13.1 Піни . . . 389
6.7.13.2 Параметри . . . 390
6.7.13.3 Вихідні параметри . . . 390
6.7.14 PWMGen . . . 391
6.7.14.1 Піни . . . 391
6.7.14.2 Параметри . . . 391
6.7.14.3 Вихідні параметри . . . 392
6.7.15 Енкодер . . . 392
6.7.15.1 Піни . . . 392
6.7.15.2 Параметри . . . 393
6.7.16 Конфігурація 5I25 . . . 393
6.7.16.1 Прошивка . . . 393
6.7.16.2 Конфігурація . . . 393
6.7.16.3 Конфігурація SSERIAL . . . 394
6.7.16.4 7I77 Ліміти . . . 394
6.7.17 Приклади конфігурацій . . . 395
6.8 MB2HAL . . . 395
6.8.1 Вступ . . . 395
6.8.2 Застосування . . . 395
6.8.3 Опції . . . 396
6.8.3.1 Розділ ініціалізації . . . 396
6.8.3.2 Розділи транзакцій . . . 396
6.8.3.3 Коди помилок . . . 397
6.8.4 Приклад конфігураційного файлу . . . 398
6.8.5 Піни . . . 403
6.8.5.1 fnct_01_read_coils . . . 403
6.8.5.2 fnct_02_read_discrete_inputs . . . 403
6.8.5.3 fnct_03_read_holding_registers . . . 403
6.8.5.4 fnct_04_read_input_registers . . . 403

6.8.5.5 fnct_05_write_single_coil . . . 404
6.8.5.6 fnct_06_write_single_register . . . 404
6.8.5.7 fnct_15_write_multiple_coils . . . 404
6.8.5.8 fnct_16_write_multiple_registers . . . 404
6.9 Драйвер частотного перетворювача Mitsub . . . 404
6.9.1 Параметри командного рядка . . . 405
6.9.2 Піни . . . 405
6.9.3 HAL приклад . . . 406
6.9.4 Налаштування частотного перетворювача Mitsubishi для послідовного використання 406
6.9.4.1 Підключення послідовного порту . . . 406
6.9.4.2 Налаштування Modbus . . . 407
6.10 Водій Мотенк . . . 407
6.10.1 Піни . . . 408
6.10.2 Параметри . . . 408
6.10.3 Функції . . . 409
6.11 Драйвер Opto22 . . . 409
6.11.1 Адаптерна плата . . . 409
6.11.2 Драйвер . . . 410
6.11.3 Піни . . . 410
6.11.4 Параметри . . . 410
6.11.5 ФУНКЦІЇ . . . 410
6.11.6 Налаштування портів вводу/виводу . . . 411
6.11.7 Нумерація контактів . . . 411
6.12 Драйвери Pico . . . 412
6.12.1 Параметри командного рядка . . . 412
6.12.2 Піни . . . 413
6.12.3 Параметри . . . 414
6.12.4 Функції . . . 415
6.13 Плутон P Драйвер . . . 415
6.13.1 Загальна інформація . . . 415
6.13.1.1 Вимоги . . . 416
6.13.1.2 Роз’єми . . . 416
6.13.1.3 Фізичні піни . . . 416
6.13.1.4 LED . . . 417
6.13.1.5 Потужність . . . 417
6.13.1.6 Інтерфейс ПК . . . 417
6.13.1.7 Перезбірка прошивки FPGA . . . 417
6.13.1.8 Для отримання додаткової інформації . . . 417
6.13.2 Сервопривід Плутона . . . 417

6.13.2.1Розпіновка . . . 418
6.13.2.2Фіксація входу та оновлення виходу . . . 420
6.13.2.3Функції, виводи та параметри HAL . . . 420
6.13.2.4Сумісне обладнання драйвера . . . 420
6.13.3Крок Плутона . . . 420
6.13.3.1Розпіновка . . . 420
6.13.3.2Фіксація входу та оновлення виходу . . . 421
6.13.3.3Таймінги ступінчастої форми хвилі . . . 421
6.13.3.4Функції, виводи та параметри HAL . . . 422
6.14Драйвер Powermax Modbus . . . 422
6.14.1Піни . . . 423
6.14.2Опис . . . 423
6.14.3Довідка: . . . 423
6.15Драйвер Servo To Go . . . 424
6.15.1Встановлення . . . 424
6.15.2Піни . . . 424
6.15.3Параметри . . . 425
6.15.4Функції . . . 425
6.16Шатл . . . 426
6.16.1Опис . . . 426
6.16.2Налаштування . . . 426
6.16.3Піни . . . 426
6.17Драйвер частотного перетворювача VFS11 . . . 427
6.17.1Параметри командного рядка . . . 427
6.17.2Піни . . . 428
6.17.3Параметри . . . 429
6.17.4INI-файл конфігурації . . . 430
6.17.5HAL приклад . . . 431
6.17.6Робота панелі . . . 432
6.17.7Відновлення помилок . . . 433
6.17.8Налаштування частотного перетворювача VFS11 для використання в Modbus 433
6.17.8.1Підключення послідовного порту . . . 433
6.17.8.2Налаштування Modbus . . . 433
6.17.9Примітка щодо програмування . . . 433

7 Приклади обладнання 435
7.1 Паралельний порт PCI . . . 435
7.2 Управління шпинделем . . . 436
7.2.1 Швидкість шпинделя 0-10 вольт . . . 436
7.2.2 PWM Швидкість шпинделя . . . 436
7.2.3 Увімкнення шпинделя . . . 437
7.2.4 Напрямок шпинделя . . . 437
7.2.5 Плавний пуск шпинделя . . . 437
7.2.6 Зворотній зв’язок шпинделя . . . 439
7.2.6.1 Синхронізований рух шпинделя . . . 439
7.2.6.2 Шпиндель на швидкості . . . 440
7.3 Підвіска MPG . . . 441
7.4 Шпиндель GS2 . . . 444
7.4.1 Приклад . . . 444

8 Класична драбина 447
8.1 Вступ до ClassicLadder . . . 447
8.1.1 Історія . . . 447
8.1.2 Вступ . . . 447
8.1.3 Приклад . . . 448
8.1.4 Базова схема фіксації ввімкнення/вимкнення . . . 449
8.2 Програмування ClassicLadder . . . 450
8.2.1 Концепції сходів . . . 450
8.2.2 Мови . . . 450
8.2.3 Компоненти . . . 450
8.2.3.1 Файли . . . 450
8.2.3.2 Модуль реального часу . . . 451
8.2.3.3 Змінні . . . 451
8.2.4 Завантаження модуля ClassicLadder, що не працює в реальному часі . . . 452
8.2.5 ClassicLadder GUI . . . 452
8.2.5.1 Менеджер розділів . . . 453
8.2.5.2 Відображення розділу . . . 453
8.2.5.3 Змінні вікна . . . 455
8.2.5.4 Вікно символів . . . 457
8.2.5.5 Вікно редактора . . . 458
8.2.5.6 Вікно конфігурації . . . 460
8.2.6 Об’єкти драбини . . . 461
8.2.6.1 КОНТАКТИ . . . 461
8.2.6.2 ТАЙМЕРИ IEC . . . 461

8.2.6.3 ТАЙМЕРИ . . . 462
8.2.6.4 МОНОСТАЛИ . . . 462
8.2.6.5 ЛІЧИЛЬНИКИ . . . 462
8.2.6.6 ПОРІВНЯТИ . . . 463
8.2.6.7 ПРИЗНАЧЕННЯ ЗМІННИХ . . . 464
8.2.6.8 КОТУШКИ . . . 466
8.2.7 Змінні ClassicLadder . . . 467
8.2.8 Програмування GRAFCET (кінцевий автомат) . . . 468
8.2.9 Modbus . . . 470
8.2.10 Налаштування MODBUS . . . 474
8.2.10.1 Інформація про MODBUS . . . 474
8.2.10.2 Помилки зв’язку . . . 474
8.2.11 Налагодження проблем Modbus . . . 475
8.2.11.1 Запит . . . 476
8.2.11.2 Відповідь на помилку . . . 477
8.2.11.3 Відповідь на дані . . . 478
8.2.11.4 Помилки MODBUS . . . 479
8.2.12 Налаштування ClassicLadder . . . 479
8.2.12.1 Додайте модулі . . . 480
8.2.12.2 Додавання логіки сходів . . . 480
8.3 Приклади ClassicLadder . . . 487
8.3.1 Лічильник упаковки . . . 487
8.3.2 Відхилити зайві імпульси . . . 487
8.3.3 Зовнішній аварійний зупинник . . . 488
8.3.4 Приклад таймера/операції . . . 492

9 Розширені теми 493

9.1 Кінематика . . . 493
9.1.1 Вступ . . . 493
9.1.1.1 Суглоби проти осей . . . 493
9.1.2 Тривіальна кінематика . . . 494
9.1.3 Нетривіальна кінематика . . . 495
9.1.3.1 Пряма трансформація . . . 496
9.1.3.2 Зворотне перетворення . . . 497
9.1.4 Деталі впровадження . . . 497
9.1.4.1 Кінематичний модуль з використанням шаблону userkins.comp . . . 499
9.2 Налаштування ”модифікованих” параметрів Денавіта-Хартенберга (DH) для ”генсеркінів” 499
9.2.1 Прелюдія . . . 499
9.2.2 Загальне . . . 499

9.2.3 Змінені DH-параметри . . . 500
9.2.4 Модифіковані DH-параметри, що використовуються в ”genserkins” . . . 500
9.2.5 Нумерація суглобів та параметрів . . . 501
9.2.6 Як почати . . . 501
9.2.7 Особливі випадки . . . 501
9.2.8 Детальний приклад (RV-6SL) . . . 501
9.2.9 Кредити . . . 520
9.3 5-осьова кінематика . . . 520
9.3.1 Вступ . . . 520
9.3.2 Конфігурації 5-осьових верстатів . . . 520
9.3.3 Орієнтація та розташування інструменту . . . 520
9.3.4 Матриці переміщення та обертання . . . 521
9.3.5 Поворотні/нахилені 5-осьові конфігурації столу . . . 522
9.3.5.1 Трансформації для верстата xyzac-trt зі зміщеннями робочої точки . . . 525
9.3.5.2 Перетворення для верстата xyzac-trt зі зміщеннями поворотних осей . . . 528
9.3.5.3 Перетворення для верстата xyzbc-trt зі зміщеннями поворотних осей . . . 531
9.3.6 Приклади повороту/нахилу столу . . . 534
9.3.6.1 Моделі моделювання Vismach . . . 534
9.3.6.2 Компенсація довжини інструменту . . . 534
9.3.7 Компоненти кінематики на замовлення . . . 535
9.3.8 Цифри . . . 536
9.3.9 ПОСИЛАННЯ . . . 538
9.4 Перемикальна кінематика (перемикачі) . . . 538
9.4.1 Вступ . . . 538
9.4.2 Перемикані кінематичні модулі . . . 539
9.4.2.1 Призначення ідентифікаційних листів . . . 539
9.4.2.2 Зворотна сумісність . . . 540
9.4.3 Піни HAL . . . 540
9.4.3.1 Зведення про піни HAL . . . 540
9.4.4 Застосування . . . 541
9.4.4.1 З’єднання HAL . . . 541
9.4.4.2 Команди G-/M-кодів . . . 541
9.4.4.3 Налаштування обмежень для INI-файлів . . . 542
9.4.4.4 Міркування щодо зміщення системи координат . . . 543
9.4.4.5 Міркування щодо зовнішнього зміщення . . . 543
9.4.5 Конфігурації симуляції . . . 544
9.4.6 Положення щодо кінематики користувача . . . 544
9.4.7 Попередження . . . 544
9.4.8 Примітки до коду . . . 545

9.5 Налаштування PID-регулятора . . . 545
9.5.1 PID Контролер . . . 545
9.5.1.1 Основи контуру керування . . . 545
9.5.1.2 Теорія . . . 546
9.5.1.3 Налаштування петлі . . . 546
9.5.1.4 Автоматичне налаштування PID-регулятора . . . 547
9.6 Перепризначення розширення G-коду . . . 549
9.6.1 Вступ: Розширення інтерпретатора RS274NGC шляхом перепризначення кодів 549
9.6.1.1 Визначення: Перепризначення кодів . . . 549
9.6.1.2 Навіщо розширювати інтерпретатор RS274NGC? . . . 549
9.6.2 Початок роботи . . . 551
9.6.2.1 Вбудовані перепризначення . . . 551
9.6.2.2 Вибір коду . . . 552
9.6.2.3 Обробка параметрів . . . 552
9.6.2.4 Обробка результатів . . . 553
9.6.2.5 Послідовність виконання . . . 553
9.6.2.6 Мінімальний приклад переробленого коду . . . 553
9.6.3 Налаштування перепризначення . . . 554
9.6.3.1 Заява REMAP . . . 554
9.6.3.2 Корисні комбінації опцій REMAP . . . 555
9.6.3.3 Параметр argspec . . . 555
9.6.4 Оновлення існуючої конфігурації для перепризначення . . . 559
9.6.5 Коди, пов’язані зі зміною інструменту переналаштування: T, M6, M61 . . . 560
9.6.5.1 Огляд . . . 560
9.6.5.2 Розуміння ролі `iocontrol` з перепризначеними кодами зміни інструменту 561
9.6.5.3 Визначення заміни M6 . . . 562
9.6.5.4 Налаштування `iocontrol` з перепризначеним M6 . . . 564
9.6.5.5 Написання змін та підготовка процедур, що передбачають букву «O» . . . 564
9.6.5.6 Внесення мінімальних змін до вбудованих кодів, включаючи `M6` . . . 565
9.6.5.7 Визначення заміни T (підготовка) . . . 566
9.6.5.8 Обробка помилок: обробка переривання . . . 567
9.6.5.9 Обробка помилок: невдача процедури NGC перепризначеного коду . . . 569
9.6.6 Перепризначення інших існуючих кодів: . . . 570
9.6.6.1 Автоматичний вибір передачі при перепрограмуванні S (встановлення швидкості шпинделя) . . . 570
9.6.6.2 Налаштування поведінки M0, M1 . . . 570
9.6.6.3 Налаштування поведінки M7, M8, M9 . . . 570
9.6.7 Створення нових циклів G-коду . . . 570
9.6.8 Налаштування вбудованого Python . . . 571

9.6.8.1 Плагін Python: конфігурація INI-файлу . . . 571
9.6.8.2 Виконання інструкцій Python з інтерпретатора . . . 571
9.6.9 Програмування вбудованого Python в інтерпретаторі RS274NGC . . . 572
9.6.9.1 Простір імен плагінів Python . . . 572
9.6.9.2 Інтерпретатор з точки зору Python . . . 572
9.6.9.3 Функції інтерпретатора `__init__` та `__delete__` . . . 573
9.6.9.4 Умови виклику: NGC до Python . . . 573
9.6.9.5 Умовні позначення викликів: Python до NGC . . . 576
9.6.9.6 Вбудовані модулі . . . 578
9.6.10 Додавання попередньо визначених іменованих параметрів . . . 579
9.6.11 Стандартні процедури клею . . . 579
9.6.11.1 T: `prepare_prolog` і `prepare_epilog` . . . 580
9.6.11.2 M6: `change_prolog` і `change_epilog` . . . 580
9.6.11.3 Цикли G-коду: `cycle_prolog` and `cycle_epilog` . . . 581
9.6.11.4 S (Встановити швидкість): `setspeed_prolog` та `setspeed_epilog` . . . 582
9.6.11.5 F (Встановити потік): `setfeed_prolog` та `setfeed_epilog` . . . 582
9.6.11.6 M61 Встановити номер інструменту: `settool_prolog` та `settool_epilog` 582
9.6.12 Перепризначене виконання коду . . . 582
9.6.12.1 Середовище виклику процедур NGC під час перепризначення . . . 582
9.6.12.2 Вкладені перепризначені коди . . . 582
9.6.12.3 Порядковий номер під час перепризначення . . . 582
9.6.12.4 Прапорці налагодження . . . 582
9.6.12.5 Налагодження вбудованого коду Python . . . 583
9.6.13 Попередній перегляд осі та виконання перепризначеного коду . . . 584
9.6.14 Перепризначені коди . . . 585
9.6.14.1 Існуючі коди, які можна перепризначити . . . 585
9.6.14.2 Наразі нерозподілені G-коди: . . . 585
9.6.14.3 Наразі нерозподілені M-коди: . . . 589
9.6.15 Короткий огляд виконання програми LinuxCNC . . . 589
9.6.15.1 Стан інтерпретатора . . . 589
9.6.15.2 Взаємодія завдання та інтерпретатора, черга та попереднє читання . . . 590
9.6.15.3 Прогнозування положення машини . . . 590
9.6.15.4 Засоби запобігання черзі порушують прогнозування позиції . . . 590
9.6.15.5 Як поводяться з тими, хто не пропускає черги . . . 591
9.6.15.6 Порядок слів та порядок виконання . . . 591
9.6.15.7 Розбір . . . 591
9.6.15.8 Виконання . . . 591
9.6.15.9 Виконання процедури . . . 592
9.6.15.10 Як зараз працює зміна інструменту . . . 592

9.6.15.1 Як працює Tx (інструмент підготовки) . . . 592
9.6.15.2 Як працює M6 (інструмент зміни) . . . 593
9.6.15.3 Як працює M61 (Зміна номера інструменту) . . . 594
9.6.16 Статус . . . 594
9.6.17 Зміни . . . 594
9.6.18 Налагодження . . . 594
9.7 Компонент Moveoff . . . 595
9.7.1 Зміна існуючої конфігурації . . . 596
9.8 Автономний перекладач . . . 600
9.8.1 Застосування . . . 600
9.8.2 Приклад . . . 601
9.9 Зміщення зовнішніх осей . . . 602
9.9.1 Налаштування INI-файлу . . . 602
9.9.2 Піни HAL . . . 602
9.9.2.1 Штифти HAL для руху по осях . . . 602
9.9.2.2 Інші контакти HAL для руху . . . 603
9.9.3 Застосування . . . 603
9.9.3.1 Обчислення зміщення . . . 603
9.9.3.2 Вимкнення/увімкнення машини . . . 603
9.9.3.3 М’які обмеження . . . 603
9.9.3.4 Нотатки . . . 604
9.9.3.5 УВАГА . . . 604
9.9.4 Пов’язані компоненти HAL . . . 605
9.9.4.1 eoffset_per_angle.comp . . . 605
9.9.5 Тестування . . . 605
9.9.6 Приклади . . . 605
9.9.6.1 eoffsets.ini . . . 606
9.9.6.2 jwp_z.ini . . . 606
9.9.6.3 dynamic_offsets.ini . . . 606
9.9.6.4 opa.ini (eoffset_per_angle) . . . 606
9.10 Інтерфейс бази даних інструментів . . . 607
9.10.1 Інтерфейс . . . 607
9.10.1.1 Налаштування INI-файлу . . . 607
9.10.1.2 **db_program** операція (v2.1) . . . 607
9.10.1.3 Застосування . . . 608
9.10.1.4 Приклад програми . . . 609
9.10.1.5 Модуль Python toolbd . . . 610
9.10.2 Конфігурації симуляції . . . 611
9.10.2.1 Нотатки . . . 611

II Застосування 612

10 Інтерфейс користувача 613

10.1 AXIS GUI . . . 613
10.1.1 Вступ . . . 613
10.1.2 Початок роботи . . . 614
10.1.2.1 Налаштування INI . . . 615
10.1.2.2 Типова сесія . . . 615
10.1.3 Вікно AXIS . . . 616
10.1.3.1 Пункти меню . . . 616
10.1.3.2 Кнопки панелі інструментів . . . 620
10.1.3.3 Область графічного відображення . . . 621
10.1.3.4 Область відображення тексту . . . 623
10.1.3.5 Ручне керування . . . 623
10.1.3.6 MDI . . . 626
10.1.3.7 Перевизначення каналу . . . 626
10.1.3.8 Коригування швидкості шпинделя . . . 627
10.1.3.9 Швидкість штовхання . . . 627
10.1.3.10 Максимальна швидкість . . . 627
10.1.4 Елементи керування клавіатурою . . . 627
10.1.4.1 Клавіші перевизначення подачі . . . 627
10.1.5 Показати стан LinuxCNC (`linuxcnctop`) . . . 628
10.1.6 MDI-інтерфейс . . . 629
10.1.7 axis-remote . . . 630
10.1.8 Ручна зміна інструменту . . . 630
10.1.9 Модулі Python . . . 630
10.1.10 Використання AXIS у режимі токарного верстата . . . 631
10.1.11 Використання AXIS у режимі різання пінопласту . . . 634
10.1.12 Розширена конфігурація . . . 635
10.1.12.1 Фільтри програми . . . 636
10.1.12.2 База даних ресурсів X . . . 637
10.1.12.3 Джоґвіст . . . 637
10.1.12.4 ~/.axisrc . . . 637
10.1.12.5 USER_COMMAND_FILE . . . 638
10.1.12.6 user_live_update() . . . 638
10.1.12.7 user_hal_pins() . . . 638
10.1.12.8 Зовнішній редактор . . . 638
10.1.12.9 Віртуальна панель керування . . . 638
10.1.12.10 Контроль попереднього перегляду . . . 638

10.1.12.11 Поторкніться до місця призначення, використовуючи фактичне положення 639
10.1.13 Axisui . . . 639
10.1.14 Підказки щодо налаштування AXIS . . . 640
10.1.14.1 Функція оновлення . . . 640
10.1.14.2 Вимкнути діалогове вікно закриття . . . 640
10.1.14.3 Зміна шрифту тексту . . . 641
10.1.14.4 Зміна швидкості переміщення за допомогою комбінацій клавіш . . . 641
10.1.14.5 Прочитайте INI-файл . . . 642
10.1.14.6 Зчитування статусу LinuxCNC . . . 642
10.1.14.7 Змінити поточний вигляд . . . 642
10.1.14.8 Створення нових пінів AXISUI HAL . . . 642
10.1.14.9 Створення нового компонента та виводів HAL . . . 642
10.1.14.10 Перемикання вкладок за допомогою контактів HAL . . . 643
10.1.14.11 Додати кнопку «Перейти на головну» . . . 644
10.1.14.12 Додати кнопку до ручної рамки . . . 644
10.1.14.13 Читання внутрішніх змінних . . . 645
10.1.14.14 Приховати віджети . . . 646
10.1.14.15 Змінити мітку . . . 647
10.1.14.16 Перенаправити існуючу команду . . . 647
10.1.14.17 Зміна кольору DRO . . . 647
10.1.14.18 Зміна кнопок панелі інструментів . . . 647
10.1.14.19 Зміна кольорів плоттера . . . 648
10.2 GMOCCAPY . . . 649
10.2.1 Вступ . . . 649
10.2.2 Вимоги . . . 650
10.2.3 Як отримати GMOCCAPY . . . 650
10.2.4 Базова конфігурація . . . 651
10.2.4.1 Розділ «ДИСПЛЕЙ» . . . 652
10.2.4.2 Розділ TRAJ . . . 654
10.2.4.3 Кнопки макросів . . . 654
10.2.4.4 Вбудовані вкладки та панелі . . . 656
10.2.4.5 Повідомлення, створені користувачем . . . 659
10.2.4.6 Контроль попереднього перегляду . . . 660
10.2.4.7 Файл команд користувача . . . 660
10.2.4.8 CSS-файл користувача . . . 661
10.2.4.9 Лісозаготівля . . . 661
10.2.5 Піни HAL . . . 662
10.2.5.1 Списки правої та нижньої кнопок . . . 662
10.2.5.2 Швидкості та корекції . . . 665

10.2.5.3 Jog HAL Pins . . . 668
10.2.5.4 Швидкості поштовху та штифт HAL Turtle-Jog . . . 668
10.2.5.5 Штифти HAL для приросту поштовху . . . 669
10.2.5.6 PIN-код для розблокування апаратного забезпечення . . . 669
10.2.5.7 Піни помилок/попереджень . . . 669
10.2.5.8 Піни HAL, створені користувачем, для повідомлень . . . 670
10.2.5.9 Штифти зворотного зв’язку шпинделя . . . 670
10.2.5.10 Шпильки для відображення інформації про хід виконання програми . . . 671
10.2.5.11 Штифти, пов’язані з інструментами . . . 671
10.2.6 Автоматичне вимірювання інструментів . . . 672
10.2.6.1 Надані піни . . . 674
10.2.6.2 Модифікації INI-файлу . . . 674
10.2.6.3 Необхідні файли . . . 675
10.2.6.4 Необхідні HAL-з’єднання . . . 675
10.2.7 Сторінка налаштувань . . . 676
10.2.7.1 Зовнішній вигляд . . . 677
10.2.7.2 Апаратне забезпечення . . . 682
10.2.7.3 Розширені налаштування . . . 684
10.2.8 Тема значків . . . 686
10.2.8.1 Тема власних значків . . . 687
10.2.8.2 Символічні ікони . . . 687
10.2.9 Розділ, специфічний для токарного верстата . . . 688
10.2.10 Розділ, специфічний для плазми . . . 691
10.2.11 Відео на YouTube . . . 691
10.2.11.1 Базове використання . . . 692
10.2.11.2 Імітація джог-колес . . . 692
10.2.11.3 Сторінка налаштувань . . . 692
10.2.11.4 Імітація апаратної кнопки . . . 692
10.2.11.5 Вкладки користувача . . . 692
10.2.11.6 Відео про вимірювання інструментів . . . 692
10.2.12 Відомі проблеми . . . 692
10.2.12.1 Дивні цифри в інформаційній області . . . 692
10.2.12.2 Не завершується макрос . . . 693
10.3 Сенсорний графічний інтерфейс користувача . . . 693
10.3.1 Конфігурація панелі . . . 694
10.3.1.1 HAL-з’єднання . . . 694
10.3.1.2 Рекомендовано для будь-якої конфігурації . . . 695
10.3.2 Налаштування . . . 695
10.3.2.1 Увімкнення Touchy . . . 695

10.3.2.2Параметри . . . 696
10.3.2.3Макроси . . . 696
10.4G-екран . . . 696
10.4.1Вступ . . . 696
10.4.1.1Щасливий файл . . . 701
10.4.1.2PyGTK . . . 701
10.4.2GladeVCP . . . 702
10.4.2.1Огляд . . . 702
10.4.2.2Створіть панель GladeVCP . . . 703
10.4.3Створення простого користувацького екрану з чистого аркуша . . . 704
10.4.4Приклад файлу обробника . . . 706
10.4.4.1Додавання функцій комбінацій клавіш . . . 707
10.4.4.2Стан Linuxcnc . . . 708
10.4.4.3Клавіші для бігу . . . 708
10.4.5Запуск Gscreen . . . 709
10.4.6Налаштування INI . . . 710
10.4.7Повідомлення діалогового вікна користувача . . . 710
10.4.7.1Скопіюйте файл обробника Stock/Glade для модифікації . . . 712
10.5QtDragon GUI . . . 713
10.5.1Вступ . . . 713
10.5.1.1QtDragon . . . 713
10.5.1.2QtDragon_lathe . . . 714
10.5.1.3QtDragon_hd . . . 715
10.5.1.4QtDragon_hd_vertical . . . 715
10.5.2Початок роботи - INI-файл . . . 715
10.5.2.1Дисплей . . . 716
10.5.2.2Параметри . . . 716
10.5.2.3Лісозаготівля . . . 716
10.5.2.4Перевизначення елементів керування . . . 716
10.5.2.5Шпиндельні елементи керування . . . 717
10.5.2.6Інкременти бігу підтюпцем . . . 717
10.5.2.7Приріст сітки . . . 717
10.5.2.8Швидкість штовхання . . . 717
10.5.2.9Система діалогів повідомлень користувача . . . 717
10.5.2.10Збудувати власні панелі VCP . . . 718
10.5.2.11Шляхи підпрограм . . . 719
10.5.2.12Контроль попереднього перегляду . . . 720
10.5.2.13Розширення/фільтри програми . . . 720
10.5.2.14Налаштування зонда/сенсорної панелі/лазера . . . 721

10.5.2.15 Виявлення переривання . . . 721
10.5.2.16 Коди запуску . . . 722
10.5.2.17 Кнопки макросів . . . 722
10.5.2.18 Файл HAL для публікації графічного інтерфейсу . . . 722
10.5.2.19 Команда HAL після графічного інтерфейсу . . . 722
10.5.2.20 Міст HAL . . . 723
10.5.2.21 Вбудовані зразки конфігурацій . . . 723
10.5.3 Прив’язки клавіш . . . 724
10.5.4 Кпонки . . . 724
10.5.5 Віртуальна клавіатура . . . 724
10.5.6 Піни HAL . . . 724
10.5.7 HAL-файли . . . 726
10.5.8 Ручна зміна інструментів . . . 726
10.5.9 Шпиндель . . . 727
10.5.10 Автоматичне підняття осі Z під час паузи програми . . . 727
10.5.11 Компенсація рівня Z . . . 728
10.5.11.1 Використання G-коду Ripper для компенсації рівня Z . . . 729
10.5.12 Зондування . . . 731
10.5.12.3 Зонд Versa . . . 732
10.5.12.2 Базовий зонд . . . 735
10.5.12.3 Налаштування віджета екрана зонда . . . 738
10.5.13 Сенсорна панель . . . 738
10.5.14 Автоматичне вимірювання інструментів . . . 739
10.5.14.0 Огляд . . . 739
10.5.14.2 Огляд робочого процесу . . . 739
10.5.14.3 Детальний приклад робочого процесу . . . 741
10.5.14.4 Вимірювання висоти заготовки зондом у QtDragon_hd . . . 742
10.5.14.5 Вимірювання висоти заготовки зондом . . . 743
10.5.14.6 Штифти для вимірювання інструментів . . . 745
10.5.14.7 Зміни INI-файлу вимірювання інструменту . . . 745
10.5.14.8 Необхідні HAL-з’єднання . . . 747
10.5.15 Бігти від лінії . . . 747
10.5.16 Лазерні кнопки . . . 747
10.5.17 Опис вкладок . . . 748
10.5.17.1 Головна вкладка . . . 748
10.5.17.2 Вкладка «Файл» . . . 748
10.5.17.3 Вкладка «Зміщення» . . . 748
10.5.17.4 Вкладка інструментів . . . 748
10.5.17.5 Вкладка стану . . . 749

10.5.17.8 Вкладка зонда . . . 749
10.5.17.9 Вкладка «Вигляд камери» . . . 749
10.5.17.10 Вкладка G-кодів . . . 749
10.5.17.11 Вкладка налаштувань . . . 749
10.5.17.12 Вкладка налаштувань . . . 750
10.5.17.13 Вкладка «Утиліти» . . . 751
10.5.17.14 Вкладка користувача . . . 751
10.5.18 Стилі . . . 751
10.5.19 Інтернаціоналізація . . . 752
10.5.20 Налаштування . . . 753
10.5.20.1 Таблиці стилів . . . 753
10.5.20.2 Код Qt Designer та Python . . . 756
10.6 NGCGUI . . . 758
10.6.1 Огляд . . . 758
10.6.2 Демонстраційні конфігурації . . . 759
10.6.3 Розташування бібліотек . . . 761
10.6.4 Автономне використання . . . 762
10.6.4.1 Автономний NGCGUI . . . 762
10.6.4.2 Автономний PyNGCGUI . . . 762
10.6.5 Вбудовування NGCGUI . . . 763
10.6.5.1 Вбудовування NGCGUI в AXIS . . . 763
10.6.5.2 Вбудовування PyNGCGUI як вкладки GladeVCP у графічний інтерфейс 764
10.6.5.3 Додаткові елементи INI-файлу, необхідні для NCGUI або PyNGCGUI . . 764
10.6.5.4 Трасер Truetype . . . 766
10.6.5.5 Специфікації шляху до INI-файлу . . . 767
10.6.5.6 Зведена інформація про елементи INI-файлу для використання NGCGUI 768
10.6.6 Вимоги до файлів для сумісності з NGCGUI . . . 770
10.6.6.1 Вимоги до підпрограми однофайлового G-коду (.ngc) . . . 770
10.6.6.2 Вимоги до файлу G-code-meta-compiler (.gcmc) . . . 772
10.6.7 Приклад DB25 . . . 774
10.6.8 Створення підпрограми . . . 776
10.7 TkLinuxCNC GUI . . . 777
10.7.1 Вступ . . . 777
10.7.2 Початок роботи . . . 777
10.7.2.1 Типовий сеанс роботи з TkLinuxCNC . . . 778
10.7.3 Елементи вікна TkLinuxCNC . . . 778
10.7.3.1 Основні кнопки . . . 778
10.7.3.2 Рядок стану зсуву дисплея . . . 779
10.7.3.3 Область відображення координат . . . 779

10.7.3.4Інтерпретатор TkLinuxCNC / Автоматичне керування програмою 779
10.7.3.5Ручне керування 780
10.7.3.6Введення коду 781
10.7.3.7Швидкість штовхання 781
10.7.3.8Перевизначення каналу 781
10.7.3.9Коригування швидкості шпинделя 781
10.7.4Елементи керування клавіатурою 781
10.8QtPlasmaC 782
10.8.1Преамбула 782
10.8.2Ліцензія 782
10.8.3Вступ 782
10.8.4Встановлення LinuxCNC 785
10.8.4.1Якщо у користувача не встановлено Linux 786
10.8.4.2Встановлення пакета (Buildbot), якщо користувач має Linux на Debian 12 (Bookworm) 786
10.8.4.3Встановлення пакета (Buildbot), якщо користувач має Linux на Debian 12 (Bookworm) або Debian 11 (Bullseye) 786
10.8.4.4Запуск інсталяції на місці, якщо у користувача встановлено Linux . . . 786
10.8.5Створення конфігурації QtPlasmaC 786
10.8.5.1Режими 786
10.8.5.2Доступно I/Os 787
10.8.5.3Рекомендовані налаштування: 789
10.8.5.4Налаштування 789
10.8.5.5Помилки залежностей Qt 795
10.8.5.6Початкове налаштування 795
10.8.6Міграція на QtPlasmaC з PlasmaC (AXIS або GMOCCAPY) 799
10.8.7Інші міркування щодо налаштування QtPlasmaC 799
10.8.7.1Низькочастотний фільтр 799
10.8.7.2Відмова від контакту 799
10.8.7.3Контактне навантаження 800
10.8.7.4Запуск робочого столу 801
10.8.7.5Файли QtPlasmaC 802
10.8.7.6INI-файл 803
10.8.8Огляд графічного інтерфейсу QtPlasmaC 805
10.8.8.1Вихід з QtPlasmaC 805
10.8.8.2ГОЛОВНА вкладка 805
10.8.8.3Попередній перегляд 813
10.8.8.4РОЗМОВНА вкладка 813
10.8.8.5Вкладка ПАРАМЕТРИ 814

10.8.8.6 Вкладка НАЛАШТУВАННЯ . . . 821
10.8.8.7 Вкладка СТАТИСТИКА . . . 824
10.8.9 Використання QtPlasmaC . . . 825
10.8.9.1 Системи одиниць . . . 826
10.8.9.2 Коди преамбули та постамбули . . . 826
10.8.9.3 Обов’язкові коди . . . 826
10.8.9.4 Координати . . . 827
10.8.9.5 Швидкість подачі різання . . . 827
10.8.9.6 Файл матеріалу . . . 827
10.8.9.7 Ручне оброблення матеріалів . . . 829
10.8.9.8 Автоматичне оброблення матеріалів . . . 830
10.8.9.9 Додавання матеріалів за допомогою магічних коментарів у G-коді . . . 831
10.8.9.10 Конвертер матеріалів . . . 832
10.8.9.11 ЛАЗЕР . . . 835
10.8.9.12 КАМЕРА . . . 837
10.8.9.13 Допуск шляху . . . 839
10.8.9.14 Призупинений рух . . . 839
10.8.9.15 Пауза в кінці монтажу . . . 839
10.8.9.16 Кілька інструментів . . . 840
10.8.9.17 Зменшення швидкості . . . 840
10.8.9.18 THC (Контролер висоти пальника) . . . 841
10.8.9.19 Компенсація різця . . . 842
10.8.9.20 Початкове відчуття висоти (IHS) Пропустити . . . 843
10.8.9.21 Зондування . . . 843
10.8.9.22 Зі зміщенням зондування . . . 844
10.8.9.23 Типи розрізів . . . 845
10.8.9.24 Різання отворів - вступ . . . 845
10.8.9.25 Різання отворів . . . 846
10.8.9.26 Різання отворів - автоматичне . . . 848
10.8.9.27 Одинарний розріз . . . 849
10.8.9.28 Товсті матеріали . . . 851
10.8.9.29 Режим сітки (різання розширеного металу) . . . 851
10.8.9.30 Ігнорувати дугу ОК . . . 852
10.8.9.31 Відновлення від порізів . . . 853
10.8.9.32 Вігти від лінії . . . 854
10.8.9.33 Писар . . . 856
10.8.9.34 Шлямистість . . . 858
10.8.9.35 Різання труб . . . 859
10.8.9.36 Користувацькі розкладки віртуальної клавіатури . . . 859

10.8.9.3 Комбінації клавіш . . . 860
10.8.9.3 MDI . . . 862
10.8.10 Бібліотека розмовних фігур . . . 863
10.8.10.1 Налаштування розмовного режиму . . . 865
10.8.10.2 Розмовні лінії та дуги . . . 866
10.8.10.3 Розмовна одинарна форма . . . 867
10.8.10.4 Розмовна група фігур . . . 868
10.8.10.5 Розмовний блок . . . 868
10.8.10.6 Розмовне збереження роботи . . . 870
10.8.11 Повідомлення про помилки . . . 870
10.8.11.1 Реєстрація помилок . . . 870
10.8.11.2 Відображення повідомлення про помилку . . . 870
10.8.11.3 Критичні помилки . . . 870
10.8.11.4 Попереджувальні повідомлення . . . 872
10.8.12 Оновлення QtPlasmaC . . . 873
10.8.12.1 Стандартне оновлення . . . 873
10.8.12.2 Постійне оновлення . . . 873
10.8.13 Зміна існуючої конфігурації QtPlasmaC . . . 873
10.8.14 Налаштування графічного інтерфейсу QtPlasmaC . . . 874
10.8.14.1 Додати власний стиль . . . 874
10.8.14.2 Створіть новий стиль . . . 874
10.8.14.3 Повернення до стилю за замовчуванням . . . 875
10.8.14.4 Користувацький код Python . . . 876
10.8.14.5 Користувацький фільтр G-коду . . . 876
10.8.15 Розширені теми QtPlasmaC . . . 877
10.8.15.1 Налаштовані кнопки користувача . . . 877
10.8.15.2 Периферійні зміщення (лазер, камера, розмітка, зонд зміщення) . . . 885
10.8.15.3 Зберігайте рух Z . . . 887
10.8.15.4 Зовнішні контакти HAL . . . 887
10.8.15.5 Приховати кнопки програм . . . 889
10.8.15.6 Режим налаштування 0 Дуга OK . . . 890
10.8.15.7 Затримка втрати дуги . . . 890
10.8.15.8 Нульове вікно . . . 891
10.8.15.9 Налаштування зондування порожнечі . . . 891
10.8.15.10 Максимальне зміщення . . . 891
10.8.15.11 Вимкнути вкладки під час автоматизованого руху . . . 892
10.8.15.12 Блокування поштовхового руху за допомогою Z+ Jog . . . 892
10.8.15.13 Виходи стану QtPlasmaC . . . 892
10.8.15.14 Налагодження QtPlasmaC Друк . . . 893

10.8.15.5 Комунікації Hypertherm PowerMax . . . 893
10.8.15.6 Рухомий Пірс . . . 894
10.8.16 Інтернаціоналізація . . . 898
10.8.17 Додаток . . . 899
10.8.17.1 Приклади конфігурацій . . . 899
10.8.17.2 Зразки NGC . . . 900
10.8.17.3 Специфічні G-коди QtPlasmaC . . . 900
10.8.17.4 Приклади G-коду QtPlasmaC . . . 901
10.8.17.5 Таблиця THCAD . . . 903
10.8.17.6 З’єднання RS485 . . . 905
10.8.17.7 Дуга в порядку з герконовим реле . . . 907
10.8.17.8 Схеми контактного навантаження . . . 909
10.8.18 Відомі проблеми . . . 909
10.8.18.1 Біг по клавіатурі . . . 909
10.8.18.2 NO_FORCE_HOMING . . . 910
10.8.19 Внесок коду в QtPlasmaC . . . 910
10.8.20 Підтримка . . . 911
10.9 MDRO GUI . . . 911
10.9.1 Вступ . . . 911
10.9.2 Початок роботи . . . 912
10.9.2.1 Параметри INI-файлу . . . 912
10.9.2.2 Параметри командного рядка . . . 913
10.9.2.3 Піни . . . 913
10.9.3 Вікно MDRO . . . 913
10.9.4 Операції з індексами . . . 914
10.9.5 Симулятор . . . 914

11 G-code програмування 915

11.1 Системи координат . . . 915
11.1.1 Вступ . . . 915
11.1.2 Система координат машини . . . 915
11.1.2.1 Переміщення координат машини: G53 . . . 915
11.1.3 Системи координат . . . 916
11.1.3.1 Система координат за замовчуванням . . . 918
11.1.3.2 Налаштування зміщень системи координат . . . 918
11.1.4 Локальні та глобальні зміщення . . . 919
11.1.4.1 Команда G52 . . . 919
11.1.5 Зміщення осей G92 . . . 919
11.1.5.1 Команди G92 . . . 919

11.1.5.2Встановлення значень G92 . . . 921
11.1.5.3Застереження щодо стійкості G92 . . . 921
11.1.5.4Застереження щодо взаємодії G92 та G52 . . . 922
11.1.6Приклади програм із використанням зміщень . . . 922
11.1.6.1Приклад програми з використанням зміщень координат заготовки . . . 922
11.1.6.2Приклад програми з використанням зміщень G52 . . . 924
11.2Компенсація інструменту . . . 924
11.2.1Дотик вимкнено . . . 924
11.2.1.1Використання G10 L1/L10/L11 . . . 925
11.2.2Таблиця інструментів . . . 925
11.2.2.1Формат таблиці інструментів . . . 925
11.2.2.2Інструмент IO . . . 927
11.2.2.3Змінювачі інструментів . . . 928
11.2.3Компенсація довжини інструменту . . . 929
11.2.4Компенсація радіуса різця . . . 930
11.2.4.1Огляд . . . 931
11.2.4.2Приклади . . . 933
11.3Графічний інтерфейс редагування інструментів . . . 934
11.3.1Огляд . . . 934
11.3.2Сортування за стовпцями . . . 935
11.3.3Вибір стовпців . . . 936
11.3.4Окреме використання . . . 936
11.4Огляд програмування G-кодом . . . 937
11.4.1Огляд . . . 937
11.4.2Формат рядка . . . 938
11.4.2.1/: Видалення блоку . . . 938
11.4.2.2Додатковий номер рядка . . . 939
11.4.2.3Слова, параметри, підпрограми, коментарі . . . 939
11.4.2.4Маркер кінця лінії . . . 940
11.4.3Числа . . . 940
11.4.4Параметри . . . 941
11.4.4.1Нумеровані параметри . . . 942
11.4.4.2Коди та параметри підпрограм . . . 944
11.4.4.3Іменовані параметри . . . 944
11.4.4.4Попередньо визначені іменовані параметри . . . 945
11.4.4.5Системні параметри . . . 946
11.4.5Піни HAL та значення INI . . . 947
11.4.6Вирази . . . 948
11.4.7Бінарні оператори . . . 948

11.4.8 Рівність та значення з плаваючою комою . . . 949
11.4.9 Функції . . . 949
11.4.10 Повторювані елементи . . . 950
11.4.11 Порядок товарів . . . 950
11.4.12 Команди та режими роботи машини . . . 951
11.4.13 Полярні координати . . . 951
11.4.14 Модальні групи . . . 953
11.4.15 Коментарі . . . 955
11.4.16 Повідомлення . . . 955
11.4.17 Реєстрація зонда . . . 956
11.4.18 Підсозаготівля . . . 956
11.4.19 Повідомлення про скасування . . . 956
11.4.20 Повідомлення про налагодження . . . 956
11.4.21 Друк повідомлень . . . 956
11.4.22 Параметри коментарів . . . 956
11.4.23 Вимоги до файлу . . . 957
11.4.24 Розмір файлу . . . 957
11.4.25 Порядок виконання G-коду . . . 957
11.4.26 Найкращі практики G-коду . . . 958
11.4.27 Лінійна та поворотна вісь . . . 959
11.4.28 Поширені повідомлення про помилки . . . 959
11.5 G-Коди . . . 960
11.5.1 Конвенції . . . 960
11.5.2 Таблиця швидкого довідника G-коду . . . 960
11.5.3 G0 Швидкий рух . . . 962
11.5.3.1 Швидка швидкість . . . 962
11.5.4 Лінійний рух G1 . . . 963
11.5.5 G2, G3 Дуговий рух . . . 963
11.5.5.1 Дуги формату центру . . . 964
11.5.5.2 Приклади формату центру . . . 966
11.5.5.3 Формат радіуса дуг . . . 968
11.5.6 G4 Житло . . . 969
11.5.7 Кубічний сплайн G5 . . . 969
11.5.8 G5.1 Квадратний сплайн . . . 970
11.5.9 G5.2 G5.3 Блок NURBS . . . 971
11.5.10 G7 Режим діаметра токарного верстата . . . 972
11.5.11 G8 Режим радіуса токарного верстата . . . 972
11.5.12 Дані таблиці інструментів для перезавантаження G10 L0 . . . 973
11.5.13 Таблиця інструментів G10 L1 . . . 973

11.5.14 G10 L2 Встановити систему координат . . . 974
11.5.15 Таблиця інструментів G10 L10 . . . 975
11.5.16 G10 L11 Набір інструментів Таблиця . . . 976
11.5.17 G10 L20 Встановити систему координат . . . 977
11.5.18 G17 - G19.1 Вибір площини . . . 977
11.5.19 Одиниці G20, G21 . . . 977
11.5.20 G28, G28.1 Перейти/Встановити попередньо визначене положення . . . 978
11.5.21 G30, G30.1 Перейти/Встановити попередньо визначену позицію . . . 978
11.5.22 G33 Синхронізований рух шпинделя . . . 979
11.5.23 G33.1 Жорстке нарізання різьби . . . 980
11.5.24 G38.*n* Прямий зонд . . . 981
11.5.25 Компенсація G40 вимкнена . . . 983
11.5.26 G41, G42 Компенсація різця . . . 984
11.5.27 G41.1, G42.1 Динамічна компенсація різця . . . 984
11.5.28 G43 Зміщення довжини інструменту . . . 985
11.5.29 G43.1 Динамічне зміщення довжини інструменту . . . 986
11.5.30 G43.2 Застосувати додаткове зміщення довжини інструменту . . . 986
11.5.31 G49 Скасувати компенсацію довжини інструменту . . . 988
11.5.32 Зміщення локальної системи координат G52 . . . 988
11.5.33 G53 Переміщення в координатах машини . . . 988
11.5.34 G54-G59.3 Вибір системи координат . . . 988
11.5.35 G61 Режим точного шляху . . . 989
11.5.36 G61.1 Режим точної зупинки . . . 989
11.5.37 G64 Змішування контурів . . . 989
11.5.38 Цикл чистової обробки токарного верстата G70 . . . 993
11.5.39 G71 G72 Цикли чорнової обробки токарного верстата . . . 994
11.5.40 Цикл свердління G73 зі стружколомленням . . . 995
11.5.41 G74 Цикл нарізання різьби ліворуч із затримкою . . . 996
11.5.42 Цикл нарізання різьби G76 . . . 997
11.5.43 Стандартні цикли G80-G89 . . . 999
11.5.43.1 Загальні слова . . . 1000
11.5.43.2 Липкі слова . . . 1000
11.5.43.3 Повторення циклу . . . 1000
11.5.43.4 Режим втягування . . . 1000
11.5.43.5 Помилки стандартного циклу . . . 1000
11.5.43.6 Попереднє та проміжне клопотання . . . 1001
11.5.43.7 Навіщо використовувати фіксований цикл? . . . 1001
11.5.44 G80 Скасувати стандартний цикл . . . 1003
11.5.45 Цикл свердління G81 . . . 1004

11.5.46 Цикл свердління G82, затримка . . . 1009
11.5.47 Цикл свердління з відведенням зубців G83 . . . 1009
11.5.48 G84 Цикл нарізання різьби праворуч, затримка . . . 1010
11.5.49 Цикл розточування G85, вихідна подача . . . 1010
11.5.50 G86 Цикл розточування, зупинка шпинделя, швидкий вихід . . . 1011
11.5.51 Цикл зворотного розточування G87 . . . 1011
11.5.52 G88 Цикл розточування, зупинка шпинделя, ручний вихід . . . 1011
11.5.53 G89 Цикл розточування, затримка, вихідна подача . . . 1011
11.5.54 G90, G91 Режим відстані . . . 1012
11.5.55 G90.1, G91.1 Режим дугової відстані . . . 1012
11.5.56 Зміщення системи координат G92 . . . 1012
11.5.57 G92.1, G92.2 Скидання зміщень G92 . . . 1013
11.5.58 G92.3 Відновлення зміщень G92 . . . 1013
11.5.59 Режим швидкості подачі G93, G94, G95 . . . 1014
11.5.60 Режим керування шпинделем G96, G97 . . . 1014
11.5.61 G98, G99 Рівень повернення стандартного циклу . . . 1015
11.6 M-Коди . . . 1015
11.6.1 Таблиця швидкого довідника M-кодів . . . 1015
11.6.2 M0, M1 Пауза програми . . . 1016
11.6.3 Кінець програми M2, M30 . . . 1016
11.6.4 Пауза зміни піддону M60 . . . 1017
11.6.5 M3, M4, M5 Шпиндельне керування . . . 1017
11.6.6 Зміна інструменту M6 . . . 1018
11.6.6.1 Ручна зміна інструменту . . . 1018
11.6.6.2 Зміна інструменту . . . 1018
11.6.7 M7, M8, M9 Контроль охолоджувальної рідини . . . 1018
11.6.8 Шпиндель M19 Orient . . . 1019
11.6.9 Керування швидкістю та подачею M48, M49 . . . 1019
11.6.10 Керування корекцією подачі M50 . . . 1020
11.6.11 M51 Керування корекцією швидкості шпинделя . . . 1020
11.6.12 Адаптивне керування подачею M52 . . . 1020
11.6.13 M53 Керування зупинкою подачі . . . 1020
11.6.14 M61 Встановити поточний інструмент . . . 1020
11.6.15 M62 - M65 Цифрове керування виходом . . . 1021
11.6.16 M66 Очікування введення . . . 1021
11.6.17 Синхронізований аналоговий вихід M67 . . . 1022
11.6.18 Аналоговий вихід M68, негайний . . . 1022
11.6.19 M70 Зберегти стан модального вікна . . . 1023
11.6.20 M71 Анулювання збереженого модального стану . . . 1024

11.6.21 M72 Відновлення модального стану . . . 1024
11.6.22 M73 Збереження та автовідновлення модального стану . . . 1025
11.6.23 M98 і M99 . . . 1026
11.6.23.1 Вибіркове відновлення модального стану . . . 1026
11.6.24 Команди, визначені користувачем M100-M199 . . . 1027
11.7 Коди O . . . 1029
11.7.1 Використання O-кодів . . . 1029
11.7.2 Нумерація . . . 1029
11.7.3 Коментарі . . . 1029
11.7.4 Підпрограми . . . 1030
11.7.4.1 Нумеровані програми в стилі Fanuc . . . 1031
11.7.5 Цикл . . . 1033
11.7.6 Умовний . . . 1034
11.7.7 Повторити . . . 1035
11.7.8 Непрямий . . . 1035
11.7.9 Виклик файлів . . . 1036
11.7.10 Значення, що повертаються підпрограмою . . . 1036
11.7.11 Помилки . . . 1037
11.8 Інші коди . . . 1037
11.8.1 F: Встановити швидкість подачі . . . 1037
11.8.2 S: Встановлення швидкості шпинделя . . . 1037
11.8.3 T: Вибрати інструмент . . . 1038
11.9 Приклади G-коду . . . 1038
11.9.1 Приклади млинів . . . 1038
11.9.1.1 Фрезерування гвинтових отворів . . . 1038
11.9.1.2 Пазування . . . 1039
11.9.1.3 Сітковий зонд . . . 1039
11.9.1.4 Розумний зонд . . . 1040
11.9.1.5 Зонд довжини інструменту . . . 1041
11.9.1.6 Зонд для отворів . . . 1041
11.9.1.7 Компенсація різця . . . 1041
11.9.2 Приклади токарних верстатів . . . 1042
11.9.2.1 Різьблення . . . 1042
11.10 Зображення в G-код . . . 1042
11.10.1 Що таке карта глибини? . . . 1042
11.10.2 Інтеграція зображення в G-код з інтерфейсом користувача AXIS . . . 1042
11.10.3 Використання `image-to-gcode` . . . 1043
11.10.4 Посилання на опцію . . . 1043
11.10.4.0 Одиниці . . . 1043

11.10.4.12 Інвертувати зображення . . . 1043
11.10.4.13 Нормалізувати зображення . . . 1043
11.10.4.14 Розгорнути межу зображення . . . 1043
11.10.4.15 Толерантність (одиниці) . . . 1043
11.10.4.16 Розмір пікселя (одиниці) . . . 1043
11.10.4.17 Швидкість подачі при зануренні (одиниць за хвилину) . . . 1044
11.10.4.18 Швидкість подачі (одиниць за хвилину) . . . 1044
11.10.4.19 Швидкість шпинделя (об/хв) . . . 1044
11.10.4.10 Шаблон сканування . . . 1044
11.10.4.11 Напрямок сканування . . . 1044
11.10.4.12 Глибина (одиниці) . . . 1044
11.10.4.13 Перехід (пікселі) . . . 1044
11.10.4.14 Діаметр інструменту . . . 1045
11.10.4.15 Безпечна висота . . . 1045
11.10.4.16 Тип інструменту . . . 1045
11.10.4.17 Мереживо облямівка . . . 1045
11.10.4.18 Кут контакту . . . 1045
11.10.4.19 Зміщення та глибина чорнової обробки за прохід . . . 1045
11.11 Відмінності RS274/NGC . . . 1046
11.11.1 Зміни з RS274/NGC . . . 1046
11.11.2 Доповнення до RS274/NGC . . . 1047

12 Віртуальна панель керування 1049

12.1 PyVCP . . . 1049
12.1.1 Вступ . . . 1049
12.1.2 Панельне будівництво . . . 1050
12.1.3 Безпека . . . 1051
12.1.4 AXIS . . . 1051
12.1.4.1 Приклад панелі . . . 1051
12.1.5 Окремо стояти . . . 1053
12.1.6 Віджети . . . 1054
12.1.6.1 Синтаксис . . . 1054
12.1.6.2 Загальні примітки . . . 1054
12.1.6.3 Мітка . . . 1055
12.1.6.4 Багатомітка . . . 1056
12.1.6.5 LEDs . . . 1056
12.1.6.6 Кпонки . . . 1058
12.1.6.7 Числові дисплеї . . . 1060
12.1.6.8 Числові вводи . . . 1063

12.1.6.9Зображення . . . 1066
12.1.6.10Контейнери . . . 1068
12.2Приклади PyVCP . . . 1073
12.2.1AXIS . . . 1073
12.2.2Плаваючі панелі . . . 1074
12.2.3Приклад кнопок поворотного перемикання . . . 1074
12.2.3.1Створення віджетів . . . 1075
12.2.3.2Встановлюйте зв’язки . . . 1077
12.2.4Порт-тестер . . . 1078
12.2.5Вимірювач обертів GS2 . . . 1081
12.2.5.1Панель . . . 1081
12.2.5.2Зв’язки . . . 1083
12.2.6Кнопка швидкого переходу додому . . . 1084
12.3GladeVCP: Віртуальна панель керування Glade . . . 1085
12.3.1Що таке GladeVCP? . . . 1085
12.3.1.1PyVCP проти GladeVCP: короткий огляд . . . 1085
12.3.2Короткий огляд панелі-прикладу . . . 1086
12.3.2.1Огляд прикладу панелі . . . 1089
12.3.2.2Огляд опису інтерфейсу користувача . . . 1090
12.3.2.3Вивчення зворотного виклику Python . . . 1090
12.3.3Створення та інтеграція користувацького інтерфейсу Glade . . . 1090
12.3.3.1Передумова: встановлення Glade . . . 1090
12.3.3.2Запуск Glade для створення нового інтерфейсу користувача . . . 1091
12.3.3.3Тестування панелі . . . 1092
12.3.3.4Підготовка файлу команд HAL . . . 1092
12.3.3.5Інтеграція в AXIS, як-от PyVCP . . . 1092
12.3.3.6Вбудовування як вкладка . . . 1093
12.3.3.7Інтеграція в Touchy . . . 1094
12.3.3.8Завантаження вбудованих панелей . . . 1095
12.3.4Параметри командного рядка GladeVCP . . . 1095
12.3.5Розуміння процесу запуску GladeVCP . . . 1097
12.3.6Довідник з віджетів HAL . . . 1098
12.3.6.1Найменування віджетів та пінів HAL . . . 1098
12.3.6.2Атрибути та методи Python віджетів HAL . . . 1099
12.3.6.3Налаштування значень піна та віджета . . . 1099
12.3.6.4Сигнал зміни виводу Hal . . . 1100
12.3.6.5Кпонки . . . 1100
12.3.6.6Масштаби . . . 1101
12.3.6.7SpinButton . . . 1102

12.3.6.8 Hal_Dial . . . 1102
12.3.6.9 Джог-колесо . . . 1104
12.3.6.10 Контроль швидкості . . . 1106
12.3.6.11 Мітка . . . 1108
12.3.6.12 Контейнери . . . 1109
12.3.6.13 LED . . . 1110
12.3.6.14 Індикатор прогресу . . . 1111
12.3.6.15 Комбінований список . . . 1111
12.3.6.16 Бари . . . 1112
12.3.6.17 Метр . . . 1113
12.3.6.18 HAL_Graph . . . 1114
12.3.6.19 Попередній перегляд траєкторії інструменту Gremlin для файлів NGC 1115
12.3.6.20 HAL_Offset . . . 1117
12.3.6.21 DRO віджет . . . 1118
12.3.6.22 Віджет Combi_DRO . . . 1120
12.3.6.23 IconView (Вибір файлу) . . . 1124
12.3.6.24 Віджет калькулятора . . . 1127
12.3.6.25 Віджет Tooleditor . . . 1127
12.3.6.26 Зсувна сторінка . . . 1129
12.3.6.27 Віджет HAL_sourceview . . . 1131
12.3.6.28 Історія MDI . . . 1132
12.3.6.29 Анімовані діаграми функцій: віджети HAL у растровому зображенні . . . 1133
12.3.7 Довідник з віджетів дій . . . 1133
12.3.7.1 Віджети дій VCP . . . 1134
12.3.7.2 VCP Action Python . . . 1134
12.3.7.3 Віджети VCP ToggleAction . . . 1135
12.3.7.4 Віджети Action_MDI Toggle та Action_MDI . . . 1136
12.3.7.5 Простий приклад: Виконання команди MDI при натисканні кнопки . . . 1136
12.3.7.6 Передача параметрів за допомогою віджетів Action_MDI та ToggleAction_MDI . . . 1136
12.3.7.7 Розширений приклад: передача параметрів підпрограмі з буквою ”O” 1137
12.3.7.8 Підготовка до дії MDI та очищення після неї . . . 1138
12.3.7.9 Використання об’єкта LinuxCNC Stat для обробки змін статусу . . . 1138
12.3.8 Програмування GladeVCP . . . 1139
12.3.8.1 Дії, визначені користувачем . . . 1139
12.3.8.2 Основна бібліотека . . . 1139
12.3.8.3 Приклад: додавання користувацьких зворотних викликів у Python . . . 1140
12.3.8.4 Події зміни значення HAL . . . 1140
12.3.8.5 Модель програмування . . . 1141

12.3.8.6 Послідовність ініціалізації . . . 1142
12.3.8.7 Кілька зворотних викликів з однаковим ім’ям . . . 1143
12.3.8.8 Прапорець GladeVCP `-U <користувачі>` . . . 1143
12.3.8.9 Постійні змінні в GladeVCP . . . 1144
12.3.8.10 Використання постійних змінних . . . 1144
12.3.8.11 Збереження стану під час вимкнення GladeVCP . . . 1146
12.3.8.12 Збереження стану при натисканні Ctrl-C . . . 1146
12.3.8.13 Ручне редагування INI-файлів (.ini) . . . 1147
12.3.8.14 Додавання контактів HAL . . . 1147
12.3.8.15 Додавання таймерів . . . 1147
12.3.8.16 Програмне налаштування властивостей віджета HAL . . . 1148
12.3.8.17 Зворотний виклик зі зміною значення за допомогою hal_glib . . . 1148
12.3.8.18 Приклади та створення власної програми GladeVCP . . . 1149
12.3.9 Найчастіші запитання . . . 1149
12.3.10 Усунення несправностей . . . 1150
12.3.11 Примітка щодо впровадження: Обробка ключів в AXIS . . . 1150
12.3.12 Додавання користувацьких віджетів . . . 1151
12.3.13 Допоміжні програми GladeVCP . . . 1151
12.4 Модулі бібліотеки GladeVCP . . . 1151
12.4.1 Інформація . . . 1152
12.4.2 Дія . . . 1154
12.5 QtVCP . . . 1156
12.5.1 Вітрина . . . 1156
12.5.2 Огляд . . . 1162
12.5.2.1 Віджети QtVCP . . . 1163
12.5.2.2 Налаштування INI . . . 1163
12.5.2.3 Файл інтерфейсу Qt Designer . . . 1164
12.5.2.4 Файли обробників . . . 1165
12.5.2.5 Модулі бібліотек . . . 1165
12.5.2.6 Теми . . . 1165
12.5.2.7 Локальні файли . . . 1166
12.5.2.8 Зміна стокових екранів . . . 1166
12.5.3 Панелі VCP . . . 1171
12.5.3.1 Вбудовані панелі . . . 1171
12.5.3.2 Спеціальні панелі . . . 1175
12.5.4 Створіть простий користувацький екран з чистого аркуша . . . 1177
12.5.4.1 Огляд . . . 1177
12.5.4.2 Включити віджети LinuxCNC у Qt Designer . . . 1177
12.5.4.3 Беріть екранний файл `.ui` . . . 1178

12.5.4.4Файл обробника . . . 1181
12.5.4.5Конфігурація INI . . . 1181
12.5.5Детальний опис файлу обробника . . . 1181
12.5.5.1Огляд . . . 1182
12.5.5.2Розділ ІМПОРТ . . . 1186
12.5.5.3Розділ «СТВОРЕННЯ ЕКЗЕМПЛЯРІВ БІБЛІОТЕК» . . . 1187
12.5.5.4Розділ класу обробника . . . 1187
12.5.5.5Розділ INICIALIZE . . . 1187
12.5.5.6Розділ СПЕЦІАЛЬНИХ ФУНКЦІЙ . . . 1187
Bibliography . . . 1188
12.5.5.7Розділ зворотних викликів стану . . . 1188
12.5.5.8ЗВОРОТНІ ВИКЛИКИ З РОЗДІЛУ ФОРМИ . . . 1188
12.5.5.9Розділ ЗАГАЛЬНИХ ФУНКЦІЙ . . . 1188
12.5.5.10Розділ ПРИВ’ЯЗКИ КЛАВІШ . . . 1189
12.5.5.11Розділ ЗАКРИТТЯ ПОДІЇ . . . 1189
12.5.6Підключення віджетів до коду Python . . . 1189
12.5.6.1Огляд . . . 1189
12.5.6.2Використання Qt Designer для додавання слотів . . . 1190
12.5.6.3Зміни обробника Python . . . 1191
12.5.7Більше інформації . . . 1192
12.6Віртуальні панелі керування QtVCP . . . 1192
12.6.1Вбудовані віртуальні панелі керування . . . 1192
12.6.1.1`копія` . . . 1192
12.6.1.2spindle_belts . . . 1193
12.6.1.3`test_dial` . . . 1195
12.6.1.4test_button . . . 1196
12.6.1.5`test_led` . . . 1196
12.6.1.6`test_panel` . . . 1197
12.6.1.7`cam_align` . . . 1198
12.6.1.8`sim_panel` . . . 1201
12.6.1.9`tool_dialog` . . . 1202
12.6.2`vismach` Панелі 3D-моделювання . . . 1203
12.6.2.1QtVCP `vismach_mill_xyz` . . . 1203
12.6.2.2QtVCP `vismach_router_atc` . . . 1204
12.6.2.3QtVCP `vismach_scara` . . . 1205
12.6.2.4QtVCP `vismach_millturn` . . . 1206
12.6.2.5QtVCP `vismach_mill_5axis_gantry` . . . 1207
12.6.2.6QtVCP `vismach_fanuc_200f` . . . 1208
12.6.3Користувацькі віртуальні панелі керування . . . 1209

12.6.4 Вбудовування віртуальних панелей керування QtVCP в екрани QtVCP 1210
12.6.4.1 Команди вбудовування 1210
12.6.4.2 Розташування вбудованих панелей 1210
12.6.4.3 Розташування користувацьких панелей 1211
12.6.4.4 Поради щодо програмування обробників 1211
12.6.4.5 Поради щодо дизайнерських віджетів 1211
12.6.4.6 Патчування обробників - створення підкласів вбудованих панелей . . . 1212
12.7 Віджети QtVCP 1213
12.7.1 Віджети лише для HAL 1213
12.7.1.1 `CheckBox` Віджет 1213
12.7.1.2 *DetachTabWidget* - Віджет контейнера зі знімними користувачем панелями 1213
12.7.1.3 `DoubleScale` - Віджет введення кнопки спінінга 1213
12.7.1.4 `FocusOverlay` - Віджет накладання фокуса 1214
12.7.1.5 `Gauge` - Віджет круглого циферблатного індикатора 1215
12.7.1.6 `GeneralHALInput` - Віджет підключення вхідних сигналів/слотів 1216
12.7.1.7 `GeneralHALOutput` - Загальні сигнали/слоти Віджет підключення виходу 1216
12.7.1.8 `GridLayout` - Віджет макета сітки 1217
12.7.1.9 *HalBar* - Індикатор рівня HAL Bar 1217
12.7.1.10 `HALPad` - Кнопки HAL Джойстик 1218
12.7.1.11 `HALLabel` - Віджет міток HAL 1220
12.7.1.12 `LCDNumber` - Віджет зчитування номерів у стилі РК-дисплея 1221
12.7.1.13 `LED` - Віджет індикатора 1221
12.7.1.14 `PushButton` - Віджет перемикання закріплення HAL 1222
12.7.1.15 `RadioButton` Віджет 1222
12.7.1.16 `Slider` - Віджет налаштування значення HAL Pin 1223
12.7.1.17 `TabWidget` - Віджет вкладки 1223
12.7.1.18 `WidgetSwitcher` - Віджет перемикача режимів перегляду макета кількох віджетів 1223
12.7.1.19 `XEmbed` - Віджет вбудовування програм 1224
12.7.2 Віджети контролера машини 1224
12.7.2.1 `ActionButton` - Віджет керування діями контролера машини 1224
12.7.2.2 `ActionToolButton` - Віджет кнопки меню додаткових дій 1227
12.7.2.3 `AxisToolButton` - Віджет «Вибрати та встановити осі» 1227
12.7.2.4 `BasicProbe` - Простий віджет зондування фрезерування 1229
12.7.2.5 `CamView` - Віджет вирівнювання заготовки та налаштування початку координат 1229
12.7.2.6 `DROLabel` - Віджет відображення положення осі 1229
12.7.2.7 `FileManager` - Віджет вибору завантаження файлів 1232
12.7.2.8 `GcodeDisplay` - Віджет відображення тексту G-коду 1233

12.7.2.9 GcodeEditor - Віджет редактора G-кодів програм 1234
12.7.2.10 GCodeGraphics - Віджет графічного фону G-коду 1235
12.7.2.11 JointEnableWidget - FIXME 1239
12.7.2.12 LogIncrements - Віджет вибору значення кроку поштовху 1240
12.7.2.13 MacroTab - Віджет спеціальних макросів 1240
12.7.2.14 OperatorValueLine - Віджет введення рядка значення оператора . . . 1243
12.7.2.15 MDILine - Віджет введення команд MDI в рядок 1244
12.7.2.16 MDIHistory - Віджет історії команд MDI 1245
12.7.2.17 MDITouchy - Віджет введення MDI на сенсорному екрані 1245
12.7.2.18 OriginOffsetView - Віджет перегляду та налаштувань походження . . 1247
12.7.2.19 RadioAxisSelector - FIXME 1248
12.7.2.20 RoundButton - Віджет ActionButton круглої форми 1249
12.7.2.21 StateLabel - Віджет відображення міток станів режимів контролера . 1249
12.7.2.22 StatusLabel - Віджет відображення міток стану змінних контролера . 1249
12.7.2.23 StatusImageSwitcher - Перемикач зображень стану контролера 1252
12.7.2.24 StatusStacked - Віджет перемикання режимів відображення стану . . 1254
12.7.2.25 ScreenOption - Віджет налаштувань загальних параметрів 1254
12.7.2.26 StatusSlider - Віджет повзунка налаштування контролера 1259
12.7.2.27 StateLED - Віджет світлодіодного індикатора стану контролера 1261
12.7.2.28 StatusAdjustmentBar - Віджет налаштування значень контролера . . . 1262
12.7.2.29 SystemToolButton - Віджет вибору системи користувача 1262
12.7.2.30 StateEnableGridlayout - Віджет контейнера з увімкненим станом контролера 1263
12.7.2.31 StatusImageSwitcher - Віджет перемикання зображень стану контролера 1263
12.7.2.32 ToolOffsetView - Інструменти Зміщення Перегляд та редагування Віджет 1263
12.7.2.33 VersaProbe - Віджет зондування фрезерування 1265
12.7.3 Віджети діалогових вікон 1265
12.7.3.1 LcncDialog - Віджет загального діалогового вікна повідомлень 1266
12.7.3.2 ToolDialog - Віджет діалогового вікна ручної зміни інструментів 1267
12.7.3.3 FileDialog - Віджет діалогу вибору файлів для завантаження та збереження 1268
12.7.3.4 OriginOffsetDialog - Віджет діалогового вікна налаштування зміщення походження 1269
12.7.3.5 ToolOffsetDialog - Віджет діалогового вікна налаштування зміщення інструмента 1270
12.7.3.6 ToolChooserDialog - Віджет діалогового вікна вибору інструментів . . 1271
12.7.3.7 MachineLog - Віджет відображення журналу подій машини 1271
12.7.3.8 MacroTabDialog - Віджет діалогового вікна запуску макросів 1273
12.7.3.9 CamViewDialog - Віджет діалогового вікна вирівнювання деталей веб-камери 1273
12.7.3.10 EntryDialog - Віджет діалогового вікна редагування рядка 1273
12.7.3.11 CalculatorDialog - Віджет діалогового вікна калькулятора 1273

12.7.3.12 `RunFromLine` - Віджет діалогового вікна «Запустити з рядка» 1275
12.7.3.13 `VersaProbeDialog` - Віджет діалогового вікна дотику до деталі 1276
12.7.3.14 `MachineLogDialog` - Віджет діалогового вікна журналів машини та налагодження1
12.7.4 Інші віджети 1277
12.7.4.1 `NurbsEditor` - Віджет редагування NURBS 1278
12.7.4.2 `JoyPad` - 5-кнопковий віджет D-pad 1278
12.7.4.3 *WebWidget* 1281
12.7.5 Віджети BaseClass/Mixin 1281
12.7.5.1 `IndicatedPushButtons` 1281
12.7.6 Віджети лише для імпорту 1284
12.7.6.1 Автоматична висота 1284
12.7.6.2 Утиліта G-коду 1284
12.7.6.3 Облицювання 1285
12.7.6.4 Коло отворів 1285
12.7.6.5 Збільшити отвір 1286
12.7.6.6 Qt NGCGUI 1286
12.7.6.7 Qt PDF 1288
12.7.6.8 Qt Vismach 1288
12.7.6.9 Коробка вибору Hal 1288
12.8 Модулі бібліотек QtVCP 1289
12.8.1 `Status` 1289
12.8.1.1 Застосування 1289
12.8.1.2 Приклад 1290
12.8.2 `Info` 1290
12.8.2.1 Доступні дані та значення за замовчуванням 1290
12.8.2.2 Інформація про діалогове вікно повідомлення користувача 1291
12.8.2.3 Інформація про вбудовану програму 1291
12.8.2.4 Помічники 1292
12.8.2.5 Застосування 1292
12.8.3 `Action` 1292
12.8.3.1 Помічники 1293
12.8.3.2 Застосування 1293
12.8.4 Калал 1295
12.8.4.1 Атрибути 1295
12.8.4.2 Константи 1296
12.8.4.3 Посилання 1296
12.8.5 QPin 1296
12.8.5.1 Сигнали 1296
12.8.5.2 Атрибути 1296

12.8.5.3 Посилання . . . 1297
12.8.5.4 Приклад . . . 1297
12.8.6 Tool . . . 1297
12.8.6.1 Помічники . . . 1297
12.8.7 Path . . . 1298
12.8.7.1 Шляхи, на які посилаються . . . 1299
12.8.7.2 Помічники . . . 1300
12.8.7.3 Застосування . . . 1300
12.8.8 VCPWindow . . . 1300
12.8.8.1 Застосування . . . 1300
12.8.9 Aux_program_loader . . . 1301
12.8.9.1 Помічники . . . 1301
12.8.9.2 Застосування . . . 1302
12.8.10 Keylookup . . . 1302
12.8.10.1 Застосування . . . 1302
12.8.10.2 Ключові визначення . . . 1304
12.8.11 Messages . . . 1306
12.8.11.1 Властивості . . . 1306
12.8.11.2 Піни HAL . . . 1307
12.8.11.3 Приклади . . . 1307
12.8.12 Багатоповідомлення . . . 1308
12.8.12.1 Властивості . . . 1308
12.8.12.2 Приклади . . . 1309
12.8.13 Notify . . . 1309
12.8.13.1 Властивості . . . 1309
12.8.14 Preferences . . . 1310
12.8.15 Player . . . 1310
12.8.15.1 Звуки . . . 1310
12.8.15.2 Застосування . . . 1311
12.8.15.3 Приклад . . . 1311
12.8.16 Віртуальна клавіатура . . . 1311
12.8.17 Дії панелі інструментів . . . 1311
12.8.17.1 Дії . . . 1312
12.8.17.2 Підменю . . . 1312
12.8.17.3 Застосування . . . 1312
12.8.17.4 Приклади . . . 1312
12.8.18 Бібліотека машинної графіки Qt Vismach . . . 1313
12.8.18.1 Вбудовані зразки . . . 1313
12.8.18.2 Бібліотека примітивів . . . 1313

12.8.18.3 Застосування . . . 1315
12.8.18.4 Більше інформації . . . 1316
12.9 QtVismach . . . 1316
12.9.1 Вступ . . . 1316
12.9.2 Ієрархія проектування машин . . . 1318
12.9.3 Запустіть скрипт . . . 1319
12.9.4 Штифти HAL. . . . 1319
12.9.5 Створення деталей . . . 1319
12.9.5.1 Імпорт файлів STL або OBJ . . . 1319
12.9.5.2 Побудувати з геометричних примітивів . . . 1320
12.9.6 Переміщення частин моделі . . . 1321
12.9.6.1 Переклад деталей моделі . . . 1321
12.9.6.2 Обертові деталі моделі . . . 1321
12.9.7 Анімація деталей . . . 1321
12.9.7.1 `HalTranslate` . . . 1321
12.9.7.2 `HalRotate` . . . 1322
12.9.7.3 *HalToolCylinder* . . . 1322
12.9.7.4 *HalToolTriangle* . . . 1322
12.9.7.5 Налаштовувані примітиви HAL . . . 1323
12.9.8 Збірка моделі . . . 1323
12.9.9 Інші функції . . . 1324
12.9.9.1 Колір . . . 1324
12.9.9.2 HALColorFlip . . . 1324
12.9.9.3 HALColorRGB . . . 1325
12.9.9.4 Проекційний дисплей . . . 1325
12.9.9.5 Проекційний дисплей HAL . . . 1325
12.9.9.6 ПриховатиКолекцію . . . 1326
12.9.9.7 Колір графіка залежно від типу руху . . . 1326
12.9.9.8 Захоплення . . . 1327
12.9.9.9 головний . . . 1327
12.9.10 Поради . . . 1328
12.9.11 Базова структура скрипта `QtVismach` . . . 1328
12.9.12 Вбудовані зразки панелей Vismach . . . 1330
12.10 QtVCP: Створення власних віджетів . . . 1330
12.10.0 Огляд . . . 1330
12.10.1.1 Віджети . . . 1330
12.10.1.2 Дизайнер Qt . . . 1330
12.10.1.3 Процес ініціалізації . . . 1331
12.10.1.4 Процес очищення . . . 1331

12.10.2 Користувацькі віджети HAL . . . 1332
12.10.3 Віджети користувацького контролера з використанням STATUS . . . 1333
12.10.3.1 У розділі «Імпорт» . . . 1334
12.10.3.2 У розділі «Створення миттєвих бібліотек» . . . 1335
12.10.3.3 У розділі «Визначення класу користувацького віджета» . . . 1335
12.10.4 Віджети налаштованого контролера з діями . . . 1338
12.10.5 Зміни властивостей таблиці стилів на основі подій . . . 1340
12.10.6 Використання таблиць стилів для зміни властивостей власного віджета . . . 1341
12.10.7 Плагіни віджетів . . . 1341
12.10.7.1 Приклад сітки . . . 1341
12.10.7.2 Приклад кнопки SystemTool . . . 1342
12.10.7.3 Створення плагіна з діалоговим вікном MenuEntry . . . 1343
12.11 Фрагменти коду файлу обробника QtVCP . . . 1346
12.11.1 Завантаження/збереження файлу налаштувань . . . 1346
12.11.2 Використання QSettings для читання/збереження змінних . . . 1347
12.11.3 Додати базовий редактор стилів . . . 1348
12.11.4 Запит на запис у діалоговому вікні . . . 1348
12.11.5 Промовте привітання стартапу . . . 1349
12.11.6 Функції панелі інструментів . . . 1350
12.11.7 Додати HAL-виводи, що викликають функції . . . 1351
12.11.8 Безпосереднє читання/запис системних HAL-виводів . . . 1352
12.11.9 Додати спеціальний повзунок максимальної швидкості на основі відсотків . . . 1352
12.11.10 Вимкнення та вимкнення безперервного поштовху . . . 1353
12.11.11 Патч класу Віджет файлового менеджера . . . 1354
12.11.12 Додавання віджетів програмним способом . . . 1356
12.11.13 Періодично оновлювати/читати об’єкти . . . 1360
12.11.14 Зовнішнє керування за допомогою ZMQ . . . 1361
12.11.14.1 Читання повідомлень ZMQ . . . 1361
12.11.14.2 Записання повідомлень ZMQ . . . 1363
12.11.15 Надсилання повідомлень у рядок стану або діалогові вікна сповіщень на робочому столі . . . 1364
12.11.16 Зміни фокусу . . . 1365
12.11.17 Параметри часу завантаження командного рядка читання . . . 1365
12.11.18 Код для зчитування налаштувань Qt . . . 1367
12.12 Розробка QtVCP . . . 1367
12.12.1 Огляд . . . 1367
12.12.2 Збудовані місця розташування . . . 1368
12.12.3 Запуск QtVCP для вимкнення . . . 1368
12.12.3.1 Запуск QtVCP . . . 1368

12.12.3.3 Вимкнення QtVCP . . . 1369
12.12.4 Інформація про шлях . . . 1369
12.12.5 Ідіосинкразії . . . 1369
12.12.5.1 Збір кодів помилок . . . 1370
12.12.5.2 Швидкість поштовху . . . 1370
12.12.5.3 Прив’язка клавіш . . . 1370
12.12.5.4 Файл налаштувань . . . 1370
12.12.5.5 Функції спеціального налаштування віджета . . . 1371
12.12.5.6 Діалоги . . . 1371
12.12.5.7 Стилі (теми) . . . 1371

13 Програмування інтерфейсу користувача 1372

13.1 Панель . . . 1372
13.1.1 Вступ . . . 1372
13.1.2 Команди завантаження . . . 1372
13.1.3 Посилання на файл panelui.ini . . . 1374
13.1.4 Довідник внутрішніх команд . . . 1378
13.1.5 Повідомлення ZMQ . . . 1380
13.1.6 Розширення файлу обробника . . . 1381
13.2 Модуль LinuxCNC на Python . . . 1383
13.2.1 Вступ . . . 1383
13.2.2 Шаблони використання інтерфейсу LinuxCNC NML . . . 1383
13.2.3 Зчитування стану LinuxCNC за допомогою модуля linuxcnc Python . . . 1384
13.2.3.1 Атрибути linuxcnc.stat . . . 1384
13.2.3.2 Словник «axis» . . . 1390
13.2.3.3 Словник «joint» . . . 1390
13.2.3.4 Словник «веретено» . . . 1392
13.2.4 Підготовка до надсилання команд . . . 1392
13.2.5 Надсилання команд через `linuxcnc.command` . . . 1393
13.2.5.1 `linuxcnc.command` атрибути . . . 1394
13.2.5.2 `linuxcnc.command` методи: . . . 1394
13.2.6 Зчитування каналу помилок . . . 1398
13.2.7 Читання значень INI-файлу . . . 1398
13.2.8 Тип `linuxcnc.positionlogger` . . . 1399
13.2.8.1 члени . . . 1399
13.2.8.2 методи . . . 1399
13.3 Модуль HAL для Python . . . 1400
13.3.1 Базове використання . . . 1400
13.3.2 Функції . . . 1400

13.4 Модуль Python GStat . . . 1403
13.4.1 Вступ . . . 1403
13.4.2 Зразок коду GStat . . . 1404
13.4.2.1 Зразок шаблону коду компонента HAL . . . 1404
13.4.2.2 Шаблон коду розширення Python GladeVCP . . . 1405
13.4.2.3 Шаблон коду розширення Python QtVCP . . . 1406
13.4.3 Повідомлення . . . 1406
13.4.4 Функції . . . 1414
13.4.5 Відомі проблеми . . . 1416
13.5 Vismach . . . 1416
13.5.1 Запустіть скрипт . . . 1418
13.5.2 Створіть контакти HAL. . . . 1418
13.5.3 Створення деталей . . . 1418
13.5.4 Рухомі частини . . . 1419
13.5.5 Анімація деталей . . . 1419
13.5.6 Збірка моделі. . . . 1420
13.5.7 Інші функції . . . 1421
13.5.8 Базова структура скрипту Vismach. . . . 1422

III Глосарій, авторське право та історія 1423

14 Зворотній бік 1424

15 Глосарій 1425

16 Авторське право 1431
16.1 Юридичний відділ . . . 1431
16.1.1 Умови авторського права . . . 1431
16.1.2 Ліцензія GNU Free Documentation . . . 1431

17 Історія LinuxCNC 1437
17.1 Походження . . . 1437
17.1.1 Зміна імені . . . 1438
17.1.2 Додаткова інформація . . . 1438

Part I

Початок роботи і конфігурація

Chapter 1

Початок роботи з LinuxCNC

1.1 Про 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: * легке виявлення та тестування без встановлення за допомогою Live Image, * легке встановлення з Live-образу, * прості у використанні графічні майстри конфігурації для швидкого створення конфігурації, специфічної для машини, * 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 * інструменти для створення графічних інтерфейсів (Glade, Qt), * the interpreter for *G-code* (the RS-274 machine tool programming language), * робота низькорівневої машинної електроніки, такої як датчики та приводи двигунів, * простий у використанні шар «макетної плати» для швидкого створення унікальної конфігурації вашої машини, * програмний PLC, програмований за допомогою сходинок.

Under the hood, LinuxCNC provides * система планування руху в реальному часі з прогнозуванням, * 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. * Функції управління рухом включають: компенсацію радіуса та довжини різака, відхилення траєкторії, обмежене заданим допуском, нарізування різьби на токарному верстаті, синхронізований рух осей, адаптивну швидкість подачі, ручне управління подачею оператором та контроль постійної швидкості. * 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

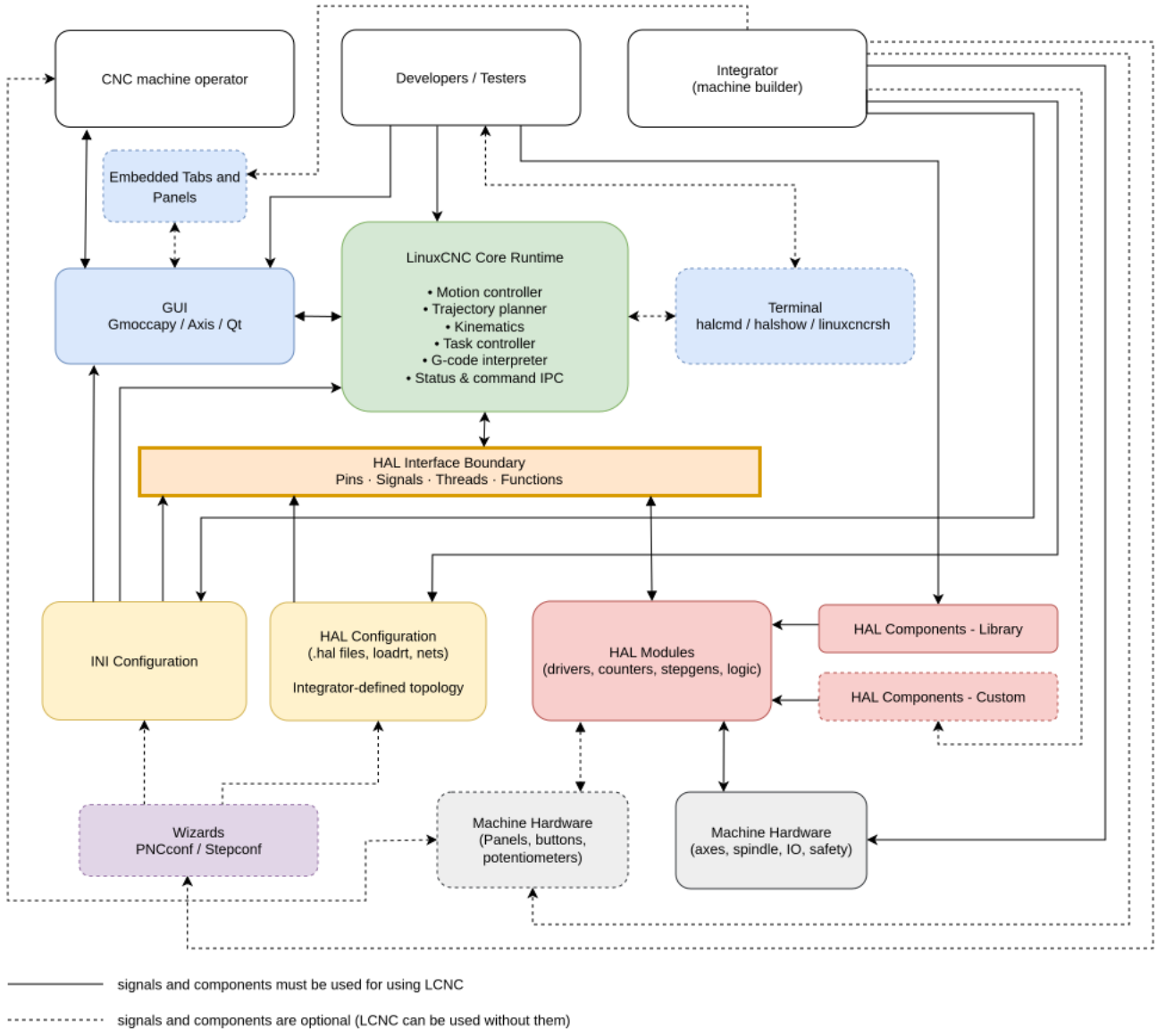


Figure 1.1: Roles of operators, integrators, developers and hardware

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.

Інтегратор
: 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 Операційна система

LinuxCNC доступний у вигляді готових до використання пакетів для дистрибутивів Debian.

1.1.3 Отримання допомоги

1.1.3.1 Веб-форум

Веб-форум можна знайти за адресою https://forum.linuxcnc.org або за посиланням у верхній частині головної сторінки linuxcnc.org.

Це досить активно, але демографічна група більше орієнтована на користувачів, ніж список розсилки. Якщо ви хочете бути впевнені, що розробники побачать ваше повідомлення, тоді слід віддати перевагу списку розсилки.

1.1.3.2 IRC

IRC розшифровується як Internet Relay Chat (Інтернет-релейний чат). Це живе з'єднання з іншими користувачами LinuxCNC. IRC-канал LinuxCNC на libera.chat — #linuxcnc.

Найпростіший спосіб потрапити на IRC – це скористатися вбудованим веб-клієнтом від libera.

Трохи етикету IRC

- Задавайте конкретні запитання… Уникайте запитань на кшталт «Хтось може мені допомогти?».
- Якщо ви справді новачок у всьому цьому, трохи подумайте над своїм запитанням, перш ніж писати його. Переконайтеся, що ви надаєте достатньо інформації, щоб хтось міг відповісти на ваше запитання або вирішити вашу проблему.
- Будьте терплячими, чекаючи на відповідь. Іноді формулювання відповіді займає деякий час, або ж усі можуть бути зайняті роботою чи чимось подібним.
- Налаштуйте свій обліковий запис IRC, використовуючи своє унікальне ім'я, щоб люди знали, хто ви. Якщо ви використовуєте клієнт Java, використовуйте те саме ім'я щоразу, коли входите в систему. Це допомагає людям запам'ятати, хто ви. Якщо ви вже були на ньому раніше, багато хто пам'ятатиме попередні розмови з вами, що заощадить час обом сторонам.

Спільний доступ до файлів

Найпоширеніший спосіб обміну файлами на IRC – це завантажити файл на один із наведених нижче або подібний сервіс і вставити посилання:

- *Для тексту*: https://pastebin.com/, https://gist.github.com/, https://0bin.net/, https://paste.debian.n
- *Для зображень*: https://imagebin.org/, https://imgur.com/, https://bayimg.com/
- *Для файлів*: https://filedropper.com/, https://filefactory.com/, https://1fichier.com/

1.1.3.3 Список розсилки

Інтернет-розсилка — це спосіб поставити питання, щоб усі учасники розсилки могли їх побачити та відповісти на них у зручний для них час. У розсилці ваші питання отримають більшу увагу, ніж в IRC, але відповіді надходять довше. Коротко кажучи, ви надсилаєте повідомлення на адресу розсилки і отримуєте щоденні дайджести або індивідуальні відповіді, залежно від того, як ви налаштували свій обліковий запис.

Ви можете підписатися на список розсилки emc-users за адресою: https://lists.sourceforge.net/-lists/listinfo/emc-users.

1.1.3.4 Веб-форум

Веб-форум можна знайти за адресою https://forum.linuxcnc.org/ або за посиланням у верхній частині головної сторінки https://linuxcnc.org/.

Це досить активно, але демографічна група більше орієнтована на користувачів, ніж список розсилки. Якщо ви хочете бути впевнені, що розробники побачать ваше повідомлення, тоді слід віддати перевагу списку розсилки.

1.1.3.5 Вікі LinuxCNC

Вікі-сайт — це веб-сайт, що підтримується користувачем, який будь-хто може доповнювати або редагувати.

Користувацький сайт LinuxCNC Wiki містить безліч інформації та порад за адресою: посилання: http://wiki.linuxcnc.org

1.1.3.6 Звіти про помилки

Повідомляйте про помилки LinuxCNC за посиланням Github: система відстеження помилок github.

1.2 Системні вимоги

1.2.1 Мінімальні вимоги

Мінімальна система для запуску LinuxCNC та Debian / Ubuntu може відрізнятися залежно від конкретного використання. Степперні системи, як правило, вимагають більш швидких потоків для генерації імпульсів кроку, ніж сервосистеми. Ви можете використовувати Live CD для тестування програмного забезпечення перед остаточним встановленням на комп'ютер. Майте на увазі, що для генерації кроків програмним забезпеченням більш важливими є показники тесту затримки, ніж швидкість процесора. Більше інформації про тест затримки можна знайти here. Крім того, LinuxCNC потрібно запускати на операційній системі, яка використовує спеціально модифікований ядро, див. Вимоги до ядра та версії.

Додаткову інформацію можна знайти на сайті вікі LinuxCNC: Вимоги до обладнання

LinuxCNC і Debian Linux повинні працювати досить добре на комп'ютері з наступними мінімальними технічними характеристиками. Ці цифри не є абсолютним мінімумом, але забезпечать достатню продуктивність для більшості крокових систем.

- 1,2 ГГц 64-бітний процесор x86 або Raspberry Pi 4 або краще.
- 512 МБ оперативної пам'яті, 4 ГБ з графічним інтерфейсом, щоб уникнути несподіванок
- Немає жорсткого диска для Live CD, 8 ГБ або більше для постійної інсталяції
- Відеокарта з роздільною здатністю щонайменше 1024x768, яка не використовує пропрієтарні драйвери NVidia або ATI fglrx. Сучасні вбудовані графічні чіпсети загалом підходять.
- Підключення до Інтернету (не є обов'язковим, але дуже корисним для оновлень та спілкування зі спільнотою LinuxCNC)

Мінімальні вимоги до обладнання змінюються разом із розвитком дистрибутивів Linux, тому перевірте веб-сайт Debian, щоб дізнатися більше про Live CD, який ви використовуєте. Для старішого обладнання може бути корисним вибрати старішу версію Live CD, якщо вона доступна.

Якщо ви не плануєте покладатися на розповсюдження готових до виконання програм («бінарних файлів») та/або маєте намір долучитися до розробки вихідного коду LinuxCNC, то, ймовірно, вам знадобиться другий комп'ютер для виконання компіляції. Незважаючи на те, що LinuxCNC і ваші розробки можуть виконуватися одночасно з точки зору дискового простору, оперативної пам'яті і навіть швидкості процесора, завантажена машина буде мати гірші затримки, тому ви навряд чи зможете одночасно компілювати дерево джерел і виробляти мікросхеми.

1.2.2 Вимоги до ядра та версії

LinuxCNC вимагає ядра, модифікованого для використання в реальному часі, для керування апаратним забезпеченням реальної машини. Однак, він може працювати на стандартному ядрі в режимі симуляції для таких цілей, як перевірка G-коду, тестування конфігураційних файлів та навчання системи. Для роботи з цими версіями ядра існують дві версії LinuxCNC, що розповсюджуються
Назви пакетів — «linuxcnc» та «linuxcnc-uspace».

Варіанти ядра реального часу: preempt-rt, RTAI та Xenomai.

Ви можете дізнатися версію ядра вашої системи за допомогою команди:

```
uname -a
```

Якщо ви бачите (як вище) `-rt-` в імені ядра, то ви використовуєте ядро preempt-rt і повинні встановити версію LinuxCNC «uspace». Ви також повинні встановити uspace для конфігурацій «sim» на ядрах, що не працюють в режимі реального часу.

Якщо в назві ядра ви бачите `-rtai-`, то ви використовуєте RTAI в реальному часі. Дивіться нижче версію для LinuxCNC, яку потрібно встановити.

1.2.2.1 Preempt-RT з пакетом *linuxcnc-uspace*

Preempt-RT — це найновіша з систем реального часу, а також версія, яка найближча до основного ядра. Ядра Preempt-RT доступні у вигляді попередньо скомпільованих пакетів з основних репозиторіїв. Їх можна знайти за пошуковим запитом «PREEMPT_RT», а завантажити та встановити — як і будь-який інший пакет. Preempt-RT зазвичай має найкращу підтримку драйверів і є єдиним варіантом для систем, що використовують драйверні карти Mesa з підключенням до мережі Ethernet. Загалом preempt-rt має найгіршу затримку серед доступних систем, але є винятки.

1.2.2.2 RTAI з пакетом «linuxcnc»

RTAI вже багато років є основою дистрибутивів LinuxCNC. Зазвичай він забезпечує найкращу продуктивність у режимі реального часу з точки зору низької затримки, але може мати гіршу підтримку периферійних пристроїв і не так багато роздільних здатностей екрану. Ядро RTAI доступне в репозиторії пакетів LinuxCNC. Якщо ви встановили систему з образу Live/Install, то перехід на інше ядро та версію LinuxCNC описано в розділі [Встановлення RTAI].

1.2.2.3 Xenomai з пакетом *linuxcnc-uspace*

Xenomai також підтримується, але вам доведеться знайти або зібрати ядро та скомпілювати LinuxCNC з вихідного коду, щоб використовувати його.

1.2.2.4 RTAI з пакетом *linuxcnc-uspace*

Також можна запустити LinuxCNC з RTAI в режимі простору користувача. Як і у випадку з Xenomai, для цього потрібно буде скомпілювати з вихідного коду.

1.2.3 Проблемне обладнання

1.2.3.1 Ноутбуки

Ноутбуки зазвичай не підходять для генерації кроків програмного забезпечення в реальному часі. Знову ж таки, тест затримки, запущений протягом тривалого часу, надасть вам інформацію, необхідну для визначення придатності.

1.2.3.2 Відеокарти

Якщо під час інсталяції з’являється вікно з роздільною здатністю екрана 800 x 600, то, найімовірніше, Debian не розпізнає вашу відеокарту або монітор. Іноді це можна вирішити, встановивши драйвери або створивши/відредагувавши файли Xorg.conf.

1.3 Отримання LinuxCNC

У цьому розділі описано рекомендований спосіб завантаження та встановлення нової версії LinuxCNC. Для сміливих користувачів також існують Альтернативні методи встановлення. Якщо ви маєте існуючу версію, яку хочете оновити, перейдіть до розділу Оновлення LinuxCNC.

Note
Для роботи з обладнанням LinuxCNC потребує спеціального ядра з розширеннями реального часу. Є три можливості: preempt-rt, RTAI або Xenomai. Крім того, існують дві версії LinuxCNC, які працюють з цими ядрами. Детальніше дивіться в таблиці нижче. Однак для тестування коду та моделювання можна запустити програму `linuxcnc-uspace` на стандартному ядрі дистрибутива.

Нові інсталяції LinuxCNC найпростіше створювати за допомогою Live/Install Image. Це гібридний образ файлової системи ISO, який можна записати на USB-накопичувач або DVD-диск і використовувати для завантаження комп'ютера. Під час завантаження вам буде запропоновано вибрати завантаження «Live»-системи (для запуску LinuxCNC без внесення постійних змін до вашого комп'ютера) або завантаження інсталятора (для встановлення LinuxCNC та його операційної системи на жорсткий диск вашого комп'ютера).

Схема процесу виглядає так:

1. Завантажте образ для активації/інсталяції.
2. Запишіть образ на USB-накопичувач або DVD-диск.
3. Завантажте систему Live, щоб протестувати LinuxCNC.
4. Завантажте інсталятор, щоб встановити LinuxCNC.

1.3.1 Завантажте зображення

У цьому розділі описано деякі методи завантаження образу Live/Install.

1.3.1.1 Звичайне завантаження

Програмне забезпечення для LinuxCNC для завантаження представлено на сторінці проекту Downloads page. Більшість користувачів будуть шукати образ диска для ПК Intel/AMD, URL-адреса буде схожа на https://www.linuxcnc.org/iso/linuxcnc_2.9.8-amd64.hybrid.iso.

Для Raspberry Pi надано кілька зображень, щоб усунути відмінності між RPi4 та RPi5.

Note
Не використовуйте звичайну дистрибуцію Raspbian для LinuxCNC, яка може бути поставлена разом із вашим стартовим набором RPi, оскільки вона не матиме ядра реального часу і ви не зможете перейти з Raspbian на образ ядра Debian.

1.3.1.2 Завантажити за допомогою zsync

zsync — це програма для завантаження, яка ефективно відновлює перервані завантаження та ефективно передає великі файли з невеликими змінами (якщо у вас є стариша локальна копія). Зверніть увагу, що для роботи програми необхідний протокол http, а не https. Використовуйте zsync, якщо завантаження зображення за допомогою методу Normal Download часто переривається.

zsync у Linux

1. Встановіть zsync за допомогою Synaptic або виконавши наступну команду у terminal

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

2. Потім виконайте цю команду, щоб завантажити ISO-образ на свій комп'ютер

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

Будь ласка, не забудьте перевірити контрольну суму завантаженого ISO-образу, як описано нижче, оскільки автентичність сервера не гарантується протоколом http.

zsync в Windows Існує порт zsync для Windows. Він працює як консольний застосунок і його можна завантажити з https://www.assembla.com/spaces/zsync-windows/documents.

1.3.1.3 Перевірте зображення

(Цей крок не потрібен, якщо ви використовували zsync)

1. Після завантаження перевірте контрольну суму образу, щоб забезпечити його цілісність.

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

або

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

1. Потім порівняйте з цими контрольними сумами

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

Перевірка md5sum у Windows або Mac Windows не постачається з програмою md5sum, але є альтернативи. Більше інформації можна знайти за адресою: Як використовувати MD5SUM

1.3.2 Записати образ на завантажувальний пристрій

ISO-образ LinuxCNC Live/Install — це гібридний ISO-образ, який можна записати безпосередньо на USB-накопичувач (флеш-накопичувач) або DVD-диск і використовувати для завантаження комп'ютера. Образ занадто великий, щоб поміститься на CD-диск.

1.3.2.1 Зображення Raspberry Pi

Образ Raspberry Pi – це повний образ SD-карти, який слід записати на SD-карту за допомогою [додатку Raspberry Pi Imager](https://www.raspberrypi.com/software/). Зверніть увагу, що додаток Imager може відкрити .zip-файл безпосередньо, без необхідності розгортати.

1.3.2.2 Образ AMD-64 (x86-64, ПК) з використанням інструментів графічного інтерфейсу

Завантажте та встановіть Balena Etcher з https://etcher.balena.io/#download-etcher (Linux, Windows, Mac) та запишіть завантажений образ на USB-накопичувач.

Якщо ваш образ не завантажується, спробуйте також Rufus. Це виглядає складніше, але, здається, сумісний з різними BIOS.

1.3.2.3 Командний рядок - Linux

1. Підключіть USB-накопичувач (наприклад, флеш-накопичувач або флеш-накопичувач).
2. Визначте файл пристрою, що відповідає USB-флешці. Цю інформацію можна знайти у виводі `sudo dmesg` після підключення пристрою. `cat /proc/partitions` також може бути корисним.
3. Використайте команду `dd`, щоб записати образ на ваш USB-накопичувач. Наприклад, якщо ваш накопичувач відображається як `/dev/sde`, тоді використовуйте цю команду:

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

1.3.2.4 Командний рядок - MacOS

1. Відкрийте термінал і введіть

```
diskutil list
```

2. Вставте USB-накопичувач і запишіть назву нового диска, який з’явиться, наприклад, /dev/disk5.
3. Відмонтуйте USB. Замість N слід підставити число, вказане вище.

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

4. Передайте дані за допомогою команди dd, як і для Linux вище. Зверніть увагу, що на початку назви диска додається літера ”r”.

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

Запис образу на DVD-диск у Linux

1. Вставте чистий DVD-диск у пристрій для запису. З’явиться вікно «Створення CD/DVD» або «Вибір типу диска». Закрийте його, оскільки ми більше не використовуватимемо його.
2. Перейдіть до завантаженого зображення у файловому браузері.
3. Клацніть правою кнопкою миші на файлі ISO-образу та виберіть «Записати на диск».
4. Виберіть швидкість запису. Рекомендується записувати на найнижчій можливій швидкості.
5. Запустіть процес горіння.

6. Якщо з’явиться вікно «виберіть ім’я файлу для образу диска», просто натисніть кнопку «OK».

Запис образу на DVD-диск у Windows

1. Завантажте та встановіть Infra Recorder, безкоштовну програму для запису зображень з відкритим кодом: https://infrarecorder.org/ .
2. Вставте чистий компакт-диск у дисковод і виберіть «Нічого не робити» або «Скасувати», якщо з’явиться діалогове вікно автозапуску.
3. Відкрийте програму Infra Recorder та виберіть меню «Дії», а потім «Записати образ».

Запис образу на DVD-диск у Mac OSX

1. Завантажте файл .iso
2. Клацніть правою кнопкою миші на файлі у вікні Finder і виберіть «Записати на диск». (Опція запису на диск з’явиться, лише якщо на пристрої встановлено або підключено оптичний привід.)

1.3.3 Тестування LinuxCNC

Підключивши USB-накопичувач або вставивши DVD-диск у привід, вимкніть комп’ютер, а потім знову увімкніть його. Це запустить комп’ютер із Live/Install Image і вибере опцію Live boot.

Note
Якщо система не завантажується з DVD-диска або USB-носія, можливо, знадобиться змінити порядок завантаження в BIOS ПК.

Після завантаження комп’ютера ви можете спробувати LinuxCNC без його встановлення. У режимі Live ви не можете створювати власні конфігурації або змінювати більшість системних налаштувань, але ви можете (і повинні) виконати тест затримки.

Щоб спробувати LinuxCNC: у меню Applications/CNC виберіть LinuxCNC. Відкриється діалогове вікно, в якому ви зможете вибрати одну з багатьох зразкових конфігурацій. На цьому етапі доцільно вибрати конфігурацію «sim». Деякі зразкові конфігурації містять екранні 3D-симулятори верстатів. Щоб їх переглянути, знайдіть «Vismach».

Щоб перевірити, чи підходить ваш комп’ютер для програмної генерації ступінчастих імпульсів, виконайте тест затримки, як показано тут.

На момент написання статті Live Image доступний тільки з ядром preempt-rt і відповідним LinuxCNC. На деяких апаратних засобах це може не забезпечити достатньої затримки. Існує експериментал версія, що використовує ядро RTAI realtime, яке часто забезпечує кращу затримку.

1.3.4 Встановлення LinuxCNC

Щоб встановити LinuxCNC з Live CD, виберіть «Встановити (графічне)» під час завантаження.

1.3.5 Оновлення для LinuxCNC

При звичайній установці Update Manager повідомить вас про оновлення LinuxCNC, коли ви підключите до Інтернету, і дозволить вам легко виконати оновлення без необхідності володіння знаннями про Linux. При запиті можна оновлювати все, крім операційної системи.



Warning
Не оновлюйте операційну систему до нової версії, якщо вас про це попросять. Однак вам слід приймати оновлення ОС, особливо оновлення безпеки.

1.3.6 Проблеми з встановленням

У рідкісних випадках вам може знадобитися скинути BIOS до налаштувань за замовчуванням, якщо під час встановлення з Live CD система не розпізнає жорсткий диск.

1.3.7 Альтернативні методи встановлення

Найпростіший і найкращий спосіб встановлення LinuxCNC — це використання Live/Install Image, як описано вище. Цей метод є максимально простим і надійним, він підходить як для початківців, так і для досвідчених користувачів. Однак, як правило, він замінить будь-яку існуючу операційну систему. Якщо на цільовому ПК є файли, які ви хочете зберегти, скористайтеся одним із методів, описаних у цьому розділі.

Крім того, для досвідчених користувачів, які знайомі з адмініструванням системи Debian (пошук образів інсталяції, робота з джерелами apt, зміна версій ядра тощо), нові інсталяції підтримуються на таких платформах: («amd64» означає «64-бітний» і не є специфічним для процесорів AMD, він працюватиме на будь-якій 64-бітній системі x86)

Debian Trixie	**amd64 & arm64**	**preempt-rt**	**linuxcnc-uspace**	**керування та моделювання машин**
Debian Trixie	amd64	RTAI	linuxcnc	керування машиною
Розповсюдження	Архітектура	Ядро	Назва пакета	Типове використання
Книжковий черв’як Debian	amd64 & arm64	preempt-rt	linuxcnc-uspace	керування та моделювання машин
Книжковий черв’як Debian	amd64	RTAI	linuxcnc	керування машиною
Debian Bullseye	amd64	preempt-rt	linuxcnc-uspace	керування та моделювання машин
Будь-який	Будь-який	Запас	linuxcnc-uspace	ТІЛЬКИ симуляція

Note
LinuxCNC версії 2.9 не підтримується на Debian 9 або старішій версії.

Ядра Preempt-RT Ядра Preempt-rt доступні для Debian у звичайному архіві debian.org. Пакет називається `linux-image-rt-*`. Просто встановіть пакет так само, як і будь-який інший пакет із менеджера пакетів Synaptic або за допомогою apt-get у командному рядку.

Ядра RTAI Ядра RTAI можна завантажити з архіву Debian на сайті linuxcnc.org. Вихідний код apt:

- 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 та ядро RTAI зараз доступні лише для 64-бітних ОС, але залишилося дуже мало систем, які не можуть працювати з 64-бітною ОС.

1.3.7.1 Встановлення на Debian Trixie (з ядром Preempt-RT)

1. Встановіть Debian Trixie (Debian 13), версія amd64. Ви можете завантажити інсталятор тут: https://www.debian.org/distrib/
2. Після запису ISO-образу та завантаження системи, якщо ви не хочете використовувати робочий стіл Gnome, виберіть «Додаткові параметри» > «Альтернативні середовища робочого столу» та виберіть те, яке вам подобається. Потім виберіть «Встановити» або «Графічна інсталяція».



Warning
Не вводьте пароль root, інакше sudo буде вимкнено, і ви не зможете виконати наступні кроки.

3. Виконайте наступну команду у terminal, щоб оновити машину до останніх пакетів.

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

Note
Можна завантажити версію LinuxCNC безпосередньо з Debian (наразі це версія 2.9.4), але новішу версію (2.9.8) можна встановити з репозиторію LinuxCNC.

4. Встановлення ядра та модулів Preempt-RT

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

5. Перезавантажте систему і виберіть ядро Linux 6.1.0-10-rt-amd64. Точна версія ядра може відрізнятися, шукайте суфікс «-rt». Він може бути прихований у підменю «Advanced options for Debian Bookworm» (Розширені параметри для Debian Bookworm) у Grub. Після входу в систему переконайтеся, що команда `PREEMPT RT` відображається наступною командою.

```
uname -v
```

6. Відкрийте меню «Додатки» > «Система» > «Менеджер пакетів Synaptic», знайдіть «linux-image», клацніть правою кнопкою миші на оригінальному не-rt і виберіть «Позначити для повного видалення». Перезавантажте систему. Це потрібно для того, щоб змусити систему завантажуватися з ядра RT. Якщо ви бажаєте зберегти обидва ядра, то інші ядра видаляти не потрібно, але буде потрібно змінити конфігурацію завантаження grub, що виходить за рамки цього документа.

7. Додайте ключ підпису архіву LinuxCNC до свого ключового кільця apt, завантаживши [скрипт інсталятора LinuxCNC](https://www.linuxcnc.org/linuxcnc-install.sh). Щоб запустити скрипт, вам потрібно зробити його виконуваним:

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

```
b''Пb''b''ob''b''тb''b''ib''b''мb'' b''мb''b''ob''b''жb''b''нb''b''ab'' b''зb''b''ab''b ↩
    ''пb''b''yb''b''cb''b''тb''b''иb''b''тb''b''иb'' b''ib''b''нb''b''cb''b''тb''b' ↩
    'ab''b''лb''b''яb''b''тb''b''ob''b''pb'':
```

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

1.3.7.2 Встановлення на Debian Trixie (з експериментальним ядром RTAI)

1. Це ядро та версію LinuxCNC можна встановити поверх інсталяції Live DVD або ж на чисту інсталяцію Debian Trixie 64-bit, як описано вище.
2. Ви можете додати ключ підпису архіву LinuxCNC та інформацію про репозиторій, завантаживши та запустивши інсталяційний скрипт, як описано вище. Якщо буде виявлено ядро RTAI, інсталяція зупиниться перед інсталяцією будь-яких пакетів.
3. Оновіть список пакетів з linuxcnc.org

```
sudo apt-get update
```

4. Видаліть існуючу версію LinuxCNC для uspace та встановіть нове ядро реального часу, RTAI та RTAI-версію LinuxCNC.

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

Перезавантажте машину, переконавшись, що система завантажується з нового ядра 5.4.258-rtai.

1.3.7.3 Встановлення на Raspbian 12

Не робіть цього. Затримки занадто великі з ядром за замовчуванням, а ядро PREEMPT_RT (RT є важливим) Debian не завантажується на Pi (станом на 1/2024). Будь ласка, зверніться до образів .iso, наданих онлайн на звичайній сторінці завантаження LinuCNC. Ви можете створити їх самостійно, дотримуючись сценаріїв, наданих онлайн.

1.4 Запуск LinuxCNC

1.4.1 Виклик LinuxCNC

Після встановлення LinuxCNC запускається так само, як і будь-яка інша програма Linux: запустіть її з terminal, виконавши команду *linuxcnc*, або виберіть її в меню *Programs -> CNC*.

1.4.2 Запуск конфігурації

Під час запуску LinuxCNC (з меню CNC або з командного рядка без вказівки INI-файлу) запускається діалогове вікно Вибір конфігурації.

Діалогове вікно «Вибір конфігурації» дозволяє користувачеві вибрати одну з існуючих конфігурацій (Мої конфігурації) або вибрати нову (з Прикладів конфігурацій) для копіювання в домашній каталог. Скопійовані конфігурації з'являться в розділі Мої конфігурації при наступному запуску вікна «Вибір конфігурації».

Селектор конфігурації пропонує вибір конфігурацій, організованих за такими параметрами:

- «Мої конфігурації» – конфігурації користувача, розташовані у файлі linuxcnc/configs у вашому домашньому каталозі.
- «Приклади конфігурацій» — вибрані приклади конфігурацій копіюються до папки linuxcnc/-configs. Після копіювання прикладу конфігурації до вашої локальної папки, програма запуску запропонує його як «Мої конфігурації». Назви, під якими представлені ці локальні конфігурації, відповідають назвам папок у папці configs/:
  - *sim* - Конфігурації, що включають симульоване обладнання. Їх можна використовувати для тестування або вивчення роботи LinuxCNC.
  - *by_interface* - Конфігурації, організовані за допомогою графічного інтерфейсу.
  - *by_machine* - Конфігурації, організовані за машиною.
  - *apps* - Програми, які не потребують запуску linuxcnc, але можуть бути корисними для тестування або спроби роботи програм, таких як PyVCP або GladeVCP.
  - *attic* - Застарілі або історичні конфігурації.

Конфігурації симулятора часто є найкориснішою відправною точкою для нових користувачів і організовані навколо підтримуваних графічних інтерфейсів:

- *axis* - Графічний інтерфейс клавіатури та миші
- *craftsman* - Графічний інтерфейс користувача з сенсорним екраном (більше не підтримується ???)
- *gmoccapy* - Сенсорний екран графічного інтерфейсу
- *gscreen* - Сенсорний екран графічного інтерфейсу
- *pyvcp_demo* – Віртуальна панель керування Python
- *qtaxis* - Сенсорний екран графічного інтерфейсу, схожий на осі
- *qtdragon* - Сенсорний екран графічного інтерфейсу
- *qtdragon_hd* - Сенсорний екран графічного інтерфейсу, висока чіткість
- *qtplasmac* - Сенсорний екран графічного інтерфейсу для плазмових столів
- *qttouchy* - Сенсорний екран графічного інтерфейсу
- *tklinuxcnc* - Графічний інтерфейс клавіатури та миші (більше не підтримується)
- *touchy* - Сенсорний екран графічного інтерфейсу
- *woodpecker* - Сенсорний екран графічного інтерфейсу

Каталог конфігурації графічного інтерфейсу може містити підкаталоги з конфігураціями, що ілюструють спеціальні ситуації або вбудовування інших програм.

Конфігурації «by_interface» організовані навколо поширених, підтримуваних інтерфейсів, таких як:

- загальна мехатроніка
- mesa
- parport
- pico
- pluto
- servotogo
- пильний
- життєво важливі системи

Для використання цих конфігурацій як відправних точок для системи може знадобитися відповідне обладнання.

Конфігурації «by_machine» організовані навколо повних, відомих систем, таких як:

- бос
- крутотуль
- scortbot erIII
- Шерлайн
- кузня
- громіздкий

Для використання цих конфігурацій може знадобитися повна система.

«Елементи програм» зазвичай є одними з наступних:

1. утиліти, які не потребують запуску linuxcnc
2. демонстрації програм, які можна використовувати з linuxcnc
   - info – створює файл із системною інформацією, яка може бути корисною для діагностики проблем.
   - gladevcp - Приклади програм GladeVCP.
   - halrun - Запускає halrun у terminal.
   - latency - Застосування для дослідження затримки
     - latency-histogram-1 - гістограма для одного сервопотоку
     - latency-histogram - гістограма
     - тест-латентності - стандартний тест
     - latency-plot - stripchart
   - parport - Заявки на тестування паркету.
   - pyvcp - Приклади pyvcp-застосунків.
   - xhc-hb04 - Застосування для тестування бездротового MPG xhc-hb04 USB

Note
У каталозі «Програми» для копіювання до каталогу користувача пропонуються лише ті програми, які користувач корисно змінив.

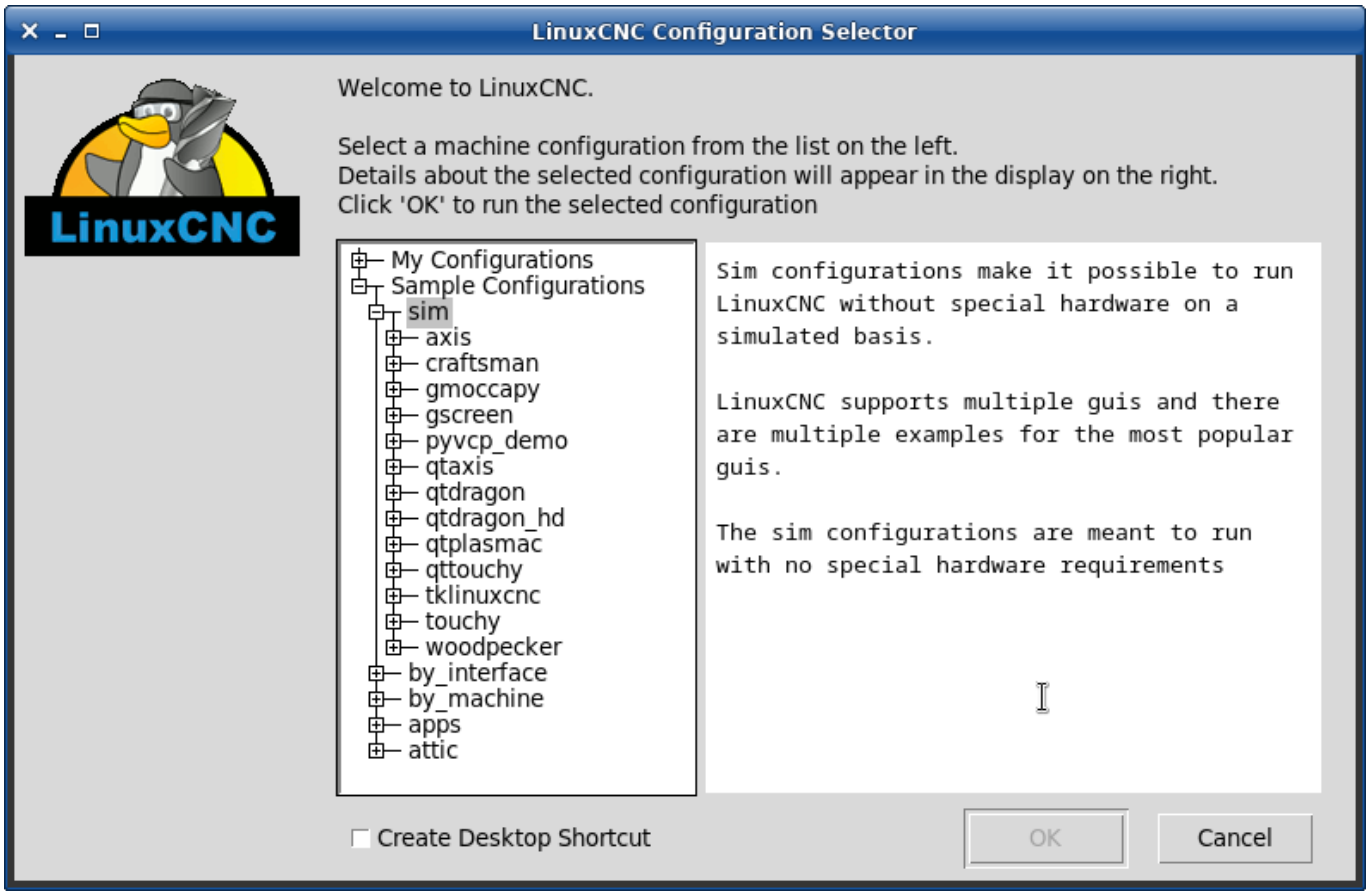


Figure 1.2: Вибір конфігурації LinuxCNC

Клацніть будь-яку зі списку конфігурацій, щоб відобразити певну інформацію про неї. Двічі клацніть конфігурацію або натисніть кнопку «OK», щоб розпочати налаштування.

Виберіть «Створити ярлик на робочому столі» та натисніть «OK», щоб додати значок на робочий стіл Ubuntu для безпосереднього запуску цієї конфігурації без відображення екрана вибору конфігурац

Коли ви вибираєте конфігурацію з розділу «Зразки конфігурацій», копія цієї конфігурації автоматично розміщується в каталозі ~/linuxcnc/configs.

1.4.3 Наступні кроки в налаштуванні

Знайшовши зразок конфігурації, який використовує те саме апаратне забезпечення інтерфейсу, що й ваш комп'ютер (або конфігурацію симулятора), та збереживши його копію у вашій домашній директорії, ви можете налаштувати його відповідно до особливостей вашого комп'ютера. Інформацію про конфігурацію див. у Посібнику інтегратора.

1.4.4 Конфігурації симулятора

Усі конфігурації, перелічені в розділі «Зразки конфігурацій/sim», призначені для роботи на будь-якому комп'ютері. Не потрібне спеціальне обладнання та підтримка в режимі реального часу.

Ці конфігурації корисні для вивчення окремих можливостей або опцій. Конфігурації симулятора організовані відповідно до графічного інтерфейсу користувача, який використовується в демонстрації. Каталог для осі містить найбільшу кількість варіантів і підкаталогів, оскільки це найбільш перевірений

графічний інтерфейс користувача. Можливості, продемонстровані за допомогою будь-якого конкретно графічного інтерфейсу користувача, можуть бути доступні і в інших графічних інтерфейсах.

1.4.5 Ресурси конфігурації

Селектор конфігурації копіює всі файли, необхідні для конфігурації, в новий підкаталог ~/linuxcnc/configs (еквівалентно: /home/username/linuxcnc/configs). Кожен створений каталог буде містити принаймні один файл INI (iniflename.ini), який використовується для опису конкретної конфігурації.

Ресурси файлів у скопійованому каталозі зазвичай містять один або кілька файлів INI (ім'я_файлу.ini) для відповідних конфігурацій та файл таблиці інструментів (ім'я_файлу_інструменту.tbl). Крім того, ресурси можуть містити файли HAL (filename.hal, filename.tcl), файл README з описом каталогу та інформацію про конкретну конфігурацію в текстовому файлі, названому на честь конкретної конфігурації (inifilename.txt). Останні два файли відображаються під час використання селектора конфігурації.

У наданих зразках конфігурацій у файлі конфігурації INI може бути вказано параметр HALFILE (ім'я файлу.hal), який відсутній у скопійованому каталозі, оскільки він знаходиться в системній бібліотеці файлів HAL. Ці файли можна скопіювати до каталогу конфігурації користувача та змінити відповідно до потреб користувача для модифікації або тестування. Оскільки при пошуку файлів HAL спочатку перевіряється каталог конфігурації користувача, місцеві модифікації матимуть пріоритет.

Селектор конфігурації створює символічне посилання в каталозі конфігурації користувача (з назвою hallib), яке вказує на системну бібліотеку файлів HAL. Це посилання спрощує копіювання файлу бібліотеки. Наприклад, щоб скопіювати файл бібліотеки core_sim.hal для внесення локальних змін:

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

1.5 Оновлення LinuxCNC

Оновлення LinuxCNC до нового минорного релізу (тобто до нової версії в тій самій стабільній серії, наприклад, з 2.9.7 до 2.9.8) відбувається автоматично, якщо ваш ПК підключено до Інтернету. Ви побачите запит на оновлення після минорного релізу разом з іншими оновленнями програмного забезпечення. Якщо у вас немає підключення до Інтернету на вашому ПК, див. Оновлення без мережі.

1.5.1 Оновіться до нової версії

У цьому розділі описано, як оновити LinuxCNC з версії 2.8.x до версії 2.9.y. Припускається, що у вас є встановлена версія 2.8, яку ви хочете оновити.

Щоб оновити LinuxCNC з версії, старшої за 2.8, спочатку необхідно оновити стару версію до 2.8, а потім виконати ці інструкції для оновлення до нової версії.

Якщо у вас немає старої версії LinuxCNC для оновлення, тоді краще зробити чисту інсталяцію нової версії, як описано в розділі Отримання LinuxCNC.

Крім того, якщо ви використовуєте Ubuntu Precise, Debian Wheezy або Debian Buster, варто розглянути можливість створення резервної копії каталогу "linuxcnc" на знімному носії та виконання чистої інсталяції новішої ОС та версії LinuxCNC, оскільки ці випуски були EOL у 2017, 2018 та 2022 роках відповідно. Якщо ви використовуєте Ubuntu Lucid, вам доведеться зробити це, оскільки Lucid більше не підтримується LinuxCNC (він був EOL у 2013 році).

Щоб оновити основні версії, наприклад 2.8 до 2.9, коли у вас є мережеве підключення на комп’ютері, вам потрібно вимкнути старі джерела linuxcnc.org apt у файлі /etc/apt/sources.list і додати нове джерело linuxcnc.org apt для 2.9, а потім оновити LinuxCNC.

Деталі залежатимуть від платформи, на якій ви працюєте. Відкрийте terminal, а потім введіть `lsb_release -ic`, щоб знайти цю інформацію:

```
lsb_release -ic
b''Дb''b''иb''b''cb''b''тb''b''pb''b''иb''b''бb'''b''юb''b''тb''b''ob''b''pb'' ID: Debian
b''Kb''b''ob''b''дb''b''ob''b''вb''b''eb'' b''ib''b''мb'''b''яb'':       Trixie
```

Ви повинні використовувати Debian Bullseye, Bookworm або Trixie, або Ubuntu 20.04 ”Focal Fossa” або новішу версію. LinuxCNC 2.9.y не працюватиме на старіших дистрибутивах.

Вам також потрібно буде перевірити, яке ядро реального часу використовується:

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

Якщо ви бачите (як вище) `-rt-` в імені ядра, то ви використовуєте ядро preempt-rt і повинні встановити версію LinuxCNC «uspace». Ви також повинні встановити uspace для конфігурацій «sim» на ядрах, що не працюють в режимі реального часу.

Якщо ви бачите «-rtai-» в назві ядра, то ви використовуєте RTAI realtime. Дивіться нижче, яку версію LinuxCNC потрібно встановити. Пакет RTAI доступний для Bookworm і Buster, але наразі недоступний для Bullseye.

1.5.1.1 Конфігурація Apt Sources

- Відкрийте вікно «Джерела програмного забезпечення». Процес цього дещо відрізняється на трьох підтримуваних платформах:
  - Debian:
    - Натисніть «Меню програм», потім «Система», а потім «Менеджер пакетів Synaptic».
    - У Synaptic натисніть меню «Налаштування», потім натисніть «Репозиторії», щоб відкрити вікно «Джерела програмного забезпечення».
  - Точність Ubuntu:
    - Натисніть на значок «Головна панель Dash» у верхньому лівому куті.
    - У полі «Пошук» введіть «програмне забезпечення», а потім натисніть на значок «Центр програмного забезпечення Ubuntu».
    - У вікні Центру програмного забезпечення Ubuntu натисніть меню «Редагувати», потім натисніть «Джерела програмного забезпечення…», щоб відкрити вікно «Джерела програмного забезпечення».
  - Ubuntu Lucid:
    - Натисніть меню «Система», потім «Адміністрування», а потім «Менеджер пакетів Synaptic».
    - У Synaptic натисніть меню «Налаштування», потім натисніть «Репозиторії», щоб відкрити вікно «Джерела програмного забезпечення».
- У вікні «Джерела програмного забезпечення» виберіть вкладку «Інше програмне забезпечення».
- Видаліть або зніміть позначки з усіх старих записів linuxcnc.org (залиште всі рядки, що не стосуються linuxcnc.org, як є).
- Натисніть кнопку «Додати» та додайте новий рядок apt. Рядок дещо відрізнятиметься на різних платформах:

Table 1.2: Табличний огляд варіантів операційної системи та відповідної конфігурації репозиторію. Конфігурацію можна виконати в графічному інтерфейсі диспетчера пакетів або у файлі /etc/apt/sources.list.

ОС / Версія реального часу	**Репозиторій**
Debian Bullseye - випередження	`deb https://linuxcnc.org bullseye base 2.9-uspace`
Debian Bookworm - випередити	`deb https://linuxcnc.org bookworm base 2.9-uspace`
Debian Bookworm - RTAI	`deb https://linuxcnc.org bookworm base 2.9-rt`
Debian Trixie - витіснення	`deb https://linuxcnc.org trixie база 2.9-міліметровий простір`
Debian Trixie - RTAI	`deb https://linuxcnc.org trixie база 2.9-rt`

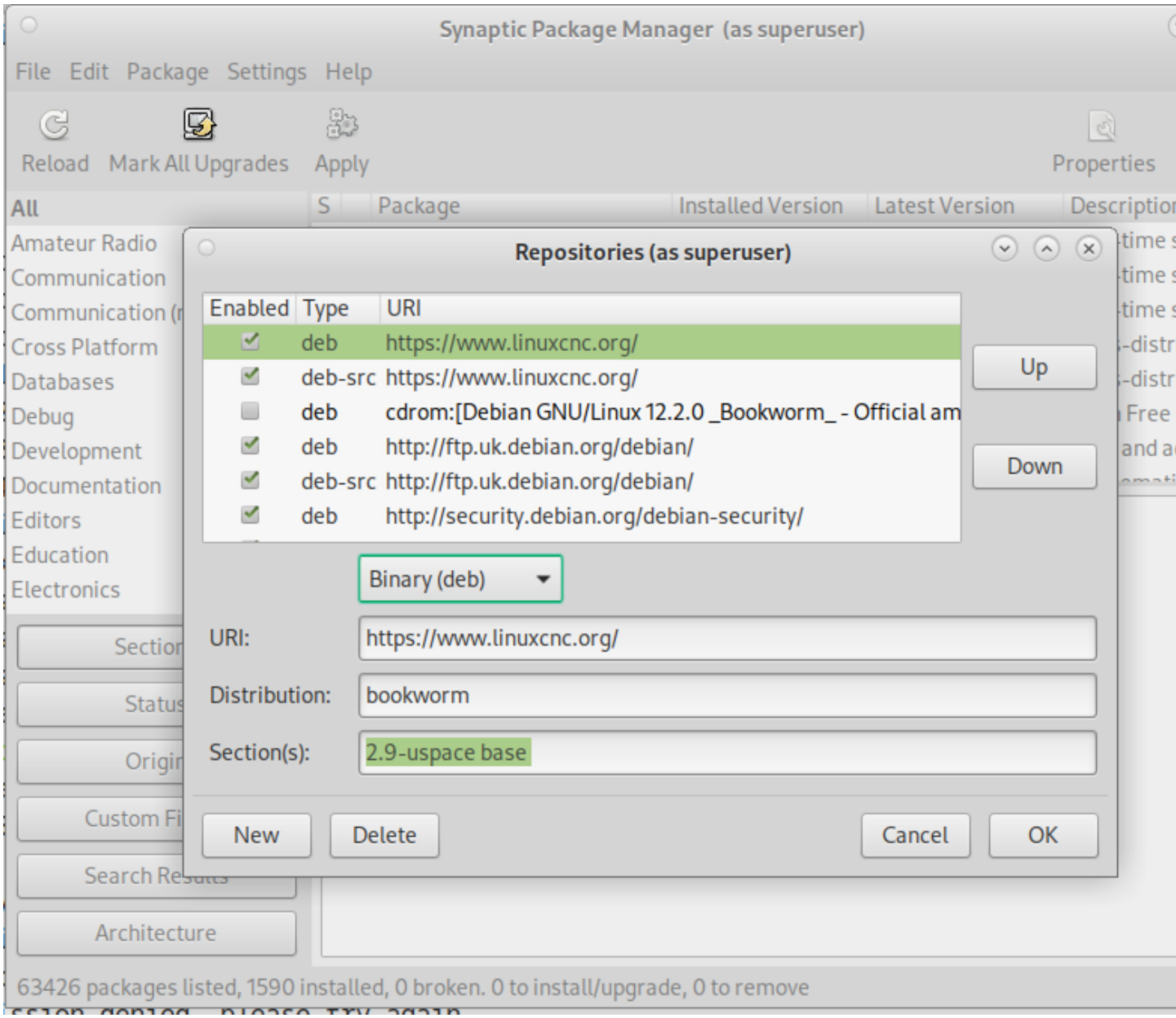


Figure 1.3: Рисунок зі скріншотом конфігурації репозиторію менеджера пакетів synaptic.

- Натисніть кнопку «Додати джерело», а потім «Закрити» у вікні «Джерела програмного забезпечення». Якщо з’явиться вікно з повідомленням про те, що інформація про доступне програмне забезпечення застаріла, натисніть кнопку «Перезавантажити».

1.5.1.2 Оновлення до нової версії

Тепер ваш комп’ютер знає, де взяти нову версію програмного забезпечення, далі нам потрібно її встановити.

Процес знову ж таки відрізняється залежно від вашої платформи.

Debian використовує менеджер пакетів Synaptic.

- Відкрийте Synaptic, використовуючи інструкції з розділу Налаштування джерел apt вище.

- Натисніть кнопку «Перезавантажити».
- Використайте функцію пошуку для пошуку `linuxcnc`.
- Пакет називається ”linuxcnc” для ядер RTAI та ”linuxcnc-uspace” для preempt-rt.
- Поставте галочку, щоб позначити нові пакети linuxcnc та linuxcnc-doc-* для оновлення. Менеджер пакетів може вибрати ряд додаткових пакетів для встановлення, щоб задовольнити залежності, які має новий пакет linuxcnc.
- Натисніть кнопку «Застосувати» та дозвольте комп’ютеру встановити новий пакет. Старий пакет linuxcnc буде автоматично оновлено до нового.

1.5.1.3 Ubuntu

- Натисніть на значок «Головна панель Dash» у верхньому лівому куті.
- У полі «Пошук» введіть «оновлення», а потім натисніть на значок «Менеджер оновлень».
- Натисніть кнопку «Перевірити», щоб отримати список доступних пакетів.
- Натисніть кнопку «Встановити оновлення», щоб встановити нові версії всіх пакетів.

1.5.2 Оновлення без мережі

Щоб оновити систему без мережевого підключення, потрібно завантажити .deb-файл, а потім встановити його за допомогою dpkg. Deb-файли можна знайти за адресою https://linuxcnc.org/-dists/.

Вам потрібно перейти за посиланням вище, щоб знайти правильний deb-файл для вашої інсталяції. Відкрийте terminal та введіть *lsb_release -ic*, щоб знайти назву випуску вашої ОС.

```
> lsb_release -ic
b''Дb''b''иb''b''cb''b''тb''b''pb''b''иb''b''бb'''b''юb''b''тb''b''ob''b''pb'' ID: Debian
b''Kb''b''ob''b''дb''b''ob''b''вb''b''eb'' b''ib''b''мb'''b''яb'':       trixie
```

Виберіть ОС зі списку, а потім виберіть потрібну основну версію, наприклад, 2.9-rt для RTAI або 2.9-uspace для preempt-rt.

Далі виберіть тип вашого комп’ютера: binary-amd64 для 64-бітного ПК або binary-arm64 (64-біт) для Raspberry Pi.

Далі виберіть потрібну версію знизу списку, наприклад, «linuxcnc-uspace_2.9.8_amd64.deb» (виберіть найновішу за датою). Завантажте deb-файл і скопіюйте його до свого домашнього каталогу. Ви можете перейменувати файл на щось коротше за допомогою файлового менеджера, наприклад, «linuxcnc_2.9.8.deb», потім відкрийте термінал і встановіть його за допомогою менеджера пакетів за допомогою цієї команди:

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

1.5.3 Оновлення файлів конфігурації для версії 2.9

1.5.3.1 Суворіша обробка підключаємих інтерпретаторів

Якщо ви просто запускаєте звичайний G-код і не знаєте, що таке підключаємий інтерпретатор, то цей розділ вас не стосується.

Рідко використовуваною функцією LinuxCNC є підтримка підключаємих інтерпретаторів, якими керує недокументований INI-файл [TASK]INTERPRETER.

Версії LinuxCNC до версії 2.9.0 обробляли неправильне налаштування [TASK]INTERPRETER, автоматичн повертаючись до використання інтерпретатора G-коду за замовчуванням.

Починаючи з версії 2.9.0, неправильне значення [TASK]INTERPRETER призведе до відмови LinuxCNC запускатися. Виправте цю ситуацію, видаливши налаштування [TASK]INTERPRETER з вашого INI-файлу, щоб LinuxCNC використовував інтерпретатор G-коду за замовчуванням.

1.5.3.2 Кантерп

Якщо ви просто запускаєте звичайний G-код і не використовуєте підключаємий інтерпретатор canterp, то цей розділ вас не стосується.

У надзвичайно малоймовірному випадку, якщо ви використовуєте canterp, майте на увазі, що модуль переміщено з /usr/lib/libcanterp.so до /usr/lib/linuxcnc/canterp.so, і налаштування [TASK]INTERPRETER відповідно потрібно змінити з libcanterp.so на canterp.so.

1.5.3.3 Обмеження шпинделя в INI

Тепер можна додавати налаштування до розділу [SPINDLE] INI-файлу.

MAX_FORWARD_VELOCITY = 20000 Максимальна швидкість шпинделя (в об/хв)

MIN_FORWARD_VELOCITY = 3000 Мінімальна швидкість шпинделя (в об/хв)

MAX_REVERSE_VELOCITY = 20000 Якщо цей параметр пропустити, він матиме значення MAX_FORWA за замовчуванням.

MIN_REVERSE_VELOCITY = 3000` Цей параметр еквівалентний MIN_FORWARD_VELOCITY, але для зворотного обертання шпинделя. Якщо його пропустити, за замовчуванням використовуватиметьс MIN_FORWARD_VELOCITY.

INCREMENT = 200 Встановлює розмір кроку для команд збільшення/зменшення швидкості шпинделя. Це значення може бути різним для кожного шпинделя. Цей параметр діє з AXIS та Touchy, але зверніть увагу, що деякі екрани керування можуть обробляти речі по-різному.

HOME_SEARCH_VELOCITY = 100 - Прийнято, але наразі нічого не робить

HOME_SEQUENCE = 0 - Прийнято, але наразі нічого не робить

1.5.4 Оновлення файлів конфігурації для версії 2.10.y

Touchy: записи Touchy MACRO тепер слід розміщувати в розділі [MACROS] INI-файлу, а не в розділі [TOUCHY]. Це частина процесу уніфікації налаштувань INI між графічними інтерфейсами.

1.5.5 Нові компоненти HAL

1.5.5.1 Не в реальному часі

mdro mqtt-видавник pi500_vfd pmx485-тест qtplasmac-cfg2prefs qtplasmac-матеріали qtplasmac-plasmac2qt qtplasmac-налаштування sim-факел svd-ps_vfd

1.5.5.2 У режимі реального часу

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

1.5.6 Нові водії

Було представлено фреймворк для керування пристроями ModBus за допомогою послідовних портів на багатьох платах Mesa. http://linuxcnc.org/docs/2.9/html/drivers/mesa_modbus.html

Новий драйвер GPIO для будь-якого GPIO, який підтримується бібліотекою gpiod, тепер включено: http://linuxcnc.org/docs/2.9/html/drivers/hal_gpio.html

1.6 Найчастіші запитання щодо Linux

Ось деякі основні команди та методи Linux для новачків у Linux. Більш повну інформацію можна знайти в Інтернеті або на сторінках довідника (man).

1.6.1 Автоматичний вхід

1.6.1.1 Debian

Debian Stretch за замовчуванням використовує середовище робочого столу Xfce з менеджером дисплеїв lightDM. Щоб отримати автоматичний вхід за допомогою Stretch:

- У терміналі скористайтеся командою:

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

- Запишіть абсолютний шлях до файлу конфігурації lightdm.conf.
- Відредагуйте цей файл за допомогою текстового редактора (gedit, nano тощо) від імені root.
- Знайдіть та розкоментуйте рядки:

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

- Встановити autologin-user=ваше_ім’я_користувача
- Збережіть та перезавантажте.

1.6.1.2 Ubuntu

При встановленні LinuxCNC за допомогою Ubuntu LiveCD за замовчуванням необхідно входити в систему кожного разу, коли ви вмикаєте комп’ютер. Щоб увімкнути автоматичний вхід, перейдіть до «Система > Адміністрування > Вікно входу». Якщо це нова інсталяція, вікно входу може з’явитися через секунду-три. Вам знадобиться пароль, який ви використовували для інсталяції, щоб отримати доступ до вікна «Налаштування вікна входу». На вкладці «Безпека» встановіть прапорець «Увімкнути автоматичний вхід» і виберіть ім’я користувача зі списку (це будете ви).

1.6.2 Автоматичний запуск

Щоб LinuxCNC запускався автоматично з вашою конфігурацією після увімкнення комп'ютера, перейдіть до «Система > Налаштування > Сеанси > Програми запуску», натисніть «Додати». Перейдіть до вашої конфігурації та виберіть файл .ini. Коли діалогове вікно вибору файлу закриється, додайте linuxcnc та пробіл перед шляхом до вашого файлу .ini.

Приклад:

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

У документації ваш відповідний файл .ini посилається на INI-файл.

1.6.3 Термінал

Багато речей потрібно робити з терміналу, наприклад, перевіряти буфер повідомлень ядра за допомогою команди «dmesg». Ubuntu та Linux Mint мають комбінацію клавіш Ctrl + Alt + t. Debian Stretch не має визначених комбінацій клавіш. Їх можна легко створити за допомогою «Менеджера конфігурації». Більшість сучасних файлових менеджерів підтримують праву клавішу для відкриття терміналу, просто переконайтеся, що ви клацаєте правою кнопкою миші на порожній області або каталозі, а не на імені файлу. Більшість ОС мають термінал як пункт меню, зазвичай у розділі «Аксесуари».

1.6.4 Сторінки користувача

Сторінка керівництва (скорочено від manual page — сторінка посібника) — це форма документації до програмного забезпечення, яка зазвичай знаходиться в UNIX або UNIX-подібних операційних системах, таких як Linux.

Щоб переглянути сторінку довідника, відкрийте термінал та дізнайтеся щось про команду find, у вікні термінала введіть:

```
man find
```

Використовуйте клавіші Page Up та Page Down для перегляду сторінки довідки, а клавішу Q — для виходу з перегляду.

Note
Перегляд сторінки man з терміналу може не дати очікуваного результату. Наприклад, якщо ви введете man abs, ви отримаєте C abs, а не LinuxCNC abs. Найкраще переглядати сторінки man LinuxCNC у форматі HTML.

1.6.5 Список модулів

Іноді під час усунення несправностей потрібно отримати список завантажених модулів. У вікні терміналу введіть:

```
lsmod
```

Якщо ви хочете надіслати вивід lsmod до текстового файлу у вікні терміналу, введіть:

```
lsmod > mymod.txt
```

Отриманий текстовий файл буде розташований у домашньому каталозі, якщо ви не змінили каталоги під час відкриття вікна терміналу, і він називатиметься mymod.txt або як ви його там назвали.

1.6.6 Редагування кореневого файлу

Коли ви відкриваєте файловий браузер і бачите, що власником файлу є root, вам необхідно виконати додаткові кроки, щоб редагувати цей файл. Редагування деяких файлів root може мати негативні наслідки. Будьте обережні при редагуванні файлів root. Зазвичай ви можете відкривати і переглядати більшість файлів root, але вони відкриватимуться в режимі «тільки для читання».

1.6.6.1 Спосіб командного рядка

Відкрийте термінал і введіть

```
sudo gedit
```

Відкрийте файл за допомогою Файл > Відкрити > Редагувати

1.6.6.2 Шлях графічного інтерфейсу

1. Клацніть правою кнопкою миші на робочому столі та виберіть «Створити панель запуску».
2. Введіть ім’я, наприклад, sudo edit.
3. Введіть команду «gksudo ”gnome-open %u”» та збережіть панель запуску на робочому столі.
4. Перетягніть файл на панель запуску, щоб відкрити та редагувати його.

1.6.6.3 Root-доступ

В Ubuntu ви можете отримати root-права, ввівши ”sudo -i” у вікні терміналу, а потім ввівши свій пароль. Будьте обережні, бо ви можете серйозно зіпсувати роботу як root-користувач, якщо не знаєте, що робите.

1.6.7 Команди терміналу

1.6.7.1 Робочий каталог

Щоб дізнатися шлях до поточного робочого каталогу у вікні терміналу, введіть:

```
pwd
```

1.6.7.2 Зміна каталогів

Щоб змінити робочий каталог на рівень вище, тобто батьківський каталог, у вікні терміналу введіть:

```
cd ..
```

Щоб переміститися на два рівні вище у вікні терміналу, введіть:

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

Щоб перейти безпосередньо до вашого домашнього каталогу, у вікні терміналу скористайтеся командою cd без аргументів:

```
cd
```

Щоб перейти до підкаталогу linuxcnc/configs у вікні терміналу, введіть:

```
cd linuxcnc/configs
```

1.6.7.3 Перелік файлів у каталозі

Щоб переглянути список усіх файлів та підкаталогів у вікні терміналу, введіть:

```
dir
```

або

```
ls
```

1.6.7.4 Пошук файлу

Команда find може дещо заплутати новачка в Linux. Основний синтаксис такий:

```
b''зb''b''нb''b''ab''b''йb''b''тb''b''иb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b' ↩
    'тb''b''pb''b''иb'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ob''b' ↩
    'гb''b''ob'' b''кb''b''ab''b''тb''b''ab''b''лb''b''ob''b''гb''b''yb'' b''дb''b''ib''b' ↩
    'їb''
```

Наприклад, щоб знайти всі файли .ini у вашому каталозі linuxcnc, спочатку потрібно скористатися командою pwd для визначення каталогу.

Відкрийте нове вікно терміналу та введіть:

```
pwd
```

А pwd може повернути такий результат:

```
/home/joe
```

З цією інформацією складіть команду ось так:

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

JustToAvoidStartingWithAHyphen FIXME -name - це ім’я файлу, який ви шукаєте, а -print вказує вивести результат у вікно терміналу. *.ini вказує find повернути всі файли, що мають розширення .ini. Зворотний слеш необхідний для екранування метасимволів оболонки. Дивіться сторінку man find для отримання додаткової інформації про find.

1.6.7.5 Пошук тексту

```
grep -irl 'b''тb''b''eb''b''кb''b''cb''b''тb'' b''дb''b''лb''b''яb'' b''пb''b''ob''b''шb''b ↩
    ''yb''b''кb''b''yb''' *
```

Це дозволить знайти всі файли, що містять «текст для пошуку» у поточному каталозі та всіх підкаталогах нижче нього, ігноруючи регістр. Параметр -i призначений для ігнорування регістру, а -r — для рекурсивного пошуку (включення всіх підкаталогів у пошук). Опція -l поверне список імен файлів, якщо ви не вкажете -l, ви також отримаєте текст, де знаходиться кожне входження «тексту для пошуку». * є символом-замінником для пошуку у всіх файлах. Дивіться сторінку довідки grep для отримання додаткової інформації.

1.6.7.6 Діагностичні повідомлення

Щоб переглянути діагностичні повідомлення, використовуйте ”dmesg” у вікні командного рядка. Щоб зберегти діагностичні повідомлення у файл, використовуйте оператор перенаправлення >, ось так:

```
dmesg > bootmsg.txt
```

Вміст цього файлу можна скопіювати та вставити онлайн, щоб поділитися ним з людьми, які намагаються допомогти вам діагностувати вашу проблему.

Щоб очистити буфер повідомлень, введіть ось що:

```
sudo dmesg -c
```

Це може бути корисним зробити безпосередньо перед запуском LinuxCNC, щоб був запис лише інформації, пов’язаної з поточним запуском LinuxCNC.

Щоб знайти вбудовану адресу паралельного порту, використовуйте grep для фільтрації інформації з dmesg.

Після завантаження відкрийте термінал і введіть:

```
dmesg|grep parport
```

1.6.8 Предмети зручності

1.6.8.1 Запуск термінала

Якщо ви хочете додати запуск терміналу до панелі вгорі екрана, зазвичай можна клацнути правою кнопкою миші на панелі вгорі екрана і вибрати «Додати до панелі». Виберіть «Запуск користувацького додатка» і «Додати». Дайте йому ім’я і введіть gnome-terminal у командному полі.

1.6.9 Проблеми з обладнанням

1.6.9.1 Інформація про обладнання

Щоб дізнатися, яке обладнання підключено до вашої материнської плати, у вікні терміналу введіть:

```
lspci -v
```

1.6.9.2 Роздільна здатність монітора

Під час встановлення Ubuntu намагається визначити налаштування монітора. Якщо це не вдається, ви залишаєтесь зі звичайним монітором із максимальною роздільною здатністю 800x600.

Інструкції щодо виправлення цього знаходяться тут:

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

1.6.10 Шляхи

Відносні шляхи Відносні шляхи базуються на стартовому каталозі, який є каталогом, що містить INI-файл. Використання відносних шляхів може полегшити переміщення конфігурацій, але вимагає хорошого розуміння специфікаторів шляхів в Linux.

```
./f0        b’’тb’’b’’eb’’ b’’cb’’b’’ab’’b’’мb’’b’’eb’’, b’’щb’’b’’ob’’ b’’йb’’ f0, b’’нb’’ ↩
    b’’ab’’b’’пb’’b’’pb’’b’’иb’’b’’кb’’b’’лb’’b’’ab’’b’’дb’’, b’’фb’’b’’ab’’b’’йb’’b’’лb’’ b ↩
    ’’зb’’ b’’іb’’b’’мb’’b’’eb’’b’’нb’’b’’eb’’b’’мb’’ f0 b’’уb’’ b’’кb’’b’’ab’’b’’тb’’b’ ↩
    ’ab’’b’’лb’’b’’ob’’b’’зb’’b’’іb’’ b’’ab’’b’’вb’’b’’тb’’b’’ob’’b’’зb’’b’’ab’’b’’вb’’b’ ↩
    ’ab’’b’’нb’’b’’тb’’b’’ab’’b’’жb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’
../f1       b’’пb’’b’’ob’’b’’cb’’b’’иb’’b’’лb’’b’’ab’’b’’єb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b ↩
    ’’нb’’b’’ab’’ b’’фb’’b’’ab’’b’’йb’’b’’лb’’ f1 b’’уb’’ b’’бb’’b’’ab’’b’’тb’’b’’ьb’’b’ ↩
    ’кb’’b’’іb’’b’’вb’’b’’cb’’b’’ьb’’b’’кb’’b’’ob’’b’’мb’’b’’уb’’ b’’кb’’b’’ab’’b’’тb’’b’ ↩
    ’ab’’b’’лb’’b’’ob’’b’’зb’’b’’іb’’
../../f2    b’’пb’’b’’ob’’b’’cb’’b’’иb’’b’’лb’’b’’ab’’b’’єb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b ↩
    ’’нb’’b’’ab’’ b’’фb’’b’’ab’’b’’йb’’b’’лb’’ f2 b’’уb’’ b’’бb’’b’’ab’’b’’тb’’b’’ьb’’b’ ↩
    ’кb’’b’’іb’’b’’вb’’b’’cb’’b’’ьb’’b’’кb’’b’’ob’’b’’мb’’b’’уb’’ b’’eb’’b’’лb’’b’’eb’’b’ ↩
    ’мb’’b’’eb’’b’’нb’’b’’тb’’b’’іb’’ b’’бb’’b’’ab’’b’’тb’’b’’ьb’’b’’кb’’b’’іb’’b’’вb’’b’ ↩
    ’cb’’b’’ьb’’b’’кb’’b’’ob’’b’’гb’’b’’ob’’ b’’кb’’b’’ab’’b’’тb’’b’’ab’’b’’лb’’b’’ob’’b’ ↩
    ’гb’’b’’уb’’
../../../f3 b’’тb’’b’’ob’’b’’щb’’b’’ob’’.
```

Chapter 2

Загальна користувацька інформація

2.1 Передмова користувача

LinuxCNC є модульною та гнучкою системою. Ці властивості змушують багатьох вважати її заплутаною сумішшю дрібниць і дивуватися, чому вона така. Ця сторінка намагається відповісти на це питання, перш ніж ви поринете в суть справи.

LinuxCNC розпочав свою діяльність у Національному інституті стандартів і технологій США. Він розвивався, використовуючи UNIX як операційну систему. UNIX зробив його особливим. Серед перших розробників UNIX сформувався набір ідей щодо написання коду, який деякі називають «способом UNIX». Ці перші автори LinuxCNC дотримувалися цих способів.

Ерік С. Реймонд у своїй книзі «Мистецтво програмування UNIX» підсумовує філософію UNIX як широко використовувану інженерну філософію «Keep it Simple, Stupid» (принцип KISS). Далі він описує, як, на його думку, ця загальна філософія застосовується як культурна норма UNIX, хоча, що не дивно, в реальній практиці UNIX неважко знайти серйозні порушення більшості з наведених нижче принципів:

- Правило модульності: Пишіть прості частини, з’єднані зрозумілими інтерфейсами.
- Правило ясності: Ясність краща за кмітливість.
- Правило композиції: Проєктуйте програми так, щоб вони були пов’язані з іншими програмами.
- Правило розділення: Відокремте політику від механізму; відокремте інтерфейси від двигунів. примітка: [Знайдено за посиланням: https://en.wikipedia.org/wiki/Separation_of_mechanism_and_polic 2022-11-13]

Пан Реймонд запропонував ще кілька правил, але ці чотири описують основні характеристики системи керування рухом LinuxCNC.

Правило **Модульності** є надзвичайно важливим. У цих посібниках ви знайдете згадки про інтерпретатор, планувальник завдань, рух або HAL. Кожен з них є модулем або набором модулів. Саме модульність дозволяє вам з’єднувати між собою лише ті частини, які необхідні для роботи вашої машини.

Правило **ясність** є надзвичайно важливим. LinuxCNC є проектом, що постійно розвивається — він не є завершеним і ніколи не буде таким. Він є достатньо досконалим, щоб працювати на більшості машин, на яких ми хочемо його використовувати. Значна частина цього прогресу досягається завдяки тому, що багато користувачів та розробників коду мають можливість переглядати роботу інших та розвивати те, що вони вже зробили.

Правило **Композиції** дозволяє нам побудувати передбачувану систему управління з багатьох доступних модулів, зробивши їх сумісними. Ми досягаємо сумісності, встановлюючи стандартні інтерфейси для наборів модулів і дотримуючись цих стандартів.

Правило **Розділення** вимагає, щоб ми створювали окремі частини, які виконують невеликі завдання. Завдяки розділенню функцій налагодження стає набагато простішим, а замінні модулі можна легко додавати до системи та порівнювати.

Що означає підхід UNIX для вас як користувача LinuxCNC? Це означає, що ви можете вибирати, як використовувати систему. Багато з цих виборів є частиною інтеграції машини, але багато з них також впливають на те, як ви будете використовувати свою машину. Під час читання ви знайдете багато місць, де вам доведеться робити порівняння. Врешті-решт ви зробите вибір: «Я буду використовувати цей інтерфейс, а не той» або «Я буду писати зміщення деталей таким чином, а не іншим». У цих посібниках ми описуємо весь спектр можливостей, доступних на даний момент.

Коли ви починаєте свою подорож у LinuxCNC, ми пропонуємо два застереження: виноска: [Посилання https://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008]

- Перефразуючи слова Дуга Гвіна про UNIX: «LinuxCNC не був розроблений для того, щоб заважати своїм користувачам робити дурниці, оскільки це також завадило б їм робити розумні речі»
- Так само можна сказати і про слова Стівена Кінга: «LinuxCNC зручний у використанні. Він просто невибірковий щодо того, з якими користувачами він дружній.»

Серія відеороликів на YouTube надає безліч доказів того, що перехід на LinuxCNC можливий незалежно від того, якою є ваша звичайна операційна система. З огляду на це, з появою адитивних технологій, таких як 3D-друк, широка IT-спільнота виявляє все більший інтерес до обробки на верстатах з ЧПК, і вам має бути можливо знайти когось із відповідними навичками/обладнанням поблизу, щоб спільно подолати початкові перешкоди.

2.2 Вступ для користувача LinuxCNC

2.2.1 Вступ

Цей документ присвячений використанню LinuxCNC і призначений для читачів, які вже встановили та налаштували цю систему. Деяка інформація про встановлення наведена в наступних розділах. Повна документація щодо встановлення та налаштування міститься в посібнику інтегратора.

2.2.2 Як працює LinuxCNC

LinuxCNC — це набір висококонфігурованих програм для управління фрезерними та токарними верстатами з числовим програмним управлінням (CNC), 3D-принтерами, роботами, лазерними та плазмовими різаками та іншими автоматизованими пристроями. Він здатний забезпечувати скоординоване управління до 9 осей руху.

По суті, LinuxCNC складається з кількох ключових компонентів, які об’єднані разом, утворюючи одну повноцінну систему:

- графічний інтерфейс користувача (GUI), який утворює базовий інтерфейс між оператором, програмним забезпеченням та самим верстатом з CNC;
- Рівень абстракції обладнання (HAL), який забезпечує метод зв’язку всіх різноманітних внутрішніх віртуальних сигналів, що генеруються та отримуються LinuxCNC, із зовнішнім світом,
- контролери високого рівня, які координують генерацію та виконання управління рухом верстата з CNC, а саме контролер руху (EMCMOT), контролер дискретного вводу/виводу (EMCIO) та виконавець завдань (EMCTASK).

Наведена нижче ілюстрація — це проста блок-схема, яка показує, як може виглядати типовий 3-осьовий фрезерний верстат з CNC та кроковими двигунами:

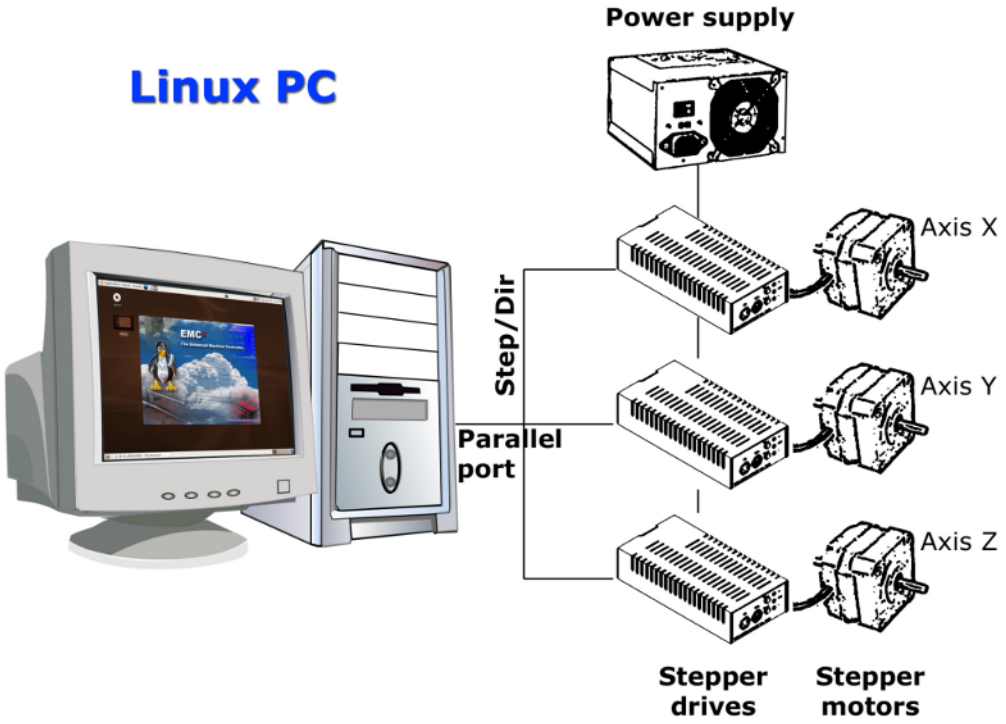


Figure 2.1: Простий верстат з керований LinuxCNC

Комп’ютер під управлінням LinuxCNC надсилає послідовність імпульсів через паралельний порт до крокових приводів, до кожного з яких підключено один кроковий двигун. Кожен привід отримує два незалежні сигнали: один сигнал для управління приводом з метою переміщення пов’язаного з ним крокового двигуна за годинниковою стрілкою або проти годинникової стрілки, а другий сигнал визначає швидкість обертання крокового двигуна.

Хоча на малюнку зображена система крокового двигуна під керуванням паралельного порту, система LinuxCNC також може використовувати широкий спектр спеціальних апаратних інтерфейсів керування рухом для підвищення швидкості та можливостей вводу-виводу. Повний перелік інтерфейсів, що підтримуються LinuxCNC, можна знайти на сторінці Supported Hardware вікі.

У більшості випадків користувачі створюють конфігурацію, специфічну для їхнього верстата, використовуючи або Майстер конфігурації крокового двигуна (для систем CNC, що працюють з використанням паралельного порту комп’ютера), або Майстер апаратного забезпечення Mesa (для більш просунутих систем, що використовують PCI-карту Mesa Anything I/O). Запуск будь-якого з цих майстрів призведе до створення на жорсткому диску комп’ютера декількох папок, що містять низку файлів конфігурації, специфічних для даного верстата з CNC, а також піктограму на робочому столі, що дозволяє легко запускати LinuxCNC.

Наприклад, якщо майстер налаштування крокового двигуна був використаний для створення налаштувань для 3-осьового фрезерного верстата з CNC, зображеного вище під назвою «My_CNC», папки, створені майстром, зазвичай містять такі файли:

- **Папка: `My_CNC`**
  - **`My_CNC.ini`**
    INI-файл містить всю основну інформацію про апаратне забезпечення, що стосується роботи

фрезерного верстата з CNC, таку як кількість кроків, які повинен зробити кожен крокový двигун для виконання одного повного оберту, максимальна швидкість, з якою може працювати кожен крокový двигун, межі переміщення кожної осі або конфігурація та поведінка кінцевих вимикачів на кожній осі.

- **My_CNC.hal**
  Цей файл HAL містить інформацію, яка повідомляє LinuxCNC, як пов'язати внутрішні віртуальні сигнали з фізичними з'єднаннями за межами комп'ютера. Наприклад, вказати контакт 4 на паралельному порту для відправки сигналу напрямку кроку осі Z або наказати LinuxCNC припинити привід двигуна осі X, коли спрацьовує кінцевий вимикач на контакті 13 паралельного порту.
- **custom.hal**
  Налаштування конфігурації верстата, що виходять за межі можливостей майстра, можна виконати, додавши в цей файл HAL додаткові посилання на інші віртуальні точки в LinuxCNC. Під час запуску сеансу LinuxCNC цей файл читається і обробляється до завантаження графічного інтерфейсу користувача. Прикладом може бути ініціювання зв'язку Modbus із двигуном шпинделя, щоб підтвердити його працездатність до відображення графічного інтерфейсу користувача.
- **custom_postgui.hal**
  Файл custom_postgui HAL дозволяє додатково налаштувати LinuxCNC, але відрізняється від custom.HAL тим, що обробляється після відображення графічного інтерфейсу користувача. Наприклад, після встановлення зв'язку Modbus з двигуном шпинделя в custom.hal, LinuxCNC може використовувати файл custom_postgui для зв'язку показань швидкості шпинделя з приводом двигуна з гістограмою, що відображається на графічному інтерфейсі користувача.
- **postgui_backup.hal**
  Це резервна копія файлу custom_postgui.hal, яка дозволяє користувачеві швидко відновити попередню конфігурацію postgui HAL. Це особливо корисно, якщо користувач хоче запустити майстер конфігурації знову під тим самим іменем «My_CNC», щоб змінити деякі параметри фрезерного верстата. Збереження конфігурації фрезерного верстата в майстрі призведе до перезапису існуючого файлу custom_postgui, але файл postgui_backup залишиться без змін.
- **tool.tbl**
  Файл таблиці інструментів містить параметризований список всіх ріжучих інструментів, що використовуються фрезерним верстатом. Ці параметри можуть включати діаметр і довжину різака і використовуються для створення каталогу даних, який повідомляє LinuxCNC, як компенсувати рух інструментів різних розмірів під час фрезерування.

- **Папка: nc_files**
  Папка nc_files надається як місце за замовчуванням для зберігання програм G-коду, що використовуються для керування фрезерним верстатом. Вона також містить низку підпапок із прикладами G-коду.

2.2.3 Графічні інтерфейси користувача

Графічний інтерфейс користувача — це частина LinuxCNC, з якою взаємодіє оператор верстата. LinuxCNC постачається з декількома типами інтерфейсів користувача, які можна вибрати, редагуючи певні поля, що містяться у файлі INI:

AXIS
AXIS, стандартний графічний інтерфейс клавіатури. Це також графічний інтерфейс за замовчуванням, який запускається, коли майстер налаштування використовується для створення панелі запуску значків на робочому столі:

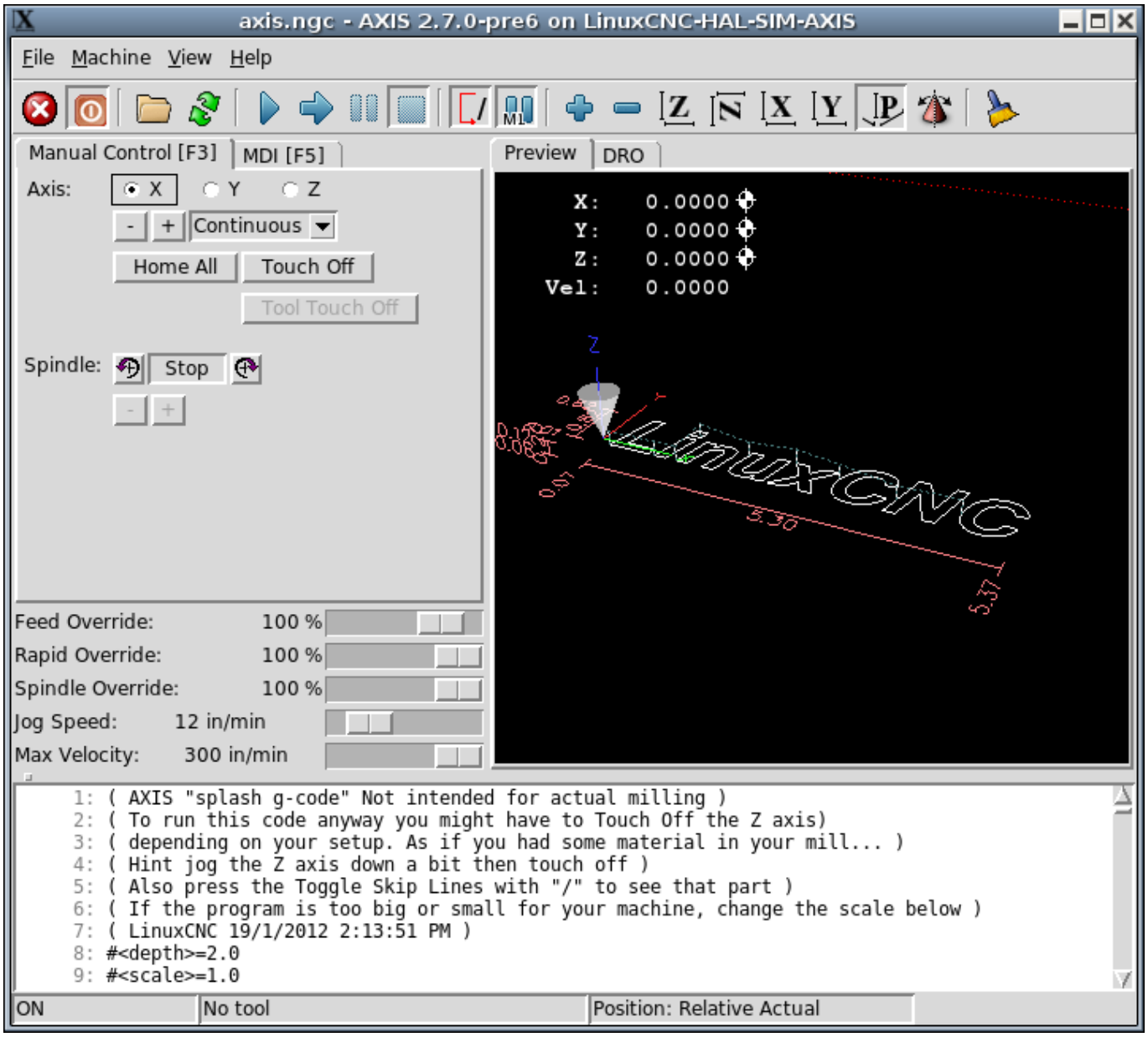


Figure 2.2: AXIS, стандартний інтерфейс графічного інтерфейсу клавіатури

Доторкливий

Touchy, графічний інтерфейс для сенсорних екранів:

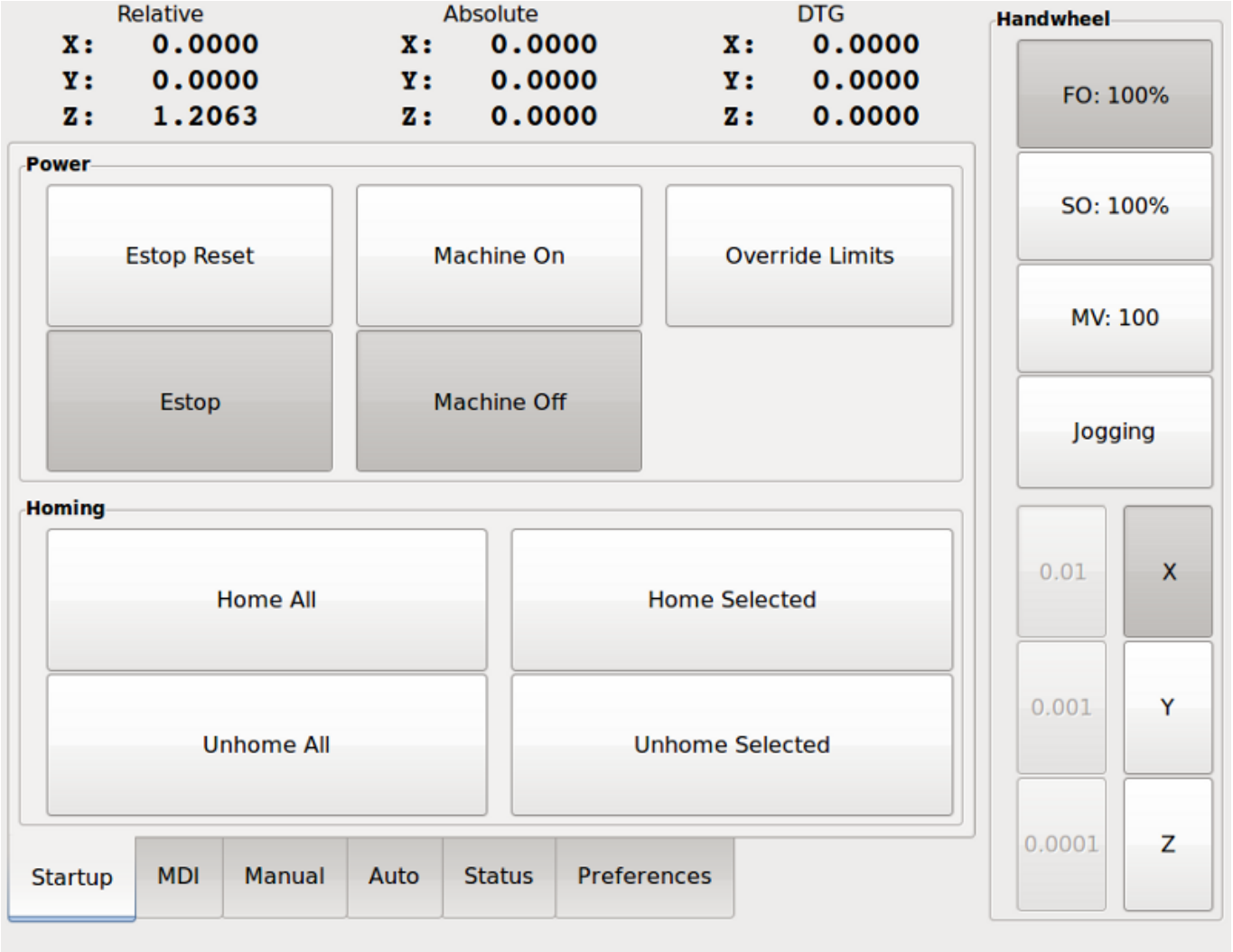


Figure 2.3: Touchy, сенсорний графічний інтерфейс

G-екран

Gscreen, графічний інтерфейс із сенсорним екраном, що налаштовується користувачем:

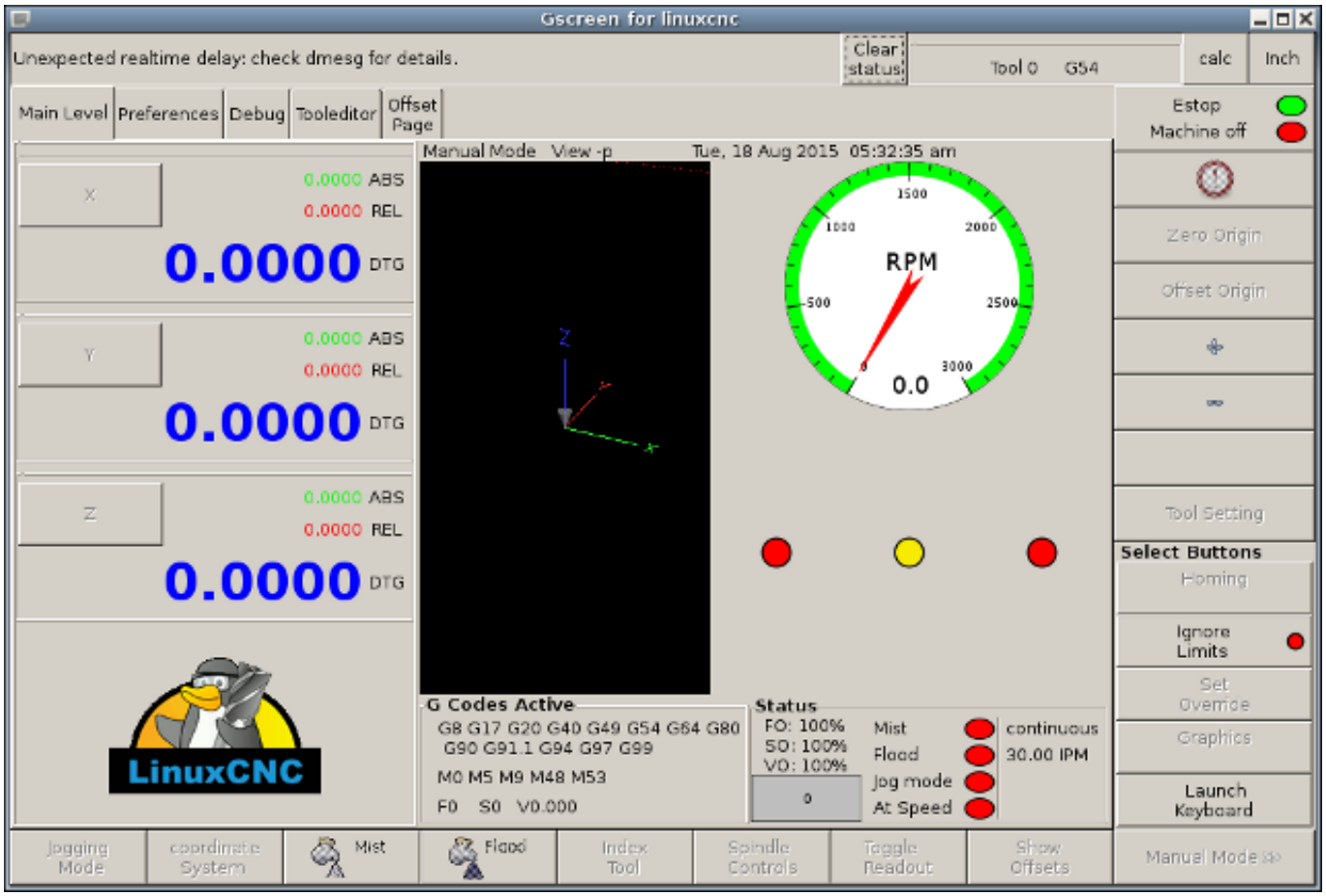


Figure 2.4: Gscreen, налаштовуваний базовий сенсорний екран GUI

GMOCCAPY

GMOCCAPY, Сенсорний графічний інтерфейс користувача на базі Gscreen. GMOCCAPY також розроблений для однаково ефективної роботи в програмах, де клавіатура та миша є переважними методами керування графічним інтерфейсом:

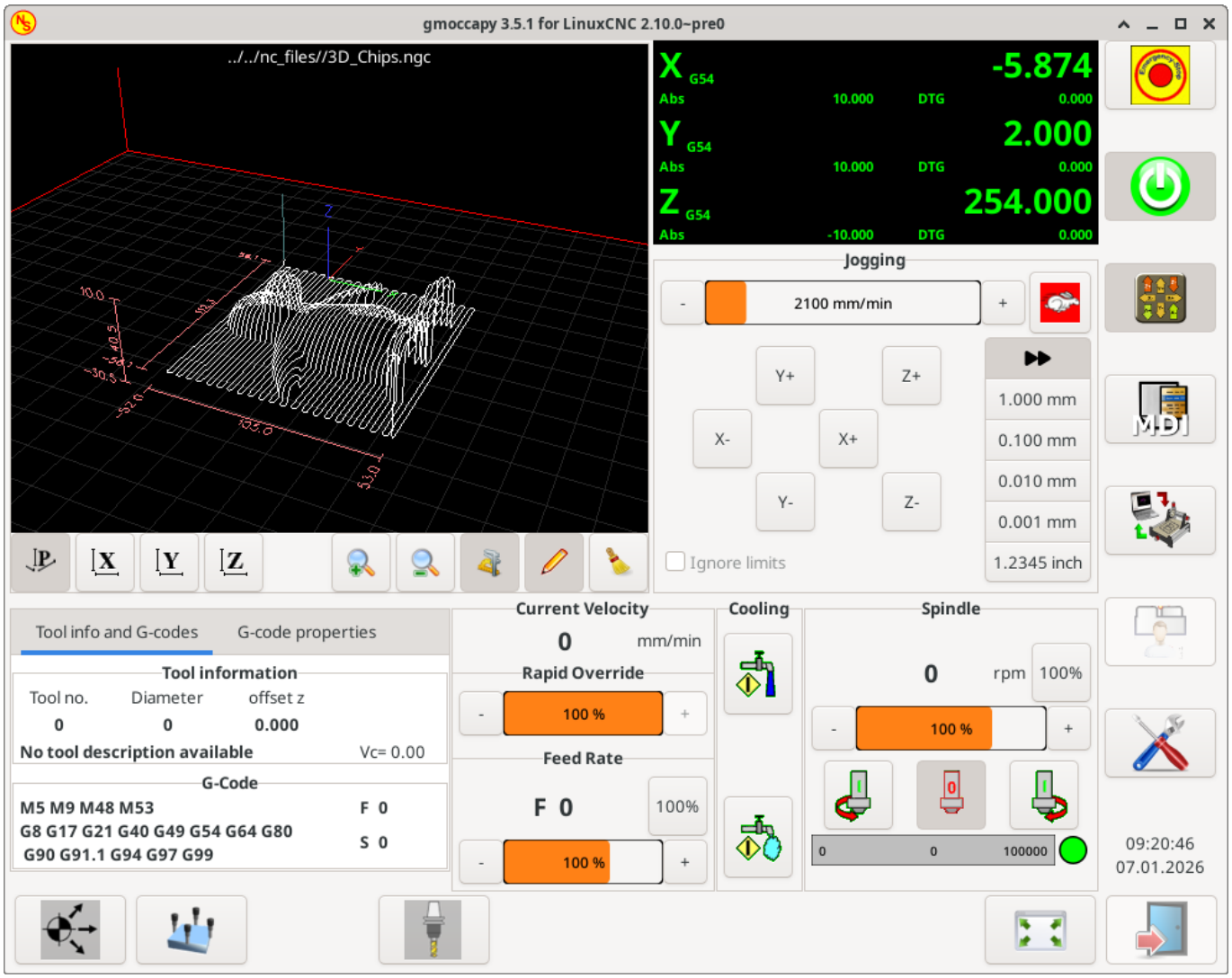


Figure 2.5: GMOCCAPY, графічний інтерфейс із сенсорним екраном на базі Gscreen

NGCGUI

NGCGUI, підпрограма GUI, що забезпечує програмування G-коду у форматі майстра. NGCGUI може працювати як самостійна програма або бути вбудована в інший GUI у вигляді серії вкладок. На наступному знімку екрана показано NGCGUI, вбудовану в AXIS:

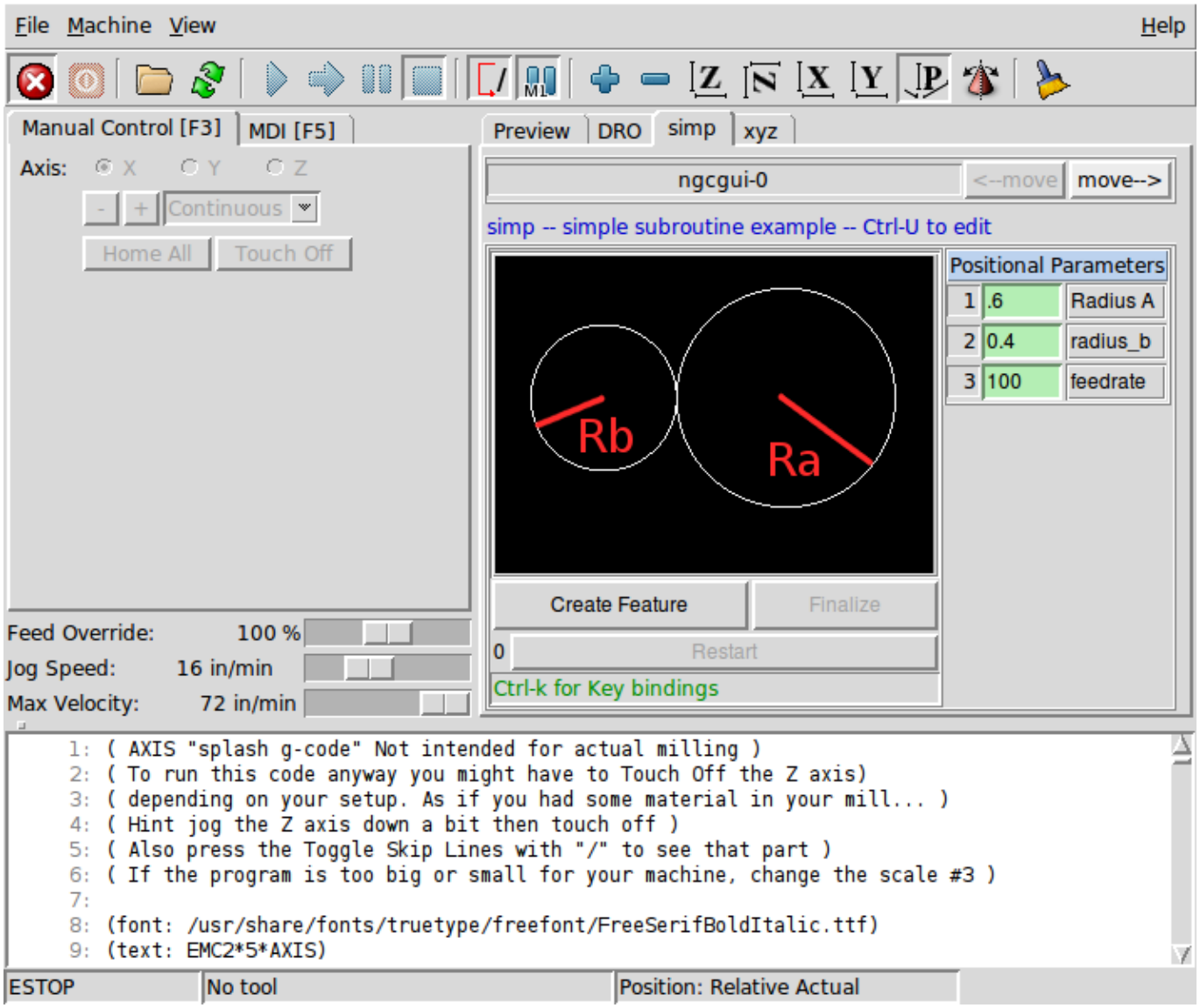


Figure 2.6: NGCGUI, графічний інтерфейс, інтегрований в AXIS

’TkLinuxCNC
TkLinuxCNC, ще один інтерфейс на основі Tcl/Tk. Колись найпопулярніший інтерфейс після AXIS.

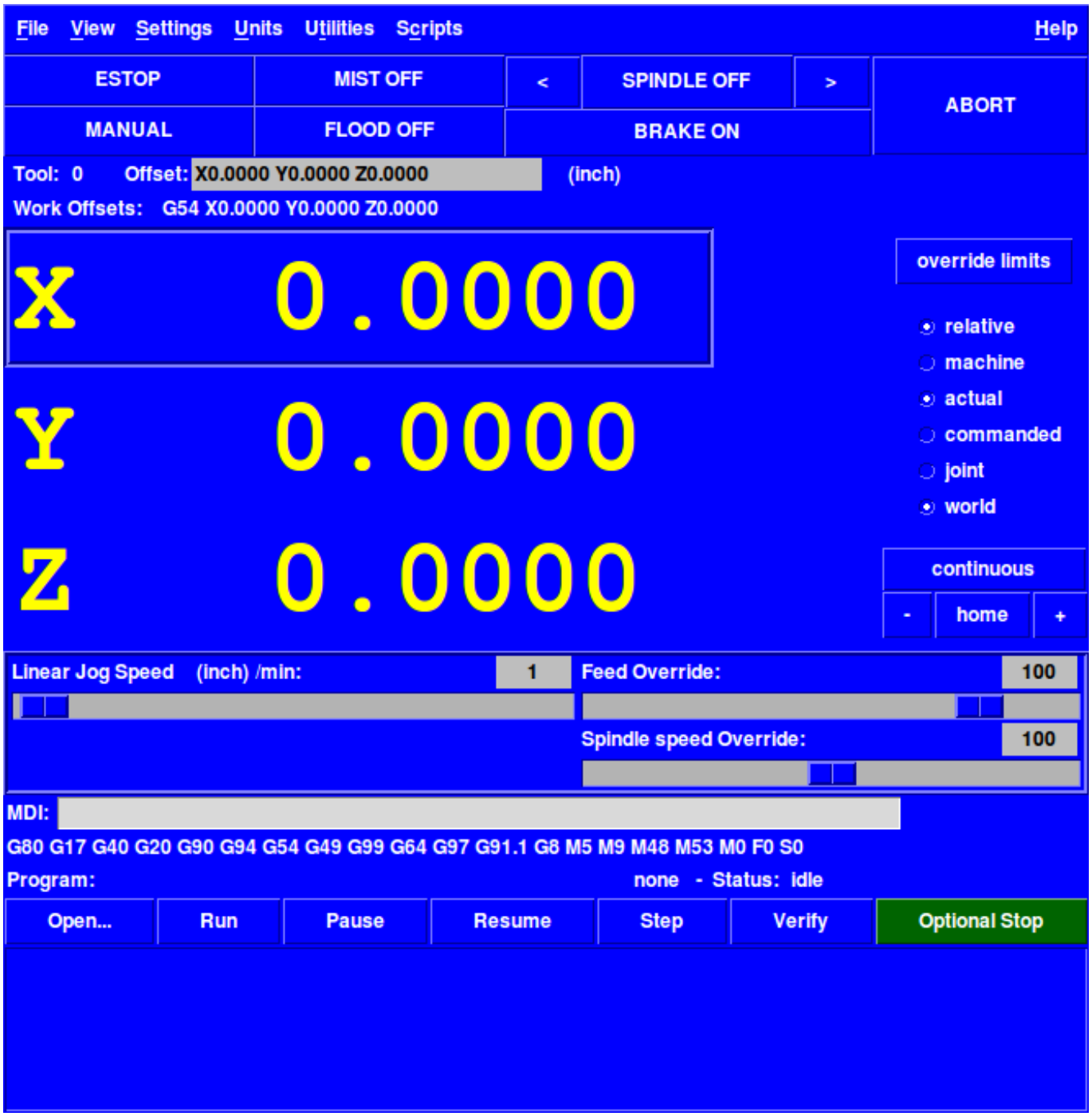


Figure 2.7: Графічний інтерфейс TkLinuxCNC

QtDragon

QtDragon, графічний інтерфейс користувача для сенсорних екранів на базі QtVCP з використанням бібліотеки PyQt5. Він доступний у двох версіях: «QtDragon» і «QtDragon_hd». Вони дуже схожі за функціональністю, але QtDragon_hd призначений для більших моніторів.

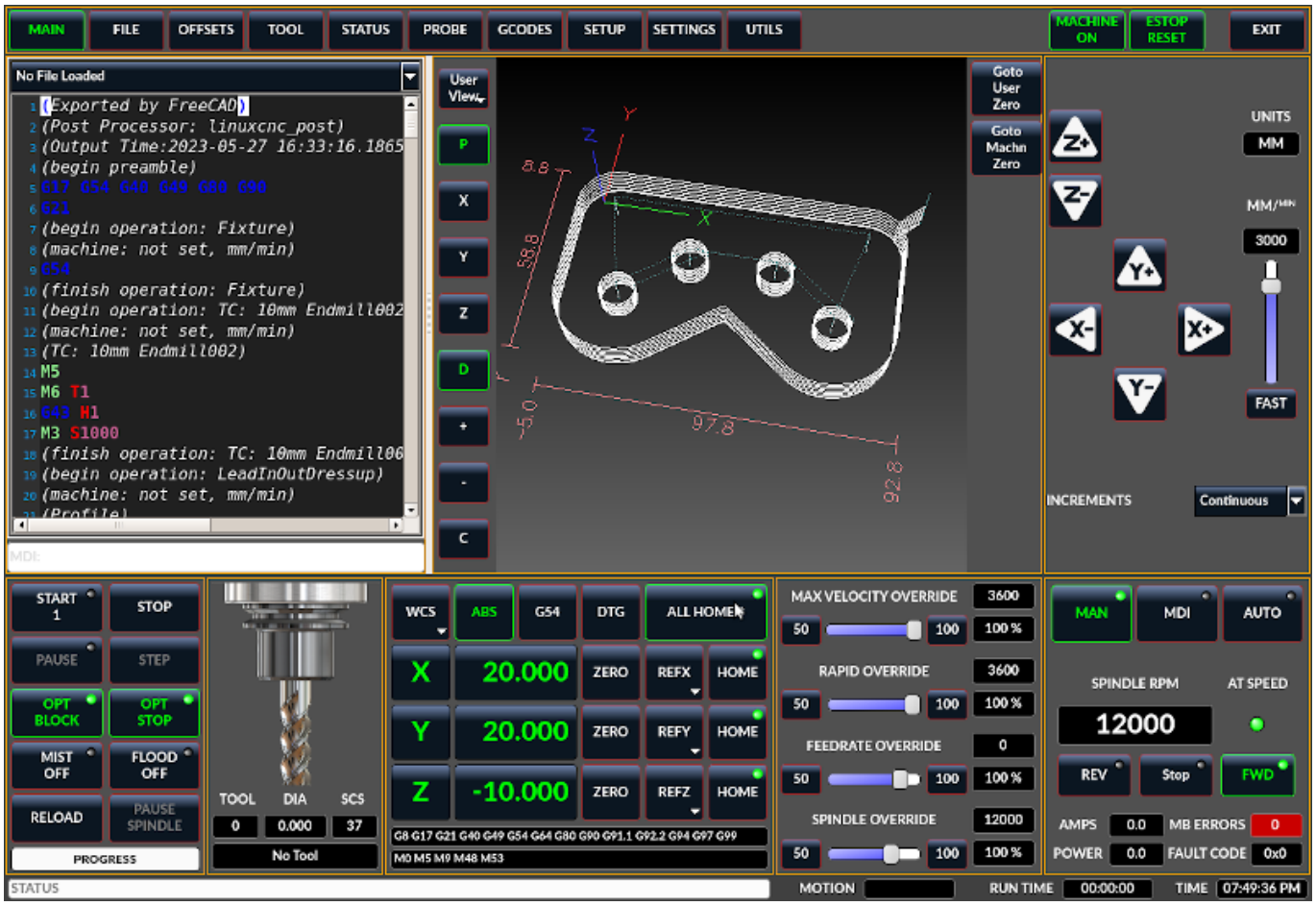


Figure 2.8: QtDragon, графічний інтерфейс із сенсорним екраном на основі QtVCP

QtPlasmaC

QtPlasmaC, Графічний інтерфейс для плазмового різання з сенсорним екраном, заснований на QtVCP та бібліотекі PyQt5. Він доступний у трьох співвідношеннях сторін: 16:9, 4:3 та 9:16.

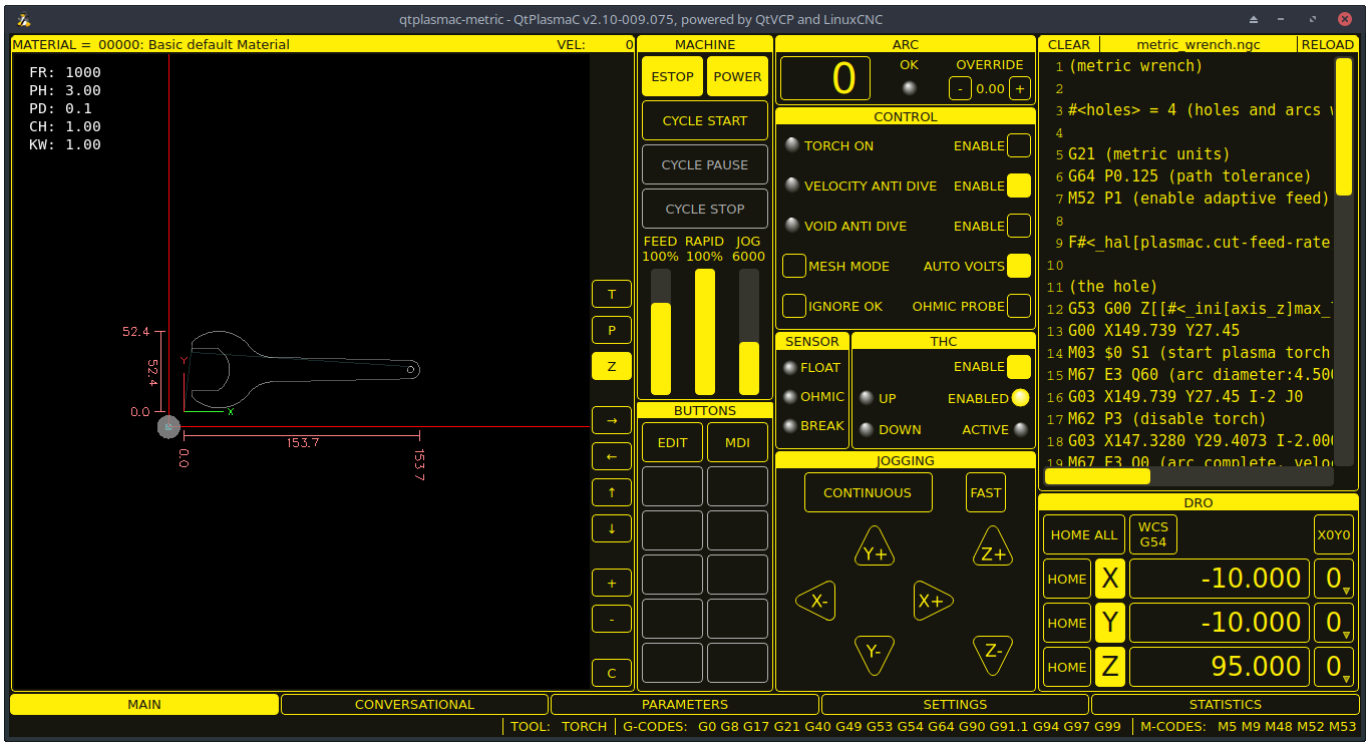


Figure 2.9: QtPlasmaC, сенсорний графічний інтерфейс для плазмового різання на основі QtVCP

2.2.4 Інтерфейс користувача

Ці інтерфейси користувача є способом взаємодії з LinuxCNC поза межами графічних інтерфейсів користувача.

halui
: Інтерфейс користувача на основі HAL, що дозволяє керувати LinuxCNC за допомогою кнопок та перемикачів

linuxcncrsh
: Інтерфейс користувача на основі Telnet, що дозволяє надсилати команди з віддалених комп’ютерів

2.2.5 Віртуальна панель керування

Як згадувалося вище, багато графічних інтерфейсів LinuxCNC можуть бути налаштовані користувачем. Це може бути зроблено для додавання індикаторів, перемикачів або повзунків до базового вигляду одного з графічних інтерфейсів для підвищення гнучкості або функціональності. У LinuxCNC пропонуються два стилі віртуальної панелі керування:

PyVCP
: PyVCP — віртуальна панель керування на базі Python, яку можна додати до графічного інтерфейсу AXIS. PyVCP використовує лише віртуальні сигнали, що містяться в шарі апаратної абстракції, такі як індикатор швидкості шпинделя або вихідний сигнал аварійної зупинки, і має простий вигляд без зайвих деталей. Це робить її чудовим вибором, якщо користувач хоче додати віртуальну панель керування з мінімальними зусиллями.

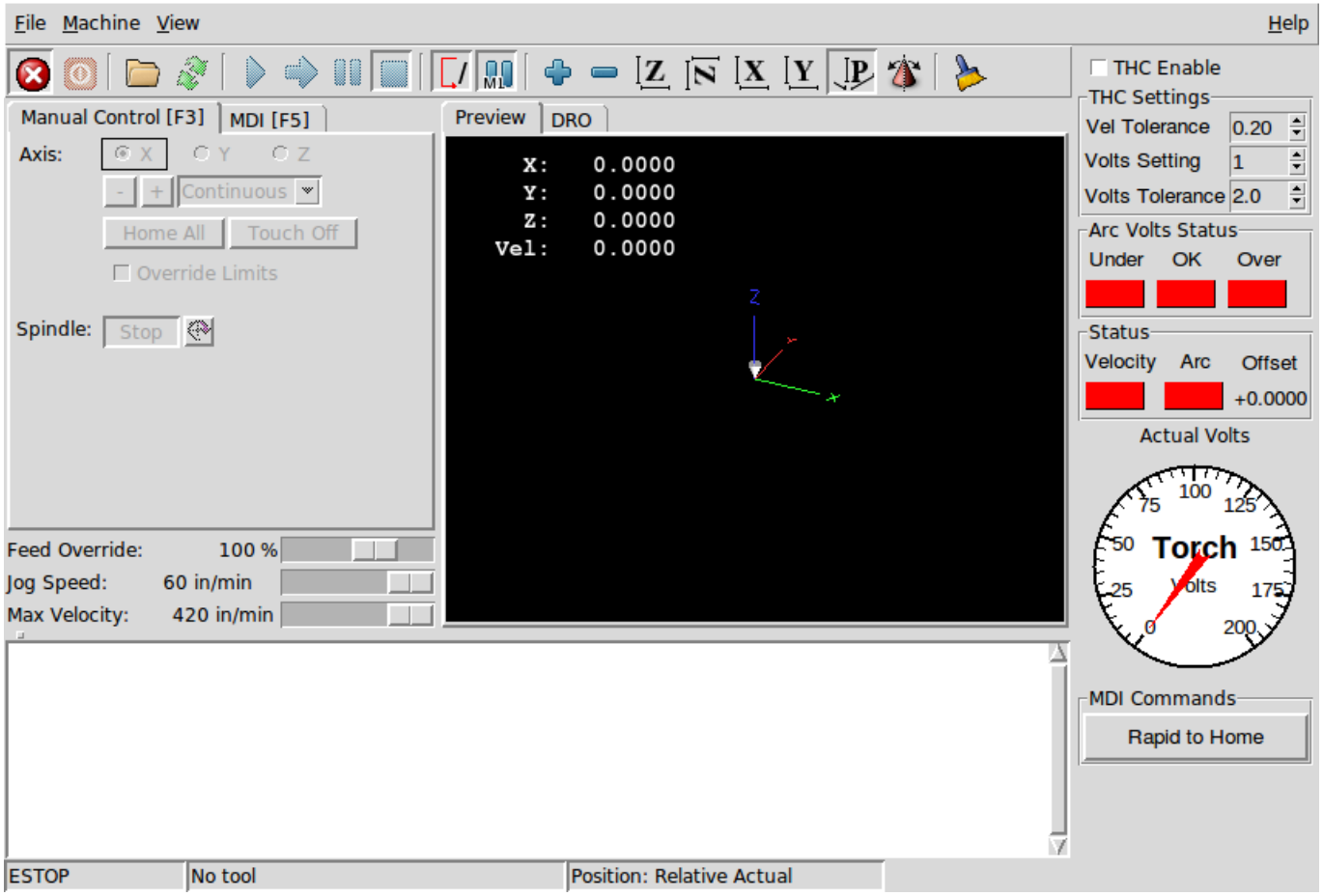


Figure 2.10: Приклад PyVCP, вбудований у графічний інтерфейс AXIS

GladeVCP

GladeVCP, віртуальна панель керування на базі Glade, яку можна додати до графічних інтерфейсів AXIS або Touchy. GladeVCP має перевагу над PyVCP, оскільки не обмежується відображенням або керуванням віртуальними сигналами HAL, а може включати інші зовнішні інтерфейси поза LinuxCNC, такі як події вікна або мережі. GladeVCP також є більш гнучкою в тому, як її можна налаштувати для відображення в графічному інтерфейсі:

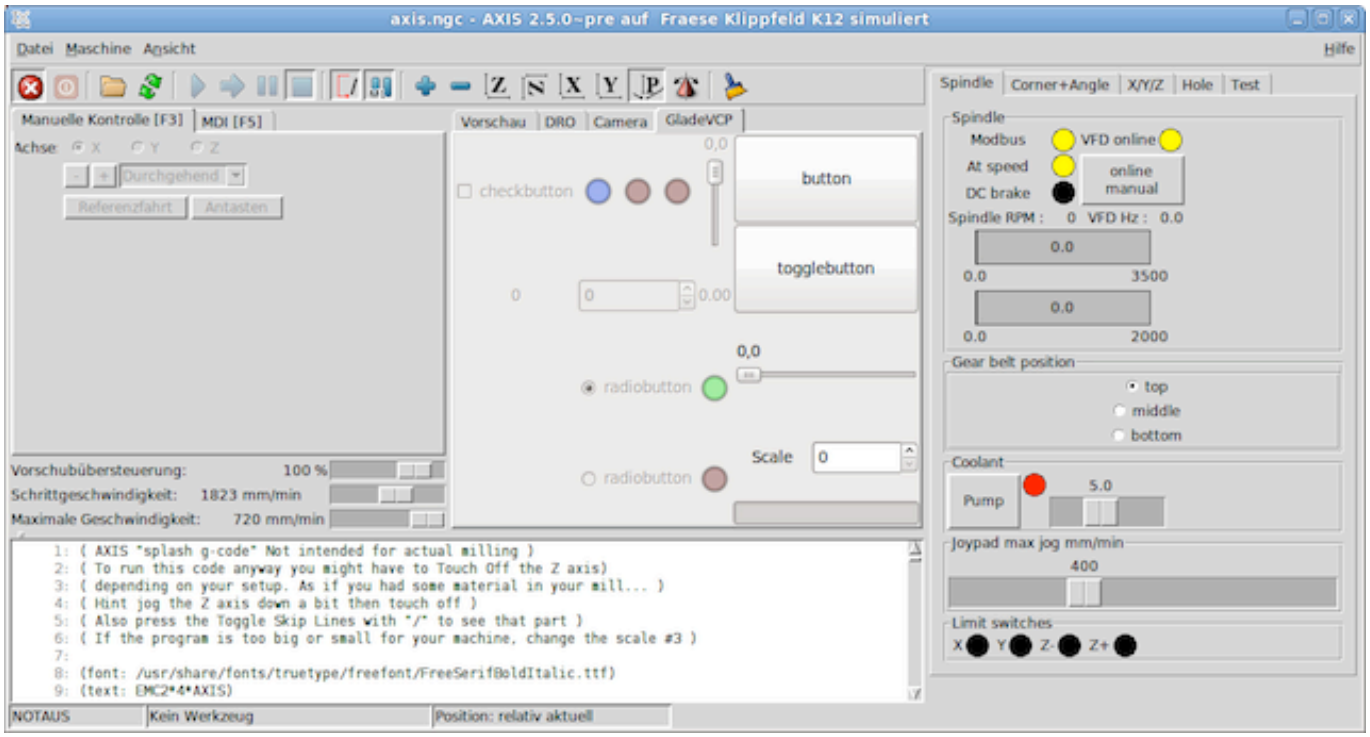


Figure 2.11: Приклад GladeVCP, вбудований у графічний інтерфейс AXIS

QtVCP

QtVCP, віртуальна панель керування на базі PyQt5, яку можна додати до більшості графічних інтерфейсів користувача або запустити як самостійну панель. QtVCP має перевагу над PyVCP, оскільки не обмежується відображенням або керуванням віртуальними сигналами HAL, а може включати інші зовнішні інтерфейси поза LinuxCNC, такі як події вікна або мережі, шляхом розширення за допомогою коду python. QtVCP також є більш гнучким у налаштуванні для відображення в графічному інтерфейсі користувача за допомогою багатьох спеціальних віджетів:

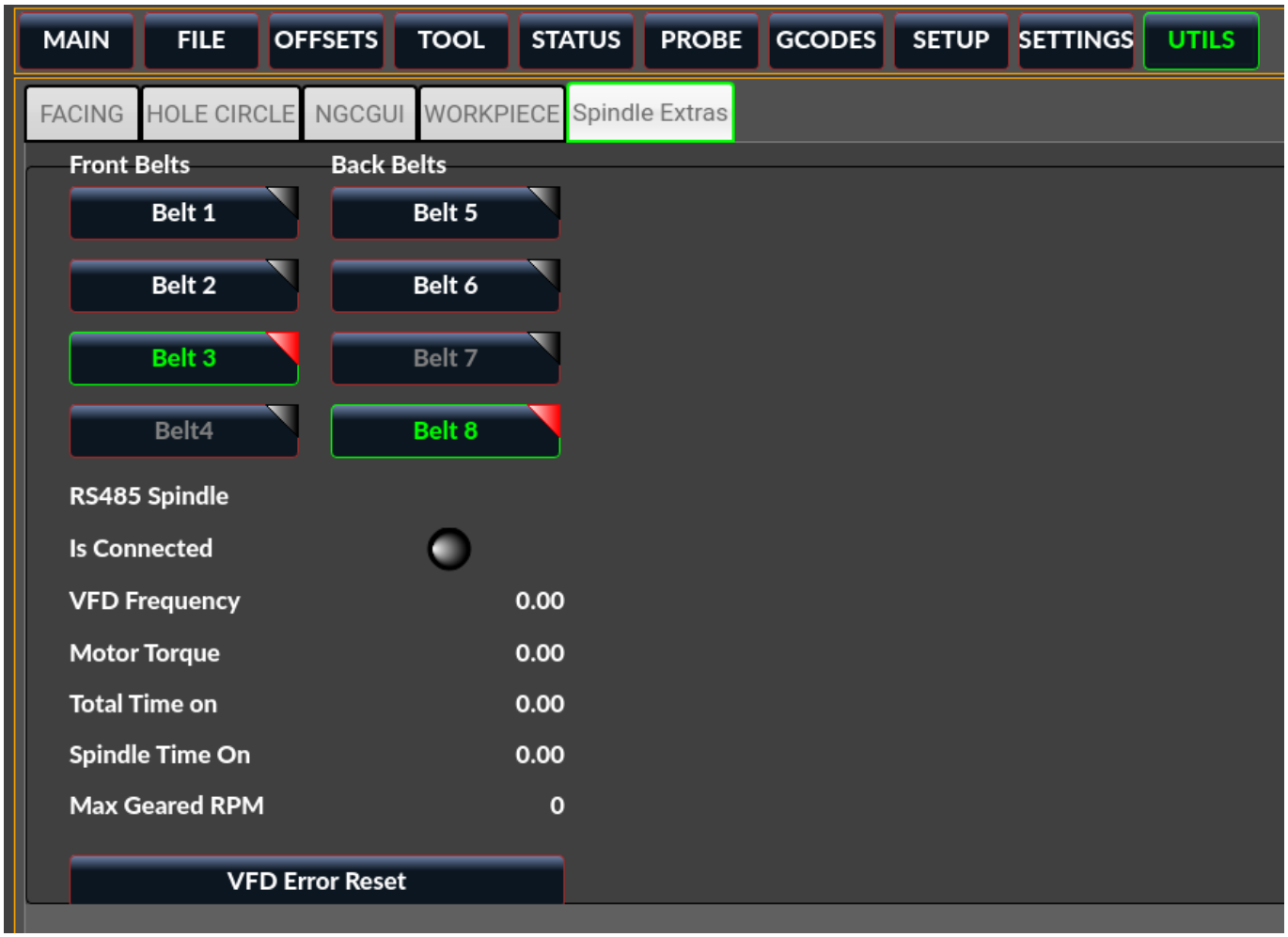


Figure 2.12: Приклад QtVCP, вбудований у графічний інтерфейс QtDragon

2.2.6 Мови

LinuxCNC використовує файли перекладу для перекладу інтерфейсів користувача LinuxCNC на багато мов, включаючи французьку, німецьку, італійську, фінську, російську, румунську, португальськ та китайську. Якщо переклад вже створено, LinuxCNC автоматично використовуватиме мову, якою ви входите в систему, під час запуску операційної системи Linux. Якщо ваша мова не перекладена, зверніться за допомогою до розробника в IRC, списку розсилки або форумі користувачів.

2.2.7 Думай як оператор CNC

Цей посібник не претендує на те, щоб навчити вас користуватися токарним або фрезерним верстатом. Щоб стати досвідченим оператором, потрібно багато часу і багато роботи. Один автор колись сказав: «Ми вчимося на досвіді, якщо він у нас є». Зламані інструменти, пошкоджені лещата і шрами — це свідчення отриманого досвіду. Гарне покриття, жорсткі допуски і обережність під час роботи — це свідчення отриманого досвіду. Жодна машина чи програма не може замінити людський досвід.

Тепер, коли ви починаєте працювати з програмним забезпеченням LinuxCNC, ви повинні поставити себе на місце оператора. Ви повинні виконувати роль людини, відповідальної за роботу верстата. Це верстат, який буде чекати ваших команд і виконувати ваші вказівки. На цих сторінках ми надамо пояснення, які допоможуть вам стати хорошим оператором CNC з LinuxCNC.

2.2.8 Режими роботи

Під час роботи LinuxCNC існує три основні режими введення команд. Це ручний, автоматичний та ручний ввід даних (MDI). Перехід з одного режиму в інший суттєво впливає на поведінку системи управління LinuxCNC. У одному режимі можна виконати певні дії, які неможливо виконати в іншому. Оператор може повернути вісь у вихідне положення в ручному режимі, але не в автоматичному або MDI. Оператор може змусити верстат виконати весь файл, заповнений G-кодами, в автоматичному режимі, але не в ручному або MDI.

У ручному режимі кожна команда вводиться окремо. У людських термінах ручна команда може бути «увімкнути охолодження» або «перемістити X зі швидкістю 25 дюймів на хвилину». Це приблизно еквівалентно натисканню перемикача або повороту маховика для осі. Ці команди зазвичай обробляються на одному з графічних інтерфейсів натисканням кнопки мишею або утриманням клавіші на клавіатурі. В автоматичному режимі подібна кнопка або клавіша може використовуватися для завантаження або запуску виконання всієї програми G-коду, яка зберігається у файлі. У режимі MDI оператор може ввести блок коду і наказати верстату виконати його, натиснувши клавішу <return> або <enter> на клавіатурі.

Деякі команди керування рухом доступні одночасно і спричиняють однакові зміни руху в усіх режимах. До них належать «Перервати», «Аварійна зупинка» та «Перевищення швидкості подачі». Такі команди не потребують додаткових пояснень.

Інтерфейс користувача AXIS приховує деякі відмінності між режимом Auto та іншими режимами, роблячи автоматичні команди доступними в більшості випадків. Він також стирає відмінності між режимами Manual та MDI, оскільки деякі команди Manual, такі як Touch Off, фактично реалізуються шляхом надсилання команд MDI. Це досягається шляхом автоматичного переходу до режиму, необхідного для виконання дії, яку запросив користувач.

2.3 Важливі концепції користувача

У цьому розділі розглядаються важливі концепції для користувача, які слід розуміти перед тим, як спробувати запустити верстат з CNC за допомогою G-коду.

2.3.1 Контроль траєкторії

2.3.1.1 Планування траєкторії

Планування траєкторії, загалом, – це засіб, за допомогою якого LinuxCNC слідує траєкторією, заданою вашою програмою G-коду, працюючи при цьому в межах можливостей вашого обладнання.

Програму G-коду ніколи не можна повністю виконати. Наприклад, уявіть, що ви вказуєте як однорядкову програму наступний рух:

```
G1 X1 F10 (G1 b''–b'' b''лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''иb''b''йb'' b''pb''b' ↩
    'yb''b''xb'', X1 b''–b'' b''пb''b''yb''b''нb''b''кb''b''тb'' b''пb''b''pb''b''иb''b' ↩
    'зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'', F10 b''–b'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'')
```

Насправді весь рух не може бути виконаний на F10, оскільки верстат повинен прискоритися з місця, рухатися до X=1, а потім сповільнитися, щоб знову зупинитися. Іноді частина руху виконується зі швидкістю F10, але для багатьох рухів, особливо коротких, задана швидкість подачі взагалі не досягається. Наявність коротких рухів у вашому G-коді може призвести до уповільнення та прискорення вашої машини для довших рухів, якщо «наївний детектор кулачка» не використовується з G64 P*n*.

Описані вище основні прискорення та уповільнення не є складними і не вимагають компромісів. У файлі INI зазначені обмеження машини, такі як максимальна швидкість осі та прискорення осі, повинні дотримуватися планувальником траєкторії.

Для отримання додаткової інформації про параметри INI-файлу Планувальника траєкторії див. розділ Траєкторія у розділі INI.

2.3.1.2 Слідування шляхом

Менш простою проблемою є проходження траєкторії. Коли ви програмуєте кут у G-коді, планувальник траєкторії може виконувати кілька дій, усі з яких у деяких випадках правильні:

- Він може сповільнитися до повної зупинки точно в координатах кута, а потім прискоритися в новому напрямку.
- Він також може виконувати так звану плавну обробку, тобто підтримувати високу швидкість подачі під час проходження через кут, що робить необхідним заокруглення кута для дотримання обмежень верстата.

Ви можете побачити, що тут є компроміс: ви можете уповільнити швидкість, щоб отримати краще відстеження траєкторії, або зберегти швидкість і отримати гірше відстеження траєкторії. Залежно від конкретного різання, матеріалу, інструменту тощо, програміст може захотіти піти на інший компроміс.

Швидкі рухи також підкоряються поточній траєкторії руху. За допомогою рухів, достатньо довгих для досягнення максимальної швидкості на верстаті з низьким прискоренням і без заданої похибки траєкторії, можна отримати досить округлий кут.

2.3.1.3 Програмування Планувальника

Команди керування траєкторією такі:

G61
(Режим точного шляху) G61 точно потрапляє до запрограмованої точки, навіть якщо це означає, що він може тимчасово повністю зупинитися, щоб змінити напрямок до наступної запрограмованої точки.

G61.1
(Режим точної зупинки) G61.1 вказує планувальнику зупинятися точно в кінці кожного сегмента. Шлях буде пройдено точно, але повна зупинка подачі може бути руйнівною для деталі або інструменту, залежно від особливостей обробки.

G64
(Режим змішування без толерантності) G64 є стандартним налаштуванням при запуску LinuxCNC. G64 є лише змішуванням, а наївний детектор кулачка не ввімкнений. G64 та G64 P0 вказують планувальнику пожертвувати точністю відстеження траєкторії, щоб зберегти швидкість подачі. Це необхідно для деяких типів матеріалів або інструментів, де точні зупинки є шкідливими, і може чудово працювати, якщо програміст пам'ятає, що траєкторія інструменту буде дещо більш криволінійною, ніж зазначено в програмі. При використанні G0 (швидких) переміщень з G64 будьте обережні при переміщеннях з проміжком і залишайте достатню відстань для уникнення перешкод, виходячи з можливостей прискорення вашої машини.

G64 P- Q-
(Режим змішування з допуском) Ця опція вмикає «наївний детектор кулачка» і дозволяє змішувати з допуском. Якщо ви програмуєте G64 P0.05, ви повідомляєте планувальнику, що

хочете безперервну подачу, але на запрограмованих кутах ви хочете, щоб вона сповільнювалася настільки, щоб траєкторія інструменту залишалася в межах 0,05 одиниць користувача від запрограмованої траєкторії. Точна величина уповільнення залежить від геометрії запрограмованого кута і обмежень верстата, але єдине, про що повинен турбуватися програміст, це допуск. Це дає програмісту повний контроль над компромісом між траєкторією і допуском. Допуск змішування можна змінювати протягом програми за необхідності. Зверніть увагу, що специфікація G64 P0 має той самий ефект, що і G64 окремо (вище), що необхідно для зворотної сумісності зі старими програмами G-коду. Дивіться розділ G64 глави G-коду.

Змішування без допуску
Контрольована точка торкнеться кожного зазначеного руху принаймні в одній точці. Машина ніколи не рухатиметься з такою швидкістю, що не зможе точно зупинитися в кінці поточного руху (або наступного руху, якщо ви зробите паузу, коли змішування вже розпочалося). Відстань від кінцевої точки руху є такою великою, як це необхідно для підтримання найкращої подачі контуру.

Наївний CAM-детектор
Послідовні рухи G1, що включають тільки осі XYZ, які відхиляються від прямої лінії менше ніж на Q-, об'єднуються в одну пряму лінію. Цей об'єднаний рух замінює окремі рухи G1 з метою згладжування з допуском. Між послідовними рухами контрольована точка пройде не більше ніж P- від фактичних кінцевих точок рухів. Контрольована точка торкнеться принаймні однієї точки на кожному русі. Машина ніколи не рухатиметься з такою швидкістю, що не зможе точно зупинитися в кінці поточного руху (або наступного руху, якщо ви зробите паузу, коли змішування вже розпочалося). При рухах G2/3 в площині G17 (XY), коли максимальне відхилення дуги від прямої лінії менше, ніж допуск G64 Q, дуга розбивається на дві лінії (від початку дуги до середини і від середини до кінця). Ці лінії потім підлягають наївному алгоритму кулачка для ліній. Таким чином, випадки лінії-дуги, дуги-дуги та дуги-лінії, а також лінії-лінії отримують переваги від «наївного детектора кулачка». Це покращує продуктивність контурування шляхом спрощення траєкторії.

На наступному малюнку синя лінія відображає фактичну швидкість машини. Червоні лінії відображають здатність машини до прискорення. Горизонтальні лінії під кожним графіком відображають заплановану рух. Верхній графік показує, як планувальник траєкторії сповільнює машину при коротких рухах, щоб залишатися в межах налаштувань прискорення машини і мати можливість точно зупинитися в кінці наступного руху. Нижній графік показує ефект Naive Cam Detector, який об'єднує рухи і краще підтримує заплановану швидкість.

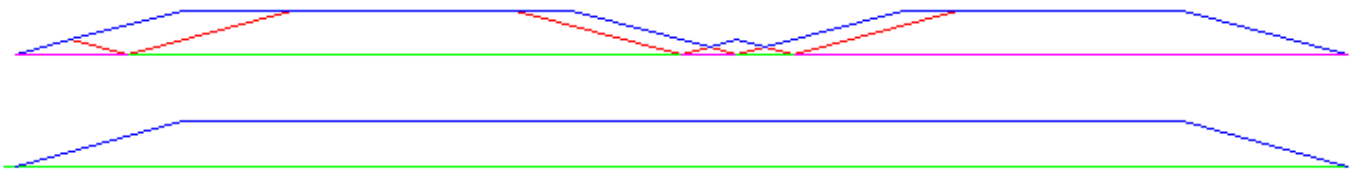


Figure 2.13: Наївний CAM-детектор

2.3.1.4 Планування переїздів

Переконайтеся, що рухи є «достатньо довгими» для вашої машини/матеріалу. В основному через правило, що машина ніколи не рухатиметься з такою швидкістю, яка не дозволить їй повністю зупинитися в кінці поточного руху, існує мінімальна довжина руху, яка дозволить машині підтримувати задану швидкість подачі з заданим налаштуванням прискорення.

Фази прискорення та уповільнення використовують половину значення MAX_ACCELERATION з файлу INI. У випадку точного зворотного руху це призводить до того, що загальне прискорення

осі дорівнює значенню MAX_ACCELERATION з файлу INI. В інших випадках фактичне прискорення машини дещо менше, ніж прискорення з файлу INI.

Щоб підтримувати швидкість подачі, рух повинен бути довшим, ніж відстань, необхідна для прискорення від 0 до бажаної швидкості подачі, а потім знову зупинки. Використовуючи A як **1/2** файлу INI MAX_ACCELERATION і F як швидкість подачі **в одиницях за секунду**, час прискорення становить $\mathbf{t_a = F/A}$, а відстань прискорення — $\mathbf{d_a = F*t_a/2}$. Час і відстань уповільнення є однаковими, що робить критичну відстань $\mathbf{d = d_a + d_d = 2 * d_a = F^2/A}$.

Наприклад, для швидкості подачі 1 дюйм за секунду та прискорення **10 дюймів/с^2** критична відстань становить $\mathbf{1^2/10 = 1/10 = 0{,}1}$ **дюйма**.

Для швидкості подачі 0,5 дюйма за секунду критична відстань становить $\mathbf{5^2/100 = 25/100 = 0{,}025}$ дюйма.

2.3.2 G-код

2.3.2.1 Значення за замовчуванням

При першому запуску LinuxCNC за замовчуванням завантажується багато G- та M-кодів. Поточні активні G- та M-коди можна переглянути на вкладці MDI у вікні «Active G-codes:» (Активні G-коди) в інтерфейсі AXIS. Ці G- та M-коди визначають поведінку LinuxCNC, і перед запуском LinuxCNC важливо зрозуміти, що робить кожен з них. За замовчуванням можна змінити під час запуску файлу G-коду і залишити в іншому стані, ніж під час запуску сеансу LinuxCNC. Найкраща практика — встановити необхідні для роботи параметри за замовчуванням у преамбулі файлу G-коду і не припускати, що параметри за замовчуванням не змінилися. Роздрукування сторінки G-коду Quick Reference може допомогти вам запам’ятати, що означає кожен з них.

2.3.2.2 Швидкість подачі

Спосіб застосування швидкості подачі залежить від того, чи є вісь, що бере участь у переміщенні, обертовою. Прочитайте та зрозумійте розділ Feed Rate, якщо у вас обертова вісь або токарний верстат.

2.3.2.3 Зміщення радіуса інструмента

Зсув радіуса інструменту (G41/42) вимагає, щоб інструмент міг торкатися будь-якої точки вздовж кожного запрограмованого руху, не зачіпаючи два сусідні рухи. Якщо це неможливо з поточним діаметром інструменту, ви отримаєте помилку. Інструмент меншого діаметра може працювати без помилок на тому самому шляху. Це означає, що ви можете запрограмувати фрезу так, щоб вона проходила по шляху, який є вужчим за фрезу, без будь-яких помилок. Докладнішу інформацію див. у розділі Cutter Compensation.

2.3.3 Самонаведення

Після запуску LinuxCNC кожна вісь повинна бути повернена в початкове положення перед запуском програми або виконанням команди MDI. Якщо ваша машина не має перемикачів початкового положення, позначка на кожній осі може допомогти повернути координати машини в одне і те ж місце кожного разу. Після повернення в початкове положення будуть використовуватися м’які обмеження, встановлені в файлі INI.

Якщо ви хочете відхилитися від стандартної поведінки або хочете використовувати інтерфейс Mini, вам потрібно встановити опцію `NO_FORCE_HOMING = 1` у розділі `[TRAJ]` вашого INI-файлу. Більш детальну інформацію про повернення в початкове положення можна знайти в посібнику з інтегратора.

2.3.4 Зміни інструментів

Існує кілька варіантів ручної заміни інструментів. Інформацію про налаштування цих варіантів див. у розділі [EMCIO]. Також див. розділи G28 та G30 у розділі «G-код».

2.3.5 Системи координат

Спочатку системи координат можуть здаватися складними. Перед запуском верстата з ЧПК необхідно зрозуміти основи систем координат, що використовуються в LinuxCNC. Детальна інформація про системи координат LinuxCNC наведена в розділі Система координат цього посібника.

2.3.5.1 G53 Координати машини

Під час переведення LinuxCNC у вихідне положення ви встановлюєте систему координат верстата G53 на 0 для кожної осі, переведеної у вихідне положення.

Жодні інші системи координат або зміщення інструменту не змінюються під час повернення до початкового положення.

Єдиний раз, коли ви переміщуєтеся в системі координат машини G53, це коли ви програмуєте G53 на тому ж рядку, що й переміщення. Зазвичай ви перебуваєте в системі координат G54.

2.3.5.2 G54-59.3 Координати користувача

Зазвичай використовується система координат G54. Коли до поточної системи координат користувача застосовується зміщення, невелика синя кулька з лініями буде знаходитися в точці machine origin, коли ваш DRO відображає «Position: Relative Actual» в AXIS. Якщо ваші зміщення є тимчасовими, використовуйте систему координат нуль з меню Machine або програму «G10 L2 P1 X0 Y0 Z0» в кінці файлу G-коду. Змініть число «P» відповідно до системи координат, в якій ви хочете очистити зміщення.

- Зміщення, збережені в системі координат користувача, зберігаються після завершення роботи LinuxCNC.
- За допомогою кнопки «Touch Off» в AXIS встановлюється зміщення для вибраної системи координат користувача.

2.3.5.3 Коли ти загубився

Якщо у вас виникають проблеми з отриманням значення 0,0,0 на DRO, коли вам здається, що це необхідно, можливо, у вас запрограмовано деякі зміщення, і їх потрібно вилучити.

- Переміщення до початку координат машини за допомогою G53 G0 X0 Y0 Z0
- Очистіть будь-яке зміщення G92 за допомогою G92.1
- Використовуйте систему координат G54 з G54
- Встановіть систему координат G54 такою ж, як і систему координат верстата, за допомогою *G10 L2 P1 X0 Y0 Z0 R0*.
- Вимкніть зміщення інструменту за допомогою G49
- Увімкніть відображення відносних координат у меню

Тепер ви повинні бути в початку координат машини X0, Y0, Z0, а відносна система координат має збігатися з системою координат машини.

2.3.6 Конфігурації машини

На наступній схемі показано типовий фрезерний верстат із зазначенням напрямку руху інструменту, фрезерного столу та кінцевих вимикачів. Зверніть увагу, як фрезерний стіл рухається у напрямку, протилежному стрілкам декартової системи координат, показаним на зображенні «Напрямок руху інструменту». Це забезпечує рух «інструменту» у правильному напрямку відносно матеріалу.

Зверніть також увагу на положення кінцевих вимикачів і напрямок активації їх кулачків. Можливі кілька комбінацій, наприклад, можна (на відміну від креслення) розмістити один фіксований кінцевий вимикач посередині столу і два рухомі кулачки для його активації. У цьому випадку межі будуть змінені на протилежні: +X буде праворуч від столу, а -X — ліворуч. Ця зміна не впливає на напрямок руху інструменту.

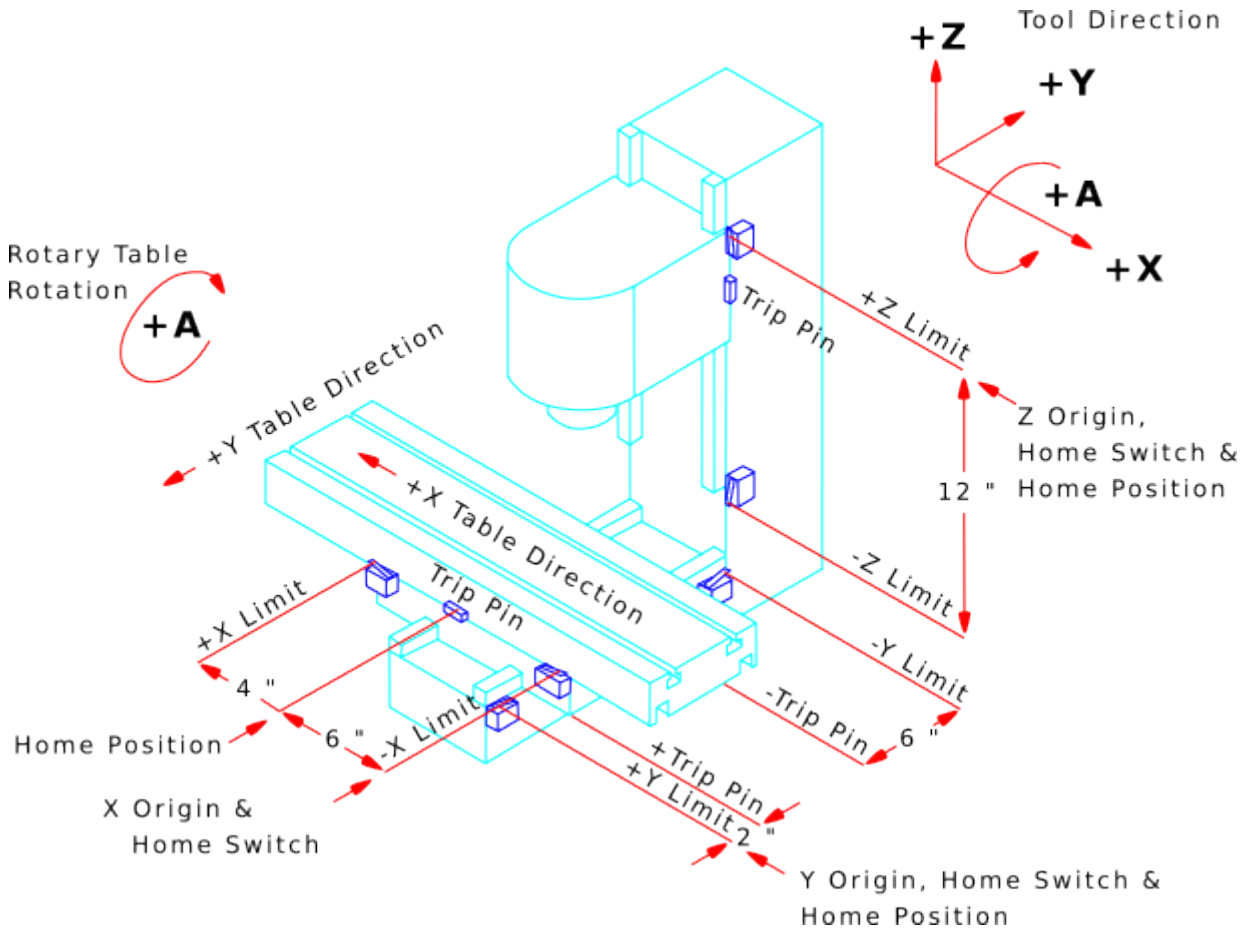


Figure 2.14: Типова конфігурація млина

На наступній діаграмі показано типовий токарний верстат із зазначенням напрямку руху інструменту та кінцевих вимикачів.

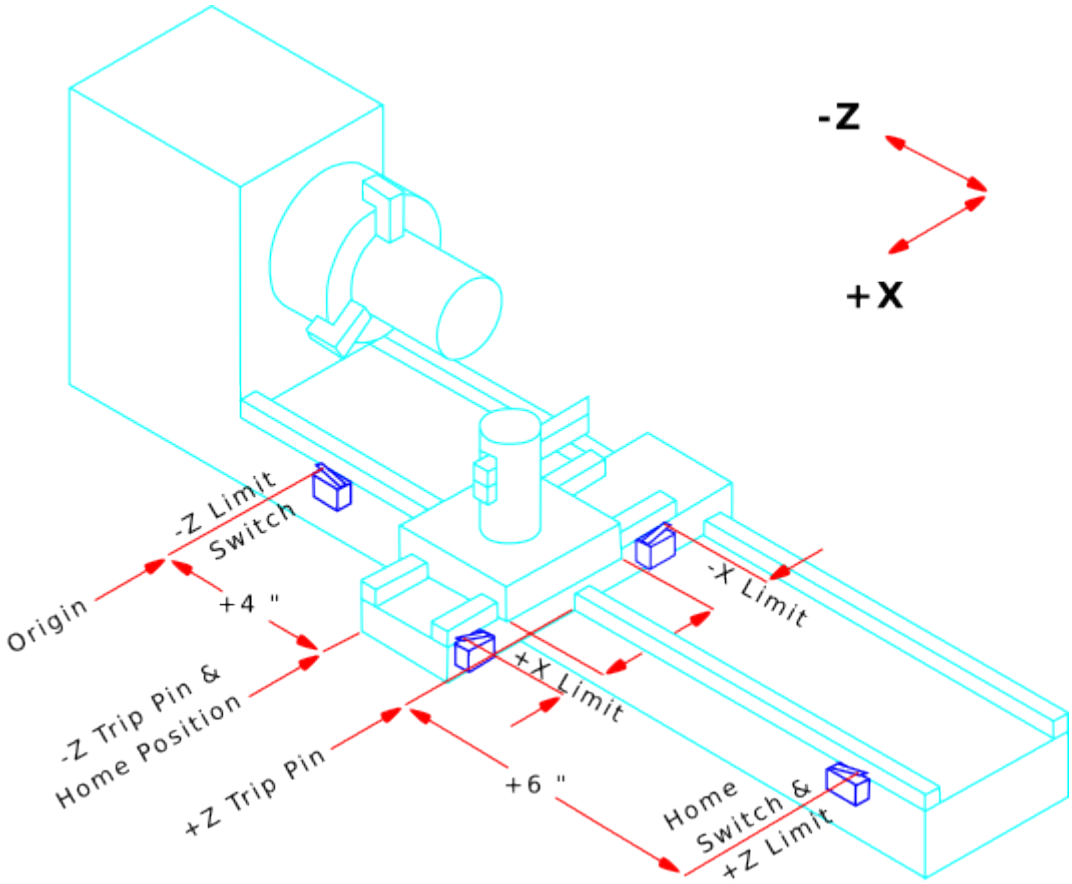


Figure 2.15: Типова конфігурація токарного верстата

2.4 Запуск LinuxCNC

2.4.1 Запуск LinuxCNC

LinuxCNC запускається за допомогою файлу скрипта *linuxcnc*.

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

Параметри скрипта linuxcnc

```
linuxcnc: b''Зb''b''ab''b''пb''b''yb''b''cb''b''кb'' LinuxCNC

b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
  $ linuxcnc -h
    This help

  $ linuxcnc [Options]
    b''Bb''b''иb''b''бb''b''eb''b''pb''b''ib''b''тb''b''ьb'' INI-b''фb''b''ab''b''йb''b' ↩
        'лb'' b''кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''yb''b''pb''b''ab''b''цb''b''ib''b ↩
        ''їb'' b''гb''b''pb''b''ab''b''фb''b''ib''b''чb''b''нb''b''ob''

  $ linuxcnc [Options] path/to/your_ini_file
    b''Hb''b''ab''b''зb''b''вb''b''ib''b''тb''b''ьb'' INI-b''фb''b''ab''b''йb''b''лb'' b' ↩
        'кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''yb''b''pb''b''ab''b''цb''b''ib''b''їb'', ↩
        b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''юb'' ↩
        b''чb''b''иb'' b''йb''b''ob''b''гb''b''ob'' b''шb''b''лb''b''яb''b''хb''
```

```
  $ linuxcnc [Options] -l
    b''Bb''b''ив''b''кb''b''ob''b''рb''b''ив''b''сb''b''тb''b''аb''b''йb''b''тb''b''еb'' b' ↩
        'рb''b''аb''b''нв''b''іb''b''шb''b''еb'' b''вв''b''ив''b''кb''b''ob''b''рb''b''ив''b ↩
        ''сb''b''тb''b''аb''b''нв''b''ив''b''йb'' INI-b''фb''b''аb''b''йb''b''лb'' b''кb''b' ↩
        'ob''b''нв''b''фb''b''іb''b''гb''b''уb''b''рb''b''аb''b''цb''b''іb''b''їb''

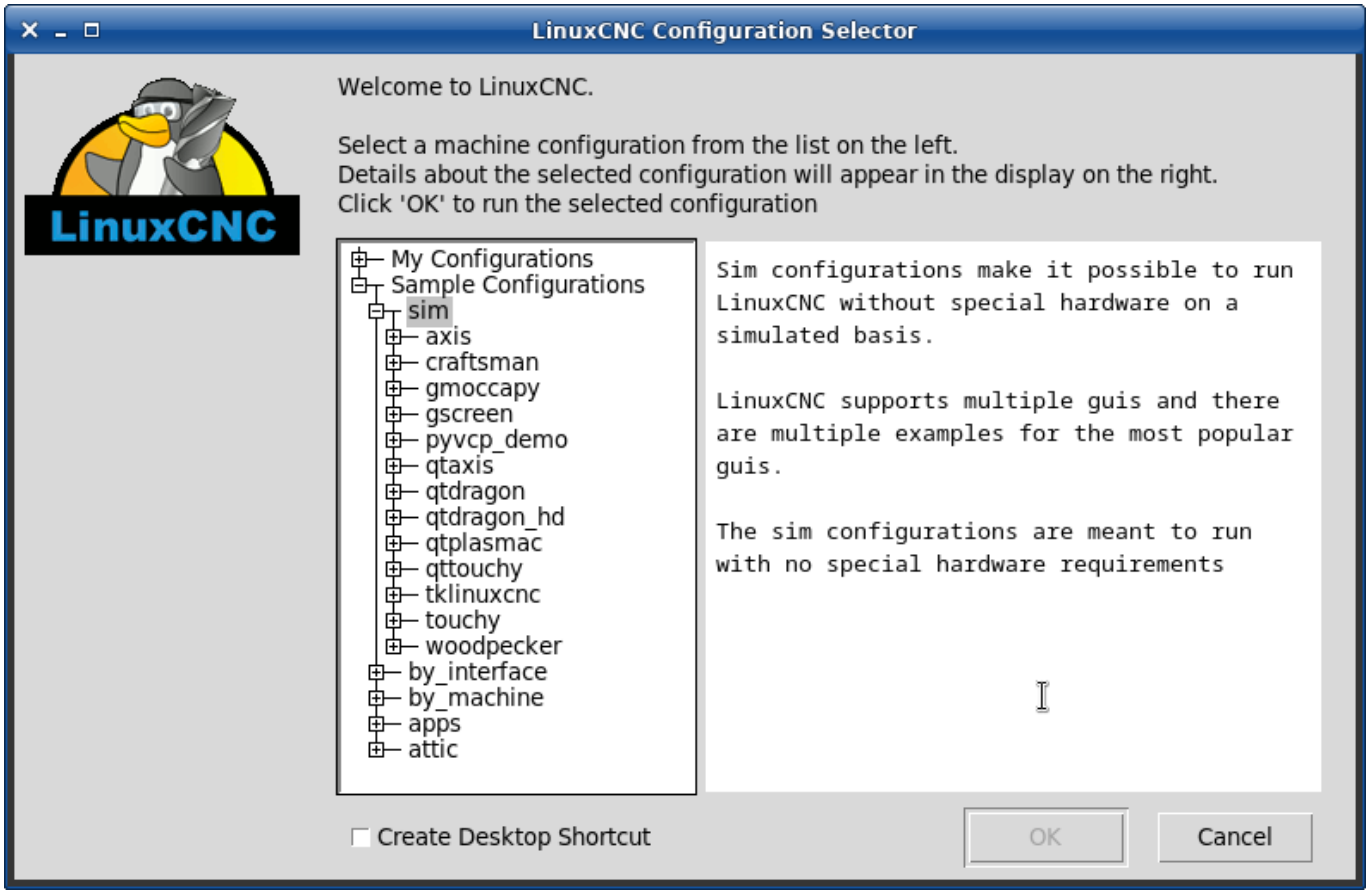
b''Ob''b''пb''b''цb''b''іb''b''їb'':
    -d: b''Уb''b''вb''b''іb''b''мb''b''кb''b''нв''b''уb''b''тb''b''ив'' b''рb''b''еb''b' ↩
        'жb''b''ив''b''мb'' «debug» (b''нв''b''аb''b''лb''b''аb''b''гb''b''ob''b''дb''b' ↩
        'жb''b''еb''b''нв''b''нв''b''яв'')
    -v: b''Уb''b''вb''b''іb''b''мb''b''кb''b''нв''b''уb''b''тb''b''ив'' b''рb''b''еb''b' ↩
        'жb''b''ив''b''мb'' «verbose» (b''дb''b''еb''b''тb''b''аb''b''лb''b''ьb''b''нв''b' ↩
        'ив''b''йb'')
    -r: b''Bb''b''ив''b''мb''b''кb''b''нв''b''уb''b''тb''b''ив'' b''пb''b''еb''b''рb''b' ↩
        'еb''b''нв''b''аb''b''пb''b''рb''b''аb''b''вв''b''лb''b''еb''b''нв''b''нв''b''яв''  ↩
        stdout b''ib'' stderr b''дb''b''ob'' ~/linuxcnc_print.txt b''ib''
        ~/linuxcnc_debug.txt, b''кb''b''ob''b''лb''b''ив'' stdin b''нв''b''еb'' b''єb'' tty ↩
            .
        b''Bb''b''ив''b''кb''b''ob''b''рb''b''ив''b''сb''b''тb''b''ob''b''вв''b''уb''b' ↩
            'єb''b''тb''b''ьb''b''сb''b''яв'' b''пb''b''рb''b''ив'' b''нв''b''еb''b''іb''b' ↩
            'нв''b''тb''b''еb''b''рb''b''аb''b''кb''b''тb''b''ив''b''вв''b''нв''b''ob''b' ↩
            'мb''b''уb'' b''зb''b''аb''b''пb''b''уb''b''сb''b''кb''b''уb'' b''тb''b''еb''b' ↩
            'сb''b''тb''b''іb''b''вв'' linuxcnc.
    -l: b''Bb''b''ив''b''кb''b''ob''b''рb''b''ив''b''сb''b''тb''b''ob''b''вв''b''уb''b' ↩
        'вв''b''аb''b''тb''b''ив'' b''ob''b''сb''b''тb''b''аb''b''нв''b''нв''b''іb''b''йb''  ↩
        b''вв''b''ив''b''кb''b''ob''b''рb''b''ив''b''сb''b''тb''b''аb''b''нв''b''ив''b''йb'' ↩
         b''фb''b''аb''b''йb''b''лb'' INI
    -k: b''Пb''b''рb''b''ob''b''дb''b''ob''b''вв''b''жb''b''уb''b''вв''b''аb''b''тb''b' ↩
        'ив'' b''пb''b''рb''b''ив'' b''нв''b''аb''b''яв''b''вв''b''нв''b''ob''b''сb''b''тb'' ↩
        b''іb'' b''пb''b''ob''b''мb''b''ив''b''лb''b''ob''b''кb'' b''уb'' b''фb''b''аb''b' ↩
        'йb''b''лb''b''аb''b''хb'' HAL
    -t "tpmodulename [parameters]"
            b''вb''b''кb''b''аb''b''зb''b''аb''b''тb''b''ив'' b''вb''b''лb''b''аb''b''сb''b ↩
                ''нв''b''ив''b''йb'' trajectory_planning_module
            b''пb''b''еb''b''рb''b''еb''b''кb''b''рb''b''ив''b''вb''b''аb''b''єb'' b''ob''b ↩
                ''пb''b''цb''b''іb''b''ob''b''нв''b''аb''b''лb''b''ьb''b''нв''b''еb'' b' ↩
                'нв''b''аb''b''лb''b''аb''b''шb''b''тb''b''уb''b''вb''b''аb''b''нв''b''нв''b ↩
                ''яв'' INI [TRAJ]TPMOD
    -m "homemodulename [parameters]"
            b''вb''b''кb''b''аb''b''зb''b''аb''b''тb''b''ив'' b''вb''b''лb''b''аb''b''сb''b ↩
                ''нв''b''ив''b''йb'' homing_module
            b''пb''b''еb''b''рb''b''еb''b''кb''b''рb''b''ив''b''вb''b''аb''b''єb'' b''ob''b ↩
                ''пb''b''цb''b''іb''b''ob''b''нв''b''аb''b''лb''b''ьb''b''нв''b''еb'' b' ↩
                'нв''b''аb''b''лb''b''аb''b''шb''b''тb''b''уb''b''вb''b''аb''b''нв''b''нв''b ↩
                ''яв'' INI [EMCMOT]HOMEMOD
    -H "dirname": b''шb''b''уb''b''кb''b''аb''b''тb''b''ив'' HAL-b''фb''b''аb''b''йb''b' ↩
        'лb''b''ив'' b''вb'' dirname b''пb''b''еb''b''рb''b''еb''b''дb'' b''пb''b''ob''b' ↩
        'шb''b''уb''b''кb''b''ob''b''мb''
                b''вb'' b''кb''b''аb''b''тb''b''аb''b''лb''b''ob''b''зb''b''іb'' INI b' ↩
                    'тb''b''аb'' b''сb''b''ив''b''сb''b''тb''b''еb''b''мb''b''нв''b''іb''b ↩
                    ''йb'' b''бb''b''іb''b''бb''b''лb''b''іb''b''ob''b''тb''b''еb''b''цb'' ↩
                    b''ib'':
                /home/git/linuxcnc-dev/lib/hallib
b''Пb''b''рb''b''ив''b''мb''b''іb''b''тb''b''кb''b''аb'':
    b''Ob''b''пb''b''цb''b''іb''b''юв'' -H "dirname" b''мb''b''ob''b''жb''b''нв''b''аb'' b' ↩
        'вb''b''кb''b''аb''b''зb''b''аb''b''тb''b''ив'' b''кb''b''іb''b''лb''b''ьb''b''кb''b ↩
        ''аb'' b''рb''b''аb''b''зb''b''іb''b''вв''
```

Якщо скрипту linuxcnc передається файл INI, він зчитує файл INI і запускає LinuxCNC. Розділ [HAL] файлу INI визначає порядок завантаження файлів HAL, якщо використовується більше

одного. Після завантаження файлів HAL=xxx.hal завантажується графічний інтерфейс, а потім файл POSTGUI=.xxx.hal. Якщо ви створюєте об'єкти PyVCP або GladeVCP з контактами HAL, ви повинні використовувати файл postgui HAL для встановлення будь-яких з'єднань з цими контактами. Дивіться розділ [HAL] конфігурації INI для отримання додаткової інформації.

2.4.1.1 Вибір конфігурації

Якщо файл INI не передається скрипту linuxcnc, він завантажує селектор конфігурації, щоб ви могли вибрати та зберегти зразок конфігурації. Після збереження зразка конфігурації його можна змінити відповідно до ваших потреб. Файли конфігурації зберігаються в каталозі linuxcnc/configs.



2.5 Огляд верстата з CNC

У цьому розділі наведено короткий опис того, як розглядається верстат з CNC з точки зору вхідного та вихідного елементів інтерпретатора.

2.5.1 Механічні компоненти

Верстат з CNC має багато механічних компонентів, які можуть контролюватися або впливати на спосіб здійснення контролю. У цьому розділі описано підмножину тих компонентів, які взаємодіють з інтерпретатором. Механічні компоненти, які не взаємодіють безпосередньо з інтерпретатором, такі як кнопки ручного переміщення, тут не описані, навіть якщо вони впливають на контроль.

2.5.1.1 Сокири

Будь-яка верстат з CNC має одну або кілька осей. Різні типи верстатів з CNC мають різні комбінації. Наприклад, «4-осьовий фрезерний верстат» може мати осі XYZA або XYZB. Токарний верстат зазвичай має осі XZ. Машина для різання пінопласту може мати осі XYUV. У LinuxCNC випадок «портальної» машини XYYZ з двома двигунами для однієї осі краще обробляється за допомогою кінематики, а не другої лінійної осі.

Note
Якщо рух механічних компонентів не є незалежним, як у випадку з шестиногими машинами, мова RS274/NGC і канонічні функції обробки все одно будуть придатними для використання, за умови, що нижчі рівні управління знають, як керувати фактичними механізмами для створення такого ж відносного руху інструменту і заготовки, який би створювався незалежними осями. Це називається «кінематикою».

Note
У LinuxCNC випадок портального верстата XYYZ з двома двигунами для однієї осі краще обробляється кінематикою, ніж додатковою лінійною віссю.

.Основні лінійні осі Осі X, Y та Z здійснюють лінійний рух у трьох взаємно ортогональних напрямках.

.Додаткові лінійні осі Осі U, V та W здійснюють лінійний рух у трьох взаємно ортогональних напрямках. Зазвичай X та U паралельні, Y та V паралельні, а Z та W паралельні.

.Осі обертання Осі A, B та C здійснюють кутовий рух (обертання). Як правило, A обертається навколо лінії, паралельної X, B обертається навколо лінії, паралельної Y, а C обертається навколо лінії, паралельної Z.

2.5.1.2 Шпиндель

Верстат з CNC зазвичай має шпиндель, який утримує один ріжучий інструмент, зонд або матеріал у випадку токарного верстата. Шпиндель може контролюватися програмним забезпеченням CNC або не контролюватися. LinuxCNC підтримує до 8 шпинделів, які можна контролювати окремо і які можуть працювати одночасно з різною швидкістю та в різних напрямках.

2.5.1.3 Охолоджувальна рідина

Охолоджувальний розчин розтоплення та охолоджувальний розпилювач можна вмикати незалежно. Мова RS274/NGC вимикає їх разом, див. розділ M7 M8 M9.

2.5.1.4 Коригування подачі та швидкості

Верстат з CNC може мати окремі елементи керування подачею та швидкістю, що дозволяє оператору вказати фактичну швидкість подачі або швидкість шпинделя, що використовується під час обробки, у певному відсотку від запрограмованої швидкості.

2.5.1.5 Перемикач видалення блоку

Верстат з CNC може мати перемикач видалення блоків. Див. розділ Видалення блоків.

2.5.1.6 Додатковий перемикач зупинки програми

Верстат з CNC може мати додатковий вимикач зупинки програми. Див. розділ Додаткова зупинка програми.

2.5.2 Компоненти керування та даних

2.5.2.1 Лінійні осі

Осі X, Y та Z утворюють стандартну правосторонню систему координат з ортогональних лінійних осей. Положення трьох механізмів лінійного руху виражаються за допомогою координат на цих осях.

Осі U, V та W також утворюють стандартну правосторонню систему координат. X та U паралельні, Y та V паралельні, а Z та W паралельні (коли A, B та C повернуті до нуля).

2.5.2.2 Осі обертання

Осі обертання вимірюються в градусах як обгорнуті лінійні осі, в яких напрямок позитивного обертання проти годинникової стрілки, якщо дивитися з позитивного кінця відповідної осі X, Y або Z. Під «обгорнутою лінійною віссю» ми маємо на увазі вісь, на якій кутове положення збільшується без обмежень (наближається до плюс нескінченності) при обертанні вісі проти годинникової стрілки і зменшується без обмежень (наближається до мінус нескінченності) при обертанні вісі за годинниковою стрілкою. Обгорнуті лінійні осі використовуються незалежно від того, чи існує механічне обмеження обертання.

За годинниковою стрілкою або проти годинникової стрілки — з точки зору заготовки. Якщо заготовка закріплена на поворотній платформі, яка обертається навколо осі обертання, обертання проти годинникової стрілки з точки зору заготовки здійснюється шляхом обертання поворотної платформи в напрямку, який (для більшості типових конфігурацій верстатів) виглядає за годинниковою стрілкою з точки зору людини, що стоїть поруч з верстатом. Примітка: [Якщо вимога паралельності порушена, розробник системи повинен вказати, як відрізнити обертання за годинниковою стрілкою від обертання проти годинникової стрілки.]

2.5.2.3 Контрольована точка

Контрольована точка — це точка, положення та швидкість руху якої контролюються. Коли зміщення довжини інструменту дорівнює нулю (значення за замовчуванням), це точка на осі шпинделя (часто називається контрольною точкою), яка знаходиться на певній фіксованій відстані за кінцем шпинделя, зазвичай поблизу кінця інструментального тримача, що вставляється в шпиндель. Розташування контрольованої точки можна перемістити вздовж осі шпинделя, вказавши деяке додатне значення зміщення довжини інструменту. Це значення зазвичай дорівнює довжині використовуваного різального інструменту, так що контрольована точка знаходиться на кінці різального інструменту. На токарному верстаті зміщення довжини інструменту можна вказати для осей X і Z, а контрольована точка знаходиться або на кінці інструменту, або трохи поза ним (де перетинаються перпендикулярні лінії, вирівняні по осі, до яких дотикаються «передня» і «бічна» частини інструменту).

2.5.2.4 Координований лінійний рух

Щоб переміщати інструмент по заданому шляху, обробний центр часто повинен координувати рух декількох осей. Термін «скоординований лінійний рух» використовується для опису ситуації, в якій, номінально, кожна вісь рухається з постійною швидкістю, а всі осі одночасно переміщуються

зі своїх початкових положень у кінцеві. Якщо рухаються тільки осі X, Y і Z (або одна чи дві з них), це створює рух по прямій лінії, звідси і походить слово «лінійний» у цьому терміні. У реальних рухах часто неможливо підтримувати постійну швидкість, оскільки на початку та/або в кінці руху необхідне прискорення або уповільнення. Однак можна контролювати осі таким чином, щоб кожна вісь завжди виконувала ту саму частину необхідного руху, що й інші осі. Це переміщує інструмент по тому самому шляху, і ми також називаємо такий рух скоординованим лінійним рухом.

Координоване лінійне переміщення може виконуватися або з переважною швидкістю подачі, або зі швидкістю поперечного переміщення, або може бути синхронізовано з обертанням шпинделя. Якщо фізичні обмеження швидкості осі не дозволяють досягти бажаної швидкості, всі осі сповільнюютьс щоб зберегти бажаний шлях.

2.5.2.5 Швидкість подачі

Швидкість переміщення контрольованої точки номінально є постійною і може бути встановлена користувачем. В інтерпретаторі швидкість подачі інтерпретується наступним чином (за винятком випадків, коли використовуються режими «інверсна швидкість подачі» або «швидкість подачі на оберт», про що див. розділ G93-G94-G95-Mode).

1. Якщо будь-яка з осей XYZ рухається, F вимірюється в одиницях за хвилину в декартовій системі XYZ, а всі інші осі (ABCUVW) рухаються так, щоб запускатися та зупинятися скоординован
2. В іншому випадку, якщо будь-яка з UVW рухається, F вимірюється в одиницях за хвилину в декартовій системі UVW, а всі інші осі (ABC) рухаються таким чином, щоб запускатися та зупинятися скоординовано.
3. В іншому випадку рух є чисто обертальним рухом, а слово F позначається в обертових одиницях у «псевдодекартовій» системі ABC.

2.5.2.6 Охолодження

Охолодження потоком або крапельним охолодженням можна вмикати окремо. Мова RS274/NGC зупиняє їх разом. Див. розділ про керування охолодженням.

2.5.2.7 Залишатися

Обробному центру може бути дано команду зупинитися (тобто утримати всі осі в нерухомому стані) на певний проміжок часу. Найчастіше зупинка використовується для розбивання та очищення стружки, тому під час зупинки шпиндель зазвичай обертається. Незалежно від режиму керування траєкторією (див. розділ Path Control), верстат зупиниться точно в кінці попереднього запрограмовано руху, як якщо б він працював у режимі точної траєкторії.

2.5.2.8 Одиниці

Одиниці виміру відстаней по осях X, Y і Z можуть бути виміряні в міліметрах або дюймах. Одиниці виміру всіх інших величин, що використовуються в управлінні верстатом, не можуть бути змінені. Для різних величин використовуються різні одиниці виміру. Швидкість обертання шпинделя вимірюється в обертах за хвилину. Положення осей обертання вимірюється в градусах. Швидкість подачі виражається в одиницях довжини за хвилину, градусах за хвилину або одиницях довжини за оберт шпинделя, як описано в розділі G93 G94 G95.

2.5.2.9 Поточна позиція

Контрольована точка завжди знаходиться в певному місці, яке називається «поточна позиція», і контролер завжди знає, де вона знаходиться. Числа, що позначають поточну позицію, повинні бути скориговані за відсутності будь-якого руху осі, якщо відбувається будь-яка з декількох подій:

1. Одиниці вимірювання довжини змінено.
2. Зміщення довжини інструменту змінено.
3. Зміщення системи координат змінено.

2.5.2.10 Вибрана площина

Завжди існує «вибрана площина», яка повинна бути площиною XY, площиною YZ або площиною XZ обробного центру. Вісь Z, звичайно, перпендикулярна площині XY, вісь X — площині YZ, а вісь Y — площині XZ.

2.5.2.11 Карусель інструментів

Кожному слоту в каруселі інструментів призначається один або нульовий інструмент.

2.5.2.12 Зміна інструменту

Обробному центру може бути наказано замінити інструменти.

2.5.2.13 Трансфер з піддонів

Два піддони можна обміняти за командою.

2.5.2.14 Перевизначення швидкості

Кнопки перевищення швидкості можуть бути активовані (вони функціонують нормально) або виведені з ладу (вони більше не мають ніякого ефекту). Мова RS274/NGC має команду, яка активує всі кнопки, і іншу, яка їх вимикає. Див. інгібування та активацію коректори швидкості. Див. також тут для отримання додаткової інформації.

2.5.2.15 Режим керування шляхом

Обробний центр може бути переведений в один з трьох режимів керування траєкторією:

режим точної зупинки
: У режимі точної зупинки машина короткочасно зупиняється в кінці кожного запрограмованого руху.

режим точного шляху
: У режимі точного шляху машина якомога точніше слідує запрограмованим шляхом, сповільнюючи рух або зупиняючись за необхідності на гострих кутах шляху.

безперервний режим
: У безперервному режимі гострі кути траєкторії можуть бути трохи заокруглені, щоб підтримувати швидкість подачі (але не більше ніж на допуск, якщо він вказаний).

Див. розділи G61 та G64.

2.5.3 Взаємодія перекладача з перемикачами

Інтерпретатор взаємодіє з кількома перемикачами. У цьому розділі взаємодії описано детальніше. Інтерпретатор у жодному разі не знає, яке налаштування має будь-який із цих перемикачів.

2.5.3.1 Перемикачі подачі та швидкості

Інтерпретатор буде інтерпретувати команди RS274/NGC, які вмикають «M48» або вимикають «M49» перемикачі подачі та швидкості. Для певних рухів, таких як поперечний рух з кінця різьби під час циклу нарізування різьби, перемикачі вимикаються автоматично.

LinuxCNC реагує на налаштування корекції швидкості та подачі, коли ці перемикачі ввімкнені.

Див. розділ M48 M49 Перевизначення для отримання додаткової інформації.

2.5.3.2 Перемикач видалення блоку

Якщо перемикач видалення блоку увімкнено, рядки G-коду, що починаються зі скісної риски (символ видалення блоку), не інтерпретуються. Якщо перемикач вимкнено, такі рядки інтерпретуютьс Зазвичай перемикач видалення блоку слід встановити перед запуском програми NGC.

2.5.3.3 Додатковий перемикач зупинки програми

Якщо цей перемикач увімкнено і виявлено код M1, виконання програми призупиняється.

2.5.4 Таблиця інструментів

Для використання інтерпретатора потрібна таблиця інструментів. Файл містить інформацію про те, які інструменти знаходяться в яких слотах пристрою зміни інструментів, а також про розмір і тип кожного інструменту. Назва таблиці інструментів визначається в INI:

```
[EMCIO]
# b''фb''b''ab''b''йb''b''лb'' b''тb''b''ab''b''бb''b''лb''b''иb''b''цb''b''іb'' b''ib''b' ↩
    'нb''b''cb''b''тb''b''pb''b''уb''b''мb''b''еb''b''нb''b''тb''b''ib''b''вb''
TOOL_TABLE = tooltable.tbl
```

Ім’я файлу за замовчуванням, ймовірно, виглядає приблизно так, як наведено вище, але ви можете надати вашому верстату власну таблицю інструментів, використовуючи ту саму назву, що й ваш INI-файл, але з розширенням tbl:

```
TOOL_TABLE = acme_300.tbl
```

або:

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

Для отримання додаткової інформації про особливості формату таблиці інструментів див. розділ Формат таблиці інструментів.

2.5.5 Параметри

У мові RS274/NGC обробний центр підтримує масив числових параметрів, визначених системним визначенням (RS274NGC_MAX_PARAMETERS). Багато з них мають конкретне застосування, особливо у визначенні систем координат. Кількість числових параметрів може збільшуватися у міру того, як розробка додає підтримку нових параметрів. Масив параметрів зберігається з часом, навіть якщо обробний центр вимкнений. LinuxCNC використовує файл параметрів для забезпечення стійкості і покладає на інтерпретатор відповідальність за підтримку файлу. Інтерпретатор читає файл при запуску і записує файл при виході.

Всі параметри доступні для використання в програмах G-коду.

Формат файлу параметрів наведено в наступній таблиці. Файл складається з будь-якої кількості рядків заголовка, за якими йде один порожній рядок, а потім будь-яка кількість рядків даних. Інтерпретатор пропускає рядки заголовка. Важливо, щоб перед даними був саме один порожній рядок (без пробілів і табуляцій). Рядок заголовка, показаний у наступній таблиці, описує стовпці даних, тому рекомендується (але не обов'язково) завжди включати цей рядок у заголовок.

Інтерпретатор читає лише перші два стовпці таблиці. Третій стовпець, «Коментар», інтерпретатор не читає.

Кожен рядок файлу містить індексний номер параметра в першому стовпці та значення, яке слід встановити для цього параметра, у другому стовпці. Значення представлено у вигляді числа з подвійною точністю з плаваючою комою всередині інтерпретатора, але десяткова крапка в файлі не потрібна. Усі параметри, наведені в наступній таблиці, є обов'язковими і повинні бути включені в будь-який файл параметрів, за винятком того, що будь-який параметр, що представляє значення осі обертання для невикористаної осі, може бути пропущений. Якщо будь-який обов'язковий параметр відсутній, буде видано повідомлення про помилку. Файл параметрів може містити будь-які інші параметри, якщо їхні номери знаходяться в діапазоні від 1 до 5400. Номери параметрів повинні бути впорядковані в порядку зростання. Якщо це не так, буде видано повідомлення про помилку. Будь-який параметр, включений у файл, що читається інтерпретатором, буде включений у файл, який він записує при виході. Оригінальний файл зберігається як резервна копія при записуванні нового файлу. Коментарі не зберігаються при записуванні файлу.

Table 2.1: Формат файлу параметрів

Номер параметра	Значення параметра	Коментар
5161	0.0	G28 Головна сторінка X
5162	0.0	G28 Головна сторінка Y

Див. розділ Parameters для отримання додаткової інформації.

2.6 Інформація для користувача токарного верстата

У цьому розділі буде надано інформацію, що стосується саме токарних верстатів.

2.6.1 Режим токарного верстата

Якщо ваш верстат з CNC є токарним, вам, ймовірно, потрібно буде внести деякі зміни до вашого INI-файлу, щоб отримати найкращі результати від LinuxCNC.

Якщо ви використовуєте дисплей AXIS, налаштуйте AXIS на відображення ваших токарних інструменті належним чином. Докладніше див. розділ INI Configuration.

Щоб налаштувати AXIS для режиму токарного верстата.

```
[DISPLAY]

# b''Пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''тb''b''eb'' b''гb''b''pb''b''ab''b' ↩
    'фb''b''ib''b''чb''b''нb''b''ob''b''мb''b''уb'' b''ib''b''нb''b''тb''b''eb''b''pb''b' ↩
    'фb''b''eb''b''йb''b''cb''b''уb'' AXIS, b''щb''b''ob'' b''нb''b''ab''b''шb'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''тb''b''ab''b''тb'' b''—b'' b''тb''b''ob''b''кb''b''ab''b''pb''b' ↩
    'нb''b''иb''b''йb''.
LATHE = TRUE
```

Режим токарного верстата в AXIS не встановлює площину за замовчуванням на G18 (XZ). Ви повинні запрограмувати це у преамбулі кожного файлу G-коду або (краще) додати це до вашого INI-файлу, ось так:

```
[RS274NGC]

# b''Mb''b''ob''b''дb''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''кb''b''ob''b''дb''b''иb'' (b' ↩
    'pb''b''eb''b''жb''b''иb''b''мb''b''иb'') G-b''кb''b''ob''b''дb''b''уb'', b''яb''b''кb'' ↩
    b''иb''b''мb''b''иb'' b''ib''b''нb''b''ib''b''цb''b''ib''b''ab''b''лb''b''ib''b''зb''b' ↩
    'уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b' ↩
    'pb''b''eb''b''тb''b''ab''b''тb''b''ob''b''pb''
# b''пb''b''ib''b''дb'' b''чb''b''ab''b''cb'' b''зb''b''ab''b''пb''b''уb''b''cb''b''кb''b' ↩
    'уb''
RS274NGC_STARTUP_CODE = G18 G20 G90
```

Якщо ви використовуєте GMOCCAPY, дивіться розділ розділ GMOCCAPY Lathe.

2.6.2 Стіл токарних інструментів

«Таблиця інструментів» — це текстовий файл, що містить інформацію про кожен інструмент. Файл знаходиться в тому ж каталозі, що й ваша конфігурація, і за замовчуванням має назву «tool.tbl». Інструменти можуть знаходитися в пристрої для зміни інструментів або бути змінені вручну. Файл можна редагувати за допомогою текстового редактора або оновлювати за допомогою G10 L1, L10, L11. У дисплеї AXIS також є вбудований редактор таблиці інструментів. Максимальна кількість записів у таблиці інструментів — 56. Максимальний номер інструменту та кишені — 99999.

Раніше версії LinuxCNC мали два різних формати таблиць інструментів для фрезерних верстатів і токарних верстатів, але починаючи з версії 2.4.x, для всіх верстатів використовується один формат таблиці інструментів. Просто ігноруйте ті частини таблиці інструментів, які не стосуються вашого верстата або які вам не потрібні. Більш детальну інформацію про особливості формату таблиці інструментів дивіться в розділі Tool Table.

2.6.3 Орієнтація токарного інструменту

На наступному рисунку показано орієнтації токарних інструментів з кутом центральної лінії кожної орієнтації та інформацією про ПЕРЕДНІЙ КУТ та ЗАДНІЙ КУТ.

ПЕРЕДНІЙ КУТ та ЗАДНІЙ КУТ розташовані за годинниковою стрілкою та починаються на лінії, паралельній Z+.

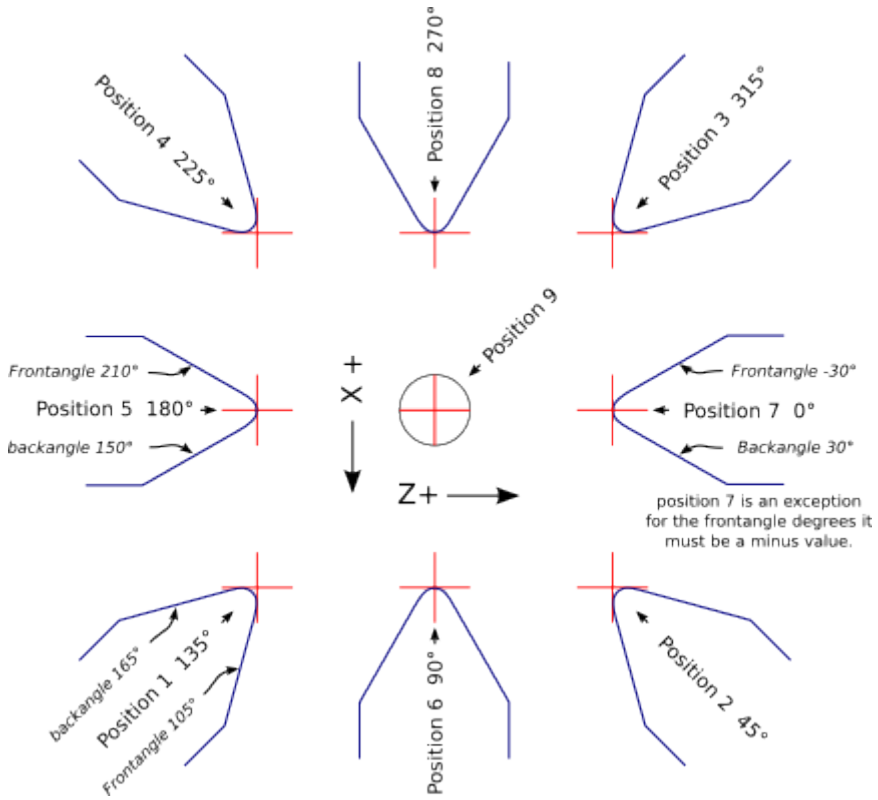
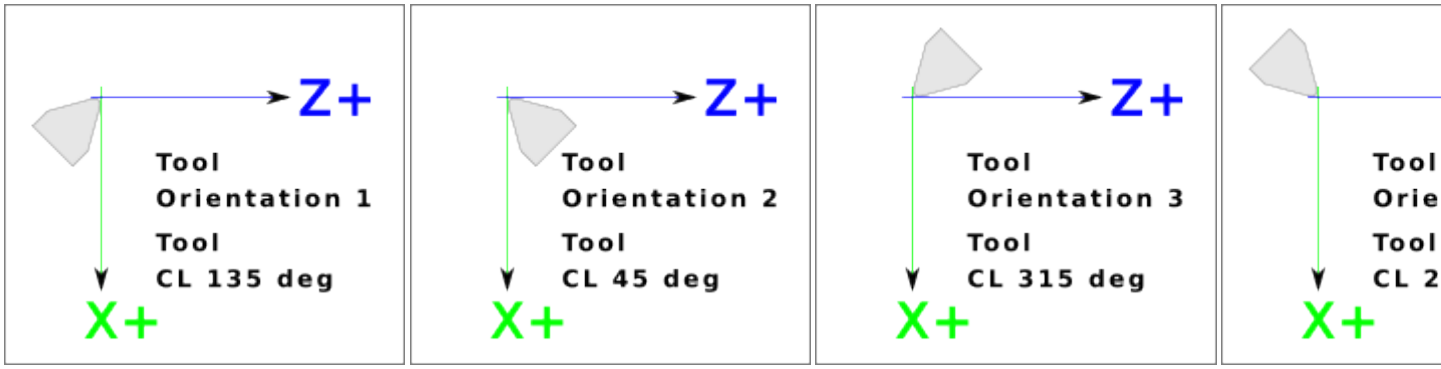


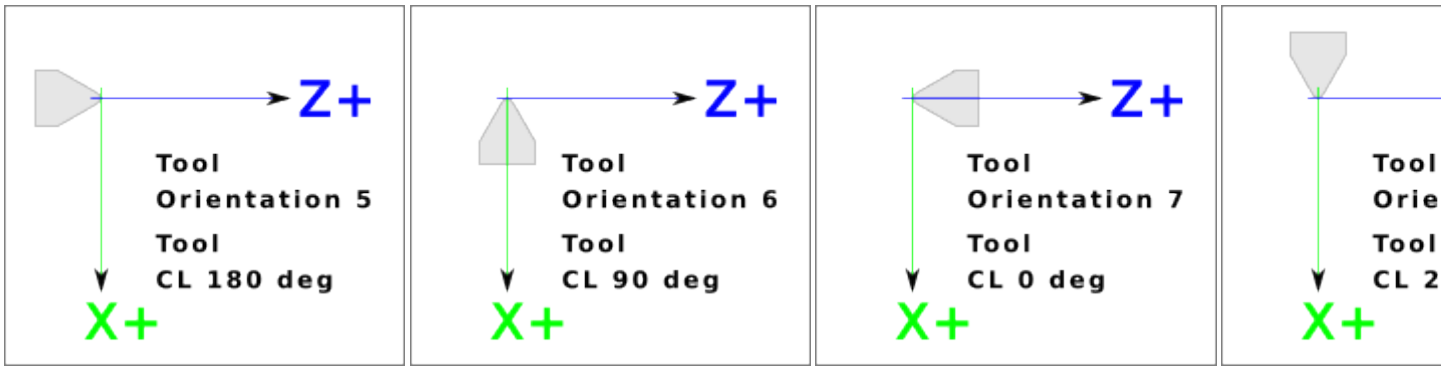
Figure 2.16: Орієнтації токарних інструментів

У AXIS на наступних рисунках показано, як виглядають позиції інструментів, введені в таблицю інструментів.

Tool Positions 1, 2, 3 & 4



Tool Positions 5, 6, 7 & 8



2.6.4 Інструмент дотику вимкнено

При роботі в режимі токарного верстата в AXIS ви можете встановити X і Z в таблиці інструментів за допомогою вікна Touch Off. Якщо у вас є інструментальна револьверна головка, зазвичай при налаштуванні револьверної головки вибирається опція «Touch off to fixture» (Відхилення від кріплення). При встановленні нуля Z матеріалу вибирається опція «Touch off to material» (Відхилення від матеріалу). Для отримання додаткової інформації про G-коди, що використовуються для інструментів, див. M6, Tn та G43. Для отримання додаткової інформації про параметри відліку інструменту в AXIS див. Tool Touch Off.

2.6.4.1 X Touch Off

Зміщення осі X для кожного інструменту зазвичай є зміщенням відносно центральної лінії шпинделя.

Один із способів полягає в тому, щоб взяти звичайний токарний інструмент і обточити заготовку до відомого діаметра. У вікні Tool Touch Off (Відключення інструменту) введіть виміряний діаметр (або радіус, якщо ви працюєте в режимі радіуса) для цього інструменту. Потім, використовуючи рідину для розмітки або маркер, покрийте деталь і підніміть кожен інструмент так, щоб він ледь торкався фарби, і встановіть його зміщення по осі X відповідно до діаметра деталі, використовуючи функцію відключення інструменту. Переконайтеся, що для всіх інструментів у кутових квадрантах радіус носа встановлений правильно в таблиці інструментів, щоб контрольна точка була правильною. Змащення інструменту автоматично додає G43, тому поточний інструмент є поточним зміщенням.

Типовий сеанс може бути таким:

1. Перевести кожну вісь у вихідне положення, якщо вона не переведена у вихідне положення.
2. Встановіть поточний інструмент за допомогою *T*n *M6 G43*, де *n* – номер інструмента.
3. Виберіть вісь X у вікні «Ручне керування».
4. Перемістіть X у відоме положення або зробіть пробний розріз і виміряйте діаметр.
5. Виберіть «Вимкнення дотику» та виберіть «Таблиця інструментів», а потім введіть положення або діаметр.
6. Виконайте ту саму послідовність для корекції осі Z.

Примітка: якщо ви перебуваєте в режимі радіуса, вам потрібно ввести радіус, а не діаметр.

2.6.4.2 Z Touch Off

Зсуви по осі Z спочатку можуть бути дещо заплутаними, оскільки зсув по осі Z складається з двох елементів. Існує зсув інструментального столу та зсув координат верстата. Спочатку розглянемо зсуви інструментального столу. Один із методів полягає у використанні фіксованої точки на токарному верстаті та встановленні зсуву по осі Z для всіх інструментів від цієї точки. Деякі використовують ніс шпинделя або торцеву поверхню патрона. Це дає вам можливість змінити інструмент і встановити його зміщення по осі Z без необхідності переналаштовувати всі інструменти.

Типовий сеанс може бути таким:

1. Перевести кожну вісь у вихідне положення, якщо вона не переведена у вихідне положення.
2. Переконайтеся, що для поточної системи координат не діють жодні зміщення.
3. Встановіть поточний інструмент за допомогою *T*n *M6 G43*, де *n* – номер інструмента.
4. Виберіть вісь Z у вікні ручного керування.
5. Піднесіть інструмент близько до керуючої поверхні.
6. За допомогою циліндра відсуньте літеру Z від керуючої поверхні, доки циліндр не пройде між інструментом та керуючою поверхнею.
7. Виберіть «Вимкнення дотику», потім «Таблиця інструментів» і встановіть положення на 0,0.
8. Повторіть для кожного інструменту, використовуючи той самий циліндр.

Тепер всі інструменти зміщені на однакову відстань від стандартного положення. Якщо ви змінюєте інструмент, наприклад свердло, повторіть вищевказані дії, і він буде синхронізований з іншими інструментами за зміщенням по осі Z. Деякі інструменти можуть вимагати невеликих обчислень для визначення контрольної точки від точки відліку. Наприклад, якщо у вас є інструмент для розрізання шириною 0,125 дюйма і ви торкаєтеся лівого боку, але хочете, щоб правий бік був Z0, введіть 0,125 дюйма у вікні відхилення.

2.6.4.3 Зсув машини Z

Після того, як у таблицю інструментів введено зміщення Z для всіх інструментів, можна використовува будь-який інструмент для встановлення зміщення верстата за допомогою системи координат верстата.

Типовий сеанс може бути таким:

1. Перевести кожну вісь у вихідне положення, якщо вона не переведена у вихідне положення.
2. Встановіть поточний інструмент за допомогою *T*n *M6*, де *n* – номер інструмента.
3. Видайте G43, щоб задіяти поточне зміщення інструменту.
4. Піднесіть інструмент до заготовки та встановіть зміщення верстата по осі Z.

Якщо ви забудете встановити G43 для поточного інструменту під час налаштування зміщення системи координат верстата, ви не отримаєте очікуваного результату, оскільки зміщення інструменту буде додано до поточного зміщення під час використання інструменту у вашій програмі.

2.6.5 Синхронізований рух шпинделя

Для синхронізованого руху шпинделя необхідний квадратурний енкодер, підключений до шпинделя з одним імпульсом індексації на оберт. Докладнішу інформацію див. на сторінці довідки motion та в розділі Приклад керування шпинделем.

Різьблення Цикл нарізання різьби G76 використовується як для внутрішньої, так і для зовнішньої різьби. Для отримання додаткової інформації див. розділ G76.

Постійна поверхнева швидкість CSS або Constant Surface Speed (постійна швидкість поверхні) використовує початок координат X верстата, змінений зміщенням інструменту X, для обчислення швидкості шпинделя в об/хв. CSS відстежує зміни зміщення інструменту. X machine origin повинен бути таким, коли еталонний інструмент (той, що має нульове зміщення) знаходиться в центрі обертання. Для отримання додаткової інформації див. розділ G96.

Подача на оберт Подача на оберт перемістить вісь Z на величину F на оберт. Це не для нарізання різьби, використовуйте G76 для нарізання різьби. Для отримання додаткової інформації див. розділ G95.

2.6.6 Дуги

Розрахунок дуг може бути досить складним завданням, не враховуючи режим радіуса і діаметра на токарних верстатах, а також орієнтацію системи координат верстата. Наступне стосується дуг центрального формату. На токарному верстаті ви повинні включити G18 у преамбулу, оскільки за замовчуванням використовується G17, навіть якщо ви перебуваєте в режимі токарного верстата, в інтерфейсі користувача AXIS. Дуги в площині G18 XZ використовують зміщення I (вісь X) і K (вісь Z).

2.6.6.1 Дуги та конструкція токарного верстата

Типовий токарний верстат має шпиндель зліва від оператора, а інструменти — з боку оператора від центральної лінії шпинделя. Зазвичай це налаштовується так, що уявна вісь Y (+) вказує на підлогу.

Для цього типу налаштування буде справедливим наступне:

- Вісь Z (+) спрямована праворуч, від шпинделя.
- Вісь X (+) спрямована до оператора, і коли шпиндель знаходиться з боку оператора, значення X додатні.

Деякі токарні верстати з інструментами на задній стороні мають уявну вісь Y (+), спрямовану вгору.

Напрямки дуги G2/G3 базуються на осі, навколо якої вони обертаються. У випадку токарних верстатів це уявна вісь Y. Якщо вісь Y (+) спрямована до підлоги, потрібно дивитися вгору, щоб дуга здавалася спрямованою у правильному напрямку. Отже, дивлячись зверху, потрібно змінити напрямок G2/G3, щоб дуга здавалася спрямованою у правильному напрямку.

2.6.6.2 Режим радіуса та діаметра

Під час розрахунку дуг у радіусному режимі вам потрібно пам’ятати лише напрямок обертання, який застосовується до вашого токарного верстата.

Під час обчислення дуг у режимі діаметра X – це діаметр, а зміщення X (I) – радіус, навіть якщо ви перебуваєте в режимі діаметра G7.

2.6.7 Шлях інструменту

2.6.7.1 Контрольна точка

Контрольна точка інструменту слідує запрограмованому шляху. Контрольна точка є перетином лінії, паралельної осям X і Z, і дотичної до діаметра кінчика інструменту, як визначено при торканні осей X і Z для цього інструменту. При точінні або торцюванні деталей з прямими сторонами шлях різання і край інструменту слідують одному і тому ж шляху. При точінні радіусів і кутів край кінчика інструменту не буде слідувати запрограмованому шляху, якщо не ввімкнено компенсацію різака. На наступних малюнках ви можете побачити, як контрольна точка не слідує за краєм інструменту, як ви могли б припустити.

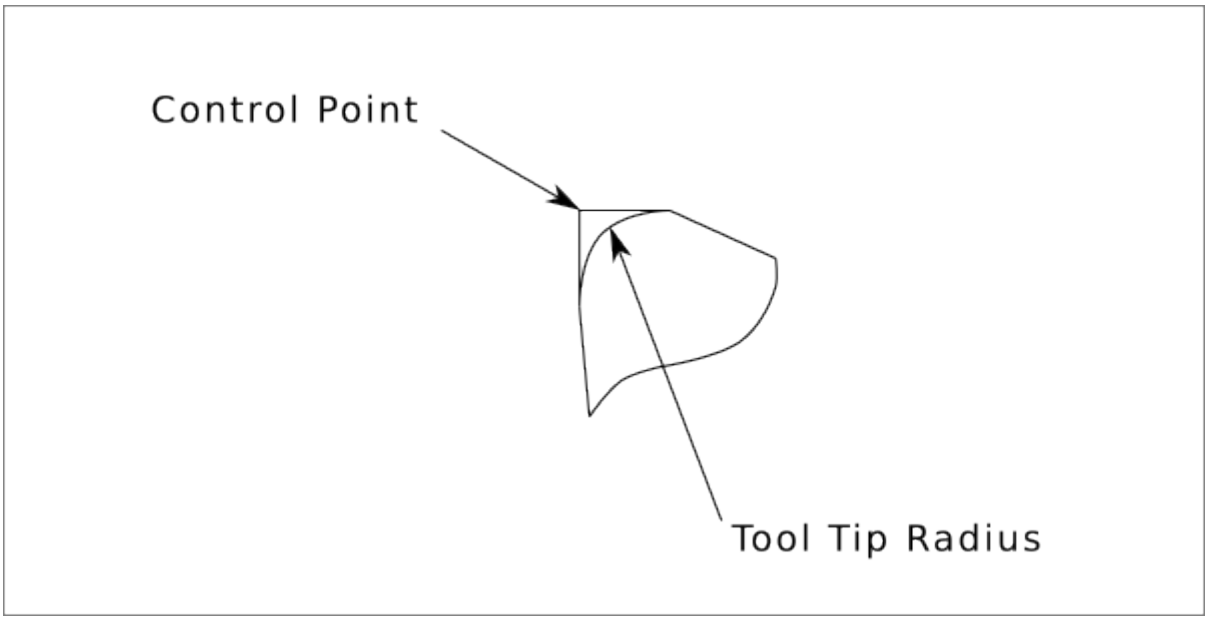


Figure 2.17: Контрольна точка

2.6.7.2 Кути різання без різального компаса

Тепер уявіть, що ми програмуємо рампу без компенсації різака. Запрограмований шлях показано на наступному малюнку. Як ви можете бачити на малюнку, запрограмований шлях і бажаний шлях різання є однаковими, якщо ми рухаємося тільки в напрямку X або Z.

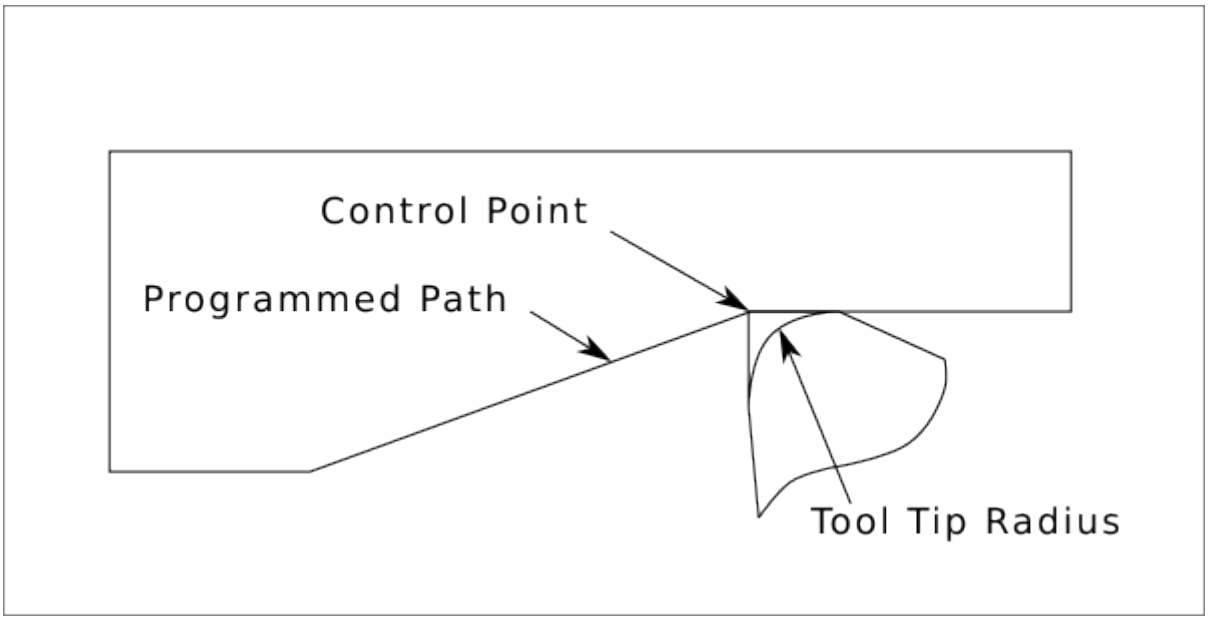


Figure 2.18: B’їзд на пандус

Тепер, коли контрольна точка просувається по запрограмованому шляху, фактична кромка різака не слідує запрограмованому шляху, як показано на наступному малюнку. Існує два способи вирішення цієї проблеми: компенсація різака та коригування запрограмованого шляху для компенсації радіуса кінчика.

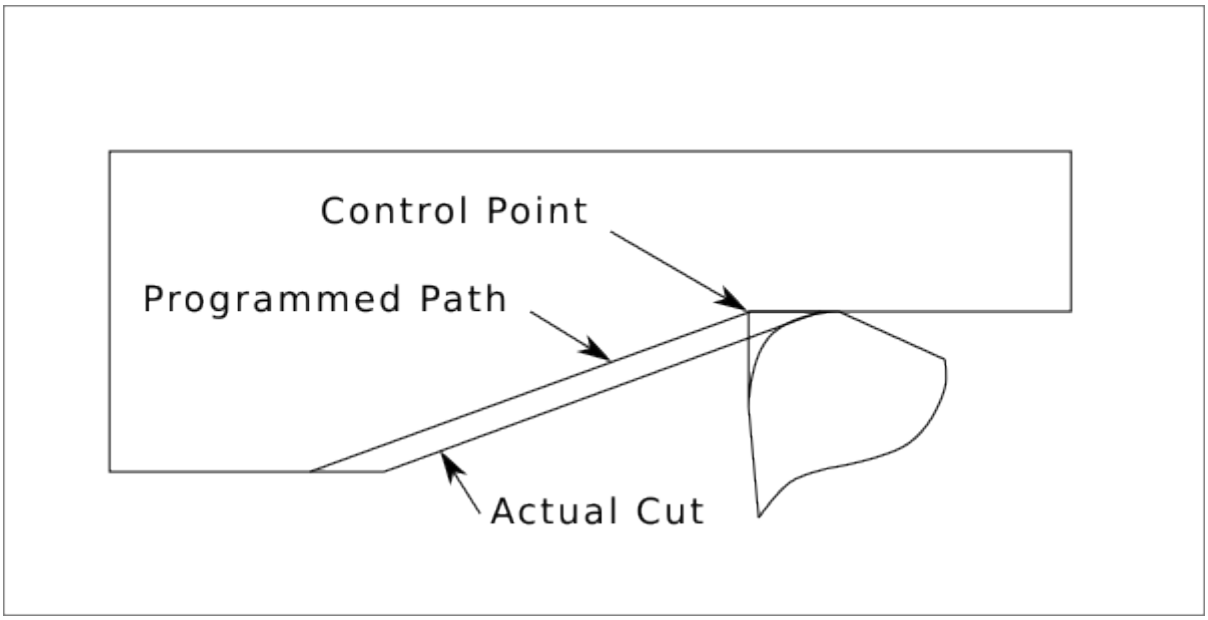


Figure 2.19: Шлях рампи

У наведеному вище прикладі це проста вправа для коригування запрограмованого шляху для отримання бажаного фактичного шляху шляхом переміщення запрограмованого шляху для рампи ліворуч від радіуса кінчика інструменту.

2.6.7.3 Вирізання радіуса

У цьому прикладі ми розглянемо, що відбувається під час радіального різання без компенсації різака. На наступному малюнку ви бачите, як інструмент обточує зовнішній діаметр деталі. Контрольна точка інструменту слідує запрограмованій траєкторії, а інструмент торкається зовнішньог діаметра деталі.

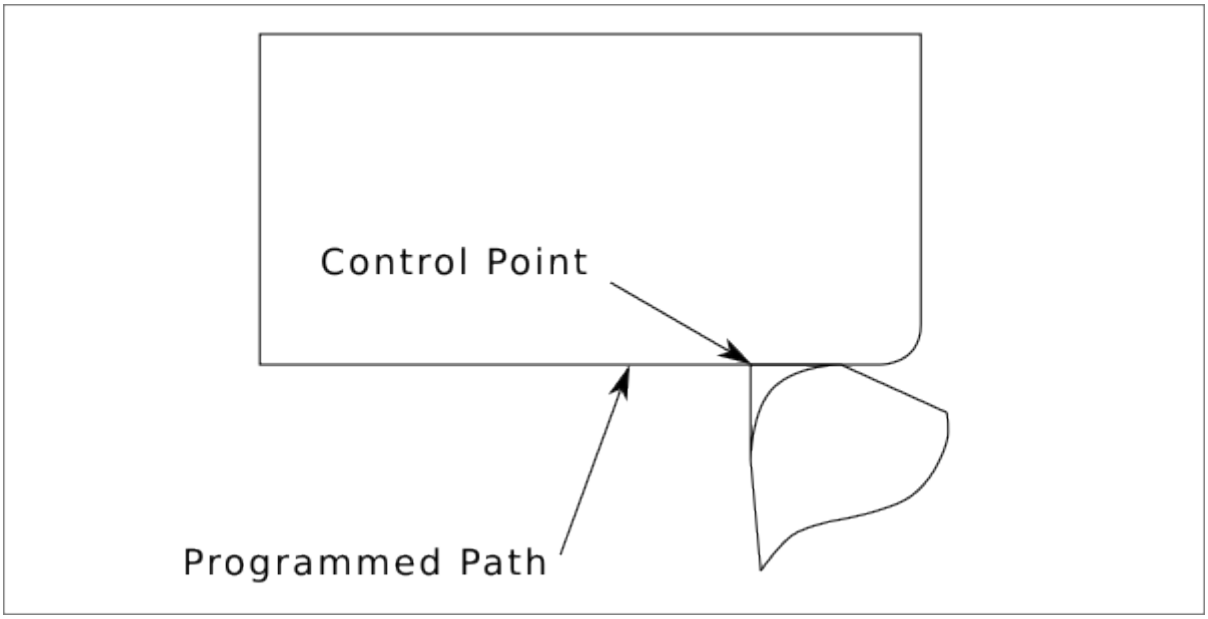


Figure 2.20: Токарний розріз

На наступному малюнку ви можете побачити, що коли інструмент наближається до кінця деталі, контрольна точка все ще слідує траєкторії, але кінчик інструменту відійшов від деталі і ріже повітря. Ви також можете побачити, що, незважаючи на те, що був запрограмований радіус, деталь фактично отримає квадратний кут.

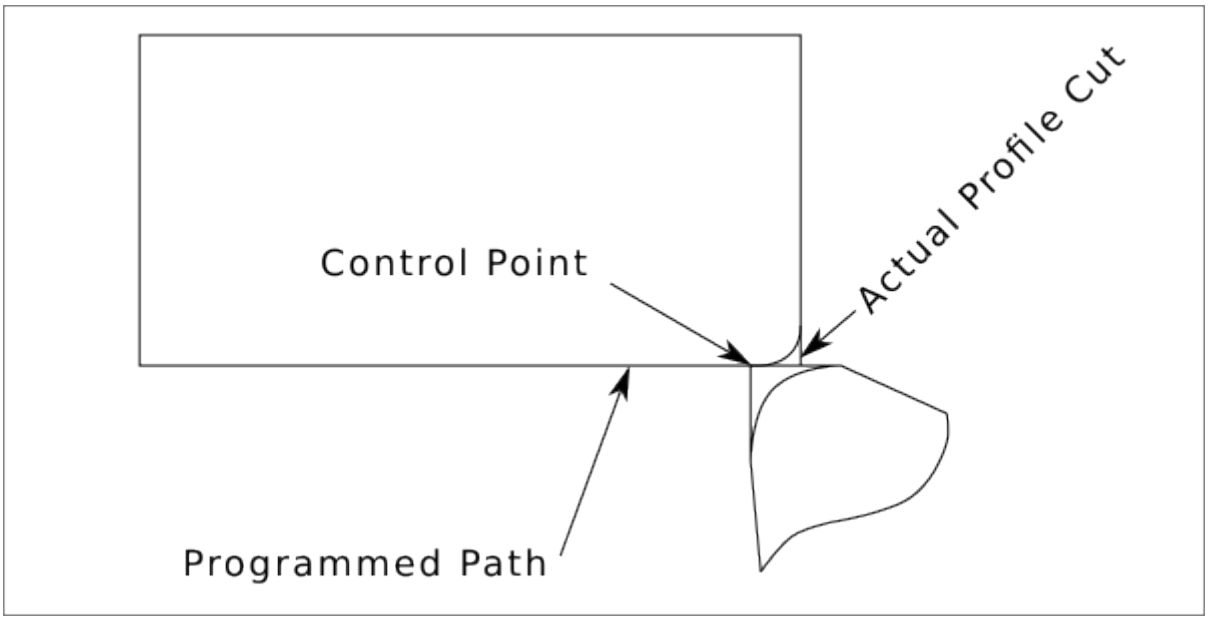


Figure 2.21: Радіусний розріз

Тепер ви можете бачити, що оскільки контрольна точка слідує запрограмованому радіусу, кінчик інструменту покинув деталь і тепер ріже повітря.

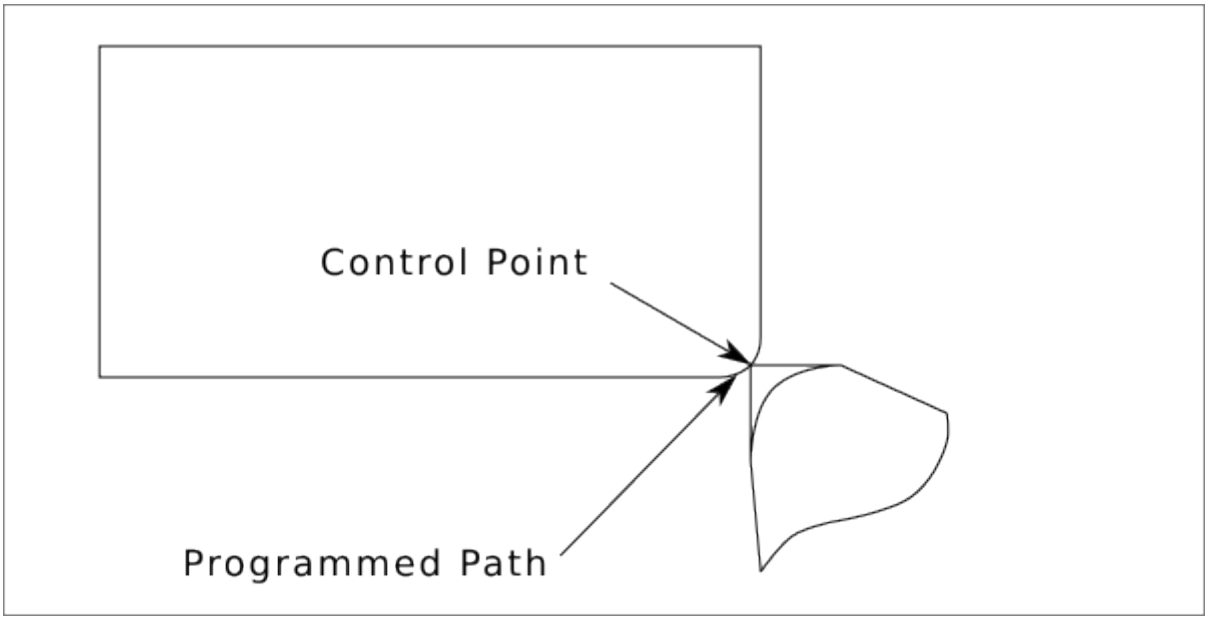


Figure 2.22: Радіусний розріз

На останньому малюнку ми бачимо, що інструмент закінчить різання поверхні, але залишить квадратний кут замість рівного радіуса. Зверніть також увагу, що якщо ви запрограмуєте закінчення різання в центрі деталі, то від радіуса інструменту залишиться невелика кількість матеріалу. Щоб закінчити різання поверхні в центрі деталі, ви повинні запрограмувати інструмент так, щоб він пройшов за центр принаймні на радіус носа інструменту.

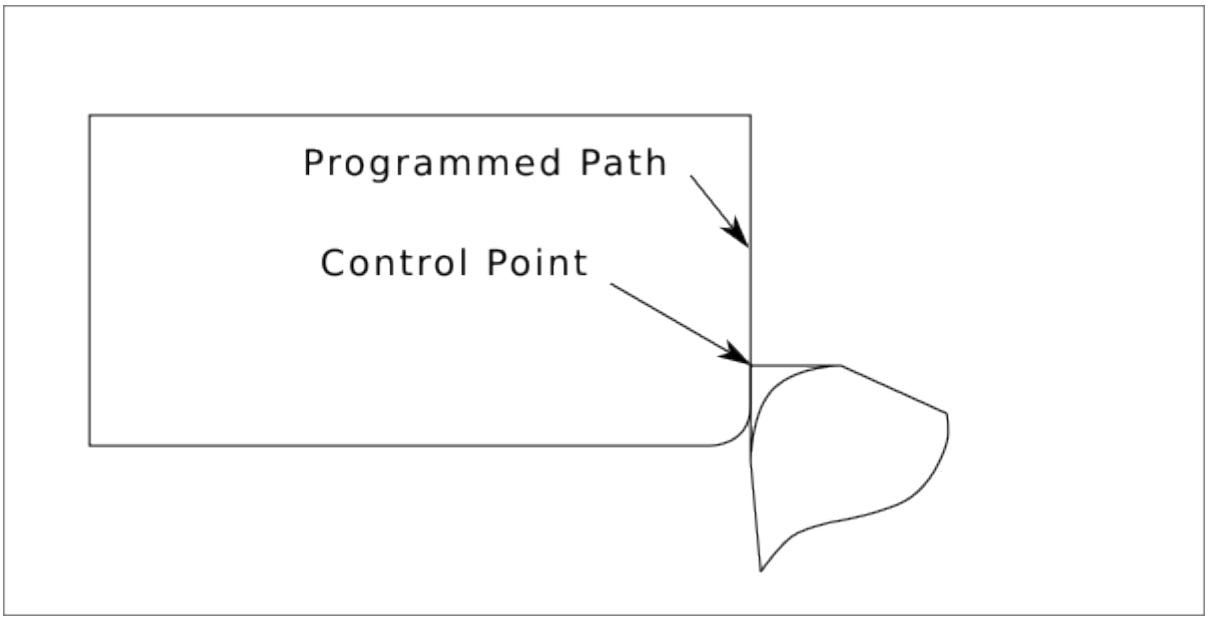


Figure 2.23: Виріз обличчя

2.6.7.4 Використання компенсації різака

- Під час використання компенсатора різця на токарному верстаті вважайте радіус вершини різця радіусом круглого різця.
- Під час використання компенсації різця шлях має бути достатньо великим для круглого інструменту, який не заглиблюється в наступну лінію.
- Під час різання прямих ліній на токарному верстаті може не знадобитися використовувати компенсатор різця. Наприклад, під час свердління отвору щільно прилягаючою розточувальною планкою у вас може не бути достатньо місця для виконання виходу.
- Вхідний рух у дугу компенсації різця важливий для отримання правильних результатів.

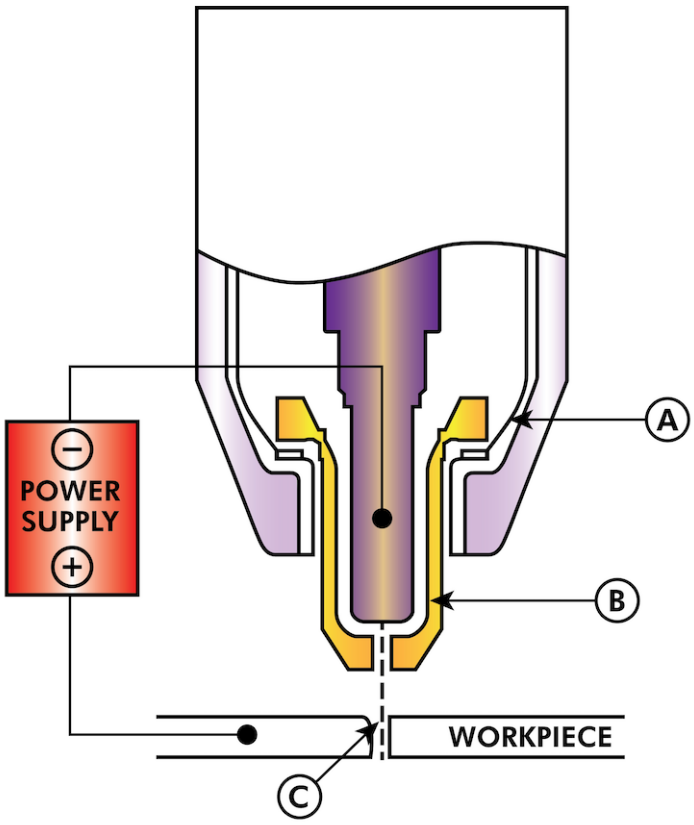
2.7 Грунтовка для плазмового різання для користувачів LinuxCNC

2.7.1 Що таке плазма?

Плазма — це четвертий стан речовини, іонізований газ, який нагрітий до надзвичайно високої температури та іонізований, завдяки чому стає електропровідним. У процесах плазмового різання та фрезерування плазмою використовується ця плазма для передачі електричної дуги на заготовку. Метал, який потрібно розрізати або видалити, плавиться під дією тепла дуги, а потім видувається. Мета плазмового різання — розділення матеріалу, а плазмове фрезерування використовується для видалення металів на контрольовану глибину та ширину.

Плазмові пальники за своєю конструкцією схожі на автомобільні свічки запалювання. Вони складаються з негативної та позитивної частин, розділених центральним ізолятором. Усередині пальника пілотна дуга запускається в зазорі між негативно зарядженим електродом та позитивно зарядженим наконечником. Після того, як пілотна дуга іонізувала плазмовий газ, перегріта колона газу протікає через невеликий отвір у наконечнику пальника, який фокусується на металі, що підлягає різанню.

У плазмовому різальному пальнику холодний газ надходить у зону B, де пілотна дуга між електродом і наконечником пальника нагріває та іонізує газ. Потім основна різальна дуга передається на заготовку через стовп плазмового газу в зоні C. Пропускаючи плазмовий газ і електричну дугу через невеликий отвір, пальник подає високу концентрацію тепла на невелику площу. Жорстка, звужена плазмова дуга показана в зоні C. Для плазмового різання використовується постійний струм (DC) прямої полярності, як показано на ілюстрації. Зона A направляє вторинний газ, який охолоджує пальник. Цей газ також допомагає високошвидкісному плазмовому газу видувати розплавлений метал з різу, забезпечуючи швидке різання без шлаку.



TYPICAL TORCH HEAD DETAIL

2.7.2 Ініціалізація дуги

Існує два основних методи ініціалізації дуги для плазмових різаків, призначених для роботи з CNC. Хоча на деяких машинах використовуються й інші методи (наприклад, запуск зі шкрябанням, який вимагає фізичного контакту з матеріалом), вони не підходять для застосування з CNC.

2.7.2.1 Високочастотний пуск

Цей тип запуску широко використовується і існує найдовше. Хоча це застаріла технологія, вона працює добре і швидко запускається. Однак через високу частоту та високу напругу, необхідні для іонізації повітря, вона має деякі недоліки. Вона часто створює перешкоди для навколишніх електронних схем і може навіть пошкодити компоненти. Також для створення пілотної дуги потрібна спеціальна схема. Недорогі моделі не мають пілотної дуги і для запуску вимагають дотику витратного матеріалу до деталі. Використання високочастотної схеми також

може збільшити проблеми з технічним обслуговуванням, оскільки зазвичай є регульовані точки, які необхідно час від часу чистити та переналаштовувати.

2.7.2.2 Початок віддачі

Цей тип пуску використовує тиск повітря, що подається до різака, щоб змусити невеликий поршень або картридж всередині головки пальника повернутися назад, створюючи невелику іскру між внутрішньою поверхнею витратного матеріалу, іонізуючи повітря і створюючи невелике плазмове полум'я. Це також створює «пілотну дугу», яка забезпечує плазмове полум'я, що залишається увімкненим, незалежно від того, чи контактує воно з металом. Це дуже хороший тип запалювання, який зараз використовується декількома виробниками. Його перевага полягає в тому, що він вимагає дещо менше схем, є досить надійним і створює набагато менше електричних перешкод.

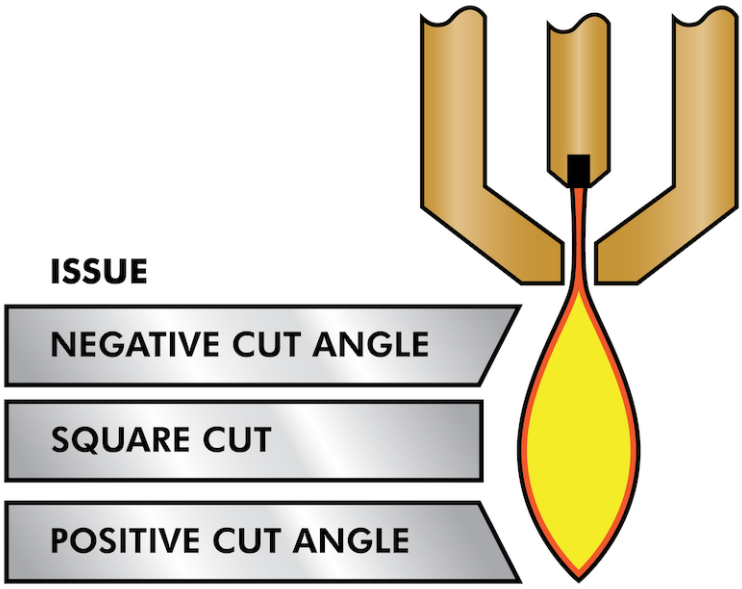
Для початкового рівня систем CNC з повітряною плазмою набагато кращим є тип з відкатом, щоб мінімізувати електричні перешкоди в електроніці та стандартних ПК, але високочастотний пуск все ще залишається домінуючим у більших машинах від 200 А і вище. Вони вимагають ПК та електроніки промислового рівня, і навіть комерційні виробники стикалися з проблемами несправностей, оскільки не врахували електричні перешкоди у своїх конструкціях.

2.7.3 Плазмовий CNC

Плазмові операції на верстатах з CNC є досить унікальними в порівнянні з фрезеруванням або точінням і є дещо нетиповим процесом. Нерівномірний нагрів матеріалу плазмовою дугою призведе до вигину та деформації листа. Більшість металевих листів не виходять з прокатного стану або преса в дуже рівному або плоскому стані. Товсті листи (30 мм і більше) можуть виходити за межі площини на 50–100 мм. Більшість інших операцій з використанням G-коду на верстатах з CNC починаються з відомої базової точки або заготовки, розмір і форма якої відомі, і G-код пишеться для зняття надлишку, а потім остаточного вирізання готової деталі. При використанні плазми невідомий стан листа унеможливлює створення G-коду, який враховуватиме ці відмінності в матеріалі.

Плазмова дуга має овальну форму, тому висоту різання необхідно контролювати, щоб мінімізувати скошені краї. Якщо пальник розташований занадто високо або занадто низько, краї можуть стати надмірно скошеними. Також дуже важливо, щоб пальник тримали перпендикулярно до поверхні.

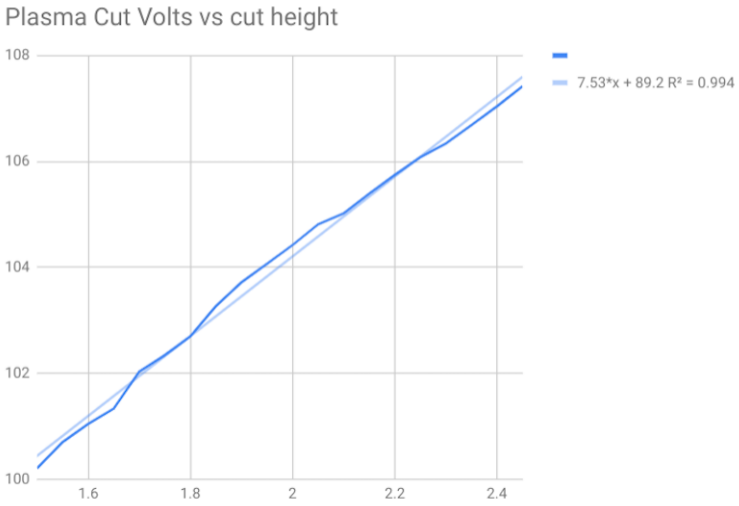
- **Відстань від пальника до робочого місця може впливати на фаску краю**



- **Негативний кут різання:** пальник розташований занадто низько, збільште відстань від пальника до робочої поверхні.
- **Позитивний кут різання:** пальник розташований занадто високо, зменште відстань між пальником та робочою поверхнею.

Note
Невелике відхилення кутів різу може бути нормальним, якщо воно знаходиться в межах допуску.

Здатність точно контролювати висоту різання в таких несприятливих і постійно мінливих умовах є дуже складним завданням. На щастя, між висотою пальника (довжиною дуги) і напругою дуги існує дуже лінійна залежність, як показано на цьому графіку.



Цей графік був складений на основі вибірки з приблизно 16 000 показань при різній висоті зрізу, а регресійний аналіз показує 7,53 В/мм з довірчою ймовірністю 99,4%. У цьому конкретному випадку вибірка була взята з машини Everlast 50 A, що керується LinuxCNC.

Напруга пальника стає ідеальною змінною для регулювання процесу, яку можна використовувати для регулювання висоти різання. Для простоти припустимо, що напруга змінюється на 10 В/мм. Це можна перерахувати в 1 В на 0,1 мм (0,004 дюйма). Основні виробники плазмових верстатів (наприклад, Hypertherm, Thermal Dynamics та ESAB) створюють таблиці різання, в яких вказано рекомендовану висоту різання та розрахункову напругу дуги на цій висоті, а також деякі додаткові дані. Отже, якщо напруга дуги на 1 В вища, ніж зазначено у специфікації виробника, контролер просто повинен опустити пальник на 0,1 мм (0,004 дюйма), щоб повернутися до бажаної висоти різання. Для управління цим процесом традиційно використовується блок регулювання висоти пальника (THC).

2.7.4 Вибір плазмового верстата для операцій з CNC

Сьогодні на ринку представлено безліч плазмових верстатів, і не всі з них підходять для використання з CNC. Плазмове різання з CNC є складною операцією, тому інтеграторам рекомендується вибирати відповідний плазмовий верстат. Недотримання цієї рекомендації може призвести до багатогодинних марних спроб усунення несправностей, пов’язаних з відсутністю функцій, які багато хто вважає обов’язковими.

Хоча правила створені для того, щоб їх порушувати, якщо ви повністю розумієте причини їх застосування, ми вважаємо, що новий конструктор плазмових столів повинен вибрати машину з такими характеристиками:

- Пуск з продувкою для мінімізації електричного шуму для спрощення конструкції
- Перевага надається машинному пальнику, але багато хто використовував ручні пальники.
- Повністю екранований наконечник пальника для забезпечення омічного зондування

Якщо у вас є бюджет, машини вищого класу забезпечать:

- Виробник надав таблиці різання, які заощадять багато годин та зменшать втрати матеріалу під час калібрування параметрів різання
- Сухі контакти для ArcOK
- Клеми для вимикача дуги
- Вихідна необроблена напруга дуги або розділена напруга дуги
- За бажанням, інтерфейс RS485, якщо використовується плазмовий різак Hypertherm і потрібно керувати ним з консолі LinuxCNC.
- Вищі робочі цикли

Останнім часом з’явився інший тип апаратів, що мають деякі з цих функцій, вартістю близько 550 доларів США. Одним із прикладів є Herocut55i, який можна придбати на Amazon, але поки що немає відгуків від користувачів. Цей апарат має пальник з відбійним механізмом, вихід ArcOK, контакти для запуску пальника та вихідну напругу дуги.

2.7.5 Типи контролю висоти пальника

Більшість пристроїв THC є зовнішніми пристроями, і багато з них мають досить грубий метод регулювання «біт-банг». Вони подають два сигнали назад до контролера LinuxCNC. Один вмикається, якщо вісь Z повинна рухатися вгору, а інший вмикається, якщо вісь Z повинна рухатися вниз. Жоден із сигналів не є істинним, якщо пальник знаходиться на правильній висоті. Популярний Proma 150 THC є одним із прикладів цього типу THC. Компонент LinuxCNC THCUD призначений для роботи з цим типом THC.

З випуском інтерфейсу напруги до частоти Mesa THCAD, LinuxCNC отримав можливість декодувати фактичну напругу пальника через вхід кодера. Це дозволило LinuxCNC контролювати вісь Z і усунути необхідність у зовнішньому обладнанні. Ранні реалізації, що використовували THCAD, повторювали підхід «біт-банг». Компонент LinuxCNC THC є прикладом такого підходу.

Джим Кольт з компанії Hypertherm офіційно заявив, що найкращі контролери THC були повністю інтегровані в сам контролер CNC. Звичайно, він мав на увазі високотехнологічні системи, виготовлені компаніями Hypertherm, Esab, Thermal Dynamics та іншими, такими як Advanced Robotic Technology в Австралії, і навіть не мріяв, що відкрите програмне забезпечення може створити системи, які за допомогою цього підходу зможуть конкурувати з високотехнологічними системами.

Включення зовнішніх зміщень в LinuxCNC V2.8 дозволило вивести управління плазмою в LinuxCNC на абсолютно новий рівень. Зовнішні зміщення - це можливість застосовувати зміщення до заданої позиції осі поза контролером руху. Це ідеально підходить для управління плазмою THC як метод регулювання висоти пальника в режимі реального часу на основі обраної нами методології управління процесом. Після низки експериментальних збірок конфігурація Plasmac була включена в LinuxCNC 2.8. Посилання: ./qtplasmac.html[QtPlasmaC] замінила Plasmac в LinuxCNC 2.9. Це був надзвичайно амбітний проект, і багато людей по всьому світу брали участь у тестуванні та вдосконаленні набору функцій. QtPlasmaC є унікальним тим, що його метою було підтримати всі THC, включаючи прості бітові, аж до складного контролю напруги пальника, якщо напруга доступна для LinuxCNC через THCAD або інший датчик напруги. Більше того, QtPlasmaC розроблений як автономна система, яка не потребує додаткових підпрограм G-коду і дозволяє користувачеві визначати власні таблиці різання, які зберігаються в системі і доступні через випадаюче меню.

2.7.6 Сигнал «Дуга в порядку»

Плазмові машини, що мають інтерфейс CNC, містять набір сухих контактів (наприклад, реле), які замикаються при встановленні дійсної дуги, і кожна сторона цих контактів виводиться на контакти інтерфейсу CNC. Виробник плазмового столу повинен підключити одну сторону цих контактів до джерела живлення, а іншу — до вхідного контакту. Це дозволяє контролеру CNC визначати, коли встановлюється дійсна дуга, а також коли дуга несподівано зникає. Тут існує потенційна пастка, коли вхід є високоомним контуром, таким як карта Mesa. Якщо сухі контакти є простим реле, існує висока ймовірність, що струм, який проходить через реле, буде меншим за мінімальну специфікацію струму. За таких умов контакти реле можуть страждати від накопичення оксиду, що з часом може призвести до переривчастої роботи контактів. Щоб запобігти цьому, на вхідному контакті контролера слід встановити резистор. Слід подбати про те, щоб цей резистор був підібраний таким чином, щоб через реле проходив мінімальний струм і він мав достатню потужність для роботи в ланцюзі. Нарешті, резистор слід встановити таким чином, щоб тепло, що виділяється, не пошкоджувало нічого під час роботи.

Якщо у вас є сигнал ArcOK, рекомендується використовувати його замість будь-якого синтезованого сигналу, щоб уникнути можливих проблем з побудовою. Синтезований сигнал, доступний із зовнішнього THC або режиму 0 QtPlasmaC, не може повністю замінити схему ArcOK у плазмовому інверторі. Були виявлені деякі проблеми з побудовою, коли через неправильну конфігурацію або несумісність із плазмовим інвертором виникав синтезований сигнал ArcOK. Однак, загалом, правильно налаштований синтезований сигнал ArcOK працює нормально.

Простий і ефективний сигнал ArcOK можна отримати за допомогою простого герконового реле. Оберніть навколо нього 3 витки одного з товстих кабелів плазмового різака, наприклад, кабелю затискача матеріалу. Помістіть реле в стару трубку від ручки для захисту і підключіть один бік реле до джерела живлення, а інший кінець — до вхідного контакту ArcOK.

2.7.7 Початкове вимірювання висоти

Оскільки висота різання є таким важливим параметром системи, а поверхня матеріалу за своєю природою нерівна, механізм осі Z потребує методу для визначення поверхні матеріалу. Це можна досягти трьома способами:

1. Датчик струму для виявлення підвищеного крутного моменту двигуна,
2. «поплавковий» вимикач та електричний або
3. «омічний» сенсорний контур, який замикається, коли екран пальника торкається матеріалу.

Вимірювання струму не є життєздатним методом для саморобних столів, але поплавкові вимикачі та омічні датчики обговорюються нижче:

2.7.7.1 Поплавкові вимикачі

Пальник встановлений на ковзній платформі, яка може підніматися, коли кінчик пальника торкається поверхні матеріалу і спрацьовує перемикач або датчик. Часто це досягається під контролем G-коду за допомогою команд G38. У цьому випадку після початкового зондування рекомендується віддалити зонд від поверхні, поки сигнал зонда не зникне при меншій швидкості. Також слід врахувати гістерезис перемикача.

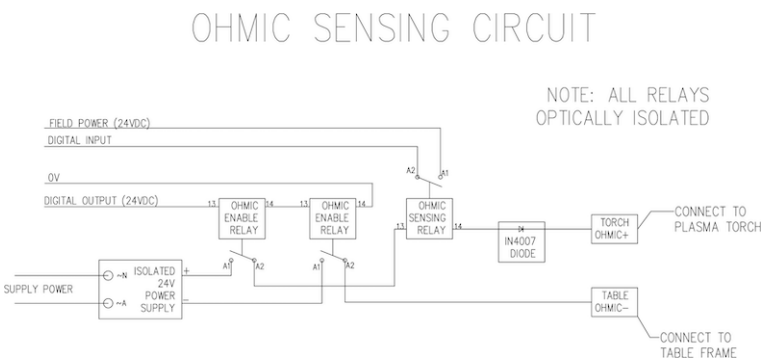
Незалежно від використовуваного методу зондування, наполегливо рекомендується використовувати поплавковий вимикач, щоб забезпечити резервний або вторинний сигнал, що запобігає пошкодженню пальника внаслідок аварії.

2.7.7.2 Омічне зондування

Омічне зондування базується на контакті між пальником і матеріалом, який діє як перемикач для активації електричного сигналу, що сприймається контролером CNC. За умови, що матеріал чистий, це може бути набагато точнішим методом зондування матеріалу, ніж поплавковий вимикач, який може спричинити відхилення поверхні матеріалу. Ця омічна схема зондування працює в надзвичайно несприятливих умовах, тому необхідно впровадити ряд засобів захисту від відмов, щоб забезпечити безпеку як електроніки CNC, так і оператора. При плазмовому різанні затискач заземлення, прикріплений до матеріалу, є позитивним, а пальник — негативним. Рекомендується:

1. Омічне зондування можна застосовувати лише там, де пальник має екран, ізольований від кінчика пальника, який передає ріжучу дугу.
2. Омічний контур використовує повністю окреме ізольоване джерело живлення, яке активує оптоізольоване реле, що дозволяє передавати зондувальний сигнал до контролера CNC.
3. Позитивний полюс кола повинен бути біля пальника
4. Обидві сторони кола повинні бути ізольовані оптоізольованими реле до початку вимірювання
5. Блокувальні діоди слід використовувати для запобігання потраплянню дугової напруги в омічний датчик.

Нижче наведено приклад схеми, яка, як було доведено, працює та сумісна з конфігурацією LinuxCNC QtPlasmaC.



2.7.7.3 Гіперсенсування за допомогою MESA THCAD-5

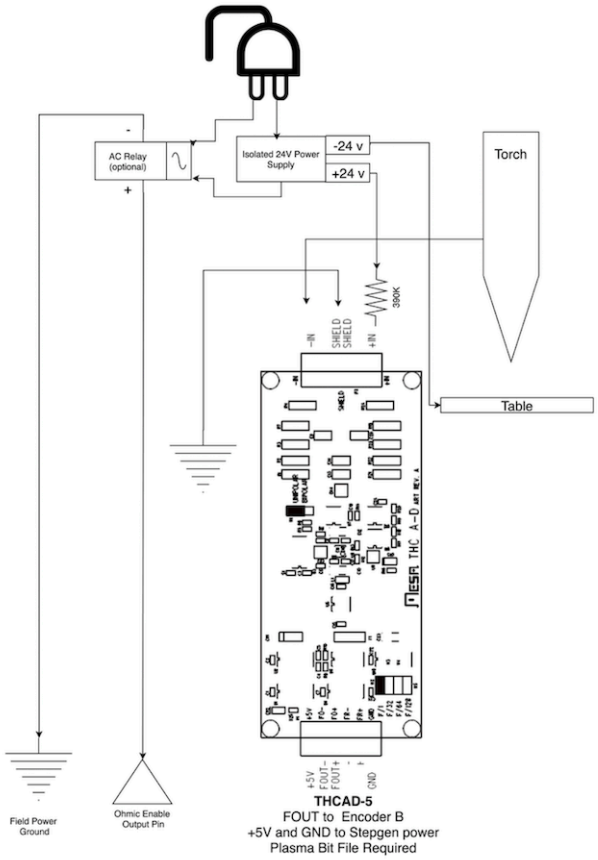
Більш досконалий метод виявлення матеріалу, що виключає використання реле та діодів, полягає у використанні іншого THCAD-5 для контролю напруги ланцюга виявлення матеріалу від ізольованого джерела живлення. Перевага цього методу полягає в тому, що THCAD розроблений для роботи в агресивному плазмовому електричному середовищі та повністю і безпечно ізолює логічну частину від частини високої напруги.

Для реалізації цього методу потрібен другий вхід енкодера.

При використанні карти Mesa доступне інше програмне забезпечення, яке забезпечує 2 додаткові входи енкодера A на контактах енкодера B та індексу енкодера. Це програмне забезпечення доступне для завантаження для плат 7I76E та 7I96 на веб-сайті Mesa на сторінках продуктів.

THCAD є достатньо чутливим, щоб бачити зростання напруги в ланцюзі при збільшенні тиску контакту. Компонент ohmic.comp, що входить до складу LinuxCNC, може контролювати напругу датчика і встановлювати поріг напруги, вище якого вважається, що контакт встановлено і вихід увімкнено. Контролюючи напругу, можна встановити нижчий поріг «розриву ланцюга», щоб створити сильний гістерезис перемикача. Це мінімізує помилкові спрацьовування. Під час тестування ми виявили, що зондування матеріалу за допомогою цього методу є більш чутливим і надійним, а також простішим у реалізації. Ще однією перевагою використання програмних виходів замість фізичних контактів вводу-виводу є те, що це звільняє контакти для використання в інших цілях. Ця перевага допомагає максимально ефективно використовувати Mesa 7I96, який має обмежену кількість контактів вводу-виводу.

На наступній схемі показано, як реалізувати схему гіперчутливості.



Ми використовували ізольований блок живлення Mean Well HDR-15 Ultra Slim DIN Rail Supply 24 V на DIN-рейці потужністю 15 Вт. Це пристрій з подвійною ізоляцією класу II, який витримує будь-яку дугову напругу, що може бути прикладена до клем.

2.7.7.4 Приклад HAL-коду для гіперсенсорики

Наступний код HAL можна вставити у файл custom.hal вашого QtPlasmaC, щоб увімкнути омічне зондування на енкодері 2 7I76E. Встановіть правильний бітовий файл і підключіть THCAD до IDX+ та IDX-. Обов’язково змініть налаштування калібрування відповідно до вашого THCAD-5.

```
# --- b''3b''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb''  ↩
    b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ab'' ---
loadrt ohmic names=ohmicsense
addf ohmicsense servo-thread

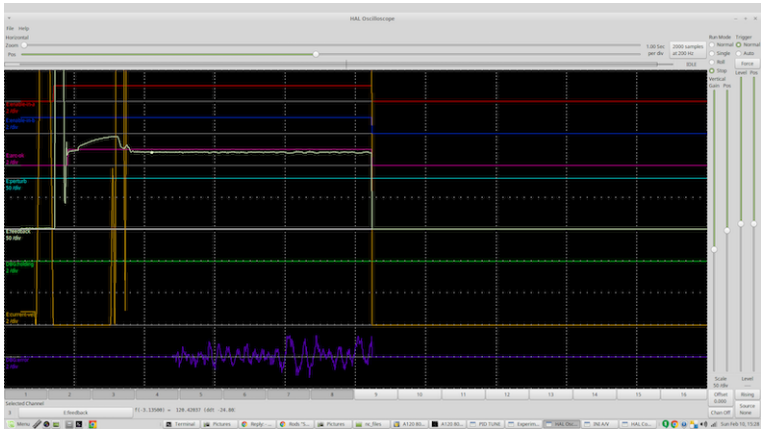
# --- b''Hb''b''Ab''b''Лb''b''Ab''b''Шb''b''Tb''b''Уb''b''Bb''b''Ab''b''Hb''b''Hb''b''Яb''  ↩
    b''Eb''b''Hb''b''Kb''b''Ob''b''Дb''b''Eb''b''Pb''b''Ab'' 7I76E 2 b''Дb''b''Лb''b''Яb'' b ↩
    ''Ob''b''Mb''b''Ib''b''Чb''b''Hb''b''Ob''b''Гb''b''Ob'' b''3b''b''Дb''b''Ab''b''Tb''b' ↩
    'Kb''b''Уb''b''Bb''b''Ab''b''Hb''b''Hb''b''Яb''---
setp hm2_7i76e.0.encoder.02.scale -1
setp hm2_7i76e.0.encoder.02.counter-mode 1
```

```
# --- b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''йb''b''тb''b''eb'' b''кb''b''ob'' ↩
    b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'' ---
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

# --- b''3b''b''ab''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''cb''b''иb''b''гb''b''нb'' ↩
    b''ab''b''лb'' b''ob''b''мb''b''ib''b''чb''b''нb''b''ob''b''гb''b''ob'' b''зb''b''ob''b' ↩
    'нb''b''дb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' QtPlasmaC ---
unlinkp db_ohmic.in
net ohmic-true ohmicsense.ohmic-on => db_ohmic.in
net plasmac:ohmic-enable    =>  ohmicsense.is-probing
```

2.7.8 Затримка THC

Коли дуга встановлюється, напруга на дузі значно підвищується, а потім стабілізується до стабільного рівня на висоті різу. Як показано зеленою лінією на зображенні нижче.



Важливо, щоб контролер плазми «зачекав» перед автоматичним вимірюванням напруги пальника та початком контролю THC. Якщо його увімкнути занадто рано, напруга буде вище бажаного рівня, і пальник буде знижуватися в спробі вирішити проблему надмірної висоти.

Під час наших випробувань цей час варіювався від 0,5 до 1,5 секунди залежно від верстата та матеріалу. Тому безпечним початковим налаштуванням є затримка в 1,5 секунди після отримання дійсного сигналу ArcOK перед увімкненням контролю THC. Якщо ви хочете скоротити цей час для певного матеріалу, програма Halscope від LinuxCNC дозволить вам побудувати графік напруги пальника та прийняти обґрунтоване рішення щодо найкоротшої безпечної затримки.

Note
Якщо швидкість різання не наближається до бажаної швидкості різання після закінчення цієї затримки, контролер повинен зачекати, поки вона буде досягнута, перш ніж увімкнути THC.

2.7.9 Вимірювання напруги пальника

Замість того, щоб покладатися на таблиці різання виробника для встановлення потрібної напруги пальника, багато людей (включно з автором) надають перевагу вимірюванню напруги під час

увімкнення THC та використанню її як заданого значення.

2.7.10 Відрив факела

Рекомендується передбачити механізм, який дозволить пальника «відірватися» або відпасти в разі удару об матеріал або підняту частину, що підрізається. Слід встановити датчик, який дозволить контролеру CNC виявити таку ситуацію і призупинити виконання програми. Зазвичай відрив здійснюється за допомогою магнітів, які фіксують пальник на платформі осі Z.

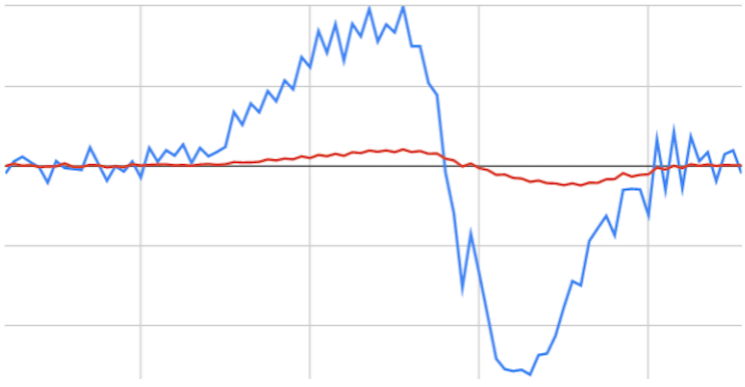
2.7.11 Кутовий замок / Захист від швидкості

Планувальник траєкторії LinuxCNC відповідає за перетворення команд швидкості та прискорення в рух, що відповідає законам фізики. Наприклад, рух сповільнюється при проходженні повороту. Хоча це не є проблемою для фрезерних верстатів або маршрутизаторів, це створює особливу проблему для плазмового різання, оскільки напруга дуги збільшується при сповільненні руху. Це призведе до того, що THC опустить пальник. Однією з величезних переваг системи управління THC, вбудованої в контролер руху LinuxCNC, є те, що вона завжди знає, що відбувається. Тому стає простим завданням контролювати поточну швидкість (`motion.current-velocity`) і призупиняти роботу THC, якщо вона падає нижче встановленого порогу (наприклад, на 10% нижче бажаної швидкості подачі).

2.7.12 Перетин порожнечі/виїмки

Якщо під час різання плазмовий пальник проходить над порожниною, напруга дуги швидко зростає, і THC реагує різким рухом вниз, що може розбити пальник об матеріал і, можливо, пошкодити його. Цю ситуацію важко виявити і вирішити. До певної міри її можна пом’якшити за допомогою хороших технік вкладання, але вона все одно може виникнути на більш товстому матеріалі, коли відпадає шматок. Це єдина проблема, яка ще не вирішена в рамках руху відкритого програмного забезпечення LinuxCNC.

Один із запропонованих методів полягає у моніторингу швидкості зміни напруги пальника з часом (dv/dt), оскільки цей параметр на порядки вищий при перетині порожнини, ніж при нормальному викривленні матеріалу. На наступному графіку показано низькороздільну діаграму dv/dt (синім кольором) при перетині порожнини. Червона крива — це ковзна середня напруги пальника.



Отже, має бути можливим порівняти середнє значення з dv/dt і зупинити роботу THC, коли dv/dt перевищує нормальний діапазон, очікуваний через викривлення. Необхідно провести додаткову роботу в цій галузі, щоб знайти дієве рішення в LinuxCNC.

2.7.13 Різання отворів та малих форм

Рекомендується уповільнювати різання під час вирізання отворів та невеликих форм.

Джон Мур каже: «Якщо ви хочете дізнатися подробиці про точне різання невеликих отворів, перегляньте рекламні листівки про технологію «True Hole Technology» від Hypertherm, а також загляньте на сайт PlasmaSpider, де користувач seanp опублікував багато інформації про свою роботу з використанням простої повітряної плазми.

Загальноприйнятий метод отримання якісних отворів діаметром від 37 мм і нижче товщини матеріалу з мінімальною конусністю за допомогою повітряної плазми полягає в наступному:

1. Використовуйте рекомендований струм різання для витратних матеріалів.
2. Використовуйте фіксовану (без THC) рекомендовану висоту скошування для витратних матеріалів
3. Зменшити рекомендовану швидкість подачі витратних матеріалів та матеріалів з 60% до 70%.
4. Почніть підведення в центрі отвору або поблизу нього.
5. Використовуйте перпендикулярне введення.
6. Без виведення, або легке перегорання, або передчасне вимикання пальника, залежно від того, що вам найкраще підходить.

Вам потрібно буде поекспериментувати, щоб отримати точний розмір отвору, оскільки пропил за цим методом буде ширшим, ніж ваш звичайний прямий розріз

Це уповільнення можна досягти шляхом маніпулювання швидкістю подачі безпосередньо в постпроце або за допомогою адаптивної подачі та аналогового виводу як входу. Це дозволяє використовувати M67/M68 для встановлення відсотка бажаної подачі для різання.

- Знання швидкості подачі

З попереднього обговорення очевидно, що контролер плазми повинен знати швидкість подачі, встановлену користувачем. Це створює проблему для LinuxCNC, оскільки швидкість подачі не зберігається LinuxCNC після буферизації та аналізу G-коду. Існує два підходи до вирішення цієї проблеми:

1. Перепризначте команду F та збережіть задану швидкість подачі в G-коді за допомогою команди M67/M68.
2. Зберігання різальних таблиць у плазмовому контролері та дозвіл на запит поточної швидкості подачі програмою G-коду (як це робить QtPlasmaC).

Новою функцією, доданою до LinuxCNC 2.9, яка є корисною для плазмового різання, є теги стану. Це додає «тег», який доступний для руху, що містить поточні швидкості подачі та швидкості для всіх активних команд руху.

2.7.14 Контакти вводу/виводу для плазмових контролерів

Плазмові різаки вимагають декількох додаткових контактів. У LinuxCNC немає жорстких правил щодо того, який контакт виконує яку функцію. У цій дискусії ми будемо припускати, що плазмовий інвертор має інтерфейс CNC, а контролер має активні високі входи (наприклад, Mesa 7I76E).

Плазмові столи можуть бути великими машинами, тому ми рекомендуємо вам витратити час на встановлення окремих вимикачів максимального/мінімального обмеження та вимикачів повернення в початкове положення для кожного з’єднання. Винятком може бути нижня межа осі Z. Коли

спрацьовує вимикач повернення в початкове положення, з'єднання сповільнюється досить повільно для забезпечення максимальної точності. Це означає, що якщо ви хочете використовувати швидкості повернення в початкове положення, які відповідають розміру столу, ви можете перевищити початкову точку спрацьовування на 50-100 мм. Якщо ви використовуєте спільний вимикач повернення в початкове положення/обмежувач, ви повинні перемістити датчик з точки спрацьовування за допомогою кінцевого HOME_OFFSET, інакше ви спричините помилку обмежувача, коли машина вийде з початкового положення. Це означає, що ви можете втратити 50 мм або більше ходу осі з спільними перемикачами повернення в початкове положення/обмеження. Це не відбувається, якщо використовуються окремі перемикачі повернення в початкове положення та обмеження.

Зазвичай потрібні такі контакти (зверніть увагу, що запропоновані з'єднання можуть не підходити для конфігурації QtPlasmaC):

2.7.14.1 Дуга в порядку (вхід)

- Інвертор замикає сухі контакти, коли встановлюється дійсна дуга
- Підключіть польове живлення до одного з виводів інвертора ArcOK.
- Підключіть інший термінал OK інвертора до вхідного контакту.
- Зазвичай підключається до одного з виводів ``motion.digital-``*<nn>* для використання з G-коду з M66

2.7.14.2 Паяльна лампа увімкнена (вихід)

- Вмикає реле для вмикання вимикача пальника в інверторі.
- Підключіть клеми пальника на інверторі до вихідних клем реле.
- Підключіть один кінець котушки до вихідного контакту.
- Підключіть інший кінець котушки до заземлення Field Power.
- Якщо використовується механічне реле, підключіть діод типу flyback (наприклад, серії IN400x) до клем котушки так, щоб смуга на діоді була спрямована до вихідного контакту.
- Якщо використовується твердотільне реле, може знадобитися дотримуватися полярності на виходах.
- За деяких обставин замість зовнішнього реле можна використовувати вбудоване шпиндельне реле на платі Mesa.
- Зазвичай підключено до `spindle.0.on`.



Warning
Наполегливо рекомендується не вмикати пальник, поки цей контакт має значення «false», інакше пальник не згасне після натискання кнопки estop.

2.7.14.3 Поплавковий вимикач (вхід)

- Використовується для поверхневого зондування. Датчик або перемикач, який активується, якщо пальник піднімається вгору, коли торкається матеріалу.
- Підключіть вихід датчика наближення до вибраного вхідного контакту. Якщо використовуються механічні перемикачі, підключіть один бік перемикача до живлення, а інший бік перемикача – до входу.
- Зазвичай підключено до `motion.probe-input`.

2.7.14.4 Увімкнення омічного датчика (вихід)

- Див. схему омічне зондування.
- Підключіть вихідний контакт до одного боку ізоляційних реле, а інший бік – до заземлення польового живлення.
- У конфігурації, відмінній від QtPlasmaC, зазвичай запускається за допомогою ``motion.digital-out-``<*nn*>, тому ним можна керувати в G-коді за допомогою `M62/M63/M64/M65`.

2.7.14.5 Омічний зонд (вхід)

- Уважно дотримуйтесь схеми омічне зондування, наведеної раніше.
- Ізольоване джерело живлення спрацьовує реле, коли екран пальника торкається матеріалу.
- Підключіть живлення польового пристрою до одного вихідного терміналу, а інший – до входу.
- Зверніть увагу на полярність реле, якщо використовуються опторозв'язані твердотільні реле.
- Зазвичай підключається до `motion.probe-input` і може бути з'єднаний з поплавковим вимикачем.

Як можна побачити, плазмові столи вимагають великої кількості контактів, і ми вже використали близько 15 входів до додавання звичайних аварійних вимикачів. Інші мають іншу думку, але на думку автора, Mesa 7I76E є кращим вибором, ніж дешевший 7I96, оскільки він підтримує MPG, перемикач вибору масштабу та осі та інші функції, які ви, можливо, захочете додати з часом. Якщо ваш стіл використовує сервоприводи, існує ряд альтернатив. Хоча є й інші постачальники, проектування вашої машини на основі екосистеми Mesa спростить використання їхньої плати THCAD для зчитування напруги дуги.

2.7.14.6 Датчик відриву пальника

- Як згадувалося раніше, слід встановити датчик відриву, який спрацьовує, якщо пальник падає.
- Зазвичай це пов'язано з `halui.program-pause`, щоб помилку можна було виправити та відновити роботу програми.

2.7.15 G-код для плазмових контролерів

Більшість контролерів плазми пропонують метод зміни налаштувань з G-коду. LinuxCNC підтримує це за допомогою `M67/M68` для аналогових команд і `M62-M65` для цифрових (команди ввімкнення/вимкнення Як це реалізується, є повністю довільним. Давайте подивимося, як це робить конфігурація LinuxCNC QtPlasmaC:

Виберіть «Налаштування матеріалу» в QtPlasmaC та використовуйте швидкість подачі для цього матеріалу.

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

Note
Користувачам з дуже великою кількістю записів у таблиці матеріалів QtPlasmaC може знадобитися збільшити параметр Q (наприклад, з Q1 до Q2).

2.7.15.1 Увімкнути/вимкнути роботу THC:

```
M62 P2 b''вb''b''иb''b''мb''b''кb''b''нb''b''eb'' THC (b''cb''b''иb''b''нb''b''xb''b''pb''b ↩
    ''ob''b''нb''b''ib''b''зb''b''ob''b''вb''b''ab''b''нb''b''ob'' b''зb'' b''pb''b''yb''b' ↩
    'xb''b''ob''b''мb'')
M63 P2 b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb'' THC (b''cb''b''иb''b''нb''b''xb''b ↩
    ''pb''b''ob''b''нb''b''ib''b''зb''b''ob''b''вb''b''ab''b''нb''b''ob'' b''зb'' b''pb''b' ↩
    'yb''b''xb''b''ob''b''мb'')
M64 P2 b''вb''b''иb''b''мb''b''кb''b''нb''b''eb'' THC (b''нb''b''eb''b''гb''b''ab''b''йb''b ↩
    ''нb''b''ob'')
M65 P2 b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb'' THC (b''нb''b''eb''b''гb''b''ab''b ↩
    ''йb''b''нb''b''ob'')
```

Зменште швидкість різання: (наприклад, для різання отворів)

```
M67 E3 Q0 b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b''шb''b' ↩
    'вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''нb''b''ab'' 100% b''вb''b' ↩
    'ib''b''дb'' b''зb''b''ab''b''дb''b''ab''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ob''b''cb''b''тb''b''ib''.
M67 E3 Q40 b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b''шb''b' ↩
    'вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''нb''b''ab'' 40% b''вb''b''ib'' ↩
    b''дb'' b''зb''b''ab''b''дb''b''ab''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''cb''b''тb''b''ib''.
M67 E3 Q60 b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b''шb''b' ↩
    'вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''нb''b''ab'' 60% b''вb''b''ib'' ↩
    b''дb'' b''зb''b''ab''b''дb''b''ab''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''cb''b''тb''b''ib''.
M67 E3 Q100 b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b''шb''b ↩
    ''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''нb''b''ab'' 100% b''вb''b' ↩
    'ib''b''дb'' b''зb''b''ab''b''дb''b''ab''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ob''b''cb''b''тb''b''ib''.
```

Компенсація різця:

```
G41.1 D#<_hal[plasmac_run.kerf-width-f]> ; b''дb''b''лb''b''яb'' b''лb''b''ib''b''вb''b' ↩
    'ob''b''їb'' b''чb''b''ab''b''cb''b''тb''b''иb''b''нb''b''иb'' b''зb''b''ab''b''пb''b' ↩
    'pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b''нb''b''ob''b''гb''b' ↩
    'ob'' b''шb''b''лb''b''яb''b''xb''b''yb''
G42.1 D#<_hal[plasmac_run.kerf-width-f]> b''дb''b''лb''b''яb'' b''пb''b''pb''b''ab''b''вb'' ↩
    b''ob''b''гb''b''ob'' b''бb''b''ob''b''кb''b''yb'' b''зb''b''ab''b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b''нb''b''ob''b''гb''b''ob'' b''шb''b' ↩
    'лb''b''яb''b''xb''b''yb''
G40 b''дb''b''лb''b''яb'' b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''нb''b''яb'' b ↩
    ''кb''b''ob''b''мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''їb''
```

Note
Інтеграторам слід ознайомитися з документацією LinuxCNC для різних команд G-коду LinuxCNC, згаданих вище.

2.7.16 Зовнішнє зміщення та плазмове різання

Зовнішні зміщення були введені в LinuxCNC з версією 2.8. Під зовнішніми мається на увазі, що ми можемо застосувати зміщення зовні до G-коду, про яке планувальник траєкторії нічого не знає. Найпростіше пояснити це на прикладі. Уявіть токарний верстат із зовнішнім зміщенням, яке застосовується за математичною формулою для обробки лопаті на кулачку. Токарний верстат сліпо обертається з фіксованим діаметром різання, а зовнішнє зміщення переміщує інструмент вперед і назад для обробки лопаті кулачка за допомогою застосованого зовнішнього зміщення.

Щоб налаштувати наш токарний верстат для обробки цього кулачка, нам потрібно виділити частину швидкості та прискорення осі для зовнішніх зміщень, інакше інструмент не зможе рухатися. Тут на допомогу приходить змінна INI OFFSET_AV_RATIO. Припустимо, ми вирішили, що нам потрібно виділити 20% швидкості та прискорення на зовнішнє зміщення осі Z. Ми встановлюємо це значення рівним 0,2. Наслідком цього є те, що максимальна швидкість та прискорення осі Z токарного верстата становлять лише 80% від можливого значення.

Зовнішні зміщення є дуже ефективним методом регулювання висоти пальника по осі Z за допомогою THC. Але плазма характеризується високими швидкостями і швидким прискоренням, тому обмежувати ці параметри не має сенсу. На щастя, в плазмовій машині ось Z або на 100% контролюється THC, або не контролюється взагалі. Під час розробки зовнішніх зміщень LinuxCNC було визнано, що рух осі Z за допомогою G-коду та THC є взаємовиключними. Це дозволяє нам обдурити зовнішні зміщення, щоб вони завжди давали 100% швидкості та прискорення. Ми можемо зробити це, подвоївши налаштування швидкості та прискорення осі Z верстата в файлі INI та встановивши OFFSET_AV_RATIO = 0,5. Таким чином, 100% максимальної швидкості та прискорення будуть доступні як для зондування, так і для THC.

Приклад: На метричній машині з двигуном NEMA23 з прямим приводом до кулькової гвинти 5 мм, максимальна швидкість 60 мм/с і прискорення 700 мм/с^2 були визначені як безпечні значення без втрати кроків. Для цієї машини встановіть вісь Z у файлі INI наступним чином:

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

З'єднання, пов'язане з цією віссю, матиме змінні швидкості та прискорення, встановлені наступним чином:

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

Для отримання додаткової інформації про зовнішні зміщення (для версії 2.8 або пізнішої) прочитайте розділ [AXIS_<letter>] Section документа про файл INI та розділ External Axis Offsets в документації LinuxCNC.

2.7.17 Зчитування напруги дуги за допомогою Mesa THCAD

Плата Mesa THCAD — це надзвичайно доступний за ціною і точний перетворювач напруги в частоту, призначений для роботи в несприятливих умовах з високим рівнем електромагнітних перешкод, що супроводжують плазмове різання. Внутрішній діапазон напруги становить 0–10 В. Цей діапазон можна легко розширити, додавши кілька резисторів, як описано в документації. Ця плата доступна у трьох версіях: новіша THCAD-5 з діапазоном 0-5 В, THCAD-10 з діапазоном 0-10 В і THCAD-300, яка попередньо відкалібрована для розширеного діапазону 300 В. Кожна плата калібрується індивідуально, і на неї наклеюється наклейка із зазначенням частоти при 0 В і повній шкалі. Для використання з LinuxCNC рекомендується вибрати дільник 1/32 за допомогою відповідного з'єднання на платі. У цьому випадку обов'язково розділіть зазначені частоти на 32. Це більш підходить для сервоприводу 1 кГц, а також дає THCAD більше часу для усереднення та згладжування вихідних даних.

Існує багато плутанини щодо того, як декодувати вихідний сигнал THCAD. Тож розглянемо на мить Mesa 7I76E та THCAD-10 з такими гіпотетичними даними калібрування:

- Повна шкала 928 кГц (928 кГц/32 = 29 кГц)
- 0 В 121,6 кГц (121,6 кГц/32 = 3,8 кГц)

Оскільки повна шкала становить 10 вольт, то частота на вольт становить:

(29000 Гц - 3800 Гц) / 10 В = 2520 Гц на вольт

Отже, якщо припустити, що у нас є вхід 5 вольт, розрахована частота буде:

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

Отже, тепер має бути досить зрозуміло, як перетворити частоту в її еквівалент напруги:

Напруга = (частота [Гц] - 3800 Гц) / (2520 Гц/В)

2.7.17.1 З'єднання THCAD

З боку високої напруги:

- Підключіть розділену або необроблену напругу дуги до I_N+ та I_N-
- Підключіть екран з'єднувального кабелю до роз'єму екранування.
- Підключіть інший вивід екрану до заземлення корпусу.

Припускаючи, що він підключений до Mesa 7I76E, підключіть вихід до входу енкодера шпинделя:

- THCAD +5 В до TB3, контакт 6 (+5 VP)
- THCAD -5 В до TB3 Pin 1 (GND)
- THCAD FOUT+ до TB3 Pin 7 (ENC A+)
- THCAD FOUT- до TB3 Pin 8 (ENC A-)

2.7.17.2 Початкове тестування THCAD

Переконайтеся, що у вашому INI-файлі є такі рядки (якщо припускати Mesa 7I76E):

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

Увімкніть контролер і відкрийте Halshow (AXIS: Show Homing Configuration), знайдіть `hm2_7i76e.0.enc pin`. При напрузі 0 В він повинен коливатися в районі частоти 0 В (у нашому прикладі 3800). Візьміть 9-вольтову батарею і підключіть її до I_N+ і I_N-. Для THCAD-10 тепер ви можете обчислити очікувану швидкість (26 480 в нашому гіпотетичному прикладі). Якщо ви пройшли цей тест, то ви готові до налаштування вашого плазмового контролера LinuxCNC.

2.7.17.3 Яку модель THCAD використовувати?

THCAD-5 корисний, якщо ви плануєте використовувати його для омічного зондування. Без сумніву, THCAD-10 є більш гнучким пристроєм, і його масштаб легко змінювати. Однак є одне застереження, яке може мати значення для деяких дешевших плазмових різаків із вбудованим дільником напруги. А саме, внутрішні резистори можуть сприйматися THCAD як частина його власного зовнішнього опору і давати помилкові результати. Наприклад, дільник 16:1 на плазмових різаках Everlast потрібно розглядати як 24:1 (а 50:1 стає 75:1). Це не є проблемою для більш відомих брендів (наприклад, Thermal Dynamics, Hypertherm, ESAB тощо). Тому, якщо ви бачите нижчі, ніж очікувані, напруги різання, можливо, краще переналаштувати THCAD для зчитування вихідної напруги дуги.

Пам'ятаючи, що напруга плазмової дуги потенційно смертельна, ось деякі запропоновані критерії.

Запуск пілотної дуги Оскільки ймовірність значних електромагнітних перешкод мала, ви зможете безпечно встановити THCAD у вашій панелі керування, якщо дотримувалися наших інструкцій з будівництва.

- Якщо у вас немає дільника напруги, встановіть масштабувальні резистори всередині плазмового різака та встановіть THCAD на панелі керування або дотримуйтесь рекомендацій для машин з високочастотним пуском.
- Якщо у вас є дільник напруги, встановіть THCAD-10 у панель керування. У нас не було проблем із цією конфігурацією та плазмовим різаком Thermal Dynamics на 120 А.

HF Початок Встановіть THCAD на інверторі, оскільки частотний сигнал набагато стійкіший до електромагнітних перешкод.

- Якщо у вас немає дільника напруги і є місце всередині плазмового різака, встановіть THCAD-300 всередину плазмового різака.
- Якщо у вас немає дільника напруги і у вас немає місця всередині плазмового різака, встановіть THCAD-10 в металевому корпусі зовні плазмового різака і встановіть 50% опору масштабування на кожному з I_N+ і I_N- всередині корпусу плазмового різака, щоб з корпусу не виходила смертельна напруга.
- Якщо у вас є дільник напруги, встановіть THCAD-10 у металевий корпус зовні плазмового різака

Необроблена напруга дуги, представлена на роз'ємі У цьому випадку, незалежно від методу запуску дуги, в схемі, ймовірно, вже є резистори, що запобігають смертельним ураженням струмом, тому рекомендується використовувати THCAD-10, щоб цей опір (зазвичай 200 кОм) можна було врахувати при виборі резистора масштабування, оскільки ці резистори спотворюють напругу, що реєструється THCAD-300.

2.7.18 Постпроцесори та вкладення

Плазма нічим не відрізняється від інших операцій з ЧПК тим, що вона:

1. Розроблено в CAD (де виводиться у форматі DXF або іноді SVG).
2. Обробляється в CAM для створення остаточного G-коду, який завантажується в машину
3. Різання деталей за допомогою команд G-коду CNC.

Деякі люди досягають хороших результатів за допомогою Inkscape та інструментів G-code, але SheetCam є дуже вигідним рішенням, а для LinuxCNC доступно чимало постпроцесорів. SheetCam має низку розширених функцій, призначених для плазмового різання, і за свою ціну є очевидним вибором для всіх, хто регулярно займається плазмовим різанням.

2.7.19 Проектування для шумних електричних середовищ

Плазмове різання за своєю суттю є надзвичайно агресивним і шумним електричним середовищем. Якщо у вас є проблеми з електромагнітними перешкодами, обладнання не буде працювати належним чином. У більш очевидному прикладі, ви можете увімкнути пальник, і комп'ютер перезавантажиться, але можуть бути й інші дивні симптоми. Практично всі вони виникають тільки під час різання пальником, часто при його першому увімкненні.

Тому розробники систем повинні ретельно підбирати компоненти та проектувати їх з нуля, щоб впоратися з цим несприятливим середовищем та уникнути впливу електромагнітних перешкод (EMI). Недотримання цих вимог може призвести до безлічі годин марних спроб усунення несправносте

Вибір мережевих плат, таких як Mesa 7I76E або дешевша 7I96, допомагає розмістити ПК подалі від електроніки та плазмового пристрою. Це обладнання також дозволяє використовувати 24-вольтні логічні системи, які набагато краще переносять шум. Компоненти слід встановлювати в

металевому корпусі, підключеному до заземлення мережі. Настійно рекомендується встановити фільтр EMI на підключенні до мережі живлення. Найпростіший спосіб — використовувати роз'єм IEC з фільтром EMI, який зазвичай використовується в ПК та електроприладах, що дозволяє досягти цього без додаткових зусиль. Сплануйте розташування компонентів у корпусі таким чином, щоб мережа живлення, дроти двигуна високої напруги та логічні сигнали були максимально віддалені один від одного. Якщо вони все ж мають перетинатися, тримайте їх під кутом 90 градусів.

Пітер Уоллес з Mesa Electronics пропонує: «Якщо у вас є сумісний з CNC плазмовий джерело з дільником напруги, я б встановив THCAD всередині корпусу електроніки разом з усім іншим апаратним забезпеченням руху. Якщо у вас є ручний плазмовий джерело і ви зчитуєте вихідну напругу плазми, я б встановив THCAD якомога ближче до плазмового джерела (навіть всередині корпусу плазмового джерела, якщо це можливо). У цьому випадку переконайтеся, що всі нижні з'єднання THCAD повністю ізольовані від джерела плазми. Якщо ви використовуєте екрановану коробку для THCAD, екран повинен бути підключений до заземлення вашого електронного корпусу, а не до заземлення джерела плазми.»

Рекомендується прокласти окремий заземлюючий провід від корпусу двигуна та пальника до центральної точки заземлення на машині. Підключіть заземлюючий провід плазми до цієї точки та, за бажанням, заземлюючий стрижень, вбитий у землю якомога ближче до машини (особливо якщо це плазмова машина з високочастотним пуском).

Зовнішня проводка до двигунів повинна бути екранованою і мати відповідний розмір, щоб витримувати струм, що проходить через ланцюг. Екран повинен залишатися непідключеним на стороні двигуна і заземленим на стороні блоку управління. Розгляньте можливість використання додаткового контакту на будь-яких роз'ємах у блоці управління, щоб заземлення могло бути продовжене через блок управління і заземлене на шасі безпосередньо на контролері крокового/серводвигуна.

Нам відомо про принаймні одного комерційного виробника систем, який мав проблеми з індукованими електричними перешкодами в омічному сенсорному контурі. Хоча це можна пом'якшити за допомогою феритових намистин і намотування кабелю, також рекомендується додати фільтр для проходження силової лінії в місці, де омічний сенсорний сигнал надходить до корпусу електроніки.

Томмі Беріша, майстер з побудови плазмових машин з обмеженим бюджетом, каже: «Якщо у вас обмежений бюджет, розгляньте можливість використання старих блоків живлення для ноутбуків. Вони дуже хороші, мають хорошу фільтрацію, повністю ізольовані, обмежують струм (це стає дуже важливим, коли щось йде не так), і підключити 2 або 3 з них послідовно легко, оскільки вони ізольовані. Майте на увазі, що деякі з них мають заземлення, підключене до негативного виходу, тому його потрібно від'єднати, що легко зробити за допомогою кабелю живлення без заземлення.»

2.7.20 Грунтові води

Мінімальний рівень води під рівнем різання пальника повинен становити приблизно 40 мм. Бажано, щоб під рейками був простір, щоб вода могла вирівнятися і витікати під час різання. Дуже добре, якщо над металевою пластиною, що ріжеться, буде трохи води, оскільки це дозволяє позбутися невеликої кількості пилу. Найкраще працювати в зануреному стані, але це не бажано для систем, що використовуються не постійно, оскільки це призводить до корозії пальника. Додавання харчової соди до води дозволить зберегти стіл у хорошому стані протягом багатьох років, оскільки вона запобігає корозії, коли рейки знаходяться під водою, а також зменшує запах водяної пари. Деякі люди використовують резервуар для води з входом для стисненого повітря, щоб за потреби підштовхувати воду з резервуара до рівня води і таким чином змінювати рівень води.

2.7.21 Столи з нижнім відводом повітря

Багато комерційних столів використовують конструкцію з нисхідним потоком повітря, тому вентилятор використовуються для всмоктування повітря через щілини, щоб уловлювати дим і іскри. Часто

столи розділені на зони, тому тільки секція під паяльником відкривається для виходу повітря, часто використовуючи пневматичні циліндри та пневматичні соленоїди для відкриття заслінок. Запуск цих зон є відносно простим, якщо використовувати вісь або положення з'єднання одного з рухомих штифтів і компонент lincurve для відображення зон витяжної вентиляції на відповідний вихідний штифт.

2.7.22 Проектування для швидкості та прискорення

У плазмовому різанні швидкість і прискорення мають вирішальне значення. Чим вище прискорення, тим менше машина повинна сповільнюватися при проходженні поворотів. Це означає, що портал повинен бути якомога легшим, не втрачаючи при цьому жорсткості на кручення. Алюмінієвий профіль розміром 100 мм × 100 мм × 2 мм має жорсткість на кручення, еквівалентну жорсткості екструдованого профілю з T-образним пазом розміром 80 мм × 80 мм, але при цьому на 62 % легший. Тож чи зручність використання T-образних пазів переважає додаткові витрати на конструкцію?

2.7.23 Відстань, пройдена за оберт двигуна

Крокові двигуни страждають від резонансу, а прямий привід шестерні, ймовірно, означає, що двигун працює в несприятливих умовах. В ідеалі для плазмових машин вважається ідеальною відстань близько 15-25 мм на один оберт двигуна, але навіть близько 30 мм на один оберт все ще є прийнятним. Гвинтовий механізм з кроком 5 мм і редуктором 3:1 або 5:1 є ідеальним для осі Z.

2.7.24 Конфігурація плазми QtPlasmaC LinuxCNC

QtPlasmaC, яке складається з компонента HAL (plasmac.hal) та повних конфігурацій для графічного інтерфейсу QtPlasmaC, отримало значний внесок від багатьох учасників руху LinuxCNC Open Source, які з 2015 року сприяли поглибленню розуміння плазмових контролерів. Було проведено багато тестувань і розроблено багато рішень, щоб QtPlasmaC досягнув свого поточного робочого стану. Було включено все, від проектування схем до управління G-кодом і конфігурації. Крім того, QtPlasmaC підтримує зовнішні THC, такі як Proma 150, але справді розкриває свій потенціал у поєднанні з контролером Mesa, оскільки це дозволяє інтегратору включити перетворювач напруги в частоту Mesa THCAD, який спеціально розроблений для роботи в агресивному плазмовому середовищі.

QtPlasmaC розроблений як автономний програмний продукт і включає можливість додавання ваших різальних схем, а також функції для використання з постпроцесором, таким як SheetCam.

Система QtPlasmaC тепер включена у версію 2.9 та вище LinuxCNC. Вона є досить зрілою та була значно вдосконалена з часу написання першої версії цього посібника. QtPlasmaC визначатиме підтримку плазми LinuxCNC на багато років вперед, оскільки вона включає всі функції пропрієтарної висококласної системи управління плазмою за ціною відкритого програмного забезпечення.

2.7.25 Контролер Hypertherm RS485

Деякі плазмові різаки Hypertherm мають інтерфейс RS485, що дозволяє контролеру (наприклад, LinuxCNC) встановлювати амперaж, тиск і режим. Багато хто використовував для цього компонент, написаний на Python, що не працює в режимі реального часу. З недавнього часу QtPlasmaC підтримує цей інтерфейс нативно. Інформацію про те, як ним користуватися, дивіться в документації QtPlasmaC.

Поєднання повільної швидкості передачі даних, що використовується Hypertherm, та компонента, що не працює в режимі реального часу, робить зміну стану машини досить повільною, тому, як правило, неможливо змінювати налаштування під час різання.

При виборі інтерфейсу RS485 для використання на стороні ПК користувачі повідомляють, що інтерфейси USB-RS485 не є надійними. Хороші надійні результати були досягнуті при використанні апаратного інтерфейсу RS232 (наприклад, PCI/PCIe або порту материнської плати) та відповідного перетворювача RS485. Деякі користувачі повідомляють про успішне використання карти Sunix P/N: SER5037A PCI RS2322 та універсального перетворювача XC4136 RS232 на RS485 (який іноді може включати також кабель USB).

2.7.26 Постпроцесори для плазмового різання

Програми CAM (Computer Aided Manufacture, комп'ютерне виробництво) є містком між CAD (Computer Aided Design, комп'ютерне проектування) та кінцевою операцією CNC (Computer Numerical Control, комп'ютерне числове управління). Вони часто містять постпроцесор, який користувач може налаштувати для визначення коду, що генерується для конкретної машини або діалекту G-коду.

Багато користувачів LinuxCNC цілком задоволені використанням Inkscape для конвертації векторних SVG-файлів у G-код. Якщо ви використовуєте плазмовий різак для хобі або домашнього використання, розгляньте цей варіант.

Однак, якщо ваші потреби є більш складними, ймовірно, найкращим і найдоступнішим за ціною рішенням є SheetCam. SheetCam підтримує як Windows, так і Linux, і для нього доступні постпроцесори включаючи конфігурацію QtPlasmaC. SheetCam дозволяє розміщувати деталі на повному аркуші матеріалу та налаштовувати набори інструментів і фрагменти коду відповідно до ваших потреб. Постпроцесори SheetCam — це текстові файли, написані мовою програмування Lua, і їх, як правило, легко модифікувати відповідно до ваших вимог. Для отримання додаткової інформації зверніться до веб-сайту SheetCam та їхнього форуму підтримки.

Ще один популярний постпроцесор входить до популярного пакета Fusion360, але включені постпроцесори потребуватимуть певного налаштування.

LinuxCNC — це програма для CNC, і обговорення методів CAM, окрім цього вступного обговорення, виходить за рамки LinuxCNC.

Chapter 3

Майстри налаштування

3.1 Майстер налаштування крокового двигуна

3.1.1 Вступ

LinuxCNC здатний керувати широким спектром обладнання, використовуючи багато різних апаратних інтерфейсів.

StepConf — це програма, яка генерує конфігураційні файли для LinuxCNC для певного класу верстатів з ЧПК: тих, що керуються через «стандартний паралельний порт» і керуються сигналами типу «крок і напрямок».

StepConf встановлюється під час встановлення LinuxCNC і знаходиться в меню CNC.

StepConf розміщує файл у каталозі linuxcnc/config для зберігання вибору для кожної створеної вами конфігурації. Коли ви щось змінюєте, вам потрібно вибрати файл, який відповідає назві вашої конфігурації. Розширення файлу — .stepconf.

Майстер StepConf найкраще працює з роздільною здатністю екрана щонайменше 800 x 600.

3.1.2 Початкова сторінка



Figure 3.1: Вступна сторінка StepConf

Перші три перемикачі не потребують пояснень:

- «Створити нове» – створює нову конфігурацію.
- «Змінити» — змінити існуючу конфігурацію. Після вибору цієї опції з’явиться вікно вибору файлу, де ви зможете вибрати файл .stepconf для зміни. Якщо ви внесли будь-які зміни до основного файлу HAL або INI, вони будуть втрачені. Зміни до файлів custom.hal та custom_postgui.hal не будуть змінені майстром StepConf. StepConf виділить останню версію lastconf, яка була створена.
- «Імпорт» — імпортує файл конфігурації Mach і намагається перетворити його на файл конфігурації LinuxCNC. Після імпорту ви перейдете на сторінки StepConf, щоб підтвердити/змінити записи. Оригінальний файл XML Mach не буде змінений.

Ці наступні параметри будуть записані у файлі налаштувань для наступного запуску StepConf.

- «Створити ярлик на робочому столі» – це розмістить посилання на файли на робочому столі.

- «Створити панель запуску робочого столу» – це розмістить панель запуску на робочому столі для запуску вашої програми.
- «Створити імітацію обладнання» – це дозволяє створити конфігурацію для тестування, навіть якщо у вас немає фактичного обладнання.

3.1.3 Основна інформація

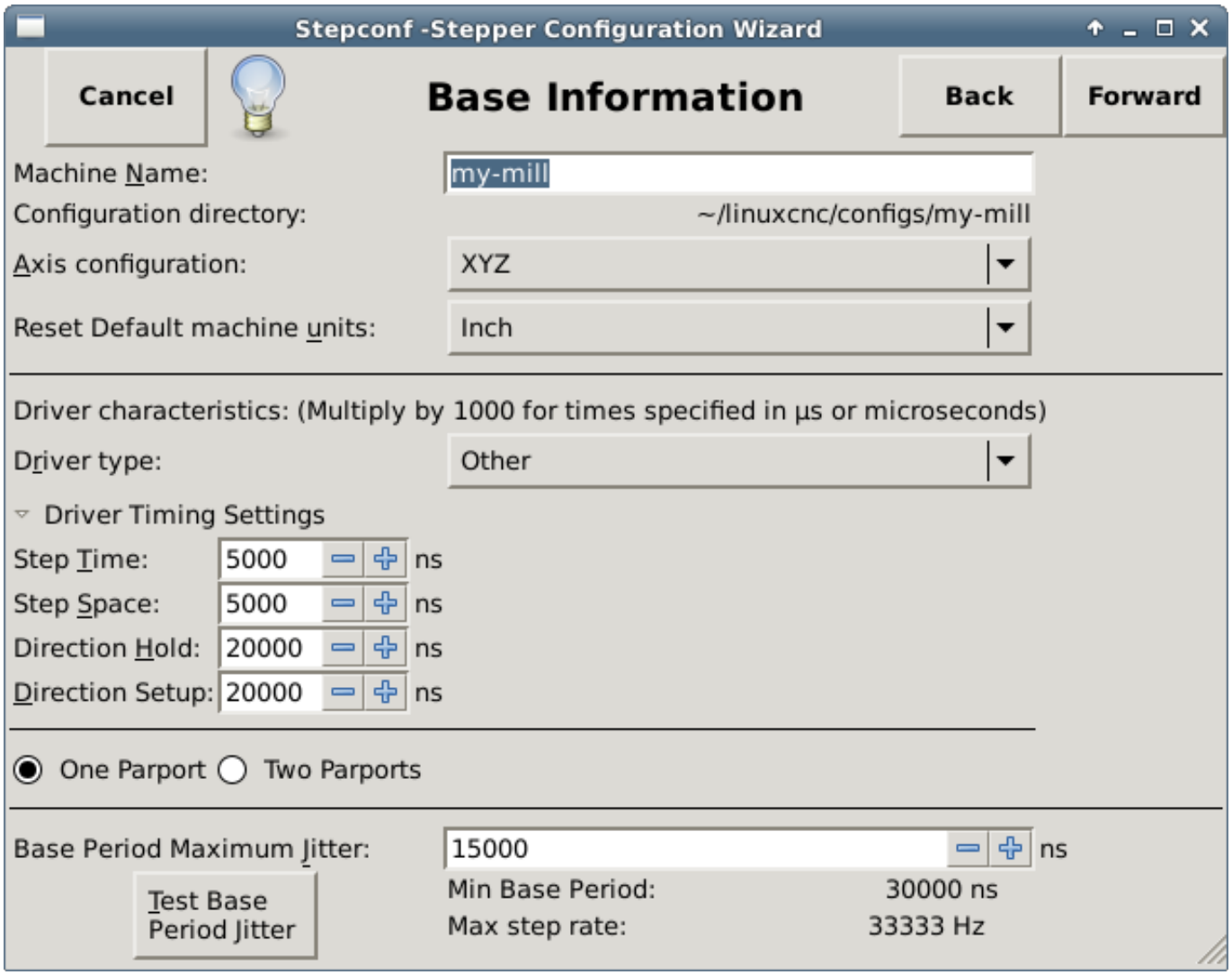


Figure 3.2: Сторінка з основною інформацією

- «Створити імітацію обладнання» – це дозволяє створити конфігурацію для тестування, навіть якщо у вас немає фактичного обладнання.
- «Назва машини» – виберіть назву для вашої машини. Використовуйте лише великі літери, малі літери, цифри, символи - та _.
- «Конфігурація осі» – виберіть XYZ (фрезерний верстат), XYZA (4-осьовий фрезерний верстат) або XZ (токарний верстат).
- «Одиниці виміру» — виберіть дюйми або міліметри. Усі наступні записи будуть вводитися у вибраних одиницях виміру. Зміна цього параметра також змінює значення за замовчуванням

у розділі «Осі». Якщо ви зміните цей параметр після вибору значень у будь-якому з розділів осей, вони будуть замінені значеннями за замовчуванням вибраних одиниць виміру.

- «Тип драйвера» — якщо у вас є один із крокових драйверів, перелічених у випадаючому списку, виберіть його. В іншому випадку виберіть «Інший» і знайдіть значення синхронізації в технічному паспорті вашого драйвера та введіть їх у «Налаштуваннях синхронізації драйвера» в наносекундах. Якщо в технічному паспорті вказано значення в мікросекундах, помножте його на 1000. Наприклад, введіть 4,5 мкс як 4500 нс.

  Список деяких популярних накопичувачів разом із їхніми значеннями синхронізації можна знайти на вікі-сторінці LinuxCNC.org за адресою Синхронізація крокового накопичувача.

  Додаткова обробка або ізоляція сигналу, така як оптопари та RC-фільтри на роз'ємних платах, можуть накладати власні обмеження за часом, на додаток до обмежень драйвера. Можливо, вам доведеться додати деякий час до вимог приводу, щоб це врахувати.

  У селекторі конфігурації LinuxCNC вже налаштовані конфігурації для Sherline. * «Час кроку» – тривалість крокового імпульсу в наносекундах. Якщо ви не впевнені щодо цього налаштування, значення 20 000 підійде для більшості накопичувачів. * «Інтервал кроків» – мінімальний час між імпульсами кроків у наносекундах. Якщо ви не впевнені щодо цього налаштування, значення 20 000 підійде для більшості накопичувачів. * «Утримання напрямку» – час утримання штифта напрямку після зміни напрямку в наносекундах. Якщо ви не впевнені щодо цього налаштування, значення 20 000 підійде для більшості накопичувачів. * «Налаштування напрямку» – час до зміни напрямку після останнього імпульсу кроку в наносекундах. Якщо ви не впевнені щодо цього налаштування, значення 20 000 підійде для більшості приводів. * «Один / Два паралельних порти» – виберіть, скільки паралельних портів потрібно налаштувати. * «Максимальний джиттер базового періоду» — введіть тут результат тесту затримки. Щоб запустити тест затримки, натисніть кнопку «Тестувати джиттер базового періоду». Докладнішу інформацію див. у розділі Тест затримки.

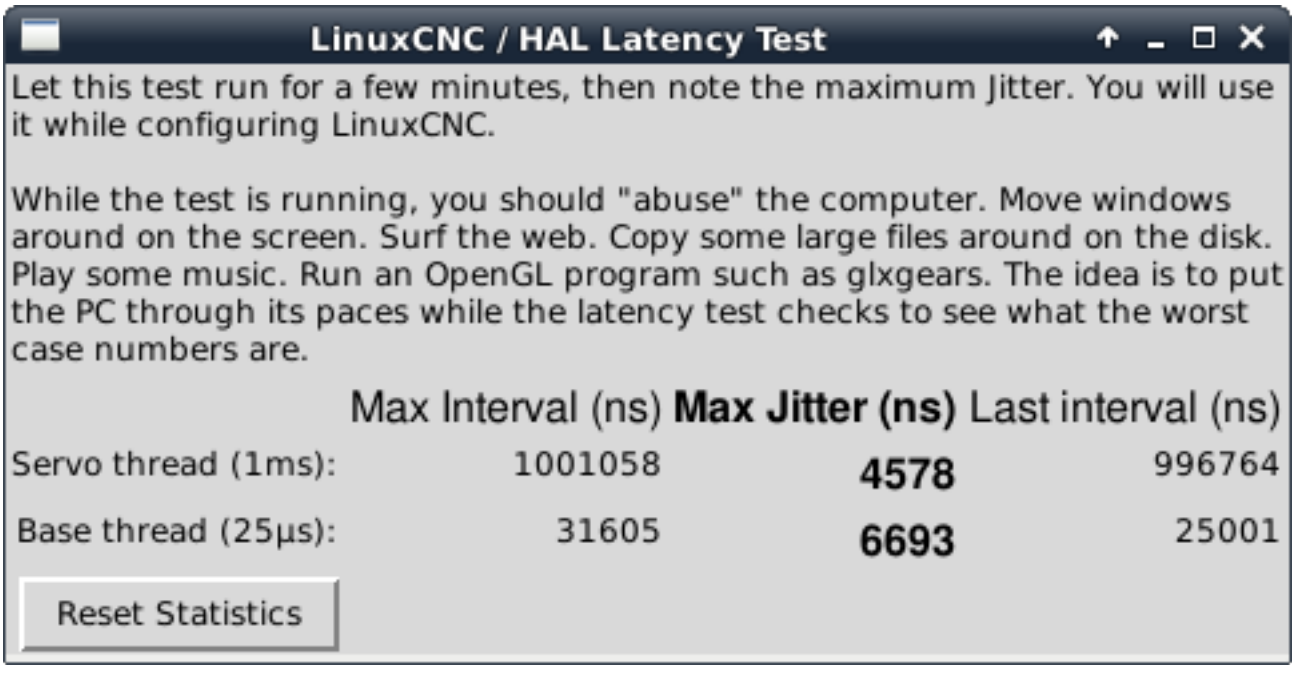


Figure 3.3: Тест на затримку

- «Максимальна швидкість кроку» – StepConf автоматично розраховує максимальну швидкість кроку на основі введених характеристик драйвера та результату тесту затримки.
- «Мінімальний базовий період» – StepConf автоматично визначає мінімальний базовий період на основі введених характеристик драйвера та результатів тесту затримки.

Важливим числом у результаті тесту затримки є «максимальне коливання». У наведеному вище прикладі 9075 наносекунд (нс) або 9,075 мікросекунд (мкс) – це найвище коливання. Введіть максимальне значення коливання в полі «Максимальне коливання базового періоду».

3.1.4 Налаштування паралельного порту

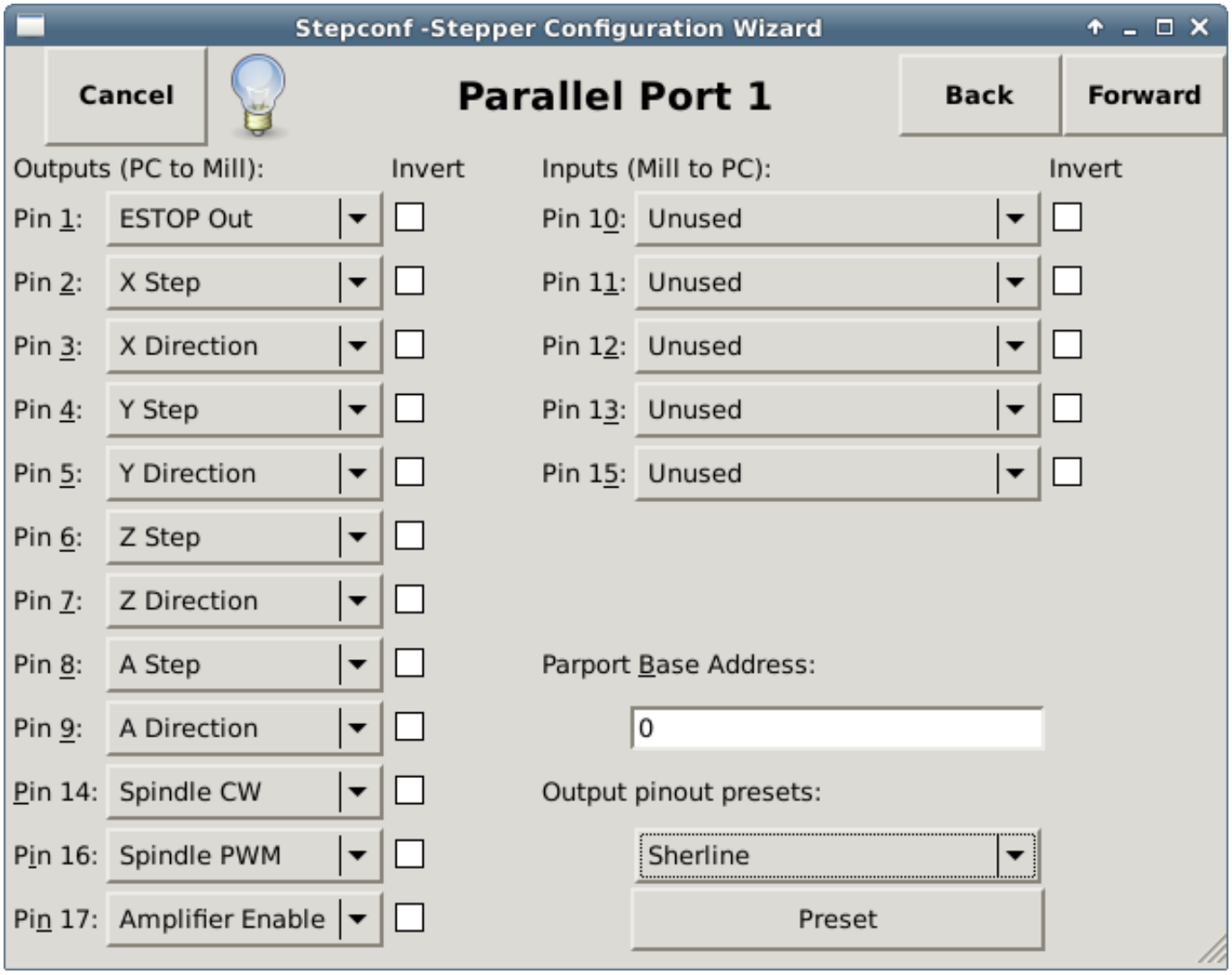


Figure 3.4: Сторінка налаштування паралельного порту

Ви можете вказати адресу у шістнадцятковому форматі (часто 0x378) або як номер порту Linux за замовчуванням (ймовірно, 0)

Для кожного виводу виберіть сигнал, який відповідає розпіновці вашого паралельного порту. Увімкніть прапорець «інвертувати», якщо сигнал інвертований (0 В для істини/активного, 5 В для хибності/неактивного).

- «Попередні налаштування вихідних розпіновок» – автоматично встановлює контакти з 2 по 9 відповідно до стандарту Sherline (напрямок на контактах 2, 4, 6, 8) або стандарту Xylotex (напрямок на контактах 3, 5, 7, 9).

- «Входи та виходи» – якщо вхід або вихід не використовується, встановіть для опції значення «Не використовується».
- «Зовнішній аварійний зупин» – цей параметр можна вибрати зі спадного списку вхідних контактів. Типовий ланцюг аварійного зупину використовує всі нормально замкнуті контакти.
- «Вимикачі відправлення та кінцеві вимикачі» – їх можна вибрати з випадаючого списку вхідних контактів для більшості конфігурацій.
- «Зарядний насос» — якщо ваша плата драйвера вимагає сигналу зарядного насоса, виберіть «Зарядний насос» зі списку, що випадає, для виходу, який ви хочете підключити до входу зарядного насоса. Вихід зарядного насоса підключається до базової нитки за допомогою Step-Conf. Вихід зарядного насоса буде приблизно 1/2 від максимальної швидкості кроку, показаної на сторінці «Базова конфігурація машини».
- «Напруга плазмової дуги» — якщо вам потрібно, щоб Mesa THCAD вводила напругу плазмової дуги, виберіть «Напруга плазмової дуги» зі списку вихідних контактів. Це дозволить активувати сторінку THCAD під час процедури налаштування для введення параметрів карти.

3.1.5 Налаштування паралельного порту 2

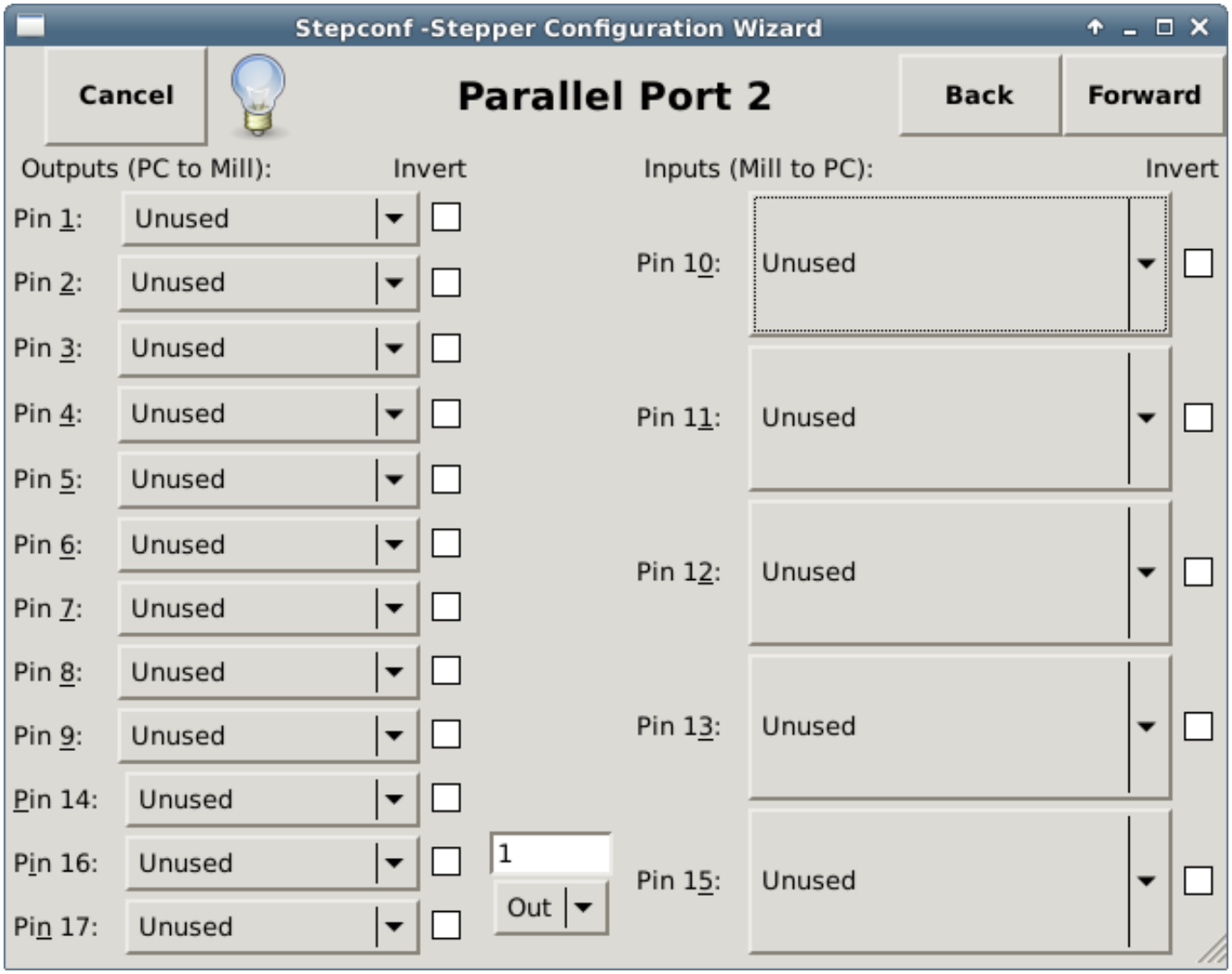


Figure 3.5: Сторінка налаштування паралельного порту 2

Другий паралельний порт (якщо вибрано) можна налаштувати, а його контакти призначити на цій сторінці. Не можна вибрати сигнали кроку та напрямку. Ви можете вибрати вхід або вихід, щоб максимізувати кількість доступних вхідних/вихідних контактів. Ви можете вказати адресу у шістнадцятковому форматі (зазвичай 0x378) або як номер порту за замовчуванням у Linux (ймовірно 1).

3.1.6 Конфігурація осі

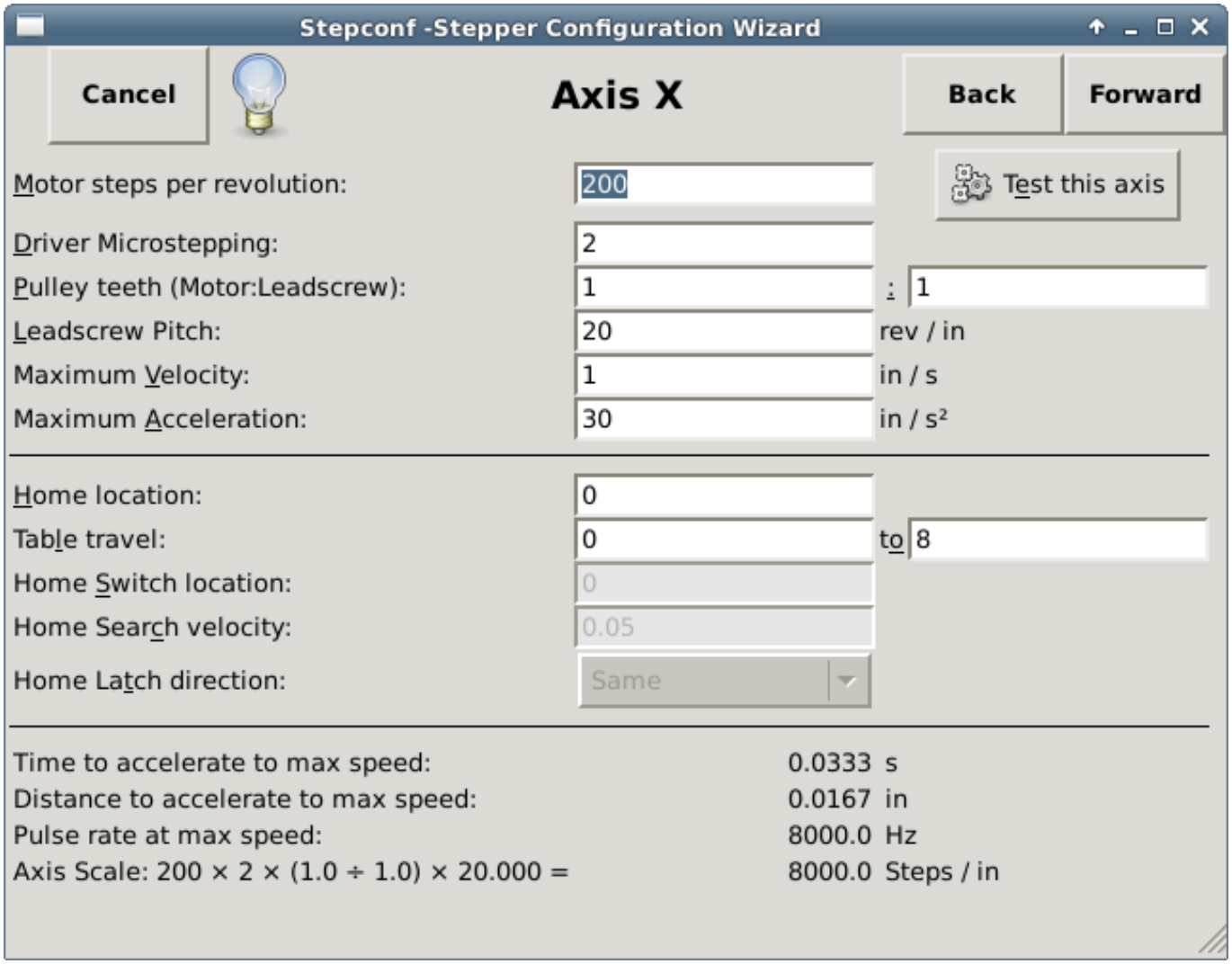


Figure 3.6: Екран конфігурації осі

- «Кількість кроків двигуна за один оберт» — кількість повних кроків за один оберт двигуна. Якщо ви знаєте, скільки градусів припадає на один крок двигуна (наприклад, 1,8 градуса), то розділіть 360 на кількість градусів на крок, щоб отримати кількість кроків за один оберт двигуна.
- «Мікрокроки драйвера» – величина мікрокроків, що виконуються драйвером. Введіть «2» для напівкроків.
- «Передавальне число шківа» – якщо у вашому верстаті є шківи між двигуном і ходовим гвинтом, введіть тут передавальне число. Якщо ні, введіть «1:1».
- «Крок ходового гвинта» — введіть тут крок ходового гвинта. Якщо ви вибрали одиниці виміру «дюйм», введіть кількість різьблень на дюйм. Якщо ви вибрали одиниці виміру «мм», введіть кількість міліметрів на оберт (наприклад, введіть 2 для 2 мм/оберт). Якщо машина рухається в неправильному напрямку, введіть тут від’ємне число замість додатного або змініть напрямок штифта для осі.
- «Максимальна швидкість» – введіть максимальну швидкість для осі в одиницях за секунду.

- «Максимальне прискорення» — правильні значення для цих параметрів можна визначити тільки експериментальним шляхом. Див. Визначення максимальної швидкості для налаштування швидкості та Визначення максимального прискорення для налаштування прискорення.
- «Початкове положення» — положення, в яке переміщується верстат після завершення процедури повернення в початкове положення для цієї осі. Для верстатів без кінцевих вимикачів це положення, в яке оператор вручну переміщує верстат перед натисканням кнопки «Початкове положення». Якщо ви поєднуєте кінцеві вимикачі та вимикачі обмеження, ви повинні переміститися з вимикача в початкове положення, інакше ви отримаєте помилку обмеження з'єднання.
- «Хід столу» – діапазон ходу для цієї осі на основі початку координат верстата. Домашнє положення має бути всередині «Ходу столу» та не дорівнювати одному зі значень ходу столу.
- «Розташування вимикача початкового положення» — місце, в якому вимикач початкового положення спрацьовує або відпускається відносно початку координат машини. Цей пункт і два наступні з'являються тільки в тому випадку, якщо в роз'ємі паралельного порту були вибрані вимикачі початкового положення. Якщо ви поєднуєте вимикачі початкового положення і кінцеві вимикачі, розташування вимикача початкового положення не може збігатися з початковим положенням, інакше ви отримаєте помилку спільного обмеження.
- «Швидкість пошуку вихідного положення» — швидкість, яка використовується під час пошуку вихідного положення перемикача. Якщо перемикач знаходиться поблизу кінця ходу, цю швидкість необхідно вибрати таким чином, щоб вісь могла сповільнитися до повної зупинки до досягнення кінця ходу. Якщо перемикач закритий тільки на короткій ділянці ходу (а не від точки спрацьовування до кінця ходу), цю швидкість потрібно вибрати так, щоб вісь могла сповільнитися до зупинки перед тим, як перемикач знову відкриється, а повернення в початкове положення завжди потрібно починати з того самого боку перемикача. Якщо на початку процедури повернення в початкове положення верстат рухається в неправильному напрямку, змініть значення «Швидкість пошуку початкового положення» на протилежне.
- «Напрямок повернення в початкове положення» — виберіть «Те саме», щоб вісь відійшла від вимикача, а потім знову наблизилася до нього з дуже низькою швидкістю. При другому замиканні вимикача встановлюється початкове положення. Виберіть «Протилежне», щоб вісь відійшла від вимикача, а при розмиканні вимикача встановлювалося початкове положення.
- «Час розгону до максимальної швидкості» – час досягнення максимальної швидкості, розрахований на основі «Максимального прискорення» та «Максимальної швидкості».
- «Відстань для розгону до максимальної швидкості» – Відстань для досягнення максимальної швидкості з місця.
- «Частота імпульсів при максимальній швидкості» — інформація, обчислена на основі значень, введених вище. Найбільша «Частота імпульсів при максимальній швидкості» визначає «BASE_PERIO Значення вище 20000 Гц можуть призвести до уповільнення часу відгуку або навіть до зависання (найшвидша частота імпульсів, яку можна використовувати, варіюється від комп'ютера до комп'ютера.)
- «Axis SCALE» – число, яке буде використано в налаштуванні INI-файлу [SCALE]. Це кількість кроків на одиницю виміру користувача.
- «Перевірити цю вісь» – відкриється вікно для тестування кожної осі. Цим можна скористатися після заповнення всієї інформації для цієї осі.

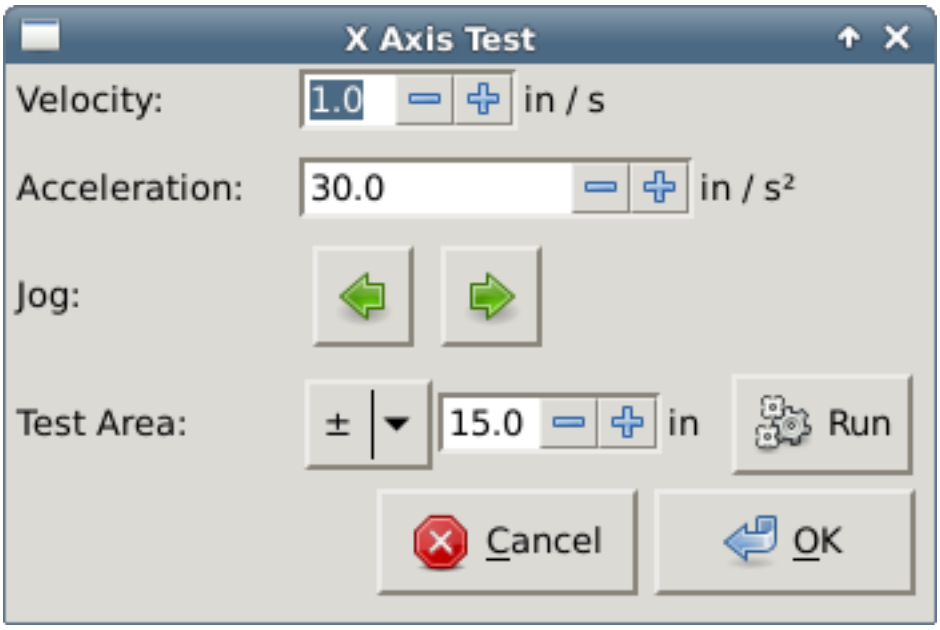


Figure 3.7: Тест осі

«Тестування цієї осі» – це базовий тестер, який виводить лише сигнали кроку та напрямку, щоб випробувати різні значення прискорення та швидкості.

Important
Щоб використовувати цю вісь для тестування, необхідно вручну ввімкнути її, якщо це потрібно. Якщо ваш драйвер має зарядний насос, його необхідно обійти. Ця вісь для тестування не реагує на сигнали кінцевих вимикачів. Використовуйте її з обережністю.

3.1.6.1 Знаходження максимальної швидкості

Почніть з низького прискорення (наприклад, **2 дюйма/с²** або **50 мм/с²**) та швидкість, яку ви хочете досягти. За допомогою відповідних кнопок перемістіть вісь до центру ходу. Будьте обережні, оскільки при низькому значенні прискорення вісь може пройти несподівано велику відстань, перш ніж зупинитися.

Після вимірювання доступної відстані руху введіть безпечну відстань у полі «Тестова зона», маючи на увазі, що після зупинки двигун може почати рухатися в несподіваному напрямку. Потім натисніть «Запустити». Машина почне рухатися вперед і назад вздовж цієї осі. У цьому тесті важливо, щоб комбінація прискорення та тестової області дозволяла машині досягти вибраної швидкості та «круїзувати» принаймні на невелику відстань — чим більша відстань, тим кращий результат тесту. Формула **d = 0,5 * v * v/a** дає мінімальну відстань, необхідну для досягнення заданої швидкості з заданим прискоренням. Якщо це зручно і безпечно, натисніть на стіл проти напрямку руху, щоб імітувати сили різання. Якщо машина зупинилася, зменшіть швидкість і почніть тест знову.

Якщо машина явно не зупинилася, натисніть кнопку «Запустити». Вісь повернеться у вихідне положення. Якщо положення неправильне, це означає, що вісь зупинилася або втратила кроки під час тесту. Зменште швидкість і повторіть тест.

Якщо машина не рухається, зупиняється або втрачає кроки, незалежно від того, наскільки низько ви зменшуєте швидкість, перевірте наступне:

- Правильні таймінги ступінчастої форми хвилі
- Правильна розпіновка, включаючи «Інвертувати» на ступінчастих контактах
- Правильне, добре екрановане кабеля
- Фізичні проблеми з двигуном, муфтою двигуна, ходовим гвинтом тощо.

Щойно ви знайдете швидкість, за якої вісь не зупиняється та не втрачає кроків під час цієї процедури тестування, зменште її на 10% та використовуйте це значення як «Максимальну швидкість» осі.

3.1.6.2 Знаходження максимального прискорення

Використовуючи максимальну швидкість, яку ви визначили на попередньому кроці, введіть значення прискорення для тестування. Використовуючи ту саму процедуру, що й вище, відрегулюйте значення прискорення вгору або вниз за необхідності. У цьому тесті важливо, щоб комбінація прискорення та тестової області дозволяла машині досягти вибраної швидкості. Після того, як ви визначили значення, при якому вісь не зупиняється і не втрачає кроки під час цієї процедури тестування, зменште його на 10% і використовуйте це значення як максимальне прискорення осі.

3.1.7 Конфігурація шпинделя

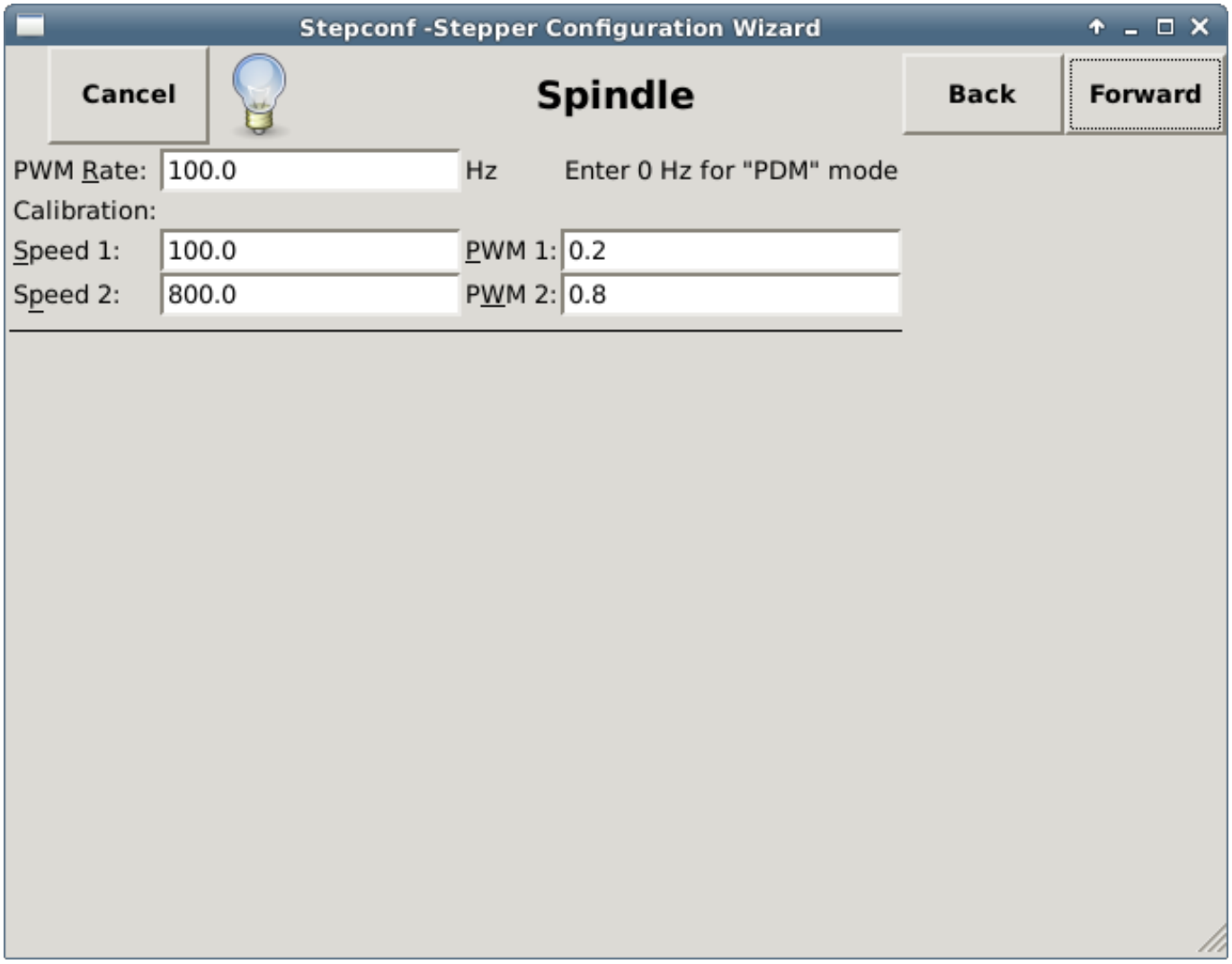


Figure 3.8: Сторінка конфігурації шпинделя

Ця сторінка відображається лише тоді, коли на сторінці «Розподіл виводів паралельного порту» для одного з виходів вибрано «PWM шпинделя».

3.1.7.1 Контроль швидкості шпинделя

Якщо на розпиновці відображається «Spindle PWM», слід ввести таку інформацію:

- «Частота PWM» — «несуча частота» сигналу ШІМ, що подається на шпиндель. Введіть «0» для режиму PDM, який корисний для генерації аналогової напруги керування. Відповідне значення див. у документації до контролера шпинделя.
- «Швидкість 1 і 2, PWM 1 і 2» — у створеному файлі конфігурації використовується проста лінійна залежність для визначення значення PWM для заданого значення RPM. Якщо значення невідомі, їх можна визначити. Докладнішу інформацію див. у розділі «Визначення калібрування шпинделя».

3.1.7.2 Синхронізований зі шпинделем рух

Коли відповідні сигнали від шпиндельного енкодера підключені до LinuxCNC через HAL, LinuxCNC підтримує нарізання різьби на токарному верстаті. Ці сигнали:

- «Індекс шпинделя» – це імпульс, який виникає один раз за оберт шпинделя.
- «Фаза шпинделя A» – це імпульс, який виникає в кількох рівномірно розташованих місцях під час обертання шпинделя.
- «Фаза шпинделя B (опціонально)» — це другий імпульс, який виникає, але з відхиленням від фази шпинделя A. Перевагами використання як A, так і B є визначення напрямку, підвищена стійкість до перешкод та підвищена роздільна здатність.

Якщо на розпиновці відображаються «Фаза шпинделя A» та «Індекс шпинделя», слід ввести таку інформацію:

- «Використовувати шпиндель на заданій швидкості» – за допомогою зворотного зв’язку від енкодера можна налаштувати LinuxCNC так, щоб він чекав, поки шпиндель досягне заданої швидкості, перш ніж розпочне подачу. Виберіть цю опцію та встановіть шкалу «достатньо близько».
- «Коефіцієнт підсилення фільтра відображення швидкості» – налаштування для регулювання стабільності візуального відображення швидкості шпинделя.
- «Циклів на оберт» – кількість циклів сигналу «Шпиндель A» протягом одного оберту шпинделя. Ця опція активується лише тоді, коли вхід встановлено на «Фаза шпинделя A»
- «Максимальна швидкість у різьбі» – максимальна швидкість шпинделя, що використовується під час нарізання різьби. Для високих обертів шпинделя або енкодера шпинделя з високою роздільною здатністю потрібне низьке значення «BASE_PERIOD».

3.1.7.3 Визначення калібрування шпинделя

Введіть такі значення на сторінці конфігурації шпинделя:

Швидкість 1:	0	**PWM 1:**	0
Швидкість 2:	1000	**PWM 2:**	1

Завершіть решту кроків процесу конфігурації, а потім запустіть LinuxCNC із вашою конфігурацією. Увімкніть верстат і виберіть вкладку MDI. Запустіть обертання шпинделя, ввівши: «M3 S100». Змініть швидкість шпинделя, ввівши інше число S: «S800». Дійсні числа (на даний момент) знаходяться в діапазоні від 1 до 1000.

Для двох різних S-чисел виміряйте фактичну швидкість шпинделя в об/хв. Запишіть S-числа та фактичні швидкості шпинделя. Запустіть StepConf знову. У полі «Speed» (Швидкість) введіть виміряну швидкість, а в полі «PWM» (Широтно-імпульсна модуляція) введіть S-число, поділене на 1000.

Оскільки криві відгуку більшості шпиндельних драйверів дещо нелінійні, найкраще:

- Переконайтеся, що дві калібрувальні швидкості не надто близькі одна до одної за кількістю обертів за хвилину.
- Переконайтеся, що дві калібрувальні швидкості знаходяться в діапазоні швидкостей, які ви зазвичай використовуєте під час фрезерування.

Наприклад, якщо ваш шпиндель обертатиметься від 0 до 8000 об/хв, але ви зазвичай використовуєте швидкості від 400 об/хв (10%) до 4000 об/хв (100%), тоді знайдіть значення PWM, які дають 1600 об/хв (40%) та 2800 об/хв (70%).

3.1.8 Опції

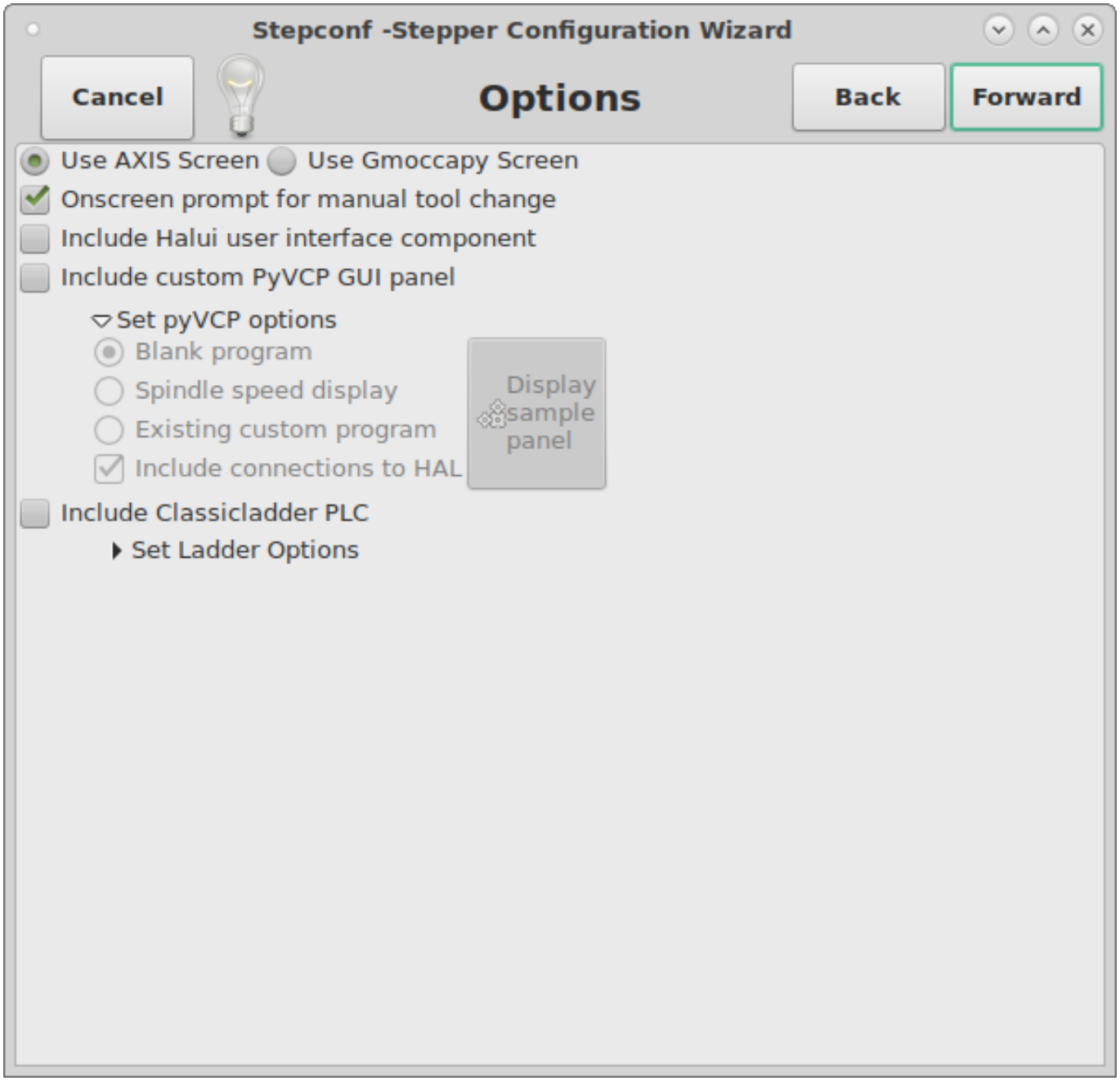


Figure 3.9: Налаштування розширених параметрів

- «Включити Halui» – це додасть компонент інтерфейсу користувача Halui. Див. розділ HALUI для отримання додаткової інформації.
- «Включити PyVCP» – ця опція додає базовий файл панелі PyVCP або файл-зразок для роботи. Див. розділ PyVCP для отримання додаткової інформації.
- «Включити ПЛК ClassicLadder» – ця опція додасть ПЛК ClassicLadder (програмований логічний контролер). Див. розділ ClassicLadder для отримання додаткової інформації.

- «На екрані з’являється запит на зміну інструменту» — якщо цей прапорець встановлено, LinuxCNC зупиниться і запропонує вам змінити інструмент, коли зустріне «M6». Ця функція зазвичай корисна тільки в тому випадку, якщо у вас є інструменти з попередньо встановленими параметрами.

3.1.9 Повна конфігурація машини

Натисніть кнопку «Застосувати», щоб записати файли конфігурації. Пізніше ви можете повторно запустити цю програму та налаштувати параметри, введені раніше.

3.1.10 Axis Подорожі та будинки

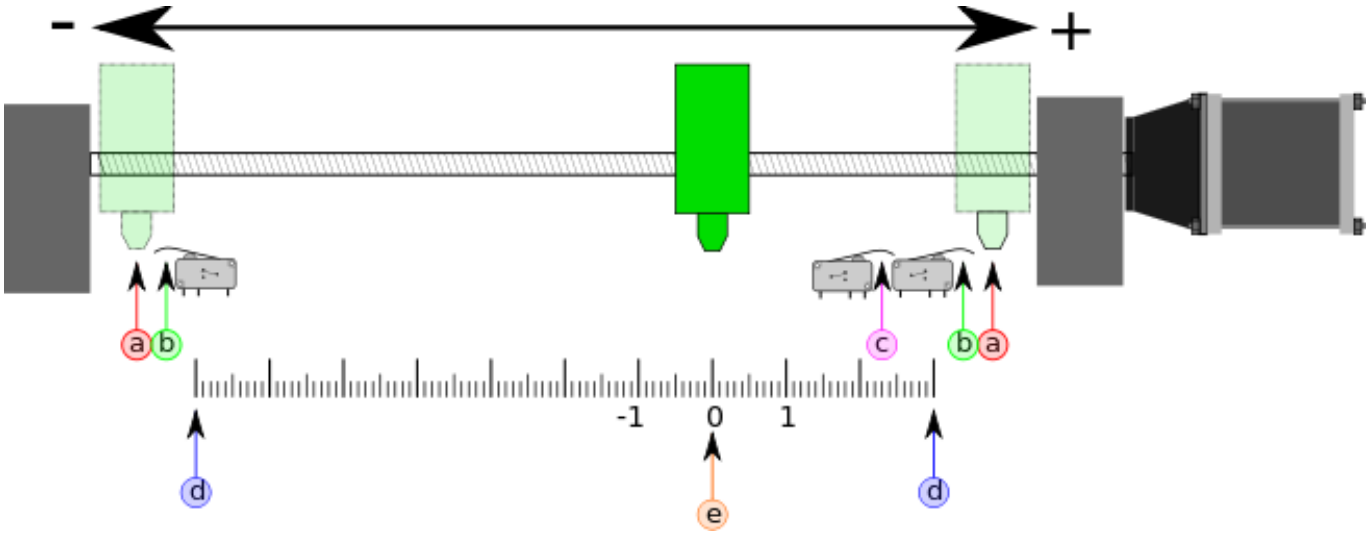


Figure 3.10: Axis Подорожі та дім

Для кожної осі існує обмежений діапазон ходу. Фізичний кінець ходу називається «жорсткою зупинкою».

Warning
Якщо буде перевищено механічну жорстку зупинку, гвинт або рама машини будуть пошкоджені!

Перед «жорсткою зупинкою» знаходиться «кінцевий вимикач». Якщо кінцевий вимикач спрацьовує під час нормальної роботи, LinuxCNC вимикає підсилювач двигуна. Відстань між «жорсткою зупинкою» і «кінцевим вимикачем» повинна бути достатньою, щоб двигун, що не живиться, міг зупинитися.

Перед «кінцевим вимикачем» є «м’який ліміт». Це ліміт, що застосовується в програмному забезпеченні після повернення в початкове положення. Якщо команда MDI або програма G-коду перевищує м’який ліміт, вона не виконується. Якщо імпульс перевищує м’який ліміт, він припиняється на рівні м’якого ліміту.

«Перемикач початкового положення» можна розмістити в будь-якому місці ходу (між жорсткими упорами). Якщо зовнішнє обладнання не вимикає підсилювачі двигуна при досягненні кінцевого вимикача, один з кінцевих вимикачів можна використовувати як перемикач початкового положення.

«Нульове положення» — це точка на осі, яка має значення 0 у системі координат верстата. Зазвичай «нульове положення» знаходиться в межах «м’яких обмежень». На токарних верстатах режим постійної швидкості поверхні вимагає, щоб «X=0» верстата відповідало центру обертання шпинделя, коли зміщення інструменту не діє.

«Початкове положення» — це місце в межах ходу, до якого вісь буде переміщена в кінці послідовності повернення в початкове положення. Це значення повинно бути в межах «м’яких обмежень». Зокрема, «початкове положення» ніколи не повинно бути рівним «м’якому обмеженню».

3.1.10.1 Робота без кінцевих вимикачів

Машина може працювати без кінцевих вимикачів. У цьому випадку лише м’які обмежувачі зупиняють машину від досягнення жорсткого упору. М’які обмежувачі працюють лише після того, як машина переведена в початкове положення.

3.1.10.2 Робота без домашніх перемикачів

Машина може працювати без перемикачів початкового положення. Якщо машина має кінцеві вимикачі, але не має перемикачів початкового положення, найкраще використовувати кінцевий вимикач як перемикач початкового положення (наприклад, вибрати «Мінімальний ліміт + Початкове положення X» у роз’ємі). Якщо машина взагалі не має перемикачів або кінцеві вимикачі не можуть бути використані як перемикачі початкового положення з іншої причини, то машина повинна бути встановлена «на око» або за допомогою відповідних міток. Повернення в початкове положення на око не є таким повторюваним, як повернення в початкове положення за допомогою вимикачів, але все одно дозволяє використовувати м’які обмеження.

3.1.10.3 Варіанти підключення домашнього та кінцевого вимикача

Ідеальним варіантом підключення зовнішніх перемикачів було б одне вхідне з’єднання на кожен перемикач. Однак паралельний порт ПК має лише 5 вхідних з’єднань, тоді як на 3-осьовій машині встановлено аж 9 перемикачів. Замість цього кілька перемикачів підключаються між собою різними способами, щоб зменшити кількість необхідних вхідних з’єднань.

На малюнках нижче показано загальну схему підключення декількох перемикачів до одного вхідного контакту. У кожному випадку, коли один перемикач спрацьовує, значення, яке відображаєтьс
на INPUT, змінюється з логічного HIGH на LOW. Однак LinuxCNC очікує значення TRUE, коли перемикач закритий, тому на сторінці конфігурації роз’ємів необхідно встановити відповідний прапорець «Invert» (Інвертувати). Резистор підтягування, показаний на схемах, підтягує вхід до високого рівня, поки не буде встановлено з’єднання із землею, після чого вхід переходить у низький рівень. В іншому випадку вхід може коливатися між увімкненим і вимкненим станом, коли ланцюг розімкнутий. Зазвичай для паралельного порту можна використовувати 47 кОм.

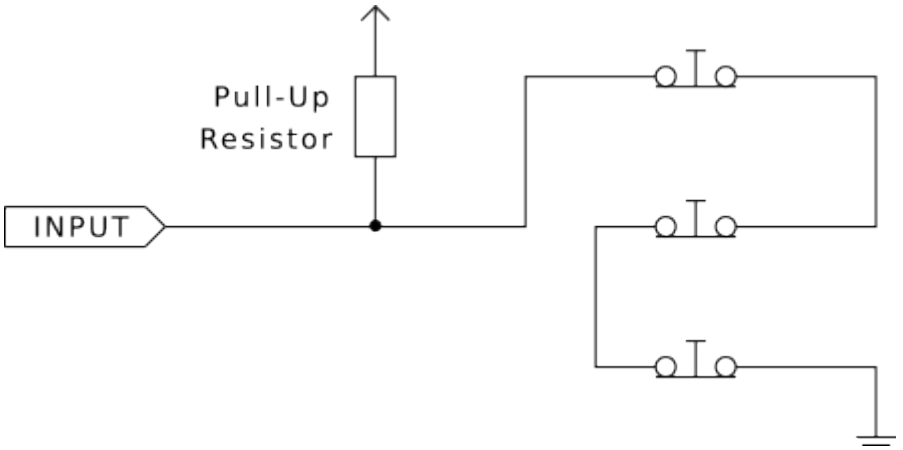


Figure 3.11: Послідовне підключення нормально замкнутих вимикачів (НЗ) (спрощена схема)

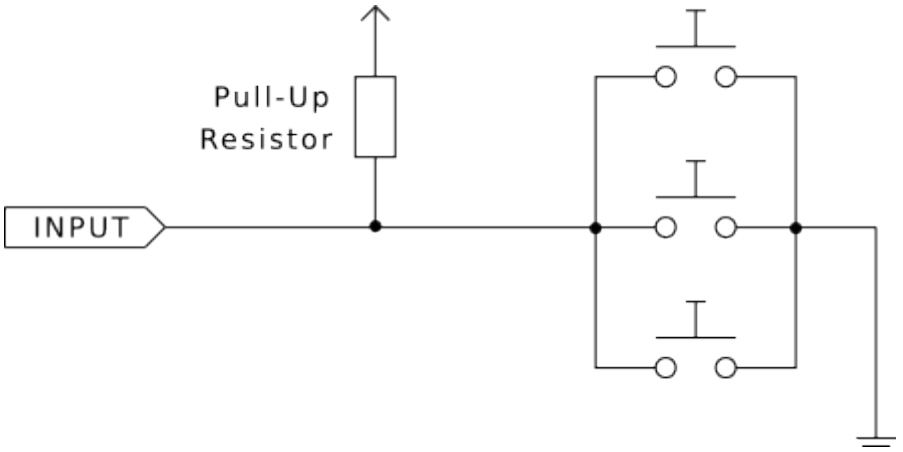


Figure 3.12: Паралельне підключення нормально розімкнутих перемикачів (Н/О) (спрощена схема)

У StepConf дозволені такі комбінації перемикачів:

- Об’єднайте домашні перемикачі для всіх осей
- Об’єднані кінцеві вимикачі для всіх осей
- Об’єднайте обидва кінцеві вимикачі для однієї осі
- Поєднайте обидва кінцеві вимикачі та вимикач дому для однієї осі
- Поєднайте один кінцевий вимикач та вимикач дому для однієї осі

Останні дві комбінації також доречні, коли використовується тип contact home.

3.2 Майстер налаштування Mesa

PnCconf створено для допомоги у створенні конфігурацій, що використовують специфічні продукти Mesa «Anything I/O».

Він може налаштовувати сервосистеми з замкнутим циклом або апаратні крокові системи. Він використовує подібний підхід «майстра», як і StepConf (використовується для програмного крокування систем, керованих паралельним портом).

PnCconf все ще перебуває на стадії розробки (бета-версія), тому є деякі помилки та відсутні функції. Будь ласка, повідомляйте про помилки та пропозиції на сторінці форуму LinuxCNC або у списку розсилки.

Існує два напрямки мислення при використанні PnCconf:

Один із них полягає у використанні PnCconf для постійної конфігурації вашої системи — якщо ви вирішите змінити параметри, перезавантажте PnCconf і дозвольте йому налаштувати нові параметри. Це буде добре працювати, якщо ваша машина є досить стандартною і ви можете використовувати власні файли для додавання нестандартних функцій. PnCconf намагається співпрацю з вами в цьому питанні.

Інший спосіб полягає у використанні PnCconf для створення конфігурації, яка наближена до бажаної, а потім вручну редагувати все, щоб пристосувати її до своїх потреб. Це буде правильним вибором, якщо вам потрібні значні модифікації, що виходять за межі можливостей PnCconf, або ви просто хочете попрацювати з LinuxCNC та дізнатися про нього більше.

Ви можете переміщатися сторінками майстра за допомогою кнопок «Вперед», «Назад» та «Скасувати». Також є кнопка довідки, яка надає деяку довідкову інформацію про сторінки, діаграми та сторінку виводу.

Tip
Сторінка довідки PnCconf повинна містити найактуальнішу інформацію та додаткові деталі.

3.2.1 Покрокові інструкції

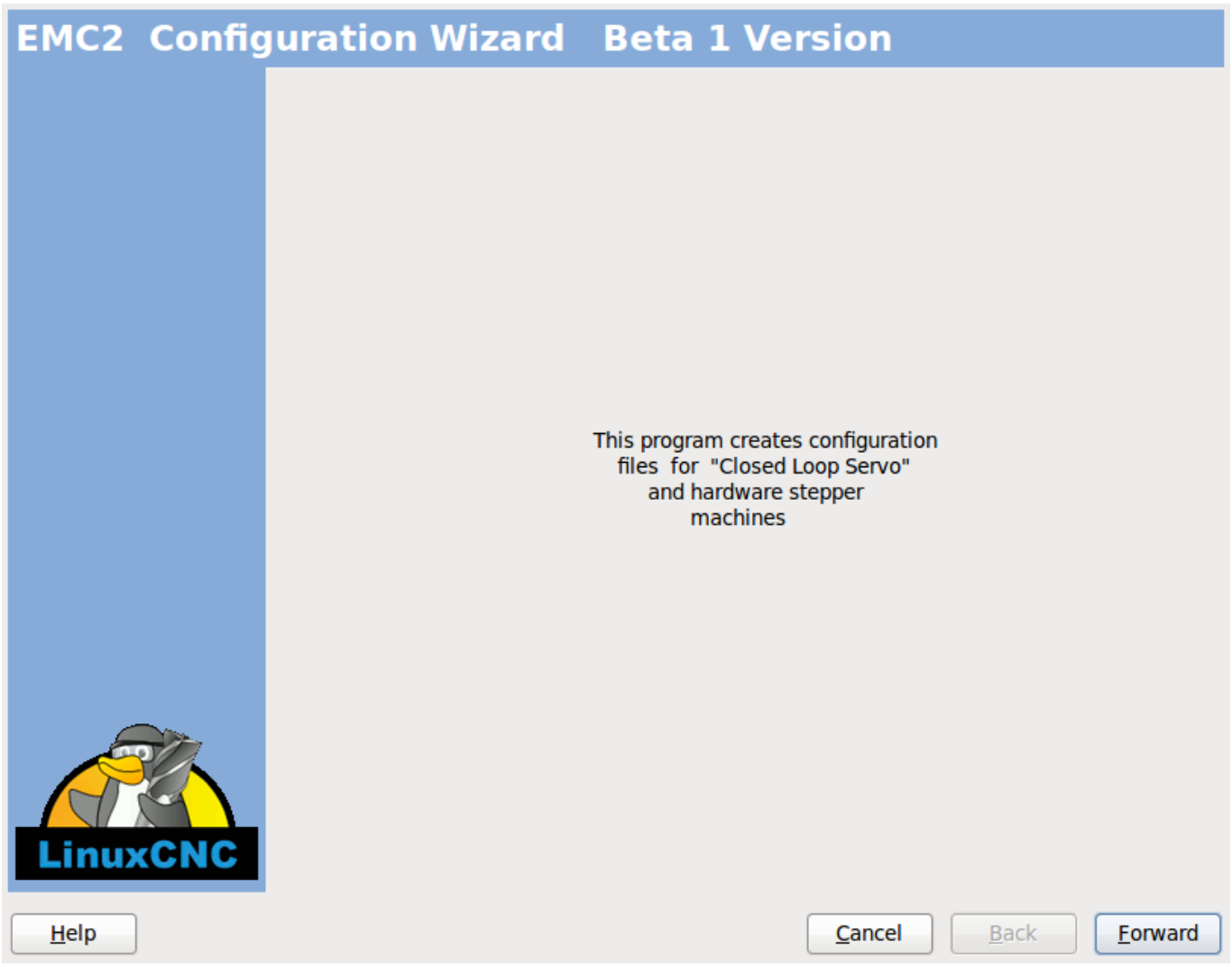


Figure 3.13: PnCconf Splash

3.2.2 Створити або редагувати

Це дозволяє вибрати раніше збережену конфігурацію або створити нову. Якщо ви виберете «Змінити конфігурацію» і натиснете «Далі», з'явиться вікно вибору файлу. PnCconf попередньо вибирає останній збережений файл. Виберіть конфігурацію, яку ви хочете редагувати. Якщо ви внесли будь-які зміни в основні файли HAL або INI, **PnCconf перезапише** ці файли, і ці зміни будуть втрачені. Деякі файли не будуть перезаписані, і PnCconf розмістить у них відповідну примітку. Це також дозволяє вибрати опції ярлика на робочому столі / запуску. Ярлик на робочому столі розмістить на робочому столі піктограму папки, яка вказує на ваші нові файли конфігурації. В іншому випадку вам доведеться шукати їх у вашій домашній папці в linuxcnc/configs.

Програма запуску з робочого столу додасть на робочий стіл піктограму для безпосереднього запуску вашої конфігурації. Ви також можете запустити її з головного меню, використовуючи селектор конфігурації «LinuxCNC», який знаходиться в меню CNC, і вибравши назву вашої конфігурації

3.2.3 Основна інформація про машину

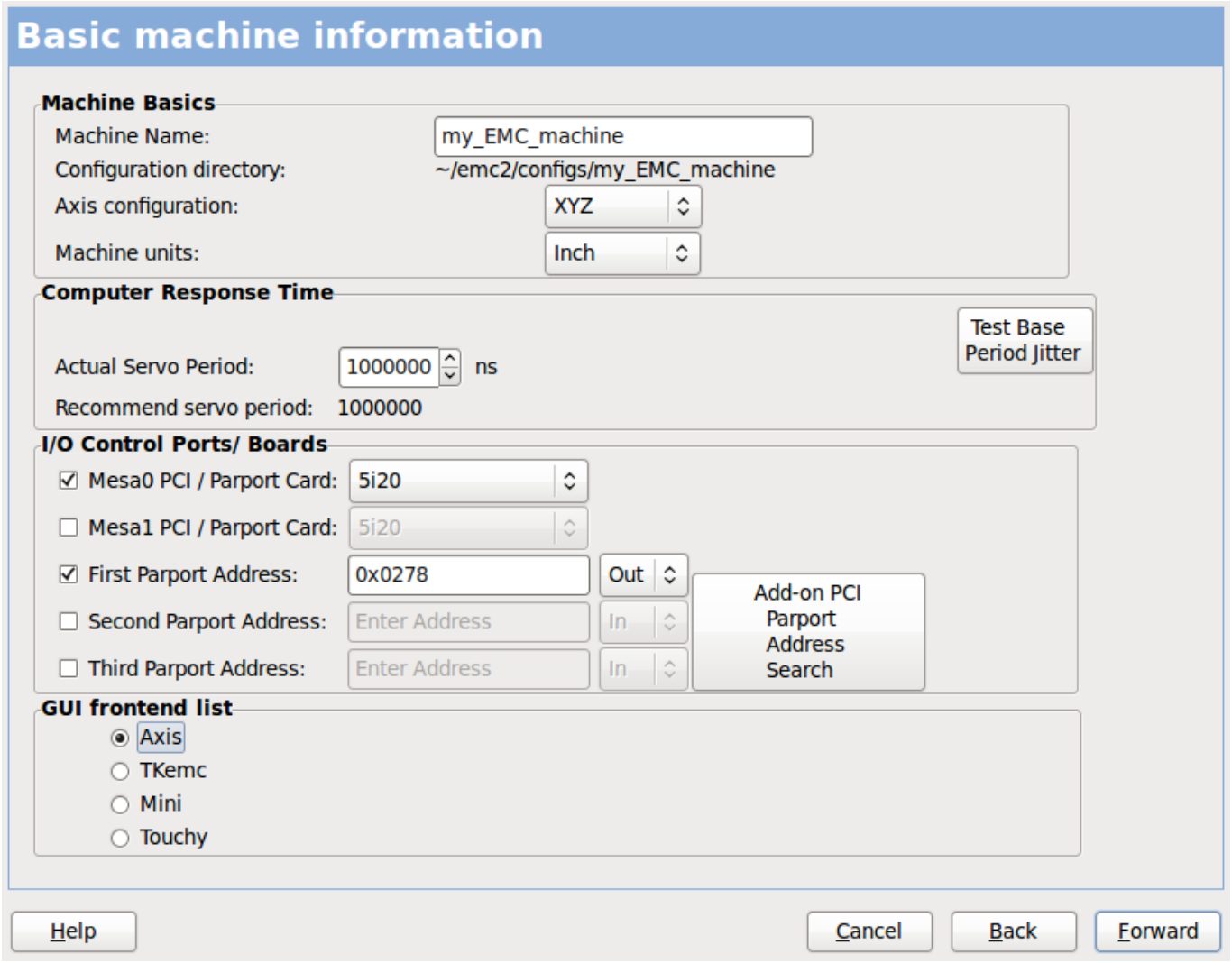


Figure 3.14: PnCconf Basic

Основи машин

Якщо ви використовуєте назву з пробілами, PnCconf замінить пробіли на символи підкреслення (як правило, Linux не любить пробіли в назвах). Вибір конфігурації осі визначає тип машини, яку ви будуєте, та доступні осі. Селектор ”Одиниці вимірювання машини” дозволяє вводити дані в метричних або імперських одиницях на наступних кроках процесу конфігурації.

Tip

Значення за замовчуванням не конвертуються під час використання метрики, тому переконайтеся, що це правильні значення!

Час відгуку комп’ютера

Період сервоприводу задає серцебиття системи. Затримка описує різницю між часом, коли

система має виконати дію, і часом, коли вона фактично її виконує. Як і залізниця, LinuxCNC вимагає, щоб усе відбувалося в дуже стислі та послідовні часові рамки, інакше трапляються погані речі. LinuxCNC вимагає та використовує операційну систему «реального часу», що просто означає, що вона має низький час відгуку (затримку). Коли LinuxCNC вимагає та виконує обчислення, його не можуть переривати запити з нижчим пріоритетом (такі як введення користувачем кнопок екрана або малювання тощо).

Тестування затримки є критично важливим і ключовим моментом для перевірки, перш ніж продовжувати. Будь ласка, дотримуйтесь інструкцій на сторінці Тест затримки, перш ніж продовжуват

Тепер нас влаштовує затримка, і ми повинні вибрати період сервоприводу. У більшості випадків період сервоприводу 1000000 нс є прийнятним (це забезпечує швидкість обчислень сервоприводу 1 кГц – 1000 обчислень за секунду). Якщо ви створюєте сервосистему із замкнутим циклом, яка керує крутним моментом (струмом), а не швидкістю (напругою), то краща швидкість була б кращою – щось на кшталт 200000 (швидкість обчислень 5 кГц). Проблема зі зниженням швидкості сервоприводу полягає в тому, що комп'ютеру залишається менше часу для виконання інших завдань, окрім обчислень LinuxCNC. Зазвичай дисплей (графічний інтерфейс користувача) стає менш чутливим. Ви повинні визначитися з балансом. Майте на увазі, що якщо ви налаштуєте свою сервосистему із замкнутим циклом, а потім зміните період сервоприводу, вам, ймовірно, доведеться налаштувати їх знову.

Порти/плати керування вводом/виводом
PnCconf здатний налаштовувати машини, що мають до двох плат Mesa та три паралельні порти. Паралельні порти можна використовувати лише для простого низькошвидкісного (сервошвидкісний) вводу/виводу.

Mesa
Ви повинні вибрати принаймні одну плату Mesa, оскільки PnCconf не налаштує паралельні порти для підрахунку енкодерів або виведення крокових або PWM-сигналів. Карти Mesa, доступні у вікні вибору, базуються на тому, що PnCconf знаходить для прошивки в системах. Існують опції для додавання власних прошивок та/або «чорного списку» (ігнорування) деяких прошивок або плат за допомогою файлу налаштувань. Якщо прошивка не знайдена, PnC-conf покаже попередження і використає внутрішню зразкову прошивку - тестування буде неможливим. Слід зауважити, що якщо ви виберете дві PCI-карти Mesa, то наразі немає можливості передбачити, яка карта є 0, а яка 1 - ви повинні протестувати - переміщення карт може змінити їх порядок. Якщо ви налаштовуєте дві карти, обидві карти повинні бути встановлені, щоб тести працювали.

Паралельний порт
До 3 паралельних портів (так званих parports) можуть використовуватися як прості входи/виходи. Ви повинні встановити адресу parport. Ви можете ввести систему нумерації паралельних портів Linux (0, 1 або 2) або ввести фактичну адресу. Адреса вбудованого parport часто становить 0x0278 або 0x0378 (записана в шістнадцятковій системі числення), але її можна знайти на сторінці BIOS. Сторінка BIOS відкривається під час першого запуску комп'ютера, для цього потрібно натиснути клавішу (наприклад, F2). На сторінці BIOS ви можете знайти адресу паралельного порту і встановити режим, такий як SPP, EPP тощо. На деяких комп'ютерах ця інформація відображається протягом декількох секунд під час запуску. Для PCI-карт паралельного порту адресу можна знайти, натиснувши кнопку «пошук адреси паралельного порту». Це відкриє сторінку довідки зі списком усіх PCI-пристроїв, які можна знайти. Там має бути посилання на пристрій паралельного порту зі списком адрес. Одна з цих адрес має працювати. Не всі паралельні порти PCI працюють належним чином. Обидва типи можна вибрати як «in» (максимальна кількість вхідних контактів) або «out» (максимальна кількість вихідних контактів).

Список інтерфейсу графічного інтерфейсу
Це визначає графічні екрани відображення, які використовуватиме LinuxCNC. Кожен з них має різні опції.

AXIS

- повністю підтримує токарні верстати.
- є найбільш розвиненим та використовуваним фронтендом
- призначений для використання з мишею та клавіатурою
- базується на Tkinter, тому природно інтегрується з PyVCP (віртуальні панелі керування на основі Python).
- має вікно 3D-графіки.
- дозволяє інтегрувати VCP збоку або в центральний виступ

'TkLinuxCNC

- яскраво-синій екран з високою контрастністю
- окреме графічне вікно
- без інтеграції VCP

Доторкливий

- Touchy був розроблений для використання з сенсорним екраном, деякими мінімальними фізичними перемикачами та колесом MPG.
- вимагає сигналів і кнопок запуску, переривання та покрокового виконання циклу
- Також потрібно вибрати штовхання MPG для спільної осі.
- базується на GTK, тому природно інтегрується з GladeVCP (віртуальними панелями керування).
- дозволяє інтегрувати панелі VCP у центральний вкладний блок
- не має графічного вікна
- зовнішній вигляд можна змінити за допомогою власних тем

QtPlasmaC

- Повнофункціональна конфігурація Plasmac на основі інфраструктури QtVCP.
- керування мишею/клавіатурою або керування сенсорним екраном
- без інтеграції VCP

3.2.4 Зовнішня конфігурація

Ця сторінка дозволяє вибрати зовнішні елементи керування, такі як поштовховий рух або перевизначе

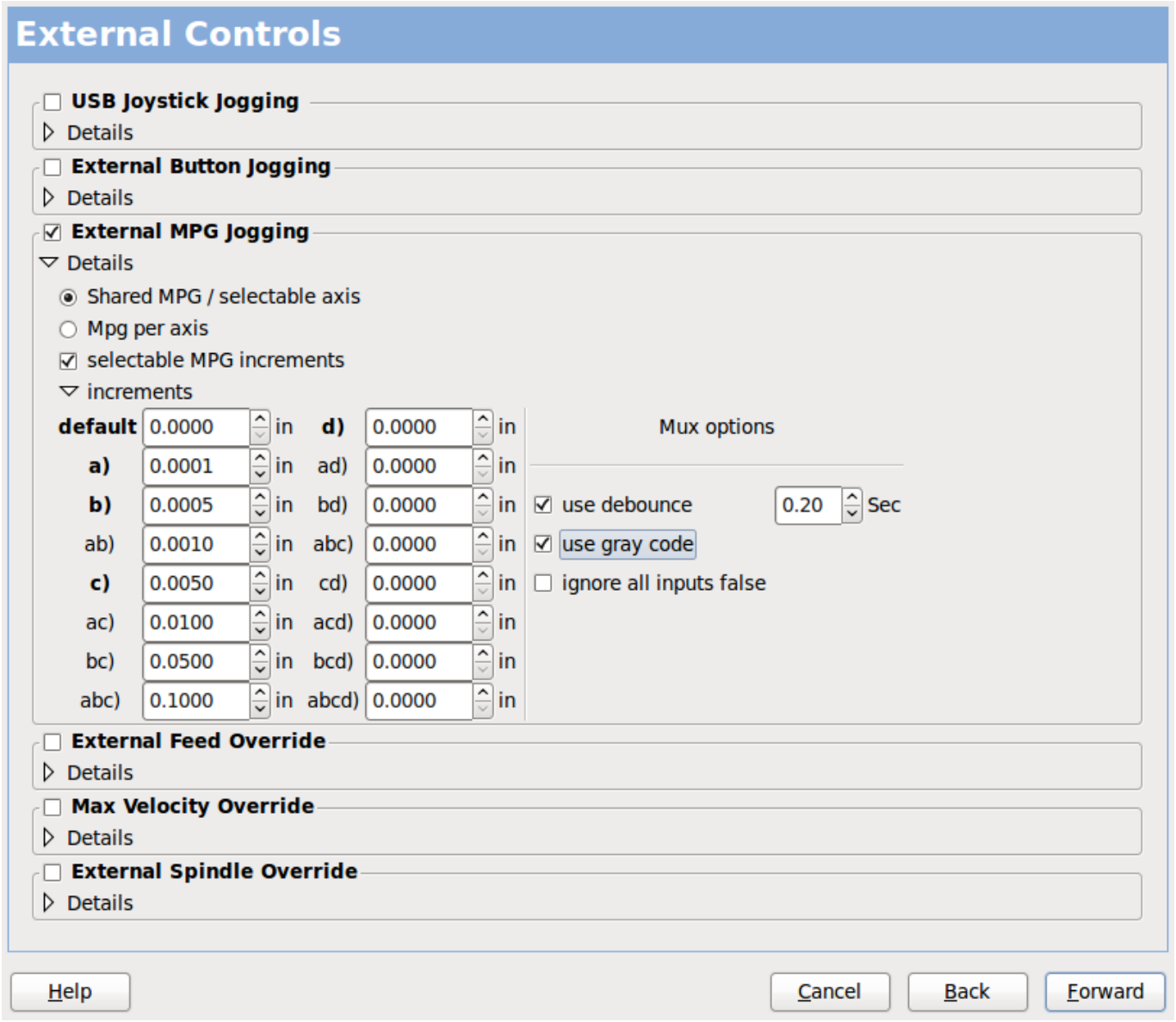


Figure 3.15: Зовнішні засоби контролю

Якщо ви виберете джойстик для ручного керування, він повинен бути постійно підключений, щоб LinuxCNC міг його завантажити. Щоб використовувати аналогові джойстики для зручного ручного керування, вам, ймовірно, доведеться додати спеціальний код HAL. Для ручного керування MPG необхідний генератор імпульсів, підключений до лічильника кодера MESA. Для управління перезаписом можна використовувати імпульсний генератор (MPG) або перемикачі (наприклад, поворотний перемикач). Зовнішні кнопки можна використовувати з джойстиком OEM на основі перемикача.

Джойстик для бігу

Потрібне встановлення в системі спеціального «правила пристрою». Це файл, який LinuxCNC використовує для підключення до списку пристроїв Linux. PnCconf допоможе підготувати цей файл.

- «Пошук правила пристрою» виконає пошук правил у системі, ви можете використовувати це, щоб знайти назви пристроїв, які ви вже створили за допомогою PnCconf.

- «Додати правило для пристрою» дозволить вам налаштувати новий пристрій, дотримуючись підказок. Вам знадобиться ваш пристрій.
- «Тестовий пристрій» дозволяє завантажити пристрій, переглянути назви його контактів та перевірити його функції за допомогою гальметра.

Переміщення джойстиком використовує компоненти HALUI та hal_input.

Зовнішні кнопки
: дозволяє переміщувати вісь за допомогою простих кнопок із заданою швидкістю переміщення. Ймовірно, найкраще підходить для швидкого переміщення.

MPG на галон
: Дозволяє використовувати ручний генератор імпульсів для штовхання осі верстата.

MPG часто зустрічаються на машинах комерційного класу. Вони видають квадратурні імпульси, які можна підрахувати за допомогою лічильника кодера MESA. PnCconf дозволяє використовувати MPG для кожної осі окремо або один MPG для всіх осей. Він дозволяє вибирати швидкість рухомого режиму за допомогою перемикачів або єдину швидкість.

Опція вибору приростів використовує компонент mux16. Цей компонент має такі опції, як усунення дребезгу та код Грея, для фільтрації необробленого вхідного сигналу перемикача.

Перевизначення
: PnCconf дозволяє змінювати швидкість подачі та/або швидкість шпинделя за допомогою генератора імпульсів (MPG) або перемикачів (наприклад, поворотних).

3.2.5 Конфігурація графічного інтерфейсу

Тут ви можете встановити значення за замовчуванням для екранів відображення, додати віртуальні панелі керування (VCP) та налаштувати деякі параметри LinuxCNC.

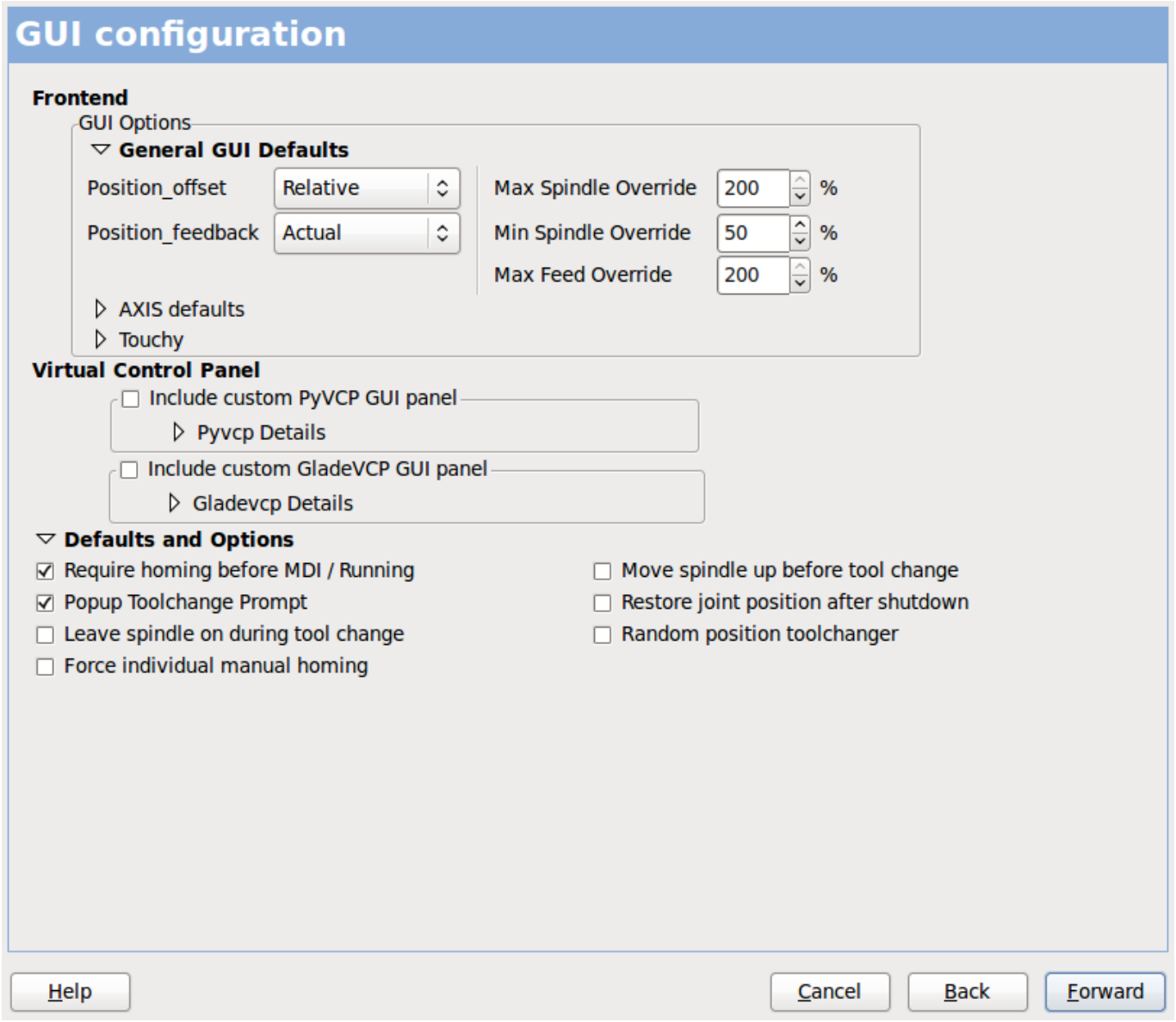


Figure 3.16: Конфігурація графічного інтерфейсу

Параметри графічного інтерфейсу користувача

Параметри за замовчуванням дозволяють вибрати загальні налаштування за замовчуванням для будь-якого екрана дисплея.

AXIS defaults — це опції, специфічні для AXIS. Якщо ви виберете опції розміру, положення або примусового максимізації, PnCconf запитає, чи можна перезаписати файл налаштувань (.axisrc). Якщо ви не додавали до цього файлу команди вручну, можна дозволити це. Опції положення та примусового максимізації можна використовувати для переміщення AXIS на другий монітор, якщо система це підтримує.

Налаштування Touchy — це опції, характерні для Touchy. Більшість опцій Touchy можна змінити під час роботи Touchy за допомогою сторінки налаштувань. Touchy використовує GTK для малювання екрану, а GTK підтримує теми. Теми контролюють основний вигляд і відчуття програми. Ви можете завантажити теми з Інтернету або редагувати їх самостійно. На комп'ютері є список поточних тем, з яких ви можете вибрати. Щоб виділити частину тексту, PnCconf дозволяє замінити

стандартні налаштування тем. Параметри position і force max можна використовувати для переміщенн Touchy на другий монітор, якщо система підтримує таку функцію.

Параметри QtPlasmaC є специфічними для QtPlasmac, будь-які загальні параметри, які не потрібні, будуть вимкнені. Якщо вибрано QtPlasmac, то наступний екран буде екраном налаштування кнопок користувача, який є специфічним для QtPlasmaC, і параметри VCP будуть недоступні.

Варіанти VCP
: Віртуальні панелі керування дозволяють додавати на екран власні елементи керування та відображення. AXIS і Touchy можуть інтегрувати ці елементи керування в екран у визначених місцях. Існує два типи VCP: PyVCP, який використовує «Tkinter» для малювання екрану, та GladeVCP, який використовує «GTK» для малювання екрану.

PyVCP
: XML-файл екрану PyVCP можна створити лише вручну. PyVCP природно вписуються в AXIS, оскільки обидва використовують TKinter.

Контакти HAL створені для підключення користувача до внутрішнього файлу HAL. Існує зразок панелі дисплея шпинделя, який користувач може використовувати як є або на основі якого можна створювати власні. Ви можете вибрати порожній файл, до якого згодом можна додати свої елементи керування, або вибрати зразок дисплея шпинделя, який відображатиме швидкість шпинделя та вказуватиме, чи шпиндель працює на заданій швидкості.

PnCconf підключить для вас відповідні контакти HAL для відображення шпинделя. Якщо ви використовуєте AXIS, панель буде інтегрована в правій частині. Якщо ви не використовуєте AXIS, панель буде відокремленою від переднього екрану.

Ви можете використовувати параметри геометрії для зміни розміру та переміщення панелі, наприклад, для переміщення її на другий екран, якщо система підтримує таку функцію. Якщо натиснути кнопку «Відобразити зразок панелі», параметри розміру та розміщення будуть збережені.

GladeVCP
: GladeVCP природно вписуються в сенсорний екран, оскільки обидва використовують GTK для малювання, але, змінивши тему GladeVCP, можна зробити так, щоб вони досить добре поєднувалися з AXIS (спробуйте Redmond).

Він використовує графічний редактор для створення XML-файлів. Піни HAL створюються для підключення користувача всередині його власного HAL-файлу.

GladeVCP також дозволяє набагато складнішу (і більш витончену) взаємодію з програмуванням, яку PnCconf наразі не використовує (див. GladeVCP у посібнику).

PnCconf має зразки панелей, які користувач може використовувати як є або на основі них. За допомогою GladeVCP PnCconf дозволить вам вибирати різні параметри на вашому зразку відображення.

У розділі «параметри зразка» виберіть потрібні. Кнопки нуля використовують команди HALUI, які ви можете редагувати пізніше в розділі HALUI.

Для автоматичного Z-відключення також потрібна класична програма відключення сходів та вибраний вхід зонда. Для цього потрібна контактна пластина для струмопровідного відключення та заземлений струмопровідний інструмент. Щоб дізнатися, як це працює, див:

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

У розділі «Параметри дисплея» розмір, положення та максимальну силу можна використовувати на «автономній» панелі для таких речей, як розміщення екрана на другому моніторі, якщо система це дозволяє.

Ви можете вибрати тему GTK, яка визначає основний вигляд і стиль панелі. Зазвичай бажано, щоб вона відповідала екрану інтерфейсу. Ці параметри будуть використані, якщо ви натиснете

кнопку «Відобразити зразок». За допомогою GladeVCP, залежно від екрану інтерфейсу, ви можете вибрати місце відображення панелі.

Ви можете примусово встановити його окремо, або за допомогою AXIS він може бути по центру або праворуч, а за допомогою Touchy він може бути по центру.

Налаштування за замовчуванням та параметри

- Вимагати повернення до початкового положення перед MDI / запуском
  - Якщо ви хочете мати можливість переміщати машину перед поверненням до початкового положення, зніміть цей прапорець.
- Спливаюче вікно інструмента
  - Виберіть між екранним підказкою для зміни інструменту або експортом назв стандартних сигналів для наданого користувачем HAL-файлу пристрою зміни інструментів
- Залиште шпиндель увімкненим під час зміни інструменту:
  - Використовується для токарних верстатів
- Примусове індивідуальне ручне самонаведення
- Перемістіть шпиндель вгору перед зміною інструменту
- Відновлення положення суглоба після вимкнення
  - Використовується для нетривіальних кінематичних машин
- Пристрої зміни інструментів у випадковому положенні
  - Використовується для пристроїв зміни інструменту, які не повертають інструмент у те саме гніздо. Вам потрібно буде додати спеціальний код HAL для підтримки пристроїв зміни інструменту.

3.2.6 Конфігурація Меси

Сторінки конфігурації Mesa дозволяють використовувати різні прошивки. На основній сторінці ви вибрали карту Mesa, тут ви вибираєте доступну прошивку і вибираєте, які і скільки компонентів доступні.

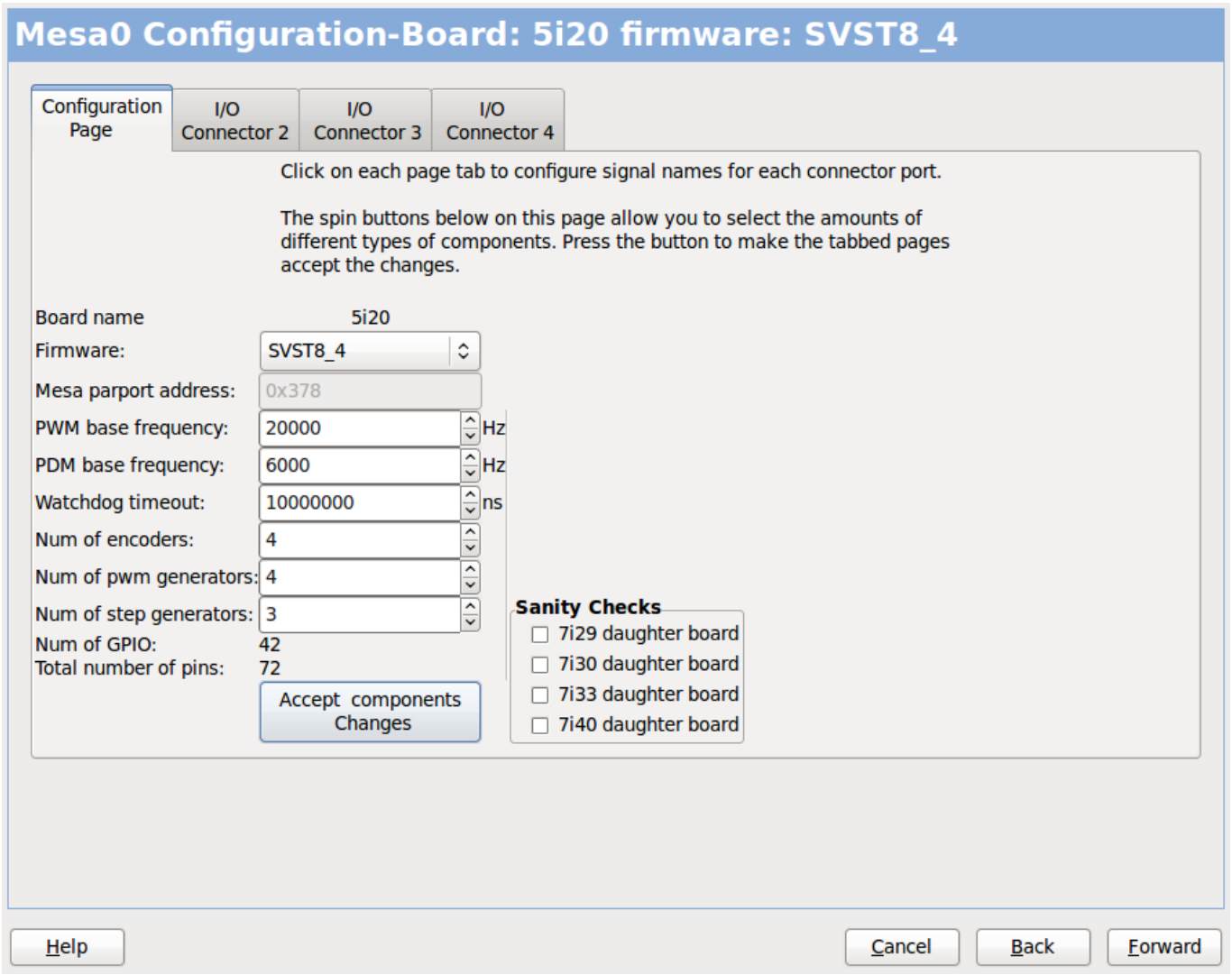


Figure 3.17: Конфігурація дошки Mesa

Адреса паралельного порту використовується тільки з картою Mesa parport, 7i43. Вбудований паралельний порт зазвичай використовує 0x278 або 0x378, хоча ви можете знайти адресу на сторінці BIOS. 7i43 вимагає, щоб паралельний порт використовував режим EPP, який також налаштовується на сторінці BIOS. Якщо ви використовуєте паралельний порт PCI, адресу можна знайти за допомогою кнопки пошуку на базовій сторінці.

Note
Багато PCI-карт не підтримують протокол EPP належним чином.

Базова частота PWM PDM та 3PWM встановлює баланс між пульсаціями та лінійністю. Якщо використовуються дочірні плати Mesa, документація до плати має містити рекомендації.

Important
Важливо дотримуватися цих правил, щоб уникнути пошкоджень та отримати найкращу продуктивність.

```
7i33 b''вb''b''иb''b''мb''b''ab''b''гb''b''ab''b''єb'' PDM b''ib'' b''6b''b''ab''b''зb''b' ↩
    'ob''b''вb''b''ob''b''ïb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''иb'' PDM 6 b' ↩
    'Mb''b''Гb''b''цb''
7i29 b''вb''b''иb''b''мb''b''ab''b''гb''b''ab''b''єb'' PWM b''ib'' b''6b''b''ab''b''зb''b' ↩
    'ob''b''вb''b''ob''b''ïb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''иb'' PWM 20 b' ↩
    'кb''b''Гb''b''цb''
7i30 b''вb''b''иb''b''мb''b''ab''b''гb''b''ab''b''єb'' PWM b''ib'' b''6b''b''ab''b''зb''b' ↩
    'ob''b''вb''b''ob''b''ïb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''иb'' PWM 20 b' ↩
    'кb''b''Гb''b''цb''
7i40 b''вb''b''иb''b''мb''b''ab''b''гb''b''ab''b''єb'' PWM b''ib'' b''6b''b''ab''b''зb''b' ↩
    'ob''b''вb''b''ob''b''ïb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''иb'' PWM 50 b' ↩
    'кb''b''Гb''b''цb''
7i48 b''вb''b''иb''b''мb''b''ab''b''гb''b''ab''b''єb'' UDM b''ib'' b''6b''b''ab''b''зb''b' ↩
    'ob''b''вb''b''ob''b''ïb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''иb'' PWM 24 b' ↩
    'кb''b''Гb''b''цb''
```

Тайм-аут сторожового таймера
: використовується для встановлення часу, протягом якого плата MESA буде чекати перед припиненням виводу даних у разі переривання зв’язку з комп’ютером. Зверніть увагу, що Mesa використовує виводи типу «активний низький», що означає: коли вивід увімкнено, він має низький рівень (приблизно 0 вольт), а коли вимкнено — високий (приблизно 5 вольт). Переконайтеся, що ваше обладнання є безпечним у вимкненому стані (при спрацьовуванні сторожового таймера).

Кількість кодерів/PWM-генераторів/КРОКОВИХ генераторів
: Ви можете вибрати кількість доступних компонентів, знявши позначку з невикористовуваних. Не всі типи компонентів доступні з усіма прошивками.

Вибір кількості компонентів, меншої за максимальну, дозволяє отримати більше GPIO-контактів. Якщо ви використовуєте дочірні плати, майте на увазі, що не можна відміняти вибір контактів, які використовує плата. Наприклад, деякі прошивки підтримують дві плати 7i33. Якщо у вас є тільки одна, ви можете відмінити вибір достатньої кількості компонентів, щоб використовувати роз’єм, який підтримував другу плату 7i33. Компоненти відміняються за номерами, починаючи з найвищого, без пропускання номерів. Якщо після цього компоненти не знаходяться там, де ви хочете, то вам доведеться використовувати інше програмне забезпечення. Програмне забезпечення визначає, де, що і в якій максимальній кількості знаходяться компоненти. Можливе використання власного програмного забезпечення, про що слід ввічливо запитати, зв’язавшись з розробниками LinuxCNC і Mesa. Використання власного програмного забезпечення в PnCconf вимагає спеціальних процедур і не завжди є можливим, хоча я намагаюся зробити PnCconf максимально гнучким.

Вибравши всі ці опції, натисніть кнопку «Прийняти зміни компонентів», і PnCconf оновить сторінки налаштувань вводу-виводу. Залежно від плати Mesa, для доступних роз’ємів будуть показані тільки вкладки вводу-виводу.

3.2.7 Налаштування вводу/виводу Mesa

Вкладки використовуються для налаштування вхідних та вихідних контактів плат Mesa. PnCconf дозволяє створювати власні назви сигналів для використання в користувацьких HAL-файлах.

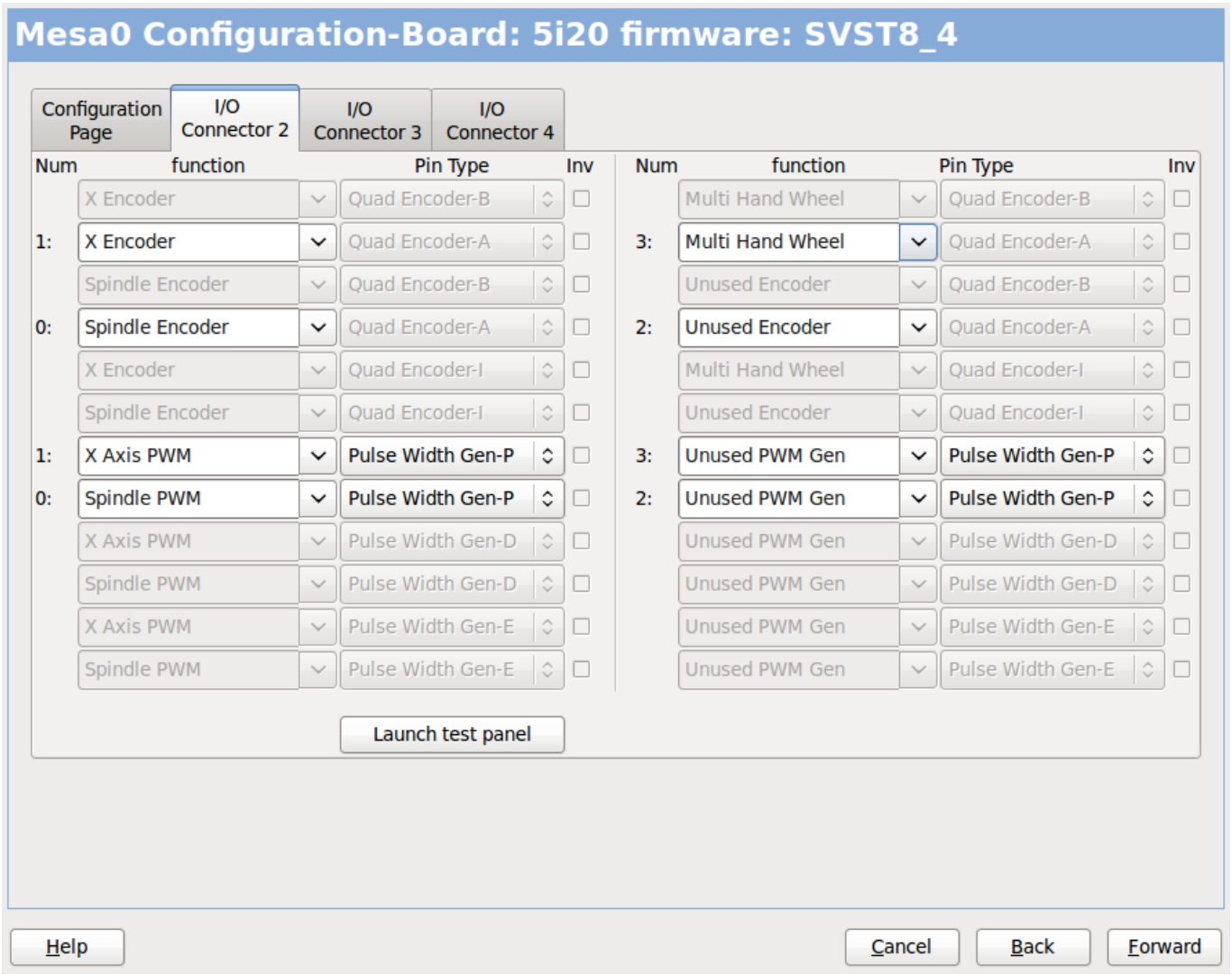


Figure 3.18: Налаштування C2 вводу/виводу Mesa

На цій вкладці з цією прошивкою компоненти налаштовані для дочірньої плати 7i33, яка зазвичай використовується з сервоприводами із замкнутим контуром. Зверніть увагу, що номери компонентів лічильників енкодера та драйверів PWM не розташовані в числовому порядку. Це відповідає вимогам дочірньої плати.

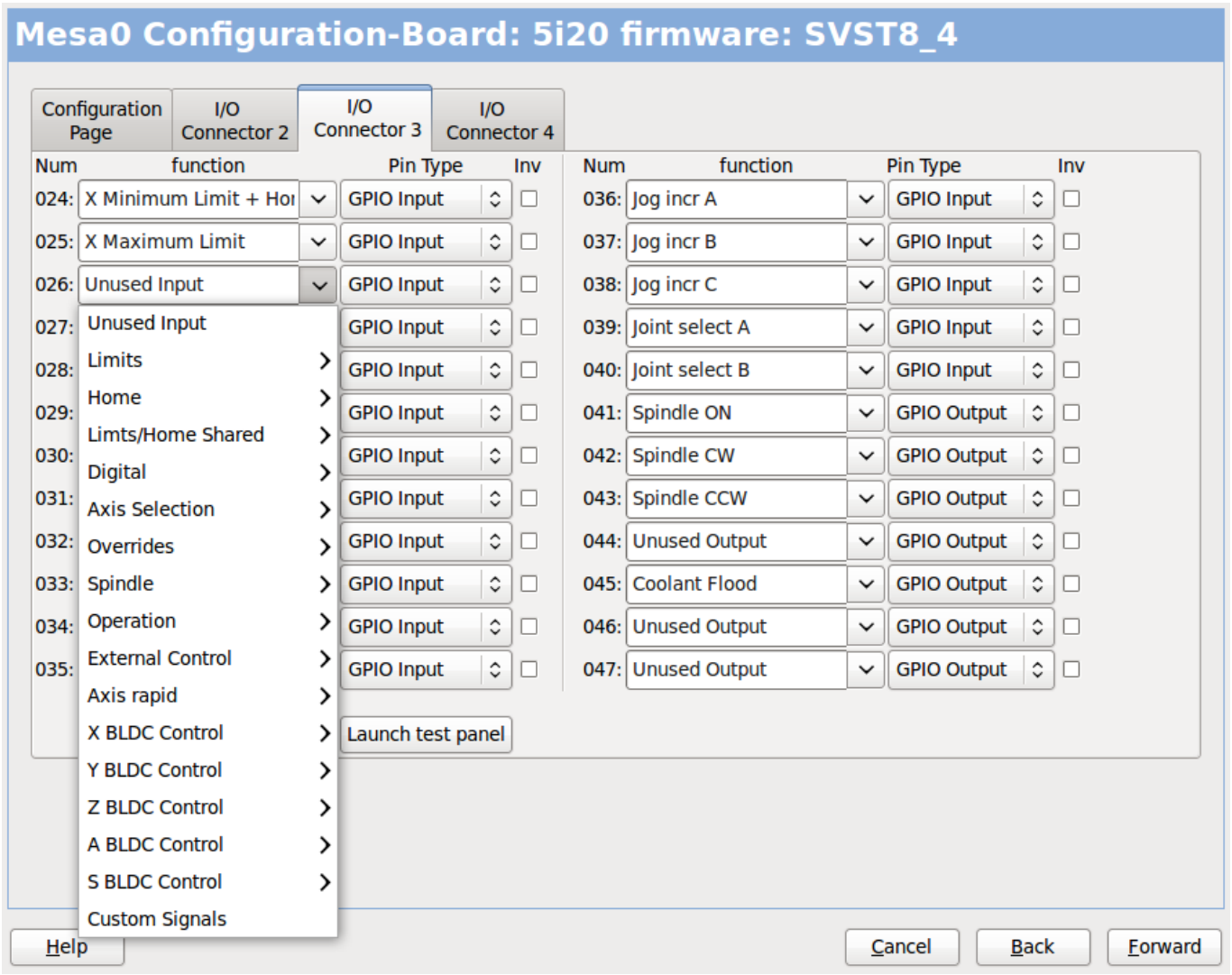


Figure 3.19: Налаштування C3 вводу/виводу Mesa

На цій вкладці всі контакти є GPIO. Зверніть увагу на 3-значні числа — вони відповідатимуть номеру контакту HAL. Контакти GPIO можна вибрати як вхідні або вихідні, а також інвертувати.

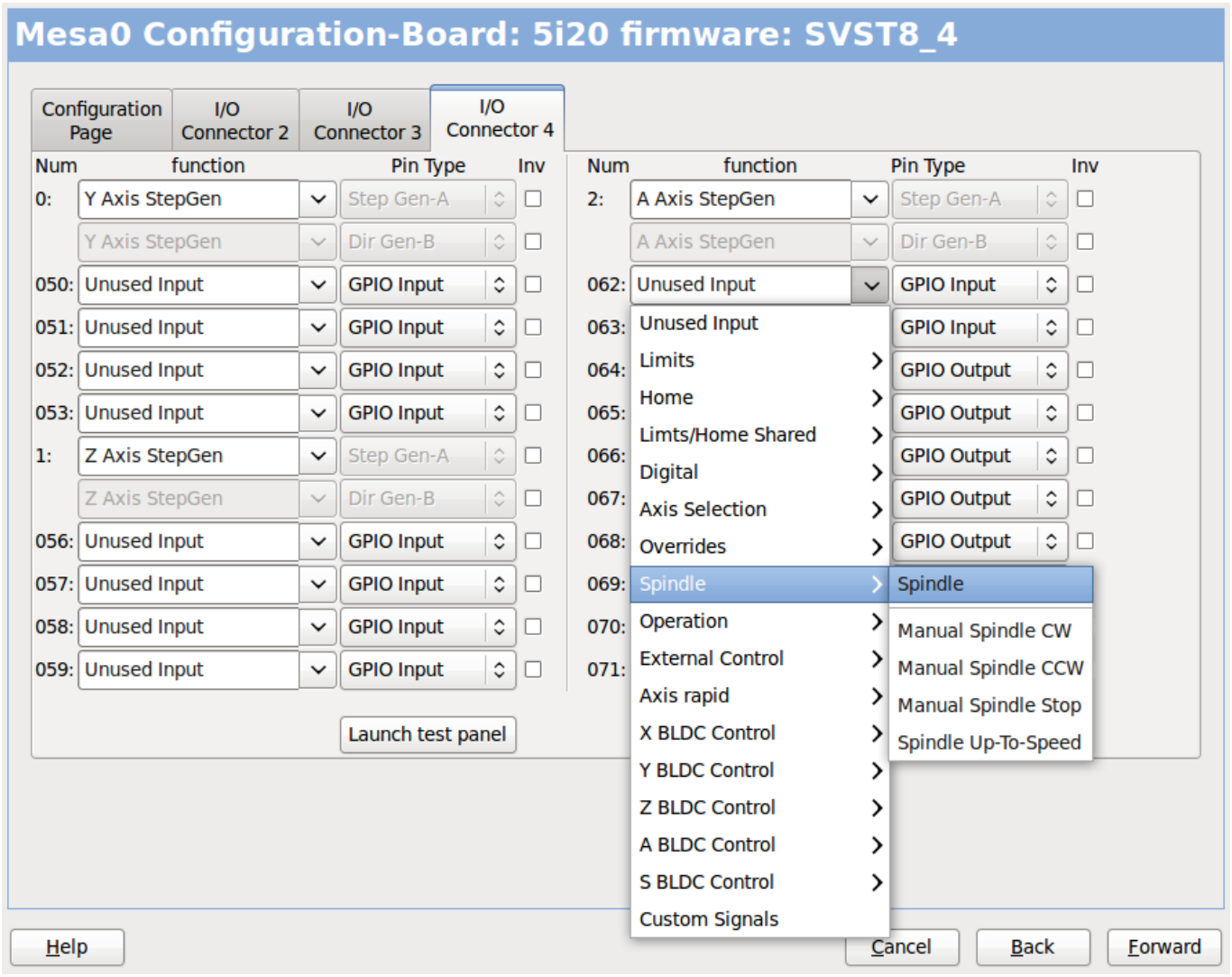
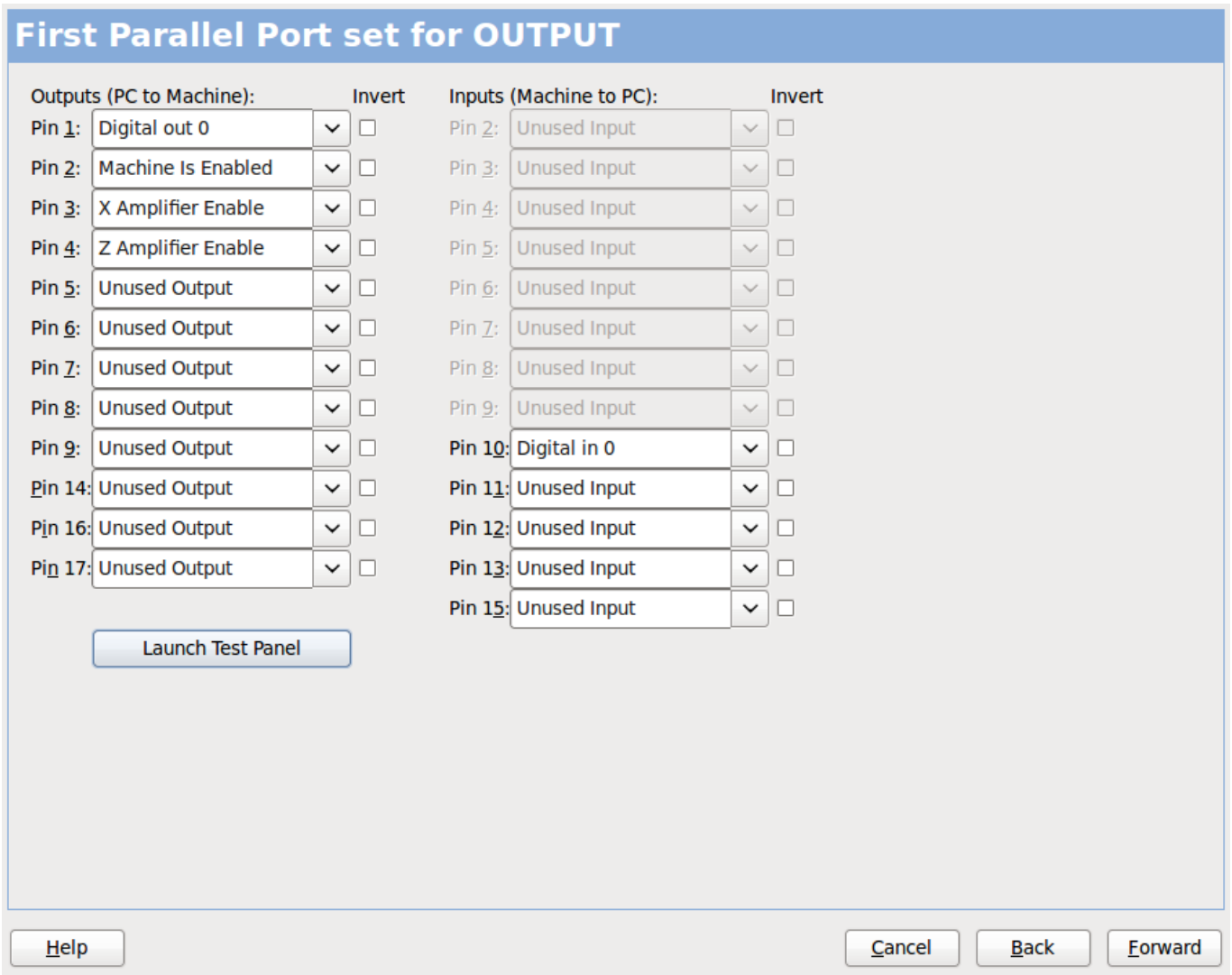


Figure 3.20: Налаштування Mesa I/O C4

На цій вкладці є поєднання генераторів кроків і GPIO. Вихідні та напрямні контакти генераторів кроків можна інвертувати. Зверніть увагу, що інвертування контакту Step Gen-A (контакту виходу кроку) змінює синхронізацію кроку. Вона повинна відповідати очікуванням вашого контролера.

3.2.8 Конфігурація паралельного порту



Паралельний порт можна використовувати для простого введення/виведення, подібно до контактів GPIO Mesa.

3.2.9 Конфігурація осі

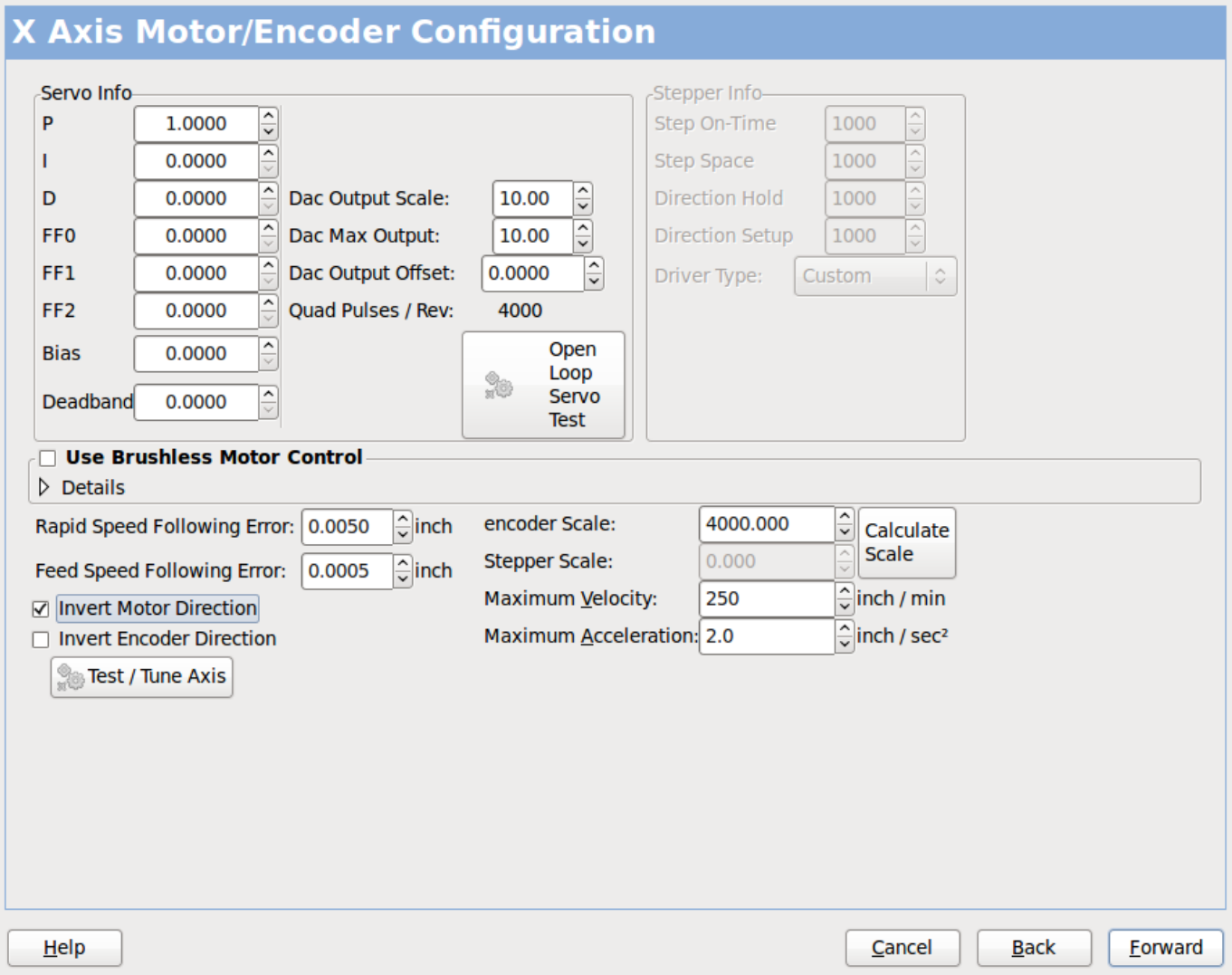


Figure 3.21: Конфігурація приводу осі

Ця сторінка дозволяє налаштовувати та тестувати комбінацію двигуна та/або енкодера. Якщо використовується серводвигун, доступний тест у розімкнутому контурі, якщо використовується кроковий двигун, доступний тест налаштування.

Тестування відкритого циклу

Тест з відкритим контуром є важливим, оскільки він підтверджує напрямок руху двигуна та енкодера. Двигун повинен рухати вісь у позитивному напрямку при натисканні кнопки «позитивний», а енкодер повинен рахувати в позитивному напрямку. Рух осі повинен відповідати стандартам, наведеним у примітці до «Довідника з машинобудування»: [«номенклатура осей» у розділі «Числове управління» в «Довіднику з машинобудування», опублікованому видавництвом Industrial Press]. В іншому випадку графічне відображення AXIS не матиме сенсу. Сподіваємося, що сторінка довідки та діаграми допоможуть розібратися в цьому. Зверніть увагу, що напрямки осей базуються на русі ІНСТРУМЕНТУ, а не на русі столу. У тесті з відкритим контуром немає прискорення, тому починайте з нижчих значень DAC. Перемістивши вісь на відому відстань, можна підтвердити масштабування енкодера. Енкодер

повинен рахувати навіть без увімкненого підсилювача, залежно від того, як до нього подається живлення.



Warning
Якщо двигун і енкодер не узгоджують напрямок відліку, то сервопривід буде рухатися в режимі реального часу під час використання PID-керування.

Оскільки на даний момент налаштування PID не можна перевірити в PnCconf, ці налаштування насправді призначені для випадків повторного редагування конфігурації – введіть перевірені налаштування PID.

Шкала DAC
Масштабування DAC, максимальний вихідний сигнал та зміщення використовуються для налаштування вихідного сигналу DAC.

Обчисл. DAC
Ці два значення є коефіцієнтами масштабування та зміщення для виходу осі на підсилювачі двигуна. Друге значення (зсув) віднімається від обчисленого виходу (у вольтах) і ділиться на перше значення (коефіцієнт масштабування) перед записом у цифро-аналогові перетворювачі. Одиниці виміру значення масштабування виражаються у вольтах на вихідний вольт цифро-аналогового перетворювача. Одиниці виміру значення зсуву виражаються у вольтах. Вони можуть використовуватися для лінеаризації цифро-аналогового перетворювача.

Зокрема, під час запису вихідних даних LinuxCNC спочатку перетворює бажаний вихід у квазі-SI одиниці виміру в необроблені значення приводу, наприклад, вольти для підсилювача DAC. Це масштабування виглядає так: Значення масштабу можна отримати аналітично, виконавши аналіз одиниць виміру, тобто одиниці виміру є [одиниці виміру SI на виході]/[одиниці виміру приводу]. Наприклад, на машині з підсилювачем у режимі швидкості, де 1 вольт відповідає швидкості 250 мм/с. Зверніть увагу, що одиниці зміщення виражені в одиницях машини, наприклад, мм/с, і вони заздалегідь віднімаються від показань датчика. Значення цього зміщення отримується шляхом знаходження значення вашого виходу, яке дає 0,0 для виходу приводу. Якщо DAC лінеаризований, це зміщення зазвичай дорівнює 0,0.

Масштаб і зміщення також можна використовувати для лінеаризації DAC, отримуючи значення, що відображають сукупний вплив коефіцієнта підсилення підсилювача, нелінійності DAC, одиниць DAC тощо. Для цього виконайте таку процедуру:

- Створіть калібрувальну таблицю для вихідного сигналу, подаючи на DAC потрібну напругу та вимірюючи результат:

Table 3.2: Вимірювання вихідної напруги

Сире	Виміряно
-10	**-9.93**
-9	**-8.83**
0	**-0.96**
1	**-0.03**
9	**9.87**
10	**10.07**

- Виконайте лінійну апроксимацію методом найменших квадратів, щоб отримати коефіцієнти a, b такі, що meas=a*raw+b

- Зверніть увагу, що нам потрібен необроблений вихідний сигнал, щоб наш виміряний результат був ідентичним заданому виходу. Це означає
  - cmd=a*raw+b
  - raw=(cmd-b)/a
- В результаті, коефіцієнти a та b з лінійної апроксимації можна використовувати безпосередньо як масштаб та зміщення для контролера.

МАКСИМАЛЬНА ВИХІДНІСТЬ
: Максимальне значення вихідного сигналу PID-компенсації, яке записується в підсилювач двигуна, у вольтах. Обчислене вихідне значення обмежується цим обмеженням. Обмеження застосовується перед масштабуванням до вихідних одиниць. Значення застосовується симетрично як до плюсової, так і до мінусової сторони.

Тест налаштування
: На жаль, тест налаштування працює тільки з системами на основі крокових двигунів. Ще раз переконайтеся, що напрямок осі правильний. Потім протестуйте систему, переміщаючи вісь вперед і назад. Якщо прискорення або максимальна швидкість занадто високі, ви втратите кроки. Під час переміщення пам’ятайте, що осі з низьким прискоренням може знадобитися деякий час, щоб зупинитися. Під час цього тесту кінцеві вимикачі не працюють. Ви можете встановити час паузи на кожному кінці тестового руху. Це дозволить вам налаштувати і зчитати показники індикатора, щоб перевірити, чи не втрачаєте ви кроки.

Крековий таймінг
: Частота кроку повинна бути адаптована до вимог контролера кроку. PnCconf надає деякі стандартні налаштування частоти контролера або дозволяє використовувати власні налаштуван Дивіться https://wiki.linuxcnc.org/cgi-bin/wiki.pl?Stepper_Drive_Timing для більш відомих значень синхронізації (можете додавати ті, які ви самі визначили). Якщо ви сумніваєтеся, використовуйт великі числа, такі як 5000, це лише обмежить максимальну швидкість.

Безщіткове керування двигуном
: Ці опції використовуються для забезпечення низькорівневого керування безщітковими двигунами за допомогою спеціального вбудованого програмного забезпечення та дочірніх плат. Вони також дозволяють перетворювати датчики HALL від одного виробника на датчики іншого виробника. Ця функція підтримується лише частково і вимагає завершення підключення HAL. Для отримання додаткової допомоги зверніться до списку розсилки або форуму.

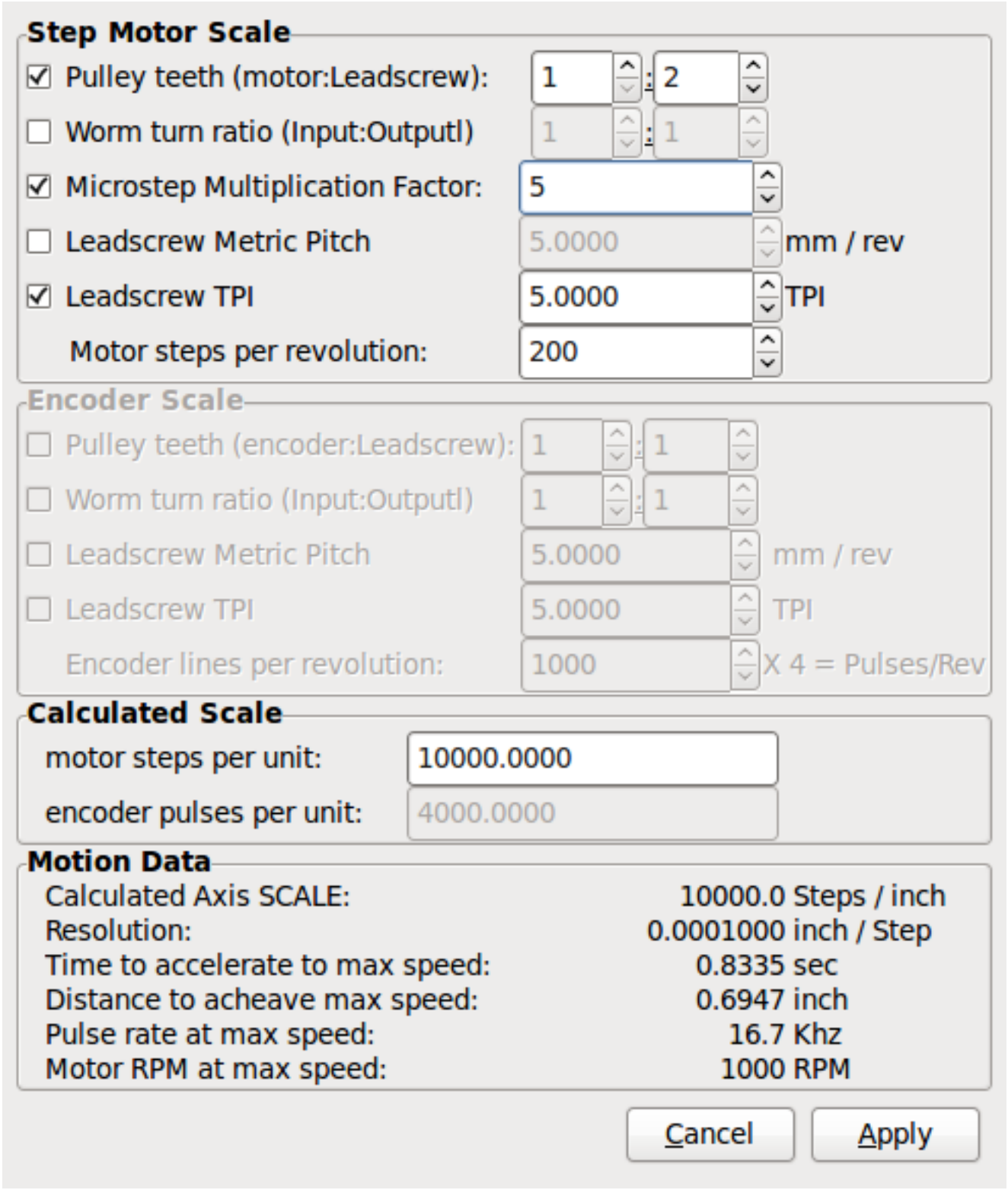


Figure 3.22: Розрахунок масштабу осей

Налаштування масштабу можна ввести безпосередньо або скористатися кнопкою «Розрахувати масштаб». Використовуйте прапорці, щоб вибрати відповідні розрахунки. Зверніть увагу, що для «зубців шківа» потрібно вказати кількість зубців, а не передавальне число. Для черв'ячного передавального числа потрібно вказати передавальне число. Якщо вас влаштовує масштаб,

натисніть «Застосувати», інакше натисніть «Скасувати» і введіть масштаб безпосередньо.

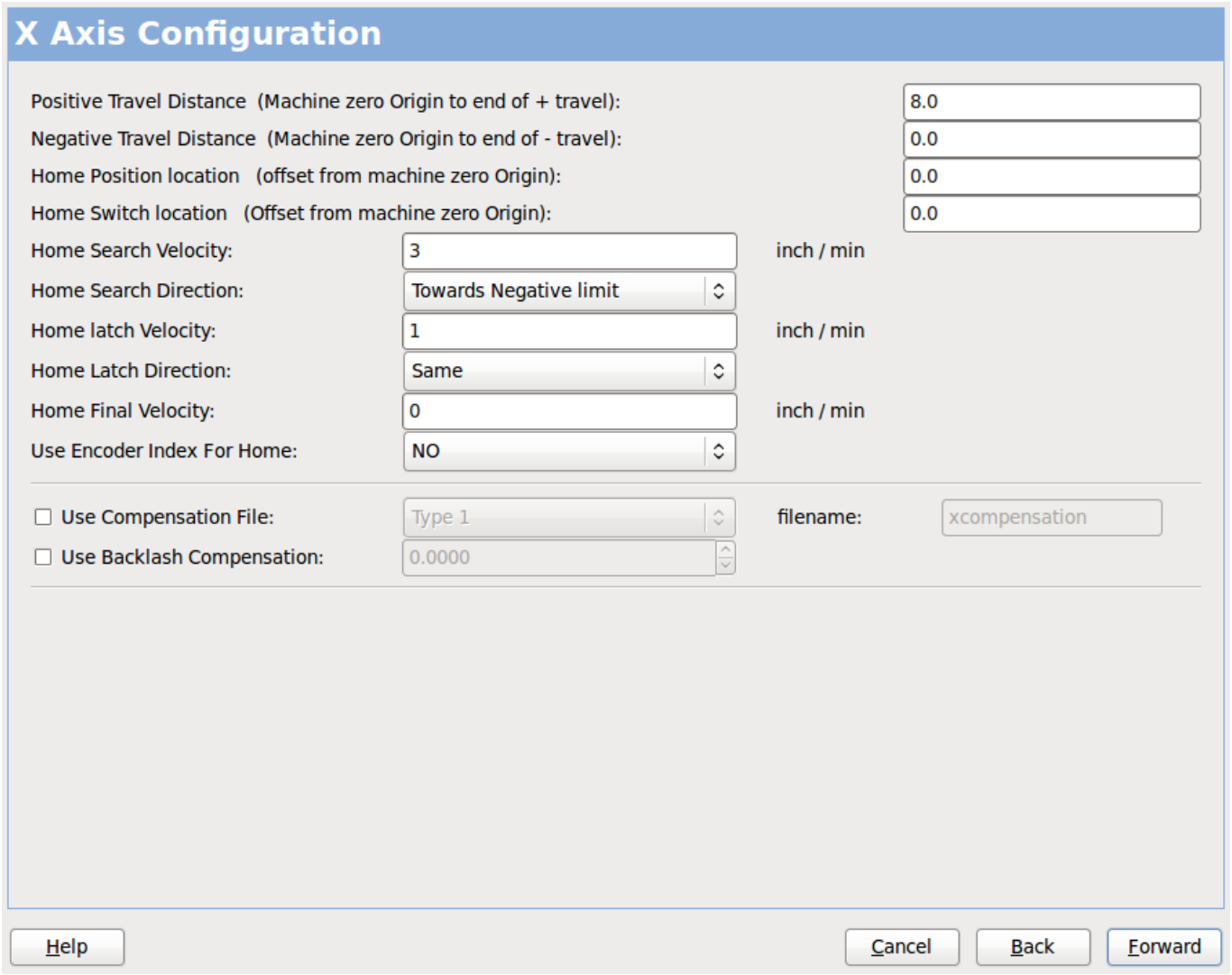


Figure 3.23: Конфігурація осі

Також зверніться до вкладки схеми, щоб побачити два приклади вимикачів додому та кінцевих вимикачів. Це два приклади багатьох різних способів встановлення вихідного положення та кінцевих вимикачів.

Important
Дуже важливо почати з руху осі в правильному напрямку, інакше правильне повернення до початкового положення буде дуже складним!

Пам’ятайте, що позитивний та негативний напрямки стосуються ІНСТРУМЕНТУ, а не столу, як зазначено в посібнику машиніста.

На типовому колінному або станинному млині

- коли СТІЛ рухається назовні, це позитивний напрямок Y

- коли СТІЛ рухається ліворуч, це позитивний напрямок X
- коли СТІЛ рухається вниз, це позитивний напрямок Z
- коли ГОЛОВКА рухається вгору, це позитивний напрямок Z

На типовому токарному верстаті

- коли ІНСТРУМЕНТ рухається праворуч, від патрона
- це позитивний напрямок Z
- коли ІНСТРУМЕНТ рухається до оператора
- це позитивний напрямок X. Деякі токарні верстати мають протилежний напрямок X (наприклад, інструмент на зворотному боці), це працюватиме добре, але графічне відображення AXIS не може відображати це.

При використанні перемикачів повернення в початкове положення та/або кінцевих вимикачів LinuxCNC очікує, що сигнали HAL будуть справжніми, коли перемикач натискається/спрацьовує. Якщо сигнал для кінцевого вимикача неправильний, LinuxCNC буде вважати, що машина постійно знаходиться в кінці діапазону. Якщо логіка пошуку перемикача повернення в початкове положення неправильна, LinuxCNC буде намагатися повернутися в неправильному напрямку. Насправді він намагається ВІДСТУПИТИ від перемикача повернення в початкове положення.

Визначтеся з місцем розташування кінцевого вимикача
Кінцеві вимикачі є резервним варіантом для програмних обмежень на випадок, якщо щось піде не так з електрообладнанням, наприклад, у разі виходу сервоприводу з ладу. Кінцеві вимикачі слід розміщувати так, щоб машина не досягала фізичного кінця руху осі. Пам'ятайте, що вісь пройде повз точку контакту, якщо рухається швидко. Кінцеві вимикачі повинні бути «активними низькими» на машині, тобто живлення проходить через вимикачі постійно - втрата живлення (відкритий вимикач) спрацьовує. Хоча їх можна підключити іншим способом, це є безпечним. Можливо, це потрібно буде інвертувати, щоб сигнал HAL в LinuxCNC був «активним високим» - TRUE означає, що вимикач спрацював. Якщо під час запуску LinuxCNC ви отримуєте попередження про досягнення межі, а вісь НЕ спрацьовує вимикач, ймовірно, рішенням буде інвертування сигналу. (використовуйте HALMETER для перевірки відповідного сигналу HAL, наприклад, joint.0.pos-lim-sw-in X axis positive limit switch)

Визначтеся з місцем розташування домашнього вимикача
Якщо ви використовуєте кінцеві вимикачі, ви також можете використовувати один з них як вимикач початкового положення. Окремий вимикач початкового положення корисний, якщо у вас є довга вісь, яка під час використання зазвичай знаходиться далеко від кінцевих вимикачів, або переміщення осі до кінців створює проблеми з перешкодами для матеріалу. Зверніть увагу, що довгий вал у токарному верстаті ускладнює повернення до кінцевих положень без зіткнення інструменту з валом, тому окремий перемикач початкового положення, розташований ближче до середини, може бути кращим варіантом. Якщо у вас є енкодер з індексом, перемикач початкового положення діє як приблизне початкове положення, а індекс буде фактичним початковим положенням.

Визначте позицію ПОЧАТКОВОЇ ТОЧКИ МАШИНИ
MACHINE ORIGIN — це те, що LinuxCNC використовує для посилання на всі системи координат користувача. Я не бачу причин, чому це повинно бути в якомусь конкретному місці. Є лише кілька G-кодів, які можуть отримати доступ до системи MACHINE COORDINATE (G53, G30 і G28). Якщо ви використовуєте опцію tool-change-at-G30, зручно мати початок координат у положенні зміни інструменту. Зазвичай найпростіше мати ORIGIN у положенні home switch.

Визначтеся з (остаточною) ДОМАШНЬОЮ ПОЗИЦІЄЮ
це просто розміщує каретку в стабільному та зручному положенні після того, як LinuxCNC визначить, де знаходиться ПОЧАТОК ВІДПРАВЛЕННЯ.

Вимірювання/розрахунок відстаней переміщення по додатній/від'ємній осі
Перемістіть вісь до початку координат. Позначте опорну точку на рухомому супорті та нерухомій опорі (так, щоб вони були на одній лінії), перемістіть верстат до кінця ходу. Виміряйте відстань між позначками, яка є однією з відстаней ходу. Перемістіть стіл до іншого кінця ходу. Виміряйте відстань між позначками знову. Це буде інша відстань ходу. Якщо ПОЧАТОК КООРДИНАТ знаходиться на одному з кінців ходу, то відстань ходу буде дорівнювати нулю.

(машина) ПОХОДЖЕННЯ
Початкова точка — це нульова точка МАШИНИ (не нульова точка, яку ви встановлюєте для різака/матеріалу). LinuxCNC використовує цю точку як відправну точку для всього іншого. Вона повинна знаходитися в межах програмних обмежень. LinuxCNC використовує розташування перемикача початкової точки для обчислення положення початкової точки (при використанні перемикачів початкової точки або необхідно встановити вручну, якщо перемикачі початкової точки не використовуються).

Відстань подорожі
Це максимальна відстань, яку вісь може пройти в кожному напрямку. Її можна виміряти безпосередньо від початку до кінцевого вимикача або ж ні. Позитивні та негативні відстані переміщення повинні складати загальну відстань переміщення.

ПОЗИТИВНА ВІДСТАНЬ ПОДОРОЖІ
Це відстань, яку проходить вісь від початку координат до позитивної відстані переміщення або загальної відстані переміщення мінус негативна відстань переміщення. Ви повинні встановити це значення на нуль, якщо початок координат розташований на позитивній межі. Це значення завжди буде нулем або додатним числом.

ВІД'ЄМНА ВІДСТАНЬ ПОДОРОЖІ
Це відстань, яку проходить вісь від початку координат до від'ємної відстані переміщення. або загальна відстань переміщення мінус додатна відстань переміщення. Ви повинні встановити це значення на нуль, якщо початок координат розташований на від'ємній межі. Це значення завжди буде нулем або від'ємним числом. Якщо ви забудете встановити це значення від'ємним, PnCconf зробить це внутрішньо.

(Фінальне) ДОМАШНЄ МАЙДАНЧИК
Це положення, в якому закінчується послідовність повернення в початкове положення. Воно відлічується від початку координат, тому може бути від'ємним або додатним залежно від того, з якого боку від початку координат воно розташоване. Якщо в (кінцевому) положенні повернення в початкове положення вам потрібно рухатися в додатньому напрямку, щоб дістатися до початку координат, то число буде від'ємним.

МІСЦЕЗНАХОДЖЕННЯ ДОМАШНЬОГО ПЕРЕКИДАЧА
Це відстань від домашнього вимикача до точки відліку. Вона може бути від'ємною або додатньою залежно від того, з якого боку від точки відліку вона розташована. Якщо в місці розташування домашнього вимикача для досягнення точки відліку необхідно рухатися в додатньому напрямку, то число буде від'ємним. Якщо встановити це значення на нуль, то точка відліку буде розташована в місці розташування кінцевого вимикача (плюс відстань для пошуку індексу, якщо він використовується).

Швидкість пошуку на головній сторінці
Швидкість пошуку початкового курсу в одиницях за хвилину.

Напрямок пошуку на головній сторінці
Встановлює напрямок пошуку домашнього вимикача або негативний (тобто до негативного кінцевого вимикача), або позитивний (тобто до позитивного кінцевого вимикача).

Швидкість фіксації будинку
Швидкість пошуку Fine Home в одиницях за хвилину.

Домашня кінцева швидкість
Швидкість, що використовується від положення фіксації до (остаточного) початкового положення в одиницях за хвилину. Встановіть значення 0 для максимальної швидкості переміщення.

Напрямок фіксатора
Дозволяє встановити напрямок засувки таким самим або протилежним напрямку пошуку.

Використовуйте індекс енкодера для дому
LinuxCNC шукатиме імпульс індексу енкодера під час етапу фіксації переміщення до початкового положення.

Використовувати файл компенсації
Дозволяє вказати назву та тип файлу Comp. Дозволяє складну компенсацію. Див. розділ AXIS розділу INI.

Використовуйте компенсацію люфту
Дозволяє налаштувати просту компенсацію люфту. Не можна використовувати з файлом компенсації. Див. розділ AXIS розділу INI.

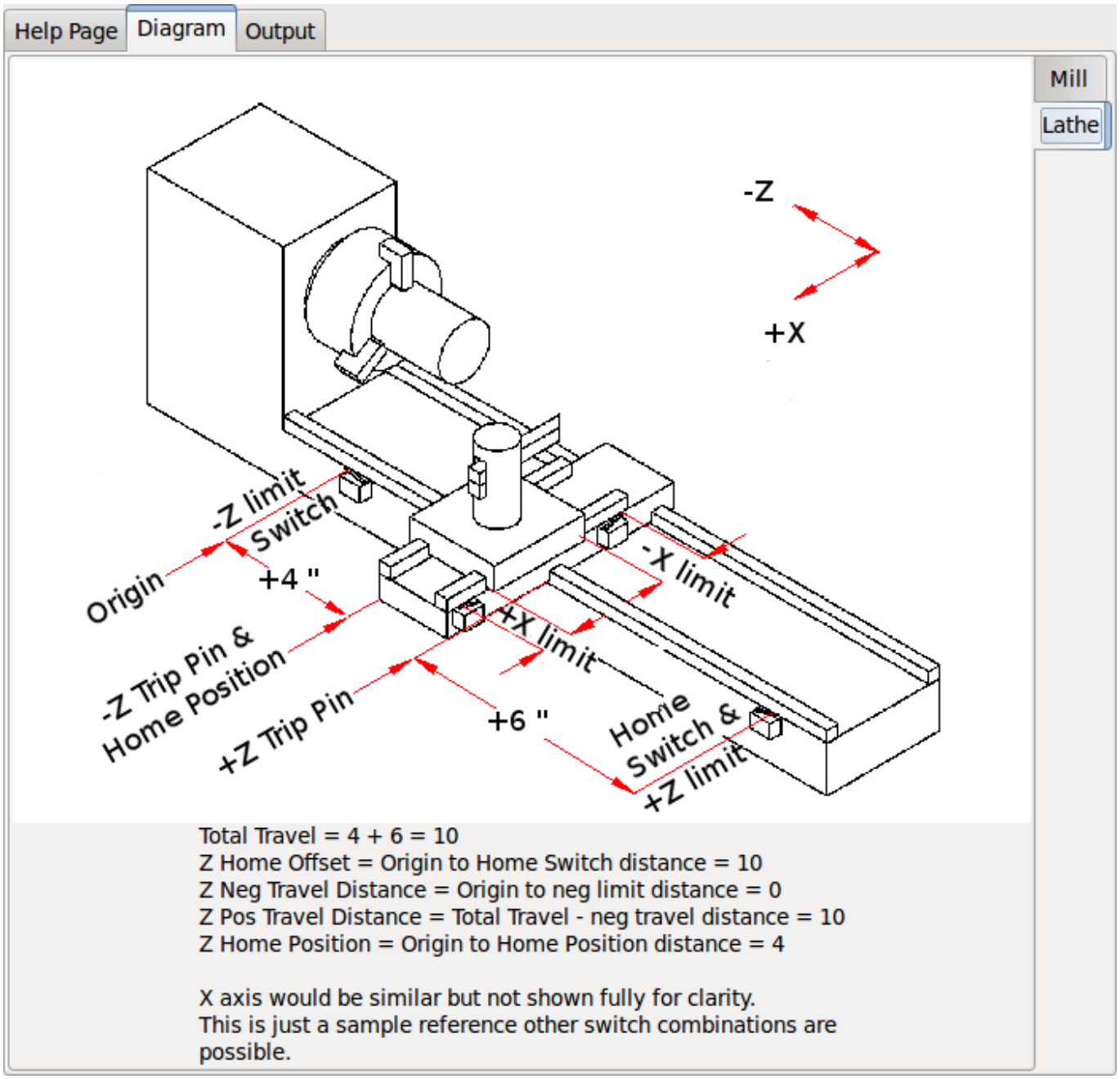


Figure 3.24: Довідкова діаграма AXIS

Діаграма повинна допомогти продемонструвати приклад кінцевих вимикачів і стандартних напрямків руху осей. У цьому прикладі вісь Z мала два кінцеві вимикачі, позитивний вимикач використовується як вимикач початкового положення. ПОВЕРХНЯ МАШИНИ (нульова точка) розташована на негативному кінцевому вимикачі. Лівий край каретки є негативним вимикачем, а правий — позитивним вимикачем. Ми хочемо, щоб КІНЦЕВА ПОЛОЖЕННЯ БУЛА на відстані 4 дюймів від ПОВЕРХНІ на позитивній стороні. Якщо каретка була переміщена до позитивного кінцевого вимикача, ми б виміряли 10 дюймів між негативним кінцевим вимикачем і негативним вимикачем.

3.2.10 Конфігурація шпинделя

Якщо ви вибрали сигнали шпинделя, то ця сторінка доступна для налаштування керування шпинделем.

Tip
Багато опцій на цій сторінці не відображатимуться, якщо відповідну опцію не було обрано на попередніх сторінках!

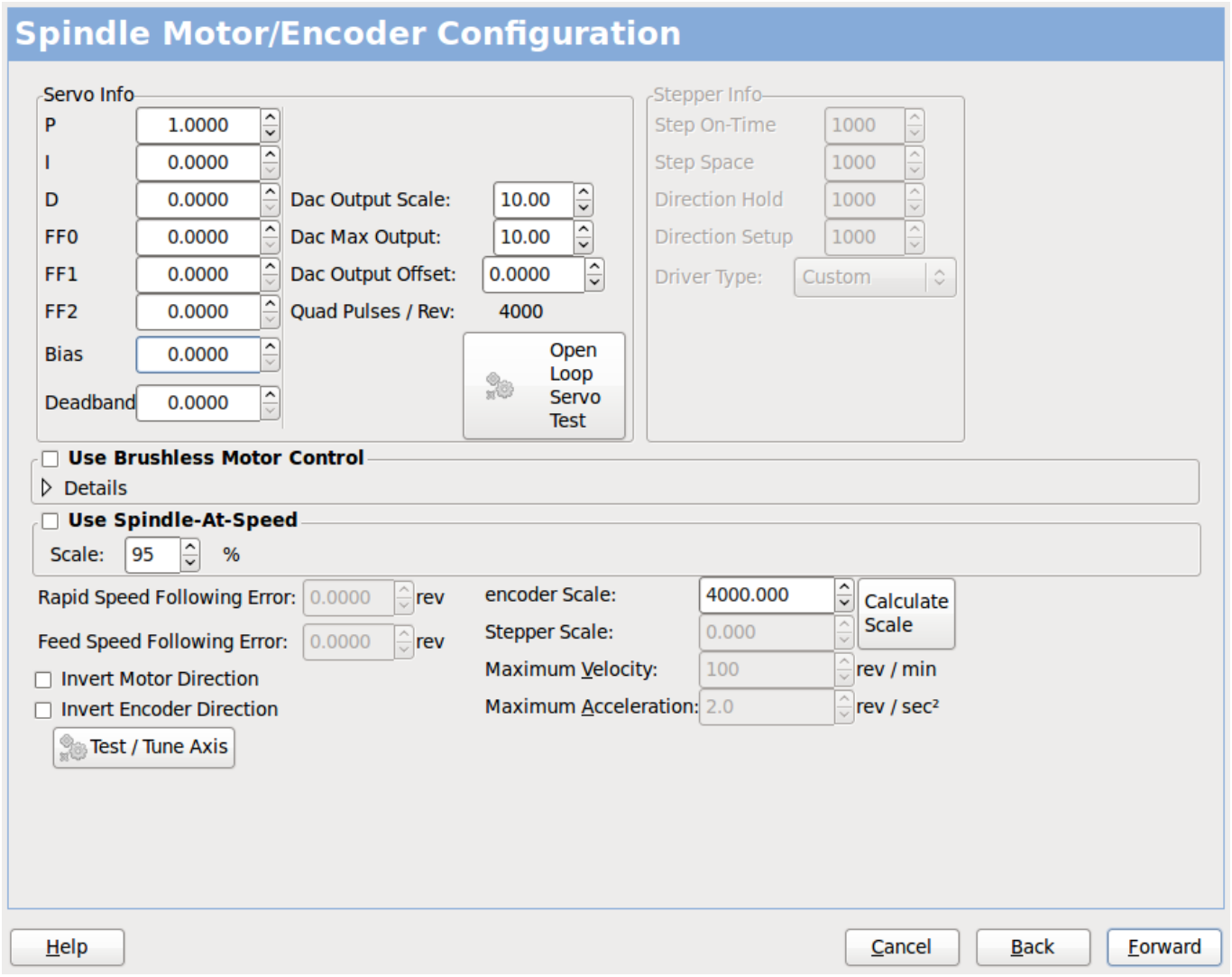


Figure 3.25: Конфігурація двигуна/енкодера шпинделя

Ця сторінка схожа на сторінку конфігурації двигуна осі.

Є деякі відмінності:

- Якщо не обрано шпиндель із кроковим приводом, обмеження прискорення чи швидкості відсутнє.
- Немає підтримки перемикання передач або діапазонів.

- Якщо ви вибрали опцію відображення шпинделя VCP, тоді можуть відображатися шкала швидкості шпинделя та налаштування фільтра.
- Функція «Шпиндель на швидкості» дозволяє LinuxCNC чекати, поки шпиндель досягне заданої швидкості, перш ніж переміщати вісь. Це особливо зручно на токарних верстатах з постійною подачею поверхні та великими змінами діаметра швидкості. Для цього потрібна зворотний зв'язок від енкодера або цифровий сигнал «Шпиндель на швидкості», який зазвичай підключається до приводу VFD.
- Якщо використовується зворотний зв'язок від енкодера, ви можете вибрати налаштування шкали швидкості шпинделя, яке визначає, наскільки близькою має бути фактична швидкість до запитуваної швидкості, щоб вважатися такою, що обертається на заданій швидкості.
- Якщо використовується зворотний зв'язок від енкодера, відображення швидкості VCP може бути нестабільним – для згладжування відображення можна використовувати налаштування фільтра. Шкалу енкодера необхідно встановити відповідно до лічильника/передачі енкодера, що використовується.
- Якщо ви використовуєте один вхід для енкодера шпинделя, ви повинні додати рядок: setp hm2_7i43.0.encoder.00.counter-mode 1 (змінивши назву плати та номер енкодера відповідно до ваших вимог) у файл HAL. Дивіться розділ Encoders Section у Hostmot2 для отримання додаткової інформації про режим лічильника.

3.2.11 Розширені параметри

Це дозволяє встановлювати команди HALUI та завантажувати програми ClassicLadder і зразки програм ladder. Якщо ви вибрали опції GladeVCP, такі як обнулення осі, з'являться відповідні команди. Дивіться розділ HALUI Chapter для отримання додаткової інформації про використання власних halcmds. Існує декілька опцій програм ladder. Програма Estop дозволяє зовнішньому перемикачу ESTOP або інтерфейсу GUI запускати Estop. Вона також має сигнал змащувального насоса з таймером. Автоматичне відключення Z здійснюється за допомогою пластини відключення, кнопки відключення GladeVCP та спеціальних команд HALUI для обнулення поточного початку координат користувача та швидкого очищення. Програма послідовного Modbus є, по суті, порожньою шаблоною програмою, яка налаштовує ClassicLadder для послідовного Modbus. Дивіться розділ ClassicLadder Chapter в посібнику.

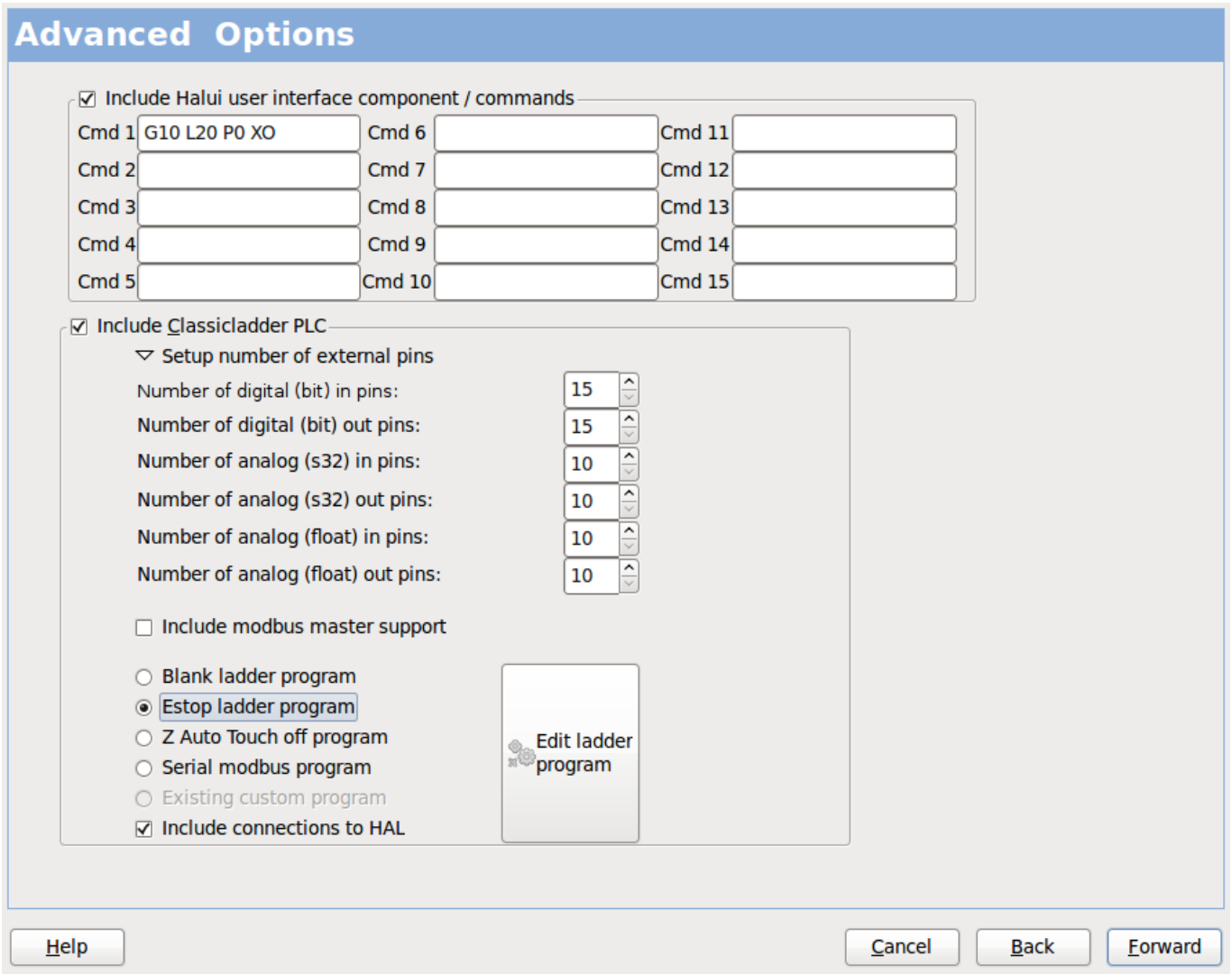


Figure 3.26: PnCconf, розширені параметри

3.2.12 Компоненти HAL

На цій сторінці ви можете додати додаткові компоненти HAL, які можуть знадобитися для користувацьк
файлів HAL. Таким чином, вам не доведеться вручну редагувати основний файл HAL, водночас
дозволяючи використовувати необхідні користувачеві компоненти.

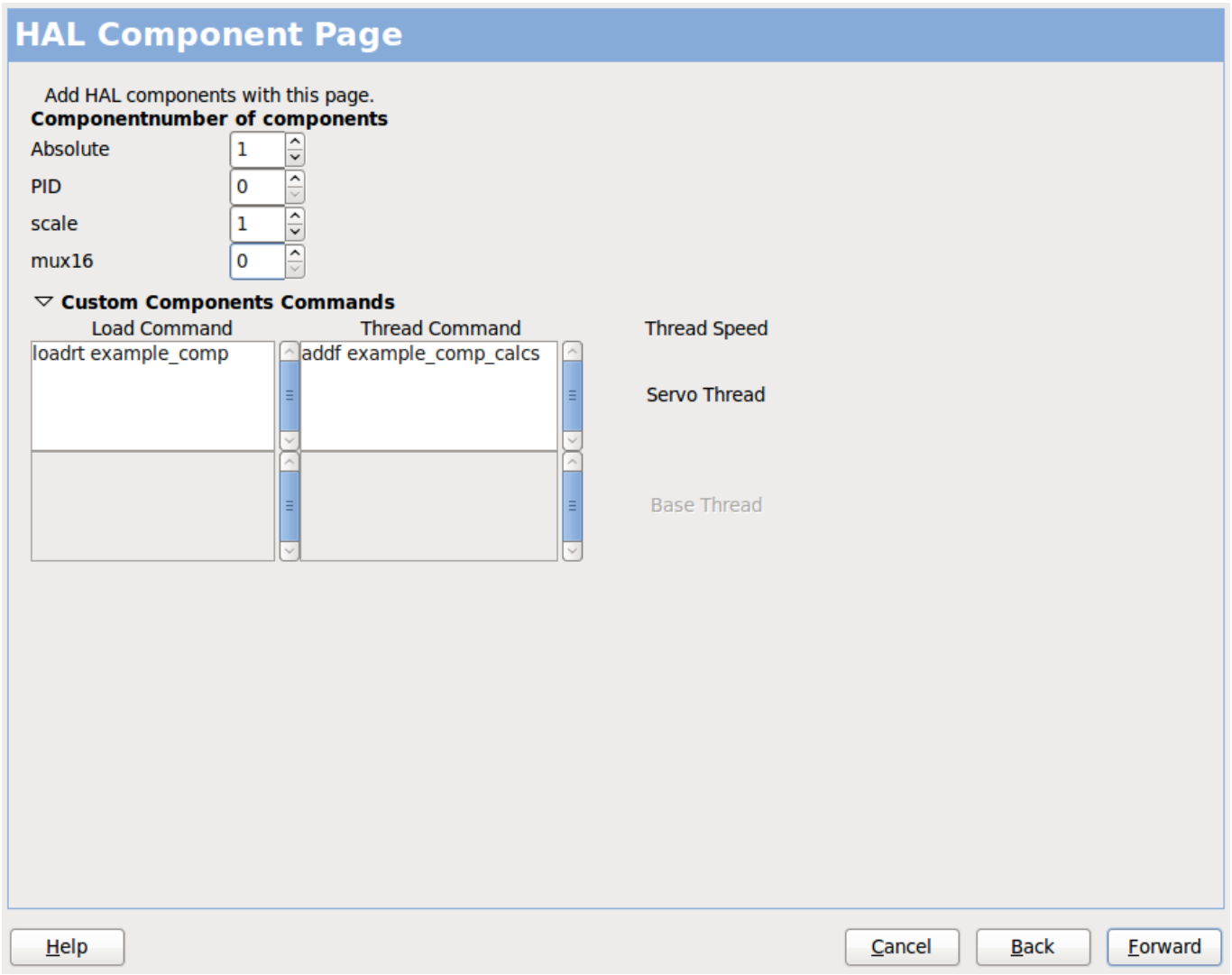


Figure 3.27: Компоненти HAL

Перший вибір — це компоненти, які pncconf використовує внутрішньо. Ви можете налаштувати pncconf для завантаження додаткових екземплярів компонентів для вашого власного HAL-файлу.

Виберіть кількість екземплярів, необхідних для вашого користувацького файлу, PnCconf додасть необхідні копії після них.

Тобто, якщо вам потрібно 2, а PnCconf потрібен 1, PnCconf завантажить 3 екземпляри та використає останній.

Команди користувацьких компонентів
: Цей вибір дозволить вам завантажити компоненти HAL, які PnCconf не використовує. Додайте команду loadrt або loadusr під заголовком «команда завантаження». Додайте команду addf під заголовком «команда потоку». Компоненти будуть додані до потоку між читанням вхідних даних і записом вихідних даних у тому порядку, в якому ви їх записуєте в «команді потоку».

3.2.13 Розширене використання PnCconf

PnCconf робить усе можливе, щоб користувач міг гнучко налаштувати параметри. PnCconf підтримує налаштування імен сигналів, завантаження компонентів, HAL-файлів та прошивок.

Існують також імена сигналів, які PnCconf завжди надає незалежно від вибраних опцій, для користувацьких файлів HAL. При певному підході більшість налаштувань повинні працювати незалежно від того, чи вибрали ви пізніше інші опції в PnCconf.

Якщо налаштування виходять за межі можливостей PnCconf, ви можете використовувати PnCconf для створення базової конфігурації або скористатися одним із зразків конфігурацій LinuxCNC і просто вручну відредагувати його відповідно до своїх потреб.

Назви власних сигналів
Якщо ви хочете підключити компонент до чогось у власному HAL-файлі, напишіть унікальну назву сигналу у комбінованому полі введення. Деякі компоненти додадуть закінчення до вашої власної назви сигналу:

Кодери додадуть <customname> +:

- позиція
- рахувати
- швидкість
- індекс-включити
- скинути

Степери додають:

- увімкнути
- підрахунки
- position-cmd
- position-fb
- velocity-fb

Додавання PWM:

- увімкнути
- значення

До контактів GPIO буде підключено лише введену назву сигналу

Таким чином, можна підключитися до цих сигналів у користувацьких HAL-файлах і при цьому мати можливість переміщувати їх пізніше.

Назви власних сигналів
Сторінку «Компоненти HAL» можна використовувати для завантаження компонентів, необхідних користувачеві для налаштування.

Завантаження користувацької прошивки
PnCconf шукає прошивку в системі, а потім шукає XML-файл, який він може перетворити на те, що він розуміє. Ці XML-файли надаються тільки для офіційно випущеної прошивки від команди LinuxCNC. Щоб використовувати власну прошивку, її потрібно перетворити на масив, який розуміє PnCconf, і додати шлях до файлу в файл налаштувань PnCconf. За замовчуванням цей шлях шукає на робочому столі папку з назвою custom_firmware і файл з назвою firmware.py.

Прихований файл налаштувань знаходиться в домашній папці користувача, має назву .pncconf-preferences і для його перегляду та редагування необхідно вибрати опцію «показати приховані файли» у файловому менеджері або ввести команду «ls» з опцією «-a» у командному рядку. Вміст цього файлу можна переглянути під час першого запуску PnCconf — натисніть кнопку довідки та перегляньте сторінку з результатами.

Зверніться до списку розсилки або форуму LinuxCNC, щоб отримати інформацію про конвертацію користувацьких прошивок. Не всі прошивки можна використовувати з PnCconf.

Користувацькі файли HAL

Існує чотири користувацькі файли, які можна використовувати для додавання команд HAL:

- Файл custom.hal призначений для команд HAL, які не потрібно виконувати після завантаження графічного інтерфейсу. Він виконується після файлу HAL з назвою конфігурації.
- custom_postgui.hal призначений для команд, які необхідно виконати після завантаження AXIS або окремого дисплея PyVCP.
- custom_gvcp.hal призначений для команд, які необхідно виконати після завантаження GladeVCP.
- shutdown.hal призначений для команд, які запускаються, коли LinuxCNC вимикається контрольованим чином.

Chapter 4

Конфігурація

4.1 Концепції інтегратора

4.1.1 Розташування файлів

LinuxCNC шукає файли конфігурації та G-коду в певному місці. Розташування залежить від того, як ви запускаєте LinuxCNC.

4.1.1.1 Встановлено

Якщо ви запускаєте LinuxCNC з Live CD або встановили його через .deb-архів і використовуєте вибірник конфігурації «LinuxCNC» з меню, LinuxCNC шукає в таких каталогах:

- Каталог LinuxCNC знаходиться за адресою */home/user-name/linuxcnc*.
- Каталоги конфігурації розташовані за адресою */home/user-name/linuxcnc/configs*.
  - Файли конфігурації знаходяться за адресою */home/user-name/linuxcnc/configs/name-of-config*.
- Файли G-коду знаходяться за адресою /home/user-name/linuxcnc/nc_files'.

Наприклад, для конфігурації під назвою Mill та імені користувача Fred структура каталогів та файлів виглядатиме так.

- */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 Командний рядок

Якщо ви запускаєте LinuxCNC з командного рядка і вказуєте ім'я та розташування файлу INI, файли можуть знаходитися в іншому місці. Щоб переглянути параметри запуску LinuxCNC з командного рядка, виконайте команду «linuxcnc -h».

Note
Додаткові розташування для деяких файлів можна налаштувати у файлі INI. Див. розділ <<sub:ini:sec:display,[DISPLAY]>> та розділ <<sub:ini:sec:rs274ngc,[RS274NGC]>>.

4.1.2 Файли

Кожен каталог конфігурації вимагає щонайменше таких файлів:

- INI-файл .ini
- Файл HAL .hal або файл HALTCL .tcl, зазначений у розділі HAL файлу INI.

Note
Для деяких графічних інтерфейсів можуть знадобитися інші файли.

За бажанням ви також можете мати:

- Файл змінних .var
  - Якщо ви пропустите файл .var у каталозі, але включите <<sub:ini:sec:rs274ngc,[RS274NGC]>> PARAMETER_FILE=somefilename.var, файл буде створено під час запуску LinuxCNC.
  - Якщо ви пропустите файл .var і пропустите елемент [RS274NGC] PARAMETER_FILE, при запуску LinuxCNC буде створено файл var з назвою rs274ngc.var. Якщо пропустити [RS274NGC]PAR можуть з'явитися деякі заплутані повідомлення.
- Файл таблиці інструментів .tbl, якщо у файлі INI вказано <<sub:ini:sec:emcmot,[EMCMOT]>> TOOL_TABLE. Для деяких конфігурацій таблиця інструментів не потрібна.

4.1.3 Крокові системи

4.1.3.1 Базовий період

BASE_PERIOD є «серцем» вашого комп'ютера LinuxCNC. Примітка: [Цей розділ стосується використан **stepgen**, вбудованого генератора імпульсів LinuxCNC. Деякі апаратні пристрої мають власний генератор імпульсів і не використовують вбудований генератор LinuxCNC. У цьому випадку зверніться до посібника з експлуатації вашого обладнання.] Кожного періоду програмний генератор кроків вирішує, чи настав час для чергового імпульсу кроку. Коротший період дозволить вам генерувати більше імпульсів за секунду, в межах обмежень. Але якщо ви зробите його занадто коротким, ваш комп'ютер буде витрачати так багато часу на генерацію імпульсів кроків, що все інше сповільнитьс до повзучої швидкості або навіть заблокується. Затримка та вимоги до крокового приводу впливають на найкоротший період, який ви можете використовувати.

Найгірші випадки затримки можуть траплятися лише кілька разів на хвилину, а ймовірність того, що затримка трапиться саме в той момент, коли двигун змінює напрямок руху, є низькою. Тому ви можете отримати дуже рідкісні помилки, які час від часу псують деталь і які неможливо усунути.

Найпростіший спосіб уникнути цієї проблеми — вибрати BASE_PERIOD, що дорівнює сумі найдовшого часу, необхідного для роботи вашого накопичувача, та найгіршого випадку затримки вашого комп'ютера. Це не завжди є найкращим вибором. Наприклад, якщо ви використовуєте привід з вимогою до часу утримання сигналу напрямку 20 мкс, а тест затримки показав, що максимальна затримка становить 11 мкс, то якщо ви встановите BASE_PERIOD на 20+11 = 31 мкс, ви отримаєте не дуже приємні 32 258 кроків на секунду в одному режимі і 16 129 кроків на секунду в іншому режимі.

Проблема полягає у вимозі щодо часу утримання 20 мкс. Це, а також затримка 11 мкс, змушує нас використовувати повільний період 31 мкс. Але програмний генератор кроків LinuxCNC має деякі параметри, які дозволяють збільшити різні часи від одного періоду до декількох. Наприклад, якщо *steplen* примітка:[steplen відноситься до параметра, який регулює продуктивність вбудованого генератора кроків LinuxCNC, *stepgen*, який є компонентом HAL. Цей параметр регулює довжину самого імпульсу кроку. Продовжуйте читати, все буде пояснено пізніше.] змінюється з 1 на 2, то між початком і кінцем імпульсу кроку буде два періоди. Аналогічно, якщо *dirhold* примітка:[dirhold відноситься до параметра, який регулює тривалість утримання напрямку.] змінюється з 1 на 3, то між імпульсом кроку і зміною напрямку контакту буде принаймні три періоди.

Якщо ми можемо використовувати «dirhold» для виконання вимоги щодо часу утримання 20 мкс, то наступним за тривалістю є час високого рівня 4,5 мкс. Додайте затримку 11 мкс до часу високого рівня 4,5 мкс, і ви отримаєте мінімальний період 15,5 мкс. Коли ви спробуєте 15,5 мкс, ви побачите, що комп'ютер працює повільно, тому ви зупинитеся на 16 мкс. Якщо ми залишимо «dirhold» на 1 (за замовчуванням), то мінімальний час між кроком і напрямком буде періодом 16 мкс мінус затримка 11 мкс = 5 мкс, що недостатньо. Нам потрібно ще 15 мкс. Оскільки період становить 16 мкс, нам потрібно ще один період. Тому ми змінюємо «dirhold» з 1 на 2. Тепер мінімальний час від кінця імпульсу кроку до зміни напрямку становить 5+16=21 мкс, і нам не потрібно турбуватися про те, що привід рухатиметься в неправильному напрямку через затримку.

Для отримання додаткової інформації про stepgen див. розділ stepgen.

4.1.3.2 Час кроку

Час кроку та інтервал між кроками на деяких накопичувачах відрізняються. У цьому випадку точка кроку стає важливою. Якщо накопичувач переходить на спадний фронт, то вихідний контакт слід інвертувати.

4.1.4 Сервосистеми

4.1.4.1 Основні операції

Сервосистеми здатні розвивати більшу швидкість і точність, ніж еквівалентні крокові системи, але вони є більш дорогими і складними. На відміну від крокових систем, сервосистеми вимагають наявності певного типу пристрою зворотного зв'язку по положенню і повинні бути відрегульовані або «налаштовані», оскільки вони не працюють відразу після вилучення з коробки, як це може бути у випадку з кроковою системою. Ці відмінності існують тому, що сервоприводи є системою «замкнутого контуру», на відміну від крокових двигунів, які зазвичай працюють в «відкритому контурі». Що означає «замкнутий контур»? Давайте розглянемо спрощену схему підключення сервомоторної системи.

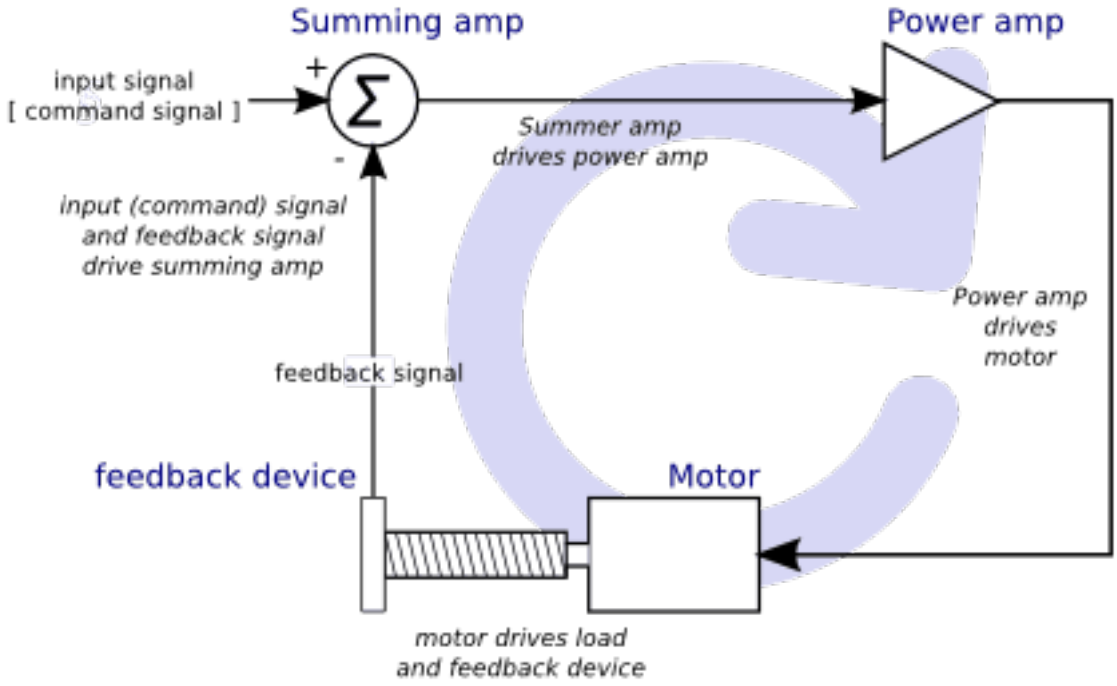


Figure 4.1: Сервоцикл

Ця схема показує, що вхідний сигнал (і сигнал зворотного зв’язку) керують сумуючим підсилювачем, сумуючий підсилювач керує підсилювачем потужності, підсилювач потужності керує двигуном, двигун керує навантаженням (і пристроєм зворотного зв’язку), а пристрій зворотного зв’язку (і вхідний сигнал) керують двигуном. Це дуже схоже на коло (замкнутий контур), де A керує B, B керує C, C керує D, а D керує A.

Якщо ви раніше не працювали з сервосистемами, спочатку це, без сумніву, здасться вам дуже дивною ідеєю, особливо в порівнянні з більш звичними електронними схемами, де вхідні сигнали плавно переходять у вихідні і ніколи не повертаються назад. Примітка: [Якщо це допоможе, найближчим еквівалентом цього в цифровому світі є «державні машини», «послідовні машини» тощо, де те, що виходи роблять «зараз», залежить від того, що входи (і виходи) робили «раніше». Якщо це не допоможе, то не звертайте уваги.] Якщо «все» контролює «все інше», як це може працювати, хто відповідає за це? Відповідь полягає в тому, що LinuxCNC «може» керувати цією системою, але для цього йому доводиться вибирати один із декількох методів керування. Метод керування, який використовує LinuxCNC, один із найпростіших і найкращих, називається PID.

PID означає пропорційний, інтегральний та похідний. Пропорційне значення визначає реакцію на поточну похибку, інтегральне значення визначає реакцію на основі суми останніх похибок, а похідне значення визначає реакцію на основі швидкості, з якою змінювалася похибка. Це три поширені математичні методи, які застосовуються для того, щоб робочий процес відповідав заданому значенню. У випадку LinuxCNC процесом, який ми хочемо контролювати, є фактичне положення осі, а заданим значенням є задане положення осі.

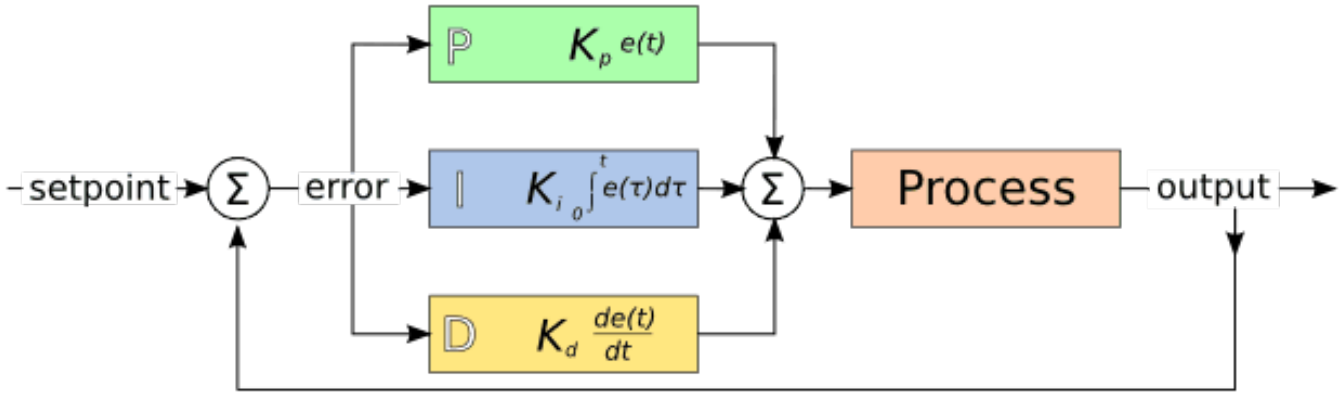


Figure 4.2: Петля PID-регулятора

«Налаштовуючи» три константи в алгоритмі PID-регулятора, регулятор може забезпечити дію управління, розроблену для конкретних вимог процесу. Реакцію регулятора можна описати з точки зору чутливості регулятора до помилки, ступеня перевищення регулятором заданого значення та ступеня коливання системи.

4.1.4.2 Пропорційний термін

Пропорційний член (іноді його називають коефіцієнтом підсилення) змінює вихідну величину пропорційно до поточного значення помилки. Високий коефіцієнт підсилення призводить до значної зміни вихідної величини при заданій зміні помилки. Якщо коефіцієнт підсилення занадто високий, система може стати нестабільною. Навпаки, малий коефіцієнт підсилення призводить до невеликої реакції вихідної величини на велику помилку вхідної величини. Якщо коефіцієнт підсилення занадто низький, дія системи управління може бути занадто слабкою при реагуванні на збурення в системі.

За відсутності збурень чисте пропорційне регулювання не стабілізується на цільовому значенні, а зберігає постійну похибку, яка є функцією пропорційного коефіцієнта підсилення та коефіцієнта підсилення процесу. Незважаючи на постійне зміщення, як теорія налаштування, так і промислова практика вказують на те, що саме пропорційний член повинен забезпечувати основну частину зміни вихідного сигналу.

4.1.4.3 Інтегральний член

Внесок інтегрального члена (іноді його називають скиданням) пропорційний як величині помилки, так і тривалості помилки. Сумуючи миттєву помилку за час (інтегруючи помилку), отримуємо накопичене зміщення, яке слід було виправити раніше. Потім накопичена помилка множиться на інтегральний коефіцієнт підсилення і додається до виходу контролера.

Інтегральний член (при додаванні до пропорційного члена) прискорює рух процесу до заданого значення і усуває залишкову помилку в стаціонарному режимі, яка виникає при використанні тільки пропорційного регулятора. Однак, оскільки інтегральний член реагує на накопичені помилки з минулого, він може призвести до того, що поточне значення перевищить задане значення (перетне задане значення і створить відхилення в іншому напрямку).

4.1.4.4 Похідний термін

Швидкість зміни помилки процесу обчислюється шляхом визначення нахилу помилки в часі (тобто її першої похідної за часом) і множення цієї швидкості зміни на коефіцієнт посилення похідної.

Похідна складова уповільнює швидкість зміни вихідного сигналу контролера, і цей ефект найбільш помітний поблизу заданого значення контролера. Отже, похідне регулювання використовується для зменшення величини перевищення, що створюється інтегральною складовою, та поліпшення загальної стабільності контролера-процесу.

4.1.4.5 Налаштування петлі

Якщо параметри PID-регулятора (коефіцієнти пропорційної, інтегральної та похідної складових) обрані неправильно, вхідний сигнал керованого процесу може бути нестабільним, тобто його вихідний сигнал розходиться, з коливанням або без нього, і обмежується лише насиченням або механічним пошкодженням. Налаштування контуру регулювання — це регулювання його параметрів (коефіцієнт підсилення/пропорційна смуга, інтегральний коефіцієнт підсилення/скидання, похідний коефіцієнт підсилення/швидкість) до оптимальних значень для бажаної реакції регулювання.

4.1.4.6 Ручне налаштування

Простий метод налаштування полягає в тому, щоб спочатку встановити значення I і D на нуль. Збільшуйте P, поки вихідний сигнал контуру не почне коливатися, після чого P слід встановити приблизно на половину цього значення для отримання відгуку типу «затухання чверті амплітуди». Потім збільшуйте I, поки будь-яке зміщення не буде виправлено за достатній для процесу час. Однак занадто велике значення I призведе до нестабільності. Нарешті, збільште D, якщо потрібно, до тих пір, поки контур не досягне прийнятної швидкості для досягнення свого опорного значення після порушення навантаження. Однак занадто велике значення D призведе до надмірної реакції та перевищення. Швидке налаштування контуру PID зазвичай призводить до невеликого перевищення, щоб швидше досягти заданого значення; однак деякі системи не можуть прийняти перевищення, і в цьому випадку потрібна «надмірно затухаюча» система із замкнутим контуром, яка вимагатиме значення P, значно меншого за половину значення P, що викликає коливання.

4.1.5 Планування траєкторії S-подібної кривої

Планування траєкторії за S-подібною кривою обмежує ривок (швидкість зміни прискорення) для забезпечення плавнішого руху. Це може зменшити вібрацію машини та покращити якість обробки поверхні, але вимагає налаштування додаткових параметрів.

4.1.5.1 Увімкнення

Встановити у INI-файлі:

```
[TRAJ]
PLANNER_TYPE = 1              # 0=b''тb''b''pb''b''ab''b''пb''b''eb''b''цb''b''ib''b''єb''b ↩
    ''пb''b''ob''b''дb''b''ib''b''бb''b''нb''b''иb''b''йb'' (b''зb''b''ab'' b''зb''b''ab''b' ↩
    'мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb''), 1=S-b' ↩
    'пb''b''ob''b''дb''b''ib''b''бb''b''нb''b''ab'' b''кb''b''pb''b''иb''b''вb''b''ab''
MAX_LINEAR_JERK = 1000.0      # b''Mb''b''ab''b''шb''b''иb''b''нb''b''нb''b''ib'' b''ob''b' ↩
    'дb''b''иb''b''нb''b''иb''b''цb''b''ib''/b''cb''^3

[JOINT_n]
MAX_JERK = 1000.0
```

S-подібне планування активне лише тоді, коли `PLANNER_TYPE = 1` та `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 Тюнінг

Почніть з консервативного значення ривка та поступово збільшуйте його:

```
MAX_JERK b’’≈b’’ 10 b’’дb’’b’’ob’’ 100 × MAX_ACCELERATION
```

Типові значення: 100-100 000 одиниць/с³ залежно від жорсткості машини та одиниць вимірювання (значення в мм зазвичай у 1000 разів більші, ніж значення в дюймах).

Збільшуйте значення MAX_LINEAR_JERK, доки рух не стане повільним або помилки слідування не збільшаться, потім трохи зменште. Тестуйте зі скоординованими рухами та дугами.

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

4.1.6 RTAI

Інтерфейс додатків реального часу (RTAI) використовується для забезпечення найкращої продуктивно в режимі реального часу (RT). Ядро з патчем RTAI дозволяє писати додатки з суворими обмеженнями за часом. RTAI дає можливість використовувати такі функції, як генерація програмних кроків, які вимагають точного часу.

4.1.6.1 ACPI

Розширений інтерфейс конфігурації та живлення (ACPI) має багато різних функцій, більшість з яких впливають на продуктивність RT (наприклад: управління живленням, вимкнення процесора, масштабування частоти процесора тощо). Ядро LinuxCNC (і, ймовірно, всі ядра з патчем RTAI) має вимкнену функцію ACPI. ACPI також відповідає за вимкнення системи після запуску процесу вимкнення, тому вам може знадобитися натиснути кнопку живлення, щоб повністю вимкнути комп’ютер. Група RTAI вдосконалює цю функцію в останніх версіях, тому ваша система LinuxCNC може вимкнутися самостійно.

4.1.7 Апаратні опції інтерфейсу комп’ютер/машина

4.1.7.1 litehm2/rv901t

Litehm2 — це незалежний від плати порт прошивки HostMot2 FPGA. Першою платою, яку він підтримує, є linsn rv901t, яка спочатку була створена як плата контролера світлодіодів, але завдяки наявним входам/виходам вона добре підходить для використання в якості контролера машини. Вона пропонує близько 80 портів вводу-виводу з 5-вольтовим буфером і може перемикатися між усіма входами і всіма виходами. Її також легко модифікувати, щоб розділити порти навпіл між входом і виходом. Rv901t підключається до комп’ютера через Gigabit або 100Mbit Ethernet.

Litehm2 базується на фреймворку LiteX, який підтримує широкий спектр плат FPGA. Наразі підтримується лише rv901t, але підтримка інших плат знаходиться в розробці.

Більше інформації можна знайти за адресою https://github.com/sensille/litehm2.

4.2 Тестування затримки

4.2.1 Що таке латентність?

Затримка — це час, який потрібен ПК, щоб зупинити поточну роботу і відповісти на зовнішній запит, наприклад, запустити один із періодичних потоків LinuxCNC у режимі реального часу. Чим менша затримка, тим швидше можна запускати потоки у режимі реального часу і тим плавнішим буде рух (а у випадку програмного крокування — ймовірно, швидшим).

Затримка набагато важливіша за швидкість процесора. Непомітний Pentium II, який реагує на переривання протягом 10 мікросекунд кожного разу, може дати кращі результати, ніж найновіший і найшвидший P4 Hyperthreading.

Процесор не є єдиним фактором, що впливає на затримку. Материнські плати, відеокарти, USB-порти та ряд інших факторів також можуть впливати на затримку. Найкращий спосіб з'ясувати, з чим ви маєте справу, — це провести тест на затримку.

Generating step pulses in software has one very big advantage - it's free. Just about every PC that has a parallel port is capable of outputting step pulses that are generated by the software.

However, software step pulses also have some disadvantages:

- обмежена максимальна швидкість кроку
- тремтіння у згенерованих імпульсах
- завантажує процесор

4.2.2 Тести на затримку

LinuxCNC містить кілька тестів затримки. Усі вони надають еквівалентну інформацію. Виконання цих тестів допоможе визначити, чи підходить комп'ютер для керування верстатом з CNC.

Note
Не запускайте LinuxCNC або StepConf під час виконання тесту затримки.

4.2.2.1 Тест на затримку

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
```

Це розпочне тест затримки з періодом базового потоку 25 мкс та періодом сервопотоку 1 мс. Періоди можна вказати в командному рядку:

```
latency-test 50000 1000000
```

Це розпочне тест затримки з періодом базового потоку 50 мкс та періодом сервопотоку 1 мс.

Щоб переглянути доступні опції, у командному рядку введіть:

```
latency-test -h
```

Після запуску тесту затримки ви повинні побачити щось подібне:

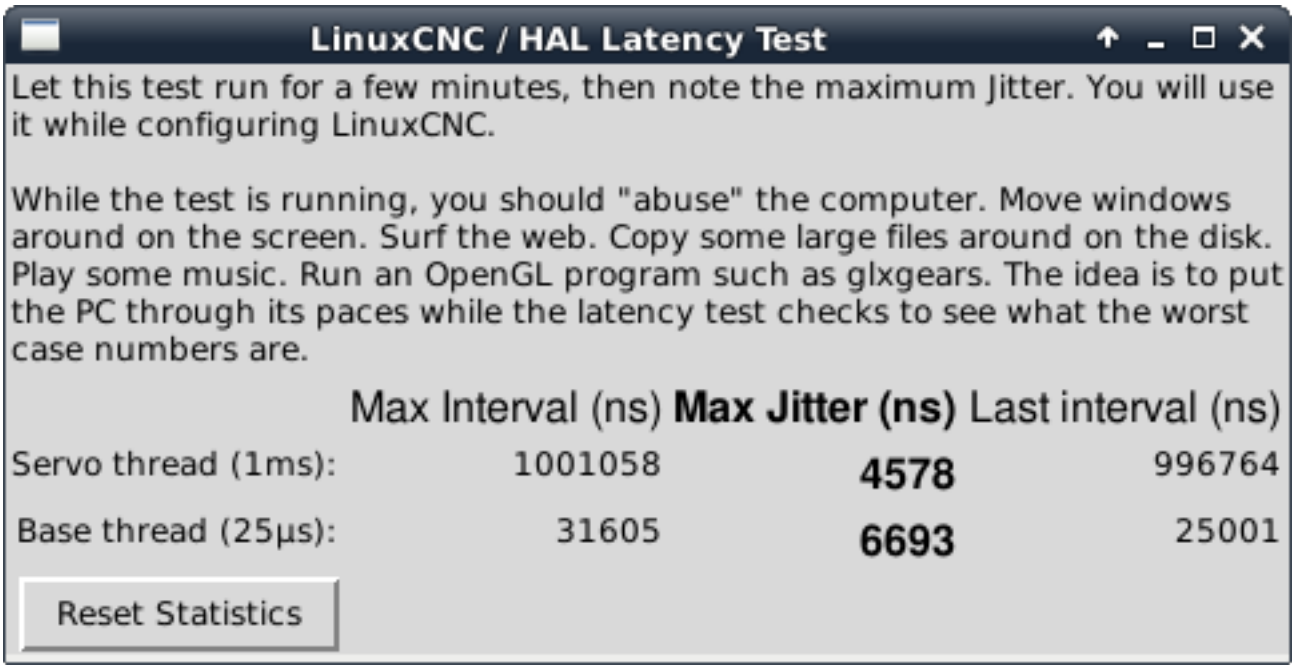


Figure 4.3: Тест затримки HAL

Під час виконання тесту слід «зловживати» комп'ютером. Переміщайте вікна по екрану. Переглядайте веб-сторінки. Копіюйте великі файли на диск. Відтворюйте музику. Запустіть програму OpenGL, наприклад glxgears. Ідея полягає в тому, щоб випробувати ПК на міцність, поки тест на затримку перевіряє, які показники є найгіршими.

Важливим числом для крокування програмного забезпечення є «максимальний джиттер» базового потоку. У наведеному вище прикладі це 6693 наносекунди (нс) або 6,693 мікросекунди (мкс). Запишіть це число і введіть його в StepConf, коли буде запропоновано.

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 отримання додаткової інформації про налаштування крокового двигуна див. розділ Налаштування крокового двигуна.

Tip
Додаткові інструменти командного рядка доступні для перевірки затримки, коли LinuxCNC не запущено.

4.2.2.2 Графік затримки

latency-plot створює стрічкову діаграму для базового та серво-потокового потоків. Може бути корисним побачити піки затримки під час запуску або використання інших програм. Використання:

```
latency-plot --help

b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яв'':
      latency-plot --help | -?
      latency-plot --hal [Options]

b''Пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ив'':
      --base ns  (b''бb''b''ab''b''зb''b''ob''b''вb''b''ив''b''йb'' b''іb''b''нb''b''тb''b' ↩
          'eb''b''pb''b''вb''b''ab''b''лb'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb'' b' ↩
          'вb'' b''нb''b''ab''b''нb''b''ob''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b' ↩
          'ab''b''хb'', b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b' ↩
          'вb''b''ab''b''нb''b''нb''b''яb''b''мb'':   25000)
      --servo ns (b''ib''b''нb''b''тb''b''eb''b''pb''b''вb''b''ab''b''лb'' b''cb''b''eb''b' ↩
          'pb''b''вb''b''ob''b''пb''b''ob''b''тb''b''ob''b''кb''b''уb'' b''вb'' b''нb''b' ↩
          'ab''b''нb''b''ob''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', b' ↩
          'зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b' ↩
          'нb''b''нb''b''яв''b''мb'': 1000000)
      --time ms  (b''ib''b''нb''b''тb''b''eb''b''pb''b''вb''b''ab''b''лb'' b''зb''b''вb''b' ↩
          'ib''b''тb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яв'' b''вb'' b''мb''b''ib''b' ↩
          'лb''b''ib''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', b''зb''b' ↩
          'ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b' ↩
          'нb''b''яв''b''мb'': 1000)
      --relative (b''вb''b''ib''b''дb''b''нb''b''ob''b''cb''b''нb''b''ив''b''йb'' b''чb''b' ↩
          'ab''b''cb'' b''зb''b''ab'' b''гb''b''ob''b''дb''b''ив''b''нb''b''нb''b''ив''b' ↩
          'кb''b''ob''b''мb'' (b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b' ↩
          'уb''b''вb''b''ab''b''нb''b''нb''b''яв''b''мb''))
      --actual   (b''фb''b''ab''b''кb''b''тb''b''ив''b''чb''b''нb''b''ив''b''йb'' b''чb''b' ↩
          'ab''b''cb'' b''зb''b''ab'' b''гb''b''ob''b''дb''b''ив''b''нb''b''нb''b''ив''b' ↩
          'кb''b''ob''b''мb'')
```

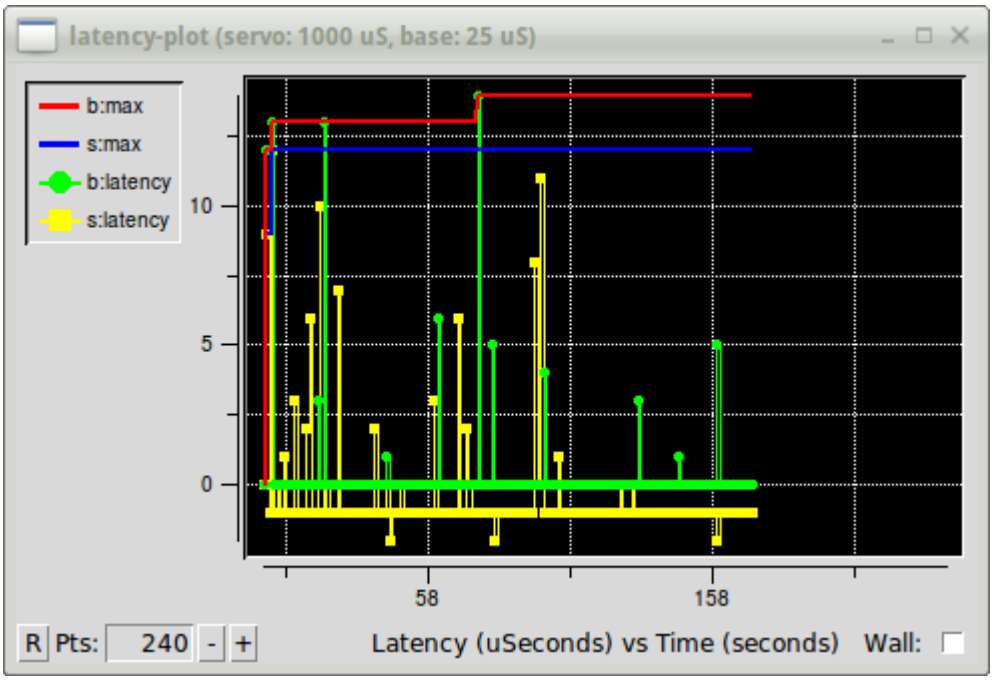


Figure 4.4: `latency-plot` Вікно

4.2.2.3 Гістограма затримки

Гістограма затримки застосунку відображає гістограму затримки (тремтіння) для базового та серво-потоків.

```
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
   latency-histogram --help | -?
   latency-histogram [Options]

b''Ob''b''пb''b''цb''b''іb''b''їb'':
  --base      ns   (b''бb''b''ab''b''зb''b''ob''b''вb''b''ив''b''йb'' b''іb''b''нb''b''тb'' ↩
      b''eb''b''pb''b''вb''b''ab''b''лb'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb'' b''вb'' ↩
       b''нb''b''ab''b''нb''b''ob''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', ↩
       b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb'' ↩
      b''нb''b''яb''b''мb'':   25000, b''мb''b''іb''b''нb''b''іb''b''мb''b''уb''b''мb'':   ↩
      5000)
  --servo     ns   (b''іb''b''нb''b''тb''b''eb''b''pb''b''вb''b''ab''b''лb'' b''cb''b''eb'' ↩
      b''pb''b''вb''b''ob''b''пb''b''ob''b''тb''b''ob''b''кb''b''уb'' b''вb'' b''нb''b''ab'' ↩
      b''нb''b''ob''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', b''зb''b''ab'' ↩
       b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b ↩
      ''мb'': 1000000, b''мb''b''іb''b''нb''b''іb''b''мb''b''уb''b''мb'': 25000)
  --bbinsize  ns   (b''бb''b''ab''b''зb''b''ob''b''вb''b''ив''b''йb'' b''pb''b''ob''b''зb'' ↩
      b''мb''b''іb''b''pb'' b''бb''b''іb''b''нb''b''ab'' b''вb'' b''нb''b''ab''b''нb''b' ↩
      'ob''b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', b''зb''b''ab'' b''зb''b ↩
      ''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'':  ↩
      100
  --sbinsize  ns   (b''pb''b''ob''b''зb''b''мb''b''іb''b''pb'' b''cb''b''eb''b''pb''b''вb'' ↩
      b''ob''b''бb''b''іb''b''нb''b''ab'' b''вb'' b''нb''b''ab''b''нb''b''ob''b''cb''b''eb'' ↩
      b''кb''b''уb''b''нb''b''дb''b''ab''b''хb'', b''зb''b''ab'' b''зb''b''ab''b''мb''b' ↩
      'ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'': 100
  --bbins     n    (b''бb''b''ab''b''зb''b''ob''b''вb''b''іb''  b''бb''b''іb''b''нb''b' ↩
      'ив'', b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b ↩
      ''нb''b''нb''b''яb''b''мb'': 200
```

```
--sbins     n    (b''cb''b''eb''b''pb''b''вb''b''ob''b''бb''b''ib''b''нb''b''иb'', b' ↩
    'зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''яb''b''мb'': 200
--logscale  0|1  (b''лb''b''ob''b''гb''b''ab''b''pb''b''иb''b''фb''b''мb''b''ib''b''чb''b ↩
    ''нb''b''ab'' b''шb''b''кb''b''ab''b''лb''b''ab'' b''ob''b''cb''b''ib'' Y, b''зb''b' ↩
    'ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b' ↩
    'яb''b''мb'': 1)
--text      note (b''дb''b''ob''b''дb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ab'' b''пb'' ↩
    b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'', b''зb''b''ab'' b''зb''b''ab''b' ↩
    'мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'': "" )
--show          (b''пb''b''ob''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''кb''b''ib'' ↩
    b''лb''b''ьb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''нb''b''eb'' b''вb''b''ib''b''дb'' ↩
    b''ob''b''бb''b''pb''b''ab''b''жb''b''eb''b''нb''b''иb''b''хb'' b''бb''b''ib''b''нb''b ↩
    ''ib''b''вb'')
--nobase         (b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''cb''b''eb''b''pb''b''вb'' ↩
    b''ob''-b''пb''b''ob''b''тb''b''ib''b''кb'')
--verbose        (b''пb''b''pb''b''ob''b''гb''b''pb''b''eb''b''cb'' b''ib'' b''нb''b' ↩
    'ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'')
--nox           (b''бb''b''eb''b''зb'' b''гb''b''pb''b''ab''b''фb''b''ib''b''чb''b''нb'' ↩
    b''ob''b''гb''b''ob'' b''ib''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b''cb''b ↩
    ''yb'', b''вb''b''ib''b''дb''b''ob''b''бb''b''pb''b''ab''b''жb''b''eb''b''нb''b''нb''b ↩
    ''яb'' elapsed,min,max,sdev b''дb''b''лb''b''яb'' b''кb''b''ob''b''жb''b''нb''b''ob''b ↩
    ''гb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''yb'')
```

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.

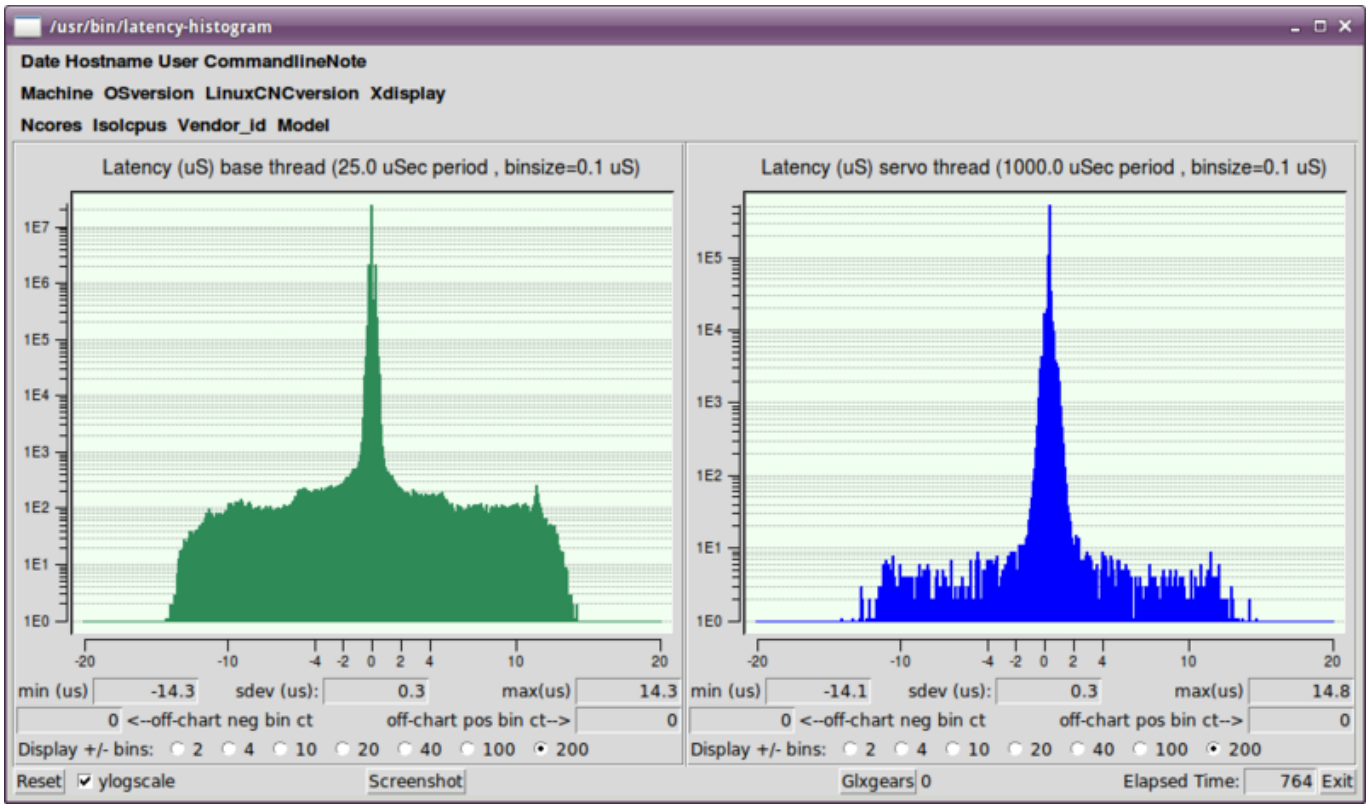


Figure 4.5: `latency-histogram` Вікно

4.2.3 Налаштування затримки

LinuxCNC може працювати на багатьох різних апаратних платформах та з багатьма різними ядрами реального часу, і всі вони можуть отримати користь від налаштування для оптимальної затримки.

Основною метою налаштування системи для LinuxCNC є резервування процесора для виключного використання завданнями LinuxCNC в режимі реального часу, щоб інші завдання (як програми користувачів, так і потоки ядра) не перешкоджали доступу LinuxCNC до цього процесора.

Якщо певні параметри налаштування вважаються загальнокорисними, LinuxCNC виконує це налаштування автоматично під час запуску, але багато параметрів налаштування є специфічними для конкретної машини і не можуть бути виконані автоматично. Особа, яка встановлює LinuxCNC, повинна експериментальним шляхом визначити оптимальні параметри налаштування для своєї системи.

4.2.3.1 Налаштування BIOS для зменшення затримки

BIOS ПК дуже різняться за своєю затримкою.

Налаштування BIOS є досить трудомістким процесом, оскільки вам доведеться перезавантажувати комп’ютер, вносити невеликі зміни в BIOS, завантажувати Linux і виконувати тест затримки (що може зайняти багато часу), щоб побачити, як зміни в BIOS вплинули на роботу системи. Потім повторіть ці дії для всіх інших налаштувань BIOS, які ви хочете випробувати.

Оскільки всі BIOS різні та нестандартні, надання детального посібника з налаштування BIOS не є практичним. Загалом, деякі речі, які можна спробувати налаштувати в BIOS:

- Вимкніть ACPI, APM та будь-які інші функції енергозбереження. Це включає все, що пов’язано з енергозбереженням, призупиненням, станами сну процесора, масштабуванням частоти процесора тощо.
- Вимкніть режим ”турбо” процесора.
- Вимкнути гіперпоточність процесора.
- Вимкнути (або іншим чином керувати) перериваннями системного керування (SMI).
- Вимкніть будь-яке обладнання, яке ви не збираєтеся використовувати.

4.2.3.2 Налаштування Preempt-RT на затримку

Ядро Preempt-RT може отримати користь від налаштування, щоб забезпечити найкращу затримку для LinuxCNC. Налаштування можна виконати через командний рядок ядра, sysctl, та через файли в `/proc` та `/sys`.

Деякі параметри налаштування, на які варто звернути увагу:

Командний рядок ядра
Деталі тут: https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

- `isolcpus`: Запобігти використанню цих процесорів більшістю процесів, що не належать до LinuxCNC, залишаючи більше процесорного часу доступним для LinuxCNC.
- `irqaffinity`: Виберіть, які процесори обслуговують переривання, щоб процесори, зарезервован для роботи в реальному часі LinuxCNC, не виконували це завдання.
- `rcu_nocbs`: Запобігти виконанню зворотних викликів RCU на цих процесорах.
- `rcu_nocb_poll`: Опитувати зворотні виклики RCU замість використання режиму сну/пробуджен
- `nohz_full`: Вимкніть тактування на цих процесорах.

Sysctl
Деталі тут: https://www.kernel.org/doc/html/latest/scheduler/sched-rt-group.html

- `sysctl.kernel.sched_rt_runtime_us`: Встановіть значення -1, щоб зняти обмеження на час, який можуть використовувати завдання в реальному часі.

4.3 Налаштування крокового двигуна

4.3.1 Отримання максимальної користі від поетапного програмного забезпеч

Генерація імпульсів кроку в програмному забезпеченні має одну дуже велику перевагу - це безкоштовно. Практично кожен ПК має паралельний порт, здатний виводити імпульси кроку, що генеруються програмним забезпеченням. Однак програмні імпульси кроку також мають деякі недоліки:

- обмежена максимальна швидкість кроку
- тремтіння у згенерованих імпульсах
- завантажує процесор

У цьому розділі наведено кілька кроків, які допоможуть вам отримати найкращі результати від кроків, згенерованих програмним забезпеченням.

4.3.1.1 Виконайте тест затримки

Процесор не є єдиним фактором, що визначає затримку. Материнські плати, графічні карти, USB-порти та багато інших речей можуть погіршувати її. Найкращий спосіб дізнатися, чого очікувати від ПК, — це провести тести затримки RT.

Виконайте тест на затримку, як описано в розділі Тест на затримку.

Під час виконання тесту слід «зловживати» комп'ютером. Переміщайте вікна по екрану. Переглядайте веб-сторінки. Копіюйте великі файли на диск. Відтворюйте музику. Запустіть програму OpenGL, наприклад glxgears. Ідея полягає в тому, щоб випробувати ПК на міцність, поки тест на затримку перевіряє, які показники є найгіршими.

Останнє число в стовпці з назвою «Max Jitter» є найважливішим. Запишіть його — воно знадобиться вам пізніше. Воно містить найгірше значення затримки за весь час виконання тесту. У наведеному вище прикладі це 6693 наносекунди, або 6,69 мікросекунди, що є відмінним результатом. Однак приклад працював лише кілька секунд (він друкує один рядок щосекунди). Ви повинні запускати тест принаймні на кілька хвилин; іноді найгірший випадок затримки трапляється не дуже часто або трапляється тільки тоді, коли ви виконуєте певну дію. У мене була одна материнська плата Intel, яка більшу частину часу працювала досить добре, але кожні 64 секунди мала дуже погану затримку в 300 мкс. На щастя, це можна виправити, див. Виправлення проблем SMI на LinuxCNC Wiki

То що означають ці результати? Якщо ваше значення «Max Jitter» менше ніж 15-20 мікросекунд (15000-20000 наносекунд), комп'ютер повинен дати дуже хороші результати з програмним крокуванням. Якщо максимальна затримка становить близько 30-50 мікросекунд, ви все одно можете отримати хороші результати, але максимальна швидкість кроку може бути трохи розчаровуючою, особливо якщо ви використовуєте мікрокрокування або маєте гвинти з дуже дрібним кроком. Якщо значення становить 100 мкс або більше (100,000 наносекунд), то ПК не підходить для програмного крокування. Числа понад 1 мілісекунду (1,000,000 наносекунд) означають, що ПК не підходить для LinuxCNC, незалежно від того, чи використовуєте ви програмне крокування чи ні.

Зверніть увагу, що якщо ви отримали високі показники, можливо, є способи їх поліпшити. Наприклад, один ПК мав дуже погану затримку (кілька мілісекунд) при використанні вбудованої відеокарти. Але відеокарта за 5 доларів вирішила цю проблему — LinuxCNC не вимагає найсучаснішого обладнання.

4.3.1.2 З'ясуйте, чого очікують ваші накопичувачі

Різні марки крокових приводів мають різні вимоги до часу на входах кроку та напрямку. Тому вам потрібно знайти (або погуглити) технічні характеристики вашого приводу.

З інструкції до Gecko G202:

```
b''Чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''ab'' b''кb''b''pb''b''ob''b''кb''b''yb'': b' ↩
    'вb''b''ib''b''дb'' 0 b''дb''b''ob'' 200 b''кb''b''Гb''b''цb''
b''Чb''b''ab''b''cb'' b''кb''b''pb''b''ob''b''кb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' "0": b''xb''b''вb''. 0,5 b''мb''b ↩
    ''кb''b''cb'' (b''кb''b''pb''b''ob''b''кb'' b''пb''b''ob'' b''cb''b''пb''b''ab''b''дb''b ↩
    ''нb''b''ob''b''мb''b''yb'' b''фb''b''pb''b''ob''b''нb''b''тb''b''yb'')
b''Чb''b''ab''b''cb'' b''кb''b''pb''b''ob''b''кb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' "1": b''xb''b''вb''. 4,5 b''мb''b ↩
    ''кb''b''cb''
b''Нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb'' b' ↩
    'нb''b''ab''b''пb''b''pb''b''яb''b''мb''b''кb''b''yb'': b''xb''b''вb''. 1 b''мb''b''кb'' ↩
    b''cb'' (b''xb''b''вb''. b''чb''b''ab''b''cb'' b''yb''b''тb''b''pb''b''иb''b''мb''b' ↩
    'ab''b''нb''b''нb''b''яb'' b''пb''b''ib''b''cb''b''лb''b''яb'' b''фb''b''pb''b''ob''b' ↩
    'нb''b''тb''b''yb'' b''кb''b''pb''b''ob''b''кb''b''yb'' 20 b''мb''b''кb''b''cb'')
```

З інструкції до Gecko G203V:

```
b''Чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''ab'' b''кb''b''pb''b''ob''b''кb''b''yb'': b' ↩
    'вb''b''ib''b''дb'' 0 b''дb''b''ob'' 333 b''кb''b''Гb''b''цb''
b''Чb''b''ab''b''cb'' b''кb''b''pb''b''ob''b''кb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' "0": b''xb''b''вb''. 2,0 b''мb''b ↩
    ''кb''b''cb'' (b''кb''b''pb''b''ob''b''кb'' b''нb''b''ab'' b''нb''b''ab''b''pb''b''ob''b ↩
    ''cb''b''тb''b''ab''b''юb''b''чb''b''ob''b''мb''b''yb'' b''фb''b''pb''b''ob''b''нb''b' ↩
    'тb''b''ib'')
b''Чb''b''ab''b''cb'' b''кb''b''pb''b''ob''b''кb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' "1": b''xb''b''вb''. 1,0 b''мb''b ↩
    ''кb''b''cb''

b''Нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb'' b' ↩
    'нb''b''ab''b''пb''b''pb''b''яb''b''мb''b''кb''b''yb'':
    200 b''нb''b''cb'' (0,2 b''мb''b''кb''b''cb'') b''дb''b''ob'' b''нb''b''ab''b''pb''b' ↩
        'ob''b''cb''b''тb''b''ab''b''юb''b''чb''b''ob''b''гb''b''ob'' b''фb''b''pb''b''ob''b ↩
        ''нb''b''тb''b''yb'' b''cb''b''тb''b''yb''b''пb''b''ib''b''нb''b''чb''b''ab''b''cb'' ↩
        b''тb''b''ob''b''гb''b''ob'' b''ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b' ↩
        'yb''
    200 b''нb''b''cb'' (0,2 b''мb''b''кb''b''cb'') b''yb''b''тb''b''pb''b''иb''b''мb''b' ↩
        'ab''b''нb''b''нb''b''яb'' b''пb''b''ib''b''cb''b''лb''b''яb'' b''нb''b''ab''b''pb'' ↩
        b''ob''b''cb''b''тb''b''ab''b''юb''b''чb''b''ob''b''гb''b''ob'' b''фb''b''pb''b' ↩
        'ob''b''нb''b''тb''b''yb'' b''cb''b''тb''b''yb''b''пb''b''ib''b''нb''b''чb''b''ab''b ↩
        ''cb''b''тb''b''ob''b''гb''b''ob'' b''ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb'' ↩
        b''yb''
```

З технічного опису Xylotex:

```
b''Mb''b''ib''b''нb''b''ib''b''мb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''чb''b' ↩
    'ab''b''cb'' b''нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''яb'' DIR b''пb''b''eb''b''pb''b''eb''b''дb'' b''пb''b''eb''b''pb''b''eb''b''дb'' ↩
    b''нb''b''ib''b''мb'' b''фb''b''pb''b''ob''b''нb''b''тb''b''ob''b''мb'' b''ib''b''мb''b' ↩
    'пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' STEP 200 b''нb''b''cb'' b''Mb''b''ib''b''нb''b' ↩
    'ib''b''мb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb''
b''чb''b''ab''b''cb'' b''yb''b''тb''b''pb''b''иb''b''мb''b''ab''b''нb''b''нb''b''яb'' DIR b ↩
    ''пb''b''ib''b''cb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b' ↩
    'ob''b''гb''b''ob'' b''фb''b''pb''b''ob''b''нb''b''тb''b''yb'' b''ib''b''мb''b''пb''b' ↩
    'yb''b''лb''b''ьb''b''cb''b''yb'' STEP 200 b''нb''b''cb''
b''Mb''b''ib''b''нb''b''ib''b''мb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''чb''b' ↩
    'ab''b''cb'' b''вb''b''иb''b''cb''b''ob''b''кb''b''ob''b''гb''b''ob'' b''pb''b''ib''b' ↩
    'вb''b''нb''b''яb'' b''ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' STEP 2,0 b' ↩
    'мb''b''кb''b''cb''
b''Mb''b''ib''b''нb''b''ib''b''мb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''чb''b' ↩
    'ab''b''cb'' b''нb''b''иb''b''зb''b''ьb''b''кb''b''ob''b''гb''b''ob'' b''pb''b''ib''b' ↩
    'вb''b''нb''b''яb'' b''ib''b''мb''b''пb''b''yb''b''лb''b''ьb''b''cb''b''yb'' STEP 1,0 b' ↩
    'мb''b''кb''b''cb''
b''Kb''b''pb''b''ob''b''кb'' b''вb''b''ib''b''дb''b''бb''b''yb''b''вb''b''ab''b''єb''b' ↩
    'тb''b''ьb''b''cb''b''яb'' b''нb''b''ab'' b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b' ↩
    'ьb''b''ob''b''мb''b''yb'' b''фb''b''pb''b''ob''b''нb''b''тb''b''ib''
```

Як тільки знайдете числа, запишіть їх також — вони знадобляться вам на наступному кроці.

4.3.1.3 Виберіть свій БАЗОВИЙ_ПЕРІОД

BASE_PERIOD — це «серцебиття» вашого комп’ютера LinuxCNC. Кожного періоду програмний генератор кроків вирішує, чи настав час для чергового імпульсу кроку. Коротший період дозволить вам генерувати більше імпульсів за секунду, в межах обмежень. Але якщо ви виберете занадто короткий період, ваш комп’ютер витрачатиме стільки часу на генерацію імпульсів кроків, що все інше сповільниться до повзучої швидкості або навіть зависне. Затримка і вимоги до крокового

приводу впливають на найкоротший період, який ви можете використовувати, як ми побачимо за хвилину.

Спочатку розглянемо приклад Gecko. G202 може обробляти імпульси кроку, які мають низький рівень протягом 0,5 мкс і високий рівень протягом 4,5 мкс, для цього потрібно, щоб вивід напрямку був стабільним за 1 мкс до спаду фронту і залишався стабільним протягом 20 мкс після спаду фронту. Найдовша вимога до часу — це час утримання 20 мкс. Простим підходом було б встановити період на 20 мкс. Це означає, що всі зміни на лініях STEP і DIR розділені 20 мкс. Все добре, правда?

Неправильно! Якби затримка була нульовою, то всі краї були б розділені на 20 мкс, і все було б добре. Але всі комп'ютери мають певну затримку. Затримка означає запізнення. Якщо комп'ютер має затримку 11 мкс, це означає, що іноді програмне забезпечення працює на 11 мкс пізніше, ніж повинно. Якщо один запуск програмного забезпечення запізнюється на 11 мкс, а наступний відбувається вчасно, затримка від першого до другого становить лише 9 мкс. Якщо перший запуск генерував імпульс кроку, а другий змінив біт напрямку, ви просто порушили вимогу часу утримання G202 в 20 мкс. Це означає, що ваш привід міг зробити крок у неправильному напрямку, і ваша деталь буде неправильного розміру.

Найгірше в цій проблемі те, що вона може траплятися дуже рідко. У найгіршому випадку затримки можуть траплятися лише кілька разів на хвилину, а ймовірність того, що затримка трапиться саме в той момент, коли двигун змінює напрямок руху, є низькою. Отже, ви отримуєте дуже рідкісні помилки, які час від часу псують деталь і які неможливо усунути.

Найпростіший спосіб уникнути цієї проблеми — вибрати BASE_PERIOD, що дорівнює сумі найдовшого часу, необхідного для роботи вашого накопичувача, та найгіршого випадку затримки вашого комп'ютера. Якщо ви використовуєте Gecko з вимогою до часу утримання 20 мкс, а тест затримки показав, що максимальна затримка становить 11 мкс, то якщо ви встановите BASE_PERIOD на 20+11 = 31 мкс (31000 наносекунд у файлі ini), ви гарантовано виконаєте вимоги до часу роботи приводу.

Але є компроміс. Для створення імпульсу кроку потрібно щонайменше два періоди. Один для запуску імпульсу, а другий для його завершення. Оскільки період становить 31 мкс, для створення імпульсу кроку потрібно 2x31 = 62 мкс. Це означає, що максимальна частота кроку становить лише 16 129 кроків на секунду. Не дуже добре. (Але не здавайтеся, у наступному розділі ми ще трохи попрацюємо над налаштуваннями)

Для Xylotex час налаштування та утримання дуже короткий, по 200 нс (0,2 мкс) кожен. Найдовший час — це 2 мкс високого рівня. Якщо у вас затримка 11 мкс, то ви можете встановити BASE_PERIOD на рівні 11+2=13 мкс. Позбавлення від довгого часу утримання 20 мкс дійсно допомагає! З періодом 13 мкс повний крок займає 2x13 = 26 мкс, а максимальна швидкість кроку становить 38 461 крок на секунду!

Але ще рано радіти. Зверніть увагу, що 13 мкс — це дуже короткий проміжок часу. Якщо ви спробуєте запускати генератор кроків кожні 13 мкс, може не вистачити часу на виконання інших операцій, і ваш комп'ютер зависне. Якщо ви прагнете досягти періоду менше 25 мкс, слід почати з 25 мкс або більше, запустити LinuxCNC і подивитися, як все реагує. Якщо все гаразд, можна поступово зменшувати період. Якщо курсор миші починає гальмувати, а все інше на ПК сповільнюється, ваш період є трохи занадто коротким. Поверніться до попереднього значення, яке дозволяло комп'ютеру працювати безперебійно.

У цьому випадку припустимо, що ви почали з 25 мкс, намагаючись досягти 13 мкс, але виявили, що межа становить приблизно 16 мкс — при меншому значенні комп'ютер не реагує належним чином. Тому ви використовуєте 16 мкс. З періодом 16 мкс і затримкою 11 мкс найкоротший час виводу буде 16-11 = 5 мкс. Приводу потрібно лише 2 мкс, тому у вас є певний запас. Запас є корисним — ви не хочете втрачати кроки через те, що занадто скоротили час.

Яка максимальна швидкість кроку? Пам'ятайте, для кроку потрібно два періоди. Ви встановили період 16 мкс, тож крок займає 32 мкс. Це дає непогані 31 250 кроків за секунду.

4.3.1.4 Використовуйте steplen, stepspace, dirsetup та/або dirhold

В останньому розділі ми досягли періоду 16 мкс і максимальної швидкості 31 250 кроків на секунду для приводу Xylotex. Але Gecko застряг на 31 мкс і не надто приємних 16 129 кроків на секунду. Приклад Xylotex є найкращим, який ми можемо зробити. Але Gecko можна вдосконалити.

Проблема з G202 полягає у вимозі часу утримання 20 мкс. Це, а також затримка 11 мкс, змушує нас використовувати повільний період 31 мкс. Але генератор кроків програмного забезпечення LinuxCNC має деякі параметри, які дозволяють збільшити час від одного періоду до декількох. Наприклад, якщо steplen змінюється з 1 на 2, то між початком і кінцем імпульсу кроку буде два періоди. Аналогічно, якщо dirhold змінюється з 1 на 3, між імпульсом кроку і зміною напрямку контакту буде принаймні три періоди.

Якщо ми можемо використовувати dirhold для дотримання вимоги до часу утримання 20 мкс, то наступним за тривалістю є час високого рівня 4,5 мкс. Додайте затримку 11 мкс до часу високого рівня 4,5 мкс, і ви отримаєте мінімальний період 15,5 мкс. Коли ви спробуєте 15,5 мкс, ви побачите, що комп'ютер працює повільно, тому ви зупинитеся на 16 мкс. Якщо ми залишимо dirhold на 1 (за замовчуванням), то мінімальний час між кроком і напрямком буде періодом 16 мкс мінус затримка 11 мкс = 5 мкс, що недостатньо. Нам потрібно ще 15 мкс. Оскільки період становить 16 мкс, нам потрібно ще один період. Тому ми змінюємо dirhold з 1 на 2. Тепер мінімальний час від кінця імпульсу кроку до зміни напрямку становить 5+16=21 мкс, і нам не потрібно турбуватися про те, що Gecko рухатиметься в неправильному напрямку через затримку.

Якщо комп'ютер має затримку 11 мкс, то комбінація базового періоду 16 мкс і значення dirhold 2 гарантує, що ми завжди будемо відповідати вимогам Gecko щодо синхронізації. Для нормального крокування (без зміни напрямку) збільшення значення dirhold не має ніякого впливу. Для виконання кожного кроку потрібно два періоди загальною тривалістю 32 мкс, і ми отримуємо ту саму швидкість 31 250 кроків на секунду, яку ми отримали з Xylotex.

Значення затримки 11 мкс, яке використовується в цьому прикладі, є дуже хорошим. Якщо ви будете працювати з цими прикладами з більшою затримкою, наприклад 20 або 25 мкс, максимальна частота кроків як для Xylotex, так і для Gecko буде нижчою. Але для обчислення оптимального значення BASE_PERIOD та для налаштування dirhold або інших параметрів генератора кроків застосовуються ті самі формули.

4.3.1.5 Без здогадок!

Для швидкої та надійної програмної системи крокового двигуна не можна просто вгадувати періоди та інші параметри конфігурації. Необхідно провести вимірювання на комп'ютері та виконати розрахунки, щоб переконатися, що приводи отримують необхідні сигнали.

Щоб полегшити розрахунки, я створив таблицю Open Office Калькулятор синхронізації кроків. Ви вводите результат тесту затримки і вимоги до синхронізації крокового приводу, а таблиця обчислює оптимальний BASE_PERIOD. Далі ви перевіряєте період, щоб переконатися, що він не сповільнить роботу вашого ПК і не заблокує його. Нарешті, ви вводите фактичний період, і таблиця покаже вам налаштування параметрів stepgen, необхідні для задоволення вимог до синхронізації вашого приводу. Вона також обчислює максимальну швидкість кроку, яку ви зможете генерувати.

Я додав кілька елементів до електронної таблиці для розрахунку максимальної швидкості та електричних розрахунків крокового двигуна.

4.4 Конфігурація INI

4.4.1 Компоненти INI-файлу

Типовий INI-файл має досить просту структуру, яка включає;

- коментарі
- розділи
- змінні

Кожен із цих елементів розділений на окремі рядки. Кожен символ кінця рядка або нового рядка створює новий елемент.

4.4.1.1 Коментарі

Коментар починається зі знака ; або #. Коли програма для читання INI-файлів бачить один із цих знаків на початку рядка, решта рядка ігнорується програмним забезпеченням. Коментарі можна використовувати для опису функції елемента INI.

```
; b''Цb''b''eb'' b''фb''b''ab''b''йb''b''лb'' b''кb''b''ob''b''нb''b''фb''b''іb''b''гb''b' ↩
    'уb''b''pb''b''ab''b''цb''b''іb''b''їb'' b''мb''b''ob''b''гb''b''ob'' b''мb''b''лb''b' ↩
    'иb''b''нb''b''ab''.
# b''Яb'' b''нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''вb''b''ab''b''вb'' b''йb''b' ↩
    'ob''b''гb''b''ob'' 12 b''cb''b''іb''b''чb''b''нb''b''яb'' 2012 b''pb''b''ob''b''кb''b' ↩
    'уb''.
```

Коментарі також можна використовувати для «вимикання» змінної. Це спрощує вибір між різними змінними.

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

У цьому списку змінна DISPLAY буде встановлена на axis, оскільки інша змінна закомментована. Якщо хтось необережно редагує такий список і залишає дві рядки незакомментованими, буде використано перший з них.

Зверніть увагу, що всередині змінної символи ”#” та ”;” не позначають коментарі:

```
INCORRECT = value     # b''іb'' b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb''

# b''Пb''b''pb''b''ab''b''вb''b''иb''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''кb''b''ob''b' ↩
    'мb''b''eb''b''нb''b''тb''b''ab''b''pb''
CORRECT = value
```

4.4.1.2 Розділи

Пов’язані частини файлу INI розділені на секції. Назва секції вказується в дужках, наприклад: `[ЦЯ_СЕКЦІЯ]`. Порядок секцій не має значення. Секції починаються з назви секції і закінчуються наступною назвою секції.

LinuxCNC використовує такі розділи:

- [EMC] загальна інформація
- [DISPLAY] налаштування, пов’язані з графічним інтерфейсом користувача
- [FILTER] налаштування вхідного фільтра програм
- [RS274NGC] налаштування, що використовуються інтерпретатором G-коду
- [EMCMOT] налаштування, що використовуються контролером руху реального часу
- [TASK] налаштування, що використовуються контролером завдань

- [HAL] вказує файли .hal
- [HALUI] Команди MDI, що використовуються HALUI
- [APPLICATIONS] Інші програми, які потрібно розпочати від LinuxCNC
- [TRAJ] додаткові налаштування, що використовуються контролером руху реального часу
- [JOINT_n] індивідуальні суглобові змінні
- [AXIS_l] окремі змінні осі
- [KINS] кінематичні змінні
- [EMCIO] налаштування, що використовуються контролером вводу/виводу

4.4.1.3 Змінні

Змінна рядок складається з імені змінної, знака рівності (=), і значення. Все, починаючи з першого символу, що не є пробілом, після = і до кінця рядка передається як значення, тому ви можете вставляти пробіли в символи рядка, якщо хочете або потрібно. Ім'я змінної часто називають ключовим словом.

Приклад змінної

```
MACHINE = b''Mb''b''ob''b''яb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''ab''
```

Змінна рядок може бути розширена до декількох рядків за допомогою символу зворотного слеша (\). Допускається максимум `MAX_EXTEND_LINES` (==20). Після символу зворотного слеша не повинно бути пробілів.

Ідентифікатори розділів не можуть бути розширені до кількох рядків.

Змінна з розширенням лінії Приклад

```
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
```

Булеві змінні Логічні значення можуть мати одне з `TRUE`, `YES` або `1` для true/увімкнено та одне з `FALSE`, `NO` або `0` для false/вимкнено. Реєстр ігнорується.

У наступних розділах детально описано кожен розділ файлу конфігурації, використовуючи зразки значень для рядків конфігурації.

Змінні, що використовуються LinuxCNC, завжди повинні використовувати назви розділів та змінних, як показано.

4.4.1.4 Розділи та змінні на замовлення

Більшість зразків конфігурацій використовують користувацькі розділи та змінні, щоб розмістити всі налаштування в одному місці для зручності.

Щоб додати користувацьку змінну до існуючого розділу LinuxCNC, просто додайте змінну в цей розділ.

Приклад користувацької змінної, де змінній «TYPE» присвоюється значення «LINEAR», а змінній «SCALE» – значення «16000».

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

Щоб додати користувацький розділ із власними змінними, додайте розділ та змінні до INI-файлу.

Приклад користувацького розділу

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

Щоб використовувати користувацькі змінні у вашому HAL-файлі, вставте назву розділу та змінної замість значення.

Приклад HAL

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

Note
Значення, що зберігається у змінній, має відповідати типу, визначеному виводом компонента.

Щоб використовувати власні змінні в G-коді, використовуйте синтаксис глобальної змінної `#<_ini[sec`
Наступний приклад показує просту процедуру відліку осі Z для фрезерного верстата або фрези з використанням зондувальної пластини.

Приклад G-коду

```
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 Включити файли

INI-файл може включати вміст іншого файлу за допомогою директиви #INCLUDE.

#ВКЛЮЧИТИ Формат

```
#INCLUDE filename
```

Ім’я файлу можна вказати так:

- файл у тому ж каталозі, що й INI-файл
- файл, розташований відносно робочого каталогу
- абсолютне ім’я файлу (починається з /)
- ім’я файлу, відносне до домашнього каталогу користувача (починається з ~)

Підтримується кілька директив #INCLUDE.

#ВКЛЮЧИТИ Приклади

```
#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
```

Директиви #INCLUDE підтримуються тільки для одного рівня розширення — включений файл не може містити додаткові файли. Рекомендоване розширення файлу — «.inc». Не використовуйте розширення «.ini» для включених файлів.

4.4.2 Розділи INI-файлу

4.4.2.1 [EMC] Розділ

- `VERSION = 1.1` - Версія формату цієї конфігурації. Будь-яке значення, відмінне від 1.1, призведе до запуску перевірки конфігурації та спроби оновлення конфігурації до нового типу конфігурації спільних осей.
- `MACHINE = My Controller` - Це назва контролера, яка відображається у верхній частині більшості графічних інтерфейсів. Ви можете вказати тут будь-яке ім'я, головне, щоб воно займало один рядок.
- `DEBUG = 0` - Рівень налагодження 0 означає, що під час запуску LinuxCNC з terminal повідомлення не будуть виводитися. Прапори налагодження зазвичай корисні лише для розробників. Інші налаштування див. у файлі src/emc/nml_intf/debugflags.h.
- `RCS_DEBUG = 1` Показати повідомлення про налагодження RCS. За замовчуванням друкувати тільки помилки (1), якщо біти EMC_DEBUG_RCS та EMC_DEBUG_RCS у `DEBUG` не встановлені, інакше друкувати все (-1). Використовуйте це для вибору повідомлень про налагодження RCS. Дивіться src/libnml/rcs/rcs_print.hh для всіх прапорців MODE.
- `RCS_DEBUG_DEST = STDOUT` - як виводити повідомлення RCS_DEBUG (NULL, STDOUT, STDERR, FILE, LOGGER, MSGBOX).
- `RCS_MAX_ERR = -1` – Число, після якого помилки RCS більше не повідомляються (-1 = нескінченно).
- `NML_FILE = /usr/share/linuxcnc/linuxcnc.nml` – Встановіть це значення, якщо ви хочете використ файл конфігурації NML, який не є файлом за замовчуванням.

4.4.2.2 [DISPLAY] Розділ

Різні програми інтерфейсу користувача використовують різні опції, і не кожна опція підтримується кожним інтерфейсом користувача. Існує кілька інтерфейсів, таких як AXIS, GMOCCAPY, Touchy, QtDragon від QtVCP та Gscreen. AXIS — це інтерфейс для використання зі звичайним комп'ютером та монітором, Touchy — для використання з сенсорними екранами. GMOCCAPY можна використовувати обома способами, він також пропонує багато підключень для апаратного керування. Опис інтерфейсів наведено в розділі «Інтерфейси» посібника користувача.

- `DISPLAY = axis` - Ім'я файлу виконуваного файлу, що забезпечує інтерфейс користувача для використання. Основні допустимі варіанти (всі в нижньому регістрі): `axis`, `touchy`, `gmoccapy`, `gscreen`, `tklinuxcnc`, `qtvcp`, `qtvcp qtdragon` або `qtvcp qtplasmac`.
- `POSITION_OFFSET = RELATIVE` — система координат (`RELATIVE` або `MACHINE`), яка відображається на цифровому індикаторі положення (DRO) під час запуску інтерфейсу користувача. Система координат RELATIVE відображає поточні зміщення координат G92 та G5*x*.

- `POSITION_FEEDBACK = COMMANDED` - Значення координат (`COMMANDED` або `ACTUAL`), яке відображається на DRO при запуску інтерфейсу користувача. У AXIS це можна змінити в меню «View» (Вигляд). Положення COMMANDED — це положення, яке запитує LinuxCNC. Положення ACTUAL — це положення зворотного зв'язку двигунів, якщо вони мають зворотний зв'язок, як більшість сервосистем. Зазвичай використовується значення COMMANDED.
- `DRO_FORMAT_MM = %+08.6f` - Перезаписати форматування DRO за замовчуванням у метричному режимі (зазвичай 3 десяткові знаки, доповнені пробілами до 6 цифр зліва). У наведеному вище прикладі будуть додані нулі, відображено 6 десяткових цифр і примусово відображено знак + для додатних чисел. Форматування відповідає практиці Python: https://docs.python.org/2/-library/string.html#format-specification-mini-language . Якщо формат не може прийняти значення з плаваючою комою, буде згенеровано помилку.
- `DRO_FORMAT_IN = % 4.1f` - Перезаписати форматування DRO за замовчуванням в імперському режимі (зазвичай 4 десяткові знаки, доповнені пробілами до 6 цифр зліва). У наведеному вище прикладі буде відображатися тільки одна десяткова цифра. Форматування відповідає практиці Python: https://docs.python.org/2/library/string.html#format-specification-mini-language . Якщо формат не може прийняти значення з плаваючою комою, буде згенеровано помилку.
- `CONE_BASESIZE = .25` - Змінити розмір конуса/бази інструмента за замовчуванням, що дорівнює 0,5, на графічному дисплеї. Допустимі значення – від 0,025 до 2,0.
- `DISABLE_CONE_SCALING = TRUE` - Будь-яке непорожнє значення (включно з "0") замінить стандартну поведінку масштабування розміру конуса/інструмента з використанням розмірів поточної завантаже програми G-коду на графічному дисплеї.
- `MAX_FEED_OVERRIDE = 1.2` – Максимальне значення корекції подачі, яке може обрати користувач. 1.2 означає 120% від запрограмованої швидкості подачі.
- `MIN_SPINDLE_OVERRIDE = 0.5` – Мінімальне значення корекції швидкості шпинделя, яке може вибрати користувач. 0.5 означає 50% від запрограмованої швидкості шпинделя. (Використовується для встановлення мінімальної швидкості шпинделя.)
- `MIN_SPINDLE_0_OVERRIDE = 0.5` - Мінімальне перевищення швидкості шпинделя, яке може вибрати користувач. 0.5 означає 50% від запрограмованої швидкості шпинделя. (Використовується для встановлення мінімальної швидкості шпинделя.) На багатошпиндельних верстатах будуть записи для кожного номера шпинделя. Використовується тільки в інтерфейсах користувача на базі QtVCP.
- `MAX_SPINDLE_OVERRIDE = 1.0` – Максимальне значення корекції швидкості шпинделя, яке може вибрати користувач. 1.0 означає 100% від запрограмованої швидкості шпинделя.
- `MAX_SPINDLE_0_OVERRIDE = 1.0` - Максимальне перевищення подачі, яке може вибрати користувач. 1.2 означає 120% від запрограмованої швидкості подачі. На багатошпиндельних верстатах будуть записи для кожного номера шпинделя. Використовується тільки в інтерфейсах користувача на базі QtVCP.
- `DEFAULT_SPINDLE_SPEED = 100` – швидкість обертання шпинделя за замовчуванням, коли шпиндель запускається в ручному режимі. Якщо цей параметр відсутній, значення за замовчуванням становить 1 об/хв для AXIS та 300 об/хв для GMOCCAPY.

  - *застаріло* - замість цього використовуйте розділ [SPINDLE_n]
- `DEFAULT_SPINDLE_0_SPEED = 100` - Стандартна швидкість обертання шпинделя при запуску шпинделя в ручному режимі. На багатошпиндельних верстатах будуть записи для кожного номера шпинделя. Використовується тільки в інтерфейсах користувача на базі QtVCP.

  - *застаріло* - замість цього використовуйте розділ [SPINDLE_n].
- `SPINDLE_INCREMENT = 200` – Приріст, що використовується під час натискання кнопок збільшення/зм Використовується лише інтерфейсами користувача на основі QtVCP.

- *застаріло* - замість цього використовуйте розділ [SPINDLE_n].

- `MIN_SPINDLE_0_SPEED = 1000` – Мінімальна швидкість обертання шпинделя, яку можна вибрати вручну. На багатошпиндельних верстатах будуть записи для кожного номера шпинделя. Використов лише інтерфейсами користувача на основі QtVCP.

  - *застаріло* - замість цього використовуйте розділ [SPINDLE_n].

- `MAX_SPINDLE_0_SPEED = 20000` – Максимальна швидкість обертання, яку можна вибрати вручну. На багатошпиндельних верстатах будуть записи для кожного номера шпинделя. Використовується лише інтерфейсами користувача на основі QtVCP.

  - *застаріло* - замість цього використовуйте розділ [SPINDLE_n].

- `PROGRAM_PREFIX = ~/linuxcnc/nc_files` - Стандартний каталог для файлів G-коду, підпрограм з іменами та M-кодів, визначених користувачем. Каталог `PROGRAM_PREFIX` шукається перед каталогами, переліченими в `[RS274]SUBROUTINE_PATH` та `[RS274]USER_M_PATH`.

- `INTRO_GRAPHIC = emc2.gif` – Зображення, що відображається на заставці.

- `INTRO_TIME = 5` – Максимальний час відображення заставки в секундах.

- `CYCLE_TIME = 100` - Час циклу графічного інтерфейсу користувача дисплея. Залежно від екрану, це може бути в секундах або мілісекундах (бажано в мілісекундах). Часто це швидкість оновлення, а не час очікування між оновленнями. Якщо час оновлення встановлено неправильно, екран може перестати реагувати або працювати дуже ривками. Значення 100 мс (0,1 с) є типовим налаштуванням, хоча можна використовувати діапазон від 50 до 200 мс (0,05–0,2 с). При недостатній потужності процесора можна досягти поліпшення, встановивши більше значення. Зазвичай стандартне значення є оптимальним.

- `PREVIEW_TIMEOUT = 5` – Час очікування (у секундах) для завантаження графічного попереднього перегляду G-коду. Наразі лише для AXIS.

- `HOMING_PROMPT = TRUE` - Будь-яке непорожнє значення (включно з ”0”) дозволить відображення повідомлення-підказки із запитом на переміщення до точки відправлення, коли в графічному інтерфейсі AXIS натиснуто кнопку ввімкнення. Натискання кнопки ”Ok” у повідомленні-підказці еквівалентне натисканню кнопки ”Home All” (або клавіші Ctrl-HOME).

- `FOAM_W = 1.5` встановлює висоту піни W.

- `FOAM_Z = 0` встановлює висоту піни по осі Z.

- `GRAPHICAL_MAX_FILE_SIZE = 20` — максимальний розмір (у мегабайтах), який буде відображатися графічно. Якщо програма перевищує це значення, буде відображатися обмежувальна рамка. За замовчуванням це значення дорівнює 20 МБ або 1/4 системної пам’яті, залежно від того, яке з них менше. Від’ємне значення інтерпретується як необмежене.

Note
Наступні елементи [DISPLAY] використовуються GladeVCP та PyVCP, див. розділ embedding a tab розділу GladeVCP або PyVCP Chapter для отримання додаткової інформації.

- `EMBED_TAB_NAME = Демонстрація GladeVCP`

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

  Note
  Різні програми інтерфейсу користувача використовують різні опції, і не кожна опція підтримується кожним інтерфейсом користувача. Детальні відомості про AXIS див. у документі AXIS GUI. Детальні відомості про GMOCCAPY див. у документі GMOCCAPY.

- DEFAULT_LINEAR_VELOCITY = .25 – швидкість за замовчуванням для лінійних переміщень, в одиницях machine за секунду.
- MIN_VELOCITY = 0.01 – Приблизне найнижче значення повзунка джогу.
- MAX_LINEAR_VELOCITY = 1.0 – Максимальна швидкість для лінійних переміщень, в одиницях виміру за секунду.
- MIN_LINEAR_VELOCITY = 0,01 – Приблизне найнижче значення повзунка джогу.
- DEFAULT_ANGULAR_VELOCITY = .25 – швидкість за замовчуванням для кутових переміщень, в одиницях вимірювання за секунду.
- MIN_ANGULAR_VELOCITY = 0.01 – Приблизне найнижче значення кутового повзунка.
- MAX_ANGULAR_VELOCITY = 1.0 – Максимальна швидкість для кутових переміщень, в машинних одиницях за секунду.
- INCREMENTS = 1 мм, 0,5 дюйма, ... - Визначає кроки, доступні для інкрементального переміщенн INCREMENTS можна використовувати для заміни значення за замовчуванням. Значення можуть бути десятковими числами (наприклад, 0,1000) або дробовими числами (наприклад, 1/16), за якими за бажанням може слідувати одиниця виміру (см, мм, мкм, дюйм, дюйм або міл). Якщо одиниця виміру не вказана, використовується одиниця виміру машини. Метричні та імперські відстані можуть бути змішаними: INCREMENTS = 1 дюйм, 1 міл, 1 см, 1 мм, 1 мкм є дійсним введенням.
- GRIDS = 10 mm, 1 in, ... – Визначає попередньо встановлені значення для ліній сітки. Значення інтерпретується так само, як INCREMENTS.
- OPEN_FILE = /full/path/to/file.ngc - Файл, який буде відображатися на попередньому перегляді при запуску AXIS. Використовуйте порожній рядок "" і жоден файл не буде завантажуватися при запуску. GMOCCAPY не використовуватиме це налаштування, оскільки воно пропонує відповідний запис на своїй сторінці налаштувань.
- EDITOR = gedit - Редактор, який використовується при виборі File > Edit для редагування G-коду з меню AXIS. Це необхідно налаштувати, щоб цей пункт меню працював. Іншим допустимим записом є gnome-terminal -e vim. Цей запис не застосовується до GMOCCAPY, оскільки GMOCCAPY має вбудований редактор.
- TOOL_EDITOR = tooledit - Редактор, який використовується для редагування таблиці інструментів (наприклад, шляхом вибору «Файл > Редагувати таблицю інструментів...» в AXIS). Інші допустимі записи: gedit, gnome-terminal -e vim та gvim. Цей запис не застосовується до GMOCCAPY, оскільки GMOCCAPY має вбудований редактор.
- PYVCP = /filename.xml – Файл опису панелі PyVCP. Див. розділ PyVCP для отримання додаткової інформації.
- PYVCP_POSITION = BOTTOM - Розташування панелі PyVCP в інтерфейсі користувача AXIS. Якщо ця змінна пропущена, панель за замовчуванням буде розташована праворуч. Єдиною допустимою альтернативою є BOTTOM. Дивіться розділ PyVCP Chapter для отримання додаткової інформації.
- LATHE = 1 - Будь-яке непорожнє значення (включаючи "0") призводить до використання осі "режиму токарного верстата" з видом зверху та з радіусом і діаметром на DRO.
- BACK_TOOL_LATHE = 1 – Будь-яке непорожнє значення (включаючи "0") призводить до використання осі "режиму токарного верстата із заднім інструментом" з інвертованою віссю X.
- FOAM = 1 - Будь-яке непорожнє значення (включаючи "0") призводить до зміни відображення осі для режиму різання піни.
- GEOMETRY = XYZABCUVW – Керує **попереднім переглядом** та **заднім відображенням** руху. Цей елемент складається з послідовності літер осей та керуючих символів, яким необов'язково передувати знак "-":

1. Літери X, Y, Z вказують на зміщення вздовж іменованої координати.
2. Літери A, B, C вказують обертання навколо відповідних осей X, Y, Z.
3. Літери U, V, W вказують на зміщення вздовж відповідних осей X, Y, Z.
4. Кожна зазначена літера повинна зустрічатися в `[TRAJ]COORDINATES`, щоб мати ефект.
5. Символ "**-**" перед будь-якою літерою інвертує напрямок операції.
6. Операції перенесення та обертання оцінюються **зправа наліво**. Тому використання `GEOMETRY=X` визначає обертання C, за яким слідує обертання B, а потім перенесення Z, Y, X. Порядок послідовних літер перенесення не має значення.
7. Правильний рядок GEOMETRY залежить від конфігурації машини та кінематики, що використову для її керування. Порядок літер є важливим. Наприклад, обертання навколо C, а потім B відрізняється від обертання навколо B, а потім C.
8. Повороти за замовчуванням застосовуються відносно початку координат верстата. Приклад: `GEOMETRY=CXYZ` спочатку переносить контрольну точку в X, Y, Z, а потім виконує поворот C навколо осі Z з центром у початку координат верстата.
9. Приклад перетворення UVW: `GEOMETRY=XYZUVW` призводить до переміщення UVW у системі координат інструмента, а XYZ – у системі координат матеріалу.
10. Машини для різання пінопласту (`FOAM = 1`) повинні вказувати «XY;UV» або залишати значення порожнім, навіть якщо це значення наразі ігнорується в режимі різання пінопласту. У майбутній версії може бути визначено значення «;», але якщо це станеться, «XY;UV» буде означати те саме, що і поточне значення за замовчуванням для пінопласту.
11. Експериментально: Якщо в рядку GEOMETRY міститься знак оклику (`!`), точки для обертань A, B, C відображаються відповідно до зміщень X, Y, Z, встановлених кодами G5x, G92. Приклад: Використання `GEOMETRY = !CXZ` для верстата з `[TRAJ]COORDINATES=XZC`. Це положення застосовується тільки для liveplots — попередній перегляд G-коду повинен виконуватися з нульовими зміщеннями G5x, G92. Це можна полегшити, встановлюючи зміщення в програмах тільки тоді, коли завдання виконується, як вказано в `#<_task> == 1`. Якщо при запуску існують ненульові зміщення через стійкість, зміщення повинні бути обнулені, а попередній перегляд перезавантажений.

Note
Якщо у файлі INI немає значення `[DISPLAY]GEOMETRY`, програма графічного інтерфейсу `[DISPLAY]DISPLAY` надає значення за замовчуванням (зазвичай "XYZABCUVW").

- `ARCDIVISION = 64` - Встановіть якість попереднього перегляду дуг. Дуги попередньо переглядаються шляхом їх поділу на кілька прямих ліній; півколо поділяється на **ARCDIVISION** частин. Більші значення забезпечують більш точний попередній перегляд, але завантажуються довше і призводять до більш повільного відображення. Менші значення забезпечують менш точний попередній перегляд, але завантаження займає менше часу і може призвести до швидшого відображення. Значення за замовчуванням 64 означає, що коло розміром до 3 дюймів буде відображатися з точністю до 1 міліметра (0,03%).
- `MDI_HISTORY_FILE =` - Ім'я локального файлу історії MDI. Якщо це не вказано, AXIS збереже історію MDI у файлі **.axis_mdi_history** у домашньому каталозі користувача. Це корисно, якщо на одному комп'ютері є кілька конфігурацій.
- `JOG_AXES =` - Порядок, в якому клавіші jog призначаються літерам осей. Стрілки вліво і вправо призначаються першій букві осі, вгору і вниз - другій, сторінка вгору/сторінка вниз - третій, а ліва і права дужки - четвертій. Якщо не вказано, значення за замовчуванням визначається з значень `[TRAJ]COORDINATES`, `[DISPLAY]LATHE` і `[DISPLAY]FOAM`.
- `JOG_INVERT =` – Для кожної літери осі напрямок подачі інвертується. Значення за замовчуванням – "X" для токарних верстатів, інакше – порожнє.

Note
Налаштування для JOG_AXES та JOG_INVERT застосовуються до ручного переміщення у світовому режимі за літерою координати осі та діють у світовому режимі після успішного повернення до вихідної позиції. Під час роботи у спільному режимі до повернення до вихідної позиції клавіші ручного переміщення на клавіатурі призначаються у фіксованій послідовності: ліворуч/праворуч: joint0, вгору/вниз: joint1, сторінка вгору/сторінка вниз: joint2, ліва/права дужка: joint3

- USER_COMMAND_FILE = mycommands.py – ім’я додаткового, конфігураційно-залежного файлу Python, що отримується з графічного інтерфейсу AXIS, замість користувацького файлу ~/.axisrc.

Note
Наступний елемент [DISPLAY] використовується лише інтерфейсом TKLinuxCNC.

- HELP_FILE = tklinucnc.txt – Шлях до файлу довідки.

4.4.2.3 [FILTER] Розділ

AXIS і GMOCCAPY мають можливість надсилати завантажені файли через програму-фільтр. Цей фільтр може виконувати будь-яке бажане завдання: щось просте, наприклад, переконатися, що файл закінчується на M2, або щось складне, наприклад, виявити, чи є вхідні дані глибинним зображенням, і згенерувати G-код для фрезерування форми, яку воно визначає. Розділ [FILTER] файлу INI контролює роботу фільтрів. Спочатку для кожного типу файлу напишіть рядок PROGRAM_EXTE
Потім вкажіть програму, яку потрібно виконати для кожного типу файлу. Ця програма отримує ім’я вхідного файлу як перший аргумент і повинна записувати код RS274NGC у стандартний вивід. Цей вивід буде відображатися в текстовій області, попередньо переглядатися в області відображення та виконуватися LinuxCNC під час запуску.

- PROGRAM_EXTENSION = .extension Опис

Якщо ваш постпроцесор виводить файли великими літерами, можливо, вам варто додати наступний рядок:

```
PROGRAM_EXTENSION = .NGC XYZ b''Пb''b''ob''b''cb''b''тb''b''пb''b''pb''b''ob''b''цb''b' ↩
    'eb''b''cb''b''ob''b''pb''
```

У наступних рядках додано підтримку конвертера зображень у G-код, що входить до комплекту LinuxCNC.

```
PROGRAM_EXTENSION = .png,.gif,.jpg # b''3b''b''ob''b''бb''b''pb''b''ab''b''жb''b''eb''b' ↩
    'нb''b''нb''b''яb'' b''гb''b''лb''b''иb''b''бb''b''иb''b''нb''b''иb'' b''уb'' b''гb''b' ↩
    'pb''b''ab''b''дb''b''ab''b''цb''b''ib''b''яb''b''xb'' b''cb''b''ib''b''pb''b''ob''b' ↩
    'гb''b''ob''
  png = image-to-gcode
  gif = image-to-gcode
  jpg = image-to-gcode
```

Приклад користувацького конвертера G-коду, розташованого в каталозі linuxcnc.

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

Note
Файл програми, пов’язаний з розширенням, повинен мати або повний шлях до програми, або знаходитися в каталозі, що знаходиться на системному шляху.

Також можна вказати інтерпретатора:

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

Таким чином, будь-який скрипт Python можна відкрити, а його вихідні дані обробляються як G-код. Один із таких прикладів скрипту доступний за адресою nc_files/holecircle.py. Цей скрипт створює G-код для свердління серії отворів по колу. Багато інших генераторів G-коду можна знайти на сайті LinuxCNC Wiki https://wiki.linuxcnc.org/.

Фільтри Python повинні використовувати функцію print для виведення результату до AXIS.

Цей приклад програми фільтрує файл і додає вісь W, щоб вона відповідала осі Z. Для роботи потрібна наявність пробілу між кожним словом осі.

```
#!/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
  Якщо встановлено змінну середовища AXIS_PROGRESS_BAR, то рядки, записані в stderr у вищезазначеному форматі, встановлюють індикатор прогресу AXIS на заданий відсоток. Ця функція повинна використовуватися будь-яким фільтром, що працює протягом тривалого часу.

4.4.2.4 [RS274NGC] Розділ

- PARAMETER_FILE = myfile.var – Файл, розташований у тому ж каталозі, що й INI-файл, який містить параметри, що використовуються інтерпретатором (зберігаються між запусками).

- `ORIENT_OFFSET = 0` – значення з плаваючою комою, що додається до параметра слова R операції M19 Orient Spindle. Використовується для визначення довільного нульового положення незалежно від орієнтації кріплення енкодера.
- `RS274NGC_STARTUP_CODE = G17 G20 G40 G49 G64 P0.001 G80 G90 G92.1 G94 G97 G98` - Рядок NC-кодів, з якими ініціалізується інтерпретатор. Це не замінює вказання модальних G-кодів у верхній частині кожного файлу NGC, оскільки модальні коди верстатів відрізняються і можуть бути змінені G-кодом, інтерпретованим раніше в сеансі.
- `SUBROUTINE_PATH = ncsubroutines:/tmp/testsubs:lathesubs:millsubs` - Вказує список до 10 каталогів, розділених двокрапкою (:), які будуть просканові, коли в G-коді вказані підпрограми з одним файлом. Ці каталоги шукаються після пошуку `[DISPLAY]PROGRAM_PREFIX` (якщо він вказаний) і перед пошуком `[WIZARD]WIZARD_ROOT` (якщо вказаний). Шляхи шукаються в порядку, в якому вони перелічені. Використовується перший підпрограмовий файл, який відповідає критеріям пошуку. Каталоги вказуються відносно поточного каталогу для файлу INI або як абсолютні шляхи. Список не повинен містити пробілів.
- `G64_DEFAULT_TOLERANCE = n` (За замовчуванням: 0) Значення P за замовчуванням для G64, якщо P не викликається.
- `G64_DEFAULT_NAIVETOLERANCE = n` (За замовчуванням: 0) Значення Q за замовчуванням для G64, якщо Q не викликається.
- `CENTER_ARC_RADIUS_TOLERANCE_INCH = n` (За замовчуванням: 0.00005)
- `CENTER_ARC_RADIUS_TOLERANCE_MM = n` (За замовчуванням: 0.00127)
- `USER_M_PATH = myfuncs:/tmp/mcodes:experimentalmcodes` - Вказує список каталогів, розділених двокрапкою (:), для функцій, визначених користувачем. Каталоги вказуються відносно поточного каталогу для файлу INI або як абсолютні шляхи. Список не повинен містити пробілів.

  Пошук виконується для кожної можливої функції, визначеної користувачем, зазвичай (M100-M199). Порядок пошуку такий:

  1. `[DISPLAY]PROGRAM_PREFIX` (якщо вказано)
  2. Якщо `[DISPLAY]PROGRAM_PREFIX` не вказано, пошук здійснюється за місцем розташування за замовчуванням: nc_files
  3. Потім пошукайте в кожному каталозі у списку `[RS274NGC]USER_M_PATH`.
     Перший знайдений у пошуку виконуваний файл M1*xx* використовується для кожного M1*xx*.

  Note
  Максимальна кількість каталогів `USER_M_PATH` визначається під час компіляції (тип: `USER_DEFINED_FUNCTION_MAX_DIRS == 5`).

- `INI_VARS = 1` (За замовчуванням: 1)
  Дозволяє програмам G-коду зчитувати значення з INI-файлу, використовуючи формат #<_ini[розділ
  Див. Параметри G-коду.
- `HAL_PIN_VARS = 1` (За замовчуванням: 1)
  Дозволяє програмам G-коду зчитувати значення виводів HAL у форматі #<_hal[елемент HAL]>. Доступ до змінних є тільки для читання. Дивіться G-code Parameters для більш детальної інформації та важливих застережень.
- `RETAIN_G43 = 0` (За замовчуванням: 0)
  Якщо встановлено, ви можете ввімкнути G43 після завантаження першого інструменту, а потім не турбуватися про це в програмі. Коли ви нарешті вивантажите останній інструмент, режим G43 скасовується.

- `OWORD_NARGS = 0` (За замовчуванням: 0)
  Якщо цю функцію ввімкнено, то викликана підпрограма може визначити кількість фактично переданих позиційних параметрів, перевіряючи параметр `#<n_args>`.
- `NO_DOWNCASE_OWORD = 0` (За замовчуванням: 0)
  Зберігати регістр слів на букву «O» в коментарях, якщо встановлено, дозволяє читати елементи HAL зі змішаним регістром у структурованих коментарях, таких як `(debug, #<_hal[MixedCaseItem]`
- `OWORD_WARNONLY = 0` (За замовчуванням: 0)
  Попереджати, а не видавати помилку, у разі помилок у підпрограмах типу O-word.
- `DISABLE_G92_PERSISTENCE = 0` (За замовчуванням: 0) Дозволити автоматичне очищення зміщення G92 під час запуску конфігурації.
- `DISABLE_FANUC_STYLE_SUB = 0` (За замовчуванням: 0) Якщо є причина вимкнути підпрограми Fanuc, встановіть значення 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
Вищезазначені шість опцій контролювалися бітовою маскою FEATURES у версіях LinuxCNC до версії 2.8. Цей тег INI більше не працюватиме.
Для довідки:

```
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` – це дійсний шлях пошуку, але майстер не повністю реалізовано, і результати його використання непередбачувані.

- `LOG_LEVEL = 0` Specify the log_level (за замовчуванням: 0)
- `LOG_FILE = file-name.log`
  Щоб вказати файл, який використовується для запису даних.
- `REMAP=M400 modalgroup=10 argspec=Pq ngc=myprocedure` Див. Remap Extending G-code розділ для отримання детальної інформації.
- `ON_ABORT_COMMAND=O <on_abort> call` Див. розділ Remap Extending G-code для отримання детальної інформації.

4.4.2.5 [EMCMOT] Розділ

Цей розділ є користувацьким і не використовується безпосередньо LinuxCNC. Більшість конфігурацій використовують значення з цього розділу для завантаження контролера руху. Для отримання додаткової інформації про контролер руху дивіться розділ Motion.

- `EMCMOT = motmod` - тут зазвичай використовується назва контролера руху.
- `BASE_PERIOD = 50000` - період виконання завдання «Базовий» у наносекундах.
- `SERVO_PERIOD = 1000000` - Це період виконання завдання ”Серво” в наносекундах.
- `TRAJ_PERIOD = 100000` - Це період виконання завдання «Планувальник траєкторії» в наносекундах.
- `COMM_TIMEOUT = 1.0` - Кількість секунд очікування на підтвердження отримання повідомлень від Motion (частини контролера руху, що працює в режимі реального часу) від Task (частини контролера руху, що працює не в режимі реального часу).
- `HOMEMOD` = *alternate_homing_module* [home_parms=value] Змінна HOMEMOD є необов’язковою. Якщо вона вказана, використовується вказаний (створений користувачем) модуль замість модуля за замовчуванням (homemod). Параметри модуля (home_parms) можуть бути включені, якщо вони підтримуються зазначеним модулем. Налаштування можна замінити з командного рядка за допомогою опції -m ($ linuxcnc -h).

4.4.2.6 [TASK] Розділ

- `TASK = milltask` – Вказує назву виконуваного файлу *task*. Виконуваний файл *task* виконує різні дії, такі як
  - спілкуватися з інтерфейсами користувача через NML,
  - зв’язуватися з планувальником руху в реальному часі через спільну пам’ять, відмінну від HAL, та
  - інтерпретувати G-код. Наразі існує лише один виконуваний файл завдання, який має сенс для 99,9% користувачів, – 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] розділ

- `HALFILE = example.hal` - Виконати файл *example.hal* під час запуску.

  Якщо `HALFILE` вказано кілька разів, файли інтерпретуються в тому порядку, в якому вони з’являються в файлі INI. Файли HAL є описовими, виконання того, що описано в файлах HAL, запускається потоками, в які вбудовані функції, а не читанням файлу HAL. Майже всі конфігурації матимуть принаймні один `HALFILE`, а крокові системи зазвичай мають два таких файли, тобто один, який визначає загальну конфігурацію крокового двигуна (*core_stepper.hal*), і один, який визначає роз’єм машини (*xxx_pinout.hal*).

  Файли HAL, вказані у змінній `HALFILES`, знаходяться за допомогою пошуку. Якщо файл із вказаною назвою знайдено у каталозі, що містить файл INI, він використовується. Якщо файл із вказаною назвою не знайдено у цьому каталозі файлів INI, пошук здійснюється за допомогою системної бібліотеки файлів HAL.

  Якщо LinuxCNC запускається за допомогою скрипта `linuxcnc` з опцією ”`-H` *dirname*”, вказане ім’я каталогу додається до початку пошуку, описаного вище, так що *dirname* шукається першим. Опція ”`-H` *dirname*” може бути вказана більше одного разу, каталоги додаються в порядку.

  HALFILE також може бути вказаний як абсолютний шлях (коли ім’я починається з символу /). Абсолютні шляхи не рекомендуються, оскільки їх використання може обмежити переміщення конфігурацій.

- `HALFILE = texample.tcl` [*arg1* [*arg2*] ...] - Виконати файл tcl *texample.tcl* під час запуску з аргументами arg1, arg2 тощо як список argv. Файли з розширенням .tcl обробляються, як зазначено вище, але для обробки використовується haltcl. Докладнішу інформацію див. у розділі HALTCL Chapter.
- `HALFILE = LIB:sys_example.hal` - Виконати файл системної бібліотеки *sys_example.hal* під час запуску. Явне використання префікса LIB: призводить до використання системної бібліотеки HALFILE без пошуку в каталозі файлів INI.
- `HALFILE = LIB:sys_texample.tcl` [*arg1* [*arg2* ...]] - Виконати файл системної бібліотеки *sys_texample* під час запуску. Явне використання префікса LIB: призводить до використання системної бібліотеки HALFILE без пошуку в каталозі файлів INI.

Елементи HALFILE визначають файли, які завантажують компоненти HAL та встановлюють сигнальні з'єднання між контактами компонентів. Поширені помилки

1. пропуск оператора addf, необхідного для додавання функції(й) компонента до потоку,
2. неповні специфікатори сигналу (мережі).

Пропуск необхідних операторів addf майже завжди є помилкою. Сигнали зазвичай містять одне або декілька вхідних з'єднань і одне вихідне з'єднання (але обидва вони не є обов'язковими). Для перевірки цих умов і відображення результатів у stdout та спливаючому графічному інтерфейсі користувача надається файл системної бібліотеки:

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

Note
Рядок LIB:halcheck.tcl повинен бути останнім [HAL]HALFILE. Вкажіть опцію «nopopup», щоб придушити спливаюче повідомлення і дозволити негайний запуск. З'єднання, встановлені за допомогою POSTGUI_HALFILE, не перевіряються.

- `TWOPASS = ON` - Використовувати двопрохідну обробку для завантаження компонентів HAL. При двопрохідній обробці рядки файлів, вказані в `[HAL]HALFILE`, обробляються у два проходи. У першому проході (pass0) зчитуються всі HALFILES і накопичуються багаторазові появи команд loadrt і loadusr. Ці накопичені команди завантаження виконуються в кінці pass0. Це накопичення дозволяє вказувати рядки завантаження більше одного разу для даного компонента (за умови, що імена names= використовуються унікально при кожному використанні). У другому проході (pass1) HALFILES перечитуються і виконуються всі команди, крім раніше виконаних команд завантаження.
- `TWOPASS = nodelete verbose` - Функцію `TWOPASS` можна активувати за допомогою будь-якого рядка, що не дорівнює нулю, включаючи ключові слова verbose та nodelete. Ключове слово verbose спричиняє виведення детальної інформації на stdout. Ключове слово nodelete зберігає тимчасові файли в /tmp.

Для отримання додаткової інформації див. розділ HAL TWOPASS.

- `HALCMD` = *command* - Виконати *command* як окрему команду HAL. Якщо `HALCMD` вказано кілька разів, команди виконуються в тому порядку, в якому вони вказані в файлі INI. Рядки `HALCMD` виконуються після всіх рядків `HALFILE`.
- `SHUTDOWN = shutdown.hal` - Виконати файл *shutdown.hal* при виході з LinuxCNC. Залежно від використовуваних драйверів обладнання, це може дозволити встановити вихідні значення на визначені значення при нормальному вимкненні LinuxCNC. Однак, оскільки немає гарантії, що цей файл буде виконаний (наприклад, у разі збою комп'ютера), він не замінює належну фізичну ланцюг аварійної зупинки або інші засоби захисту від збою програмного забезпечення.

- `POSTGUI_HALFILE = example2.hal` - Виконати *example2.hal* після того, як GUI створив свої HAL-контакти. Деякі GUI створюють HAL-контакти і підтримують використання postgui halfile для їх використання. GUI, які підтримують postgui HAL-файли, включають Touchy, AXIS, Gscreen і GMOCCAPY.
  Докладнішу інформацію див. у розділі PyVCP з AXIS.
- `HALUI = halui` – додає контакти інтерфейсу користувача HAL.
  Для отримання додаткової інформації див. розділ Інтерфейс користувача HAL.

4.4.2.8 [HALUI] розділ

- `MDI_COMMAND = G53 G0 X0 Y0 Z0` - Команду MDI можна виконати за допомогою `halui.mdi-command-0`
  Збільшуйте номер для кожної команди, переліченої в розділі [HALUI]. Також можна запускати підпрограми. `MDI_COMMAND = o<yoursub> CALL [#<yourvariable>]`

4.4.2.9 [APPLICATIONS] Розділ

LinuxCNC може запускати інші програми до запуску вказаного графічного інтерфейсу користувача. Програми можуть запускатися після вказаної затримки, щоб забезпечити виконання дій, що залежать від графічного інтерфейсу користувача (наприклад, створення контактів HAL, специфічних для графічного інтерфейсу користувача).

- `DELAY =` *значення* - кількість секунд, протягом яких слід почекати перед запуском інших програм. Затримка може бути необхідною, якщо програма залежить від дій `[HAL]POSTGUI_HALFILE` або від створених графічним інтерфейсом користувача контактів HAL (за замовчуванням `DELAY=0`).
- 'APP =` *appname* [arg1 [arg2 ...]]' - Програма, яку потрібно запустити. Ця специфікація може бути включена кілька разів. Назва програми може бути вказана явно як абсолютне або тильда-іменуване ім'я файлу (перший символ - / або ~), відносне ім'я файлу (перші символи імені файлу - ./) або як файл у каталозі INI-файлів. Якщо за цими іменами не знайдено виконуваного файлу, для пошуку програми використовується шлях пошуку користувача PATH.
  Приклади:
  - Імітуйте вхідні дані для виводів HAL для тестування (використовуючи sim_pin — простий графічний інтерфейс для встановлення вхідних даних для параметрів, непідключених виводів або сигналів без записувачів):
    ```
    APP = sim_pin motion.probe-input halui.abort motion.analog-in-00
    ```
  - Викличте halshow з попередньо збереженим списком спостереження. Оскільки LinuxCNC встановлює робочий каталог на каталог для INI-файлу, ви можете звертатися до файлів у цьому каталозі (наприклад: my.halshow):
    ```
    APP = halshow my.halshow
    ```
  - Або ж можна вказати файл списку спостереження, ідентифікований повним шляхом:
    ```
    APP = halshow ~/saved_shows/spindle.halshow
    ```
  - Відкрийте halscope, використовуючи попередньо збережену конфігурацію:
    ```
    APP = halscope -i my.halscope
    ```

4.4.2.10 [TRAJ] Розділ



Warning
Новий Планувальник траєкторій (TP) увімкнено за замовчуванням. Якщо у вашому розділі [TRAJ] немає налаштувань TP, LinuxCNC використовує такі значення за замовчуванням:
ARC_BLEND_ENABLE = 1
ARC_BLEND_FALLBACK_ENABLE = 0
ARC_BLEND_OPTIMIZATION_DEPTH = 50
ARC_BLEND_GAP_CYCLES = 4
ARC_BLEND_RAMP_FREQ = 100

Розділ [TRAJ] містить загальні параметри для модуля планування траєкторії в режимі «рух».

- `ARC_BLEND_ENABLE = 1` - Увімкнути новий TP. Якщо встановлено на 0, TP використовує параболічне змішування (випередження на 1 сегмент) (За замовчуванням: 1).
- `ARC_BLEND_FALLBACK_ENABLE = 0` - Опціонально повертається до параболічного змішування, якщо розрахункова швидкість є вищою. Однак ця оцінка є приблизною, і, здається, що просто вимкнення цієї функції забезпечує кращу продуктивність (за замовчуванням: 0).
- `ARC_BLEND_OPTIMIZATION_DEPTH = 50` - Глибина прогнозування в кількості сегментів.

  Щоб трохи розширити це, ви можете вибрати це значення дещо довільно. Ось формула для оцінки того, яка «глибина» вам потрібна для певної конфігурації:

```
# 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)
```

  Отже, машині з максимальною швидкістю осі 10 IPS, максимальним прискоренням 100 IPS^2 та періодом сервообміну 0,001 с знадобиться:

  10 / (2,0 * 100 * 0,001) = 50 сегментів, щоб завжди досягати максимальної швидкості вздовж найшвидшої осі.

  На практиці це число не так важливо налаштовувати, оскільки попередній перегляд рідко потребує повної глибини, якщо тільки у вас немає великої кількості дуже коротких сегментів. Якщо під час тестування ви помітили дивні уповільнення і не можете зрозуміти, звідки вони беруться, спочатку спробуйте збільшити цю глибину, використовуючи наведену вище формулу.

  Якщо ви все ще бачите дивні уповільнення, це може бути пов’язано з тим, що у вашій програмі є короткі сегменти. Якщо це так, спробуйте додати невеликий допуск для виявлення наївної CAM. Гарне емпіричне правило таке:

```
# min_length ~= v_req * t_c
# where:
# v_req = desired velocity in UU / sec
# t_c = servo period (seconds)
```

  Якщо ви хочете рухатися по траєкторії зі швидкістю 1 IPS = 60 IPM, а період сервоприводу становить 0,001 с, то будь-які сегменти, коротші за min_length, сповільнюватимуть рух. Якщо ви встановите допуск Naive CAM приблизно на рівні цієї мінімальної довжини, надто короткі сегменти будуть об’єднані, щоб усунути це вузьке місце. Звичайно, занадто високе значення толерантності означає великі відхилення траєкторії, тому вам доведеться трохи поекспериментувати, щоб знайти оптимальне значення. Я б почав з 1/2 від min_length, а потім збільшував би значення

за необхідності. * ARC_BLEND_GAP_CYCLES = 4 Наскільки коротким має бути попередній сегмент, перш ніж планувальник траєкторії його «споживатиме».

Часто при злитті кругових дуг між злиттями залишаються короткі відрізки ліній. Оскільки геометрія повинна бути круговою, ми не можемо злити всю лінію, якщо наступна лінія трохи коротша. Оскільки планувальник траєкторії повинен торкатися кожного сегмента принаймні один раз, це означає, що дуже маленькі сегменти значно уповільнюють процес. Моє рішення цієї проблеми полягає в тому, щоб «спожити» короткий сегмент, зробивши його частиною дуги злиття. Оскільки лінія + злиття є одним сегментом, нам не потрібно уповільнюватися, щоб торкнутися дуже короткого сегмента. Ймовірно, вам не доведеться змінювати це налаштування. * ARC_BLEND_RAMP_FREQ = 20 - Це «гранична» частота для використання зростаючої швидкості.

«Поступове прискорення» в цьому випадку означає постійне прискорення на всьому відрізку. Це менш оптимальний варіант, ніж трапецієподібний профіль швидкості, оскільки прискорення не є максимальним. Однак, якщо відрізок є достатньо коротким, то не вистачає часу для значного прискорення перед тим, як ми досягнемо наступного відрізка. Згадайте короткі відрізки лінії з попереднього прикладу. Оскільки це лінії, прискорення на поворотах відсутнє, тому ми можемо вільно прискорюватися до необхідної швидкості. Однак, якщо ця лінія знаходиться між двома дугами, то вона повинна буде знову швидко сповільнитися, щоб не перевищити максимальну швидкість наступного сегмента. Це означає, що ми маємо стрибок прискорення, а потім стрибок уповільнення, що спричиняє великий ривок, при дуже невеликому прирості продуктивності. Це налаштування є способом усунення цього ривка для коротких сегментів.

В основному, якщо сегмент буде завершений за час, менший за 1 / ARC_BLEND_RAMP_FREQ, ми не турбуємося про трапецієподібний профіль швидкості на цьому сегменті і використовуємо постійне прискорення. (Встановлення ARC_BLEND_RAMP_FREQ = 1000 еквівалентно постійному використанню трапецієподібного прискорення, якщо сервоконтур становить 1 кГц).

Ви можете охарактеризувати найгірший випадок втрати продуктивності, порівнявши швидкість, якої досягає трапецієподібний профіль, зі швидкістю пандуса:

```
# 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 вищезгаданої машини пульсація для частоти відсічення 20 Гц становить 100 / (4 * 20) = 1,25 IPS. Це здається високим показником, але слід пам’ятати, що це лише найгірший варіант оцінки. Насправді трапецієподібний профіль руху обмежується іншими факторами, такими як нормальне прискорення або необхідна швидкість, тому фактична втрата продуктивності повинна бути набагато меншою. Збільшення частоти відсічення може підвищити продуктивність, але зробити рух більш різким через нерівномірність прискорення. Для початку слід вибрати значення в діапазоні від 20 Гц до 200 Гц.

Зрештою, жодні налаштування не пришвидшать траєкторію інструменту з великою кількістю маленьких, вузьких кутів, оскільки ви обмежені прискоренням на поворотах.

- SPINDLES = 3 – Кількість шпинделів для підтримки. Вкрай важливо, щоб це число збігалося з параметром ”num_spindles”, переданим модулю руху.
- COORDINATES = X Y Z - Назви осей, що контролюються. Допустимі тільки X, Y, Z, A, B, C, U, V, W. У G-коді приймаються тільки осі, названі в COORDINATES. Допускається писати назву осі більше одного разу (наприклад, X Y Y Z для портальної машини). Для загальної «кінематики Трівкіна» номери з’єднань присвоюються послідовно відповідно до параметра Трівкіна «coordinates=». Отже, для Трівкіна «coordinates=xz» joint0 відповідає X, а joint1 відповідає Z. Інформацію про Трівкіна та інші кінематичні модулі див. на сторінці довідки про кінематику («$ man kins»).
- LINEAR_UNITS = <одиниці>_ - Визначає «машинні одиниці» для лінійних осей. Можливі варіанти: мм або дюйми. Це не впливає на лінійні одиниці в коді CNC (це роблять слова G20 та G21).

- ANGULAR_UNITS = *<одиниці виміру>* - Вказує «машинні одиниці виміру» для осей обертання. Можливі варіанти: «deg», «degree» (360 на коло), «rad», «radian» (2*п на коло), «grad» або «gon» (400 на коло). Це не впливає на кутові одиниці виміру коду CNC. У RS274NGC слова A-, B- і C- завжди виражаються в градусах.
- DEFAULT_LINEAR_VELOCITY = 0.0167 – Початкова швидкість для поштовхів лінійних осей у машинних одиницях за секунду. Значення, що відображається в *AXIS*, дорівнює машинним одиницям за хвилину.
- DEFAULT_LINEAR_ACCELERATION = 2.0 – У машинах з нетривіальною кінематикою, прискорення, що використовується для телеоптичних (декартових) штовхань, у «машинних одиницях» за секунду за секунду.
- MAX_LINEAR_VELOCITY = 5.0 – Максимальна швидкість для будь-якої осі або скоординованого руху в «машинних одиницях» за секунду. Відображене значення дорівнює 300 одиницям за хвилину.
- MAX_LINEAR_ACCELERATION = 20.0 – Максимальне прискорення для будь-якого руху осі або координованої осі в «машинних одиницях» за секунду за секунду.
- PLANNER_TYPE = 0 - Вибирає тип планувальника траєкторії: 0 = трапецієподібна (за замовчуванням), 1 = S-подібна крива з обмеженням ривка. Планування за S-подібною кривою активне лише тоді, коли PLANNER_TYPE = 1 ТА 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* - Якщо встановлено непусте значення, положення суглобів зберігаються між запусками в цьому файлі. Це дозволяє машині запускатися з тими самими координатами, що були на момент вимкнення. Це передбачає, що під час вимкнення живлення машина не рухалася. Якщо не встановлено, положення суглобів не зберігаються і будуть починатися з 0 кожного разу при запуску LinuxCNC. Це може бути корисно для невеликих машин без перемикачів початкового положення. При використанні інтерфейсу Mesa resolver цей файл може бути використаний для емуляції абсолютних енкодерів і усунення необхідності повернення в початкове положення (без втрати точності). Більш детальну інформацію дивіться на сторінці довідки hostmot2.
- NO_FORCE_HOMING = 1 - За замовчуванням LinuxCNC змушує користувача повернути верстат у вихідне положення перед виконанням будь-якої команди MDI або програми. Зазвичай перед поверненням у вихідне положення дозволено тільки ручне переміщення. Для конфігурацій, що використовують ідентичну кінематику, установка NO_FORCE_HOMING = 1 дозволяє користувачеві виконувати переміщення MDI і запускати програми без попереднього повернення машини у вихідне положення. Інтерфейси, що використовують ідентичну кінематику без можливості повернення у вихідне положення, повинні мати цю опцію встановленою на 1.



Warning
LinuxCNC не знатиме меж переміщення вашого суглоба при використанні NO_FORCE_HOMING = 1.

- HOME = 0 0 0 0 0 0 0 0 0 - Початкове положення в світовій системі координат, необхідне для кінематичних модулів, які обчислюють світові координати за допомогою kinematicsForward() при переході з режиму з'єднання в режим телеоперації. Можна вказати до дев'яти значень

координат (X Y Z A B C U V W), невикористані кінцеві елементи можна опустити. Це значення використовується тільки для машин з нетривіальною кінематикою. На машинах з тривіальною кінематикою (фрезерні, токарні, портальні типи) це значення ігнорується. Примітка: Конфігурація шестиногих роботів sim вимагає значення, відмінного від нуля, для координати Z.

- TPMOD = *alternate_trajectory_planning module* [tp_parms=value]
  Змінна TPMOD є необов'язковою. Якщо вона вказана, використовується вказаний (створений користувачем) модуль замість модуля за замовчуванням (tpmod). Параметри модуля (tp_parms) можуть бути включені, якщо вони підтримуються вказаним модулем. Це налаштування можна замінити з командного рядка за допомогою опції -t ($ linuxcnc -h).
- NO_PROBE_JOG_ERROR = 0 - Дозволити обхід перевірки спрацьовування зонда під час ручного переміщення.
- NO_PROBE_HOME_ERROR = 0 - Дозволити обійти перевірку спрацьовування зонда під час виконання переведення в початкове положення.

4.4.2.11 [KINS] Розділ

- JOINTS = 3 - Вказує кількість шарнірів (двигунів) у системі. Наприклад, машина trivkins XYZ з одним двигуном для кожної осі має 3 шарніри. Портальна машина з одним двигуном на кожній з двох осей і двома двигунами на третій осі має 4 шарніри. (Ця конфігураційна змінна може використовуватися графічним інтерфейсом користувача для встановлення кількості шарнірів (num_joints), вказаної в модулі руху (motmod).)
- KINEMATICS = trivkins - Вкажіть кінематичний модуль для модуля руху. Графічні інтерфейси можуть використовувати цю змінну для вказання рядка loadrt у файлах HAL для модуля motmod. Для отримання додаткової інформації про кінематичні модулі див. сторінку довідки: $ man kins.

4.4.2.12 [AXIS_<letter>] Розділ

<літера> визначає одне з: X Y Z A B C U V W

- TYPE = LINEAR - Тип цієї осі, або LINEAR, або ANGULAR. Необхідно, якщо ця вісь не є типом осі за замовчуванням. Типи осей за замовчуванням: X, Y, Z, U, V, W = LINEAR та A, B, C = ANGULAR. Це налаштування діє в графічному інтерфейсі AXIS, але зверніть увагу, що інші графічні інтерфейси можуть по-різному обробляти дані.
- MAX_VELOCITY = 1.2 – Максимальна швидкість для цієї осі в machine units за секунду.
- MAX_ACCELERATION = 20.0 - Максимальне прискорення для цієї осі в машинних одиницях за секунду в квадраті.
- MAX_JERK = 0.0 - Максимальний ривок для цієї осі в одиницях виміру за секунду, підсумованих у кубі. Використовується, коли ввімкнено планування траєкторії S-подібної кривої. Якщо встановлено значення 0 (за замовчуванням), обмеження ривка для кожної осі не застосовується.
- MIN_LIMIT = -1000 - Мінімальне обмеження (м'яке обмеження) для руху осі, в одиницях машини. При перевищенні цього обмеження контролер припиняє рух осі. Ось повинна бути повернена в початкове положення, перш ніж MIN_LIMIT набуде чинності. Для обертової осі (типу A, B, C) з необмеженим обертанням, яка не має MIN_LIMIT для цієї осі в розділі [AXIS_<літера>], використовується значення -1e99.
- MAX_LIMIT = 1000 - Максимальний ліміт (м'який ліміт) для руху осі, в одиницях машини. Коли цей ліміт перевищується, контролер припиняє рух осі. Ось повинна бути повернена в початкове положення, перш ніж MAX_LIMIT вступить в силу. Для обертової осі (типу A, B, C) з необмеженим обертанням, яка не має MAX_LIMIT для цієї осі в розділі [AXIS_<літера>], використовується значення 1e99.

- `WRAPPED_ROTARY = 1` - Якщо для осі ANGULAR встановлено значення 1, вісь буде рухатися в діапазоні 0-359,999 градусів. Додатні числа перемістять вісь у додатному напрямку, а від'ємні числа - у від'ємному напрямку.
- `LOCKING_INDEXER_JOINT = 4` - Це значення вибирає шарнір, який буде використовуватися для індексатора блокування для вказаної осі *<літера>*. У цьому прикладі шарнір має номер 4, що відповідає осі B для системи XYZAB з кінематикою Трівкінса (тотожною). Після встановлення G0-рух для цієї осі ініціює розблокування за допомогою `joint.4.unlock pin`, потім чекає на `joint.4.is-unlocked` pin, а потім переміщує шарнір із швидкістю, встановленою для цього шарніра. Після переміщення `joint.4.unlock` буде false, і рух чекатиме, поки `joint.4.is-unlocked` не стане false. Переміщення з іншими шарнірами не дозволяється під час переміщення заблокованого поворотного шарніра. Щоб створити штифти розблокування, використовуйте параметр motmod:

```
unlock_joints_mask=jointmask
```

  Біти маски спільного доступу є: (LSB)0:joint0, 1:joint1, 2:joint2, ...

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

- `OFFSET_AV_RATIO = 0.1` - Якщо значення не дорівнює нулю, цей елемент дозволяє використовувати вхідні контакти HAL для зміщення зовнішніх осей:

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

Див. розділ: Зовнішні зміщення осей для отримання інформації про використання.

4.4.2.13 [JOINT_<num>] Розділи

<num> визначає номер суглоба 0 ... (num_joints-1). Значення *num_joints* встановлюється за допомогою `[KINS]JOINTS=`.

Розділи `[JOINT_0]`, `[JOINT_1]` тощо містять загальні параметри для окремих компонентів у модулі управління з'єднаннями. Назви розділів з'єднань починаються з нумерації 0 і проходять через кількість з'єднань, зазначену в записі `[KINS]JOINTS`, мінус 1.

Зазвичай (для систем, що використовують «кінематику Трівкінса», існує відповідність 1:1 між суглобом та літерою координати осі):

- 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

Інші кінематичні модулі з тотожною кінематикою доступні для підтримки конфігурацій з частковими наборами осей. Наприклад, використовуючи тривкіни з `coordinates=XZ`, зв'язки між осями та суглобами:

- JOINT_0 = X
- JOINT_1 = Z

Для отримання додаткової інформації про модулі кінематики див. сторінку довідки *kins* (у терміналі UNIX введіть `man kins`).

- `TYPE = LINEAR` - Тип з'єднання, або `LINEAR`, або `ANGULAR`.
- `UNITS = INCH` - Якщо вказано, цей параметр замінює пов'язаний параметр `[TRAJ] UNITS`, наприклад, `[TRAJ]LINEAR_UNITS`, якщо `TYPE` цього з'єднання — `LINEAR`, `[TRAJ]ANGULAR_UNITS`, якщо `TYPE` цього з'єднання — `ANGULAR`.
- `MAX_VELOCITY = 1.2` - Максимальна швидкість для цього з'єднання в machine units за секунду.
- `MAX_ACCELERATION = 20.0` - Максимальне прискорення для цього з'єднання в машинних одиницях за секунду в квадраті.
- `MAX_JERK = 0.0` - Максимальний ривок для цього з'єднання в одиницях виміру за секунду, підсумованих у кубі. Використовується, коли ввімкнено планування траєкторії S-подібної кривої. Якщо встановлено значення 0 (за замовчуванням), обмеження ривка для кожного з'єднання не застосовується.
- `BACKLASH = 0.0000` - Люфт в одиницях машини. Значення компенсації люфту може бути використано для компенсації невеликих недоліків в апаратному забезпеченні, що використовується для приводу суглоба. Якщо до суглоба додається люфт і ви використовуєте крокові двигуни, то `STEPGEN_MAXACCEL` повинен бути збільшений в 1,5-2 рази від `MAX_ACCELERATION` для суглоба. Надмірна компенсація люфту може спричинити ривки суглоба під час зміни напрямку. Якщо для суглоба вказано COMP_FILE, BACKLASH не використовується.
- `COMP_FILE =` *file.extension* - Файл компенсації складається з карти інформації про положення суглоба. Значення файлу компенсації вказані в одиницях виміру машини. Кожен набір значень знаходиться в одному рядку, розділеному пробілом. Перше значення є номінальним значенням (задане положення). Друге і третє значення залежать від налаштування `COMP_FILE_TYPE`. Точки між номінальними значеннями інтерполюються між двома номінальними значеннями. Файли компенсації повинні починатися з найменшого номінального значення і бути в порядку зростання до найбільшого номінального значення. Імена файлів чутливі до регістру і можуть містити літери та/або цифри. Наразі обмеження в LinuxCNC становить 256 тріплетів на шарнір.

  Якщо для з'єднання вказано `COMP_FILE`, `BACKLASH` не використовується.
- `COMP_FILE_TYPE = 0` або `1` – Визначає тип файлу компенсації. Перше значення – це номінальна (задана) позиція для обох типів.
  Для кожного `COMP_FILE` необхідно вказати `COMP_FILE_TYPE`.
  - «Тип 0»: Друге значення визначає фактичне положення, коли шарнір рухається в позитивному напрямку (збільшення значення). Третє значення визначає фактичне положення, коли шарнір рухається в негативному напрямку (зменшення значення).

    Приклад типу 0

    ```
    -1.000 -1.005 -0.995
    0.000 0.002 -0.003
    1.000 1.003 0.998
    ```
  - «Тип 1»: Друге значення визначає позитивне відхилення від номінального значення під час руху в позитивному напрямку. Третє значення визначає негативне відхилення від номінального значення під час руху в негативному напрямку.

    Приклад типу 1

    ```
    -1.000 0.005 -0.005
    0.000 0.002 -0.003
    1.000 0.003 -0.004
    ```

- MIN_LIMIT = -1000 - мінімальне обмеження для руху з'єднання, в одиницях машини. Коли це обмеження досягається, контролер припиняє рух з'єднання. Для поворотного з'єднання з необмеженим обертанням, яке не має MIN_LIMIT для цього з'єднання в розділі [JOINT_N], використовується значення -1e99.
- MAX_LIMIT = 1000 - Максимальний ліміт для руху з'єднання, в одиницях машини. Коли цей ліміт досягнуто, контролер припиняє рух з'єднання. Для поворотного з'єднання з необмеженим обертанням, яке не має MAX_LIMIT для цього з'єднання в розділі [JOINT_N], використовується значення 1e99.

Note
Для кінематики **ідентичності** налаштування [JOINT_N]MIN_LIMIT/MAX_LIMIT повинні дорівнювати або перевищувати відповідні (один до одного ідентичності) обмеження [AXIS_L]. Ці налаштування перевіряються під час запуску, коли вказано модулі кінематики trivkins.

Note
Налаштування [JOINT_N]MIN_LIMIT/MAX_LIMIT застосовуються під час ручного переміщення в режимі з'єднання перед поверненням у вихідне положення. Після повернення у вихідне положення координатні обмеження `[AXIS_L]MIN_LIMIT`/MAX_LIMIT використовуються як обмеження для переміщення осі (літери координати) та для планування траєкторії, що використовується для переміщень G-коду (програм та команд MDI). Планувальник траєкторії працює в декартовому просторі (XYZABCUVW) і не має інформації про рух суглобів, реалізований **будь-яким** кінематичним модулем. Можливе порушення меж суглобів для G-коду, який дотримується обмежень позиції планування траєкторії, коли використовується неідентична кінематика. Модуль руху завжди виявляє порушення меж позиції суглобів і несправності, якщо вони відбуваються під час виконання команд G-коду. Див. також пов'язану проблему GitHub issue #97.

- MIN_FERROR = 0,010 - це значення в одиницях машини, на яке шарнір може відхилятися від заданого положення при дуже низьких швидкостях. Якщо MIN_FERROR менше FERROR, то ці два значення утворюють лінію точок спрацьовування помилки. Це можна уявити як графік, де один вимір - це швидкість, а інший - дозволена похибка слідування. Зі збільшенням швидкості величина похибки слідування також збільшується до значення FERROR.
- FERROR = 1.0 - FERROR - це максимально допустима похибка слідування в одиницях машини. Якщо різниця між заданим і виміряним положенням перевищує це значення, контролер відключає серворозрахунки, встановлює всі виходи на 0.0 і відключає підсилювачі. Якщо в файлі INI присутній MIN_FERROR, використовуються похибки слідування, пропорційні швидкості. Тут максимально допустима похибка пропорційна швидкості, причому FERROR застосовується до швидкої швидкості, встановленої [TRAJ]MAX_VELOCITY, а пропорційно менші похибки для повільніших швидкостей. Максимально допустима похибка завжди буде більшою за MIN_FERROR. Це запобігає випадковому припиненню руху через невеликі похибки для нерухомих осей. Невеликі похибки завжди будуть присутні через вібрацію тощо.
- LOCKING_INDEXER = 1 - Вказує на те, що шарнір використовується як фіксуючий індексатор.

Ці параметри пов'язані з перенаправленням, для кращого пояснення прочитайте розділ Налаштування перенаправлення.

- HOME = 0.0 - Положення, в яке перейде шарнір після завершення послідовності повернення до початкової точки.
- HOME_OFFSET = 0.0 - Спільне положення перемикача початкової позиції або імпульсу індексу в machine units. Коли під час процесу повернення до початкової позиції знайдено початкову точку, це положення присвоюється цій точці. При спільному використанні перемикачів початкової

позиції та кінцевого положення і використанні послідовності повернення в початкову позицію, яка залишає перемикач початкової позиції/кінцевого положення в перемиканому стані, зміщення початкової позиції може бути використане для визначення положення перемикача початкової позиції, відмінного від 0, якщо бажано, щоб положення HOME було 0.

- `HOME_SEARCH_VEL = 0.0` - Початкова швидкість повернення в вихідне положення в одиницях виміру машини за секунду. Знак позначає напрямок руху. Значення нуль означає, що поточне місцезнаходження є вихідним положенням машини. Якщо ваша машина не має вимикачів вихідного положення, залиште це значення нульовим.
- `HOME_LATCH_VEL = 0.0` - Швидкість повернення до початкового положення в одиницях машинного перемикача за секунду. Знак позначає напрямок руху.
- `HOME_FINAL_VEL = 0.0` - Швидкість у машинних одиницях за секунду від початкового положення фіксатора до початкового положення. Якщо залишити на 0 або не враховувати у швидкості шарніра, використовується швидкість. Має бути додатним числом.
- `HOME_USE_INDEX = NO` - Якщо енкодер, що використовується для цього з'єднання, має індексний імпульс, а карта руху має можливість приймати цей сигнал, ви можете встановити значення «yes». Якщо значення «yes», це вплине на тип використовуваного шаблону повернення в початкове положення. Наразі ви не можете повернутись в початкове положення за допомогою крокових двигунів, якщо не використовуєте StepGen у режимі швидкості та PID.
- HOME_INDEX_NO_ENCODER_RESET = NO — використовуйте YES, якщо енкодер, що використовуєть для цього з'єднання, не скидає лічильник при виявленні імпульсу індексу після активації виводу HAL index_enable з'єднання. Застосовується тільки для HOME_USE_INDEX = YES.
- `HOME_IGNORE_LIMITS = NO` - Коли ви використовуєте кінцевий вимикач як вимикач початкового положення, це значення слід встановити на YES. При встановленні значення YES кінцевий вимикач для цього з'єднання ігнорується під час повернення в початкове положення. Ви повинні налаштувати повернення в початкове положення таким чином, щоб в кінці руху в початкове положення вимикач початкового положення/кінцевий вимикач не перебував у перемикан стані, інакше після руху в початкове положення ви отримаєте помилку кінцевого вимикача.
- `HOME_IS_SHARED = <n>` - Якщо вхід додому використовується спільно більш ніж одним спільним набором, встановіть <n> на 1, щоб запобігти запуску повернення додому, якщо один із спільних перемикачів уже закритий. Встановіть <n> на 0, щоб дозволити повернення додому, якщо перемикач закритий.
- `HOME_ABSOLUTE_ENCODER = 0 | 1 | 2` - Використовується для вказівки, що суглоб використовує абсолютний енкодер. При запиті на повернення в початкове положення поточне значення суглоба встановлюється на значення `HOME_OFFSET`. Якщо налаштування `HOME_ABSOLUTE_ENCODER` дорівнює 1, машина виконує звичайний кінцевий рух до значення `HOME`. Якщо налаштування `HOME_ABSOLUTE_ENCODER` дорівнює 2, кінцевий рух не виконується.
- `HOME_SEQUENCE = <n>` - Використовується для визначення послідовності «Home All». <n> повинен починатися з `0`, `1` або `-1`. Додаткові послідовності можуть бути вказані з числами, що збільшуються на 1 (в абсолютному значенні). Пропускати номери послідовностей не дозволяється. Якщо `HOME_SEQUENCE` пропущено, шарнір не буде повернено в початкове положення за допомогою функції «Home All». Одночасно можна повернути в початкове положення більше одного з'єднання, вказавши однаковий номер послідовності для декількох з'єднань. Від'ємний номер послідовності використовується для відстрочки остаточного переміщення всіх з'єднань, що мають цей (від'ємний або додатний) номер послідовності. Додаткову інформацію див.: HOME SEQUENCE.
- `VOLATILE_HOME = 0` - Якщо ця опція увімкнена (встановлено значення 1), цей шарнір буде виведений з вихідного положення, якщо живлення машини вимкнено або увімкнено аварійне зупинення. Це корисно, якщо ваша машина має вимикачі вихідного положення і не має зворотного зв'язку щодо положення, наприклад, машина з кроковим приводом і приводом за напрямком.

Ці параметри стосуються з'єднань, керованих сервоприводами.

Warning
Нижче наведено власні записи INI-файлу, які ви можете знайти у зразковому INI-файлі або у файлі, створеному за допомогою майстра. Вони не використовуються програмним забезпеченням LinuxCNC. Вони призначені лише для того, щоб зібрати всі налаштування в одному місці. Більш детальну інформацію про власні записи INI-файлу див. у підрозділі Власні розділи та змінні.

Наступні елементи можуть використовуватися компонентом PID, і припущення полягає в тому, що вихідний сигнал – це вольти.

- `DEADBAND = 0.000015` - Наскільки близько це достатньо близько, щоб вважати двигун у положенні, у machine units.

  Часто це значення встановлюється на відстань, еквівалентну 1, 1,5, 2 або 3 імпульсам енкодера, але суворих правил немає. Більш вільні (більші) налаштування дозволяють зменшити «полювання» сервоприводу за рахунок зниження точності. Більш жорсткі (менші) налаштування забезпечують вищу точність за рахунок більшого «полювання» сервоприводу. Чи дійсно це більш точно, якщо це також більш невизначено? Як правило, якщо це можливо, краще уникати або принаймні обмежувати «полювання» сервоприводу.

  Будьте обережні, не опускаючись нижче 1 лічильника енкодера, оскільки ви можете створити ситуацію, в якій сервопривід не зможе нормально працювати. Це може призвести до «полювання» (повільного), «нервозності» (швидкого) і навіть до «скрипу», який легко сплутати з коливанням, спричиненим неправильним налаштуванням. Краще спочатку залишити один-два лічильники вільними, поки ви не пройдете хоча б «грубе налаштування».

  Приклад розрахунку машинних одиниць на імпульс енкодера для використання при визначенні значення `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` - Використовується hm2-servo та деякими іншими. Bias — це постійна величина, яка додається до вихідного сигналу. У більшості випадків її слід залишати на рівні нуля. Однак іноді вона може бути корисною для компенсації зміщень у сервопідсилювачах або для врівноваження ваги об'єкта, що рухається вертикально. Bias вимикається, коли цикл PID вимкнений, так само як і всі інші компоненти вихідного сигналу.
- `P = 50` - Пропорційний коефіцієнт підсилення для спільного сервоприводу. Це значення помножує похибку між заданим і фактичним положенням в одиницях машини, що призводить до впливу на обчислюване напруження для підсилювача двигуна. Одиниці виміру коефіцієнта підсилення P - вольти на одиницю машини, наприклад, $\frac{volts}{unit}$
- `I = 0` - Інтегральне підсилення для спільного сервоприводу. Це значення множиться на сукупну похибку між заданим і фактичним положенням в одиницях машини, що дає вклад у обчислюване напруження для підсилювача двигуна. Одиниці виміру підсилення I - вольти на одиницю машини в секунду, наприклад, $\frac{volts}{unit\ second}$
- `D = 0` - Похідна коефіцієнт підсилення для суглобового сервоприводу. Це значення множиться на різницю між поточним і попереднім помилками, що призводить до внеску в обчислюване напруження для підсилювача двигуна. Одиниці виміру коефіцієнта підсилення D - вольти на одиницю машини в секунду, наприклад, $\frac{volts}{unit\ second}$

- `FF0 = 0` - Коефіцієнт підсилення прямого зв'язку 0-го порядку. Це число множиться на задане положення, що дає вклад у розраховане напруження для підсилювача двигуна. Одиниці виміру коефіцієнта підсилення FF0 - вольти на одиницю машини, наприклад, $\frac{volts}{unit}$
- `FF1 = 0` - Коефіцієнт підсилення прямого зв'язку 1-го порядку. Це число множиться на зміну заданого положення за секунду, що дає вклад у обчислюване напруження для підсилювача двигуна. Одиниці виміру коефіцієнта підсилення FF1 - вольти на одиницю машини за секунду, наприклад, $\frac{volts}{unit\ second}$
- `FF2 = 0` - Коефіцієнт підсилення прямого зв'язку 2-го порядку. Це число множиться на зміну заданого положення в секунду на секунду, що дає вклад у розраховане напруження для підсилювача двигуна. Одиниці виміру коефіцієнта підсилення FF2 - вольти на одиницю машини в секунду на секунду, наприклад, $\frac{volts}{unit\ second^2}$
- `OUTPUT_SCALE = 1.000`
- `OUTPUT_OFFSET = 0.000`

  Ці два значення є коефіцієнтами масштабування та зміщення для спільного виходу на підсилювачі двигуна.

  Друге значення (зсув) віднімається від обчисленого виходу (у вольтах) і ділиться на перше значення (коефіцієнт масштабування) перед записом у цифро-аналогові перетворювачі. Одиниці виміру на шкалі значень виражені у вольтах на вихідний вольт цифро-аналогового перетворювача. Одиниці виміру на шкалі значень зсуву виражені у вольтах. Вони можуть використовуватися для лінеаризації цифро-аналогового перетворювача. Зокрема, під час запису вихідних даних LinuxCNC спочатку перетворює бажаний вихід у квазі-SI одиницях у необроблені значення приводу, наприклад, вольти для підсилювача DAC. Це масштабування виглядає так: $raw = \frac{output - of}{scale}$

  Значення масштабу можна отримати аналітично, виконавши аналіз одиниць, тобто одиниці вимірювання [одиниці вимірювання SI на виході]/[одиниці вимірювання приводу]. Наприклад, на машині з підсилювачем режиму швидкості, де 1 В відповідає швидкості 250 мм/с.

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

  Зверніть увагу, що одиниці зміщення вимірюються в одиницях машини, наприклад мм/с, і вони заздалегідь віднімаються від показань датчика. Значення цього зміщення отримується шляхом визначення значення вашого виходу, яке дає 0,0 для виходу приводу. Якщо DAC лінеаризований, це зміщення зазвичай дорівнює 0,0.

  Масштаб та зміщення також можна використовувати для лінеаризації DAC, що призводить до значень, що відображають комбінований вплив коефіцієнта посилення підсилювача, нелінійності DAC, одиниць DAC тощо.

  Для цього виконайте таку процедуру.

  1. Створіть калібрувальну таблицю для вихідного сигналу, подаючи на DAC потрібну напругу та вимірюючи результат.
  2. Виконайте лінійну апроксимацію методом найменших квадратів, щоб отримати коефіцієнти a, b такі, що $measured = a * raw + b$
  3. Зверніть увагу, що нам потрібен необроблений вихідний сигнал, щоб наш виміряний результат був ідентичним заданому виходу. Це означає
     a. $command = a * raw + b$
     b. $raw = (command - b)/a$

4. В результаті, коефіцієнти a та b з лінійної апроксимації можна використовувати безпосередньо як масштаб та зміщення для контролера.

Дивіться наступну таблицю для прикладу вимірювань напруги.

Table 4.1: Вимірювання вихідної напруги

Сире	Виміряно
-10	-9.93
-9	-8.83
0	-0.03
1	0.96
9	9.87
10	10.87

- `MAX_OUTPUT = 10` - Максимальне значення вихідного сигналу PID-компенсації, яке записується в підсилювач двигуна, у вольтах. Обчислене значення вихідного сигналу обмежується цим значенням. Обмеження застосовується перед масштабуванням до вихідних одиниць. Значення застосовується симетрично як до плюсової, так і до мінусової сторони.
- `INPUT_SCALE = 20000` - у зразках конфігурацій
- `ENCODER_SCALE = 20000` - у вбудованих конфігураціях PnCconf

Вказує кількість імпульсів, що відповідає переміщенню однієї одиниці машини, як встановлено в розділі `[TRAJ]`. Для лінійного з'єднання одна одиниця машини буде дорівнювати налаштуванню `LINEAR_UNITS`. Для кутового з'єднання одна одиниця дорівнює налаштуванню в `ANGULAR_UNITS`. Друге число, якщо воно вказане, ігнорується. Наприклад, для енкодера з 2000 імпульсами на оберт, передачі 10 обертів/дюйм і бажаних одиниць виміру в дюймах ми маємо:

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

Ці параметри стосуються з'єднань, керованих кроковими двигунами.



Warning
Нижче наведено власні записи INI-файлу, які ви можете знайти у зразковому INI-файлі або у файлі, створеному за допомогою майстра. Вони не використовуються програмним забезпеченням LinuxCNC і призначені лише для того, щоб зібрати всі налаштування в одному місці. Більш детальну інформацію про власні записи INI-файлу див. у підрозділі Власні розділи та змінні.

Компонент StepGen може використовувати наступні елементи.

- `SCALE = 4000` - у Зразках конфігурацій
- `STEP_SCALE = 4000` - у вбудованих конфігураціях PnCconf

Вказує кількість імпульсів, що відповідає переміщенню однієї одиниці машини, як встановлено в розділі `[TRAJ]`. Для крокових систем це кількість імпульсів кроку, що видаються на одиницю машини. Для лінійного з'єднання одна одиниця машини буде дорівнювати налаштуванню `LINEAR_UNIT` Для кутового з'єднання одна одиниця дорівнює налаштуванню в `ANGULAR_UNITS`. Для сервосистем це кількість імпульсів зворотного зв'язку на одиницю машини. Друге число, якщо воно вказане, ігнорується.

Наприклад, для крокового двигуна на 1,8 градуса з напівкроковим перемиканням, зубчастою передачею 10 обертів/дюйм та бажаними machine units дюймів, ми маємо:

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

Note
У старих файлах INI та HAL для цього значення використовувалося `INPUT_SCALE`.

- `ENCODER_SCALE = 20000` (опціонально використовується в конфігураціях, створених за допомогою PnCconf) - Вказує кількість імпульсів, що відповідає переміщенню на одну одиницю машини, як встановлено в розділі `[TRAJ]`. Для лінійного з'єднання одна одиниця машини буде дорівнювати налаштуванню `LINEAR_UNITS`. Для кутового з'єднання одна одиниця дорівнює налаштуванню в `ANGULAR_UNITS`. Друге число, якщо воно вказане, ігнорується. Наприклад, для енкодера з 2000 імпульсами на оберт, передачі 10 обертів/дюйм і бажаних одиниць виміру в дюймах ми маємо:

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

- `STEPGEN_MAXACCEL = 21.0` - Обмеження прискорення для генератора кроків. Воно повинно бути на 1% до 10% більшим, ніж спільне `MAX_ACCELERATION`. Це значення покращує налаштування «позиційного циклу» StepGen. Якщо ви додали компенсацію люфту до з'єднання, то воно повинно бути в 1,5-2 рази більшим, ніж `MAX_ACCELERATION`.
- `STEPGEN_MAXVEL = 1.4` - Старіші файли конфігурації також мають обмеження швидкості для генератора кроків. Якщо воно вказане, воно також повинно бути на 1% до 10% більшим, ніж спільне `MAX_VELOCITY`. Подальші випробування показали, що використання `STEPGEN_MAXVEL` не покращує налаштування позиційного циклу StepGen.

4.4.2.14 [SPINDLE_<num>] Розділ(и))

<num> вказує номер шпинделя 0 ... (num_spindles-1)
Значення *num_spindles* встановлюється за допомогою `[TRAJ]SPINDLES=` .
За замовчуванням максимальна швидкість шпинделя в прямому та зворотному напрямках становить приблизно 2147483000 об/хв.
За замовчуванням мінімальна швидкість шпинделя в прямому та зворотному напрямках становить 0 об/хв.
За замовчуванням приріст становить 100 об/хв.
Ви можете змінити ці значення за замовчуванням, встановивши наступні змінні INI:

Note
Ці налаштування стосуються компонента контролера руху. Екрани керування можуть додатково обмежувати ці налаштування.

- `MAX_FORWARD_VELOCITY = 20000` Максимальна швидкість шпинделя (в об/хв) для зазначеного шпинделя. Необов'язково. Це також встановить MAX_REVERSE_VELOCITY на від'ємне значення, якщо не перевизначено.
- `MIN_FORWARD_VELOCITY = 3000` Мінімальна швидкість обертання шпинделя (в об/хв) для вказаного шпинделя. Необов'язково. Багато шпинделів мають мінімальну швидкість, нижче якої вони не повинні працювати. Будь-яка команда швидкості обертання шпинделя нижче цього обмеження буде /збільшена/ до цього обмеження.
- `MAX_REVERSE_VELOCITY = 20000` Якщо це значення не вказано, за замовчуванням буде використовува значення `MAX_FORWARD_VELOCITY`. Воно може використовуватися у випадках, коли швидкість шпинделя в зворотному напрямку обмежена. Для шпинделів, які не повинні працювати в зворотному напрямку, встановіть значення нуль. У цьому контексті «максимальне» означає абсолютну величину швидкості шпинделя.

- `MIN_REVERSE_VELOCITY = 3000` `Цей параметр еквівалентний MIN_FORWARD_VELOCITY, але для зворотного обертання шпинделя. Якщо його пропустити, за замовчуванням використовуватиметься MIN_FORWARD_VELOCITY.
- `INCREMENT = 200` Встановлює розмір кроку для команд збільшення/зменшення швидкості шпинделя. Це значення може бути різним для кожного шпинделя. Це налаштування діє для AXIS і Touchy, але зверніть увагу, що деякі екрани керування можуть обробляти дані по-різному.
- `HOME_SEARCH_VELOCITY = 100` - FIXME: Повернення шпинделя в початкове положення ще не працює. Встановлює швидкість повернення (об/хв) для шпинделя. Шпиндель буде обертатися з цією швидкістю під час послідовності повернення в початкове положення, доки не буде знайдено індекс шпинделя, після чого положення шпинделя буде встановлено на нуль. Зверніть увагу, що немає сенсу встановлювати для вихідного положення шпинделя будь-яке інше значення, крім нуля, тому така можливість не передбачена.
- `HOME_SEQUENCE = 0` - FIXME: Повернення шпинделя до вихідного положення ще не працює. Контролює, в якій точці загальної послідовності повернення до вихідного положення відбуваються обертання шпинделя. Встановіть `HOME_SEARCH_VELOCITY` на нуль, щоб уникнути обертання шпинделя під час послідовності повернення до вихідного положення.

4.4.2.15 [EMCIO] Розділ

- `TOOL_TABLE = tool.tbl` - Файл, що містить інформацію про інструмент, описану в посібнику користувача.
- `DB_PROGRAM = db_program` - Шлях до виконуваного файлу програми, яка керує даними інструменту. Якщо вказано DB_PROGRAM, запис TOOL_TABLE ігнорується.
- `TOOL_CHANGE_POSITION = 0 0 2` - Вказує положення XYZ, до якого слід переміститися під час зміни інструменту, якщо використовуються три цифри. Вказує положення XYZABC, якщо використовуються 6 цифр. Вказує положення XYZABCUVW, якщо використовуються 9 цифр. Зміни інструменту можна комбінувати. Наприклад, якщо комбінувати підйом пінолі з позицією зміни, можна спочатку перемістити Z, а потім X і Y.
- `TOOL_CHANGE_WITH_SPINDLE_ON = 1` - Шпиндель залишатиметься увімкненим під час зміни інструмен якщо значення дорівнює 1. Корисно для токарних верстатів або машин, де матеріал знаходиться в шпинделі, а не в інструменті.
- `TOOL_CHANGE_QUILL_UP = 1` - Вісь Z буде переміщена до нульового положення верстата перед зміною інструменту, якщо значення дорівнює 1. Це те саме, що й видача `G0 G53 Z0`.
- `TOOL_CHANGE_AT_G30 = 1` - Якщо значення дорівнює 1, верстат переміщується до опорної точки, визначеної параметрами 5181-5186 для G30. Докладнішу інформацію див. у розділах G-code Parameters та G-code G30-G30.1.
- `RANDOM_TOOLCHANGER = 1` - Це стосується верстатів, які не можуть повернути інструмент у гніздо, з якого він вийшов. Наприклад, верстати, які обмінюють інструмент в активному гнізді інструментом у шпинделі.

4.5 Конфігурація самонаведення

4.5.1 Огляд

Повернення в початкове положення встановлює нульову точку координат верстата G53. М'які обмеження визначаються відносно початкової точки верстата. М'які обмеження автоматично уповільнюють і зупиняють осі, перш ніж вони досягнуть кінцевих вимикачів. Правильно налаштований

і функціонуючий верстат не буде рухатися за межі м’яких (програмних) обмежень, а початкова точка верстата буде встановлена так само повторно, як і механізм повернення в початкове положення/ Linuxcnc можна повернути в початкове положення на око (за допомогою вирівнювальних міток), за допомогою вимикачів, за допомогою вимикачів і індекса кодера або за допомогою абсолютних кодерів. Повернення в початкове положення здається досить простим — просто перемістіть кожну шарнірну частину в відоме місце і відповідно налаштуйте внутрішні змінні LinuxCNC. Однак різні машини мають різні вимоги, і повернення в початкове положення насправді є досить складним.

Note
Хоча LinuxCNC можна використовувати без перемикачів відправлення/процедур відправлення додому або кінцевих вимикачів, це позбавляє додаткової безпеки програмних обмежень.

4.5.2 Передумова

Самонаведення спирається на деякі фундаментальні припущення щодо машини.

- Негативні та позитивні напрямки базуються на Tool Movement, що може відрізнятися від фактичного руху верстата. Тобто, на фрезерному верстаті зазвичай рухається стіл, а не інструмент.
- Все посилається на нульовий початок координат верстата G53, початок координат може бути будь-де (навіть поза межами того місця, де можна переміститися)
- Початок нульової точки машини G53 зазвичай знаходиться в межах області програмних обмежень, але не обов’язково.
- Зміщення перемикача самонаведення встановлює місцезнаходження початку координат, але навіть воно посилається на початок координат.
- Під час використання хомінгу за індексом енкодера зміщення початкового перемикача обчислюєтьс з опорного положення енкодера після спрацювання початкового перемикача.
- Негативні програмні межі – це максимальне значення, яке можна перемістити в негативному напрямку після повернення до початкового положення (але вони можуть бути не негативними в абсолютному сенсі)
- Позитивні межі м’якого (програмного) забезпечення — це максимальне переміщення в позитивному напрямку після повернення в початкове положення. (але вони можуть бути не позитивними в абсолютному сенсі, хоча зазвичай їх встановлюють як позитивні числа)
- Програмні (виробничі) обмеження знаходяться всередині області кінцевих вимикачів.
- (Кінцеве) положення виходу всередині області м’якого обмеження
- (Якщо використовується домашнє переміщення на основі перемикачів), перемикач(і) домашнього переміщення або використовує(ють) кінцеві вимикачі (спільний домашній перемикач / кінцевий вимикач), або, якщо використовується окремий домашній перемикач, знаходиться(ють) всередині області кінцевих вимикачів.
- Якщо ви використовуєте окремий перемикач повернення до вихідної позиції, можна почати повернення до вихідної позиції з неправильної сторони перемикача, що в поєднанні з опцією HOME_IGNORE_LIMITS призведе до серйозної аварії. Ви можете уникнути цього, змусивши перемикач повернення до вихідної позиції змінювати свій стан, коли тригер знаходиться на певній стороні, доки він знову не пройде точку спрацьовування. Іншими словами, стан перемикача повернення в початкове положення повинен відображати положення фіксатора відносно перемикач (тобто *до* або *після* перемикача) і повинен залишатися таким, навіть якщо фіксатор проходить повз перемикач у тому ж напрямку.

Note
Хоча можна використовувати LinuxCNC з початком координат G53 поза межами програмних обмежень верстата, якщо ви використовуєте G28 або G30 без налаштування параметрів, він за замовчуванням повертається до початку координат. Це призведе до спрацьовування кінцевих вимикачів до досягнення позиції.

4.5.3 Приклад розташування окремого домашнього вимикача

У цьому прикладі показано мінімальні та максимальні кінцеві вимикачі з окремим початковим вимикачем.

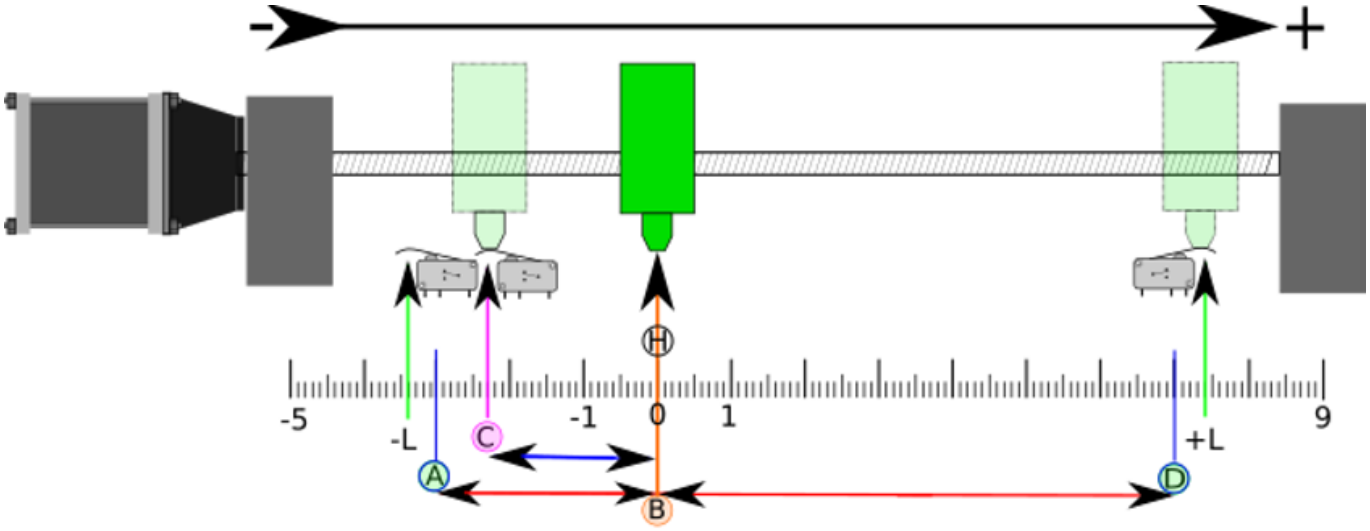


Figure 4.6: Демонстративне розташування окремих перемикачів

- A – негативна м’яка межа
- B – це координата машини G53
- C – точка спрацьовування домашнього вимикача
- D – додатна м’яка межа
- H – кінцеве положення «дома» (HOME) = 0 одиниць
- -L та +L – це точки спрацьовування кінцевих вимикачів
- A<->B – це від’ємні м’які межі (MIN_LIMITS) = -3 одиниці
- B<->C – це домашнє_зміщення (HOME_OFFSET) = -2,3 одиниці
- B<->D – це додатні м’які ліміти (MAX_LIMITS) = 7 одиниць
- A<->D – загальний пробіг = 10 одиниць
- У цьому прикладі збільшено відстань між кінцевими вимикачами та програмними обмежувачами (-L<->A та D<-+L)
- Зверніть увагу, що між кінцевими вимикачами та фактичним фізичним жорстким контактом існує відстань для вибігу після вимкнення підсилювача.

Note
Повернення до вихідної позиції встановлює систему координат G53, тоді як початок координат (нульова точка) може бути будь-де. Встановлення нульової точки на від'ємному м'якому обмеженні робить усі координати G53 додатними, що, ймовірно, найлегше запам'ятати. Зробіть це, встановивши MIN_LIMIT = 0 і переконавшись, що MAX_LIMIT є додатним.

4.5.4 Приклад схеми спільного граничного/домашнього перемикача

У цьому прикладі показано граничний вимикач максимального значення та комбінований граничний/д
вимикач мінімального значення.

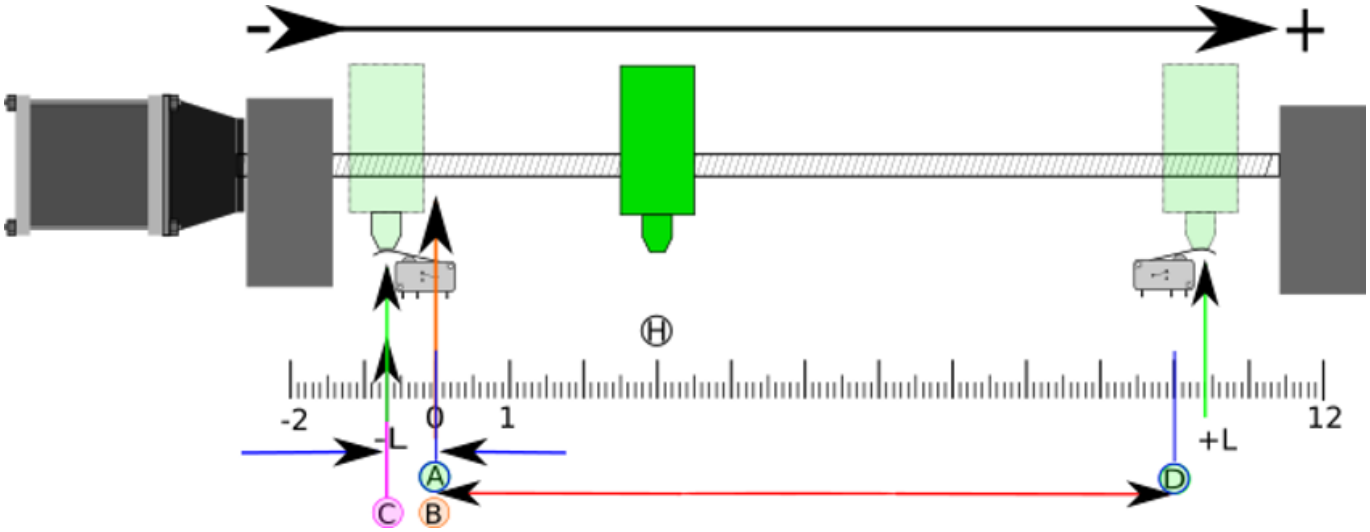


Figure 4.7: Демонстративна схема спільного комутатора

- A – негативна м'яка межа.
- B – це початок координат машини G53.
- C – це точка спрацьовування домашнього перемикача, спільна з точкою спрацьовування мінімальног
граничного значення (-L).
- D – додатна м'яка границя.
- H – кінцеве положення «дому» (HOME) = 3 одиниці.
- -L та +L – це точки спрацювання кінцевого вимикача.
- A<->B – це від'ємні м'які межі (MIN_LIMITS) = 0 одиниць.
- B<->C – це зміщення_до_дома (HOME_OFFSET) = -0,7 одиниці.
- B<->D – це додатні м'які ліміти (MAX_LIMITS) 10 одиниць.
- A<->D – загальний пробіг = 10 одиниць.
- У цьому прикладі збільшено відстань між кінцевими вимикачами та програмними граничними
вимикачами (-L<->A та D<->+L).
- Зверніть увагу, що між кінцевими вимикачами та фактичним фізичним жорстким контактом
існує відстань для вибігу після вимкнення підсилювача.

4.5.5 Послідовність самонаведення

Існує чотири можливі послідовності повернення до вихідної позиції, визначені знаком HOME_SEARCH_
і HOME_LATCH_VEL, а також відповідними параметрами конфігурації, як показано в наступній таблиці. Існують дві основні умови: HOME_SEARCH_VEL і HOME_LATCH_VEL мають однаковий знак або протилежні знаки. Більш детальний опис функцій кожного параметра конфігурації наведено в наступному розділі.

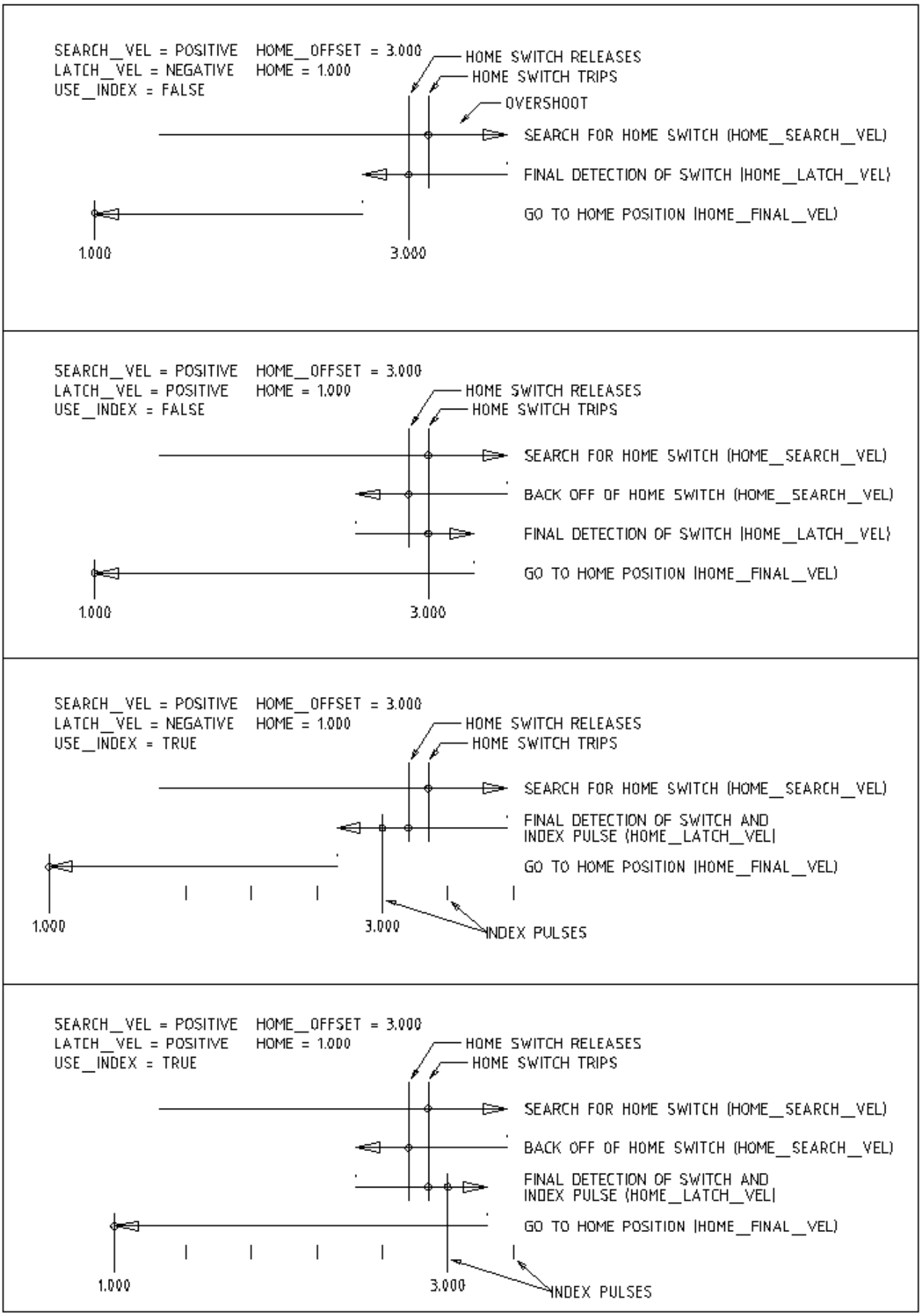


Figure 4.8: Послідовності самонаведення

4.5.6 Конфігурація

Наведені нижче дії точно визначають, як поводиться домашня послідовність. Вони визначені в розділі [JOINT_n] INI-файлу.

Тип самонаведення	HOME_SEARCH_VEL	HOME_LATCH_VEL	HOME_USE_INDEX
Негайно	0	0	НІ
Тільки індекс	0	ненульовий	ТАК
Тільки для комутатора	ненульовий	ненульовий	НІ
Перемикач та індекс	ненульовий	ненульовий	ТАК

Note
Будь-які інші комбінації можуть призвести до помилки.

4.5.6.1 HOME_SEARCH_VEL

Ця змінна вимірюється в одиницях машинних одиниць за секунду.

Значення за замовчуванням дорівнює нулю. Значення нуль призводить до того, що LinuxCNC вважатиме, що перемикач додому відсутній; етап пошуку для переміщення додому пропускається.

Якщо HOME_SEARCH_VEL не дорівнює нулю, то LinuxCNC припускає, що є перемикач початкового положення. Спочатку перевіряється, чи перемикач початкового положення вже спрацював. Якщо спрацьовує, він відсуває перемикач на HOME_SEARCH_VEL. Напрямок відсунення протилежний знаку HOME_SEARCH_VEL. Потім він шукає перемикач початкового положення, рухаючись у напрямку, визначеному знаком HOME_SEARCH_VEL, зі швидкістю, визначеною його абсолютним значенням. Коли перемикач початкового положення виявлено, з'єднання зупиниться якомога швидше, але завжди буде деяке перевищення. Величина перевищення залежить від швидкості. Якщо вона занадто висока, шарнір може перевищити межу настільки, що вдарить по кінцевому вимикачу або зіткнеться з кінцем ходу. З іншого боку, якщо HOME_SEARCH_VEL занадто низька, повернення в початкове положення може зайняти багато часу.

4.5.6.2 HOME_LATCH_VEL

Ця змінна вимірюється в одиницях машинних одиниць за секунду.

Вказує швидкість і напрямок, які LinuxCNC використовує при остаточному точному визначенні положення перемикача початкового положення (якщо він є) і імпульсу індексу (якщо він є). Зазвичай це буде повільніше, ніж швидкість пошуку, щоб максимізувати точність. Якщо HOME_SEARC і HOME_LATCH_VEL мають однаковий знак, то фаза фіксації виконується під час руху в тому ж напрямку, що і фаза пошуку. (У цьому випадку LinuxCNC спочатку відсуває перемикач, а потім знову рухається до нього зі швидкістю фіксації.) Якщо HOME_SEARCH_VEL і HOME_LATCH_VEL мають протилежні знаки, фаза фіксації виконується під час руху в напрямку, протилежному до фази пошуку. Це означає, що LinuxCNC зафіксує перший імпульс після відсунення перемикача. Якщо HOME_SEARCH_VEL дорівнює нулю (що означає відсутність перемикача початкового положення а цей параметр не дорівнює нулю, LinuxCNC переходить до пошуку імпульсу індексу. Якщо HOME_SEARCH_VEL не дорівнює нулю, а цей параметр дорівнює нулю, це є помилкою, і операція повернення в початкове положення не буде виконана. Значення за замовчуванням дорівнює нулю.

4.5.6.3 HOME_FINAL_VEL

Ця змінна вимірюється в одиницях машинних одиниць за секунду.

Він визначає швидкість, яку LinuxCNC використовує при переміщенні з HOME_OFFSET в положення HOME. Якщо HOME_FINAL_VEL відсутній в файлі INI, то для цього переміщення використовується максимальна швидкість з'єднання. Значення повинно бути додатним числом.

4.5.6.4 HOME_IGNORE_LIMITS

Може містити значення YES / NO. Значенням за замовчуванням для цього параметра є NO. Цей прапорець визначає, чи буде LinuxCNC ігнорувати вхідний сигнал кінцевого вимикача для цього з'єднання під час повернення в початкове положення. Це налаштування не ігноруватиме вхідні сигнали кінцевих вимикачів для інших з'єднань. Якщо у вас немає окремого вимикача початкового положення, встановіть для цього параметра значення YES і підключіть сигнал кінцевого вимикача до входу вимикача початкового положення з'єднання в HAL. LinuxCNC ігноруватиме вхідний сигнал кінцевого вимикача для цього з'єднання під час повернення в початкове положення. Щоб використовувати тільки один вхід для всіх повернень у вихідне положення та обмежень, вам доведеться заблокувати сигнали кінцевих вимикачів з'єднань, які не повертаються у вихідне положення, в HAL і повертати у вихідне положення по одному з'єднанню за раз.

4.5.6.5 HOME_USE_INDEX

Вказує, чи є імпульс індексації. Якщо прапор є істинним (HOME_USE_INDEX = YES), LinuxCNC буде фіксуватися на передньому фронті імпульсу індексації. Якщо прапор є хибним, LinuxCNC буде фіксуватися на передньому або задньому фронті перемикача початкового положення (залежно від знаків HOME_SEARCH_VEL та HOME_LATCH_VEL). Значенням за замовчуванням є NO.

Note
HOME_USE_INDEX вимагає з'єднань у вашому HAL-файлі з `joint.n.index-enable` з `encoder.n.index-enable`.

4.5.6.6 HOME_INDEX_NO_ENCODER_RESET

За замовчуванням встановлено значення NO. Використовуйте значення YES, якщо енкодер, що використовується для цього з'єднання, не скидає лічильник при виявленні імпульсу індексу після активації виводу HAL index_enable з'єднання. Застосовується тільки для HOME_USE_INDEX = YES.

4.5.6.7 HOME_OFFSET

Це визначає розташування нульової точки початку координат системи координат верстата G53. Це відстань (зсув) у одиницях з'єднання від початку координат верстата до точки спрацьовування вимикача або імпульсу індексу. Після виявлення точки спрацьовування вимикача/імпульсу індексу LinuxCNC встановлює положення координат з'єднання на HOME_OFFSET, тим самим визначаючи початок координат, від якого відраховуються м'які обмеження. Значення за замовчуванням дорівнює нулю.

Note
Розташування перемикача дому, як зазначено змінною HOME_OFFSET, може бути всередині або поза програмними граничними вимикачами. Вони будуть спільними з апаратними кінцевими вимикачами або всередині них.

4.5.6.8 HOME

Позиція, в яку переміститься шарнір після завершення послідовності повернення в початкове положення. Після виявлення перемикача початкового положення або перемикача початкового положення, а потім імпульсу індексації (залежно від конфігурації) та встановлення координати цієї точки в HOME_OFFSET, LinuxCNC виконує переміщення в HOME як останній крок процесу повернення в початкове положення. Значення за замовчуванням дорівнює нулю. Зверніть увагу, що навіть якщо цей параметр збігається з HOME_OFFSET, шарнір трохи перевищить фіксовану позицію під час зупинки. Тому в цей момент завжди буде невеликий рух (якщо HOME_SEARCH_VEL не дорівнює нулю і весь етап пошуку/фіксації не був пропущений). Цей остаточний рух буде виконаний з максимальною швидкістю шарніра, якщо не встановлено HOME_FINAL_VEL.

Note
Відмінність між «HOME_OFFSET» і «HOME» полягає в тому, що «HOME_OFFSET» спочатку встановлює початкове положення і масштаб на машині, застосовуючи значення «HOME_OFFSET» до місця, де було знайдено початкове положення, а потім «HOME» вказує, куди повинен переміститися шарнір на цьому масштабі.

4.5.6.9 HOME_IS_SHARED

Якщо для цього з'єднання немає окремого входу для перемикача повернення в початкове положення, але є кілька миттєвих перемикачів, підключених до одного і того ж контакту, встановіть це значення на 1, щоб запобігти запуску повернення в початкове положення, якщо один із спільних перемикачів вже закритий. Встановіть це значення на 0, щоб дозволити повернення в початкове положення, навіть якщо перемикач вже закритий.

4.5.6.10 HOME_ABSOLUTE_ENCODER

Використовується для абсолютних енкодерів. Коли надходить запит на переведення з'єднання в початкове положення, поточне положення з'єднання встановлюється на значення *[JOINT_n]HOME_OF*

Остаточний перехід до позиції *[JOINT_n]HOME* є необов'язковим відповідно до налаштування *HOME_ABSOLUTE_ENCODER*:

```
HOME_ABSOLUTE_ENCODER = 0 (b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb'' ↩
    b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'') b''шb''b''ab''b''pb''b''нb''b''іb''b''pb'' b ↩
    ''нb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b' ↩
    'уb''b''єb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b' ↩
    'eb''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''иb''b' ↩
    'йb'' b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb'', b''ob''b''cb''b''тb''b''ab''b' ↩
    'тb''b''ob''b''чb''b''нb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''іb''b''щb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' [JOINT_n]HOME.
HOME_ABSOLUTE_ENCODER = 2 b''Ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b' ↩
    'йb'' b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb'', b''Бb''b''Eb''b''Зb'' b''ob''b' ↩
    'cb''b''тb''b''ab''b''тb''b''ob''b''чb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''eb''b' ↩
    'pb''b''eb''b''мb''b''іb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' [JOINT_n] ↩
    HOME.
```

Note
Налаштування HOME_IS_SHARED ігнорується без попередження.

Note
Прохання про пересадку суглоба мовчки ігнорується.

4.5.6.11 HOME_SEQUENCE

Використовується для визначення послідовності повернення до вихідного положення для декількох шарнірів **HOME ALL** та забезпечення порядку повернення до вихідного положення (наприклад, Z не може бути повернутий до вихідного положення, якщо X ще не повернутий до вихідного положення). Суглоб може бути повернений у вихідне положення після того, як всі суглоби з нижчим (абсолютним значенням) HOME_SEQUENCE вже були повернені у вихідне положення і знаходяться в HOME_OFFSET. Якщо два суглоби мають однакове HOME_SEQUENCE, вони можуть бути повернені у вихідне положення одночасно.

Note
Якщо HOME_SEQUENCE не вказано, то з'єднання не буде перенаправлено в початкове положення послідовністю **HOME ALL** (але може бути перенаправлено в початкове положення окремими командами перенаправлення, специфічними для з'єднання).

Початкове число HOME_SEQUENCE може дорівнювати 0, 1 (або -1). Абсолютне значення номерів послідовності повинно збільшуватися на одиницю — пропуск номерів послідовності не підтримується. Якщо номер послідовності пропущено, повернення до вихідної позиції **HOME ALL** зупиниться після завершення останнього дійсного номера послідовності.

Негативні значення HOME_SEQUENCE вказують, що суглоби в послідовності повинні **синхронізувати кінцевий рух** до [JOINT_n]HOME, чекаючи, поки всі суглоби в послідовності будуть готові. Якщо будь-який суглоб має **негативне** значення HOME_SEQUENCE, то всі суглоби з тим самим абсолютним значенням (позитивним або негативним) значення елемента HOME_SEQUENCE синхронізують кінцевий рух.

Негативне значення HOME_SEQUENCE також застосовується до команд повернення в початкове положення одного суглоба. Якщо значення HOME_SEQUENCE є **негативним**, всі суглоби, що мають однакове абсолютне значення HOME_SEQUENCE, будуть **повернуті в початкове положення разом із синхронізованим кінцевим рухом**. Якщо значення HOME_SEQUENCE дорівнює нулю або є позитивним, команда повернення суглоба в початкове положення поверне в початкове положення тільки вказаний суглоб.

Спільний режим переміщення суглобів з негативним HOME_SEQUENCE заборонений. У звичайних застосуваннях портальних систем таке переміщення може призвести до порушення вирівнювання (деформації). Зверніть увагу, що звичайне переміщення у світових координатах завжди доступне після повернення машини в початкове положення.

Приклади для системи з 3 суглобами

Дві послідовності (0,1), без синхронізації

```
[JOINT_0]HOME_SEQUENCE = 0
[JOINT_1]HOME_SEQUENCE = 1
[JOINT_2]HOME_SEQUENCE = 1
```

Дві послідовності, суглоби 1 та 2 синхронізовані

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

Зі змішаними позитивними та негативними значеннями, суглоби 1 та 2 синхронізовані

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE =  1
```

Одна послідовність, без синхронізації

```
[JOINT_0]HOME_SEQUENCE =  0
[JOINT_1]HOME_SEQUENCE =  0
[JOINT_2]HOME_SEQUENCE =  0
```

Одна послідовність, усі суглоби синхронізовані

```
[JOINT_0]HOME_SEQUENCE = -1
[JOINT_1]HOME_SEQUENCE = -1
[JOINT_2]HOME_SEQUENCE = -1
```

4.5.6.12 VOLATILE_HOME

Якщо це налаштування є істинним, цей шарнір стає нефіксованим, коли машина переходить у стан ВИМКНЕНО. Це підходить для будь-якого шарніра, який не зберігає положення, коли привід шарніра вимкнений. Деякі крокові приводи, особливо мікрокрокові приводи, можуть цього потребувати.

4.5.6.13 LOCKING_INDEXER

Якщо це з'єднання є блокувальним поворотним індексатором, воно розблокується перед поверненням до початкового положення та заблокується після цього.

4.5.6.14 Негайне самонаведення

Якщо шарнір не має перемикачів початкового положення або не має логічного початкового положення, як поворотний шарнір, і ви хочете, щоб цей шарнір повертався у початкове положення при натисканні кнопки «Home All» (Повернути все у початкове положення) в графічному інтерфейсі AXIS, то для цього шарніра необхідні наступні записи INI.

```
HOME_SEARCH_VEL = 0
HOME_LATCH_VEL = 0
HOME_USE_INDEX = NO
HOME_OFFSET = 0 (b''Ab''b''бb''b''ob'' b''зb''b''мb''b''iь''b''щb''b''eb''b''нb''b''нb''b' ↩
    'яb'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ob''b''їь'' b''пb''b' ↩
    'ob''b''зb''b''иb''b''цb''b''iь''b''їь'' (HOME))
HOME_SEQUENCE = 0 (b''ab''b''бb''b''ob'' b''iь''b''нb''b''шb''b''иb''b''йb'' b''дb''b''iь'' ↩
    b''йb''b''сb''b''нb''b''иb''b''йb'' b''пb''b''ob''b''pb''b''яb''b''дb''b''кb''b''ob''b' ↩
    'вb''b''иb''b''йb'' b''нb''b''ob''b''мb''b''eb''b''pb'')
```

Note
Значення за замовчуванням для невизначених HOME_SEARCH_VEL, HOME_LATCH_VEL, HOME_USE_INDEX, HOME та HOME_OFFSET дорівнюють **нулю**, тому їх можна опустити при запиті на негайне повернення в початкове положення. Зазвичай слід вказати дійсне число HOME_SEQUENCE, оскільки опущення HOME_SEQUENCE виключає суглоб із поведінки **HOME ALL**, як зазначено вище.

4.5.6.15 Запобігання самонаведенню

Вивід HAL (motion.homing-inhibit) призначений для заборони ініціювання самонаведення як для режиму "Home All", так і для режиму індивідуального спільного самонаведення.

Деякі системи використовують можливості синхронізації кінцевих рухів повернення в початкове положення суглобів, що контролюються негативними елементами файлу INI [JOINT_N]HOME_SEQUEN За замовчуванням, можливості синхронізації забороняють **суглобові** рухи перед поверненням в початкове положення, щоб запобігти **суглобовим** рухам, які можуть призвести до порушення вирівнювання машини (наприклад, викривлення порталу).

Системний інтегратор може дозволити **спільне** переміщення перед поверненням у вихідне положення за допомогою логіки HAL, яка перемикає елементи [JOINT_N]HOME_SEQUENCE. Ця логіка також повинна активувати контакт **motion.homing-inhibit**, щоб гарантувати, що повернення у вихідне положення не буде випадково ініційовано, коли ввімкнено **спільне** переміщення.

Приклад: Синхронізовані суглоби 0,1 з використанням негативної послідовності (-1) для синхронізован повернення в початкове положення з перемикачем (allow_jjog), який вибирає позитивну послідовність (1) для індивідуального **суглобового** переміщення перед поверненням в початкове положення (частковий 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: b''пb''b''ib''b''нb'', b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b''нb''b' ↩
    'иb''b''йb'' b''вb''b''ib''b''pb''b''тb''b''уb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'юb'' b''пb''b''ab''b''нb''b''eb''b''лb''b''лb''b''юb'' b''ab''b''бb''b''ob'' b''ab''b' ↩
    'пb''b''ab''b''pb''b''ab''b''тb''b''нb''b''иb''b''мb'' b''пb''b''eb''b''pb''b''eb''b' ↩
    'мb''b''иb''b''кb''b''ab''b''чb''b''eb''b''мb''
net hsequence_select <= allow_jjog
net hsequence_select => home_sequence_mux.sel
net hsequence_select => motion.homing-inhibit
```

Note
Піни INI HAL (наприклад, ini.N.home_sequence) недоступні до запуску milltask, тому виконання вищезазначених команд HAL слід відкласти за допомогою файлу HAL postgui або затриманого скрипта [APPLICATION]APP=.

Note
Для синхронізації руху декількох суглобів у реальному часі потрібні додаткові з'єднання HAL для штифтів типу Manual-Pulse-Generator (MPG) (`joint.N.enable`, `joint.N.scale`, `joint.N.counts`).

Приклад конфігурації симуляції (gantry_jjog.ini), який демонструє спільне штовхання під час використання негативних початкових послідовностей, знаходиться в каталозі: configs/sim/axis/gantry/.

4.6 Конфігурація токарного верстата

4.6.1 Площина за замовчуванням

Коли інтерпретатор LinuxCNC був вперше написаний, він був розроблений для фрезерних верстатів. Ось чому площиною за замовчуванням є XY (G17). Звичайний токарний верстат використовує тільки площину XZ (G18). Щоб змінити площину за замовчуванням, вставте наступний рядок у файл INI в розділі RS274NGC.

```
RS274NGC_STARTUP_CODE = G18
```

Вищезазначене можна перезаписати в програмі G-коду, тому завжди встановлюйте важливі речі у преамбулі файлу G-коду.

4.6.2 Налаштування INI

Наступні налаштування INI необхідні для режиму токарного верстата в Axis на додаток до звичайних налаштувань у файлі INI або замість них. Ці історичні налаштування використовують ідентичну кінематику (trivkins) та «три» з'єднання (0,1,2), що відповідають координатам x, y, z. З'єднання 1 для невикористаної осі y є необхідним, але не використовується в цих історичних конфігураціях. Модельовані конфігурації токарного верстата можуть використовувати ці історичні налаштування. GMOCCAPY також використовує згадані налаштування, але пропонує додаткові налаштування. Детальніше див. розділ GMOCCAPY.

```
[DISPLAY]
DISPLAY = axis
LATHE = 1
...

[KINS]
KINEMATICS = trivkins
JOINTS = 3

[TRAJ]
COORDINATES = X Z
...

[JOINT_0]
...
[JOINT_2]
...
[AXIS_X]
...
[AXIS_Z]
...
```

Завдяки включенню joints_axes можна створити простішу конфігурацію лише з двома необхідними з'єднаннями, вказавши trivkins за допомогою параметра *coordinates*=:

```
[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 Швидкий старт для степпера

У цьому розділі передбачається, що ви виконали стандартну інсталяцію з Live CD. Після інсталяції рекомендується підключити комп'ютер до Інтернету і дочекатися появи вікна менеджера оновлень, щоб завантажити останні оновлення для LinuxCNC і Ubuntu, перш ніж продовжувати.

4.7.1 Тест на затримку

Тест затримки визначає, наскільки повільно процесор вашого комп'ютера реагує на запит. Деякі апаратні засоби можуть переривати обробку, що може призвести до пропуску кроків під час роботи верстата з ЧПК. Це перше, що вам потрібно зробити. Дотримуйтесь інструкцій here, щоб запустити тест затримки.

4.7.2 Шерлайн

Якщо у вас є Sherline, доступно кілька попередньо визначених конфігурацій. Це можна зробити в головному меню CNC/EMC, потім вибрати конфігурацію Sherline, яка відповідає вашій, і зберегти копію.

4.7.3 Ксилотекс

Якщо у вас є Xylotex, ви можете пропустити наступні розділи та перейти одразу до Майстра налаштування крокового двигуна. LinuxCNC забезпечив швидке налаштування для машин Xylotex.

4.7.4 Інформація про машину

Зберіть інформацію про кожну вісь вашого верстата.

Час спрацьовування приводу вимірюється в наносекундах. Якщо ви не впевнені в часі спрацьовування, багато популярних приводів включені в майстер налаштування крокових двигунів. Зверніть увагу, що деякі новіші приводи Gecko мають інший час спрацьовування, ніж оригінальні. Список приводів (https://wiki.linuxcnc.org/) також знаходиться на веб-сайті LinuxCNC wiki, який підтримується користувачами, де представлено більше приводів.

Вісь	Тип приводу	Час кроку (нс)	Кроковий інтервал (нс)	Утримання директора (нс)	Налаштування директорії (нс)
X					
Y					
Z					

4.7.5 Інформація про розпіновку

Зберіть інформацію про з’єднання вашого комп’ютера з паралельним портом ПК.

Вихідний контакт	Типова функція	Якщо відрізняється	Вхідний пін	Типова функція	Якщо відрізняється
1	Вихід з екстреної зупинки		10	X Limit/Home	
2	X Крок		11	Y Limit/Home	
3	X Напрямок		12	Z Limit/Home	
4	Y Крок		13	A Limit/Home	
5	Y Напрямок		15	Зонд в	
6	Z Крок				
7	Z Напрямок				
8	A Крок				
9	A Напрямок				
14	Шпиндель за годинниковою стрілкою				
16	Шпиндель PWM				
17	Увімкнення підсилювача				

Зверніть увагу, що будь-які невикористані контакти слід встановити у випадаючому списку на «Невикористані». Їх завжди можна змінити пізніше, повторно запустивши StepConf.

4.7.6 Механічна інформація

Зберіть інформацію про кроки та передачі. Результатом цього є кількість кроків на одиницю виміру користувача, яка використовується для SCALE у файлі INI.

Вісь	Кроки/Оберти	Мікрокроки	Зуби двигуна	Зубці ходового гвинта	Крок ходового гвинта
X					
Y					
Z					

- «Кроки на оберт» – це кількість кроків, необхідних для обертання крокового двигуна на один оберт. Типово 200.

- «Мікрокроки» — це кількість кроків, необхідних приводу для переміщення крокового двигуна на один повний крок. Якщо мікрокрокування не використовується, це число дорівнюватиме 1. Якщо мікрокрокування використовується, значення залежатиме від апаратного забезпечення крокового приводу.
- «Зубці двигуна та зубці ходового гвинта» – це якщо у вас є якийсь редуктор (шестерні, ланцюг, ремінь ГРМ тощо) між двигуном та ходовим гвинтом. Якщо ні, то встановіть для них обидва значення на 1.
- «Крок ходового гвинта» — це величина переміщення (в одиницях виміру користувача) за один оборот ходового гвинта. Якщо ви використовуєте дюйми, то це дюйми на оборот. Якщо ви використовуєте міліметри, то це міліметри на оборот.

Кінцевий результат, який ви шукаєте, це кількість кроків виводу CNC, необхідних для переміщення однієї одиниці користувача (дюйми або мм).

Example 4.1 Одиниці дюйми

```
b''Kb''b''pb''b''ob''b''кb''b''ob''b''вb''b''иb''b''йb'' b''дb''b''вb''b''иb''b''гb''b' ↩
    'yb''b''нb'' = 200 b''кb''b''pb''b''ob''b''кb''b''ib''b''вb'' b''нb''b''ab'' b''ob''b' ↩
    'бb''b''eb''b''pb''b''тb''
b''Пb''b''pb''b''иb''b''вb''b''ib''b''дb'' = 10 b''мb''b''ib''b''кb''b''pb''b''ob''b''кb''b ↩
    ''pb''b''ob''b''кb''b''ib''b''вb'' b''нb''b''ab'' b''кb''b''pb''b''ob''b''кb''
b''3b''b''yb''b''бb''b''цb''b''ib'' b''дb''b''вb''b''иb''b''гb''b''yb''b''нb''b''ab'' = 20
b''3b''b''yb''b''бb''b''цb''b''ib'' b''хb''b''ob''b''дb''b''ob''b''вb''b''ob''b''гb''b' ↩
    'ob'' b''гb''b''вb''b''иb''b''нb''b''тb''b''ab'' = 40
b''Kb''b''pb''b''ob''b''кb'' b''хb''b''ob''b''дb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'гb''b''вb''b''иb''b''нb''b''тb''b''ab'' = 0,2000 b''дb''b''юb''b''йb''b''мb''b''ab'' b' ↩
    'нb''b''ab'' b''ob''b''бb''b''eb''b''pb''b''тb''
```

Згідно з наведеною вище інформацією, ходовий гвинт рухається на 0,200 дюйма за оберт. - Двигун обертається 2000 разів за 1 оберт ходового гвинта. - Привід приймає 10 мікрокрокових входів, щоб зробити крок один раз. - Приводу потрібно 2000 кроків, щоб повернути кроковий двигун на один оберт.

Отже, необхідний масштаб:

$$\frac{200 motorsteps}{1 motorrev} \times \frac{10 microsteps}{1 motorstep} \times \frac{2 motorrevs}{1 leadscrewrev} \times \frac{1 leadscrewrevs}{0.2000 inch} = \frac{20{,}000 microsteps}{inch}$$

Example 4.2 Одиниці вимірювання мм

```
    b''Kb''b''pb''b''ob''b''кb''b''ob''b''вb''b''иb''b''йb'' b''дb''b''вb''b''иb''b''гb''b' ↩
        'yb''b''нb'' = 200 b''кb''b''pb''b''ob''b''кb''b''ib''b''вb'' b''нb''b''ab'' b''ob'' ↩
        b''бb''b''eb''b''pb''b''тb''
b''Пb''b''pb''b''иb''b''вb''b''ib''b''дb'' = 8 b''мb''b''ib''b''кb''b''pb''b''ob''b''кb''b' ↩
    'pb''b''ob''b''кb''b''ib''b''вb'' b''нb''b''ab'' b''кb''b''pb''b''ob''b''кb''
b''3b''b''yb''b''бb''b''цb''b''ib'' b''дb''b''вb''b''иb''b''гb''b''yb''b''нb''b''ab'' = 30
b''3b''b''yb''b''бb''b''цb''b''ib'' b''хb''b''ob''b''дb''b''ob''b''вb''b''ob''b''гb''b' ↩
    'ob'' b''гb''b''вb''b''иb''b''нb''b''тb''b''ab'' = 90
b''Kb''b''pb''b''ob''b''кb'' b''хb''b''ob''b''дb''b''ob''b''вb''b''ob''b''гb''b''ob'' b' ↩
    'гb''b''вb''b''иb''b''нb''b''тb''b''ab'' = 5,00 b''мb''b''мb'' b''нb''b''ab'' b''ob''b' ↩
    'бb''b''eb''b''pb''b''тb''
```

З вищезазначеної інформації: - Ходовий гвинт переміщується на 5,00 мм за оберт. - Двигун обертається 3000 разів за 1 оберт ходового гвинта. - Привід приймає 8 мікрокрокових входів, щоб зробити крок один раз. - Приводу потрібно 1600 кроків, щоб повернути кроковий двигун на один оберт.

Отже, необхідний масштаб:

$$\frac{200\,full\,steps}{1\,rev} x \frac{8\,microsteps}{1\,step} x \frac{3\,revs}{1\,leadscrew\,rev} x \frac{1\,leadscrew\,rev}{5.00\text{mm}} = \frac{960\,steps}{1\,mm}$$

4.8 Конфігурація крокового двигуна

4.8.1 Вступ

Найкращий спосіб налаштування стандартного крокового двигуна – за допомогою майстра налаштуван кроків. Див. розділ Майстер налаштування кроків.

У цьому розділі описано деякі з найпоширеніших налаштувань для ручного налаштування системи на основі крокових двигунів. Ці системи використовують крокові двигуни з приводами, які приймають сигнали кроку та напрямку.

Це одна з найпростіших схем, оскільки двигуни працюють у розімкнутому циклі (зворотний зв'язок від двигунів не надходить), проте систему потрібно правильно налаштувати, щоб двигуни не зупинялися та не втрачали кроки.

Більша частина цього розділу базується на зразку конфігурації, випущеному разом із LinuxCNC. Конфігурація називається stepper_inch і її можна знайти, запустивши Вибір конфігурації.

4.8.2 Максимальна швидкість кроку

При програмному формуванні кроків максимальна частота кроків становить один крок на два BASE_PERIOD для виводу кроків і напрямків. Максимальна запитувана частота кроків є добутком MAX_VELOCITY осі та її INPUT_SCALE. Якщо запитувана частота кроків є недосяжною, виникають такі помилки, особливо під час швидких переміщень і рухів G0.

Якщо ваш драйвер крокового двигуна може приймати квадратурний вхідний сигнал, використовуйте цей режим. З квадратурним сигналом можливий один крок для кожного BASE_PERIOD, що подвоює максимальну швидкість кроку.

Інші способи вирішення проблеми полягають у зменшенні одного або декількох з наступних параметрів: BASE_PERIOD (занадто низьке значення цього параметра призведе до того, що машина перестане реагувати на команди або навіть заблокується), INPUT_SCALE (якщо ви можете вибрати різні розміри кроку на вашому крововому драйвері, змініть передавальне число шківів або крок ходового гвинта) або MAX_VELOCITY і STEPGEN_MAXVEL.

Якщо жодна з дійсних комбінацій BASE_PERIOD, INPUT_SCALE та MAX_VELOCITY не є прийнятною, розгляньте можливість використання апаратного генерування кроків (наприклад, за допомогою універсального крокового контролера, що підтримується LinuxCNC, карт Mesa та інших).

4.8.3 Розпіновка

Однією з основних вад EMC було те, що ви не могли вказати розклад виводів без перекомпіляції вихідного коду. EMC2 був набагато гнучкішим, і тому тепер в LinuxCNC (завдяки шару абстракції апаратного забезпечення) ви можете легко вказати, який сигнал куди надходить. Дивіться HAL Basics для отримання додаткової інформації про HAL.

Як описано у вступі до HAL та навчальному посібнику, у нас є сигнали, контакти та параметри всередині HAL.

Note
Ми представляємо одну вісь, щоб не ускладнювати розгляд, всі інші схожі.

Для нашої розпіновки релевантні такі:

```
signals: Xstep, Xdir & Xen
pins: parport.0.pin-XX-out & parport.0.pin-XX-in
```

Залежно від того, що ви вибрали у файлі INI, ви використовуєте або standard_pinout.hal, або xylotex_pinout.hal. Це два файли, які вказують HAL, як пов’язувати різні сигнали та контакти. Далі ми розглянемо файл standard_pinout.hal.

4.8.3.1 Стандартна розпіновка HAL

Цей файл містить кілька команд HAL і зазвичай виглядає так:

```
# b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’дb’’b’’ab’’b’’pb’’b’’тb’’b’’нb’’b’’иb’’b’’йb’’ b’’фb’’b’ ↩
    ’ab’’b’’йb’’b’’лb’’ b’’кb’’b’’ob’’b’’нb’’b’’фb’’b’’ib’’b’’гb’’b’’уb’’b’’pb’’b’’ab’’b’ ↩
    ’цb’’b’’ib’’b’’їb’’ b’’pb’’b’’ob’’b’’зb’’b’’пb’’b’’ib’’b’’нb’’b’’ob’’b’’вb’’b’’кb’’b’ ↩
    ’иb’’ b’’дb’’b’’лb’’b’’яb’’ 3-b’’ob’’b’’cb’’b’’ьb’’b’’ob’’b’’вb’’b’’иb’’b’’хb’’ b’’кb’’b ↩
    ’’pb’’b’’ob’’b’’кb’’b’’ob’’b’’вb’’b’’иb’’b’’хb’’ b’’дb’’b’’вb’’b’’иb’’b’’гb’’b’’уb’’b’ ↩
    ’нb’’b’’ib’’b’’вb’’
# b’’вb’’b’’иb’’b’’кb’’b’’ob’’b’’pb’’b’’иb’’b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’нb’’b’’яb’’ b’ ↩
    ’пb’’b’’ab’’b’’pb’’b’’пb’’b’’ob’’b’’pb’’b’’тb’’b’’уb’’ b’’дb’’b’’лb’’b’’яb’’ b’’вb’’b’ ↩
    ’вb’’b’’ob’’b’’дb’’b’’уb’’/b’’вb’’b’’иb’’b’’вb’’b’’ob’’b’’дb’’b’’уb’’
#
# b’’cb’’b’’пb’’b’’ob’’b’’чb’’b’’ab’’b’’тb’’b’’кb’’b’’уb’’ b’’зb’’b’’ab’’b’’вb’’b’’ab’’b’ ↩
    ’нb’’b’’тb’’b’’ab’’b’’жb’’b’’тb’’b’’eb’’ b’’дb’’b’’pb’’b’’ab’’b’’йb’’b’’вb’’b’’eb’’b’ ↩
    ’pb’’ b’’пb’’b’’ab’’b’’pb’’b’’пb’’b’’ob’’b’’pb’’b’’тb’’b’’уb’’
loadrt hal_parport cfg=”0x0378”
#
# b’’дb’’b’’ab’’b’’лb’’b’’ib’’ b’’пb’’b’’ib’’b’’дb’’b’’кb’’b’’лb’’b’’юb’’b’’чb’’b’’иb’’b’ ↩
    ’тb’’b’’иb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’ib’’b’’їb’’ parport b’’дb’’b’’ob’’ b’ ↩
    ’пb’’b’’ob’’b’’тb’’b’’ob’’b’’кb’’b’’ib’’b’’вb’’
# b’’cb’’b’’пb’’b’’ob’’b’’чb’’b’’ab’’b’’тb’’b’’кb’’b’’уb’’ b’’зb’’b’’чb’’b’’иb’’b’’тb’’b’ ↩
    ’ab’’b’’тb’’b’’иb’’ b’’вb’’b’’xb’’b’’ob’’b’’дb’’b’’иb’’
addf parport.0.read base-thread 1
# b’’ob’’b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’нb’’b’’єb’’ b’’зb’’b’’ab’’b’’пb’’b’’иb’’b’’cb’’b’ ↩
    ’ab’’b’’тb’’b’’иb’’ b’’вb’’b’’иb’’b’’xb’’b’’ob’’b’’дb’’b’’иb’’
addf parport.0.write base-thread -1

# b’’нb’’b’’ab’’b’’pb’’b’’eb’’b’’шb’’b’’тb’’b’’ib’’ b’’пb’’b’’ib’’b’’дb’’b’’кb’’b’’лb’’b’ ↩
    ’юb’’b’’чb’’b’’иb’’b’’тb’’b’’иb’’ b’’фb’’b’’ib’’b’’зb’’b’’иb’’b’’чb’’b’’нb’’b’’ib’’ b’ ↩
    ’пb’’b’’ib’’b’’нb’’b’’иb’’ b’’дb’’b’’ob’’ th
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

# b’’cb’’b’’тb’’b’’вb’’b’’ob’’b’’pb’’b’’иb’’b’’тb’’b’’иb’’ b’’cb’’b’’иb’’b’’гb’’b’’нb’’b’ ↩
    ’ab’’b’’лb’’ b’’дb’’b’’лb’’b’’яb’’ b’’зb’’b’’вb’’b’’ob’’b’’pb’’b’’ob’’b’’тb’’b’’нb’’b’ ↩
    ’ob’’b’’гb’’b’’ob’’ b’’зb’’b’’вb’’’b’’яb’’b’’зb’’b’’кb’’b’’уb’’ estop
net estop-loop iocontrol.0.user-enable-out iocontrol.0.emc-enable-in

# b’’cb’’b’’тb’’b’’вb’’b’’ob’’b’’pb’’b’’иb’’b’’тb’’b’’иb’’ b’’cb’’b’’иb’’b’’гb’’b’’нb’’b’ ↩
    ’ab’’b’’лb’’b’’иb’’ b’’дb’’b’’лb’’b’’яb’’ b’’зb’’b’’вb’’b’’ob’’b’’pb’’b’’ob’’b’’тb’’b’ ↩
    ’нb’’b’’ob’’b’’гb’’b’’ob’’ b’’зb’’b’’вb’’’b’’яb’’b’’зb’’b’’кb’’b’’уb’’ b’’зb’’b’’ab’’b’ ↩
    ’вb’’b’’ab’’b’’нb’’b’’тb’’b’’ab’’b’’жb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ b’’ib’’b’’нb’’b’ ↩
    ’cb’’b’’тb’’b’’pb’’b’’уb’’b’’мb’’b’’eb’’b’’нb’’b’’тb’’b’’уb’’
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

```
# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''кb''b''ob''b' ↩
    'нb''b''тb''b''ab''b''кb''b''тb'' b''кb''b''ob''b''нb''b''тb''b''pb''b''ob''b''лb''b' ↩
    'eb''b''pb''b''ab'' b''pb''b''yb''b''xb''b''yb'' ”b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'eb''b''лb''b''ьb'' b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob''” b' ↩
    'дb''b''ob'' b''фb''b''ib''b''зb''b''иb''b''чb''b''нb''b''ob''b''гb''b''ob'' b''кb''b' ↩
    'ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''yb''
net spindle-on spindle.0.on => parport.0.pin-09-out

###
### b''Bb''b''иb'' b''мb''b''ob''b''жb''b''eb''b''тb''b''eb'' b''вb''b''иb''b''кb''b''ob''b ↩
    ''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''вb''b''ab''b''тb''b''иb'' b''щb''b' ↩
    'ob''b''cb''b''ьb'' b''пb''b''ob''b''дb''b''ib''b''бb''b''нb''b''eb'', b''щb''b''ob''b' ↩
    'бb'' b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''пb''b''eb''b' ↩
    'pb''b''eb''b''pb''b''иb''b''вb''b''чb''b''ab''b''cb''b''тb''b''ib'' b''пb''b''pb''b' ↩
    'иb''b''вb''b''ob''b''дb''b''иb'', b''кb''b''ob''b''лb''b''иb'' b''мb''b''ab''b''шb''b' ↩
    'иb''b''нb''b''ab'' b''вb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ab''
### b''cb''b''иb''b''гb''b''нb''b''ab''b''лb'' Xen b''вb''b''иb''b''зb''b''нb''b''ab''b' ↩
    'чb''b''eb''b''нb''b''ob'' b''вb'' core_stepper.hal
###

# net Xen => parport.0.pin-01-out

###
### b''Яb''b''кb''b''щb''b''ob'' b''вb''b''ab''b''мb'' b''пb''b''ob''b''тb''b''pb''b''ib''b ↩
    ''бb''b''eb''b''нb'' b''ab''b''кb''b''тb''b''иb''b''вb''b''нb''b''иb''b''йb'' b''нb''b' ↩
    'иb''b''зb''b''ьb''b''кb''b''иb''b''йb'' b''pb''b''ib''b''вb''b''eb''b''нb''b''ьb'' b' ↩
    'дb''b''лb''b''яb'' b''цb''b''ьb''b''ob''b''гb''b''ob'' b''кb''b''ob''b''нb''b''тb''b' ↩
    'ab''b''кb''b''тb''b''yb'', b''ib''b''нb''b''вb''b''eb''b''pb''b''тb''b''yb''b''йb''b' ↩
    'тb''b''eb'' b''йb''b''ob''b''гb''b''ob'' b''ob''b''cb''b''ьb'' b''тb''b''ab''b''кb'':
###

# setp parport.0.pin-01-out-invert 1

###
### b''3b''b''pb''b''ab''b''зb''b''ob''b''кb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b' ↩
    'кb''b''ab''b''чb''b''ab'' b''дb''b''ob''b''мb''b''yb'' b''нb''b''ab'' b''ob''b''cb''b' ↩
    'ib'' X (b''вb''b''ib''b''cb''b''ьb'' 0). b''Пb''b''ob''b''дb''b''ab''b''йb''b''тb''b' ↩
    'eb'' b''cb''b''иb''b''гb''b''нb''b''ab''b''лb'',
### b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''вb''b''xb''b' ↩
    'ib''b''дb''b''нb''b''иb''b''йb'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb'' b' ↩
    'пb''b''ob''b''pb''b''тb''b''yb'' b''пb''b''ob''b''pb''b''тb''b''yb'' b''дb''b''ob'' b' ↩
    'cb''b''иb''b''гb''b''нb''b''ab''b''лb''b''yb'', b''ab'' b''пb''b''ob''b''тb''b''ib''b' ↩
    'мb'' b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''cb''b' ↩
    'иb''b''гb''b''нb''b''ab''b''лb''
### b''дb''b''ob'' b''вb''b''xb''b''ib''b''дb''b''нb''b''ob''b''гb''b''ob'' b''кb''b''ob''b ↩
    ''нb''b''тb''b''ab''b''кb''b''тb''b''yb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b' ↩
    'кb''b''ab''b''чb''b''ab'' b''дb''b''ob''b''мb''b''yb'' b''ob''b''cb''b''ib'' 0 LinuxCNC ↩
    .
###

# net Xhome parport.0.pin-10-in => joint.0.home-sw-in

###
### b''Cb''b''пb''b''ib''b''лb''b''ьb''b''нb''b''ib'' b''дb''b''ob''b''мb''b''ab''b''шb''b' ↩
    'нb''b''ib'' b''кb''b''ob''b''мb''b''yb''b''тb''b''ab''b''тb''b''ob''b''pb''b''иb'' b' ↩
    'нb''b''ab'' b''ob''b''дb''b''нb''b''ob''b''мb''b''yb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'лb''b''eb''b''лb''b''ьb''b''нb''b''ob''b''мb''b''yb'' b''пb''b''ob''b''pb''b''тb''b' ↩
    'yb''?
### b''цb''b''eb'' b''нb''b''ob''b''pb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ob'', b''пb'' ↩
    b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''ob''b''дb''b''нb''b' ↩
    'ab''b''кb''b''ob''b''вb''b''иb''b''йb'' b''cb''b''иb''b''гb''b''нb''b''ab''b''лb'' b' ↩
    'дb''b''ob'' b''вb''b''cb''b''ib''b''xb'' b''ob''b''cb''b''eb''b''йb'', b''ab''b''лb''b' ↩
```

```
    ’еb’’ b’’ob’’b’’бb’’b’’ob’’b’’вb’’’b’’яb’’b’’зb’’b’’кb’’b’’ob’’b’’вb’’b’’ob’’
### b’’вb’’b’’cb’’b’’тb’’b’’аb’’b’’нb’’b’’ob’’b’’вb’’b’’іb’’b’’тb’’b’’ьb’’ HOME_IS_SHARED b ↩
    ’’тb’’b’’аb’’ HOME_SEQUENCE b’’уb’’ b’’фb’’b’’аb’’b’’йb’’b’’лb’’b’’іb’’ INI.
###

# b’’мb’’b’’еb’’b’’рb’’b’’еb’’b’’жb’’b’’еb’’b’’вb’’b’’іb’’ b’’дb’’b’’ob’’b’’мb’’b’’аb’’b’ ↩
    ’шb’’b’’нb’’b’’іb’’ b’’кb’’b’’ob’’b’’мb’’b’’уb’’b’’тb’’b’’аb’’b’’тb’’b’’ob’’b’’рb’’b’ ↩
    ’иb’’ <= parport.0.pin-10-in
# b’’мb’’b’’еb’’b’’рb’’b’’еb’’b’’жb’’b’’еb’’b’’вb’’b’’іb’’ b’’дb’’b’’ob’’b’’мb’’b’’аb’’b’ ↩
    ’шb’’b’’нb’’b’’іb’’ b’’кb’’b’’ob’’b’’мb’’b’’уb’’b’’тb’’b’’аb’’b’’тb’’b’’ob’’b’’рb’’b’ ↩
    ’иb’’ => joint.0.home-sw-in
# b’’мb’’b’’еb’’b’’рb’’b’’еb’’b’’жb’’b’’еb’’b’’вb’’b’’іb’’ b’’дb’’b’’ob’’b’’мb’’b’’аb’’b’ ↩
    ’шb’’b’’нb’’b’’іb’’ b’’кb’’b’’ob’’b’’мb’’b’’уb’’b’’тb’’b’’аb’’b’’тb’’b’’ob’’b’’рb’’b’ ↩
    ’иb’’ => joint.1.home-sw-in
# b’’мb’’b’’еb’’b’’рb’’b’’еb’’b’’жb’’b’’еb’’b’’вb’’b’’іb’’ b’’дb’’b’’ob’’b’’мb’’b’’аb’’b’ ↩
    ’шb’’b’’нb’’b’’іb’’ b’’кb’’b’’ob’’b’’мb’’b’’уb’’b’’тb’’b’’аb’’b’’тb’’b’’ob’’b’’рb’’b’ ↩
    ’иb’’ => joint.2.home-sw-in

###
### b’’Зb’’b’’рb’’b’’аb’’b’’зb’’b’’ob’’b’’кb’’ b’’ob’’b’’кb’’b’’рb’’b’’еb’’b’’мb’’b’’иb’’b’ ↩
    ’хb’’ b’’кb’’b’’іb’’b’’нb’’b’’цb’’b’’еb’’b’’вb’’b’’иb’’b’’хb’’ b’’вb’’b’’иb’’b’’мb’’b’ ↩
    ’иb’’b’’кb’’b’’аb’’b’’чb’’b’’іb’’b’’вb’’ b’’нb’’b’’аb’’ b’’ob’’b’’cb’’b’’іb’’ X (b’’вb’’ ↩
    b’’іb’’b’’cb’’b’’ьb’’ 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

###
### b’’Tb’’b’’аb’’b’’кb’’ b’’cb’’b’’аb’’b’’мb’’b’’ob’’, b’’яb’’b’’кb’’ b’’уb’’ b’’пb’’b’ ↩
    ’рb’’b’’иb’’b’’кb’’b’’лb’’b’’аb’’b’’дb’’b’’іb’’ b’’зb’’b’’іb’’ b’’cb’’b’’пb’’b’’іb’’b’ ↩
    ’лb’’b’’ьb’’b’’нb’’b’’иb’’b’’мb’’b’’иb’’ b’’дb’’b’’ob’’b’’мb’’b’’аb’’b’’шb’’b’’нb’’b’ ↩
    ’іb’’b’’мb’’b’’иb’’ b’’вb’’b’’иb’’b’’мb’’b’’иb’’b’’кb’’b’’аb’’b’’чb’’b’’аb’’b’’мb’’b’ ↩
    ’иb’’, b’’вb’’b’’иb’’ b’’мb’’b’’ob’’b’’жb’’b’’еb’’b’’тb’’b’’еb’’ b’’зb’’’b’’єb’’b’’дb’’b ↩
    ’’нb’’b’’аb’’b’’тb’’b’’иb’’ b’’мb’’b’’іb’’b’’жb’’ b’’cb’’b’’ob’’b’’бb’’b’’ob’’b’’юb’’
### b’’кb’’b’’іb’’b’’нb’’b’’цb’’b’’еb’’b’’вb’’b’’іb’’ b’’вb’’b’’иb’’b’’мb’’b’’иb’’b’’кb’’b’ ↩
    ’аb’’b’’чb’’b’’іb’’.  b’’Бb’’b’’уb’’b’’дb’’b’’ьb’’b’’тb’’b’’еb’’ b’’ob’’b’’бb’’b’’еb’’b’ ↩
    ’рb’’b’’еb’’b’’жb’’b’’нb’’b’’іb’’, b’’яb’’b’’кb’’b’’щb’’b’’ob’’ b’’вb’’b’’иb’’ b’’нb’’b’ ↩
    ’аb’’b’’тb’’b’’рb’’b’’аb’’b’’пb’’b’’иb’’b’’тb’’b’’еb’’ b’’нb’’b’’аb’’ b’’ob’’b’’дb’’b’ ↩
    ’иb’’b’’нb’’ b’’зb’’ b’’нb’’b’’иb’’b’’хb’’, LinuxCNC b’’зb’’b’’уb’’b’’пb’’b’’иb’’b’’нb’’ ↩
    b’’иb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’, b’’аb’’b’’лb’’b’’еb’’ b’’нb’’b’’еb’’ b’’зb’’b’’мb’’ ↩
    b’’ob’’b’’жb’’b’’еb’’ b’’пb’’b’’ob’’b’’вb’’b’’іb’’b’’дb’’b’’ob’’b’’мb’’b’’иb’’b’’тb’’b’ ↩
    ’иb’’
### b’’вb’’b’’аb’’b’’мb’’, b’’яb’’b’’кb’’b’’иb’’b’’йb’’ b’’вb’’b’’иb’’b’’мb’’b’’иb’’b’’кb’’ ↩
    b’’аb’’b’’чb’’/b’’вb’’b’’іb’’b’’cb’’b’’ьb’’ b’’вb’’b’’иb’’b’’йb’’b’’шb’’b’’ob’’b’’вb’’ b ↩
    ’’зb’’ b’’лb’’b’’аb’’b’’дb’’b’’уb’’.  b’’Бb’’b’’уb’’b’’дb’’b’’ьb’’b’’тb’’b’’еb’’ b’’ob’’ ↩
    b’’бb’’b’’еb’’b’’рb’’b’’еb’’b’’жb’’b’’нb’’b’’іb’’, b’’вb’’b’’іb’’b’’дb’’b’’нb’’b’’ob’’b’ ↩
    ’вb’’b’’лb’’b’’юb’’b’’юb’’b’’чb’’b’’иb’’ b’’рb’’b’’ob’’b’’бb’’b’’ob’’b’’тb’’b’’уb’’ b’ ↩
    ’зb’’ b’’цb’’b’’ьb’’b’’ob’’b’’гb’’b’’ob’’
### b’’кb’’b’’рb’’b’’аb’’b’’йb’’b’’нb’’b’’ьb’’b’’ob’’b’’гb’’b’’ob’’ b’’пb’’b’’ob’’b’’лb’’b’ ↩
    ’ob’’b’’жb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’, b’’щb’’b’’ob’’b’’бb’’ b’’уb’’b’’нb’’b’’иb’’b’ ↩
    ’кb’’b’’нb’’b’’уb’’b’’тb’’b’’иb’’ b’’рb’’b’’іb’’b’’зb’’b’’кb’’b’’ob’’b’’їb’’ b’’зb’’b’ ↩
    ’уb’’b’’пb’’b’’иb’’b’’нb’’b’’кb’’b’’иb’’.
###

# net Xlimits parport.0.pin-13-in => joint.0.neg-lim-sw-in joint.0.pos-lim-sw-in
```

Рядки, що починаються з символу #, є коментарями, і їх єдине призначення — допомогти читачеві ознайомитися з файлом.

4.8.3.2 Огляд

Існує кілька операцій, які виконуються під час виконання/інтерпретації standard_pinout.hal:

- Драйвер Parport завантажується (див. розділ Parport для отримання детальнішої інформації).
- Функції читання та запису драйвера parport призначаються базовому потоку (примітка: [Найшвидш потік у системі LinuxCNC, зазвичай код виконується кожні кілька десятків мікросекунд.]).
- Сигнали кроку та напрямку для осей X, Y, Z підключаються до контактів на парпорті.
- Підключаються додаткові сигнали вводу/виводу (петля виходу з ладу, петля пристрою зміни інструментів).
- Сигнал увімкнення шпинделя визначається та пов’язується з контактом парпорту.

4.8.3.3 Зміна standard_pinout.hal

Якщо ви хочете змінити файл standard_pinout.hal, вам потрібен лише текстовий редактор. Відкрийте файл і знайдіть частини, які ви хочете змінити.

Якщо ви хочете, наприклад, змінити контакт для сигналів кроків та напрямків осі X, все, що вам потрібно зробити, це змінити номер в назві *parport.0.pin-XX-out*:

```
net Xstep parport.0.pin-03-out
net Xdir  parport.0.pin-02-out
```

можна змінити на:

```
net Xstep parport.0.pin-02-out
net Xdir  parport.0.pin-03-out
```

або, в принципі, будь-який інший «вихідний» штифт, який вам подобається.

Підказка: переконайтеся, що до одного контакту підключено не більше одного сигналу.

4.8.3.4 Зміна полярності сигналу

Якщо зовнішнє обладнання очікує сигнал ”активного низького рівня”, встановіть відповідний параметр *-invert*. Наприклад, щоб інвертувати сигнал керування шпинделем:

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

4.8.3.5 Додавання PWM-регулювання швидкості шпинделя

Якщо ваш шпиндель можна керувати за допомогою PWM-сигналу, використовуйте компонент *pwmgen* для створення сигналу:

```
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 # b''3b''b''мb''b''ib''b''нb''b''ib''b''тb''b''ьb'' b''мb''b''ab'' ↩
    b''кb''b''cb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''уb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b' ↩
    'лb''b''яb'' b''вb'' b''оb''b''бb''b''eb''b''pb''b''тb''b''ab''b''xb'' b''зb''b''ab'' b' ↩
    'xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb''
```

Це передбачає, що реакція контролера шпинделя на PWM є простою: 0% PWM дає 0 об/хв, 10% PWM дає 180 об/хв тощо. Якщо для обертання шпинделя необхідний мінімальний PWM, дотримуйтесь прикладу в зразковій конфігурації «nist-lathe», щоб використовувати компонент «scale».

4.8.3.6 Додавання сигналу ввімкнення

Деякі підсилювачі (приводи) потребують сигналу ввімкнення, перш ніж приймати та керувати рухом двигунів. З цієї причини вже існують визначені сигнали під назвою «Xen», «Yen», «Zen».

Щоб їх з'єднати, скористайтеся наступним прикладом:

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

Ви можете мати один контакт, який вмикає всі приводи, або кілька, залежно від вашої конфігурації. Однак зверніть увагу, що зазвичай, коли одна вісь виходить з ладу, всі інші приводи також вимикаються, тому наявність лише одного сигналу/контакту вмикання для всіх приводів є загальноприйнятою практикою.

4.8.3.7 Зовнішня кнопка ESTOP

Файл standard_pinout.hal не передбачає зовнішньої кнопки аварійної зупинки. Для отримання додаткової інформації про зовнішню кнопку аварійної зупинки див. сторінку довідки estop_latch.

4.9 Діагностика крокових двигунів

Якщо те, що ви отримуєте, не відповідає вашим очікуванням, часто ви просто отримуєте певний досвід. Навчання на досвіді покращує ваше розуміння цілого. Діагностувати проблеми найкраще методом «розділяй і володарюй». Під цим я маю на увазі, що якщо ви зможете щоразу видаляти половину змінних з рівняння, ви найшвидше знайдете проблему. У реальному світі це не завжди так, але зазвичай це хороший початок.

4.9.1 Поширені проблеми

4.9.1.1 Кроковий рух на один крок

Найпоширенішою причиною того, що кроковий двигун не рухається в новій установці, є те, що сигнали кроку та напрямку помінялися місцями. Якщо ви по черзі натискаєте клавіші «вперед» та «назад», а кроковий двигун рухається на один крок щоразу в одному напрямку, це є підказкою.

4.9.1.2 Без руху степерів

Багато накопичувачів мають контакт увімкнення або потребують насоса заряду для активації виходу.

4.9.1.3 Відстань неправильна

Якщо ви наказуєте осі переміститися на певну відстань, а вона не переміщується на цю відстань, то налаштування масштабу неправильне.

4.9.2 Повідомлення про помилки

4.9.2.1 Помилка після

Поняття помилки слідування є дивним, коли мова йде про крокові двигуни. Оскільки вони є системою з відкритим контуром, немає зворотного зв'язку щодо положення, який би повідомляв, чи ви насправді вийшли за межі діапазону. LinuxCNC обчислює, чи може він встигати за необхідним рухом, і якщо ні, то видає помилку слідування. Помилки слідування зазвичай є результатом одного з наступних факторів у крокових системах.

- FERROR занадто малий
- MIN_FERROR занадто мале значення
- MAX_VELOCITY занадто швидко
- MAX_ACCELERATION занадто швидко
- BASE_PERIOD встановлено занадто довго
- До осі додано люфт

Будь-яка з вищезазначених причин може призвести до того, що імпульси в режимі реального часу не зможуть підтримувати необхідну частоту кроків. Це може статися, якщо ви не провели тест на затримку достатньо довго, щоб отримати точні дані для введення в майстер StepConf, або якщо ви встановили занадто високі значення максимальної швидкості або максимального прискорення.

Якщо ви додали люфт, вам потрібно збільшити STEPGEN_MAXACCEL до подвійного значення MAX_ACCELERATION у розділі AXIS файлу INI для кожної осі, до якої ви додали люфт. LinuxCNC використовує «додаткове прискорення» при зміні напрямку руху, щоб компенсувати люфт. Без корекції люфту прискорення генератора кроків може бути лише на кілька відсотків вище за прискорення планувальника руху.

4.9.2.2 Помилка RTAPI

Коли ви отримуєте цю помилку:

```
RTAPI: b''Пb''b''Ob''b''Mb''b''Иb''b''Лb''b''Kb''b''Ab'': b''Hb''b''eb''b''ob''b''чb''b' ↩
    'ib''b''кb''b''yb''b''вb''b''ab''b''нb''b''ab'' b''зb''b''ab''b''тb''b''pb''b''иb''b' ↩
    'мb''b''кb''b''ab'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b' ↩
    'yb'' b''чb''b''ab''b''cb''b''ib'' b''дb''b''лb''b''яb'' b''зb''b''ab''b''вb''b''дb''b' ↩
    'ab''b''нb''b''нb''b''яb'' n
```

Ця помилка генерується rtapi на основі повідомлення від RTAI про прострочення терміну. Зазвичай це означає, що значення BASE_PERIOD у розділі [EMCMOT] файлу ini встановлено занадто низьким. Ви повинні запустити тест затримки на тривалий період часу, щоб перевірити, чи є затримки, які можуть спричинити цю проблему. Якщо ви використовували майстер StepConf, запустіть його знову, повторіть тест Base Period Jitter і відрегулюйте Base Period Maximum Jitter на сторінці Basic Machine Information. Можливо, вам доведеться залишити тест працювати протягом тривалого періоду часу, щоб з'ясувати, чи якесь обладнання спричиняє періодичні проблеми.

LinuxCNC відстежує кількість циклів процесора між викликами потоку реального часу. Якщо якийсь елемент вашого обладнання спричиняє затримки або ваші потоки реального часу налаштовані занадто швидко, ви отримаєте цю помилку.

Note
Ця помилка відображається лише один раз за сеанс. Якщо ваш BASE_PERIOD був занадто низьким, ви можете отримувати сотні тисяч повідомлень про помилки на секунду, якщо відображається більше одного.

4.9.3 Тестування

4.9.3.1 Час кроку

Якщо ви бачите, що вісь опиняється в неправильному місці після декількох рухів, ймовірно, що ви не маєте правильних значень часу утримання напрямку або часу кроку для ваших крокових драйверів. Кожна зміна напрямку може призводити до втрати одного або декількох кроків. Якщо двигуни зупиняються, можливо, ви встановили занадто високі значення MAX_ACCELERATION або MAX_VELOCITY для цієї осі.

Наступна програма перевірить конфігурацію осі Z на правильність налаштування. Скопіюйте програму в каталог \~/emc2/nc_files і назвіть її TestZ.ngc або подібним чином. Обнуліть верстат із Z = 0,000 на верхній частині столу. Завантажте та запустіть програму. Вона виконає 200 рухів вперед і назад від 0,5 до 1 дюйма. Якщо у вас є проблема з конфігурацією, ви побачите, що кінцеве положення не буде 0,500 дюйма, як показує вікно осі. Щоб перевірити іншу вісь, просто замініть Z на вашу вісь у рядках G0.

```
(b''тb''b''eb''b''cb''b''тb''b''ob''b''вb''b''ab'' b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''ab'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b' ↩
    'pb''b''кb''b''иb'' b''вb''b''тb''b''pb''b''ab''b''тb''b''иb'' b''пb''b''ob''b''зb''b' ↩
    'иb''b''цb''b''ib''b''їb'' b''ob''b''cb''b''ib'' Z)
(msg, b''тb''b''eb''b''cb''b''тb'' 1 b''кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''уb''b' ↩
    'pb''b''ab''b''цb''b''ib''b''їb'' b''ob''b''cb''b''ib'' Z)
G20 #1000=100 (b''цb''b''иb''b''кb''b''лb'' 100 b''pb''b''ab''b''зb''b''ib''b''вb'')
(b''цb''b''eb''b''йb'' b''цb''b''иb''b''кb''b''лb'' b''мb''b''ab''b''єb'' b''зb''b''ab''b' ↩
    'тb''b''pb''b''иb''b''мb''b''кb''b''иb'' b''пb''b''ib''b''cb''b''лb''b''яb'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b''ьb'')
(b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b''яb''b''єb'' b''нb''b''ab''b''лb''b' ↩
    'ab''b''шb''b''тb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''пb''b''pb''b''иb''b' ↩
    'cb''b''кb''b''ob''b''pb''b''eb''b''нb''b''нb''b''яb'' b''тb''b''ab'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''ob''b''cb''b''тb''b''ib'')
o100 while [#1000]
   G0 Z1.000
   G4 P0.250
   G0 Z0.500
   G4 P0.250
   #1000 = [#1000 - 1]
o100 endwhile
(b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'тb''b''eb''b''cb''b''тb'' 2 b''кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''уb''b''pb''b' ↩
    'ab''b''цb''b''ib''b''їb'' b''ob''b''cb''b''ib'' Z S b''дb''b''лb''b''яb'' b''пb''b' ↩
    'pb''b''ob''b''дb''b''ob''b''вb''b''жb''b''eb''b''нb''b''нb''b''яb'')
M1 (b''зb''b''уb''b''пb''b''иb''b''нb''b''кb''b''ab'' b''тb''b''уb''b''тb'')
#1000=100 (b''цb''b''иb''b''кb''b''лb'' 100 b''pb''b''ab''b''зb''b''ib''b''вb'')
(b''нb''b''ab''b''cb''b''тb''b''уb''b''пb''b''нb''b''иb''b''йb'' b''цb''b''иb''b''кb''b' ↩
    'лb'' b''нb''b''eb'' b''мb''b''ab''b''єb'' b''зb''b''ab''b''тb''b''pb''b''иb''b''мb''b' ↩
    'ob''b''кb'' b''пb''b''ib''b''cb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'ib''b''щb''b''eb''b''нb''b''ьb'')
(b''тb''b''eb''b''cb''b''тb''b''уb''b''єb'' b''чb''b''ab''b''cb'' b''уb''b''тb''b''pb''b' ↩
    'иb''b''мb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab''b''пb''b''pb''b''яb''b''мb''b' ↩
    'кb''b''уb'' b''нb''b''ab'' b''кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''уb''b''pb''b' ↩
    'ab''b''цb''b''ib''b''їb'' b''дb''b''pb''b''ab''b''йb''b''вb''b''eb''b''pb''b''ab'', b' ↩
    'ab'' b''тb''b''ab''b''кb''b''ob''b''жb'' b''мb''b''ab''b''кb''b''cb''b''иb''b''мb''b' ↩
    'ab''b''лb''b''ьb''b''нb''b''eb'' b''пb''b''pb''b''иb''b''cb''b''кb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''нb''b''яb'')
o101 while [#1000]
   G0 Z1.000
   G0 Z0.500
   #1000 = [#1000 - 1]
o101 endwhile
(msg, b''Гb''b''ob''b''тb''b''ob''b''вb''b''ob''...Z b''пb''b''ob''b''вb''b''иb''b''нb''b' ↩
```

```
    'eb''b''нb'' b''бb''b''уb''b''тb''b''иb'' b''тb''b''оb''b''чb''b''нb''b''оb'' b''нb''b' ↩
    'ab'' 0,5 b''дb''b''юb''b''йb''b''мb''b''ab'' b''нb''b''ab''b''дb'' b''cb''b''тb''b' ↩
    'оb''b''лb''b''оb''b''мb'')
M2
```

4.10 Фільтрувати програми

4.10.1 Вступ

Більшість екранів LinuxCNC мають можливість надсилати завантажені файли через «програму фільтрації» або використовувати програму фільтрації для створення G-коду. Такий фільтр може виконувати будь-яке бажане завдання: від такого простого, як перевірка, чи файл закінчується на «M2», до такого складного, як генерація G-коду з зображення.

4.10.2 Налаштування INI для програмних фільтрів

Розділ «[FILTER]» файлу INI контролює роботу фільтрів. Спочатку для кожного типу файлу напишіть рядок «PROGRAM_EXTENSION». Потім вкажіть програму, яку потрібно виконати для кожного типу файлу. Ця програма отримує ім’я вхідного файлу як перший аргумент і повинна записувати код rs274ngc у стандартний вивід. Цей вивід буде відображатися в текстовій області, попередньо переглядатися в області відображення та виконуватися LinuxCNC при натисканні «Run». Наступні рядки додають підтримку конвертера «image-to-gcode», що входить до складу LinuxCNC:

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

Також можна вказати інтерпретатора:

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

Таким чином, можна відкрити будь-який скрипт Python, а його вихідні дані будуть оброблятися як G-код. Один із таких прикладів скрипту доступний за адресою «nc_files/holecircle.py». Цей скрипт створює G-код для свердління низки отворів по колу.

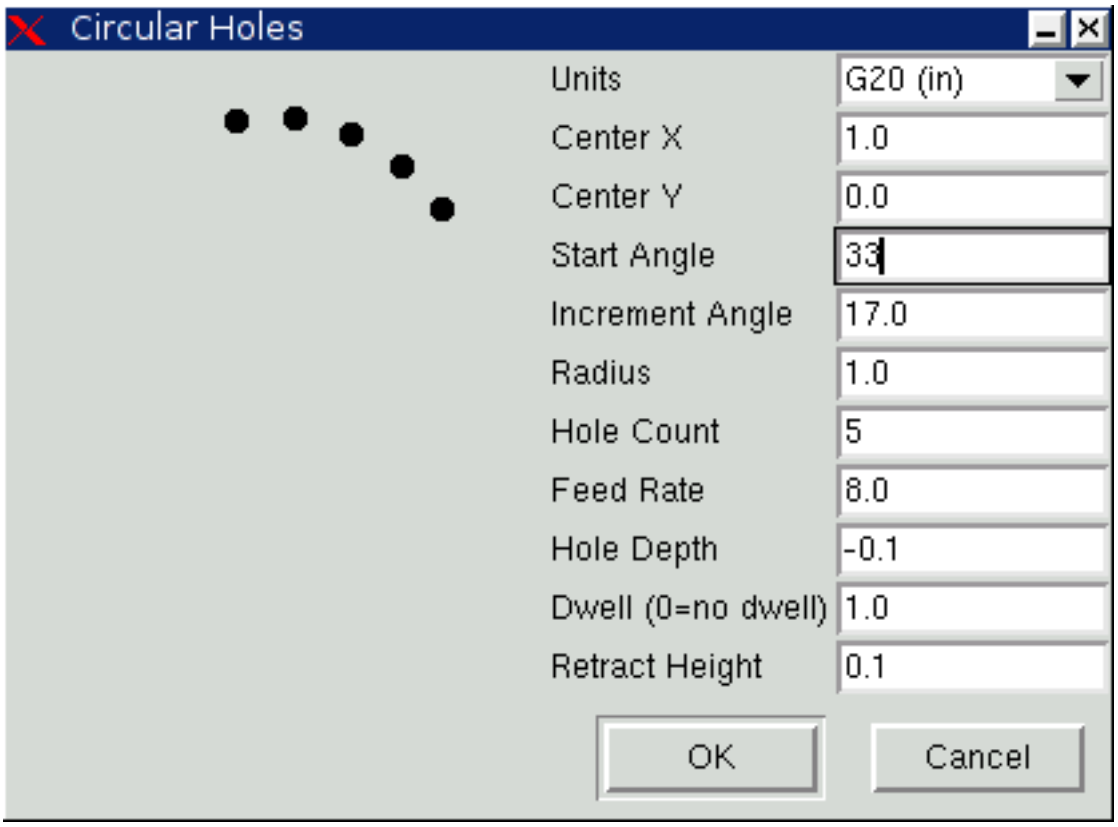


Figure 4.9: Круглі отвори

Якщо програма-фільтр надсилає рядки до stderr такого вигляду:

```
FILTER_PROGRESS=10
```

Це встановить індикатор виконання на екрані на задане значення (10 у цьому випадку) у відсотках. Цю функцію слід використовувати будь-якому фільтру, який працює протягом тривалого часу.

4.10.3 Створення програм фільтрації на базі Python

Ось дуже простий приклад механізму фільтрування: при запуску через екран Linucnc, який пропонує фільтрування програм, він буде генерувати і записувати рядок G-коду кожні 100^-ті секунди в стандартний вивід. Він також надсилає повідомлення про хід виконання в стандартний потік помилок UNIX. У разі виникнення помилки він надсилає повідомлення про помилку і завершує роботу з кодом виходу 1.

```
import time
import sys

for i in range(0,100):
    try:
        # b''ib''b''мb''b''ib''b''тb''b''yb''b''вb''b''ab''b''тb''b''иb'' b''чb''b''ab''b' ↩
            'cb'' b''ob''b''бb''b''чb''b''иb''b''cb''b''лb''b''eb''b''нb''b''нb''b''яb''
        time.sleep(.1)

        # b''вb''b''иb''b''вb''b''eb''b''cb''b''тb''b''иb'' b''pb''b''яb''b''дb''b''ob''b' ↩
            'кb'' G-b''кb''b''ob''b''дb''b''yb''
        print('G0 X1', file=sys.stdout)
```

```
        # b''пb''b''pb''b''ob''b''гb''b''pb''b''eb''b''cb'' b''ob''b''нb''b''ob''b''вb''b' ↩
            'лb''b''eb''b''нb''b''нb''b''яb''
        print('FILTER_PROGRESS={}'.format(i), file=sys.stderr)
    except:
        # b''Цb''b''eb'' b''пb''b''pb''b''иb''b''зb''b''вb''b''ob''b''дb''b''иb''b''тb''b' ↩
            'ьb'' b''дb''b''ob'' b''пb''b''ob''b''яb''b''вb''b''иb'' b''пb''b''ob''b''вb''b' ↩
            'ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''pb''b' ↩
            'ob'' b''пb''b''ob''b''мb''b''иb''b''лb''b''кb''b''уb''
        print('Error; But this was only a test', file=sys.stderr)
        raise SystemExit(1)
```

Ось подібна програма, але вона насправді може фільтрувати. Вона відкриває діалогове вікно PyQt5 з кнопкою скасування. Потім вона читає програму по рядках і передає її до стандартного виводу. У процесі роботи вона оновлює будь-який процес, що слухає стандартний вивід помилок.

```
#!/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()

        # b''ob''b''6b''b''чb''b''иb''b''cb''b''лb''b''иb''b''тb''b''иb'' b''вb''b''ib''b' ↩
            'дb''b''cb''b''ob''b''тb''b''кb''b''ob''b''вb''b''иb''b''йb'' b''ib''b''нb''b' ↩
            'тb''b''eb''b''pb''b''вb''b''ab''b''лb'' b''ob''b''нb''b''ob''b''вb''b''лb''b' ↩
            'eb''b''нb''b''нb''b''яb''
        self.bump = 100/float(len(self.temp))

        self._timer = QTimer()
        self._timer.timeout.connect(self.process)
        self._timer.start(100)
```

```
    def reject(self):
        # b''Цb''b''eb'' b''вb''b''иb''b''дb''b''аb''b''єb'' b''пb''b''оb''b''вb''b''іb''b' ↩
            'дb''b''оb''b''мb''b''лb''b''еb''b''нb''b''нb''b''яb'' b''пb''b''рb''b''оb'' b' ↩
            'пb''b''оb''b''мb''b''иb''b''лb''b''кb''b''уb''
        print('You asked to cancel before finished.', file=sys.stderr)
        raise SystemExit(1)

    def process(self):
        try:
            # b''оb''b''тb''b''рb''b''иb''b''мb''b''аb''b''тb''b''иb'' b''нb''b''аb''b' ↩
                'сb''b''тb''b''уb''b''пb''b''нb''b''иb''b''йb'' b''рb''b''яb''b''дb''b''оb'' ↩
                b''кb'' b''кb''b''оb''b''дb''b''уb''
            codeLine = self.temp[self.line]

            # b''оb''b''бb''b''рb''b''оb''b''бb''b''иb''b''тb''b''иb'' b''рb''b''яb''b' ↩
                'дb''b''оb''b''кb'' b''яb''b''кb''b''оb''b''сb''b''ьb''

            # b''вb''b''иb''b''шb''b''тb''b''оb''b''вb''b''хb''b''нb''b''уb''b''тb''b''иb'' ↩
                 b''оb''b''бb''b''рb''b''оb''b''бb''b''лb''b''еb''b''нb''b''иb''b''йb'' b' ↩
                'кb''b''оb''b''дb''
            print(codeLine, file=sys.stdout)
            self.line +=1

            # b''пb''b''рb''b''оb''b''гb''b''рb''b''еb''b''сb'' b''оb''b''нb''b''оb''b' ↩
                'вb''b''лb''b''еb''b''нb''b''нb''b''яb''
            self._percentDone += self.bump
            print('FILTER_PROGRESS={}'.format(int(self._percentDone)), file=sys.stderr)

            # b''яb''b''кb''b''щb''b''оb'' b''вb''b''иb''b''кb''b''оb''b''нb''b''аb''b' ↩
                'нb''b''оb'', b''зb''b''аb''b''вb''b''еb''b''рb''b''шb''b''уb''b''єb''b' ↩
                'тb''b''ьb''b''сb''b''яb'' b''бb''b''еb''b''зb'' b''пb''b''оb''b''мb''b' ↩
                'иb''b''лb''b''кb''b''иb''/b''пb''b''оb''b''вb''b''іb''b''дb''b''оb''b''мb'' ↩
                b''лb''b''еb''b''нb''b''нb''b''яb'' b''пb''b''рb''b''оb'' b''пb''b''оb''b' ↩
                'мb''b''иb''b''лb''b''кb''b''уb''
            if self._percentDone >= 99:
                print('FILTER_PROGRESS=-1', file=sys.stderr)
                self.infile.close()
                raise SystemExit(0)

        except Exception as e:
            # b''Цb''b''eb'' b''вb''b''иb''b''дb''b''аb''b''єb'' b''пb''b''оb''b''вb''b' ↩
                'іb''b''дb''b''оb''b''мb''b''лb''b''еb''b''нb''b''нb''b''яb'' b''пb''b''рb'' ↩
                b''оb'' b''пb''b''оb''b''мb''b''иb''b''лb''b''кb''b''уb''
            print(('Something bad happened:',e), file=sys.stderr)
            # b''цb''b''eb'' b''сb''b''иb''b''гb''b''нb''b''аb''b''лb''b''іb''b''зb''b' ↩
                'уb''b''єb'' b''пb''b''рb''b''оb'' b''тb''b''eb'', b''щb''b''оb'' b''мb''b' ↩
                'аb''b''єb'' b''вb''b''іb''b''дb''b''оb''b''бb''b''рb''b''аb''b''жb''b''аb'' ↩
                b''тb''b''иb''b''сb''b''яb'' b''пb''b''оb''b''вb''b''іb''b''дb''b''оb''b' ↩
                'мb''b''лb''b''еb''b''нb''b''нb''b''яb'' b''пb''b''рb''b''оb'' b''пb''b' ↩
                'оb''b''мb''b''иb''b''лb''b''кb''b''уb''
            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 (Рівень абстракції апаратного забезпечення)

5.1 Вступ до HAL

LinuxCNC призначений для взаємодії з апаратним забезпеченням. Але мало хто з користувачів має апаратне забезпечення з однаковими характеристиками — воно може бути схожим, але не однаковим. І навіть для абсолютно однакового апаратного забезпечення можуть існувати різні способи його використання, наприклад, для різних матеріалів або з різними фрезами, що вимагатиме адаптації управління вже працюючою системою. Була потрібна абстракція, щоб спростити налаштування LinuxCNC для широкого спектру апаратних пристроїв. На найвищому рівні це може бути просто спосіб, що дозволяє завантажувати та з'єднувати між собою низку «будівельних блоків» для складання складної системи.

У цьому розділі представлено рівень абстракції апаратного забезпечення. Ви побачите, що багато з його складових є драйверами для апаратних пристроїв. Однак HAL може не тільки налаштовувати драйвери апаратного забезпечення.

5.1.1 Огляд HAL

Рівень абстракції обладнання (або з посиланням на link:https://en.wikipedia.org/wiki/2001:_A_Space_Ody
одіссея 2001 року» (просто «HAL») – це програмне забезпечення для

- забезпечують інфраструктуру для зв'язку з багатьма програмними та апаратними компонентами системи та між ними.
- за бажанням обробляти та/або перевизначати цю інформацію під час її потоку від компонента до компонента.

Сам по собі цей Middleware не має відношення до його застосування на CNC. Наприклад, пошук в Інтернеті виявив астрономічне застосування для управління телескопами за допомогою Linux-CNC. Двигуни переміщують телескоп у потрібне положення, і він повинен знати, як співвіднести активність двигуна з ефектом цього позиціонування в реальному світі. Така синхронізація положень двигунів з положеннями в реальному світі нагадує те, що повинні робити верстати з CNC або космічні апарати.

Будь-який контролер машини повинен знати:

- про його внутрішній стан та як це відображається на навколишнє середовище (координати машини, стан перемикачів/регуляторів),

- як очікується, що виконавчі механізми змінять цей стан,
- як забезпечити оновлення внутрішнього стану за допомогою датчиків (енкодерів, зондів).

Шар HAL складається з частин (які називаються «компонентами»), які

- пов’язані один з одним, наприклад, для оновлення даних про положення або для того, щоб алгоритм планування повідомляв двигунам про наступний крок.
- може знати, як спілкуватися з обладнанням,
- може просто обробляти вхідні дані та надавати вихідні дані іншим компонентам,
- завжди періодично виконуються або
  - з дуже високою частотою виконання в кілька мікросекунд (мкс), що називається базовим потоком, наприклад, до
    1. дати крокovому двигуну тригер для кроку вперед або
    2. зчитати положення, представлене кодером.
  - з нижчою частотою кожну мілісекунду (мс), наприклад, до
    1. скоригувати планування наступних рухів для виконання інструкції G-коду.
  - як компоненти ”користувацького простору”, що не працюють у реальному часі та виконують ”головний цикл”, як і будь-яке інше програмне забезпечення, і можуть бути перервані або затримані, коли решта системи зайнята або перевантажена.

Взяті разом, HAL дозволяє

1. програмувати для машини, яку програміст безпосередньо не знає, але може покладатися на програмний інтерфейс із чітко визначеним впливом на машину. Цей інтерфейс може бути використаний для
   - скажіть машині, що робити
   - послухайте, що машина хоче розповісти про стан, у якому вона знаходиться.
2. Вертикальні абстракції: Системний інтегратор-людина такої машини використовує HAL
   - описати, як виглядає машина та який кабель керує тим, який двигун приводить у рух яку вісь.
   - Опис машини, інтерфейси програміста та інтерфейс користувача якимось чином «зустрічаютьс в цьому абстрактному шарі.
3. Горизонтальні абстракції:
   - Не всі машини мають усілякі функції
   - Фрезерні, токарні верстати та роботи мають багато спільного
     - характеристики (двигуни, шарніри тощо),
     - алгоритми планування своїх рухів.

HAL не взаємодіє безпосередньо з користувачем. Але було передбачено кілька інтерфейсів, які дозволяють маніпулювати HAL

- з командного рядка за допомогою команди ”halcmd”.
- зі скриптів Python та
- з програм на C/C++,

але жоден з цих інтерфейсів не є «самим HAL».

HAL сама по собі не є програмою, вона складається з одного або декількох списків завантажених програм (компонентів), які періодично виконуються (у строгій послідовності), та області спільної пам’яті, яку ці компоненти використовують для обміну даними. Основний скрипт HAL виконується тільки один раз під час запуску машини, налаштовуючи потоки реального часу та місця спільної пам’яті, завантажуючи компоненти та налаштовуючи зв’язки даних між ними («сигнали» та «контакти»).

В принципі, кілька машин можуть спільно використовувати загальний HAL, щоб забезпечити їх взаємодію, проте поточна реалізація LinuxCNC обмежена одним інтерпретатором і одним модулем завдань. Наразі це майже завжди інтерпретатор G-коду і «milltask» (який, як виявилося, також добре працює для токарних верстатів і цілком підходить для роботів), але ці модулі можна вибирати під час завантаження. Зі зростанням інтересу до управління декількома взаємодіючими машинами, подолання цього обмеження, ймовірно, є одним з головних кроків для майбутнього розвитку LinuxCNC. Однак це дещо складно, і спільнота все ще обмірковує свої думки з цього приводу.

HAL лежить в основі LinuxCNC і використовується та/або розширюється всіма частинами LinuxCNC, включаючи графічні інтерфейси користувача. Інтерпретатор G-коду (або альтернативної мови) знає, як інтерпретувати G-код, і перекладає його в операції машини, запускаючи сигнали в HAL. Користувач може запитувати HAL різними способами, щоб отримати інформацію про його стан, який також відображає стан машини. Під час написання версії 2.9 графічні інтерфейси користувача все ще є винятком із цього правила і можуть знати те, чого HAL не знає (і не повинен знати).

5.1.2 Зв’язок

HAL особливий тим, що може дуже швидко спілкуватися

- з іншими програмами, але зокрема
- з його компонентами, які зазвичай виконуються в одному з потоків реального часу.

І під час спілкування частина LinuxCNC, з якою відбувається спілкування, не потребує підготовки до спілкування: всі ці дії виконуються асинхронно, тобто жоден компонент не перериває своє звичайне виконання для отримання сигналу, а сигнали можуть надсилатися відразу, тобто програма може чекати, поки надійде певне повідомлення, наприклад сигнал увімкнення, але їй не потрібно готуватися до отримання цього повідомлення.

Система зв’язку

- представляє та контролює все обладнання, підключене до системи,
- запускає та зупиняє інші програми, що взаємодіють.

Зв’язок з апаратним забезпеченням самої машини здійснюється відповідними спеціалізованими компонентами HAL.

Шар HAL є спільним простором, в якому всі численні частини, що складають LinuxCNC, обмінюються інформацією. Цей простір має контакти, які ідентифікуються за назвою, хоча інженер LinuxCNC може віддати перевагу асоціації з контактом електронної схеми. Ці контакти можуть передавати числові та логічні значення, булеві, плаваючі та цілочисельні значення з знаком і без знака. Існує також (відносно новий) тип контакту під назвою hal_port, призначений для байтових потоків, та фреймворк для обміну більш складними даними під назвою hal_stream (який використовує приватну спільну область пам’яті, а не контакт HAL). Ці два останні типи використовуються відносно рідко.

За допомогою HAL ви можете надіслати сигнал на цей контакт. Кожна частина HAL може зчитати значення сигналу з цього контакту. Це триває доти, доки на цей контакт не буде

надіслано новий сигнал, який замінить попереднє значення. Основна система обміну повідомленнями HAL не залежить від CNC, але HAL постачається з великою кількістю компонентів, які знають багато про CNC і передають цю інформацію через контакти. Є контакти, що представляють

- статична інформація про машину
- поточний стан машини
  - кінцеві вимикачі
  - позиції, що підраховуються кроковими двигунами або вимірюються енкодерами
- одержувачам інструкцій
  - ручне керування положенням машини (”поштовх”)
  - позиції, які крокові двигуни повинні зайняти далі

За аналогією з електронними кабелями, контакти можуть бути з’єднані між собою, тому значення, що змінюється в одному контакті, служить вхідним сигналом для іншого контакту. Компоненти HAL готують такі вхідні та вихідні контакти і, таким чином, автоматично запускаються для виконання.

Компоненти HAL Багато «експертних» програмних компонентів LinuxCNC зазвичай реалізовані як *компоненти* HAL, які концептуально також називаються *модулями*. Ці комп’ютерні експерти постійно зчитують з HAL інформацію про стан, якого повинна досягти машина, і порівнюють цей бажаний стан із станом, в якому машина перебуває в даний момент. Коли існує різниця між тим, що повинно бути, і тим, що є в даний момент, виконуються певні дії для зменшення цієї різниці, при цьому постійно записуються оновлення поточного стану назад в простір даних HAL.

Є компоненти, що спеціалізуються на взаємодії з кроковими двигунами, а інші компоненти знають, як керувати сервоприводами. На вищому рівні деякі компоненти знають, як осі машини розташовані в 3D, а інші знають, як виконувати плавний рух від однієї точки простору до іншої. Токарні верстати, фрезерні верстати та роботи будуть відрізнятися за компонентами LinuxCNC, які є активними, тобто завантаженими конфігураційним файлом HAL для цієї машини. Проте дві машини можуть виглядати дуже по-різному, оскільки побудовані для дуже різних цілей, але якщо вони обидві використовують сервомотори, то вони все одно можуть використовувати один і той самий сервокомпонент HAL.

Походження стимулу до переїзду На найнижчому (найближчому до апаратного забезпечення) рівні, наприклад для крокових двигунів, опис стану цього двигуна є дуже інтуїтивним: це кількість кроків у певному напрямку. Різниця між бажаним положенням і фактичним положенням перетворюєт на рух. Швидкість, прискорення та інші параметри можуть бути внутрішньо обмежені в самому компоненті або, за бажанням, обмежені компонентами, розташованими вище за течією. (Наприклад, у більшості випадків значення положення осі, що надсилаються до компонентів генератора кроків, вже обмежені та сформовані відповідно до налаштованих обмежень машини або поточної швидкості подачі.)

Будь-який рядок G-коду інтерпретується та запускає набір процедур, які, у свою чергу, знають, як взаємодіяти з компонентами, що знаходяться на середньому шарі, наприклад, для створення кола.

Піни та сигнали HAL займає особливе місце в серцях програмістів завдяки способу представлення потоку даних між модулями. Коли традиційні програмісти думають про змінні, адреси або порти вводу-виводу, HAL посилається на «контакти». Ці контакти з’єднуються або отримують значення за допомогою сигналів. Подібно до того, як інженер-електрик з’єднує дроти між контактами компонентів млина, інженер HAL встановлює потік даних між контактами екземплярів модулів.

Графічний інтерфейс LinuxCNC (AXIS, GMOCCAPY, Touchy тощо) відображає стан деяких контактів (наприклад, кінцевих вимикачів), але для усунення несправностей і налаштування існують також інші графічні інструменти: Halshow, Halmeter, Halscope і Halreport.

У решті цього вступу представлено

- синтаксис того, як контакти різних компонентів з’єднані у файлах конфігурації HAL, та
- програмне забезпечення для перевірки значень контактів
  - у будь-який момент,
  - розвиваючись з часом.

5.1.3 Проектування системи HAL

.HAL базується на традиційних методах проектування систем.

HAL базується на тих самих принципах, що використовуються для проектування апаратних схем і систем, тому спочатку варто розглянути ці принципи. Будь-яка система, включаючи верстат з CNC, складається з взаємопов’язаних компонентів. Для верстата з CNC цими компонентами можуть бути головний контролер, сервопідсилювачі або крокові приводи, двигуни, енкодери, кінцеві вимикачі, пускові пристрої, можливо, частотний перетворювач для приводу шпинделя, PLC для управління зміною інструменту тощо. Виробник верстата повинен вибрати, змонтувати та з’єднати ці деталі, щоб створити цілісну систему.

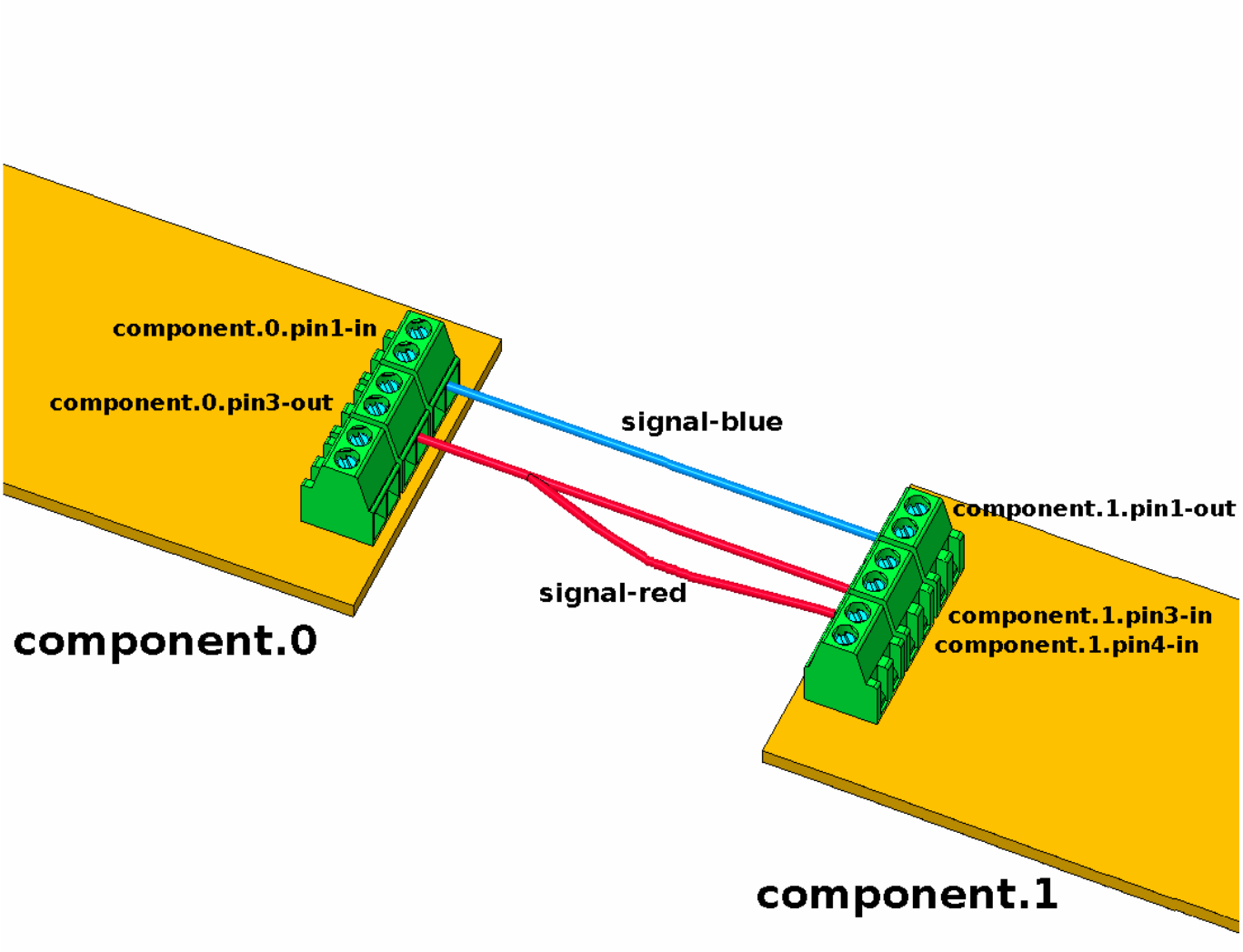


Figure 5.1: Концепція HAL - З’єднання подібних електричних кіл.

Рисунок один буде записано в коді HAL ось так:

```
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 Вибір деталі

Конструктор машини не повинен турбуватися про те, як працює кожна окрема деталь. Він розглядає їх як «чорні скриньки». На етапі проектування він вирішує, які деталі він буде використовува — крокові або сервоприводи, сервопідсилювач якої марки, які та скільки кінцевих вимикачів тощо. Рішення інтегратора про те, які конкретні компоненти використовувати, базується на функціях цих компонентів та технічних характеристиках, наданих виробником пристрою. Розмір двигуна і навантаження, яке він повинен приводити в рух, впливатимуть на вибір підсилювача, необхідного для його роботи. Вибір підсилювача може впливати на типи зворотного зв'язку, необхідні підсилювачу, і сигнали швидкості або положення, які повинні надсилатися до підсилювача від системи управління.

У світі HAL інтегратор повинен вирішити, які компоненти HAL необхідні. Зазвичай для кожної інтерфейсної плати потрібен драйвер. Додаткові компоненти можуть знадобитися для програмного генерування імпульсів кроку, функціональності PLC та широкого спектру інших завдань.

5.1.3.2 Проектування взаємозв'язків

Розробник апаратної системи не тільки вибирає деталі, але й вирішує, як ці деталі будуть з'єднані між собою. Кожна чорна скринька має клеми, можливо, тільки дві для простого вимикача або десятки для сервоприводу чи PLC. Вони повинні бути з'єднані між собою. Двигуни підключаються до сервопідсилювачів, кінцеві вимикачі підключаються до контролера тощо. Під час роботи над проектом конструктор створює велику схему підключення, яка показує, як усі деталі повинні бути з'єднані між собою.

Під час використання HAL компоненти з'єднуються між собою сигналами. Розробник повинен вирішити, які сигнали потрібні та що вони повинні з'єднувати.

5.1.3.3 Впровадження

Після завершення складання схеми підключення настає час збирати машину. Необхідно придбати та встановити деталі, а потім з'єднати їх між собою відповідно до схеми підключення. У фізичній системі кожне з'єднання являє собою шматок дроту, який потрібно відрізати та підключити до відповідних клем.

HAL надає ряд інструментів, які допомагають «побудувати» систему HAL. Деякі з цих інструментів дозволяють «підключити» (або відключити) окремий «провід». Інші інструменти дозволяють зберегти повний перелік усіх деталей, проводів та іншої інформації про систему, щоб її можна було «відновити» за допомогою однієї команди.

5.1.3.4 Тестування

Дуже мало машин працюють правильно з першого разу. Під час тестування конструктор може використовувати вимірювальний прилад, щоб перевірити, чи працює кінцевий вимикач, або виміряти напругу постійного струму, що подається на сервомотор. Він може підключити осцилограф, щоб перевірити налаштування приводу або знайти електричні перешкоди. Він може виявити проблему, яка вимагає зміни схеми підключення; можливо, деталь потрібно підключити по-іншому або замінити на щось зовсім інше.

HAL надає програмні еквіваленти вольтметра, осцилографа, генератора сигналів та інших інструменті необхідних для тестування та налаштування системи. Ті самі команди, що використовуються для побудови системи, можуть бути використані для внесення необхідних змін.

5.1.3.5 Короткий зміст

Цей документ призначений для людей, які вже знають, як здійснювати таку інтеграцію апаратних систем, але не знають, як підключити апаратне забезпечення до LinuxCNC. Дивіться розділ Приклад віддаленого запуску в документації HAL UI Examples.

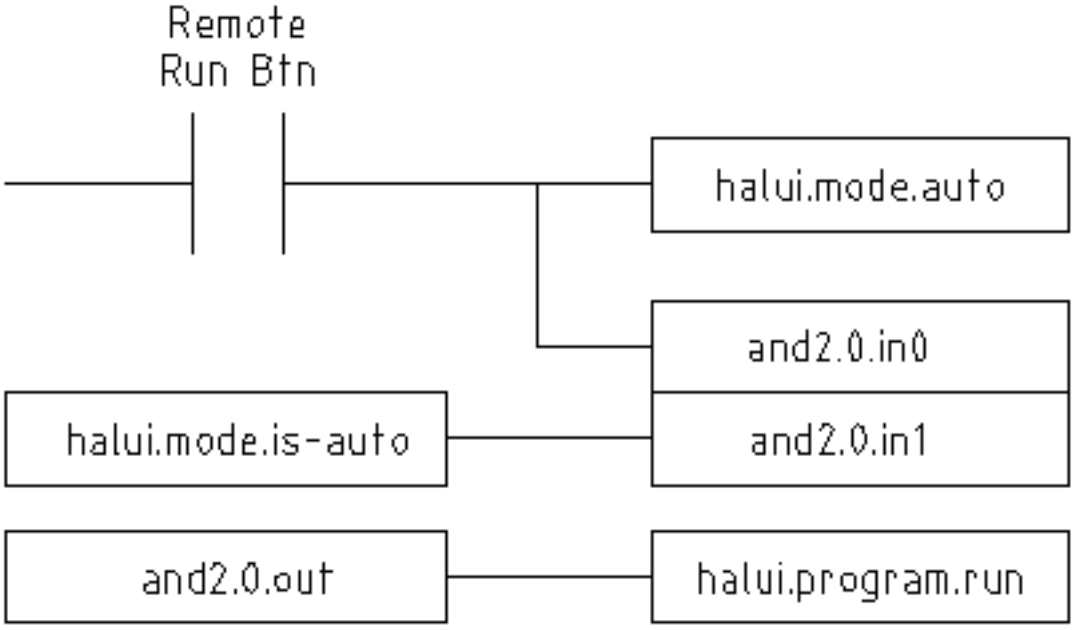


Figure 5.2: Приклад віддаленого запуску (схема)

Традиційна конструкція обладнання, описана вище, закінчується на межі головного блоку управління. За межами блоку управління знаходиться низка відносно простих блоків, з'єднаних між собою для виконання необхідних функцій. Всередині блок управління є великою загадкою — величезною чорною скринькою, яка, сподіваємося, працює.

HAL розширює цей традиційний метод проектування апаратного забезпечення на внутрішню частину великої чорної скриньки. Він перетворює драйвери пристроїв і навіть деякі внутрішні частини контролера на менші чорні скриньки, які можна з'єднувати між собою і навіть замінювати, як і зовнішнє апаратне забезпечення. Це дозволяє «схемі підключення системи» показувати частину внутрішнього контролера, а не тільки велику чорну коробку. І найголовніше, це дозволяє інтегратору тестувати та модифікувати контролер, використовуючи ті ж методи, які він використовував би для решти апаратного забезпечення.

Такі терміни, як двигуни, підсилювачі та енкодери, знайомі більшості інтеграторів машин. Коли ми говоримо про використання надзвичайно гнучкого екранованого кабелю з вісьмома провідниками для підключення енкодера до плати входу сервоприводу в комп'ютері, читач відразу розуміє, про що йдеться, і задається питанням: «Які роз'єми мені знадобляться для кожного кінця?». Такий самий підхід є необхідним для HAL, але конкретний хід думок може зайняти трохи часу, щоб налагодитися. Використання термінів HAL спочатку може здатися трохи дивним, але концепція роботи від одного з'єднання до наступного є такою самою.

Ідея розширення схеми підключення на внутрішню частину контролера і є сутністю HAL. Якщо ви звикли до ідеї взаємозв'язку апаратних чорних скриньок, то, ймовірно, не матимете проблем із використанням HAL для взаємозв'язку програмних чорних скриньок.

5.1.4 Концепції HAL

Цей розділ є глосарієм, що містить визначення основних термінів HAL, але він дещо відрізняється від традиційного глосарію, оскільки терміни в ньому не розташовані в алфавітному порядку. Вони розташовані відповідно до їхнього взаємозв'язку або послідовності в системі HAL.

Компонент:: Коли ми говорили про дизайн апаратного забезпечення, ми називали окремі елементи «деталями», «будівельними блоками», «чорними скриньками» тощо. Еквівалентом HAL є «компонент» або «компонент HAL». У цьому документі використовується термін «компонент HAL», коли існує ймовірність плутанини з іншими видами компонентів, але зазвичай використовується просто «компонент». Компонент HAL — це частина програмного забезпечення з чітко визначеними входами, виходами та поведінкою, яку можна встановлювати та з'єднувати між собою за потреби. + + Багато компонентів HAL моделюють поведінку матеріальної частини машини, і **контакт** дійсно може бути призначений для підключення до **фізичного контакту** на пристрої для зв'язку з ним, звідси і походить назва. Але найчастіше це не так. Уявіть собі модернізацію ручного токарного/фрезерного верстата. LinuxCNC реалізує те, як машина представляє себе зовнішньому світу, і другорядним є те, чи реалізація малювання кола вже реалізована на машині, чи надається з LinuxCNC. І зазвичай до уявної модернізації додають кнопки, які **сигналізують** про дію, наприклад, аварійну зупинку. LinuxCNC і машина стають єдиним цілим. І це завдяки HAL.

Параметр:: Багато апаратних компонентів мають налаштування, які не пов'язані з іншими компонента але все одно потребують доступу. Наприклад, сервопідсилювачі часто мають підлаштувальні потенціометри для налаштування та тестові точки, до яких можна підключити вимірювальний прилад або осцилограф для перегляду результатів налаштування. Компоненти HAL також можуть мати такі елементи, які називаються «параметрами». Існує два типи параметрів: вхідні параметри еквівалентні трим-потенціометрам — це значення, які користувач може налаштовувати і які залишаються фіксованими після встановлення. Вихідні параметри не можуть бути налаштовані користувачем — вони еквівалентні тестовим точкам, які дозволяють контролювати внутрішні сигнали.

Закріпити:: Апаратні компоненти мають термінали, які використовуються для їх з'єднання між собою. Еквівалентом HAL є «контакт» або «контакт HAL». «Контакт HAL» використовується в разі необхідності, щоб уникнути плутанини. Всі контакти HAL мають назви, і ці назви використовуютьс при їх з'єднанні між собою. Контакти HAL є програмними об'єктами, які існують тільки всередині комп'ютера.

Physical_Pin:: Багато пристроїв вводу-виводу мають реальні фізичні контакти або термінали, які підключаються до зовнішнього обладнання, наприклад, контакти роз'єму паралельного порту. Щоб уникнути плутанини, їх називають «фізичними контактами». Це ті елементи, які «виступають» у реальний світ.

Note
Ви можете задатися питанням, який зв'язок існує між HAL_pins, physical_pins та зовнішніми елементами, такими як кодери або карта STG: тут ми маємо справу з інтерфейсами типу перетворення/конвертації даних.

Сигнал:: У фізичній машині клеми реальних апаратних компонентів з'єднані між собою дротами. Еквівалентом дроту в HAL є «сигнал» або «сигнал HAL». Сигнали HAL з'єднують контакти HAL між собою відповідно до вимог виробника машини. Сигнали HAL можна від'єднувати та під'єднувати за бажанням (навіть під час роботи машини).

Тип:: При використанні реального обладнання ви не підключали б 24-вольтовий релейний вихід до аналогового входу +/-10 В сервопідсилювача. Контакти HAL мають ті самі обмеження, які залежать від їхнього типу. І контакти, і сигнали мають типи, і сигнали можна підключати тільки до контактів того самого типу. Наразі існує 4 типи, а саме:

+ - біт - одне значення TRUE/FALSE або ON/OFF - float — 64-бітове число з плаваючою комою, з роздільною здатністю приблизно 53 біти та динамічним діапазоном понад 1000 біт. - u32 – 32-бітове беззнакове ціле число, допустимі значення від 0 до 4 294 967 295 - s32 – 32-бітове знакове

ціле число, допустимі значення від -2 147 483 648 до +2 147 483 647 - u64 – 64-бітове беззнакове ціле число, допустимі значення від 0 до 18 446 744 073 709 551 615 - s64 – 64-бітове знакове ціле число, допустимі значення від -9,223,372,036,854,775,808 до +9,223,372,036,854,775,807

Функція:: Реальні апаратні компоненти, як правило, реагують на вхідні сигнали миттєво. Наприклад, якщо вхідна напруга на сервопідсилювачі змінюється, вихідна напруга також змінюється автоматично Проте програмні компоненти не можуть діяти «автоматично». Кожен компонент має специфічний код, який повинен бути виконаний, щоб компонент міг виконувати свої функції. У деяких випадках цей код просто виконується як частина компонента. Однак у більшості випадків, особливо в компонентах реального часу, код повинен виконуватися в певній послідовності та з певними інтервалами. Наприклад, вхідні дані повинні бути прочитані перед тим, як виконувати обчислення на вхідних даних, а вихідні дані не повинні записуватися до завершення обчислень. У цих випадках код стає доступним для системи у вигляді однієї або декількох «функцій». Кожна функція є блоком коду, який виконує певну дію. Системний інтегратор може використовувати «потоки» для планування серії функцій, які будуть виконуватися в певному порядку і через певні проміжки часу.

Нитка:: «Потік» — це список функцій, які виконуються через певні проміжки часу в рамках завдання, що виконується в режимі реального часу. Коли потік створюється вперше, він має певний проміжок часу (період), але не має функцій. Функції можна додавати до потоку, і вони будуть виконуватися послідовно кожного разу, коли потік запускається.

Наприклад, припустимо, що ми маємо компонент parport з назвою hal_parport. Цей компонент визначає один або кілька контактів HAL для кожного фізичного контакту. Контакти описані в розділі документації цього компонента: їхні назви, як кожен контакт пов’язаний з фізичним контактом, чи є вони інвертованими, чи можна змінити полярність тощо. Але цього недостатньо, щоб передати дані з контактів HAL на фізичні контакти. Для цього потрібен код, і саме тут на сцену виходять функції. Компонент parport потребує щонайменше двох функцій: одна для зчитування фізичних вхідних контактів і оновлення контактів HAL, інша для отримання даних з контактів HAL і запису їх на фізичні вихідні контакти. Обидві ці функції є частиною драйвера parport.

5.1.5 Компоненти HAL

Кожен компонент HAL — це програмне забезпечення з чітко визначеними вхідними даними, вихідними даними та поведінкою, яке можна встановлювати та з’єднувати між собою за потреби. У розділі HAL Components List наведено перелік усіх доступних компонентів та короткий опис їхніх функцій.

5.1.6 Проблеми з синхронізацією в HAL

На відміну від моделей фізичного з’єднання між чорними ящиками, на яких, як ми вже говорили, базується HAL, просте з’єднання двох контактів за допомогою HAL-сигналу далеко не відповідає дії фізичного корпусу.

Справжня релейна логіка складається з реле, з’єднаних між собою, і коли контакт розмикається або замикається, струм негайно протікає (або зупиняється). Інші котушки можуть змінювати стан тощо, і все це просто «відбувається». Але в логіці типу PLC це працює не так. Зазвичай під час одного проходу по драбині кожна сходинка оцінюється в порядку її появи і тільки один раз за прохід. Ідеальним прикладом є драбина з однією сходинкою, з контактом NC, з’єднаним послідовно з котушкою. Контакт і котушка належать до одного реле.

Якби це було звичайне реле, то щойно на котушку подається живлення, контакти починають розмикатися та знеструмлювати її. Це означає, що контакти знову замикаються тощо, і т.д. Реле перетворюється на зумер.

У випадку з PLC, якщо котушка вимкнена, а контакт закритий, коли PLC починає оцінювати щабель, то після завершення цього проходу котушка вмикається. Той факт, що вмикання котушки

відкриває контакт, який її живить, ігнорується до наступного проходу. На наступному проході PLC бачить, що контакт відкритий, і знеструмлює котушку. Таким чином, реле все ще швидко перемикається між увімкненим і вимкненим станом, але з частотою, яка визначається тим, як часто PLC оцінює щабель.

У HAL функція — це код, який оцінює щабель (щаблі). Фактично, реальна версія ClassicLadder, що підтримує HAL, експортує функцію, яка саме це і робить. Тим часом, потік — це те, що запускає функцію через певні проміжки часу. Так само, як ви можете вибрати, щоб PLC оцінював усі свої щаблі кожні 10 мс або кожну секунду, ви можете визначити потоки HAL з різними періодами.

Різниця між потоками полягає не в тому, що саме потік робить, а в тому, які функції з ним пов’язані. Справжня відмінність полягає просто в тому, як часто потік виконується.

У LinuxCNC у вас може бути потік тривалістю 50 мкс та потік тривалістю 1 мс. Вони будуть створені на основі BASE_PERIOD та SERVO_PERIOD, фактичний час залежить від значень у вашому INI-файлі.

Наступний крок – вирішити, що має робити кожен потік. Деякі з цих рішень однакові в (майже) будь-якій системі LinuxCNC. Наприклад, обробник команд руху завжди додається до servo-thread.

Інші з’єднання будуть виконані інтегратором. Вони можуть включати підключення функцій зчитування енкодера драйвера STG і запису DAC до сервопотоку, або підключення функції Step-Gen до базового потоку, разом з функцією (функціями) parport для запису кроків в порт.

5.2 Основи HAL

Цей документ містить довідку з основ HAL.

5.2.1 Команди HAL

Більш детальну інформацію можна знайти на сторінці довідки для halcmd: виконайте команду *man halcmd* у вікні терміналу.

Щоб переглянути конфігурацію HAL та перевірити стан контактів і параметрів, скористайтеся вікном «HAL Configuration» (Конфігурація HAL) в меню «Machine» (Машина) в AXIS. Щоб переглянути стан контакту, відкрийте вкладку «Watch» (Перегляд) і натисніть на кожен контакт, який ви хочете переглянути, і він буде доданий до вікна перегляду.

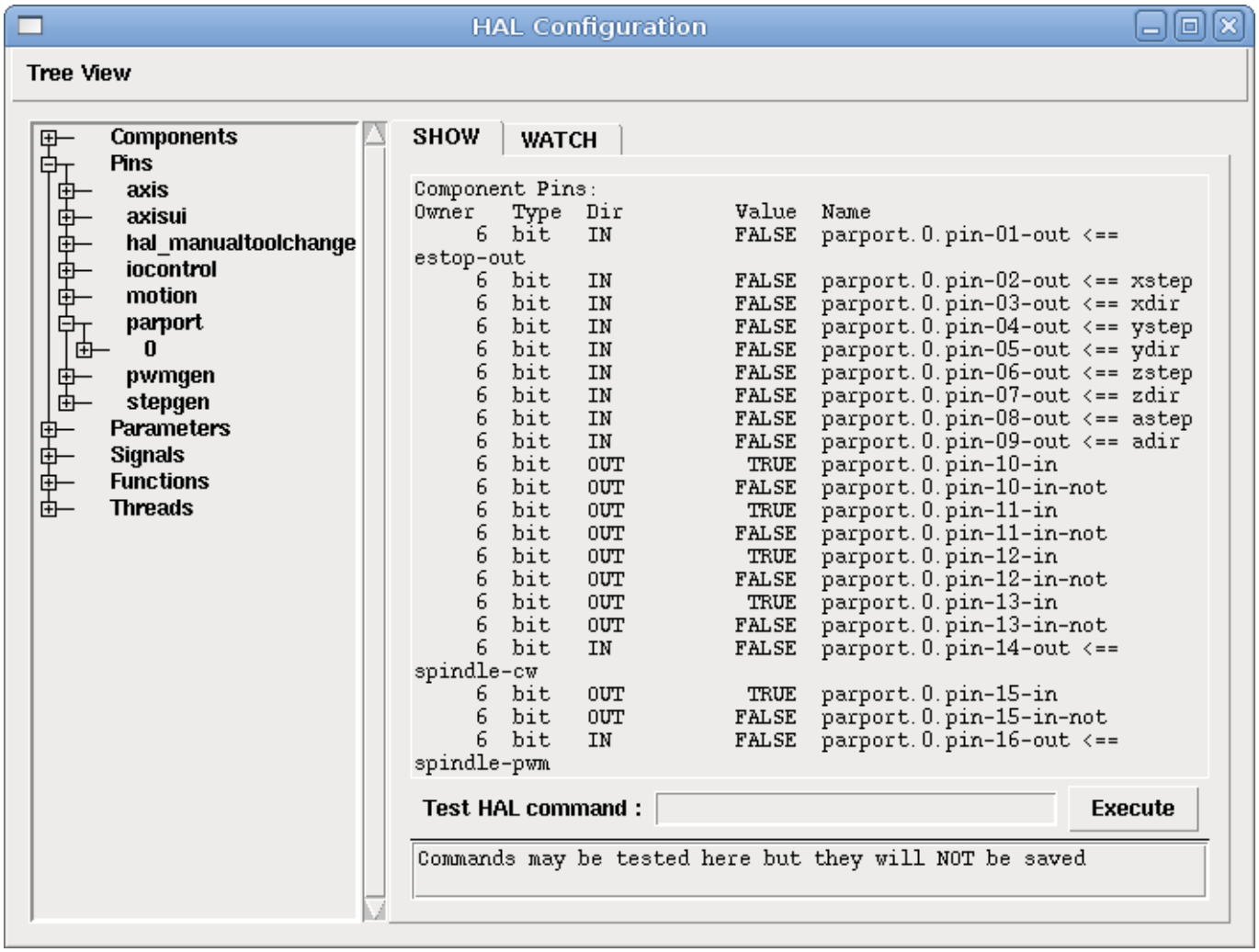


Figure 5.3: Вікно конфігурації HAL

5.2.1.1 loadrt

Команда `loadrt` завантажує компонент HAL реального часу. Функції компонента реального часу потрібно додати до потоку, щоб вони оновлювалися з частотою потоку. Неможливо завантажити компонент, що не працює в режимі реального часу, у простір реального часу.

Синтаксис та приклад loadrt

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

5.2.1.2 addf

Команда `addf` додає функцію до потоку реального часу. Якщо для створення конфігурації використовув майстер StepConf, було створено два потоки (``base-thread`` та ``servo-thread``).

`addf` додає функцію *functname* до потоку *threadname*. За замовчуванням функції додаються в тому порядку, в якому вони знаходяться у файлі. Якщо вказано *position*, функція додається до цього місця в потоці. Від'ємне значення *position* вказує на позицію відносно кінця потоку. Наприклад, «1» — початок потоку, «-1» — кінець потоку, «-3» — третє місце від кінця.

Для деяких функцій важливо завантажувати їх у певному порядку, наприклад функції читання та запису parport. Назва функції зазвичай складається з назви компонента та цифри. У наведеному нижче прикладі завантажується компонент «or2», а команда «show function» показує назву функції or2.

```
$ halrun
halcmd: loadrt or2
halcmd: show function
b''Eb''b''кb''b''cb''b''пb''b''ob''b''pb''b''тb''b''ob''b''вb''b''ab''b''нb''b''ib'' b' ↩
    'фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb'':
Owner   CodeAddr  Arg       FP   Users  Name
 00004  f8bc5000  f8f950c8  NO       0   or2.0
```

Ви повинні додати функцію з компонента HAL реального часу до потоку, щоб функція оновлювалася зі швидкістю потоку. Зазвичай існує два потоки, як показано в цьому прикладі. Деякі компоненти використовують математику з плаваючою комою і повинні бути додані до потоку, який підтримує математику з плаваючою комою. FP вказує, чи підтримується математика з плаваючою комою в цьому потоці.

```
$ halrun
halcmd: loadrt motmod base_period_nsec=55555 servo_period_nsec=1000000 num_joints=3
halcmd: b''пb''b''ob''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''тb''b''eb''b''мb''b' ↩
    'yb''
b''Tb''b''eb''b''мb''b''иb'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'мb''b''yb'' b''чb''b''ab''b''cb''b''ib'':
     Period  FP     Name               (     Time, Max-Time )
     995976  YES          servo-thread (        0,        0 )
      55332  NO            base-thread (        0,        0 )
```

- базовий потік (високошвидкісний потік): цей потік обробляє елементи, що потребують швидкої реакції, такі як створення крокових імпульсів, читання та запис паралельного порту. Не підтримує математику з плаваючою комою.
- серво-потік (потік з низькою швидкістю): цей потік обробляє елементи, які можуть терпіти повільнішу реакцію, такі як контролер руху, ClassicLadder та обробник команд руху, і підтримує обчислення з плаваючою комою.

Синтаксис та приклад addf

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

> **Note**
> Якщо компонент потребує потоку з плаваючою комою, це зазвичай повільніший сервопотік.

5.2.1.3 loadusr

Команда `loadusr` завантажує компонент HAL, що не працює в режимі реального часу. Програми, що не працюють в режимі реального часу, є окремими процесами, які за бажанням можуть взаємодіяти з іншими компонентами HAL за допомогою контактів і параметрів. Ви не можете завантажувати компоненти, що працюють в режимі реального часу, в простір, що не працює в режимі реального часу.

Прапори можуть бути одним або кількома з наступних:

-W	чекати готовності компонента. Вважається, що компонент має таку ж назву, як і перший аргумент команди.
-Wn <name>	чекати на компонент, який матиме задане <name>. Це застосовується лише якщо компонент має опцію name.
-w	чекати на завершення програми
-i	ігнорувати значення, що повертається програмою (з опцією -w)
-n	назвати компонент, коли це є допустимим варіантом для цього компонента.

Синтаксис та приклади loadusr

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

Англійською це означає «loadusr чекати на назву шпинделя компонент gs2_vfd назва шпинделя».

5.2.1.4 net

Команда `net` створює «з'єднання» між сигналом і одним або декількома контактами. Якщо сигнал не існує, net створює новий сигнал. Це замінює необхідність використання команди newsig. Опціональні стрілки напрямку <=, => і <=> полегшують розуміння логіки при читанні командного рядка `net` і не використовуються командою net. Стрілки напрямку повинні бути відокремлені пробілом від імен контактів.

Синтаксис та приклади слова 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
```

У наведеному вище прикладі `home-x` — це назва сигналу, `joint.0.home-sw-in` — це контакт «Direction IN», <= — це необов'язкова стрілка напрямку, а `parport.0.pin-11-in` — це контакт «Direction OUT». Це може здатися заплутаним, але мітки входу та виходу для контакту паралельного порту вказують на фізичний спосіб роботи контакту, а не на те, як він обробляється в HAL.

Вивід можна підключити до сигналу, якщо він підпорядковується таким правилам:

- Вивід IN завжди можна підключити до сигналу.
- Вивід IO можна підключити, якщо на сигналі немає виводу OUT.
- Вивід OUT можна підключити, лише якщо на сигналі немає інших виводів OUT або IO.

Одне й те саме «ім'я сигналу» можна використовувати в кількох мережевих командах для підключення додаткових контактів, якщо дотримуватися наведених вище правил.

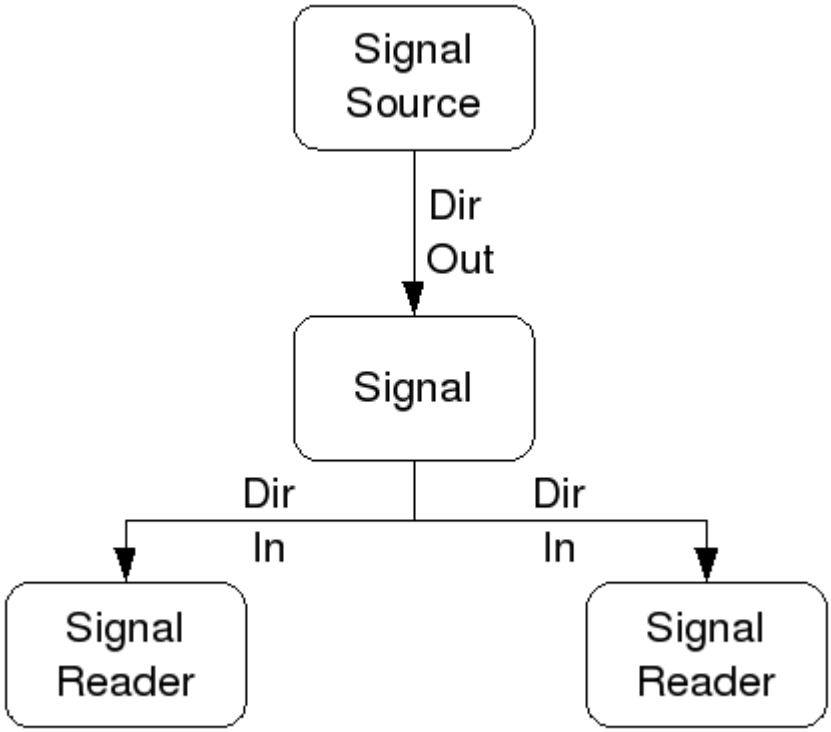


Figure 5.4: Напрямок сигналу

Цей приклад показує сигнал xStep, джерелом якого є `stepgen.0.out`, та два зчитувачі, `parport.0.pin-` та `parport.0.pin-08-out`. В основному значення `stepgen.0.out` надсилається до сигналу xStep, а потім це значення надсилається до `parport.0.pin-02-out` та `parport.0.pin-08-out`.

```
#   b''cb''b''ив''b''гb''b''нb''b''аb''b''лb''    b''дb''b''жb''b''eb''b''pb''b''eb''b' ↩
    'лb''b''оb''           b''пb''b''pb''b''ив''b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b ↩
    ''нb''b''яb''         b''пb''b''pb''b''ив''b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb''
net xStep stepgen.0.out => parport.0.pin-02-out parport.0.pin-08-out
```

Оскільки сигнал xStep містить значення `stepgen.0.out` (джерело), ви можете знову використовувати той самий сигнал, щоб надіслати значення іншому зчитувачу. Для цього просто використовуйте сигнал із зчитувачами на іншому рядку.

```
#   b''cb''b''ив''b''гb''b''нb''b''аb''b''лb''       b''пb''b''уb''b''нb''b''кb''b''тb'' b' ↩
    'пb''b''pb''b''ив''b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b''нb''b''яb''2
net xStep => parport.0.pin-06-out
```

Контакти вводу/виводу Значення енкодера, подібного до контакту вводу/виводу.*N*.index-enable, можна зчитувати або встановлювати відповідно до дозволеного компонентом параметра.

5.2.1.5 setp

Команда `setp` встановлює значення виводу або параметра. Дійсні значення залежатимуть від типу виводу або параметра. Якщо типи даних не збігаються, це вважається помилкою.

Деякі компоненти мають параметри, які потрібно встановити перед використанням. Параметри можна встановити перед використанням або під час роботи за потреби. Ви не можете використовувати setp на виводі, підключеному до сигналу.

Синтаксис та приклади setp

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

5.2.1.6 sets

Команда `sets` встановлює значення сигналу.

Синтаксис та приклади множин

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

Це помилка, якщо:

- Назва сигналу не існує
- Якщо сигнал вже має записувач
- Якщо значення не є правильним типом для сигналу

5.2.1.7 unlinkp

Команда `unlinkp` від'єднує пін від підключеного сигналу. Якщо перед виконанням команди до піну не було підключено жодного сигналу, нічого не відбувається. Команда `unlinkp` корисна для усунення несправностей.

Синтаксис та приклади unlinkp

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

5.2.1.8 Застарілі команди

Наступні команди є застарілими і можуть бути видалені з майбутніх версій. У будь-якій новій конфігурації слід використовувати команду net. Ці команди включені, щоб старі конфігурації продовжували працювати.

Команда `linksp` створює «з'єднання» між сигналом та одним виводом.

Синтаксис та приклади linksp

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

Команду `linksp` було замінено командою `net`.

Команда `linkps` створює «з'єднання» між одним виводом та одним сигналом. Це те саме, що й `linksp`, але аргументи протилежні.

Синтаксис та приклади linkps

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

Команду `linkps` було замінено командою `net`.

Команда `newsig` створює новий HAL-сигнал з іменем *<signame>* та типом даних *<type>*. Тип має бути *bit*, *s32*, *u32*, *s64*, *u64* або *float*. Помилка, якщо *<signame>* вже існує.

Синтаксис та приклади newsig

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

Більше інформації можна знайти в посібнику HAL або на сторінках довідки для `halrun`.

5.2.2 Дані HAL

5.2.2.1 Біт

Бітове значення є увімкненим або вимкненим.

- значення бітів = true або 1 та false або 0 (True, TRUE, true є дійсними)

5.2.2.2 Float

«Число з плаваючою комою» — це число з плаваючою комою. Іншими словами, десяткова кома може переміщуватися за потреби.

- значення з плаваючою комою = 64-бітове значення з плаваючою комою, з роздільною здатністю приблизно 53 біт та динамічним діапазоном понад 2^{10} (~ 1000) біт.

Для отримання додаткової інформації про числа з плаваючою комою див.:

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

5.2.2.3 s32

Число *s32* — це ціле число, яке може мати від’ємне або додатне значення.

- Значення s32 = цілі числа від -2147483648 до 2147483647

5.2.2.4 u32

Число «u32» — це ціле число, яке є лише додатним.

- значення u32 = цілі числа від 0 до 4294967295

5.2.2.5 s64

Число «s64» — це ціле число, яке може мати від’ємне або додатне значення.

- значення s64 = цілі числа від -9,223,372,036,854,775,808 до +9,223,372,036,854,775,807

5.2.2.6 u64

Число «u64» — це ціле число, яке є лише додатним.

- значення u64 = цілі числа від 0 до 18,446,744,073,709,551,615

5.2.3 Файли HAL

Якщо ви використовували майстер налаштування Stepper для створення конфігурації, у вашому каталозі конфігурації буде до трьох файлів HAL.

- «my-mill.hal» (якщо ваша конфігурація має назву «my-mill»). Цей файл завантажується першим і його не слід змінювати, якщо ви використовували майстер налаштування крокового двигуна.
- «custom.hal». Цей файл завантажується наступним і перед завантаженням графічного інтерфейсу. Тут ви розміщуєте власні команди HAL, які хочете завантажити перед завантаженням графічного інтерфейсу.
- *custom_postgui.hal* Цей файл завантажується після завантаження графічного інтерфейсу користувач Тут ви розміщуєте власні команди HAL, які потрібно завантажити після завантаження графічного інтерфейсу користувача. Усі команди HAL, які використовують віджети PyVCP, потрібно розміщувати тут.

5.2.4 Параметр HAL

До кожного компонента HAL автоматично додаються два параметри під час його створення. Ці параметри дозволяють визначити час виконання компонента.

`.time`	Час – це кількість циклів процесора, необхідних для виконання функції.
`.tmax`	Tmax – це максимальна кількість циклів процесора, необхідних для виконання функції.

`tmax` — це параметр читання/запису, тому користувач може встановити його на 0, щоб позбутися першої ініціалізації під час виконання функції.

5.2.5 Основні логічні компоненти

HAL містить кілька логічних компонентів, що працюють у режимі реального часу. Логічні компоненти працюють за «таблицею істинності», яка визначає вихідні дані для будь-яких вхідних даних. Зазвичай це бітові маніпулятори, які працюють за таблицями істинності електричних логічних вентилів.

Щоб дізнатися більше про компоненти, див Список компонентів HAL або сторінки довідки.

5.2.5.1 and2

Компонента `and2` — це двовхідний елемент типу ”і”. Таблиця істинності нижче показує вихід на основі кожної комбінації вхідних даних.

Синтаксис

```
and2 [count=N] | [names=name1[,name2...]]
```

Функції

```
and2.n
```

Піни

```
and2.N.in0 (bit, in)
and2.N.in1 (bit, in)
and2.N.out (bit, out)
```

Table 5.3: Таблиця істинності i2

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

5.2.5.2 не

Компонент not є бітовим інвертором.

Синтаксис

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

Функції

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

Піни

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

Table 5.4: Таблиця істинності слова «не»

in	out
True	False
False	True

5.2.5.3 or2

Компонент or2 — це двовхідний елемент-або.

Синтаксис

```
or2[count=n] | [names=name1[,name2...]]
```

Функції

```
or2.n
```

Піни

```
or2.n.in0 (bit, in)
or2.n.in1 (bit, in)
or2.n.out (bit, out)
```

Table 5.5: Таблиця істинності or2

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

5.2.5.4 xor2

Компонент `xor2` — це двовхідний вентиль XOR (виключне АБО).

Синтаксис

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

Функції

```
xor2.n
```

Піни

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

Table 5.6: Таблиця істинності xor2

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

5.2.6 Логічні приклади

.Приклад використання `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
```

У наведеному вище прикладі одна копія and2 завантажується в простір реального часу і додається до серво-потоку. Далі pin-11 паралельного порту підключається до біту in0 логічної функції «і». Далі pin-12 підключається до біта in1 логічної функції «і». Насамкінець ми підключаємо вихідний біт and2 до паралельного порту pin-14. Отже, згідно з таблицею істинності для and2, якщо виводи 11 і 12 увімкнені, то вихідний вивід 14 буде увімкнений.

5.2.7 Компоненти конверсії

5.2.7.1 weighted_sum

Зважена сума перетворює групу бітів у ціле число. Перетворення є сумою «ваги» наявних бітів плюс будь-яке зміщення. Це схоже на «двійково-кодовану десяткову систему», але з більшою кількістю опцій. Біт «утримання» перериває обробку вхідних даних, щоб значення «суми» більше не змінювалося.

Синтаксис для завантаження компонента weighted_sum

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

Створює групи ``weighted_sum``, кожна з яких має задану кількість вхідних бітів (розмір).

Щоб оновити weighted_sum, process_wsums має бути приєднаний до потоку.

Додати process_wsums до потоку серво

```
addf process_wsums servo-thread
```

Що оновлює компонент weighted_sum.

У наступному прикладі, копії вікна конфігурації AXIS HAL, біти *0* та *2* мають значення TRUE, вони не мають зміщення. Вага (*weight*) біта 0 дорівнює 1, біта 2 – 4, тому сума дорівнює 5.

Table 5.7: Компонентні контакти weighted_sum

Власник	Тип	Ви	Значення	Ім’я
10	біт	У	TRUE	wsum.0.bit.0.in
10	s32	I/O	1	wsum.0.bit.0.weight
10	біт	У	FALSE	wsum.0.bit.1.in
10	s32	I/O	2	wsum.0.bit.1.weight
10	біт	У	TRUE	wsum.0.bit.2.in
10	s32	I/O	4	wsum.0.bit.2.weight
10	біт	У	FALSE	wsum.0.bit.3.in
10	s32	I/O	8	wsum.0.bit.3.weight
10	біт	У	FALSE	wsum.0.hold
10	s32	I/O	0	wsum.0.offset
10	s32	Вихід	5	wsum.0.sum

5.3 HAL TWOPASS

5.3.1 TWOPASS

У цьому розділі описано опцію, що дозволяє використовувати кілька команд завантаження для декількох екземплярів одного і того ж компонента в різних місцях файлу або в різних файлах.

Внутрішньо це вимагає дворазового читання файлу HAL, звідси і назва TWOPASS. Підтримувана з версії LinuxCNC 2.5, обробка TWOPASS конфігураційних файлів LinuxCNC сприяє їх модуляризації та читабельності. Нагадаємо, що конфігураційні файли LinuxCNC вказуються у файлі INI LinuxCNC як `[HAL]HALFILE=filename`.

Зазвичай набір одного або декількох конфігураційних файлів LinuxCNC повинен використовувати єдиний рядок `loadrt` для завантаження компонента реального часу, який може створювати кілька екземплярів компонента. Наприклад, якщо ви використовуєте компонент AND-елемента з двома входами (and2) у трьох різних місцях вашої конфігурації, вам потрібно буде десь вказати один рядок:

Приклад, що призводить до створення компонентів реального часу з іменами за замовчуван and2.0, and2.1, and2.2.

```
loadrt and2 count=3
```

Конфігурації легше читаються, якщо вказати опцію `names=` для компонентів, де вона підтримується, наприклад:

Приклад команди завантаження, що призводить до явно іменованих компонентів aa, ab, ac.

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

Відстеження компонентів та їхніх назв може бути проблемою з точки зору обслуговування, оскільки при додаванні (або видаленні) компонента необхідно знайти та оновити єдину директиву loadrt, що застосовується до цього компонента.

Обробка TWOPASS активується шляхом включення параметра INI-файлу в розділ [HAL], де "anystring" може бути будь-яким ненульовим рядком.

```
[HAL]

TWOPASS = anystring
```

З увімкненим TWOPASS ви можете мати кілька специфікацій, таких як:

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

Ці команди можуть з'являтися в різних HAL-файлах. HAL-файли обробляються в порядку їх появи в INI-файлі, у кількох призначеннях HALFILE.

Опцію TWOPASS можна вказати разом із опціями для додавання виводу для налагодження (verbose) та для запобігання видаленню тимчасових файлів (nodelete). Опції розділяються комами.

Приклад

```
[HAL]
TWOPASS = on,verbose,nodelete
```

При обробці TWOPASS спочатку зчитуються всі файли [HAL]HALFILES і накопичуються всі входження директив loadrt для кожного модуля. Компоненти, що не працюють у режимі реального часу (loadusr), завантажуються послідовно, але в початковому проході не виконуються інші команди LinuxCNC.

Note
Компоненти, що не працюють у реальному часі, повинні використовувати параметр очікування (-W), щоб переконатися, що компонент готовий до виконання інших команд.

Після початкового проходу модулі реального часу завантажуються (loadrt) автоматично

- з числом, що дорівнює загальній кількості, при використанні опції *count*= або
- з усіма окремими іменами, вказаними під час використання опції 'names=".

Потім виконується другий прохід для виконання всіх інших інструкцій LinuxCNC, зазначених у HALFILES. Команди addf, які пов'язують функції компонента з виконанням потоку, виконуються в порядку появи з іншими командами під час цього другого проходу.

Хоча ви можете використовувати параметри *count*= або *names*=, вони взаємовиключні — для даного модуля можна вказати лише один тип.

Обробка TWOPASS є найбільш ефективною при використанні опції «names=». Ця опція дозволяє надавати унікальні імена, які є мнемонічними або іншим чином пов'язані з конфігурацією. Наприклад, якщо ви використовуєте похідний компонент для оцінки швидкостей і прискорень на кожній координаті (x, y, z), використання методу «count=» дасть незрозумілі імена компонентів, такі як ddt.0, ddt.1, ddt.2 тощо.

Або ж можна скористатися опцією *names*=", наприклад:

```
loadrt ddt names=xvel,yvel,zvel
...
loadrt ddt names=xaccel,yaccel,zaccel
```

призводить до компонентів із доречними назвами xvel, yvel, zvel, xaccel, yaccel, zaccel.

Багато компіляторів, що постачаються з дистрибутивом, створені за допомогою утиліти halcompile і підтримують опцію *names*=. До них відносяться загальні логічні компоненти, які є сполучною ланкою багатьох конфігурацій LinuxCNC.

Створені користувачами компіляції, які використовують утиліту halcompile, також автоматично підтримують опцію names=. Окрім компіляцій, створених за допомогою утиліти halcompile, опцію names= підтримують також багато інших компіляцій. До компіляцій, що підтримують опцію names=, належать: at_pid, encoder, encoder_ratio, pid, siggen та sim_encoder.

Двоступенева обробка відбувається до завантаження графічного інтерфейсу користувача. При використанні [HAL]POSTGUI_HALFILE зручно розмістити всі декларації [HAL]POSTGUI_HALFILE loadrt для необхідних компонентів у попередньо завантаженому файлі HAL.

Приклад розділу HAL при використанні 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 <- b''pb''b''яв''b''дв''b''кb''b''ив'' loadrt b''дв''b''лв'' ↩
    b''яв'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нв''b''eb''b''нв''b''тb''b''ib''b''вв'' b' ↩
    'yb'' postgui.hal

POSTGUI_HALFILE = postgui.hal
HALUI = halui
```

5.3.2 Графічний інтерфейс користувача після публікації

Деякі графічні інтерфейси підтримують файли HAL, які обробляються після запуску графічного інтерфейсу для підключення контактів LinuxCNC, створених графічним інтерфейсом. При використанні файлу HAL postgui з обробкою TWOPASS включіть всі елементи loadrt для компонентів, доданих

файлами HAL postgui, в окремий файл HAL, який обробляється перед графічним інтерфейсом. Команди `addf` також можуть бути включені в файл.

Приклад

```
[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 Виключення файлів .hal

Обробка TWOPASS перетворює файли «.hal» в еквівалентні файли «.tcl» і використовує haltcl для пошуку команд loadrt і addf з метою накопичення та консолідації їх використання. Очікуються параметри Loadrt, які відповідають простим параметрам «names=» (або «count=»), що приймаються генератором компонентів HAL («halcompile»). Більш складні параметри, що містяться в спеціалізовани компонентах LinuxCNC, можуть оброблятися некоректно.

Файл «.hal» можна виключити з обробки TWOPASS, включивши магічну коментарну рядок в будь-якому місці файлу «.hal». Магічна коментарна рядок повинна починатися з рядка: `#NOTWOPASS`. Файли, вказані з цією магічною коментарною рядком, обробляються halcmd з використанням опцій `-k` (продовжувати при помилці) і `-v` (детальний вивід).

Це положення про виключення може бути використане для ізоляції проблем або для завантаження будь-якого спеціального компонента LinuxCNC, який не потребує або не отримує переваг від обробки TWOPASS.

Зазвичай, порядок loadrt компонентів реального часу не є критичним, але порядок loadrt для спеціальних компонентів можна забезпечити, помістивши такі директиви loadrt у виключений файл.

Note
Хоча порядок директив loadrt зазвичай не є критичним, порядок директив addf часто дуже важливий для належної роботи компонентів сервоциклу.

Приклад виключеного HAL-файлу

```
$ cat twopass_excluded.hal
# b''Hb''b''ab''b''cb''b''тb''b''yb''b''пb''b''нb''b''иb''b''йb'' b''мb''b''ab''b''гb''b' ↩
    'ib''b''чb''b''нb''b''иb''b''йb'' b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b' ↩
    'pb'' b''пb''b''pb''b''иb''b''зb''b''вb''b''ob''b''дb''b''иb''b''тb''b''ьb'' b''дb''b' ↩
    'ob'' b''тb''b''ob''b''гb''b''ob'', b''щb''b''ob'' b''цb''b''eb''b''йb'' b''фb''b''ab''b ↩
    ''йb''b''лb''
# b''бb''b''yb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''лb''b''юb''b''чb''b''eb''b''нb''b' ↩
    'ob'' b''зb'' b''дb''b''вb''b''ob''b''пb''b''pb''b''ob''b''хb''b''ib''b''дb''b''нb''b' ↩
    'ob''b''їb'' b''ob''b''бb''b''pb''b''ob''b''бb''b''кb''b''иb'':
# NOTWOPASS

# b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'' b''нb''b''ab''b''лb''b' ↩
    'ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''ib'' b''cb''b' ↩
    'кb''b''лb''b''ab''b''дb''b''нb''b''иb''b''мb''b''иb'' b''ob''b''пb''b''цb''b''ib''b' ↩
    'яb''b''мb''b''иb'':
loadrt mycomponent parm1=”abc def” parm2=ghi
show pin mycomponent
```

```
# b''зb''b''ab''b''мb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''cb''b''пb''b' ↩
    'eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''иb''b''xb'' b''кb''b''ob''b''мb''b' ↩
    'пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ib''b''вb''
loadrt component_1
loadrt component_2
```

Note
Регістр та пробіли в магічному коментарі ігноруються. Завантаження компонентів, які використовують параметри names= або count= (зазвичай створюються за допомогою halcompile), не повинно використовуватися у виключених файлах, оскільки це усуне переваги обробки TWOPASS. Команди LinuxCNC, які створюють сигнали (net), та команди, які встановлюють порядок виконання (addf), не повинні розміщуватися у виключених файлах. Це особливо стосується команд addf, оскільки їх порядок може бути важливим.

5.3.4 Приклади

Приклади використання TWOPASS для симулятора наведено в каталогах:

```
configs/sim/axis/twopass/
configs/sim/axis/simtcl/
```

5.4 Підручник з HAL

5.4.1 Вступ

Конфігурація переходить від теорії до пристрою – тобто пристрою HAL. Для тих, хто хоч трохи знайомий з комп'ютерним програмуванням, цей розділ – це «Привіт, світе» HAL.

halrun може бути використаний для створення робочої системи. Це інструмент командного рядка або текстового файлу для конфігурації та налаштування. Наведені нижче приклади ілюструють його налаштування та роботу.

5.4.2 Halcmd

halcmd — це інструмент командного рядка для роботи з HAL. Більш повна сторінка довідки для halcmd існує і встановлюється разом з LinuxCNC, з вихідного коду або з пакета. Якщо LinuxCNC було скомпільовано як *run-in-place*, сторінка довідки не встановлюється, але доступна в головному каталозі LinuxCNC за допомогою наступної команди:

```
$ man -M docs/man halcmd
```

5.4.2.1 Нотація

У цьому підручнику команди для операційної системи зазвичай показані без підказки, що надається оболонкою UNIX, тобто зазвичай знака долара ($) або знака решітки/подвійного хрестика (#). При безпосередньому спілкуванні з HAL через halcmd або halrun підказки показані в прикладах. Вікно терміналу знаходиться в розділі «Програми/Аксесуари» головної панелі меню Ubuntu.

Приклад команди терміналу - підказки

```
me@computer:~linuxcnc$ halrun
(b''бb''b''yb''b''дb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b''ab''b''нb''b''ob'' b' ↩
    'яb''b''кb'' b''yb'' b''нb''b''ab''b''cb''b''тb''b''yb''b''пb''b''нb''b''ob''b''мb''b' ↩
    'yb'' b''pb''b''яb''b''дb''b''кb''b''yb'')
halrun

(b''зb''b''ab''b''пb''b''иb''b''тb'' halcmd: b''бb''b''yb''b''дb''b''eb'' b''пb''b''ob''b' ↩
    'кb''b''ab''b''зb''b''ab''b''нb''b''ob'' b''пb''b''ib''b''дb'' b''чb''b''ab''b''cb'' b' ↩
    'зb''b''ab''b''пb''b''yb''b''cb''b''кb''b''yb'' HAL)
halcmd: loadrt counter
halcmd: show pin
```

5.4.2.2 Автозаповнення за допомогою клавіші Tab

Ваша версія `halcmd` може підтримувати автозавершення за допомогою клавіші Tab. Замість завершення імен файлів, як це робить оболонка, вона завершує команди за допомогою ідентифікаторів HAL. Вам потрібно буде ввести достатню кількість літер для унікального збігу. Спробуйте натиснути клавішу Tab після запуску команди HAL:

Автозаповнення за допомогою клавіші Tab

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

5.4.2.3 Середовище RTAPI

RTAPI означає «інтерфейс програмування додатків у реальному часі». Багато компонентів HAL працюють у реальному часі, і всі компоненти HAL зберігають дані у спільній пам'яті, щоб компоненти реального часу могли отримати до них доступ. Звичайна Linux не підтримує програмування у реальному часі або тип спільної пам'яті, який потрібен HAL. На щастя, існують операційні системи реального часу (RTOS), які надають необхідні розширення для Linux. На жаль, кожна RTOS працює трохи по-різному.

Щоб вирішити ці проблеми, команда LinuxCNC розробила RTAPI, який забезпечує єдиний спосіб взаємодії програм з RTOS. Якщо ви програміст, який хоче працювати над внутрішніми компонентами LinuxCNC, вам може бути корисно вивчити файл «linuxcnc/src/rtapi/rtapi.h», щоб зрозуміти API. Але якщо ви звичайна людина, все, що вам потрібно знати про RTAPI, це те, що він (і RTOS) повинен бути завантажений в пам'ять вашого комп'ютера, перш ніж ви почнете працювати з HAL.

5.4.3 Простий приклад

5.4.3.1 Завантаження компонента

Для цього підручника ми будемо вважати, що ви успішно встановили Live CD і, якщо використовуєте RIP [1], запустіть скрипт «rip-environment» для підготовки вашої оболонки. У цьому випадку все, що вам потрібно зробити, це завантажити необхідні модулі RTOS і RTAPI в пам'ять. Просто виконайте наступну команду з терміналу:

Завантаження HAL

[1] Run In Place, коли вихідні файли завантажені в каталог користувача і компілюються та виконуються безпосередньо звідти.

```
cd linuxcnc
halrun
halcmd:
```

Завантаживши ОС реального часу та RTAPI, ми можемо перейти до першого прикладу. Зверніть увагу, що тепер командний рядок відображається як «halcmd:». Це пов’язано з тим, що наступні команди будуть інтерпретуватися як команди HAL, а не команди оболонки.

Для першого прикладу ми будемо використовувати компонент HAL під назвою «siggen», який є простим генератором сигналів. Повний опис компонента «siggen» можна знайти в розділі SigGen цього посібника. Це компонент реального часу. Щоб завантажити компонент «`siggen`», використовуйте команду HAL `loadrt`.

Завантаження знаку

```
halcmd: loadrt siggen
```

5.4.3.2 Вивчення HAL

Тепер, коли модуль завантажено, настав час представити `halcmd`, інструмент командного рядка, що використовується для налаштування HAL. У цьому підручнику буде представлено лише деякі функції halcmd. Більш повний опис можна знайти в `man halcmd` або в розділі HAL Commands цього документа. Першою функцією halcmd є команда «show». Ця команда відображає інформацію про поточний стан HAL. Щоб показати всі встановлені компоненти:

Показати компоненти з halrun/halcmd

```
halcmd: show comp

    b''Зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''ib'' b''кb''b' ↩
        'ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''иb'' HAL:
    ID     Type  Name                                PID   State
    3      RT    siggen                                    ready
    2      User  halcmd2177                          2177  ready
```

Оскільки *halcmd* сам по собі також є компонентом HAL, він завжди буде відображатися у списку. Число після «`halcmd`» у списку компонентів є ідентифікатором процесу UNIX. Можна запустити декілька копій halcmd одночасно (наприклад, у різних вікнах терміналу), тому PID додається в кінець імені, щоб зробити його унікальним. У списку також відображається компонент «siggen», який ми встановили на попередньому кроці. «RT» під «Тип» вказує, що «siggen» є компонентом реального часу. «Користувач» під «Тип» вказує, що це компонент нереального часу.

Далі, давайте подивимося, які піни пропонує `siggen`:

Показати піни

```
halcmd: show pin

b''Bb''b''иb''b''вb''b''ob''b''дb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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
```

Ця команда відображає всі контакти в поточному HAL. Складна система може мати десятки або сотні контактів. Але зараз є тільки дев’ять контактів. З них вісім є плаваючими, а один — бітовим (булевим). Шість передають дані з компонента «siggen», а три використовуються для передачі налаштувань у компонент. Оскільки ми ще не виконали код, що міститься в компоненті, деякі контакти мають значення нуль.

Наступний крок – це перевірка параметрів:

Показати параметри

```
halcmd: show param

b''Пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'':
Owner   Type  Dir         Value   Name
     3  s32   RO              0   siggen.0.update.time
     3  s32   RW              0   siggen.0.update.tmax
```

Команда «show param» показує всі параметри в HAL. На даний момент кожен параметр має значення за замовчуванням, яке було задано під час завантаження компонента. Зверніть увагу на стовпець із назвою «Dir». Параметри з позначкою «-W» є параметрами, що записуються, які ніколи не змінюються самим компонентом, а призначені для зміни користувачем з метою керування компонентом. Пізніше ми побачимо, як це зробити. Параметри з позначкою «R-» є параметрами тільки для читання. Вони можуть бути змінені тільки компонентом. Нарешті, параметри з позначкою «RW» є параметрами для читання і запису. Це означає, що вони змінюються компонентом, але також можуть бути змінені користувачем. Примітка: Параметри `siggen.0.update.t` і `siggen.0.update.tmax` призначені для налагодження і не будуть розглядатися в цьому розділі.

Більшість компонентів реального часу експортують одну або декілька функцій для фактичного запуску коду реального часу, який вони містять. Давайте подивимося, які функції експортував «siggen»:

Показати функції за допомогою halcmd`

```
halcmd: show funct

b''Eb''b''кb''b''cb''b''пb''b''ob''b''pb''b''тb''b''ob''b''вb''b''ab''b''нb''b''ib'' b' ↩
    'фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb'':
Owner   CodeAddr  Arg       FP   Users  Name
00003   f801b000  fae820b8  YES      0  siggen.0.update
```

Компонент siggen експортував одну функцію. Він вимагає плаваючої точки. Наразі він не пов’язаний з жодним потоком, тому *users* дорівнює нулю. Примітка: [Поля CodeAddr та Arg використовувалися під час розробки і, ймовірно, мають зникнути.].

5.4.3.3 Запуск коду в реальному часі

Щоб фактично запустити код, що міститься у функції `siggen.0.update`, нам потрібен потік реального часу. Компонент під назвою «threads» використовується для створення нового потоку. Створимо потік під назвою «test-thread» з періодом 1 мс (1000 мкс або 1000000 нс):

```
halcmd: loadrt threads name1=test-thread period1=1000000
```

Давайте подивимося, чи це спрацювало:

Показати теми

```
halcmd: show thread

b''Tb''b''eb''b''мb''b''иb'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'мb''b''yb'' b''чb''b''ab''b''cb''b''ib'':
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
```

Так, вийшло. Період не дорівнює точно 1000000 нс через обмеження апаратного забезпечення, але ми маємо потік, який працює з приблизно правильною швидкістю і може обробляти функції з плаваючою комою. Наступним кроком є підключення функції до потоку:

Додати функцію

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

До цього часу ми використовували `halcmd` тільки для перегляду HAL. Однак цього разу ми використали команду `addf` (додати функцію), щоб фактично змінити щось у HAL. Ми наказали *halcmd* додати функцію `siggen.0.update` до потоку *test-thread*, і якщо ми знову подивимося на список потоків, то побачимо, що це вдалося:

```
halcmd: show thread

b''Tb''b''eb''b''мb''b''иb'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'мb''b''yb'' b''чb''b''ab''b''cb''b''ib'':
     Period  FP     Name               (     Time, Max-Time )
     999855  YES    test-thread        (        0,        0 )
                 1 siggen.0.update
```

Перед тим, як компонент «siggen» почне генерувати сигнали, потрібно виконати ще один крок. При першому запуску HAL потоки фактично не працюють. Це дозволяє повністю налаштувати систему перед запуском коду реального часу. Коли ви будете задоволені налаштуваннями, можете запустити код реального часу таким чином:

```
halcmd: start
```

Тепер генератор сигналів працює. Давайте розглянемо його вихідні контакти:

```
halcmd: show pin

b''Bb''b''иb''b''вb''b''ob''b''дb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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
```

І давайте ще раз подивимося:

```
halcmd: show pin

b''Bb''b''иb''b''вb''b''ob''b''дb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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
```

Ми швидко послідовно виконали дві команди «show pin», і ви можете бачити, що вихідні дані більше не дорівнюють нулю. Вихідні дані синуса, косинуса, пилкоподібної хвилі та трикутника постійно змінюються. Вихідні дані квадратної хвилі також працюють, однак вони просто перемикаються з +1,0 на -1,0 кожного циклу.

5.4.3.4 Зміна параметрів

Справжня сила HAL полягає в тому, що ви можете змінювати значення. Наприклад, ми можемо використовувати команду `setp` для встановлення значення параметра. Давайте змінимо амплітуду генератора сигналів з 1,0 до 5,0:

Встановити PIN-код

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

Ще раз перевірте параметри та контакти

```
halcmd: show param

b''Пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'':
Owner   Type  Dir         Value  Name
     3  s32   RO           1754  siggen.0.update.time
     3  s32   RW          16997  siggen.0.update.tmax

halcmd: show pin

b''Bb''b''иb''b''вb''b''ob''b''дb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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
```

Зверніть увагу, що значення параметра `siggen.0.amplitude` змінилося на 5, і що тепер значення виводів більші.

5.4.3.5 Збереження конфігурації HAL

Більшість того, що ми робили з `halcmd` до цього моменту, полягало просто в перегляді елементів за допомогою команди «show». Однак дві з команд фактично змінили ситуацію. У міру того, як ми будемо проектувати більш складні системи за допомогою HAL, ми будемо використовувати багато команд для налаштування елементів саме так, як нам потрібно. HAL має пам'ять слона і збереже цю конфігурацію, поки ми її не вимкнемо. Але що буде наступного разу? Ми не хочемо вручну вводити купу команд щоразу, коли хочемо використовувати систему.

Збереження конфігурації всього HAL однією командою.

```
halcmd: save

# b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''иb''
loadrt threads name1=test-thread period1=1000000
loadrt siggen
```

```
# b''пb''b''cb''b''eb''b''вb''b''дb''b''ob''b''нb''b''ib''b''мb''b''иb'' b''вb''b''иb''b' ↩
    'вb''b''ob''b''дb''b''ib''b''вb''
# b''cb''b''иb''b''гb''b''нb''b''ab''b''лb''b''иb''
# b''мb''b''eb''b''pb''b''eb''b''жb''b''ib''
# b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'мb''b''eb''b''тb''b''pb''b''ib''b''вb''
setp siggen.0.update.tmax 14687
# b''пb''b''ob''b''cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b''пb''b' ↩
    'ob''b''тb''b''ob''b''кb''b''иb''/b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb'' b' ↩
    'вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b''yb'' b''чb''b''ab''b' ↩
    'cb''b''ib''
addf siggen.0.update test-thread
```

Результатом виконання команди save є послідовність команд HAL. Якщо ви почнете з порожнього HAL і виконаєте всі ці команди, ви отримаєте конфігурацію, яка існувала на момент виконання команди «save». Щоб зберегти ці команди для подальшого використання, ми просто перенаправимо вихідні дані у файл:

Збережіть конфігурацію у файл за допомогою halcmd

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

5.4.3.6 Вихід з Халруна

Коли ви закінчите сеанс HAL, введіть exit у командному рядку "halcmd:". Це поверне вас до системного командного рядка і закриє сеанс HAL. Не закривайте вікно терміналу, не завершивши сеанс HAL.

Вихід з HAL

```
halcmd: exit
```

5.4.3.7 Відновлення конфігурації HAL

Щоб відновити конфігурацію HAL, збережену у файлі «saved.hal», нам потрібно виконати всі ці команди HAL. Для цього ми використовуємо «-f _<ім'я файлу>_», яке зчитує команди з файлу, та «-I» (велика літера i), яке показує командний рядок halcmd після виконання команд:

Запуск збереженого файлу

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

Зверніть увагу, що у файлі saved.hal немає команди "start". Потрібно виконати її ще раз (або відредагувати файл saved.hal, щоб додати її туди).

5.4.3.8 Вилучення HAL з пам'яті

Якщо відбувається неочікуване завершення сеансу HAL, можливо, доведеться вивантажити HAL, перш ніж зможе розпочатися інший сеанс. Для цього введіть таку команду у вікні терміналу.

Вилучення HAL

```
halrun -U
```

5.4.4 Півметра

Ви можете створювати дуже складні системи HAL, навіть не використовуючи графічний інтерфейс. Однак є щось приємне в тому, щоб бачити результат своєї роботи. Першим і найпростішим інструментом GUI для HAL є halmeter. Це дуже проста програма, яка є еквівалентом HAL зручного мультиметра (або аналогового вимірювача для старої гвардії).

Це дозволяє спостерігати за контактами, сигналами або параметрами, відображаючи поточне значення цих сутностей. Це дуже простий у використанні додаток для графічних середовищ. У консольному типі:

```
halmeter
```

З'являться два вікна. Вікно вибору є найбільшим і містить три вкладки:

- В одному перераховані всі виводи, що наразі визначені в HAL,
- один перераховує всі сигнали,
- в одному перераховані всі параметри.

Клацніть на вкладку, а потім на один з елементів, щоб вибрати його. У невеликому вікні буде показано назву та значення вибраного елемента. Відображення оновлюється приблизно 10 разів на секунду. Щоб звільнити місце на екрані, вікно вибору можна закрити за допомогою кнопки *Закрити*. У невеликому вікні, прихованому під вікном вибору під час запуску програми, кнопка *Вибрати* знову відкриває вікно вибору, а кнопка *Вийти* зупиняє програму та закриває обидва вікна.

Можна запустити кілька halmeter одночасно, що дозволяє візуалізувати кілька елементів одночасно. Щоб відкрити halmeter і звільнити консоль, запустивши його у фоновому режимі, виконайте наступну команду:

```
halmeter &
```

Можна запустити halmeter і змусити його відразу відобразити елемент. Для цього додайте аргументи *pin|sig|par[am] name* у командному рядку. Він відобразить сигнал, контакт або параметр *name* відразу після запуску. Якщо вказаний елемент не існує, він запуститься у звичайному режимі.

Нарешті, якщо елемент призначений для відображення, можна додати *-s* перед pin|sig|param, щоб halmeter використовував ще менше вікно. Назва елемента буде відображатися в рядку заголовка, а не під значенням, і кнопка не буде відображатися. Це корисно для відображення великої кількості halmeterів на невеликому просторі.

Ми знову використаємо компонент siggen, щоб перевірити halmeter. Якщо ви щойно завершили попередній приклад, то ви можете завантажити siggen, використовуючи збережений файл. Якщо ні, ми можемо завантажити його так само, як ми робили раніше:

```
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 цьому етапі компонент siggen завантажено та працює. Час запустити halmeter.

Початковий півметра

```
halcmd: loadusr halmeter
```

Перше вікно, яке ви побачите, це вікно «Вибір елемента для дослідження».

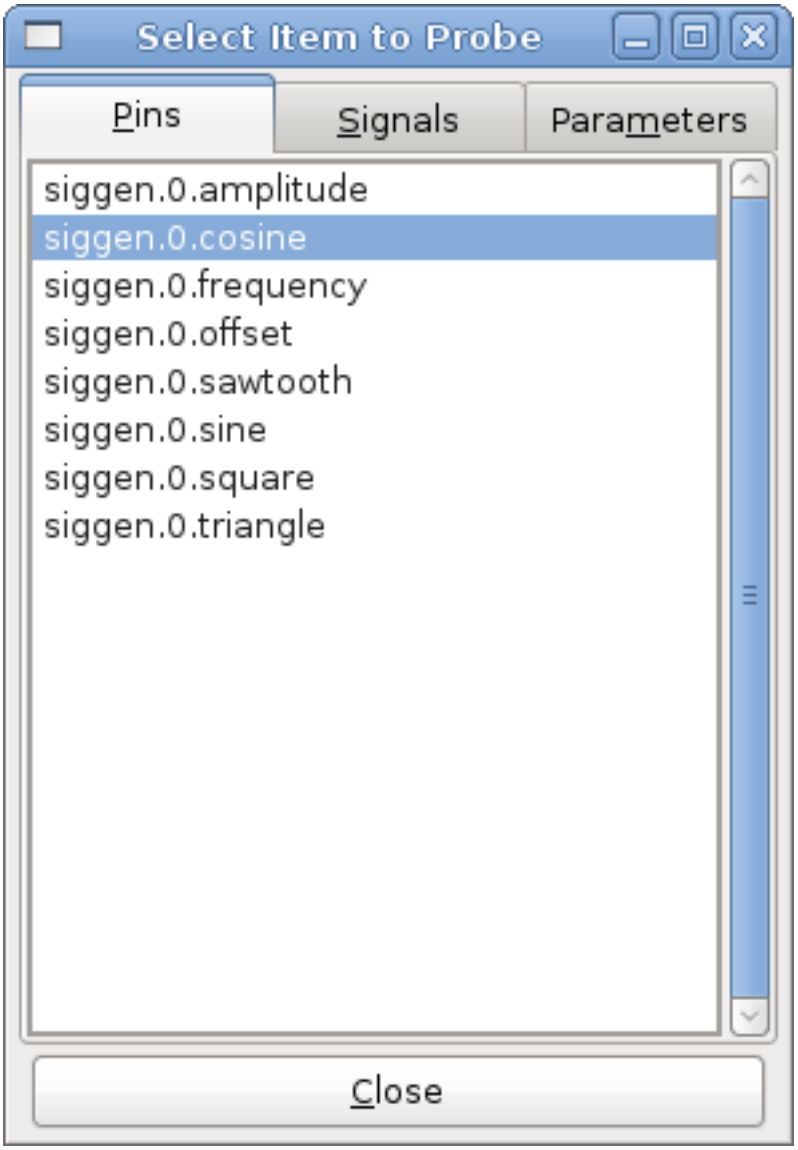


Figure 5.5: Вікно вибору Halmeter

Це діалогове вікно має три вкладки. Перша вкладка відображає всі контакти HAL в системі. Друга вкладка відображає всі сигнали, а третя — всі параметри. Спочатку ми хочемо розглянути контакт `siggen.0.cosine`, тому клацніть на ньому, а потім натисніть кнопку «Закрити». Діалогове вікно вибору зонда закриється, а вимірювач буде виглядати приблизно так, як показано на малюнку нижче.



Figure 5.6: Півметрове вікно

Щоб змінити відображення на лічильнику, натисніть кнопку «Вибрати», яка поверне вікно «Вибір елемента для вимірювання».

Ви повинні побачити зміну значення, коли siggen генерує свою косинусоподібну хвилю. Halmeter оновлює свій дисплей приблизно 5 разів на секунду.

Щоб вимкнути Halmeter, просто натисніть кнопку виходу.

Якщо ви хочете переглянути більше ніж один контакт, сигнал або параметр одночасно, ви можете просто запустити більше халметрів. Вікно халметра було навмисно зроблено дуже маленьким, щоб ви могли мати багато з них на екрані одночасно.

5.4.5 Приклад Stepgen

До цього моменту ми завантажили лише один компонент HAL. Але основна ідея HAL полягає в тому, щоб дозволити вам завантажувати та підключати низку простих компонентів для створення складної системи. У наступному прикладі буде використано два компоненти.

Перш ніж приступити до створення цього нового прикладу, ми хочемо почати з чистого аркуша. Якщо ви щойно завершили один із попередніх прикладів, нам потрібно видалити всі компоненти та перезавантажити бібліотеки RTAPI та HAL.

```
halcmd: exit
```

5.4.5.1 Встановлення компонентів

Тепер ми завантажимо компонент генератора імпульсів. Детальний опис цього компонента дивіться в розділі stepgen посібника Integrator Manual. У цьому прикладі ми будемо використовувати тип управління StepGen «velocity». Наразі ми можемо пропустити деталі і просто виконати наступні команди.

У цьому прикладі ми використовуватимемо тип керування *velocity* з компонента `stepgen`.

```
halrun
halcmd: loadrt stepgen step_type=0,0 ctrl_type=v,v
halcmd: loadrt siggen
halcmd: loadrt threads name1=fast fp1=0 period1=50000 name2=slow period2=1000000
```

Перша команда завантажує два генератори кроків, обидва налаштовані на генерацію кроків типу 0. Друга команда завантажує нашого старого знайомого siggen, а третя створює два потоки: швидкий з періодом 50 мікросекунд (мкс) і повільний з періодом 1 мілісекунди (мс). Швидкий потік не підтримує функції з плаваючою комою.

Як і раніше, ми можемо використовувати `halcmd show`, щоб переглянути HAL. Цього разу у нас набагато більше виводів та параметрів, ніж раніше:

```
halcmd: show pin

b''Bb''b''ив''b''вb''b''ob''b''дb''b''ив'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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

b''Пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ив'':
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 З’єднання контактів із сигналами

У нас є двоступеневі генератори імпульсів і генератор сигналів. Тепер настав час створити кілька сигналів HAL для з’єднання двох компонентів. Ми будемо вважати, що двоступеневі генератори імпульсів керують осями X і Y машини. Ми хочемо переміщати стіл по колу. Для цього ми будемо надсилати косинусний сигнал на вісь X і синусний сигнал на вісь Y. Модуль siggen створює синус і косинус, але нам потрібні «проводи», щоб з’єднати модулі між собою. У HAL «проводи» називаються сигналами. Нам потрібно створити два таких сигнали. Ми можемо назвати їх як завгодно, для цього прикладу це будуть «X-vel» і «Y-vel». Сигнал «X-vel» призначений для передачі від косинусного виходу генератора сигналів до входу швидкості першого імпульсного генератора. Першим кроком є підключення сигналу до виходу генератора сигналів. Для підключення сигналу до виводу ми використовуємо команду net.

мережева команда

```
halcmd: net X-vel <= siggen.0.cosine
```

Щоб побачити ефект команди `net`, ми знову покажемо сигнали.

```
halcmd: show sig

b''Cb''b''ив''b''гb''b''нв''b''ab''b''лв''b''ив'':
Type          Value  Name     (linked to)
float             0  X-vel <== siggen.0.cosine
```

Коли сигнал підключений до одного або декількох виводів, команда show відображає список виводів, що йдуть відразу за назвою сигналу. «Стрілка» показує напрямок потоку даних — в даному випадку дані надходять від виводу `siggen.0.cosine` до сигналу `X-vel`. Тепер підключимо `X-vel` до входу швидкості генератора імпульсів.

```
halcmd: net X-vel => stepgen.0.velocity-cmd
```

Ми також можемо підключити сигнал осі Y `Y-vel`. Він призначений для передачі від синусоїдального виходу генератора сигналів до входу другого генератора імпульсів кроку. Наступна команда виконує в одному рядку те, що дві команди `net` виконували для `X-vel`.

```
halcmd: net Y-vel siggen.0.sine => stepgen.1.velocity-cmd
```

Тепер давайте остаточно розглянемо сигнали та підключені до них контакти.

```
halcmd: show sig

b''Cb''b''ив''b''гb''b''нв''b''ab''b''лв''b''ив'':
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» чітко показує, як саме дані проходять через HAL. Наприклад, сигнал «X-vel» надходить з виводу `siggen.0.cosine` і переходить на вивод `stepgen.0.velocity-cmd`.

5.4.5.3 Налаштування виконання в реальному часі - потоки та функції

Якщо уявити собі дані, що протікають по «дротах», то зрозуміти, що таке контакти і сигнали, досить просто. Трохи складніше з потоками і функціями. Функції містять комп’ютерні інструкції, які фактично виконують завдання. Потоки — це метод, за допомогою якого ці інструкції виконуються, коли це необхідно. Спочатку давайте розглянемо функції, які нам доступні.

```
halcmd: show funct

b''Eb''b''кb''b''cb''b''пb''b''ob''b''pb''b''тb''b''ob''b''вb''b''ab''b''нb''b''ib'' b' ↩
    'фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb'':
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
```

Як правило, вам доведеться звертатися до документації по кожному компоненту, щоб дізнатися, що роблять його функції. У даному випадку функція `siggen.0.update` використовується для оновлення вихідних даних генератора сигналів. Кожного разу, коли вона виконується, вона обчислює значення синусоїдальних, косинусоїдальних, трикутних і квадратних вихідних даних. Щоб отримати плавні сигнали, вона повинна працювати з певними інтервалами.

Інші три функції пов'язані з генераторами крокових імпульсів.

Перший, `stepgen.capture_position`, використовується для зворотного зв'язку щодо положення. Він фіксує значення внутрішнього лічильника, який підраховує імпульси кроку в міру їх генерації. Припускаючи, що кроки не пропускаються, цей лічильник вказує положення двигуна.

Основною функцією генератора імпульсів кроку є `stepgen.make_pulses`. Кожного разу, коли запускається «make_pulses», вона вирішує, чи настав час зробити крок, і якщо так, то відповідно встановлює виходи. Для плавних імпульсів кроку вона повинна працювати якомога частіше. Оскільки вона повинна працювати дуже швидко, «make_pulses» є високо оптимізованою і виконує лише кілька обчислень. На відміну від інших, він не потребує математичних операцій з плаваючою комою.

Остання функція, `stepgen.update-freq`, відповідає за масштабування та деякі інші обчислення, які потрібно виконувати лише тоді, коли змінюється команда частоти.

Для нашого прикладу це означає, що ми хочемо запустити `siggen.0.update` з помірною швидкістю, щоб обчислити значення синуса і косинуса. Відразу після запуску `siggen.0.update` ми хочемо запустити `stepgen.update_freq`, щоб завантажити нові значення в генератор імпульсів. Нарешті, нам потрібно запустити `stepgen.make_pulses` якомога швидше для отримання рівномірних імпульсів. Оскільки ми не використовуємо зворотний зв'язок по положенню, нам взагалі не потрібно запускати `stepgen.capture_position`.

Ми запускаємо функції, додаючи їх до потоків. Кожен потік виконується з певною швидкістю. Давайте подивимося, які потоки у нас є.

```
halcmd: show thread

b''Tb''b''eb''b''мb''b''иb'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'мb''b''yb'' b''чb''b''ab''b''cb''b''ib'':
     Period  FP     Name               (     Time, Max-Time )
     996980  YES                  slow (        0,        0 )
      49849  NO                   fast (        0,        0 )
```

Два потоки були створені під час завантаження `threads`. Перший, «slow», працює кожну мілісекунду і здатний виконувати функції з плаваючою комою. Ми будемо використовувати його для `siggen.0.upda` і `stepgen.update_freq`. Другий потік — «fast», який працює кожні 50 мікросекунд (µs) і не підтримує плаваючу кому. Ми будемо використовувати його для `stepgen.make_pulses`. Щоб підключити функції до відповідного потоку, ми використовуємо команду `addf`. Спочатку ми вказуємо функцію, а потім потік.

```
halcmd: addf siggen.0.update slow
halcmd: addf stepgen.update-freq slow
halcmd: addf stepgen.make-pulses fast
```

Після того, як ми надамо ці команди, ми можемо знову виконати команду `show thread`, щоб побачити, що сталося.

```
halcmd: show thread

b''Tb''b''eb''b''мb''b''иb'' b''вb'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'мb''b''yb'' b''чb''b''ab''b''cb''b''ib'':
     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
```

Тепер за кожним потоком йдуть імена функцій у порядку, в якому ці функції будуть виконуватися.

5.4.5.4 Налаштування параметрів

Ми майже готові запустити нашу систему HAL. Однак нам ще потрібно налаштувати кілька параметрів. За замовчуванням компонент siggen генерує сигнали, які коливаються від +1 до -1. Для нашого прикладу це підходить, ми хочемо, щоб швидкість столу змінювалася від +1 до -1 дюйма в секунду. Однак масштабування генератора імпульсів кроку не зовсім правильне. За замовчуванням він генерує вихідну частоту 1 крок на секунду при вхідному сигналі 1,0. Малоймовірно, що один крок на секунду дасть нам один дюйм на секунду руху столу. Припустимо, що ми маємо гвинт з 5 обертами на дюйм, підключений до крокового двигуна з 200 кроками на оберт і 10-кратним мікрокрокуванням. Отже, для одного оберту гвинта потрібно 2000 кроків, а для переміщення на один дюйм — 5 обертів. Це означає, що загальне масштабування становить 10000 кроків на дюйм. Нам потрібно помножити швидкість, що вводиться в генератор імпульсів кроку, на 10000, щоб отримати правильний результат. Саме для цього і призначений параметр `stepgen.n.velocity-scale`. У цьому випадку осі X і Y мають однакове масштабування, тому ми встановлюємо параметри масштабування для обох 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
```

Це масштабування швидкості означає, що коли контакт `stepgen.0.velocity-cmd` дорівнює 1,0, генератор імпульсів буде генерувати 10000 імпульсів на секунду (10 кГц). З описаними вище двигуном і ходовим гвинтом це призведе до руху осі зі швидкістю рівно 1,0 дюйма на секунду. Це ілюструє ключову концепцію HAL — такі речі, як масштабування, виконуються на найнижчому можливому рівні, в даному випадку в генераторі імпульсів кроку. Внутрішній сигнал `X-vel` — це швидкість столу в дюймах на секунду, а інші компоненти, такі як `siggen`, взагалі не знають (і не дбають) про масштабування. Якщо ми змінимо ходовий гвинт або двигун, ми змінимо тільки параметр масштабування генератора імпульсів кроку.

5.4.5.5 Запустіть це!

Тепер у нас все налаштовано, і ми готові до запуску. Як і в першому прикладі, ми використовуємо команду `start`.

```
halcmd: start
```

Хоча зовні нічого не відбувається, всередині комп'ютера генератор імпульсів видає імпульси, що змінюються від 10 кГц вперед до 10 кГц назад і назад кожну секунду. Пізніше в цьому підручнику ми побачимо, як вивести ці внутрішні сигнали для запуску двигунів у реальному світі, але спочатку ми хочемо подивитися на них і побачити, що відбувається.

5.4.6 Галскоп

Попередній приклад генерує кілька дуже цікавих сигналів. Але багато з того, що відбувається, відбувається надто швидко, щоб побачити це за допомогою halmeter. Щоб детальніше розглянути, що відбувається всередині HAL, нам знадобиться осцилограф. На щастя, HAL має такий осцилограф, який називається halscope.

Halscope складається з двох частин — частини, що працює в режимі реального часу і зчитує сигнали HAL, та частини, що не працює в режимі реального часу і забезпечує графічний інтерфейс користувача та відображення. Однак вам не потрібно про це турбуватися, оскільки частина, що не працює в режимі реального часу, автоматично завантажить частину, що працює в режимі реального часу, коли це буде потрібно.

Якщо LinuxCNC запущено в терміналі, ви можете запустити halscope за допомогою наступної команди.

Запуск Halscope

```
halcmd loadusr halscope
```

Якщо LinuxCNC не працює або файл autosave.halscope не відповідає контактам, доступним у поточній версії LinuxCNC, відкриється вікно графічного інтерфейсу, а відразу після цього з’явиться діалогове вікно «Функція реального часу не пов’язана», яке виглядає так, як показано на малюнку нижче. Щоб змінити частоту дискретизації, клацніть лівою кнопкою миші на полі «Зразки».

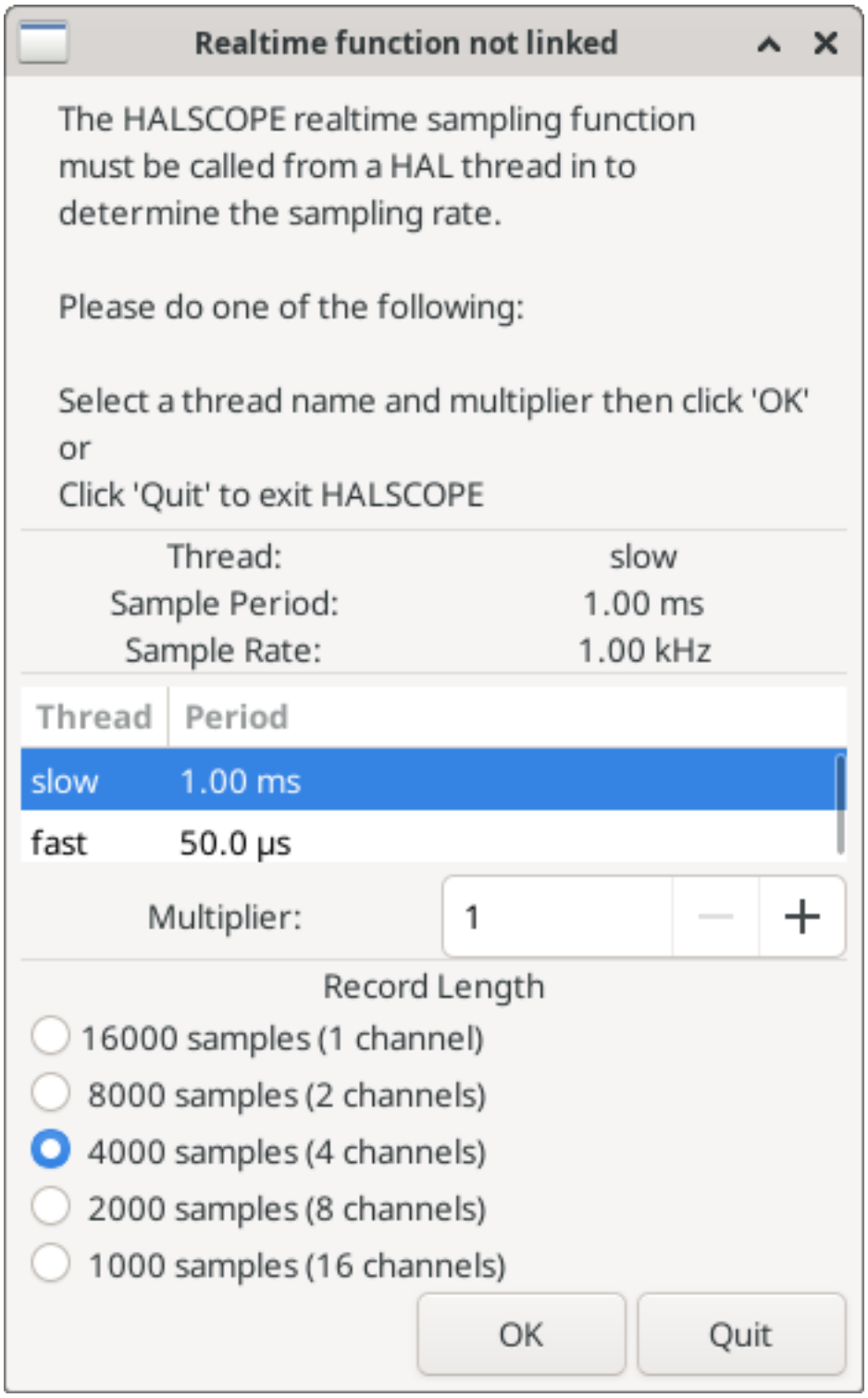


Figure 5.7: Діалогове вікно не пов’язане з функцією реального часу

У цьому діалоговому вікні ви можете встановити частоту дискретизації для осцилографа. Наразі ми хочемо дискретизувати один раз на мілісекунду, тому натисніть на нитку 1,00 мс «slow» і залиште множник на рівні 1. Ми також залишимо довжину запису на рівні 4000 дискретизацій, щоб ми могли використовувати до чотирьох каналів одночасно. Коли ви виберете рядок і натиснете «OK», діалогове вікно зникне, а вікно осцилографа буде виглядати приблизно так, як на малюнку

нижче.

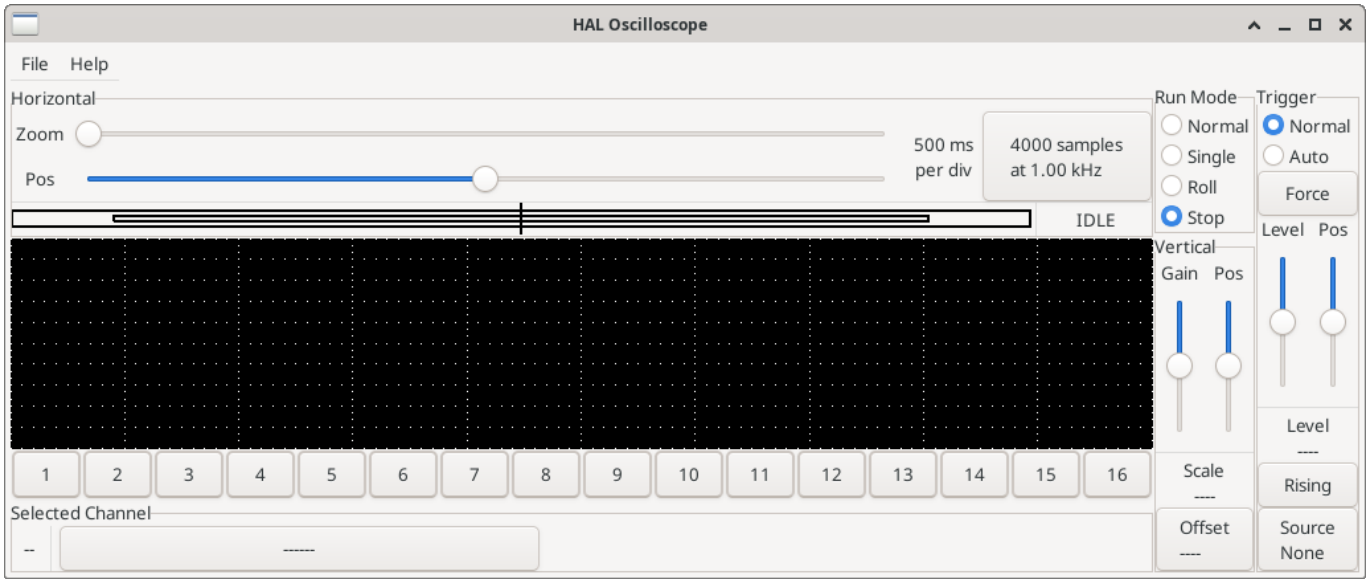


Figure 5.8: Початкове вікно області застосування

5.4.6.1 Підключення зондів осцилографа

На цьому етапі Halscope готовий до використання. Ми вже вибрали частоту дискретизації та довжину запису, тому наступним кроком є визначення того, що саме потрібно спостерігати. Це еквівалентно підключенню «віртуальних зондів осцилографа» до HAL. Halscope має 16 каналів, але кількість каналів, які можна використовувати одночасно, залежить від довжини запису — більше каналів означає коротші записи, оскільки обсяг пам'яті, доступний для запису, фіксований і становить приблизно 16 000 зразків.

Кнопки каналів розташовані внизу екрана halscope. Натисніть кнопку «1», і ви побачите діалогове вікно «Вибір джерела каналу», як показано на малюнку нижче. Це діалогове вікно дуже схоже на те, що використовується в Halmeter. Ми хочемо переглянути сигнали, які ми визначили раніше, тому натискаємо вкладку «Сигнали», і діалогове вікно відображає всі сигнали в HAL (у цьому прикладі їх тільки два).

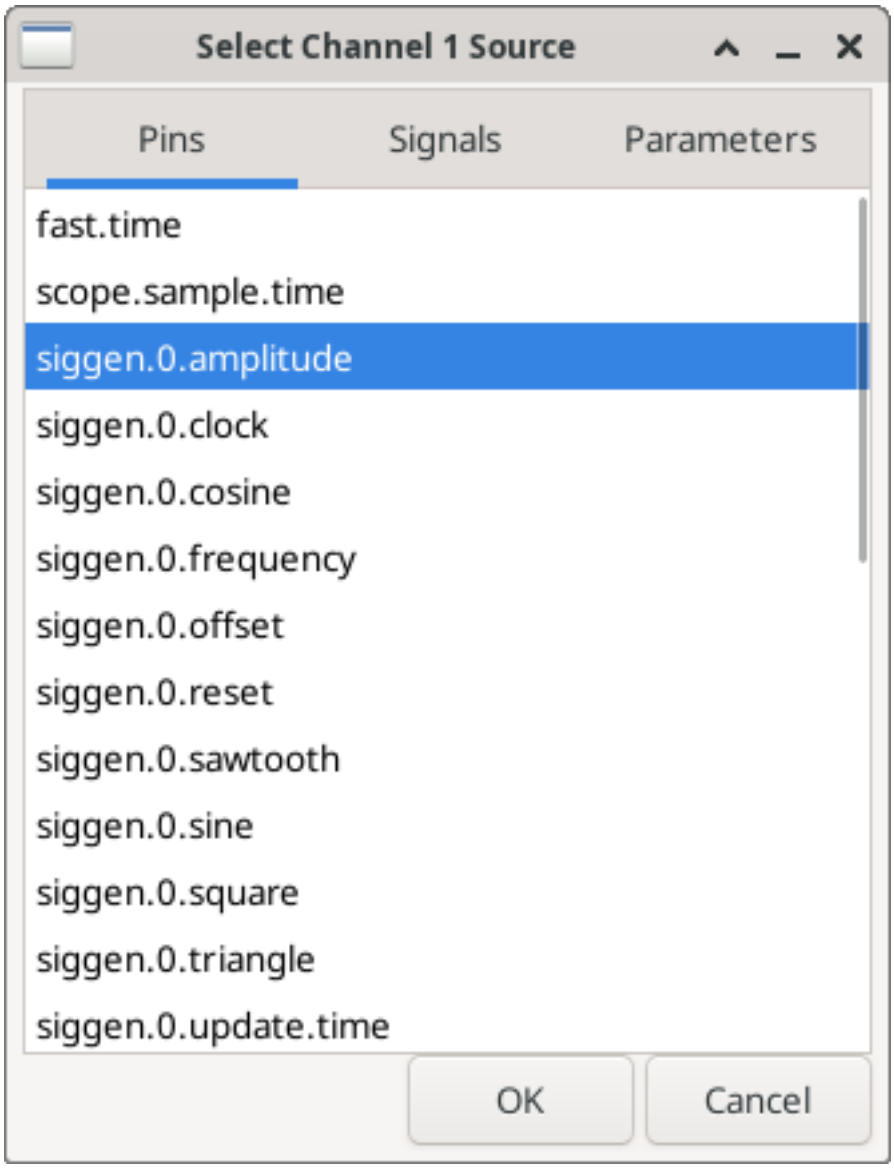


Figure 5.9: Виберіть джерело каналу

Щоб вибрати сигнал, просто клацніть на ньому. У цьому випадку ми хочемо, щоб канал 1 відображав сигнал «X-vel». Клацніть на вкладці «Сигнали», потім натисніть «X-vel», діалогове вікно закриється, і канал тепер вибрано.

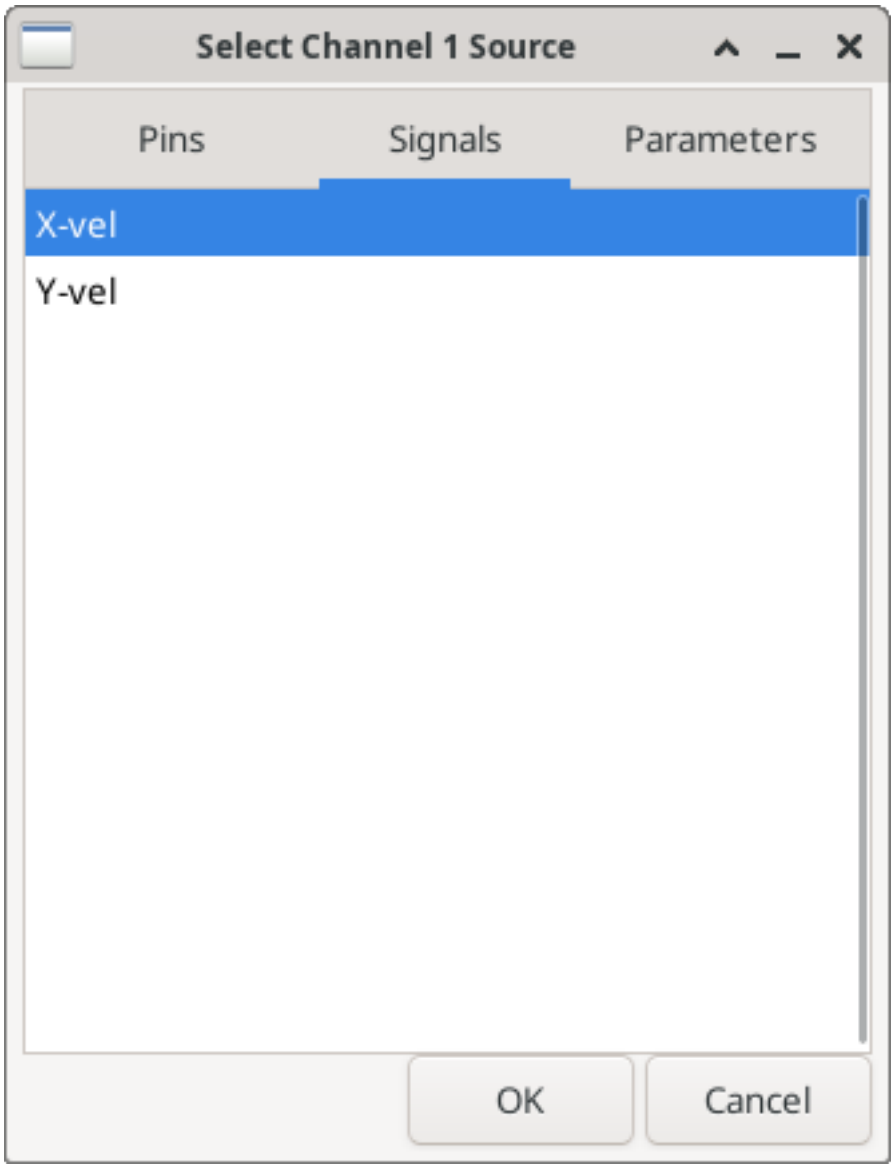


Figure 5.10: Виберіть сигнал

Натискається кнопка каналу 1, і під рядом кнопок з’являються номер каналу 1 та назва «X-vel». Цей дисплей завжди показує вибраний канал — на екрані може бути багато каналів, але вибраний канал виділяється, і різні елементи керування, такі як вертикальне положення та масштаб, завжди працюють саме для вибраного каналу.

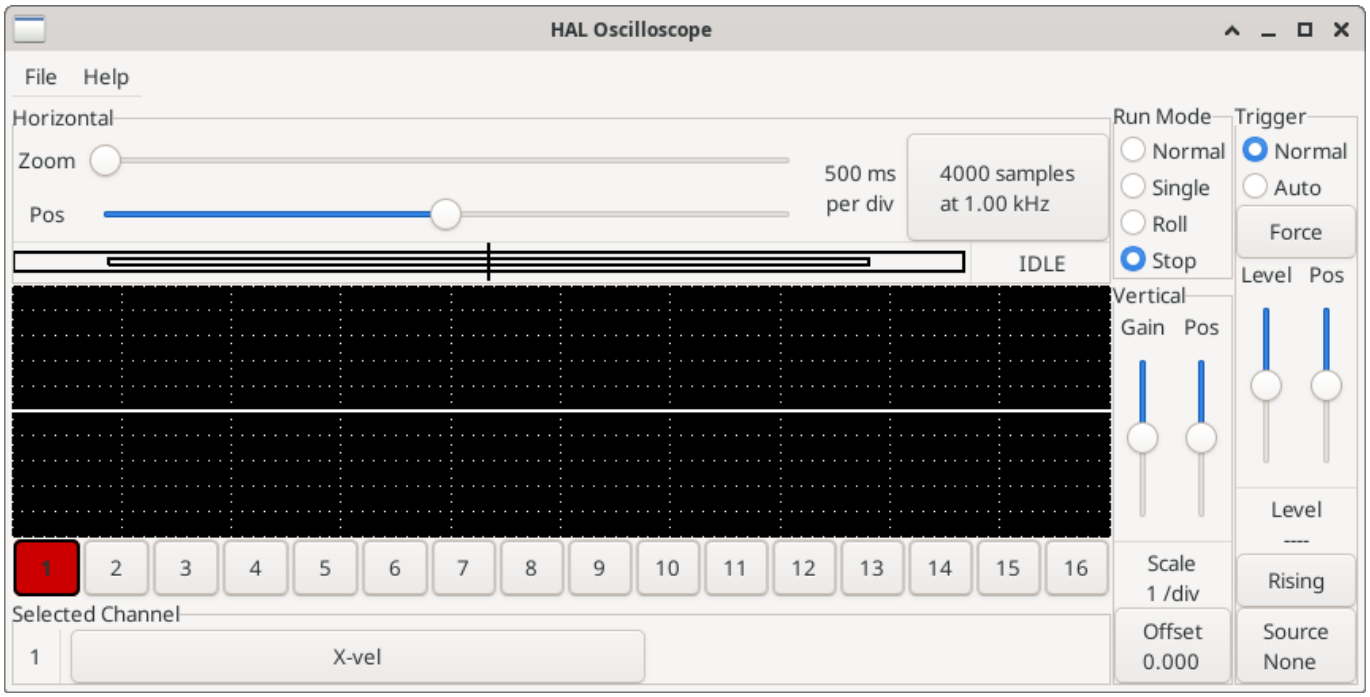


Figure 5.11: Галскоп

Щоб додати сигнал до каналу 2, натисніть кнопку «2». Коли з'явиться діалогове вікно, натисніть вкладку «Сигнали», а потім натисніть «Y-vel». Ми також хочемо подивитися на вихідні сигнали прямокутної та трикутної форми. До цих контактів не підключено жодних сигналів, тому ми використовуємо вкладку «Контакти». Для каналу 3 виберіть «siggen.0.triangle», а для каналу 4 — «siggen.0.square».

5.4.6.2 Захоплення наших перших хвильових форм

Тепер, коли ми підключили кілька датчиків до HAL, настав час записати деякі хвильові форми. Щоб запустити осцилограф, натисніть кнопку «Normal» у розділі «Run Mode» екрана (у правому верхньому куті). Оскільки ми маємо довжину запису 4000 зразків і отримуємо 1000 зразків на секунду, halscope знадобиться приблизно 2 секунди, щоб заповнити половину свого буфера. Протягом цього часу індикатор прогресу над головним екраном показуватиме заповнення буфера. Коли буфер заповниться наполовину, осцилограф чекатиме на тригер. Оскільки ми ще не налаштували тригер, він чекатиме вічно. Щоб запустити його вручну, натисніть кнопку «Force» у розділі «Trigger» у верхньому правому куті. Ви побачите, як заповнюється решта буфера, а потім на екрані з'являться записані хвильові форми. Результат буде схожий на зображення на малюнку нижче.

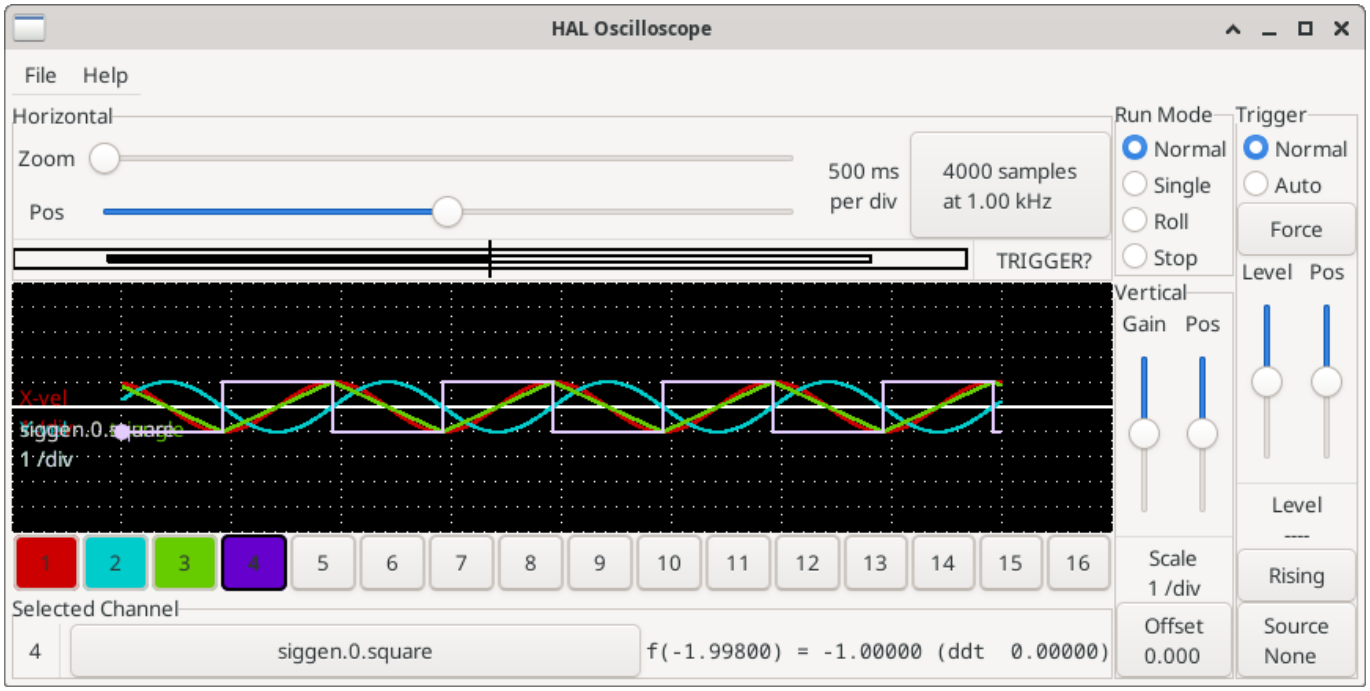


Figure 5.12: Захоплені форми хвиль

Поле «Вибраний канал» внизу показує, що фіолетовий трек є вибраним на даний момент, канал 4, який відображає значення виводу `siggen.0.square`. Спробуйте натиснути кнопки каналів від 1 до 3, щоб виділити інші три треки.

5.4.6.3 Вертикальні коригування

Сліди досить важко розрізнити, оскільки всі чотири знаходяться один над одним. Щоб виправити це, ми використовуємо елементи керування «Vertical» (Вертикальний) у вікні праворуч екрана. Ці елементи керування діють на поточний вибраний канал. Під час регулювання підсилення зверніть увагу, що воно охоплює великий діапазон — на відміну від реального осцилографа, цей може відображати сигнали від дуже малих (піко-одиниць) до дуже великих (тера-одиниць). Елемент керування положенням переміщує відображуваний слід вгору і вниз тільки по висоті екрана. Для більших регулювань слід використовувати кнопку зміщення.

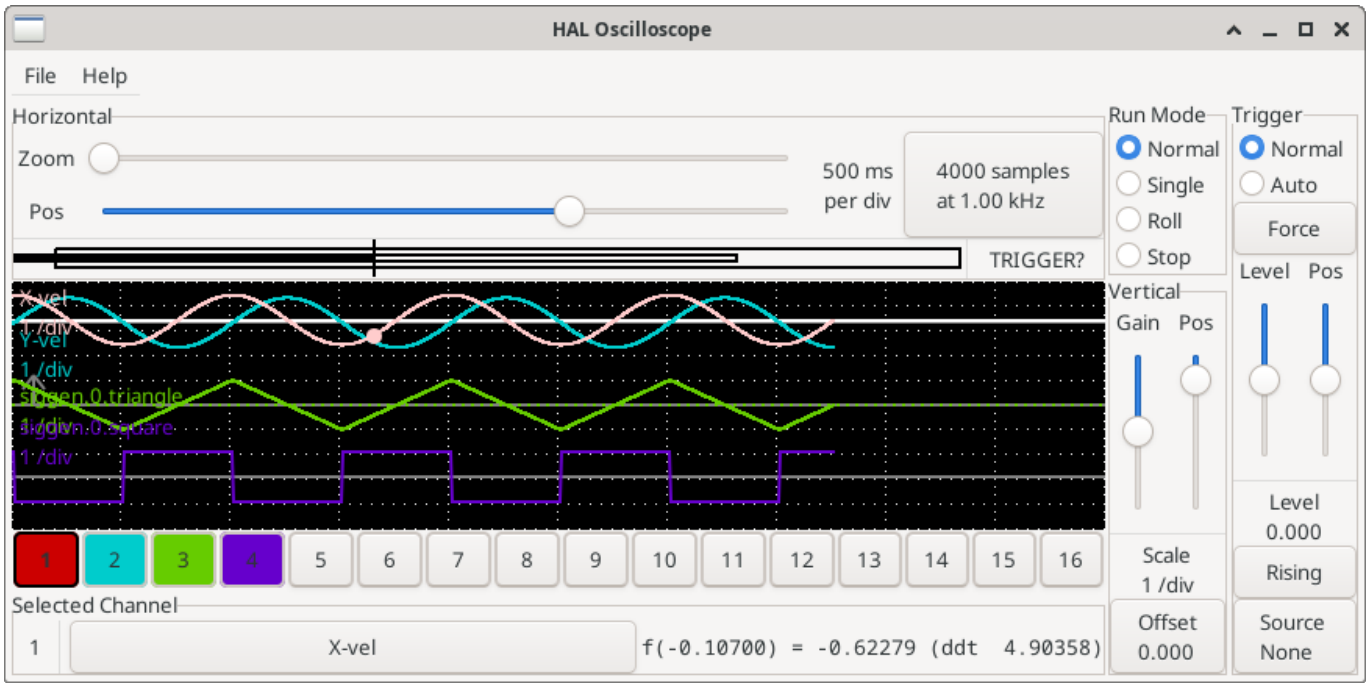


Figure 5.13: Вертикальне регулювання

Велика кнопка «Вибраний канал» внизу вказує, що канал 1 є вибраним каналом і відповідає сигналу «X-vel». Спробуйте натиснути на інші канали, щоб відобразити їхні криві та мати можливість переміщати їх за допомогою курсору «Pos».

5.4.6.4 Запуск

Використання кнопки «Force» є досить незадовільним способом запуску осцилографа. Щоб налаштувати справжній запуск, натисніть кнопку «Source» внизу праворуч. З’явиться діалогове вікно «Trigger Source», яке є просто списком усіх датчиків, що наразі підключені. Виберіть датчик, який буде використовуватися для запуску, натиснувши на нього. У цьому прикладі ми будемо використовувати канал 3, трикутну хвилю, як показано на наступному малюнку.

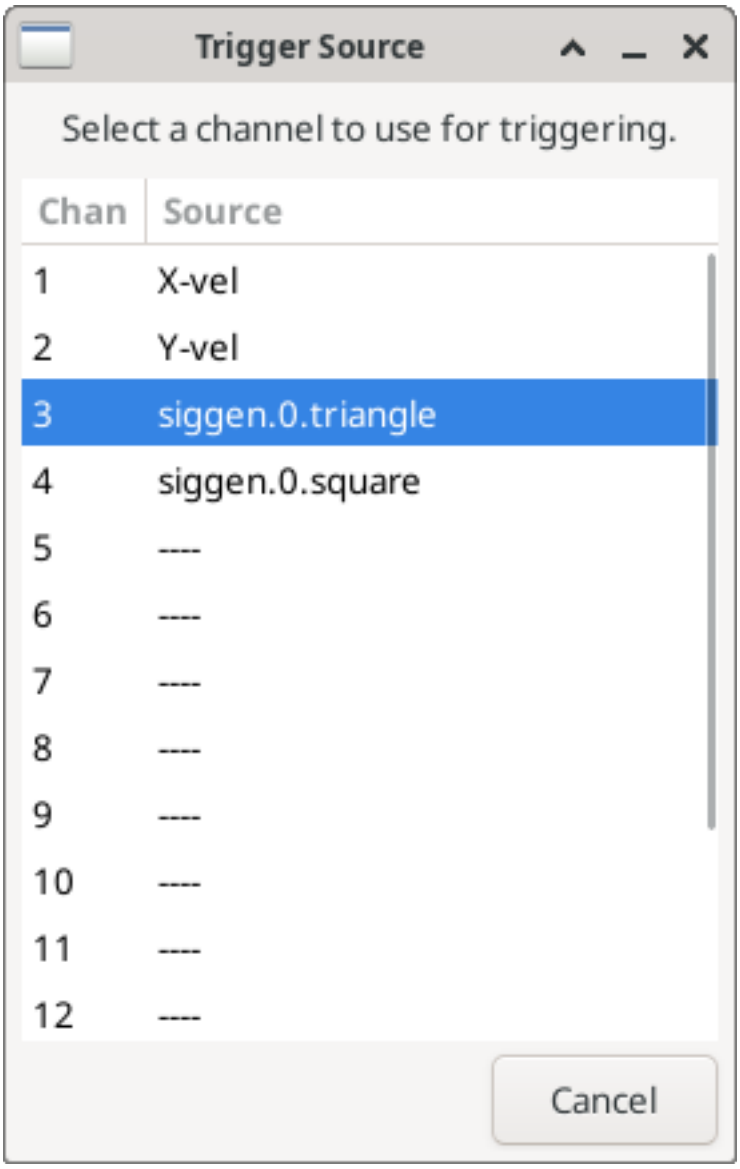


Figure 5.14: Діалогове вікно джерела тригера

Після встановлення джерела тригера ви можете налаштувати рівень тригера та його положення за допомогою повзунків у полі «Тригер» праворуч. Рівень можна налаштувати від верхньої до нижньої частини екрана, він відображається під повзунками. Положення — це місце розташування точки тригера в загальному записі. Коли повзунок знаходиться внизу, точка тригера знаходиться в кінці запису, і halscope відображає те, що відбувалося до точки тригера. Коли повзунок знаходиться вгорі, точка тригера знаходиться на початку запису, відображаючи те, що відбулося після його спрацьовування. Точка тригера відображається у вигляді вертикальної лінії у вікні прогресу над екраном. Полярність тригера можна змінити, натиснувши кнопку трохи нижче індикатора рівня тригера. Тоді вона стане *descendant*. Зверніть увагу, що зміна положення тригера зупиняє осцилограф після налаштування положення. Ви можете перезапустити осцилограф, натиснувши кнопку *Normal* у групі *Run mode*.

Тепер, коли ми налаштували вертикальні елементи керування та спусковий механізм, дисплей прицілу виглядає приблизно так, як показано на наступному малюнку.

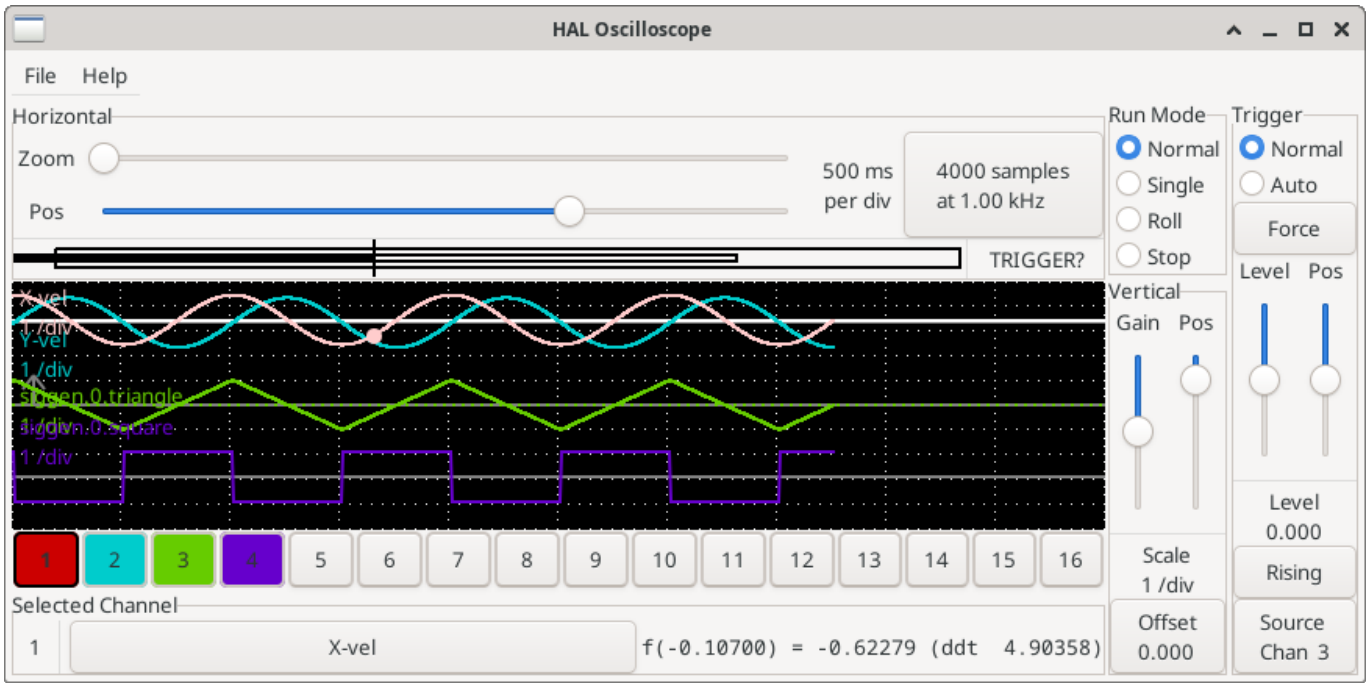


Figure 5.15: Форми хвиль із запуском

5.4.6.5 Горизонтальні коригування

Щоб детально розглянути частину хвильової форми, можна скористатися повзунком масштабування у верхній частині екрана, щоб розширити хвильові форми по горизонталі, а також повзунком положення, щоб визначити, яка частина збільшеної хвильової форми буде видима. Однак іноді простого розширення хвильових форм недостатньо, і потрібно збільшити частоту дискретизації. Наприклад, ми хотіли б розглянути фактичні імпульси, що генеруються в нашому прикладі. Оскільки імпульси можуть мати тривалість лише 50 мкс, дискретизація з частотою 1 кГц є недостатньо швидкою. Щоб змінити частоту дискретизації, натисніть кнопку, яка відображає кількість зразків і частоту дискретизації, щоб відкрити діалогове вікно «Вибір частоти дискретизації». У цьому прикладі ми натиснемо на рядок 50 мкс, «швидкий», що дає нам частоту дискретизації приблизно 20 кГц. Тепер замість відображення даних за приблизно 4 секунди, один запис становить 4000 зразків при 20 кГц, або приблизно 0,20 секунди.

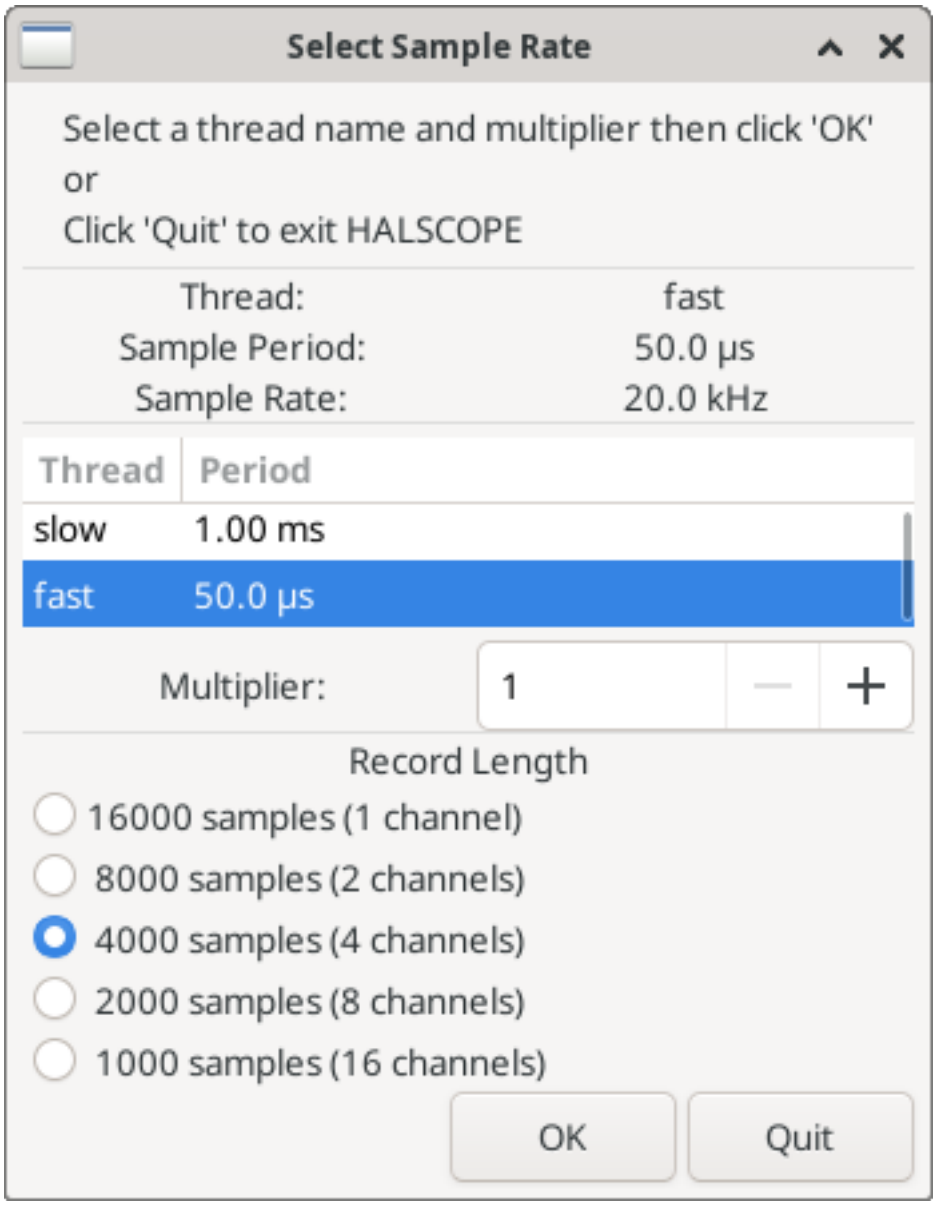


Figure 5.16: Діалогове вікно частоти дискретизації

5.4.6.6 Більше каналів

Тепер давайте розглянемо імпульси кроку. Halscope має 16 каналів, але в цьому прикладі ми використовуємо тільки 4 одночасно. Перш ніж вибрати інші канали, нам потрібно вимкнути декілька. Натискання на кнопку вибраного каналу (чорна рамка) вимкне канал. Тому натисніть на кнопку каналу 2, а потім натисніть на цю кнопку ще раз, і канал вимкнеться. Потім двічі натисніть на канал 3 і зробіть те саме для каналу 4. Незважаючи на те, що канали вимкнені, вони все одно пам'ятають, до чого вони підключені, і насправді ми продовжимо використовувати канал 3 як джерело тригера. Щоб додати нові канали, виберіть канал 5 і виберіть контакт `stepgen.0.dir`, потім канал 6 і виберіть `stepgen.0.step`. Потім натисніть режим роботи «Normal», щоб запустити осцилограф, і налаштуйте горизонтальне масштабування на 5 мс на поділку. Ви повинні побачити, як імпульси кроку сповільнюються, коли команда швидкості (канал 1) наближається до нуля, потім контакт напрямку змінює стан і імпульси кроку знову прискорюються. Можливо, ви захочете збільшити коефіцієнт підсилення на каналі 1 до приблизно 20 мілі на

поділку, щоб краще бачити зміну команди швидкості. Результат повинен виглядати як на наступному малюнку.

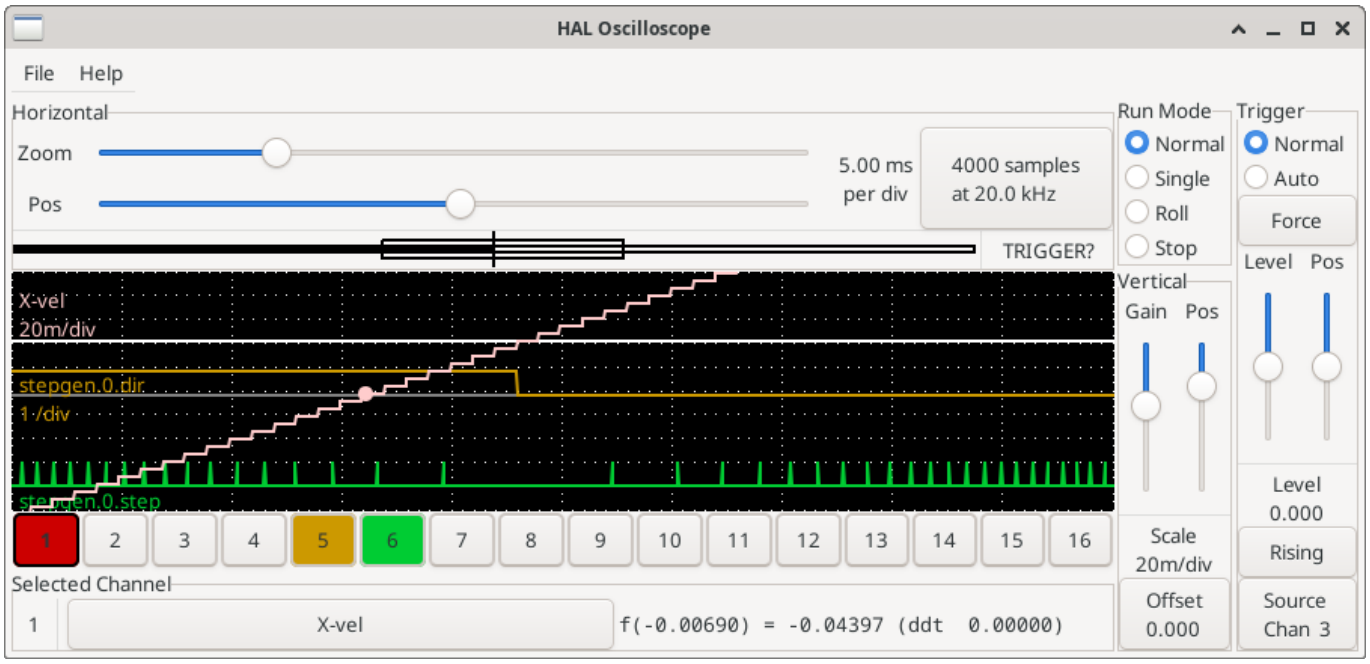


Figure 5.17: Крокові імпульси

5.4.6.7 Більше зразків

Якщо ви хочете записати більше семплів одночасно, перезапустіть режим реального часу та завантажте halscope з числовим аргументом, який вказує на кількість семплів, які ви хочете записати.

```
halcmd loadusr halscope 80000
```

Якщо компонент «scope_rt» ще не завантажено, halscope завантажить його і запросить загалом 80000 зразків, так що при вибірці 4 каналів одночасно буде 20000 зразків на канал. (Якщо «scope_rt» вже завантажено, числовий аргумент для halscope не матиме впливу).

5.5 Приклади HAL

Усі ці приклади передбачають, що ви починаєте з конфігурації на основі StepConf і маєте два потоки «base-thread» та «servo-thread». Майстер StepConf створить порожні файли custom.hal та custom_postgui.hal. Файл custom.hal буде завантажено після файлу HAL конфігурації, а файл custom_postgui.hal — після завантаження графічного інтерфейсу користувача.

5.5.1 Підключення двох виходів

Щоб підключити два виходи до входу, можна використовувати компонент or2. Компонент or2 працює таким чином: якщо один із входів or2 увімкнений, то вихід or2 увімкнений. Якщо жоден із входів or2 не увімкнений, вихід or2 вимкнений.

Наприклад, щоб дві кнопки PyVCP були підключені до одного світлодіода.

Файл .xml для вказівки PyVCP підготувати графічний інтерфейс користувача з двома кнопками (з назвами ”button-1” та ”button-2”) та світлодіодом (з назвою ”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, який зчитується після налаштування графічного інтерфейсу та готовності портів до прийняття логіки, описаної 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
```

Коли ви запускаєте цей приклад у симуляторі осей, створеному за допомогою майстра StepConf, ви можете відкрити термінал і переглянути контакти, створені за допомогою *loadrt or2*, ввівши `halcmd show pin or2` у терміналі.

Запуск halcmd у командному рядку UNIX для показу виводів, створених за допомогою модуля or2.

```
$ halcmd show pin or2
b’’Bb’’b’’ив’’b’’вв’’b’’ов’’b’’дв’’b’’ив’’ b’’кв’’b’’ов’’b’’мв’’b’’пв’’b’’ов’’b’’нв’’b’ ↩
    ’ев’’b’’нв’’b’’тв’’b’’ів’’b’’вв’’:
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
```

З команди HAL `show pin or2` видно, що контакт `button-1` підключений до контакту `or2.0.in0`. Зі стрілки напрямку видно, що кнопка є виходом, а `or2.0.in0` — входом. Вихід з or2 підключений до входу світлодіода.

5.5.2 Ручна зміна інструменту

У цьому прикладі передбачається, що ви створюєте власну конфігурацію і бажаєте додати вікно HAL Manual Toolchange (Ручна зміна інструменту HAL). HAL Manual Toolchange в першу чергу корисний, якщо у вас є попередньо налаштовані інструменти і ви зберігаєте зміщення в таблиці інструментів. Якщо вам потрібно торкатися кожного інструменту при зміні, то найкраще просто розділити ваш G-код. Щоб використовувати вікно HAL Manual Toolchange, вам в основному потрібно

1. завантажити компонент hal_manualtoolchange,

2. потім надішліть iocontrol *tool change* до hal_manualtoolchange *change* та
3. надішліть hal_manualtoolchange *changed* назад до iocontrol *tool changed*.

Для зовнішнього введення сигналу про завершення зміни інструменту передбачено контакт.

Це приклад ручної зміни інструменту «за допомогою» 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
```

Це приклад ручної зміни інструменту «без» компонента ручної зміни інструменту HAL:

```
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 Обчисліть швидкість

У цьому прикладі використовуються функції «ddt», «mult2» та «abs» для обчислення швидкості однієї осі. Докладнішу інформацію про компоненти реального часу див. у довідкових сторінках або списку компонентів HAL (Section 5.1.5).

Перше, що потрібно зробити, це перевірити конфігурацію, щоб переконатися, що ви не використовуєте жодних компонентів реального часу. Для цього відкрийте вікно HAL Configuration (Конфігурація HAL) і знайдіть компоненти в розділі pin (контакти). Потім знайдіть файл HAL, в який вони завантажуються, збільште кількість і налаштуйте екземпляр на правильне значення. Додайте наступне до файлу custom.hal.

Завантажте компоненти реального часу.

```
loadrt ddt count=1
loadrt mult2 count=1
loadrt abs count=1
```

Додайте функції до потоку, щоб він оновлювався.

```
addf ddt.0 servo-thread
addf mult2.0 servo-thread
addf abs.0 servo-thread
```

Встановіть зв’язки.

```
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 цьому останньому розділі ми встановлюємо `mult2.0.in1` на 60, щоб конвертувати дюйми за секунду в дюйми за хвилину (IPM), які ми отримуємо з `ddt.0.out`.

`xpos-cmd` надсилає задану позицію до `ddt.0.in`. ddt обчислює похідну зміни вхідних даних.

Значення `ddt2.0.out` множиться на 60, щоб отримати IPM.

`mult2.0.out` надсилається до `abs` для отримання абсолютного значення.

На наступному рисунку показано результат, коли вісь X рухається зі швидкістю 15 дюймів за хвилину в мінусовому напрямку. Зверніть увагу, що ми можемо отримати абсолютне значення або з виводу `abs.0.out`, або з сигналу `X-IPM`.

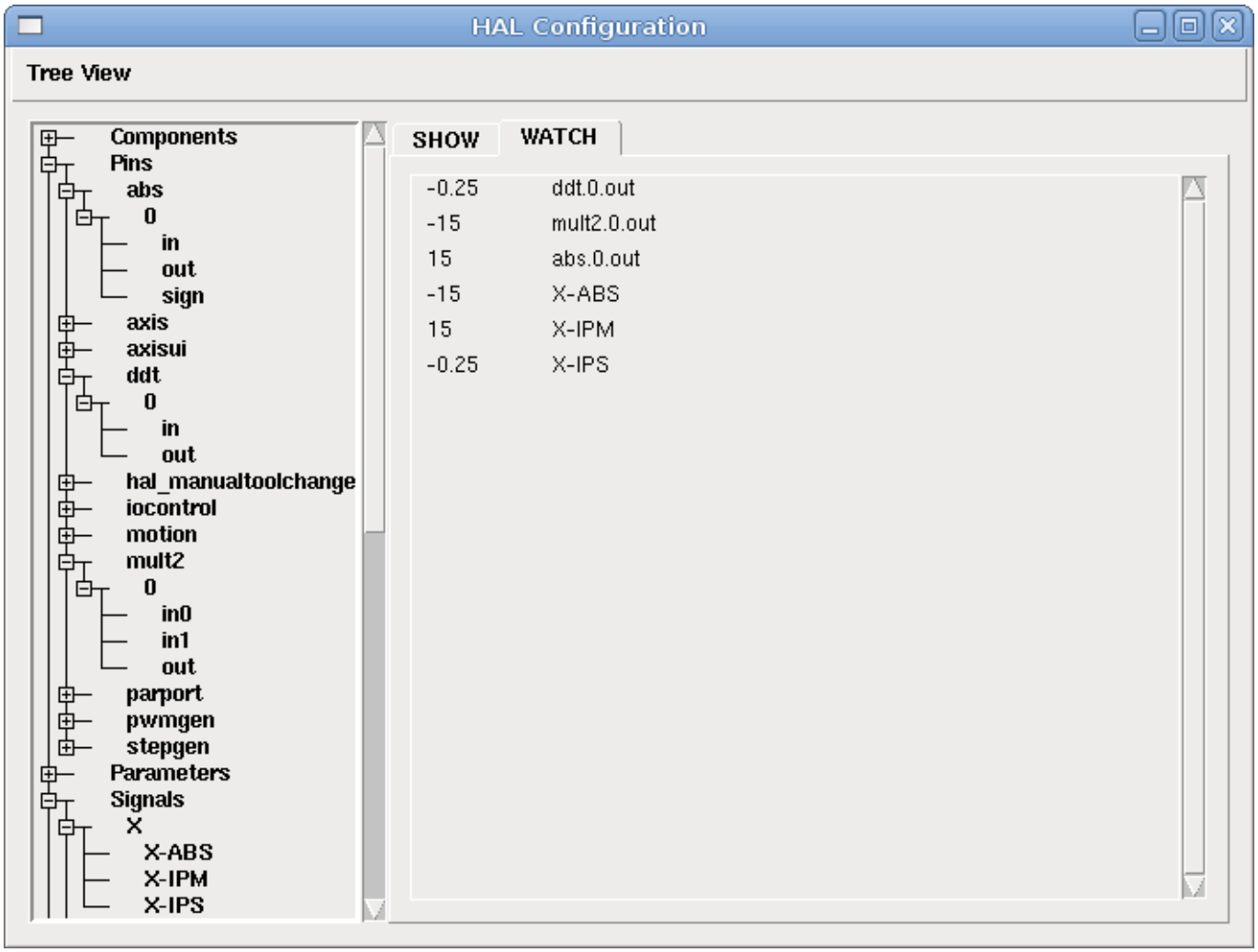


Figure 5.18: HAL: Приклад швидкості

5.5.4 Деталі плавного запуску

У цьому прикладі показано, як компоненти HAL «lowpass», «limit2» або «limit3» можна використовувати для обмеження швидкості зміни сигналу.

У цьому прикладі ми маємо сервомотор, що приводить в рух шпиндель токарного верстата. Якщо ми просто використаємо задані швидкості шпинделя на сервоприводі, він спробує перейти від поточної швидкості до заданої швидкості якомога швидше. Це може спричинити проблему або пошкодити привід. Щоб уповільнити швидкість зміни, ми можемо надіслати `spindle.N.speed-out` через обмежувач перед PID, щоб значення команди PID повільніше змінювалося на нові налаштування.

Три вбудовані компоненти, що обмежують сигнал:

- *limit2* обмежує діапазон та першу похідну сигналу.
- «limit3» обмежує діапазон, першу та другу похідні сигналу.

- «Низькочастотний» використовує експоненціально зважену ковзну середню для відстеження вхідного сигналу.

Щоб знайти більше інформації про ці компоненти HAL, перегляньте сторінки довідки (man).

Помістіть наступний код у текстовий файл під назвою softstart.hal. Якщо ви не знайомі з Linux, помістіть файл у свій домашній 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
```

Відкрийте вікно терміналу та запустіть файл за допомогою наступної команди.

```
halrun -I softstart.hal
```

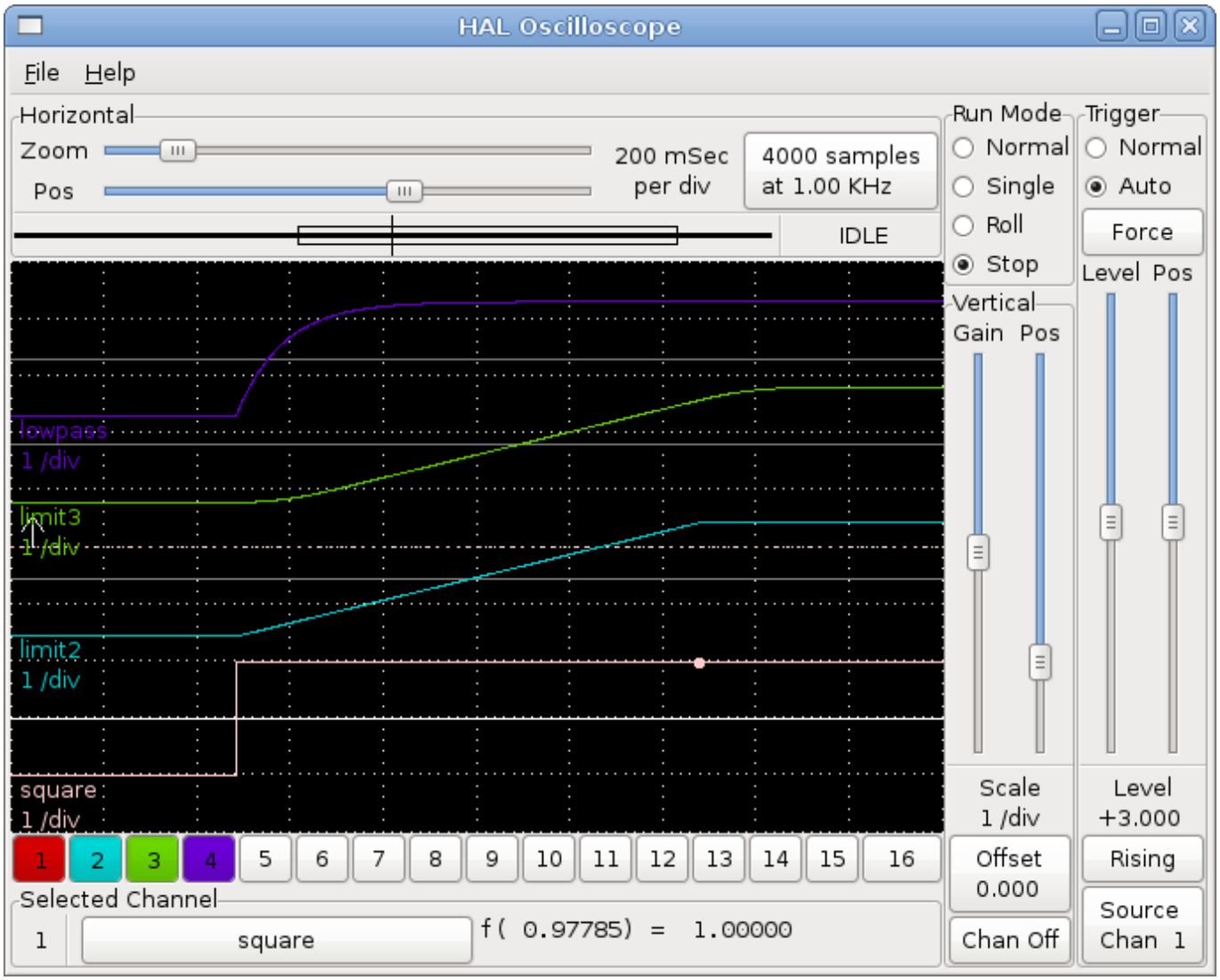
Коли осцилограф HAL вперше запуститься, натисніть кнопку «OK», щоб прийняти потік даних за замовчуванням.

Далі потрібно додати сигнали до каналів. Клацніть на каналі 1, потім виберіть «квадрат» на вкладці «Сигнали». Повторіть для каналів 2-4 та додайте низькочастотний сигнал, limit2 та limit3.

Далі, щоб налаштувати сигнал запуску, натисніть кнопку «Джерело немає» та виберіть квадрат. Кнопка зміниться на «Джерело каналу 1».

Далі натисніть кнопку «Одинарний» у полі «Режим виконання». Це розпочне виконання, і після його завершення ви побачите свої траси.

Щоб розділити сигнали для кращого бачення, клацніть на каналі, а потім скористайтеся повзунком «Позиція» у полі «Вертикальна», щоб встановити положення.



Щоб побачити ефект зміни значень заданих точок будь-якого з компонентів, ви можете змінити їх у вікні терміналу. Щоб побачити, як різні налаштування коефіцієнта підсилення впливають на низькочастотний фільтр, просто введіть наступне у вікні терміналу та спробуйте різні налаштування.

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

Після зміни налаштування знову запустіть осцилограф, щоб побачити зміни.

Коли ви закінчите, введіть «exit» у вікні терміналу, щоб закрити halrun і закрити halscope. Не закривайте вікно терміналу, коли halrun працює, оскільки це може залишити в пам’яті деякі елементи, які можуть перешкоджати завантаженню LinuxCNC.

Для отримання додаткової інформації про Halscope дивіться посібник з HAL та навчальний посібник.

5.5.5 Автономний HAL

У деяких випадках може знадобитися запустити екран GladeVCP тільки з HAL. Наприклад, уявіть, що у вас є пристрій з кроковим приводом, для якого потрібно тільки запустити кроковий двигун. Для вашої програми достатньо простого інтерфейсу «Пуск/Стоп», тому немає необхідності завантажув та налаштовувати повноцінну програму CNC.

У наступному прикладі ми створили просту панель GladeVCP з одним кроковим механізмом.

Базовий синтаксис

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''тb''b''eb'' b''гb''b''pb''b' ↩
    'ab''b''фb''b''ib''b''чb''b''нb''b''иb''b''йb'' b''ib''b''нb''b''тb''b''eb''b''pb''b' ↩
    'фb''b''eb''b''йb''b''cb'' winder.glade b''тb''b''ab'' b''нb''b''ab''b''зb''b''вb''b' ↩
    'ib''b''тb''b''ьb'' b''йb''b''ob''b''гb''b''ob'' winder
loadusr -Wn winder gladevcp -c winder -u handler.py winder.glade

# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ib''b''вb'' b''pb''b' ↩
    'eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''чb''b''ab''b''cb''b''yb''
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"

# b''дb''b''ob''b''дb''b''ab''b''вb''b''ab''b''тb''b''иb'' b''фb''b''yb''b''нb''b''кb''b' ↩
    'цb''b''ib''b''їb'' b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''ib''b''вb''
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

# b''cb''b''тb''b''вb''b''ob''b''pb''b''юb''b''вb''b''ab''b''тb''b''иb'' HAL-b''зb'''b' ↩
    'єb''b''дb''b''нb''b''ab''b''нb''b''нb''b''яb''
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''пb''b''ob''b''чb''b''ab''b''тb''b''иb'' b''тb''b''eb''b''мb''b''иb''
start

# b''зb''b''ab''b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''yb''b''йb''b''тb''b''eb'' b' ↩
    'нb''b''ab''b''cb''b''тb''b''yb''b''пb''b''нb''b''ib'' b''pb''b''яb''b''дb''b''кb''b' ↩
    'иb'' b''пb''b''ib''b''дb'' b''чb''b''ab''b''cb'' b''тb''b''eb''b''cb''b''тb''b''yb''b' ↩
    'вb''b''ab''b''нb''b''нb''b''яb'' b''тb''b''ab'' b''вb''b''иb''b''кb''b''ob''b''pb''b' ↩
    'иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''йb''b''тb''b''eb'' b''ib''b''нb''b''тb''b' ↩
    'eb''b''pb''b''ab''b''кb''b''тb''b''иb''b''вb''b''нb''b''yb''
# b''ob''b''пb''b''цb''b''ib''b''юb'' halrun -I -f start.hal, b''щb''b''ob''b''бb'' b''мb'' ↩
    b''ab''b''тb''b''иb'' b''зb''b''мb''b''ob''b''гb''b''yb'' b''пb''b''ob''b''кb''b''ab''b' ↩
    'зb''b''yb''b''вb''b''ab''b''тb''b''иb'' b''пb''b''ib''b''нb''b''иb'' b''тb''b''ob''b' ↩
    'щb''b''ob''.

# b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'', b''пb''b''ob''b''кb''b' ↩
    'иb'' b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''иb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'pb''b''ob''b''бb''b''ob''b''тb''b''ab'' b''гb''b''pb''b''ab''b''фb''b''ib''b''чb''b' ↩
    'нb''b''ob''b''гb''b''ob'' b''ib''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b' ↩
    'cb''b''yb'' GladeVCP b''зb'' b''нb''b''ab''b''зb''b''вb''b''ob''b''юb'' winder
waitusr winder

# b''зb''b''yb''b''пb''b''иb''b''нb''b''иb''b''тb''b''иb'' b''пb''b''ob''b''тb''b''ob''b' ↩
    'кb''b''иb'' HAL
stop

# b''вb''b''иb''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''вb''b' ↩
    'cb''b''ib'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''иb'' HAL ↩
    b''пb''b''eb''b''pb''b''eb''b''дb'' b''вb''b''иb''b''хb''b''ob''b''дb''b''ob''b''мb''
unloadrt all
```

5.6 Основні компоненти

Див. також сторінки довідки «motion(9)».

5.6.1 Рух

Ці виводи та параметри створюються модулем реального часу *motmod*.

Цей модуль надає інтерфейс HAL для планувальника руху LinuxCNC.

По суті, motmod приймає список точок маршруту та генерує гарний змішаний та обмежений обмеженнями потік позицій суглобів, які подаються на приводи двигунів.

За бажанням, кількість цифрових входів/виходів встановлюється за допомогою *num_dio*. Кількість аналогових входів/виходів встановлюється за допомогою *num_aio*, за замовчуванням — 4 кожен. Кількість шпинделів встановлюється за допомогою *num_spindles*, за замовчуванням — 1.

Назви контактів та параметрів, що починаються з *axis.*L та *joint.*N, зчитуються та оновлюються функцією контролера руху.

Рух завантажується командою motmod. Kins слід завантажити перед рухом.

```
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* – період виконання завдання «Base» у наносекундах. Це найшвидший потік у машині.

Note
У сервосистемах, як правило, немає причин, щоб «base_period_nsec» було меншим за «servo_period_nsec». На машинах із програмним генеруванням кроків «base_period_nsec» визначає максимальну кількість кроків за секунду. За відсутності вимог до великої довжини кроку та крокового простору, абсолютна максимальна частота кроків становить один крок на «base_period_nsec». Таким чином, «base_period_nsec», показаний вище, дає абсолютну максимальну частоту кроків 20 000 кроків на секунду. 50 000 нс (50 мкс) є досить консервативним значенням. Найменше значення, яке можна використовувати, пов'язане з результатом тесту затримки, необхідною довжиною кроку та швидкістю процесора. Вибір занадто низького значення «base_period_nsec» може призвести до появи повідомлення «Несподівана затримка в реальному часі», блокування або спонтанного перезавантаження.

- *servo_period_nsec = 1000000* - це період завдання «Серво» в наносекундах. Це значення буде округлено до цілого кратного «base_period_nsec». Цей період використовується навіть у системах на основі крокових двигунів.

  Це швидкість, з якою обчислюються нові положення двигуна, перевіряється похибка, оновлюються вихідні значення PID тощо. Більшість систем не потребують зміни цього значення. Це швидкість оновлення планувальника руху низького рівня.

- *traj_period_nsec = 100000* - це період завдання «Планувальник траєкторії» в наносекундах. Це значення буде округлено до цілого кратного «servo_period_nsec». За винятком машин з незвичайною кінематикою (наприклад, гексаподів), немає причин робити це значення більшим за «servo_period_nsec».

5.6.1.1 Опції

Якщо потрібна кількість цифрових входів/виходів більша за значення за замовчуванням 4, ви можете додати до 64 цифрових входів/виходів, використовуючи опцію num_dio під час завантаження *motmod*.

Якщо потрібна кількість аналогових входів/виходів більша за значення за замовчуванням 4, ви можете додати до 16 аналогових входів/виходів, використовуючи опцію num_aio під час завантаження *motmod*.

Параметр unlock_joints_mask використовується для створення контактів для з'єднання, яке використов як блокувальний індексатор (зазвичай поворотний). Біти маски вибирають з'єднання. Молодший біт маски вибирає з'єднання 0. Приклад:

```
unlock_joints_mask=0x38 selects joints 3,4,5
```

5.6.1.2 Піни

Ці виводи, параметри та функції створюються модулем реального часу *motmod*.

- *motion.adaptive-feed* - (float, in) Коли адаптивна подача ввімкнена за допомогою *M52 P1* , задана швидкість множиться на це значення. Цей ефект є мультиплікативним із значенням перевищення рівня подачі NML та *motion.feed-hold*. Починаючи з версії 2.9 LinuxCNC, можна використовувати від'ємне значення адаптивної подачі для виконання траєкторії G-коду у зворотному напрямку.
- *motion.analog-in-00* - (float, in) Ці виводи (00, 01, 02, 03 або більше, якщо налаштовано) керуються M66.
- *motion.analog-out-00* - (float, out) Ці виводи (00, 01, 02, 03 або більше, якщо налаштовано) керуються M67 або M68.
- *motion.coord-error* - (біт, вихід) TRUE, коли рух зіткнувся з помилкою, такою як перевищення програмного ліміту
- *motion.coord-mode* - (біт, вихід) TRUE, коли рух знаходиться в «координованому режимі», на відміну від «телеоптичного режиму»
- *motion.current-vel* - (float, out) Поточна швидкість інструменту в одиницях користувача за секунду.
- *motion.digital-in-00* - (біт, вхід) Ці виводи (00, 01, 02, 03 або більше, якщо налаштовано) керуються M62-65.
- *motion.digital-out-00* - (біт, вихід) Ці контакти (00, 01, 02, 03 або більше, якщо налаштовано) керуються *M62-65*.
- *motion.distance-to-go* - (float,out) Відстань, що залишилася в поточному русі.
- «motion.enable» — (біт, вхід) Якщо цей біт встановлений у значення FALSE, рух зупиняється, машина переходить у стан «машина вимкнена» і оператору відображається відповідне повідомлення Для нормального руху встановіть цей біт у значення TRUE.
- *motion.feed-hold* - (біт, вхід) Коли керування зупинкою подачі ввімкнено за допомогою *M53 P1*, і цей біт має значення TRUE, швидкість подачі встановлюється на 0.
- *motion.feed-inhibit* - (біт, вхід) Коли цей біт має значення TRUE (ІСТИНА), швидкість подачі встановлюється на 0. Це буде затримуватися під час синхронізованих переміщень шпинделя до кінця переміщення.
- *motion.in-position* - (біт, вихід) TRUE, якщо машина знаходиться в положенні.
- *motion.motion-enabled* - (біт, вихід) TRUE у стані *машина ввімкнена*.

- *motion.motion-type* - (s32, out) Ці значення взяті з src/emc/nml_intf/motion_types.h
  - 0: Холостий (без руху)
  - 1: Траверс
  - 2: Лінійна подача
  - 3: Подача дуги
  - 4: Зміна інструменту
  - 5: Зондування
  - 6: Індексування обертової осі
- *motion.on-soft-limit* - (біт, вихід) TRUE, коли машина знаходиться на м’якому граничному значенні.
- *motion.probe-input* - (біт, вхід) *G38.n* використовує значення на цьому виводі для визначення моменту контакту зонда. TRUE (ІСТИНА) для замкнутого (торкання) контакту зонда, FALSE (ХИБНІСТЬ) для розімкнутого контакту зонда.
- *motion.program-line* - (s32, out) Поточний рядок програми під час виконання. Нуль, якщо програма не виконується, або між рядками під час покрокового виконання.
- «motion.requested-vel» — (float, out) Поточна запитувана швидкість в одиницях користувача за секунду. Це значення є налаштуванням F-слова з файлу G-коду, яке може бути зменшене з урахуванням обмежень швидкості та прискорення машини. Значення на цьому виводі не відображає перевищення подачі або будь-які інші коригування.
- *motion.teleop-mode* - (біт, вихід) TRUE, коли рух відбувається в «режимі телеоп», на відміну від «координованого режиму»
- *motion.tooloffset.x … motion.tooloffset.w* - (float, out, один на вісь) показує фактичне зміщення інструменту; воно може походити з таблиці інструментів (*G43* активний) або з G-коду (*G43.1* активний)
- *motion.on-soft-limit* - (біт, вихід) TRUE, коли машина знаходиться на м’якому граничному значенні.
- *motion.probe-input* - (біт, вхід) *G38.n* використовує значення на цьому виводі для визначення моменту контакту зонда. TRUE (ІСТИНА) для замкнутого (торкання) контакту зонда, FALSE (ХИБНІСТЬ) для розімкнутого контакту зонда.
- *motion.program-line* - (s32, out) Поточний рядок програми під час виконання. Нуль, якщо програма не виконується, або між рядками під час покрокового виконання.
- «motion.requested-vel» — (float, out) Поточна запитувана швидкість в одиницях користувача за секунду. Це значення є налаштуванням F-слова з файлу G-коду, яке може бути зменшене з урахуванням обмежень швидкості та прискорення машини. Значення на цьому виводі не відображає перевищення подачі або будь-які інші коригування.
- *motion.teleop-mode* - (біт, вихід) TRUE, коли рух відбувається в «режимі телеоп», на відміну від «координованого режиму»
- *motion.tooloffset.x … motion.tooloffset.w* - (float, out, один на вісь) показує фактичне зміщення інструменту; воно може походити з таблиці інструментів (*G43* активний) або з G-коду (*G43.1* активний)

5.6.1.3 Параметри

Багато з цих параметрів служать допоміжними засобами налагодження та можуть бути змінені або видалені в будь-який час.

- *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) Використовується для налагодження.
- *motion.debug-bit-1* - (bit, RO) Це використовується для налагодження.
- *motion.debug-float-0* - (float, RO) Це використовується для налагодження.
- *motion.debug-float-1* - (float, RO) Це використовується для налагодження.
- *motion.debug-float-2* - (float, RO) Це використовується для налагодження.
- *motion.debug-float-3* - (float, RO) Це використовується для налагодження.
- *motion.debug-s32-0* - (s32, RO) Це використовується для налагодження.
- *motion.debug-s32-1* - (s32, RO) Це використовується для налагодження.
- *motion.servo.last-period* - (u32, RO) Кількість циклів ЦП між викликами серво-потоку. Зазвичай це число, поділене на швидкість ЦП, дає час у секундах і може бути використане для визначення, чи відповідає контролер руху в реальному часі своїм обмеженням за часом
- *motion.servo.last-period-ns* - (float, RO)

5.6.1.4 Функції

Зазвичай ці функції додаються до серво-потоку в показаному порядку.

- *motion-command-handler* – Отримує та обробляє команди руху
- *motion-controller* – Запускає контролер руху LinuxCNC

5.6.2 Шпиндель

LinuxCNC може керувати до восьми шпинделів. Рух генеруватиме такі контакти: *N* (ціле число від 0 до 7) замінює номер шпинделя.

5.6.2.1 Піни

* *spindle*.N.*at-speed* - (біт, вхід) Рух буде призупинено, доки цей вивід не стане TRUE, за таких умов: **перед першим рухом подачі після кожного запуску шпинделя або зміни швидкості;** перед початком кожного ланцюга синхронізованих шпинделем рухів; ** а в режимі CSS — при кожному швидкому переході до подачі. Цей вхідний сигнал можна використовувати для забезпечення необхідної швидкості шпинделя перед початком різання або для уповільнення шпинделя токарного верстата в режимі CSS після великого до малого проходу перед початком наступного проходу на великому діаметрі. Багато VFD мають вихід «at speed» (на швидкості). В іншому випадку цей сигнал легко генерувати за допомогою компонента «HAL near», порівнюючи запитувану і фактичну швидкість шпинделя. * *spindle*.N.*brake* - (біт, вихід) TRUE, коли слід застосувати гальмо шпинделя. * *spindle*.N.*forward* - (біт, вихід) TRUE, коли шпиндель має обертатися вперед. * *spindle*.N.*index-enable* - (біт, введення/виведення) Для правильної роботи синхронізованих переміщен шпинделя цей контакт має бути підключений до контакту індексування енкодера шпинделя. * *spindle*.N.*inhibit* - (біт, вхід) Коли цей біт має значення TRUE (ІСТИНА), швидкість шпинделя встановлюється на 0. * *spindle*.N.*on* - (біт, вихід) TRUE, коли шпиндель має обертатися. * *spindle*.N.*reverse* - (біт, вихід) TRUE, коли шпиндель має обертатися назад * *spindle*.N.*revs* - (float, in)

Для коректної роботи синхронізованих рухів шпинделя цей сигнал повинен бути підключений до контакту положення енкодера шпинделя. Положення енкодера шпинделя повинно бути масштабоване таким чином, щоб оберти шпинделя збільшувалися на 1,0 за кожний оберт шпинделя за годинниковою стрілкою (*M3*). * *spindle*.N.*speed-in* - (float, in) Зворотний зв'язок фактичної швидкості шпинделя в обертах за секунду. Використовується для руху з подачею на оберт (*G95*). Якщо драйвер енкодера шпинделя не має виходу швидкості, ви можете створити відповідний, надіславши положення шпинделя через компонент «ddt». Якщо у вас немає енкодера шпинделя, ви можете зациклити «spindle.*N*.speed-out-rps». * *spindle*.N.*speed-out* - (float, out) Задана швидкість шпинделя в обертах за хвилину. Додатна для руху шпинделя вперед (*M3*), від'ємна для руху шпинделя назад (*M4*). * *spindle*.N.*speed-out-abs* - (float, out) Задана швидкість шпинделя в обертах за хвилину. Це завжди буде додатним числом. * *spindle*.N.*speed-out-rps* - (float, out) Задана швидкість шпинделя в обертах за секунду. Додатна для руху шпинделя вперед (*M3*), від'ємна для руху шпинделя назад (*M4*). * *spindle*.N.*speed-out-rps-abs* - (float, out) Задана швидкість шпинделя в обертах за секунду. Це завжди буде додатним числом. * *spindle*.N.*orient-angle* - (float,out) Бажана орієнтація шпинделя для M19. Значення параметра R-слова M19 плюс значення параметра INI [RS274NGC]ORIE
* *spindle*.N.*orient-mode* - (s32,out) Бажаний режим обертання шпинделя M19. За замовчуванням 0. * *spindle*.N.*orient* - (вихід, біт) Вказує на початок циклу орієнтації шпинделя. Встановлюється командою M19. Скидається будь-якою з команд M3, M4 або M5. Якщо під час орієнтації шпинделя значення spindle-orient-fault не дорівнює нулю, команда M19 завершується з помилкою. * *spindle*.N.*is-oriented* - (вхід, біт) Контакт підтвердження для орієнтації шпинделя. Завершує цикл орієнтації. Якщо орієнтація шпинделя була справжньою, коли було підтверджено, що шпиндель орієнтований, контакт орієнтації шпинделя очищується, а контакт блокування шпинделя підтверджує
Також підтверджується контакт гальма шпинделя. * *spindle*.N.*orient-fault* - (s32, in) Вхідний код помилки для циклу орієнтації. Будь-яке значення, відмінне від нуля, призведе до переривання циклу орієнтації. * *spindle*.N.*lock* - (bit, out) Штифт повної орієнтації шпинделя. Очищається будь-яким з M3, M4 або M5.

Використання штифта HAL для шпинделя M19 Orient Концептуально шпиндель знаходиться в одному з наступних режимів:

- режим обертання (за замовчуванням)
- пошук потрібного режиму орієнтації
- режим повної орієнтації.

Коли виконується M19, шпиндель переходить у режим «пошуку бажаного орієнтування», і активується контакт HAL `spindle.__N__.orient`. Бажане цільове положення визначається контактами `spindle.__` і `spindle.__N__.orient-fwd` і керується параметрами M19 R і P.

Логіка підтримки HAL повинна реагувати на `spindle.__N__.orient`, переміщуючи шпиндель у бажане положення. Після завершення цього процесу логіка HAL повинна підтвердити це, активувавши контакт `spindle.__N__.is-oriented`.

Motion підтверджує це, скасовуючи сигнал на виводі `spindle.__N__.orient` і подаючи сигнал на виводі `spindle.__N__.locked`, щоб вказати режим «орієнтація завершена». Він також подає сигнал на виводі `spindle.__N__.brake`. Тепер шпиндель перебуває в режимі «орієнтація завершена».

Якщо під час виконання `spindle.__N__.orient` є істинним, а `spindle.__N__.is-oriented` ще не підтверджено, контакт `spindle.__N__.orient-fault` має значення, відмінне від нуля, команда M19 переривається, відображається повідомлення з кодом помилки, а черга рухів очищається. Шпиндель повертається в режим обертання.

Крім того, будь-яка з команд M3, M4 або M5 скасовує режим «пошук бажаної орієнтації» або «орієнтація завершена». Це позначається скасуванням активності обох контактів `spindle-orient` та `spindle-locked`.

Вивід «spindle-orient-mode» відображає слово M19 P і має інтерпретуватися наступним чином:

- 0: обертати за годинниковою стрілкою або проти годинникової стрілки для найменшого кутового переміщення

- 1: завжди обертати за годинниковою стрілкою
- 2: завжди обертати проти годинникової стрілки

Його можна використовувати з компонентом HAL `orient`, який забезпечує значення команди PID на основі положення енкодера шпинделя, `кут орієнтації шпинделя` та `режим орієнтації шпинделя`.

5.6.3 Осі та шарнірні штифти та параметри

Ці контакти та параметри створюються модулем «motmod» у режимі реального часу. [У машинах з «тривіальною кінематикою» існує однозначна відповідність між шарнірами та осями.] Вони зчитуються та оновлюються функцією «motion-controller».

Дивіться сторінку довідки з руху *motion(9)* для отримання детальної інформації про виводи та параметри.

5.6.4 iControl

iocontrol — приймає команди вводу/виводу не в реальному часі через NML, взаємодіє з HAL.

Контакти HAL iocontrol вмикаються та вимикаються в режимі, що не є режимом реального часу. Якщо ви маєте суворі вимоги до синхронізації або просто потребуєте більше входів/виходів, розгляньте можливість використання синхронізованих входів/виходів у режимі реального часу, що надаються motion.

5.6.4.1 Піни

* *iocontrol.0.coolant-flood* (біт, вихід) TRUE, коли запитується затоплення охолоджувальної рідини. * *iocontrol.0.coolant-mist* (біт, вихід) TRUE, коли запитується охолоджувальний туман. * *iocontrol.0.emc-enable-in* (біт, in) Повинно бути встановлено значення FALSE (ХИБНІСТЬ), якщо існує зовнішня умова аварійної зупинки. * *iocontrol.0.tool-change* (біт, вихід) TRUE, коли запитується зміна інструменту. * *iocontrol.0.tool-changed* (біт, вхід) Повинно бути встановлене на TRUE після завершення зміни інструменту. * *iocontrol.0.tool-number* (s32, out) Поточний номер інструменту. * *iocontrol.0.tool-prep-number* (s32, out) Номер наступного інструменту з T-слова RS274NGC. * *iocontrol.0.tool-prepare* (біт, вихід) TRUE, коли запитується підготовка інструменту. * *iocontrol.0.tool-prepared* (біт, вхід) Має бути встановлене на TRUE після завершення підготовки інструменту. * *iocontrol.0.user-enable-out* (біт, вихід) ХИБНІСТЬ, коли існує внутрішня умова аварійної зупинки. * *iocontrol.0.user-request-enable* (біт, вихід) TRUE, коли користувач запросив скасування аварійної зупинки.

5.6.5 Налаштування INI

Ряд налаштувань INI доступні як вхідні контакти HAL.

5.6.5.1 Піни

N позначає номер суглоба, *L* позначає літеру осі.

- *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
Контакти min_limit і max_limit для кожної осі працюють безперервно після повернення в початкове положення. Контакти ferror і min_ferror для кожної осі працюють, коли машина увімкнена і не знаходиться в положенні. Контакти max_velocity і max_acceleration для кожної осі зчитуються, коли машина увімкнена і motion_state вільний (повернення в початкове положення або ручне переміщення), але не зчитуються, коли виконується програма (автоматичний режим) або в режимі MDI. Отже, зміна значень контактів під час виконання програми не матиме ефекту, доки програма не буде зупинена і motion_state знову не стане вільним.

- *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
Значення виводів traje_arc_blend вибірково перевіряються безперервно, але зміна значень виводів під час роботи програми може не мати негайного ефекту через чергу 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

Піни планування траєкторії S-подібної кривої (вибірка здійснюється безперервно, можна змінювати під час виконання):

- *ini.traj_planner_type* - (s32, in) [TRAJ]PLANNER_TYPE
- *ini.traj_max_jerk* - (float, in) [TRAJ]MAX_LINEAR_JERK

Штифти обмеження ривків на осі (де *L* is x, y, z, a, b, c, u, v, або w):

- *ini*.L.*max_jerk* - (float, in) [AXIS__L_]MAX_JERK

Штифти обмеження ривка для кожного шарніра (де *N* – номер шарніра 0-8):

- *ini*.N.*max_jerk* - (float, in) [JOINT__N_]MAX_JERK

5.7 Список компонентів HAL

5.7.1 Компоненти

Більшість команд у наведеному нижче списку мають власні сторінки довідки. Деякі з них мають розширені описи, інші — обмежені. З цього списку ви дізнаєтеся, які компоненти існують, і можете використовувати `man` *name* у командному рядку UNIX, щоб отримати додаткову інформацію. Щоб переглянути інформацію на сторінці довідки, у вікні терміналу введіть:

```
man axis
```

В залежності від налаштувань системи UNIX може знадобитися явно вказати розділ сторінки man. Якщо ви не можете знайти сторінку man або її назва вже використовується іншим інструментом UNIX, а сторінка man LinuxCNC знаходиться в іншому розділі, спробуйте явно вказати розділ, наприклад, `man _section-no_ axis`, де *section-no* = 1 для компонентів, що не працюють в режимі реального часу, і 9 для компонентів, що працюють в режимі реального часу.

Note
Див. також розділ «Сторінки довідника» за посиланням:../index.html[головна сторінка документації] або посиланням:../man/[список каталогів]. Для пошуку на сторінках довідника скористайтеся інструментом UNIX `apropos`.

5.7.1.1 Користувацькі інтерфейси (не в режимі реального часу)

axis	Графічний інтерфейс AXIS LinuxCNC (вдосконалений контролер верстата)
axis-remote	Інтерфейс дистанційного керування AXIS
gmoccapy	Зручний графічний інтерфейс користувача LinuxCNC
gscreen	Зручний графічний інтерфейс користувача LinuxCNC
halui	Спостерігайте за пінами HAL та керуйте LinuxCNC через NML
mdro	тільки ручний цифровий зчитування (DRO)
ngcgui	Фреймворк для генерації G-коду в діалоговому режимі на контролері
panelui	
pyngcgui	Реалізація NGCGUI на Python
touchy	AXIS - графічний інтерфейс користувача TOUCHY LinuxCNC

gladevcp	Віртуальна панель керування для LinuxCNC на основі віджетів Glade, Gtk та HAL
gladevcp_demo	GladeVCP — використовується зразками конфігурацій для демонстрації Glade Virtual_demo
gremlin_view	Попередній перегляд графіки G-коду
moveoff_gui	Графічний інтерфейс для компонента Moveoff

pyui	Утиліта для панелей
pyvcp	Віртуальна панель керування для LinuxCNC
pyvcp_demo	Демонстраційний компонент віртуальної панелі керування Python
qtvcp	Віртуальна панель керування на базі Qt

5axisgui	Графічний інтерфейс віртуальної машини Vismach
hbmgui	Графічний інтерфейс віртуальної машини Vismach
hexagui	Графічний інтерфейс віртуальної машини Vismach
lineardelta	Графічний інтерфейс віртуальної машини Vismach
maho600gui	hexagui — графічний інтерфейс віртуальної машини Vismach
max5gui	hexagui — графічний інтерфейс віртуальної машини Vismach
melfagui	Графічний інтерфейс віртуальної машини Vismach
puma560gui	puma560agui - графічний інтерфейс віртуальної машини Vismach
pumagui	Графічний інтерфейс віртуальної машини Vismach
rotarydelta	Графічний інтерфейс віртуальної машини Vismach
scaragui	Графічний інтерфейс віртуальної машини Vismach
xyzac-trt-gui	Графічний інтерфейс віртуальної машини Vismach
xyzbc-trt-gui	Графічний інтерфейс віртуальної машини Vismach
xyzab-tdr-gui	Графічний інтерфейс віртуальної машини Vismach

5.7.1.2 Рух (не в реальному часі)

io	iocontrol - взаємодіє з HAL або G-кодом не в реальному часі
iocontrol	Взаємодіє з HAL або G-кодом не в реальному часі
mdi	Надсилайте команди G-коду з терміналу до запущеного екземпляра LinuxCNC
milltask	Контролер завдань не в реальному часі для LinuxCNC

5.7.1.3 Драйвери обладнання

elbpcom	Зв’язок з Ethernet-картами Mesa
gs2_vfd	Компонент HAL нереального часу для частотних перетворювачів Automation Direct GS2
hy_gt_vfd	Компонент HAL, що не працює в режимі реального часу, для частотних перетворювачів Huanyang серії GT
hy_vfd	Компонент HAL нереального часу для частотних перетворювачів Huanyang
mb2hal	MB2HAL — це універсальний компонент HAL нереального часу для зв’язку з одним або кількома пристроями Modbus. Підтримуються Modbus RTU та Modbus TCP.
mitsub_vfd	Компонент HAL, що не працює в режимі реального часу, для частотних перетворювачів Mitsubishi A500 F500 E500 A500 D700 E700 серії F700 (інші можуть працювати)
monitor-xhc-hb04	Контролює підвісний пульт XHC-HB04 та попереджає про відключення
pi500_vfd	Драйвер Modbus Powtran PI500
pmx485	Зв’язок Modbus з плазмовим різаком Powermax
pmx485-test	Тестування зв’язку Modbus за допомогою плазмового різака Powermax

shuttle	керувати контактами HAL за допомогою пристроїв ShuttleXpress, ShuttlePRO та ShuttlePRO2, виготовлених компанією Contour Design
svd-ps_vfd	Компонент HAL нереального часу для частотних перетворювачів SVD-P(S)
vfdb_vfd	Компонент HAL нереального часу для частотно-регульованих приводів Delta VFD-B
vfs11_vfd	Компонент HAL нереального часу для частотних приводів Toshiba-Schneider VF-S11
wj200_vfd	Драйвер Modbus Hitachi wj200
xhc-hb04	Компонент HAL, що не працює в реальному часі, для підвісного брелока xhc-hb04
xhc-hb04-accels	Застарілий скрипт для бігового колеса
xhc-whb04b-6	Компонент HAL для бездротового USB-пристрою XHC WHB04B-6, що не працює в режимі реального часу

5.7.1.4 Mesa та інші плати вводу/виводу (реального часу)

hal_ppmc	Pico Systems driver для аналогових сервоприводів, PWM та крокових контролерів
hal_bb_gpio	Драйвер для контактів GPIO BeagleBone
hal_parport	Компонент HAL реального часу для зв’язку з одним або кількома паралельними портами ПК
hm2_7i43	Драйвер Mesa Electronics для плати вводу-виводу 7I43 EPP Anything з HostMot2. (Див. сторінку довідки для отримання додаткової інформації)
hm2_7i90	Драйвер LinuxCNC HAL для плати вводу-виводу Mesa Electronics 7I90 EPP Anything з прошивкою HostMot2
hm2_eth	Драйвер LinuxCNC HAL для плат вводу-виводу Mesa Electronics Ethernet Anything з прошивкою HostMot2
hm2_pci	Драйвер Mesa Electronics для плат вводу/виводу 5I20, 5I22, 5I23, 4I65 та 4I68 Anything з прошивкою HostMot2. (Див. сторінку довідки для отримання додаткової інформації)
hm2_rpspi	Драйвер LinuxCNC HAL для плат вводу-виводу Mesa Electronics SPI Anything з прошивкою HostMot2
hm2_spi	Драйвер LinuxCNC HAL для плат вводу-виводу Mesa Electronics SPI Anything з прошивкою HostMot2
hostmot2	Mesa Electronics driver для прошивки HostMot2.
max31855	Підтримка перетворювача термопари в цифровий перетворювач MAX31855 з використанням SPI з розрядним зв’язком
mesa_7i65	Драйвер Mesa Electronics для восьмиосьової сервокарти 7I65. (Див. сторінку довідки для отримання додаткової інформації)
mesa_pktgyro_test	Простий тест PktUART з гіроскопом Microstrain 3DM-GX3-15
mesa_uart	Приклад компонента, що демонструє, як отримати доступ до UART Hostmot2
opto_ac5	Драйвер реального часу для карт opto22 PCI-AC5
pluto_servo	Pluto-P driver та прошивка для FPGA з паралельним портом, для сервоприводів
pluto_step	Pluto-P driver для FPGA паралельного порту, для крокових мікроконтролерів
serport	Драйвер апаратного забезпечення для цифрових бітів введення/виведення послідовних портів 8250 та 16550
setsserial	Утиліта для налаштування параметрів Smart Serial NVRAM
sserial	hostmot2 - Драйвер Smart Serial LinuxCNC HAL для віддалених карт Mesa Electronics HostMot2 Smart-Serial

5.7.1.5 Утиліти (не в режимі реального часу)

hal-histogram	Відображає значення виводу HAL у вигляді гістограми
halcompile	Збірка, компіляція та встановлення компонентів LinuxCNC HAL
halmeter	Спостерігайте за контактами, сигналами та параметрами HAL
halcmd	Керування HAL LinuxCNC з командного рядка
halcmd_twopass	Утилітний скрипт, що використовується під час розбору HAL-файлів. Він дозволяє використовувати кілька команд завантаження для кількох екземплярів одного й того ж компонента.
halreport	Створює звіт про стан HAL
halrmt	Інтерфейс дистанційного керування для LinuxCNC
halrun	Керування HAL LinuxCNC з командного рядка
halsampler	Вибірка даних з HAL у режимі реального часу
halscope	Програмний осцилограф для перегляду сигналів та сигналів HAL у реальному часі
halshow	Показати параметри, контакти та сигнали HAL
halstreamer	Потокова передача файлових даних у HAL у режимі реального часу
haltcl	Керує HAL LinuxCNC з командного рядка за допомогою Tcl
image-to-gcode	Перетворює растрові зображення в G-код
inivar	Запит до INI-файлуе
latency-histogram	Будує гістограму затримки машини
latency-plot	Інший спосіб перегляду показників затримки
latency-test	Тестує затримку системи в реальному часі
linuxcncmkdesktop	Створіть значок на робочому столі для LinuxCNC
modcompile	Утиліта для компіляції драйвера Modbuss
motion-logger	Реєстрація команд руху, надісланих з LinuxCNC
pncconf	Майстер налаштування для карток Mesa
sim_pin	Графічний інтерфейс користувача для відображення та налаштування одного або кількох входів HAL
stepconf	Майстер налаштування для машин з паралельним портом
update_ini	Конвертує INI-файли формату 2.7 у формат 2.8
debuglevel	Встановлює рівень налагодження для частини LinuxCNC, яка не працює в реальному часі
emccalib	Налаштовуйте змінні ini-файлу на льоту за допомогою опції збереження
hal_input	Керування контактами HAL за допомогою будь-якого пристрою введення Linux, включаючи пристрої USB HID
linuxcnc_info	Збирає інформацію про версію LinuxCNC та хост
linuxcnc_module_helper	Надає root-доступ для системного обладнання
linuxcnc_var	Отримує змінні LinuxCNC
linuxcnc	LinuxCNC (Покращений контролер верстата)
linuxcnclcd	Графічний інтерфейс користувача LinuxCNC для відображення символів на РК-дисплеї
linuxcncrsh	Текстовий інтерфейс для керування LinuxCNC через мережу
linuxcncsvr	Дозволяє мережевий доступ до внутрішніх компонентів LinuxCNC через NML
linuxcnctop	Опис статусу Live LinuxCNC
rs274	Автономний інтерпретатор G-коду
schedrmt	Планувальник на базі Telnet для LinuxCNC
setup_designer	Скрипт для налаштування системи для використання Qt Designer
teach-in	Переведіть машину в потрібне положення та запишіть статистикуе
tool_mmap_read	Компонент системи баз даних інструментів (альтернатива класичній таблиці інструментів)

tool_watch	Компонент системи баз даних інструментів (альтернатива класичній таблиці інструментів)
tooledit	Редактор таблиці інструментів

5.7.1.6 Обробка сигналів (реальний час)

and2	Двовходовий елемент I. Щоб вихід був істинним, обидва входи повинні бути true. (and2)
bitwise	Обчислює різні побітові операції над двома вхідними значеннями
dbounce	Фільтрування шумних цифрових входів Details
debounce	Фільтрувати шумні цифрові входи Деталі description
demux	Виберіть один з кількох вихідних контактів цілим числом та/або окремими бітами
edge	Детектор країв
estop_latch	Фіксатор аварійної зупинки
flipflop	D-подібний тригер
logic	Загальний логічний функціональний компонент
lut5	5-входова логічна функція на основі таблиці пошуку description
match8	8-бітний двійковий детектор збігів
multiclick	Single-, double-, triple-, та детектор чотириразового клацання
multiswitch	Перемикає між заданою кількістю вихідних бітів
not	Інвертор
oneshot	Генератор одноразових імпульсів
or2	Двовходовий вентиль АБО
reset	Скидає сигнал вводу-виводу
select8	8-бітний двійковий детектор збігів.
tof	Таймер IEC TOF - затримка спадаючого фронту сигналу
toggle	Вмикання, вимикання за допомогою кнопок швидкого натискання
toggle2nist	Кнопка перемикання на логіку nist
ton	Таймер IEC TON - затримка наростаючого фронту сигналу
timedelay	Еквівалент реле із затримкою часу.
tp	IEC TP timer - генерувати високий імпульс визначеної тривалості на наростаючому фронті
tristate_bit	Розміщує сигнал на контакті вводу/виводу лише тоді, коли він увімкнений, подібно до тристабільного буфера в електроніці
tristate_float	Розміщує сигнал на контакті вводу/виводу лише тоді, коли він увімкнений, подібно до тристабільного буфера в електроніці
xor2	Двовходовий вентиль XOR (ексклюзивне АБО)

abs_s32	Обчислює абсолютне значення та знак цілочисельного вхідного сигналу
abs_s64	Обчислює абсолютне значення та знак 64-бітного цілочисельного вхідного сигналу
abs	Обчислює абсолютне значення та знак вхідного сигналу з плаваючою комою
biquad	Біквадровий БІХ-фільтр
blend	Виконайте лінійну інтерполяцію між двома значеннями
comp	Двовходовий компаратор з гістерезисом
counter	Рахує вхідні імпульси (застаріло). Використовуйте компонент encoder.
ddt	Обчислює похідну вхідної функції.
deadzone	Повертає центр, якщо він знаходиться в межах порогового значення.
div2	Частка двох вхідних даних з плаваючою комою.
hypot	Калькулятор гіпотенузи (евклідової відстані) з трьома входами.
ilowpass	Низькочастотний фільтр з цілочисельними входами та виходами

integ	Інтегратор
invert	Обчислює обернену величину вхідного сигналу.
filter_kalman	Одновимірний фільтр Калмана, також відомий як лінійна квадратична оцінка (LQE)
knob2float	Перетворює лічильники (ймовірно, з кодувальника) на значення з плаваючою комою.
led_dim	Компонент HAL для димування світлодіодів
lowpass	Низькочастотний фільтр
limit1	Обмежує вихідний сигнал значенням між мінімумом і максимумом. примітка: [Коли вхідним сигналом є позиція, це означає, що «позиція» обмежена.]
limit2	Обмежує вихідний сигнал значеннями між min та max. Обмежує швидкість його зміни значеннями менше maxv за секунду. примітка: [Коли вхідним сигналом є позиція, це означає, що «позиція» та «швидкість» обмежені.]
limit3	Обмежте вихідний сигнал, щоб він знаходився в межах між мінімальним і максимальним значеннями. Обмежте швидкість зміни сигналу до менше ніж maxv за секунду. Обмежте другу похідну до менше ніж MaxA за секунду в квадраті. Примітка: [Коли вхідним сигналом є положення, це означає, що обмежуються «положення», «швидкість» і «прискорення».].
lincurve	Одновимірна таблиця пошуку
maj3	Обчисліть більшість із 3 вхідних даних
minmax	Відстежує мінімальні та максимальні значення вхідних сигналів та виходів.
mult2	Добуток двох вхідних даних.
mux16	Виберіть одне з 16 вхідних значень (мультиплексор).
mux2	Виберіть одне з двох вхідних значень (мультиплексор).
mux4	Виберіть одне з чотирьох вхідних значень (мультиплексор).
mux8	Виберіть одне з восьми вхідних значень (мультиплексор).
mux_generic	Виберіть одне з кількох вхідних значень (мультиплексор).
near	Визначте, чи два значення приблизно рівні.
offset	Додає зміщення до вхідного значення та віднімає його від значення зворотного зв’язку.
safety_latch	фіксатор для сигналів помилок
sample_hold	Зразок та зберігання.
scaled_s32_sums	Сума чотирьох вхідних даних (кожен зі шкалою)
scale	Застосовує масштаб та зміщення до вхідних даних.
sum2	Сума двох вхідних даних (кожен з коефіцієнтом підсилення) та зміщення.
time	Накопичені дані таймера виконання за час ГГ:ХХ:СС для «активного» входу.
timedelta	Компонент, який вимірює поведінку часу планування потоків.
updown	Лічи у напрямку збільшення або зменшення, з додатковими обмеженнями та поведінкою циклу.
wcomp	Віконний компаратор.
watchdog	Контролюйте від одного до тридцяти двох входів на наявність «серцебиття».
weighted_sum	Перетворити групу бітів на ціле число.
xhc_hb04_util	утиліта для зручності xhc-hb04

5.7.1.7 Генерація сигналів (реального часу)

charge_pump	Створює прямокутну хвилю для входу «зарядового насоса» деяких плат контролера.

pwmgen	Програмна генерація PWM/PDM, див. description.
siggen	Генератор сигналів, див. description.
sim_encoder	Імітований квадратурний кодер, див. description.
stepgen	Програмна генерація крокових імпульсів, див. description.

bin2gray	Перетворює число у представлення за кодом Грея
bitmerge	Перетворює окремі вхідні біти на беззнакове число 32
bitslice	Перетворює вхідний сигнал без знаку 32 на окремі біти
conv_bit_float	Перетворює з біта на число з плаваючою комою
conv_bit_s32	Перетворює з біт в s32
conv_bit_u32	Перетворює з біт на u32
conv_float_s32	Перетворює з числа з плаваючою комою на s32
conv_float_u32	Перетворює з числа з комою в u32
conv_s32_bit	Конвертує з s32 в біт
conv_s32_float	Перетворює з s32 на число з плаваючою комою
conv_s32_u32	Конвертує з s32 в u32
conv_u32_bit	Конвертує з u32 в біт
conv_u32_float	Конвертує з u32 у число з плаваючою комою
conv_u32_s32	Конвертує з u32 в s32
conv_bit_s64	Перетворити значення з біта на s64
conv_bit_u64	Перетворити значення з біта на u64
conv_float_s64	Перетворити значення з числа з плаваючою комою на s64
conv_float_u64	Перетворити значення з float на u64
conv_s32_s64	Перетворити значення з s32 на s64
conv_s32_u64	Конвертувати значення з s32 в u64
conv_s64_bit	Перетворити значення з s64 на біт
conv_s64_float	Перетворити значення з s64 на число з плаваючою комою
conv_s64_s32	Перетворити значення з s64 на s32
conv_s64_u32	Конвертувати значення з s64 в u32
conv_s64_u64	Конвертувати значення з s64 в u64
conv_u32_s64	Конвертувати значення з u32 в s64
conv_u32_u64	Перетворити значення з u32 на u64
conv_u64_bit	Перетворити значення з u64 на біт
conv_u64_float	Перетворити значення з u64 на число з плаваючою комою
conv_u64_s32	Перетворити значення з u64 на s32
conv_u64_s64	Перетворити значення з u64 на s64
conv_u64_u32	Перетворити значення з u64 на u32
gray2bin	Перетворює вхідний код Грея на двійковий

5.7.1.8 Кінематика (реальний час)

corexy_by_hal	Кінематика CoreXY
differential	Кінематика диференціальної передачі
gantry	Компонент HAL для LinuxCNC для керування кількома з’єднаннями з однієї осі
gantrykins	Кінематичний модуль, який пов’язує одну вісь з кількома суглобами.
genhexkins	Надає шість ступенів свободи в положенні та орієнтації (XYZABC). Розташування двигунів визначається під час компіляції.
genserkins	Кінематика, яка може моделювати загальний маніпулятор із послідовним з’єднанням до 6 кутових шарнірів.
gentrivkins	Відповідність 1:1 між шарнірами та осями. Більшість стандартних фрезерних та токарних верстатів використовують модуль тривіальної кінематики.
kins	Кінематичні визначення для LinuxCNC.

lineardeltakins	Кінематика лінійного дельта-робота
matrixkins	Калібрована кінематика для 3-осьового верстатаs
maxkins	Кінематика настільного 5-осьового фрезерного верстата «max» з похилою головкою (вісь B) та горизонтальним поворотним механізмом, встановленим на столі (вісь C). Забезпечує рух UVW в оберненій системі координат.
millturn	Перемикальна кінематика для токарно-фрезерного верстата
pentakins	
pumakins	Кінематика роботів типу PUMA.
rosekins	Кінематика двигуна Rose
rotatekins	Осі X та Y повернуті на 45 градусів порівняно з суглобами 0 та 1.
scarakins	Кінематика роботів типу SCARA.
tripodkins	Суглоби представляють відстань контрольованої точки від трьох заздалегідь визначених місць (двигунів), що дає три ступені свободи в положенні (XYZ).
userkins	Шаблон для кінематики, створеної користувачем
xyzab_tdr_kins	Перемикальна кінематика для 5-осьового верстата з поворотним столом A та B
xyzacb_trsrn	Кінематика, що перемикається, для 6-осьового верстата з поворотним столом C, поворотним шпинделем B та обертальним шпинделем A
xyzbca_trsrn	Кінематика, що перемикається, для 6-осьового верстата з поворотним столом B, поворотним шпинделем C та обертальним шпинделем A

5.7.1.9 Керування рухом (у режимі реального часу)

feedcomp	Помножте вхідне значення на відношення швидкості струму до швидкості подачі.
homecomp	Шаблон модуля самонаведення
limit_axis	Обмеження осей на основі динамічного діапазону
motion	Приймає команди руху NML, взаємодіє з HAL у режимі реального часу
simple_tp	Цей компонент є простим планувальником траєкторії для однієї осі, таким самим, як той, що використовується для бігового руху в LinuxCNC.
tpcomp	Каркас модуля планування траєкторії (tp)

5.7.1.10 Керування двигуном (у режимі реального часу)

at_pid	Пропорційний/інтегральний/диференціальний контролер з автоматичним налаштуванням.
bldc	Компонент керування BLDC та сервоприводом змінного струму
clarke2	Двовхідна версія перетворення Кларка
clarke3	Перетворення Кларка (з 3-фазного на декартове)
clarkeinv	Зворотне перетворення Кларка
encoder	Програмне підрахунок сигналів квадратурного енкодера, див. description.
pid	Пропорційний/інтегральний/диференціальний контролер, description.
pwmgen	Програмна генерація PWM/PDM, див. description.
stepgen	Програмна генерація крокових імпульсів, див. description.

5.7.1.11 Моделювання/Тестування

axistest	Використовується для тестування осі. Використовується в PnCConf.

rtapi_app	створює імітацію реального середовища
sim-torch	Модельований плазмовий торкh
sim_axis_hardware	Компонент для імітації дому та кінцевого вимикачаs
sim_home_switch	Симулятор домашнього перемикача
sim_matrix_kb	конвертувати вхідні дані HAL-пінів у код клавішіs
sim_parport	Компонент для імітації виводів компонента hal_parport
sim_spindle	Моделювання шпинделя з індексними імпульсамие
simulate_probe	імітувати вхід зонда

5.7.1.12 Інше (у реальному часі)

anglejog	Зсув двох осей (або суглобів) під кутом
classicladder	Програмний PLC реального часу на основі логіки сходинок. Див. розділ ClassicLadder для отримання додаткової інформації.
charge_pump	Створює прямокутну хвилю для входу «зарядового насоса» деяких плат контролера.
encoder_ratio	Електронний механізм для синхронізації двох осей.
enum	Перерахування цілих чисел у біти
eoffset_per_angle	Обчислення зовнішнього зміщення на кут
gladevcp (У режимі реального часу)	відображає віртуальні панелі керування, побудовані за допомогою GTK / GLADE
histobins	Утиліта гістограмних контейнерів для скриптів/hal-histogram
joyhandle	Встановлює нелінійні рухи джойстика, зони нечутливості та масштаби.
latencybins	Утиліта Comp для скриптів/гістограми затримки
message	Відобразити повідомлення
moveoff	Компонент для зміщень лише для HAL
raster	Вихідна потужність лазера базується на попередньо запрограмованих даних растрування
sampler	Вибірка даних з HAL у режимі реального часу.
siggen	Генератор сигналів, див. description.
sphereprobe	Дослідіть уявну півкулю.
threads	Створює потоки HAL у режимі реального часу.
threadtest	Компонент для тестування поведінки потоків.
steptest	Використовується StepConf для перевірки значень прискорення та швидкості для осі.
streamer	Потокова передача файлових даних у HAL у режимі реального часу.
supply	Встановити вихідні контакти значеннями з параметрів (застаріло).

laserpower	Масштабує вихідну потужність лазера на основі вхідної потужності швидкості та відстані, що залишилася
lcd	Потокова передача даних HAL на LCD-екран
matrix_kb	Перетворити цілі числа на виводи HAL. За потреби просканувати матрицю портів вводу/виводу для створення цих цілих чисел.

gearchange	Виберіть один з двох діапазонів швидкостей.
orient	Забезпечити вхід команди PID для режиму орієнтації на основі поточного положення шпинделя, цільового кута та модуля орієнтаціїе
spindle	Керування шпинделем з різним прискоренням та уповільненням, а також додатковим масштабуванням перемикання передач
spindle_monitor	Виявлення швидкості обертання шпинделя на високій та низькій швидкості

carousel	Орієнтувати карусель змінника інструментів, використовуючи різні схеми кодуванняs
hal_manualtoolchange	Компонент HAL, що не працює в режимі реального часу, для ручної зміни інструментів.
thc	Контроль висоти пальника за допомогою плати Mesa THC або будь-якого аналогового входу для швидкості
thcud	Вхід керування висотою пальника вгору/вниз
ohmic	Компонент HAL для LinuxCNC, який використовує Mesa THCAD (аналого-цифрову карту) для омічного зондування
plasmac	Контролер плазмового різака

5.7.2 Виклики 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 Виклики RTAPI

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 Описи компонентів HAL

У цьому розділі наведено детальну інформацію про основні функції LinuxCNC, які вимагають точного часу для

- генерація сигналів, які інтерпретуються апаратним забезпеченням (наприклад, двигунами) або
- для інтерпретації сигналів, що надсилаються апаратним забезпеченням (наприклад, кодерами).

5.8.1 StepGen

Цей компонент забезпечує програмне формування імпульсів кроку у відповідь на команди положення або швидкості. У режимі положення він має вбудований попередньо налаштований контур положення, тому налаштування PID не потрібне. У режимі швидкості він приводить в дію двигун із заданою швидкістю, дотримуючись обмежень швидкості та прискорення. Це компонент, що працює тільки в режимі реального часу, і залежно від швидкості процесора тощо, здатний досягати максимальної частоти імпульсів від 10 кГц до, можливо, 50 кГц. Блок-схема генератора імпульсів показує три блок-схеми, кожна з яких є генератором імпульсів з одним кроком. Перша схема призначена для крокового типу «0» (крок і напрямок). Друга — для крокового типу «1» (вгору/вниз або псевдо-PWM), а третя — для крокових типів від 2 до 14 (різні крокові схеми). Перші дві схеми показують управління в режимі положення, а третя — в режимі швидкості. Режим управління і крокoвий тип встановлюються незалежно, і можна вибрати будь-яку комбінацію.

Блок-схема генератора крокових імпульсів у режимі позиціонування

:images/stepgen-block-diag.png

Завантаження компонента stepgen

```
halcmd: loadrt stepgen step_type=<type-array> [ctrl_type=<ctrl_array>]
```

<type-array>
— це послідовність десяткових цілих чисел, розділених комами. Кожне число призводить до завантаження генератора однокрокових імпульсів, значення якого визначає тип кроку.

<ctrl_array>
— це послідовність символів «p» або «v», розділених комами, для позначення положення або режиму швидкості.

ctrl_type
необов’язковий, якщо його пропустити, усі генератори кроків будуть у режимі позиціонування.

Наприклад:

```
halcmd: loadrt stepgen step_type=0,0,2 ctrl_type=p,p,v
```

Встановить три генератори кроків. Перші два використовують тип кроку «0» (крок і напрямок) і працюють у режимі положення. Останній використовує тип кроку «2» (квадратура) і працює у режимі швидкості. Значенням за замовчуванням для «<config-array>» є «0,0,0», що встановить три генератори типу «0» (крок/напрямок). Максимальна кількість генераторів кроків становить 8 (як визначено MAX_CHAN у stepgen.c). Кожен генератор є незалежним, але всі вони оновлюються за допомогою однієї і тієї ж функції (функцій) одночасно. У наведених нижче описах *<chan>* є номером конкретного генератора. Перший генератор має номер 0.

Вивантаження компонента stepgen

```
halcmd: unloadrt stepgen
```

5.8.1.1 Піни

Від вибраного типу кроку та типу керування.

- (float) `stepgen.`__<chan>__.position-cmd` – Бажане положення двигуна, в одиницях положення (лише в режимі положення).
- (float) `stepgen.`__<chan>__.velocity-cmd` – Бажана швидкість двигуна, в одиницях положення за секунду (лише в режимі швидкості).
- s32) `stepgen.`__<chan>__.counts` – Позиція зворотного зв'язку в лічильниках, оновлюється за допомогою *capture_position()*.
- (float) `stepgen.`__<chan>__.position-fb` – Позиція зворотного зв'язку в одиницях позиції, оновлюється за допомогою *capture_position()*.
- (bit) `stepgen.`__<chan>__.enable` - Вмикає вихідні кроки — якщо значення false, кроки не генеруються.
- (bit) `stepgen.`__<chan>__.step` - Вихід ступінчастого імпульсу (лише тип ступінчастого імпульсу 0).
- (bit) `stepgen.`__<chan>__.dir` - Вихід напрямку (лише тип кроку 0).
- (bit) `stepgen.`__<chan>__.up` - Вихід псевдо-PWM UP (лише тип кроку 1).
- (bit) `stepgen.`__<chan>__.down` - Вихід псевдо-PWM ВНИЗ (лише тип кроку 1).
- (bit) `stepgen.`__<chan>__.phase-A` - Вихід фази A (лише типи ступенів 2-14).
- (bit) `stepgen.`__<chan>__.phase-B` - Вихід фази B (лише для типів ступенів 2-14).
- (bit) `stepgen.`__<chan>__.phase-C` - Вихід фази C (лише типи ступенів 3-14).
- (bit) `stepgen.`__<chan>__.phase-D` - Вихід фази D (лише для типів ступенів 5-14).
- (bit) `stepgen.`__<chan>__.phase-E` - Вихід фази E (лише для типів ступенів 11-14).

5.8.1.2 Параметри

* (float) `stepgen.`__<chan>__.position-scale` - Кроки на одиницю позиції. Цей параметр використовується як для виходу, так і для зворотного зв'язку. * (float) `stepgen.`__<chan>__.maxvel` - Максимальна швидкість, в одиницях положення за секунду. Якщо 0,0, не має значення. * (float) `stepgen.`__<chan>__ - Максимальна швидкість розгону/уповільнення, в одиницях позицій за секунду в квадраті. Якщо 0,0, не має значення. * (float) `stepgen.`__<chan>__.frequency` – Поточна швидкість кроків, у кроках за секунду. * (float) `stepgen.`__<chan>__.steplen` – Тривалість імпульсу кроку (тип кроку 0 та 1) або мінімальний час у заданому стані (типи кроків 2-14), у наносекундах. * (float) `stepgen.`__<ch - Мінімальний інтервал між двома імпульсами кроку (тільки типи кроку 0 і 1) в наносекундах. Встановіть значення 0, щоб увімкнути функцію «doublefreq» генератора кроку. Щоб використовувати «doublefreq», необхідно увімкнути функцію parport reset function. * (float) `stepgen.`__<chan>__.dirsetu - Мінімальний час від зміни напрямку до початку наступного імпульсу кроку (лише для типу кроку 0), у наносекундах. * (float) `stepgen.`__<chan>__.dirhold` - Мінімальний час від кінця ступінчастого імпульсу до зміни напрямку (лише для кроку типу 0), у наносекундах. * (float) `stepgen.`__<chan>__.dirdelay` - Мінімальний час від будь-якого кроку до кроку в протилежному напрямку (лише для типів кроків 1-14), у наносекундах. * (s32) `stepgen.`__<chan>__.rawcounts` - Кількість необроблених відгуків, оновлена за допомогою *make_pulses()*.

У режимі позиціонування значення maxvel і maxaccel використовуються внутрішнім контуром позиціонування, щоб уникнути генерації послідовностей імпульсів, за якими двигун не може слідувати. При встановленні значень, що відповідають двигуну, навіть велика миттєва зміна заданої позиції призведе до плавного трапецієподібного переміщення до нового місця. Алгоритм працює шляхом вимірювання як похибки позиції, так і похибки швидкості, а також обчислення прискорення, яке намагається одночасно зменшити обидві похибки до нуля. Для отримання більш детальної інформації, включаючи вміст поля «контрольне рівняння», зверніться до коду.

У режимі швидкості maxvel є простим обмеженням, яке застосовується до заданої швидкості, а maxaccel використовується для підвищення фактичної частоти, якщо задана швидкість різко змінюється. Як і в режимі положення, правильні значення цих параметрів забезпечують, що двигун може слідувати за генерованим імпульсним поїздом.

5.8.1.3 Типи кроків

Генератор кроків підтримує 15 різних *послідовностей кроків*:

Тип кроку 0 Тип кроку 0 є стандартним типом кроку та напрямку. При налаштуванні на тип кроку 0 існують чотири додаткові параметри, які визначають точний час сигналів кроку та напрямку. На наступному малюнку показано значення цих параметрів. Параметри вказані в наносекундах, але будуть округлені до цілого кратного періоду потоку для потоку, який викликає *make_pulses()*. Наприклад, якщо *make_pulses()* викликається кожні 16 мкс, а steplen дорівнює 20000, то імпульси кроку будуть мати тривалість 2 x 16 = 32 мкс. Значенням за замовчуванням для всіх чотирьох параметрів є 1 нс, але автоматичне округлення починає діяти під час першого виконання коду. Оскільки один крок вимагає *steplen* нс високого і *stepspace* нс низького рівня, максимальна частота дорівнює 1000000000, поділеному на (*steplen* *stepspace*). Якщо *maxfreq* встановлено вище цього обмеження, воно буде автоматично знижено. Якщо *maxfreq* дорівнює нулю, воно залишиться нульовим, але вихідна частота все одно буде обмежена.

Під час використання драйвера паралельного порту частоту кроку можна подвоїти за допомогою функції parport reset разом із налаштуванням *doublefreq* StepGen.

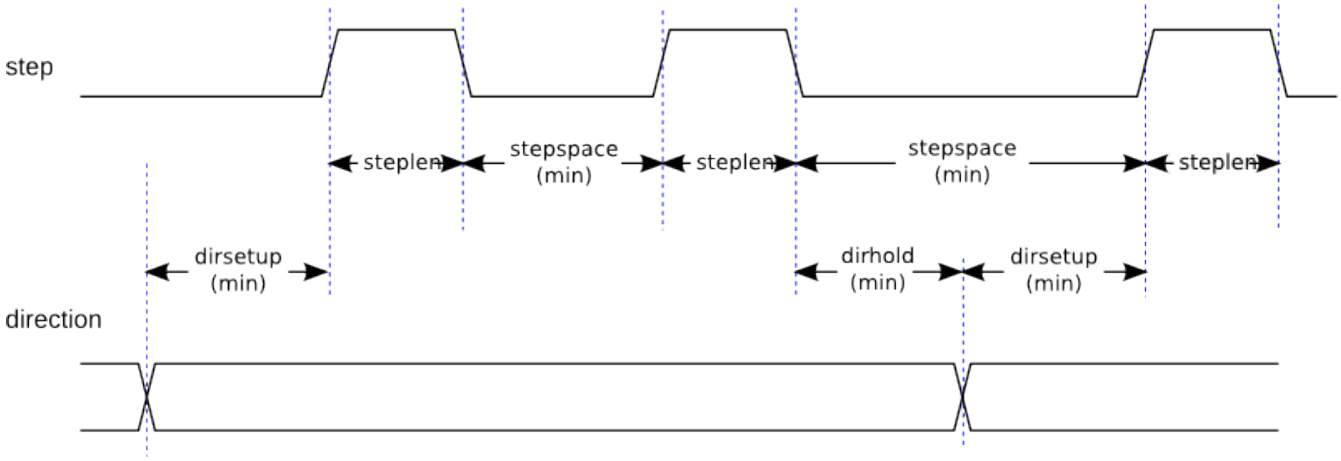


Figure 5.19: Час кроку та напрямку

Тип кроку 1 Тип кроку 1 має два виходи: вгору і вниз. Імпульси з'являються на одному або іншому, залежно від напрямку руху. Кожен імпульс має тривалість *steplen* нс, а імпульси розділені між собою щонайменше на *stepspace* нс. Максимальна частота така сама, як і для типу кроку 0. Якщо *maxfreq* встановлено вище за обмеження, воно буде знижено. Якщо *maxfreq* дорівнює нулю, воно залишиться нульовим, але вихідна частота все одно буде обмежена.



Warning
Не використовуйте функцію скидання парпорту з типами кроків 2–14. Це може призвести до неочікуваних результатів.

Тип кроку 2 - 14 Типи кроків від 2 до 14 базуються на стані і мають від двох до п’яти виходів. На кожному кроці лічильник стану збільшується або зменшується. Двофазний, трифазний, чотирифазний і п’ятифазний типи показують вихідні схеми як функцію лічильника стану. Максимальна частота становить 1000000000, поділену на *steplen*, і, як і в інших режимах, *maxfreq* буде знижено, якщо воно перевищує обмеження.

Дво- та трифазні типи ступінчастої системи Типи сходинок: дво- та трифазні

Чотирифазні типи ступінчастого перемикання Типи сходинок: чотирифазні

П’ятифазні типи кроків Типи сходинок: п’ятифазні

5.8.1.4 Функції

Компонент експортує три функції. Кожна функція діє на всі генератори крокових імпульсів — запуск різних генераторів у різних потоках не підтримується.

- (funct) `stepgen.make-pulses` – Високошвидкісна функція для генерації та підрахунку імпульсів (без плаваючої коми).
- (funct) `stepgen.update-freq` - Функція низької швидкості виконує перетворення положення в швидкість, масштабування та обмеження.
- (funct) `stepgen.capture-position` - Функція низької швидкості для зворотного зв’язку, оновлення фіксаторів та масштабування положення.

Високошвидкісна функція «stepgen.make-pulses» повинна виконуватися в дуже швидкому потоці, від 10 до 50 мкс, залежно від можливостей комп’ютера. Період цього потоку визначає максимальну частоту кроку, оскільки «steplen», «stepspace», «dirsetup», «dirhold» і «dirdelay» округлюються до цілого кратного періоду потоку в наносекундах. Дві інші функції можуть викликатися з набагато меншою частотою.

5.8.2 PWMgen

Цей компонент забезпечує програмне формування сигналів PWM (імпульсно-широтно-модульованих) та PDM (імпульсно-щільнісно-модульованих). Це виключно компонент реального часу, який залежно від швидкості процесора тощо здатний формувати сигнали PWM з частотою від кількох сотень герц із досить високою роздільною здатністю до, можливо, 10 кГц з обмеженою роздільною здатністю.

Завантаження PWMgen

```
loadrt pwmgen output_type=<config-array>
```

«<config-array>» — це серія десяткових чисел, розділених комами. Кожне число викликає завантажен одного генератора PWM, а значення числа визначає тип виходу. У наведеному нижче прикладі буде встановлено три генератори PWM. Стандартне значення відсутнє, якщо «<config-array>» не вказано, генератори PWM не встановлюватимуться. Максимальна кількість генераторів частоти — 8 (як визначено MAX_CHAN у pwmgen.c). Кожен генератор є незалежним, але всі вони оновлюються одночасно за допомогою однієї і тієї ж функції (функцій). У наведених нижче описах «<*chan*>» — це номер конкретного генератора. Перший генератор має номер 0.

Приклад завантаження PWMgen

```
loadrt pwmgen output_type=0,1,2
```

Встановить три генератори PWM. Перший буде використовувати вихід типу *0* (тільки PWM), наступний буде використовувати вихід типу 1 (PWM і напрямок), а третій буде використовувати вихід типу 2 (ВГОРУ і ВНИЗ). Немає значення за замовчуванням, якщо <*config-array*> не вказано, жоден генератор PWM не буде встановлено. Максимальна кількість генераторів частоти становить 8 (як визначено MAX_CHAN у pwmgen.c). Кожен генератор є незалежним, але всі вони оновлюються за допомогою однієї і тієї ж функції (функцій) одночасно. У наведених нижче описах <*chan*> є номером конкретного генератора. Нумерація генераторів PWM починається з 0.

Розвантаження PWMgen

```
unloadrt pwmgen
```

5.8.2.1 Типи виводу

PWM-генератор підтримує три різні «типи виходу».

- *Output type 0* - Тільки вихідний контакт PWM. Приймаються лише позитивні команди, від'ємні значення вважаються нулем (і на них впливатиме параметр *min-dc*, якщо він не дорівнює нулю).
- «Тип виходу 1» — PWM/PDM і виводи напрямку. Позитивні та негативні входи будуть виводитися як позитивні та негативні PWM. Контакт напрямку є хибним для позитивних команд і істинним для негативних команд. Якщо для управління потрібно позитивний PWM як для CW, так і для CCW, використовуйте компонент abs, щоб перетворити сигнал PWM на позитивне значення, коли вводиться негативний вхід.
- «Тип виходу 2» — виводи UP і DOWN. Для позитивних команд сигнал ШІМ з'являється на виводі up, а вивід down залишається неправдивим. Для негативних команд сигнал PWM з'являється на виводі down, а вивід up залишається неправдивим. Тип виходу 2 підходить для керування більшістю H-мостів.

5.8.2.2 Піни

Кожен PWM-генератор матиме такі контакти:

- (float) pwmgen.__<chan>__.value - Значення команди, у довільних одиницях. Масштабуватиметься параметром «scale» (див. нижче).
- (bit) pwmgen.__<chan>__.enable - Вмикає або вимикає виходи PWM-генератора.

Кожен PWM-генератор також матиме деякі з цих контактів, залежно від вибраного типу виходу:

- (bit) pwmgen.__<chan>__.pwm - PWM (або PDM) вихід (лише типи виводів 0 та 1).
- (bit) pwmgen.__<chan>__.dir - Вихід напрямку (лише вихід типу 1).
- (bit) pwmgen.__<chan>__.up - PWM/PDM вихід для позитивного вхідного значення (лише вихід типу 2).
- (bit) pwmgen.__<chan>__.down - PWM/PDM вихід для від'ємного вхідного значення (лише вихід типу 2).

5.8.2.3 Параметри

- (float) pwmgen.`__<chan>__.scale` - Коефіцієнт масштабування для перетворення значення з довільних одиниць в робочий цикл. Наприклад, якщо масштаб встановлено на 4000, а вхідне значення, передане до pwmgen.`__<chan>__.value`, дорівнює 4000, то робочий цикл буде 100% (завжди увімкнено). Якщо значення дорівнює 2000, то це буде 50% прямокутна хвиля 25 Гц.
- (float) pwmgen.`__<chan>__.pwm-freq` - Бажана частота PWM, в Гц. Якщо 0,0, генерує PDM замість PWM. Якщо встановлено вище внутрішніх обмежень, наступний виклик *update_freq()* встановить його на внутрішнє обмеження. Якщо значення відмінне від нуля, а *dither* має значення false, наступний виклик *update_freq()* встановить його на найближче ціле число, кратне періоду функції *make_pulses()*.
- (біт) pwmgen.`__<chan>__.dither-pwm` - Якщо true, вмикає дзеркальне відображення для досягнення середніх частот PWM або робочих циклів, які неможливо отримати за допомогою чистого PWM. Якщо false, частота PWM і робочий цикл будуть округлені до значень, які можна досягти точно.
- (float) pwmgen.`__<chan>__.min-dc` – Мінімальний робочий цикл, між 0.0 та 1.0 (робочий цикл дорівнюватиме нулю, коли його вимкнено, незалежно від цього налаштування).
- (float) pwmgen.`__<chan>__.max-dc` - Максимальний робочий цикл, між 0,0 та 1,0.
- (float) pwmgen.`__<chan>__.curr-dc` - Поточний робочий цикл - після всіх обмежень та округлень (лише читання).

5.8.2.4 Функції

Компонент експортує дві функції. Кожна функція діє на всі генератори PWM — запуск різних генераторів у різних потоках не підтримується.

- (funct) pwmgen.make-pulses - Високошвидкісна функція для генерації PWM-хвиль (без плаваючої коми). Високошвидкісна функція pwmgen.make-pulses повинна виконуватися в базовому (найшвидш потоці, від 10 до 50 мкс, залежно від можливостей комп'ютера. Період цього потоку визначає максимальну несучу частоту PWM, а також роздільну здатність сигналів PWM або PDM. Якщо базовий потік становить 50 000 нс, то кожні 50 мкс модуль вирішує, чи настав час змінити стан виходу. При 50% робочому циклі і частоті ШІМ 25 Гц це означає, що вихід змінює стан кожні (1/25) с / 50 мкс * 50% = 400 ітерацій. Це також означає, що ви маєте 800 можливих значень робочого циклу (без дрожання).
- (функція) pwmgen.update - Функція низької швидкості для масштабування та обмеження значення, а також обробки інших параметрів. Це функція модуля, яка виконує більш складні математичні обчислення, щоб визначити, скільки базових періодів вихідний сигнал повинен бути високим, а скільки - низьким.

5.8.3 Енкодер

Цей компонент забезпечує програмне підрахування сигналів від квадратурних (або одноімпульсних) енкодерів. Це виключно компонент реального часу, який залежно від швидкості процесора, затримки тощо здатний забезпечувати максимальну швидкість підрахунку від 10 кГц до, можливо, 50 кГц.

Базова частота повинна становити 1/2 від швидкості, щоб врахувати шум і коливання частоти. Наприклад, якщо на шпинделі встановлено датчик з частотою 100 імпульсів на оберт, а максимальна частота обертання становить 3000 об/хв, максимальна базова частота повинна становити 25 мкс. Датчик з частотою 100 імпульсів на оберт матиме 400 імпульсів. Швидкість шпинделя 3000 об/хв = 50 об/с (обертів за секунду). 400 * 50 = 20 000 імпульсів за секунду або 50 мкс між імпульсами.

Блок-схема лічильника енкодеру — це блок-схема одного каналу лічильника енкодеру.

Блок-схема лічильника енкодерів

:images/encoder-block-diag.png

Завантаження кодера

```
halcmd: loadrt encoder [num_chan=<counters>]
```

<counters> — кількість лічильників кодера, які ви хочете встановити. Якщо *num_chan* не вказано, буде встановлено три лічильники. Максимальна кількість лічильників — 8 (як визначено MAX_CHAN у encoder.c). Кожен лічильник є незалежним, але всі вони оновлюються одночасно за допомогою однієї і тієї ж функції (функцій). У наведених нижче описах *<chan>* — це номер конкретного лічильника. Перший лічильник має номер 0.

Розвантаження енкодера

```
halcmd: unloadrt encoder
```

5.8.3.1 Піни

- `encoder._<chan>_.counter-mode` (біт, ввід/вивід) (за замовчуванням: FALSE) — увімкнення режиму лічильника. Якщо значення true, лічильник підраховує кожен передній фронт вхідного сигналу фази A, ігноруючи значення фази B. Це корисно для підрахунку вихідного сигналу одноканального (неквадратурного) датчика. Якщо значення false, підрахунок відбувається в квадратурному режимі.
- `encoder._<chan>_.missing-teeth` (s32, In) (за замовчуванням: 0) - Увімкнення використання індексу відсутніх зубців. Це дозволяє одному виводу IO надавати інформацію як про положення, так і про індекс. Якщо колесо енкодера має 58 зубців, з яких два відсутні, розташовані так, ніби їх 60 (що є типовим для автомобільних датчиків колінчастого вала), то шкала положення повинна бути встановлена на 60, а відсутні зубці - на 2. Щоб використовувати цей режим, режим лічильника повинен бути встановлений на true. Цей режим буде працювати для різьблення на токарному верстаті, але не для жорсткого нарізування різьби.
- `encoder._<chan>_.counts` (s32, Out) - Позиція в лічильниках кодера.
- `encoder._<chan>_.counts-latched` (s32, Out) - На даний момент не використовується.
- `encoder._<chan>_.index-enable` (біт, ввід/вивід) - Якщо True, `counts` і `position` скидаються до нуля при наступному передньому фронті фази Z.
  Одночасно `index-enable` скидається до нуля, щоб вказати, що відбувся передній фронт. Контакт `index-enable` є двонаправленим. Якщо `index-enable` має значення False, канал фази Z кодера ігнорується, а лічильник працює в звичайному режимі. Драйвер кодера ніколи не встановлює для `index-enable` значення True. Однак деякі інші компоненти можуть це робити.
- `encoder._<chan>_.latch-falling` (біт, In) (за замовчуванням: TRUE) - Наразі не використовується.
- `encoder._<chan>_.latch-input` (біт, In) (за замовчуванням: TRUE) - Наразі не використовується.
- `encoder._<chan>_.latch-rising` (bit, In) - На даний момент не використовується.
- `encoder._<chan>_.min-speed-estimate` (float, in) - Визначає мінімальну справжню величину швидкості, при якій швидкість буде оцінена як відмінна від нуля, а інтерпольоване положення буде інтерпольовано. Одиниці виміру `min-speed-estimate` такі самі, як і одиниці виміру `velocity`. Коефіцієнт масштабування, в одиницях на одиницю довжини. Занадто низьке значення цього параметра призведе до того, що швидкість буде довго повертатися до 0 після припинення надходження імпульсів від кодера.
- `encoder._<chan>_.phase-A` (біт, In) – Фаза A сигналу квадратурного енкодера.

- encoder._<chan>_.phase-B (біт, In) – Фаза B сигналу квадратурного енкодера.
- encoder._<chan>_.phase-Z (bit, In) - Фаза Z (індексний імпульс) сигналу квадратурного енкодера.
- encoder._<chan>_.position (float, Out) - Позиція в масштабованих одиницях (див. позиція-шкала).
- encoder._<chan>_.position-interpolated (float, Out) - Позиція в масштабованих одиницях, інтерпольована між значеннями енкодера.
  position-interpolated намагається інтерполювати між значеннями енкодера на основі останньої виміряної швидкості. Дійсно тільки тоді, коли швидкість є приблизно постійною і перевищує min-speed-estimate. Не використовуйте для контролю положення, оскільки його значення є некоректним при низьких швидкостях, під час зміни напрямку та під час зміни швидкості. Однак це дозволяє використовувати енкодер з низьким ppr (включаючи енкодер з одним імпульсом на оберт) для нарізування різьби на токарному верстаті, а також може мати й інші застосування.
- encoder._<chan>_.position-latched (float, Out) - На даний момент не використовується.
- encoder._<chan>_.position-scale (float, I/O) - Коефіцієнт масштабування в одиницях на одиницю довжини. Наприклад, якщо position-scale дорівнює 500, то 1000 відліків кодера буде передано як позиція 2,0 одиниці.
- encoder._<chan>_.rawcounts (s32, In) - Необроблений лічильник, визначений лічильниками оновлення. Це значення оновлюється частіше, ніж лічильники та позиція. На нього також не впливає скидання чи індексний імпульс.
- encoder._<chan>_.reset (bit, In) - Якщо значення True, значення «counts» та «position» негайно обнуляються.
- encoder._<chan>_.velocity (float, Out) - Швидкість у масштабованих одиницях за секунду. encoder використовує алгоритм, який значно зменшує квантувальний шум у порівнянні з простим диференціюванням вихідних даних «позиції». Коли величина справжньої швидкості нижча за мінімальну оцінку швидкості, вихідні дані швидкості дорівнюють 0.
- encoder._<chan>_.x4-mode (біт, ввід/вивід) (за замовчуванням: TRUE) - Увімкнення режиму times-4. Якщо значення true, лічильник підраховує кожен фронт квадратурної хвилі (чотири підрахунки за повний цикл). Якщо значення false, підрахунок відбувається лише один раз за повний цикл. У режимі лічильника цей параметр ігнорується. Режим 1x корисний для деяких джог-коліс.

5.8.3.2 Параметри

- 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 Функції

Компонент експортує дві функції. Кожна функція діє на всі лічильники кодера — запуск різних лічильників у різних потоках не підтримується.

- (funct) encoder.update-counters – Високошвидкісна функція для підрахунку імпульсів (без плаваючої коми).
- (funct) encoder.capture-position – Низькошвидкісна функція для оновлення засувок та масштабува положення.

5.8.4 PID

Цей компонент забезпечує пропорційні/інтегральні/похідні контури регулювання. Це виключно компонент реального часу. Для спрощення в цьому описі ми говоримо про контури положення, однак цей компонент можна використовувати для реалізації інших контурів зворотного зв'язку, таких як швидкість, висота пальника, температура тощо. Блок-схема контуру PID — це блок-схема одного контуру PID.

Блок-схема контуру PID-регулятора

:images/pid-block-diag.png

Завантаження PID

```
halcmd: loadrt pid [num_chan=<loops>] [debug=1]
```

<loops> — кількість циклів PID, які ви хочете встановити. Якщо *num_chan* не вказано, буде встановлено один цикл. Максимальна кількість циклів — 16 (як визначено MAX_CHAN у pid.c). Кожен цикл є повністю незалежним. У наведених нижче описах *<loopnum>* — номер конкретного циклу. Перший цикл має номер 0.

Якщо вказано `debug=1`, компонент експортуватиме кілька додаткових контактів, які можуть бути корисними під час налагодження та налаштування. За замовчуванням додаткові контакти не експортуються, щоб заощадити спільний простір пам'яті та уникнути захаращення списку контактів.

Розвантаження PID

```
halcmd: unloadrt pid
```

5.8.4.1 Піни

Три найважливіші шпильки - це

- (float) `pid.`` __<номер циклу>__.command` – Бажана позиція, як її наказано іншим системним компонентом.
- (float) `pid.`` __<loopnum>__.feedback` - Поточне положення, виміряне пристроєм зворотного зв'язку, таким як енкодер.
- (float) `pid.`` __<номер циклу>__.output` – Команда швидкості, яка намагається переміститися з поточної позиції в бажану.

Для позиційної петлі `.command` і `.feedback` вимірюються в одиницях позиції. Для лінійної осі це можуть бути дюйми, міліметри, метри або будь-які інші відповідні одиниці виміру. Аналогічно, для кутової осі це можуть бути градуси, радіани тощо. Одиниці виміру виводу `.output` представляють зміну, необхідну для того, щоб зворотний зв'язок відповідав команді. Таким чином, для контуру положення `.output` є швидкістю в дюймах/с, мм/с, градусах/с тощо. Одиниці часу завжди вимірюються в секундах, а одиниці швидкості відповідають одиницям положення. Якщо команда та зворотний зв'язок вимірюються в метрах, то вихідні дані вимірюються в метрах за секунду.

Кожен контур має два виводи, які використовуються для моніторингу або керування загальною роботою компонента.

- *(float) pid.*<loopnum>*.error* - Дорівнює «.command» мінус «.feedback».
- *(bit) pid.*<loopnum>*.enable* - Біт, який дозволяє цикл. Якщо *.enable* має значення false (хибність), усі інтегратори скидаються, а вихід примусово встановлюється на нуль. Якщо *.enable* має значення true (істина), цикл працює нормально.

Виводи, що використовуються для повідомлення про насичення. Насичення виникає, коли вихідний сигнал блоку PID досягає своєї максимальної або мінімальної межі.

- *(bit) pid.*<loopnum>*.saturated* - Істина, коли вихід насичений.
- *(float) pid.*<номер_циклу>*.saturated_s* – Час насичення виводу.
- *(s32) pid.*<номер_циклу>*.saturated_count* – Час насичення виводу.

Коефіцієнти підсилення, обмеження та інші «настроювані» функції ПІД-регулятора доступні у вигляді виводів, що дозволяє динамічно налаштовувати їх для більш розширених можливостей налаштування.

- *(float) pid.*<номер_циклу>*.Pgain* – Пропорційне посилення
- *(float) pid.*<номер циклу>*.Igain* - Інтегральне посилення
- *(float) pid.*<номер циклу>*.Dgain* – Похідне підсилення
- *(float) pid.*<loopnum>*.bias* - Постійне зміщення на виході
- *(float) pid.*<номер циклу>*.FF0* - Прямий зв’язок нульового порядку - вихід пропорційний команді (положенню).
- *(float) pid.*<loopnum>*.FF1* - Прямий зв’язок першого порядку - вихідний сигнал пропорційний похідній від команди (швидкості).
- *(float) pid.*<loopnum>*.FF2* - Прямий зв’язок другого порядку - вихід пропорційний другій похідній команди (прискорення).
- *(float) pid.*<номер циклу>*.deadband* – Кількість помилок, яка буде проігнорована
- *(float) pid.*<loopnum>*.maxerror* - Обмеження на помилку
- *(float) pid.*<loopnum>*.maxerrorI* - Обмеження на інтегратор помилок
- *(float) pid.*<loopnum>*.maxerrorD* - Обмеження на похідну помилки
- *(float) pid.*<loopnum>*.maxcmdD* - Обмеження на похідну команди
- *(float) pid.*<loopnum>*.maxcmdDD* - Обмеження на похідну команди 2nd
- *(float) pid.*<loopnum>*.maxoutput* - Обмеження вихідного значення

All *max** обмеження реалізовано таким чином, що якщо значення цього параметра дорівнює нулю, обмеження немає.

Якщо під час встановлення компонента було вказано *debug=1*, буде експортовано чотири додаткові виводи:

- *(float) pid.*<loopnum>*.errorI* - Інтеграл похибки.
- *(float) pid.*<loopnum>*.errorD* - Похідна помилки.
- *(float) pid.*<loopnum>*.commandD* - Похідна від команди.
- *(float) pid.*<loopnum>*.commandDD* - 2-га похідна команди.

5.8.4.2 Функції

Компонент експортує одну функцію для кожного циклу PID. Ця функція виконує всі обчислення, необхідні для циклу. Оскільки кожен цикл має свою власну функцію, окремі цикли можуть бути включені в різні потоки і виконуватися з різною швидкістю.

- *(funct) pid.*<номер_циклу>*.do_pid_calcs* – Виконує всі обчислення для одного циклу PID-регулятора.

Якщо ви хочете зрозуміти точний алгоритм, який використовується для обчислення вихідного сигналу PID-контуру, зверніться до

- figure Блок-схема циклу PID-регулятора,
- коментарі на початку *emc2/src/hal/components/pid.c*, і, звичайно ж, до
- сам код.

Обчислення циклу виконуються у функції C *calc_pid()*.

5.8.5 Імітований кодер

Модельований енкодер саме таким і є. Він генерує квадратурні імпульси з індексним імпульсом зі швидкістю, що контролюється виводом HAL. Здебільшого корисний для тестування.

Завантаження SIM-енкодера

```
halcmd: loadrt sim-encoder num_chan=<number>
```

<number> кількість кодерів, які потрібно імітувати. Якщо не вказано, буде встановлено один кодер. Максимальна кількість — 8 (як визначено MAX_CHAN у sim_encoder.c).

Розвантаження SIM-енкодера

```
halcmd: unloadrt sim-encoder
```

5.8.5.1 Піни

- (float) `sim-encoder.` `__<chan-num>__.speed` - Команда швидкості для імітованого вала.
- (bit) `sim-encoder.` `__<chan-num>__.phase-A` - Квадратурний вихід.
- (bit) `sim-encoder.` `__<chan-num>__.phase-B` - Квадратурний вихід.
- (біт) `sim-encoder.` `__<chan-num>__.phase-Z` - Індексний імпульсний вихід.

Коли `.speed` додатне значення, `.phase-A` переважає над `.phase-B`.

5.8.5.2 Параметри

- (u32) `sim-encoder.` `__<chan-num>__.ppr` – Імпульсів на оберт.
- (float) `sim-encoder.` `__<chan-num>__.scale` - Коефіцієнт масштабування для `.speed`. За замовчуванням встановлено значення 1,0, що означає, що `.speed` вимірюється в обертах за секунду. Змініть значення на 60 для обертів за хвилину, на 360 для градусів за секунду, на 6,283185 (= 2*π) для радіан за секунду тощо.

Зверніть увагу, що кількість імпульсів за оберт не те саме, що кількість відліків за оберт. Імпульс – це повний квадратурний цикл. Більшість лічильників енкодерів рахують чотири рази протягом одного повного циклу.

5.8.5.3 Функції

Компонент експортує дві функції. Кожна функція впливає на всі імітовані кодери.

- (funct) `sim-encoder.make-pulses` - Високошвидкісна функція для генерації квадратурних імпульсів (без плаваючої коми).
- (funct) `sim-encoder.update-speed` – Низькошвидкісна функція для зчитування `.speed`, масштабуванн та налаштування `.make-pulses`.

5.8.6 Усунення дребезгу

Функція усунення дребезгу – це компонент реального часу, який може фільтрувати збої, що створюються контактами механічних перемикачів. Він також може бути корисним в інших застосуваннях, де необхідно відхиляти короткі імпульси.

Усунення дребезгу завантаження

```
halcmd: loadrt debounce cfg=<config-string>
```

<config-string>
: Це послідовність десяткових цілих чисел, розділених комами. Кожне число встановлює групу однакових фільтрів усунення дребезгу, це число визначає, скільки фільтрів знаходиться в групі.

Приклад усунення дребезгу завантаження

```
halcmd: loadrt debounce cfg=1,4,2
```

встановить три групи фільтрів. Група 0 містить один фільтр, група 1 містить чотири, а група 2 містить два фільтри. Значенням за замовчуванням для *<config-string>* є «1», що встановить одну групу, яка містить один фільтр. Максимальна кількість груп — 8 (як визначено MAX_GROUPS у debounce.c). Максимальна кількість фільтрів у групі обмежена лише спільним простором пам’яті. Кожна група є повністю незалежною. Усі фільтри в одній групі є ідентичними, і всі вони оновлюються однією і тією ж функцією одночасно. У наведених нижче описах *<G>* є номером групи, а *<F>* є номером фільтра в групі. Перший фільтр — це група 0, фільтр 0.

Розвантаження з усуненням дребезгу

```
halcmd: unloadrt debounce
```

5.8.6.1 Піни

Кожен окремий фільтр має два контакти.

- (bit) `debounce.``__<G>__.__<F>__.in` - Вхід фільтра *<F>* у групі *<G>*.
- (bit) `debounce.``__<G>__.__<F>__.out` - Вихід фільтра *<F>* у групі *<G>*.

5.8.6.2 Параметри

Кожна група фільтрів має один параметр[2].

• (s32) `debounce.` `__<G>__.delay` - Затримка фільтра для всіх фільтрів у групі <*G*>.

Затримка фільтра вимірюється в одиницях періодів потоку. Мінімальна затримка дорівнює нулю. Вихід фільтра з нульовою затримкою точно повторює його вхід - він нічого не фільтрує. Зі збільшенням значення `.delay` відкидаються все довші і довші спотворення. Якщо `.delay` дорівнює 4, відкидаються всі спотворення, менші або рівні чотирьом періодам потоку.

5.8.6.3 Функції

Кожна група фільтрів має одну функцію, яка оновлює всі фільтри в цій групі «одночасно». Різні групи фільтрів можна оновлювати з різних потоків у різні періоди.

• (funct) `debounce.`<*G*> - Оновлює всі фільтри в групі <*G*>.

5.8.7 SigGen

SigGen — це компонент реального часу, який генерує прямокутні, трикутні та синусоїдальні хвилі. Він в основному використовується для тестування.

Завантаження знаку

```
halcmd: loadrt siggen [num_chan=<chans>]
```

<chans>
- це кількість генераторів сигналів, які ви хочете встановити. Якщо *numchan* не вказано, буде встановлено один генератор сигналів. Максимальна кількість генераторів - 16 (як визначено MAX_CHAN у siggen.c). Кожен генератор є повністю незалежним. У наведених нижче описах

<chan>
номер конкретного генератора сигналів (номери починаються з 0).

Розвантаження підпису

```
halcmd: unloadrt siggen
```

5.8.7.1 Піни

Кожен генератор має п’ять вихідних контактів.

• (float) `siggen.` `__<chan>__.sine` - вихід синусоїди.

• (float) `siggen.` `__<chan>__.cosine` - Косинусний вихід.

• (float) `siggen.` `__<chan>__.sawtooth` - Пилкоподібний вихід.

• (float) `siggen.` `__<chan>__.triangle` - Вихід трикутної хвилі.

[2]Кожен окремий фільтр також має внутрішню змінну стану. Існує перемикач часу компіляції, який може експортувати цю змінну як параметр. Це призначено для тестування і в нормальних умовах просто марнує спільну пам’ять.

- (float) `siggen. `__<chan>__.square` - Вихід прямокутної хвилі.

Усі п’ять виходів мають однакову частоту, амплітуду та зсув.

Окрім вихідних контактів, є три керуючі контакти:

- (float) `siggen. `__<chan>__.frequency` - Встановлює частоту в герцах, значення за замовчуванням — 1 Гц.
- (float) `siggen. `__<chan>__.amplitude` - Встановлює пікову амплітуду вихідних сигналів, за замовчува – 1.
- (float) `siggen. `__<chan>__.offset` - Встановлює зміщення постійного струму вихідних сигналів, за замовчуванням — 0.

Наприклад, якщо `siggen.0.amplitude` дорівнює 1,0, а `siggen.0.offset` дорівнює 0,0, вихідні значення будуть коливатися від -1,0 до +1,0. Якщо `siggen.0.amplitude` дорівнює 2,5, а `siggen.0.offse` дорівнює 10,0, вихідні значення будуть коливатися від 7,5 до 12,5.

5.8.7.2 Параметри

Немає. виноска:[До версії 2.1 частота, амплітуда та зсув були параметрами. Їх було змінено на виводи, щоб дозволити керування іншими компонентами.]

5.8.7.3 Функції

- (funct) `siggen. `__<chan>__.update` - Обчислює нові значення для всіх п’яти виходів.

5.8.8 `lut5`

Компонент `lut5` — це логічний компонент з 5 входами, що базується на таблиці пошуку.

- `lut5` не потребує потоку з плаваючою комою.

Завантаження lut5

```
loadrt lut5 [count=N|names=name1[,name2...]]
addf lut5.N servo-thread | base-thread
setp lut5.N.function 0xN
```

lut5 Обчислювальна функція Щоб обчислити шістнадцяткове число для функції, починаючи зверху, поставте 1 або 0, щоб вказати, чи буде цей рядок істинним чи хибним. Далі запишіть кожне число у стовпці виводу, починаючи зверху і записуючи їх справа наліво. Це буде двійкове число. За допомогою калькулятора з програмним інтерфейсом, подібним до того, що є в Ubuntu, введіть двійкове число, а потім перетворіть його в шістнадцяткове, і це буде значенням функції.

Table 5.28: `lut5` Таблиця пошуку

Біт 4	Біт 3	Біт 2	Біт 1	Біт 0	Вихід
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	

Table 5.28: (continued)

Біт 4	Біт 3	Біт 2	Біт 1	Біт 0	Вихід
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 Приклад двох входів У наступній таблиці ми вибрали вихідний стан для кожного рядка, який ми хочемо встановити як «true».

Table 5.29: Приклад таблиці пошуку для двох входів `lut5`

Біт 4	Біт 3	Біт 2	Біт 1	Біт 0	Вихід
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	1

Дивлячись на стовпець виводу нашого прикладу, ми хочемо, щоб вивід був увімкнений, коли біт 0 або біт 0 і біт 1 увімкнені, і ніщо інше. Двійкове число - «b1010» (поверніть вивід на 90 градусів за годинниковою стрілкою). Введіть це число в калькулятор, потім змініть відображення на шістнадцяткове, і число, необхідне для функції, - «0xa». Шістнадцятковий префікс - «0x».

5.9 Генератор компонентів HAL

5.9.1 Вступ

У цьому розділі представлено компоненти компілятора HAL, тобто додаток знань деяких механіків про те, як працювати з машиною. Слід зазначити, що такі компоненти не обов'язково безпосередньо пов'язані з апаратним забезпеченням. Часто це так, але не обов'язково, наприклад, може бути компонент для перетворення між імперською та метричною системами вимірювання, тому цей розділ не вимагає заглиблюватися у взаємодію з апаратним забезпеченням.

Написання компонента HAL може бути нудним процесом, більша частина якого полягає у викликах функцій *rtapi_* та *hal_* та пов'язаній з цим перевірці помилок. *halcompile* автоматично напише весь цей код за вас. Компіляція компонента HAL також набагато простіша при використанні *halcompile*, незалежно від того, чи є компонент частиною дерева джерел LinuxCNC, чи знаходиться поза ним.

Наприклад, простий компонент, такий як "ddt", написаний на C, займає близько 80 рядків коду. Еквівалентний компонент дуже короткий, якщо його написати за допомогою препроцесора *halcompile*:

Приклад простого компонента

```
component ddt "b''0b''b''6b''b''чb''b''иb''b''cb''b''лb''b''ib''b''тb''b''ьb'' b''пb''b' ↩
    'ob''b''xb''b''ib''b''дb''b''нb''b''yb'' b''вb''b''xb''b''ib''b''дb''b''нb''b''ob''b' ↩
    'їb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb''";
pin in float in;
pin out float out;
variable double old;
option period no;
function _;
b''лb''b''ib''b''цb''b''eb''b''нb''b''зb''b''ib''b''яb'' "GPL"; // b''вb''b''кb''b''ab''b' ↩
    'зb''b''yb''b''єb'' b''нb''b''ab'' GPL b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' 2 b' ↩
    'ab''b''6b''b''ob'' b''пb''b''ib''b''зb''b''нb''b''ib''b''шb''b''ob''b''їb'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''ib''b''їb''
;;
float tmp = in;
out = (tmp - old) / fperiod;
old = tmp;
```

5.9.2 Встановлення

Щоб скомпілювати компонент, якщо використовується пакетна версія LinuxCNC, пакети розробки необхідно встановити за допомогою Synaptic з головного меню «Система -> Адміністрування -> Менеджер пакетів Synaptic» або запустивши одну з наступних команд у терміналі:

Встановлення пакетів розробки для LinuxCNC

```
sudo apt install linuxcnc-dev
# b''ab''b''6b''b''ob''
sudo apt install linuxcnc-uspace-dev
```

Інший метод — використовувати менеджер пакетів Synaptic з меню «Програми» для встановлення пакетів `linuxcnc-dev` або `linuxcnc-uspace-dev`.

5.9.3 Компіляція

5.9.3.1 Усередині дерева вихідних кодів

Помістіть файл `.comp` в каталог вихідного коду `linuxcnc/src/hal/components` та перезапустіть `make`. Файли *Comp* автоматично виявляються системою збірки.

Якщо файл `.comp` є драйвером для обладнання, його можна помістити в `linuxcnc/src/hal/drivers` і він буде зібраний, якщо LinuxCNC не налаштовано як симулятор не реального часу.

5.9.3.2 Компоненти реального часу поза деревом вихідного коду

halcompile може обробити, скомпілювати та встановити компонент реального часу за один крок, розмістивши `rtexample.ko` в каталозі модулів реального часу LinuxCNC:

```
[sudo] halcompile --install rtexample.comp
```

Note
sudo (для отримання root-дозволів) потрібен під час використання LinuxCNC з встановленого deb-пакета. Під час використання збірки Run-In-Place (RIP) root-права не потрібні.

Або ж він може обробляти та компілюватися за один крок, залишаючи `example.ko` (або `example.so` для симулятора) у поточному каталозі:

```
halcompile --compile rtexample.comp
```

Або ж він може просто обробити файл, залишивши `example.c` у поточному каталозі:

```
halcompile rtexample.comp
```

halcompile також можна скомпілювати та встановити компонент, написаний на C, використовуючи опції `--install` та `--compile`, показані вище:

```
[sudo] halcompile --install rtexample2.c
```

Документацію в ручному форматі також можна створити з інформації в розділі оголошення:

```
halcompile --document -o example.9 rtexample.comp
```

Отриману сторінку довідки «example.9» можна переглянути за допомогою

```
man ./example.9
```

або скопійовано до стандартного місця для сторінок довідника.

5.9.3.3 Компоненти, що не працюють у реальному часі, поза межами дерева вихідного коду

halcompile може обробляти, компілювати, встановлювати та документувати компоненти, що не працюють у реальному часі:

```
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 деяких бібліотек (наприклад, modbus) може бути необхідно додати додаткові аргументи компілятора та лінкера, щоб компілятор міг знайти та зв'язати бібліотеки. У випадку файлів .comp це можна зробити за допомогою операторів "option" у файлі `.comp`. Для файлів `.c` це неможливо, тому замість цього можна використовувати параметри `--extra-compile-args` та `--extra-link-args`. Наприклад, цю командну рядок можна використовувати для компіляції компонента `vfdb_vfd.c` поза деревом.

```
halcompile --userspace --install --extra-compile-args="-I/usr/include/modbus" --extra-link- ↩
    args="-lm -lmodbus -llinuxcncini" vfdb_vfd.c
```

Note
Ефект використання додаткових аргументів як з командного рядка, так і з файлу не визначено.

5.9.4 Використання компонента

Компоненти потрібно завантажити та додати до потоку, перш ніж їх можна буде використовувати. Надану функціональність можна потім викликати безпосередньо та повторно одним із потоків, або її викликають інші компоненти, які мають власні відповідні тригери.

Приклад HAL-скрипта для встановлення компонента (ddt) та його виконання кожну мілісекунду.

```
loadrt threads name1=servo-thread period1=1000000
loadrt ddt
addf ddt.0 servo-thread
```

Більше інформації про `loadrt` та `addf` можна знайти у Основи HAL.

Щоб протестувати свій компонент, ви можете скористатися прикладами з HAL Tutorial.

5.9.5 Визначення

- **компонент** – Компонент – це окремий модуль реального часу, який завантажується за допомогою `Halcmd loadrt`. Один файл `.comp` визначає один компонент. Назва компонента та назва файлу повинні збігатися.
- **інстанція** - Компонент може мати нуль або більше інстанцій. Кожна інстанція компонента створюється однаковою (всі вони мають однакові контакти, параметри, функції та дані), але поводиться незалежно, коли їхні контакти, параметри та дані мають різні значення.
- **singleton** - Компонент може бути «singleton», і в цьому випадку створюється тільки один екземпляр. Рідко має сенс писати компонент «singleton», якщо тільки в системі не може бути буквально тільки один об'єкт такого типу (наприклад, компонент, призначенням якого є надання контакту поточного часу UNIX, або драйвер апаратного забезпечення для внутрішнього динаміка ПК).

5.9.6 Створення екземпляра

Для синглтона один екземпляр створюється під час завантаження компонента.

Для не-однозаписового об'єкта параметр модуля «count» визначає, скільки пронумерованих екземпляр буде створено. Якщо «count» не вказано, параметр модуля «names» визначає, скільки іменованих екземплярів буде створено. Якщо не вказано ні «count», ні «names», створюється один пронумерований екземпляр.

5.9.7 Неявні параметри

Функціям неявно передається параметр «period», який є часом у наносекундах від останнього періоду виконання компонента. Функції, що використовують плаваючу точку, також можуть посилатися на «fperiod», який є часом у секундах з плаваючою точкою, або (period*1e-9). Це може бути корисно в компонентах, які потребують інформації про час. Див. також «option period» нижче.

5.9.8 Синтаксис

Файл `.comp` складається з кількох оголошень, за якими йде `;;` в окремому рядку, а потім код на C, що реалізує функції модуля.

Декларації включають:

- *component HALNAME (DOC);*
- *pin PIN-НАПРЯМОК ТИПУ HALNAME ([РОЗМІР]|[МАКСИМУМ: РОЗМІР УМОВИ]) (якщо УМОВА) (= ПОЧАТКОВЕЗНАЧЕННЯ) (DOC);*
- *param НАПРЯМОК_ПАРАМЕТРУ ТИП HALNAME ([РОЗМІР]|[МАКСИМУМ: РОЗМІР_УМОВИ]) (якщо УМОВА) (= ПОЧАТКОВЕ_ЗНАЧЕННЯ) (DOC) ;*
- *function HALNAME (fp | nofp) (DOC);*
- *option OPT (VALUE);*
- *variable CTYPE STARREDNAME ([SIZE]);*
- *опис DOC;*
- *приклади DOC;*
- *нотатки DOC;*
- *див. також DOC;*
- *ліцензія LICENSE;*
- *автор AUTHOR;*
- *включають HEADERFILE;*

Дужки позначають додаткові елементи. Вертикальна риски позначають альтернативні варіанти. Слова, написані «ВЕЛИКИМИ ЛІТЕРАМИ», позначають змінний текст, як показано нижче:

- *NAME* – стандартний ідентифікатор C
- *STARREDNAME* - Ідентифікатор C з нулем або більше символів * перед ним. Цей синтаксис можна використовувати для оголошення змінних екземпляру, які є покажчиками. Зверніть увагу, що через граматику між символом * та іменем змінної не може бути пробілів.
- «HALNAME» — розширений ідентифікатор. При використанні для створення ідентифікатора HAL усі підкреслення замінюються на тире, а всі кінцеві тире або крапки видаляються, так що «this_name_» перетворюється на «this-name», а якщо ім’я «_», то кінцева крапка також видаляється, так що «function _» дає ім’я функції HAL, наприклад «component. <num>» замість «component.<num>»

  Якщо присутній, префікс *hal_* видаляється з початку назви компонента під час створення виводів, параметрів та функцій.

В ідентифікаторі HAL для контакту або параметра символ # позначає елемент масиву і повинен використовуватися разом з оголошенням «[SIZE]». Символи сітки замінюються числом, доповненим нулями, довжина якого дорівнює кількості символів #.

Під час використання для створення ідентифікатора C до HALNAME застосовуються такі зміни:

1. Будь-які символи ”#” та будь-які символи ”.”, ”_” або ”-” безпосередньо перед ними вилучається.
2. Будь-які символи ”.” та ”-” замінюються на ”_”.
3. Повторювані символи ”_” замінюються на один символ ”_”.

Заключний символ ”_” зберігається, щоб можна було використовувати ідентифікатори HAL, які в іншому випадку конфліктували б із зарезервованими іменами або ключовими словами (наприклад, *min*).

ІМ’Я ХАЛЬМАНА	**Ідентифікатор C**	**Ідентифікатор HAL**
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* - Вираз, що включає змінну «personality», яка не дорівнює нулю на момент створення виводу або параметра.
- *SIZE* - Число, яке визначає розмір масиву. Елементи масиву нумеруються від 0 до *SIZE*-1.
- *MAXSIZE : CONDSIZE* - Число, яке визначає максимальний розмір масиву, за яким слідує вираз, що містить змінну «personality» і який завжди обчислюється як менше за «MAXSIZE». При створенні масиву його розмір буде дорівнювати «CONDSIZE».
- *DOC* - Рядок, що документує елемент. Рядок може бути рядком у стилі C, взятим у подвійні лапки, наприклад:

```
”b’’Bb’’b’’иb’’b’’бb’’b’’иb’’b’’pb’’b’’ab’’b’’єb’’ b’’пb’’b’’ob’’b’’тb’’b’’pb’’b’’ib’’b’ ↩
    ’бb’’b’’нb’’b’’eb’’ b’’pb’’b’’eb’’b’’бb’’b’’pb’’b’’ob’’: TRUE b’’ob’’b’’зb’’b’’нb’’b’ ↩
    ’ab’’b’’чb’’b’’ab’’b’’єb’’ b’’пb’’b’’ab’’b’’дb’’b’’ib’’b’’нb’’b’’нb’’b’’яb’’, FALSE b’ ↩
    ’ob’’b’’зb’’b’’нb’’b’’ab’’b’’чb’’b’’ab’’b’’єb’’ b’’зb’’b’’pb’’b’’ob’’b’’cb’’b’’тb’’b’ ↩
    ’ab’’b’’нb’’b’’нb’’b’’яb’’”
```

  або рядок у стилі Python, взятий у потрійні лапки, який може містити вбудовані символи нового рядка та лапок, наприклад:

```
”””b’’Дb’’b’’лb’’b’’яb’’ b’’пb’’b’’ob’’b’’яb’’b’’cb’’b’’нb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’  ↩
    b’’вb’’b’’пb’’b’’лb’’b’’иb’’b’’вb’’b’’уb’’ b’’цb’’b’’ьb’’b’’ob’’b’’гb’’b’’ob’’ b’’пb’’ ↩
    b’’ab’’b’’pb’’b’’ab’’b’’мb’’b’’eb’’b’’тb’’b’’pb’’b’’ab’’, b’’тb’’b’’ab’’b’’кb’’b’’ob’’ ↩
    b’’жb’’ b’’вb’’b’’ib’’b’’дb’’b’’ob’’b’’мb’’b’’ob’’b’’гb’’b’’ob’’ b’’яb’’b’’кb’’ ”b’ ↩
    ’cb’’b’’фb’’b’’eb’’b’’pb’’b’’ab’’ b’’зb’’b’’ob’’b’’тb’’b’’ab’’”,
b’’зb’’b’’нb’’b’’ab’’b’’дb’’b’’ob’’b’’бb’’b’’иb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’щb’’b’ ↩
    ’ob’’b’’нb’’b’’ab’’b’’йb’’b’’мb’’b’’eb’’b’’нb’’b’’шb’’b’’eb’’ b’’дb’’b’’вb’’b’’ab’’ b’ ↩
    ’ab’’b’’бb’’b’’зb’’b’’ab’’b’’цb’’b’’иb’’.

b’’Cb’’b’’пb’’b’’ob’’b’’дb’’b’’ib’’b’’вb’’b’’ab’’b’’юb’’b’’cb’’b’’яb’’, b’’цb’’b’’ib’’ b’ ↩
    ’ab’’b’’бb’’b’’зb’’b’’ab’’b’’цb’’b’’иb’’ b’’дb’’b’’ob’’b’’пb’’b’’ob’’b’’мb’’b’’ob’’b’ ↩
    ’гb’’b’’лb’’b’’иb’’ b’’вb’’b’’ab’’b’’мb’’ b’’зb’’b’’pb’’b’’ob’’b’’зb’’b’’уb’’b’’мb’’b’ ↩
    ’ib’’b’’тb’’b’’иb’’ ”zot”
better.”””
```

Або рядку може передувати літерал *r*, і в цьому випадку рядок інтерпретується як необроблений рядок Python.

Строка документації має формат «groff -man». Більш детальну інформацію про цей формат розмітки дивіться у «groff_man(7)». Пам'ятайте, що «halcompile» інтерпретує символи «\» у строках, тому, наприклад, щоб встановити курсив для слова «example», напишіть:

```
"\\fIexample\\fB"
```

У цьому випадку r-рядки особливо корисні, оскільки зворотні скісну риску в r-рядку не потрібно подвоювати:

```
r"\fIexample\fB"
```

- *TYPE* - Один із типів HAL: *bit*, *s32*, *u32*, *s64*, *u64* або *float*. Для *s32* та *u32* також можна використовувати назви *signed* та *unsigned*, але перевага надається *s32* та *u32*.
- *PINDIRECTION* - Одне з наступного: «in», «out» або «io». Компонент встановлює значення для виводу «out», зчитує значення з виводу «in», а також може зчитувати або встановлювати значення виводу «io».
- *PARAMDIRECTION* - Одне з наступного: *r* або *rw*. Компонент встановлює значення для параметра *r*, і він може зчитувати або встановлювати значення параметра *rw*.
- *STARTVALUE* - Вказує початкове значення виводу або параметра. Якщо його не вказано, значення за замовчуванням — «0» або «FALSE», залежно від типу елемента.
- *HEADERFILE* - Ім'я файлу заголовка, вказане в подвійних лапках (`include "myfile.h";`) або в кутових дужках (`include <systemfile.h>;`). Файл заголовка буде включений (за допомогою #include мови C) у верхній частині файлу, перед оголошеннями контактів і параметрів.

5.9.8.1 Функції HAL

- *fp* - Вказує на те, що функція виконує обчислення з плаваючою комою.
- *nofp* - Вказує, що виконуються тільки цілочисельні обчислення. Якщо не вказано ні того, ні іншого, припускається *fp*. Ні *halcompile*, ні gcc не можуть виявити використання обчислень з плаваючою комою у функціях, позначених *nofp*, але використання таких операцій призводить до невизначеної поведінки.

5.9.8.2 Опції

Наразі визначені такі опції:

- *option singleton yes* - (за замовчуванням: no)
  Не створювати параметр модуля *count* і завжди створювати один екземпляр. З *singleton* елементи називаються *component-name.item-name*, а без *singleton* елементи для пронумерованих екземплярів називаються *component-name.<num>.item-name*.
- *option default_count number* - (default: 1)
  Зазвичай, параметр модуля *count* за замовчуванням має значення 1. Якщо вказано, *count* за замовчуванням матиме це значення.
- *option count_function yes* - (default: no)
  Зазвичай кількість екземплярів, що створюються, вказується в параметрі модуля «count»; якщо вказано «count_function», замість цього використовується значення, яке повертає функція «int get_count(void)», а параметр модуля «count» не визначається.

- *option rtapi_app no* - (за замовчуванням: yes)
  Зазвичай функції `rtapi_app_main()` та `rtapi_app_exit()` визначаються автоматично. З *option rtapi_app no* вони не визначаються і повинні бути надані в коді C. Використовуйте наступні прототипи:

```
‘int rtapi_app_main(void);‘

‘void rtapi_app_exit(void);‘
```

  Під час реалізації власної функції `rtapi_app_main()` викличте функцію `int export(char *prefix, long extra_arg)`, щоб зареєструвати виводи, параметри та функції для `prefix`.

- *option data TYPE* - (за замовчуванням: немає) **застаріле**
  Якщо вказано, кожен екземпляр компонента матиме пов’язаний блок даних типу *TYPE* (який може бути простим типом, таким як *float*, або назвою типу, створеного за допомогою *typedef*). У нових компонентах слід використовувати *variable*.

- *option extra_setup yes* - (за замовчуванням: ні)
  Якщо вказано, викликати функцію, визначену параметром *EXTRA_SETUP*, для кожного екземпляра. Якщо використовується автоматично визначений параметр *rtapi_app_main*, *extra_arg* – це номер цього екземпляра.

- *option extra_cleanup yes* - (за замовчуванням: ні)
  Якщо вказано, викликати функцію, визначену параметром *EXTRA_CLEANUP*, з автоматично визначеного *rtapi_app_exit* або, у разі виявлення помилки, з автоматично визначеного *rtapi_app_mai*

- *option userspace yes* - (за замовчуванням: no)
  Якщо вказано, цей файл описує компонент, що не працює в режимі реального часу (раніше відомий як «userspace»), а не звичайний (тобто в режимі реального часу) компонент. Компонент, що не працює в режимі реального часу, може не мати функцій, визначених директивою *function*. Натомість, після створення всіх екземплярів, викликається функція C `void user_mainloop(voi`
  Коли ця функція повертається, компонент завершується. Зазвичай, *user_mainloop()* використовує *FOR_ALL_INSTS()* для виконання дії оновлення для кожного екземпляра, а потім короткочасно переходить у режим очікування. Іншою поширеною дією в *user_mainloop()* може бути виклик циклу обробки подій інструментарію графічного інтерфейсу користувача.

- *option userinit yes* - (за замовчуванням: no)
  Ця опція ігнорується, якщо опція *userspace* (див. вище) встановлена на *no*. Якщо вказано *userinit*, функція *userinit(argc,argv)* викликається перед *rtapi_app_main()* (а отже, перед викликом *hal_init()*). Ця функція може обробляти аргументи командного рядка або виконувати інші дії. Її тип повернення — «void»; вона може викликати «exit()», якщо бажає завершити роботу, а не створювати компонент HAL (наприклад, через недійсність аргументів командного рядка).

- *option extra_link_args ”...”* - (за замовчуванням: ””)
  Ця опція ігнорується, якщо опція «userspace» (див. вище) встановлена на «no». При зв’язуванні компонента, що не працює в режимі реального часу, вказані аргументи вставляються в рядок зв’язку. Зверніть увагу, що оскільки компіляція відбувається в тимчасовому каталозі, «-L.» посилається на тимчасовий каталог, а не на каталог, де знаходиться вихідний файл .comp. Цей параметр можна встановити в командному рядку halcompile за допомогою -extra-link-args=”-L.....”. Ця альтернатива дозволяє встановити додаткові прапорці в тих випадках, коли вхідний файл є файлом .c, а не файлом .comp.

- *option extra_compile_args ”...”* - (за замовчуванням: ””)
  Ця опція ігнорується, якщо опція «userspace» (див. вище) встановлена на «no». Під час компіляції компонента, що не працює в режимі реального часу, вказані аргументи вставляються в командний рядок компілятора. Якщо вхідний файл є файлом .c, цю опцію можна встановити в командному рядку halcompile за допомогою --extra-compile-args=”-I.....”. Ця альтернатива дозволяє встановити додаткові прапори у випадках, коли вхідний файл є файлом .c, а не файлом .comp.

- *option homemod yes* - (за замовчуванням: ні)
  Модуль — це користувацький модуль самонаведення, що завантажується за допомогою `[EMCMOT]HOME`
  .
- *option tpmod yes* - (за замовчуванням: ні)
  Модуль – це користувацький модуль планування траєкторії (tp), що завантажується за допомогою `[TRAJ]TPMOD=`*modulename* .
- «option period no» — (за замовчуванням: yes)
  Керує неявним параметром «period» функції (функцій), визначеної (визначених) у компоненті. Стандартна функція має неявний параметр «period». Багато компонентів не використовують параметр «period», що спричиняє попередження компілятора про «невикористаний параметр». Встановлення «option period no» створює оголошення функції, в якому опускається параметр «period», що запобігає попередженню. Встановлення цієї опції також запобігає визначенню «fperiod», оскільки воно залежить від «period».

Якщо значення опції не вказано, це еквівалентно вказівці «опція … так».
Результат присвоєння опції невідповідного значення є невизначеним.
Результат використання будь-якої іншої опції є невизначеним.

5.9.8.3 Ліцензія та авторство

- `LICENSE` - Вкажіть ліцензію модуля для документації та для оголошення модуля MODULE_LICENSE().
  Наприклад, щоб вказати, що ліцензія модуля — GPL версії 2 або пізнішої версії:

```
‘b’’лв’’b’’ib’’b’’цb’’b’’eb’’b’’нb’’b’’зb’’b’’ib’’b’’яb’’ ”GPL”; // b’’вb’’b’’кb’’b’’ab’’ ↩
    b’’зb’’b’’yb’’b’’єb’’ b’’нb’’b’’ab’’ GPL b’’вb’’b’’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’ 2  ↩
    b’’ab’’b’’бb’’b’’ob’’ b’’пb’’b’’ib’’b’’зb’’b’’нb’’b’’ib’’b’’шb’’b’’ob’’b’’їb’’ b’’вb’’ ↩
    b’’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’‘
```

  Для отримання додаткової інформації про значення MODULE_LICENSE() та додаткові ідентифікатор
  ліцензій див. *<linux/module.h>* або сторінку довідки *rtapi_module_param(3)*.

  Ця декларація **обов’язкова**.

- `AUTHOR` - Вкажіть автора модуля для документації.

5.9.8.4 Зберігання даних для кожного екземпляра

- `variable CTYPE STARREDNAME; + variable CTYPE STARREDNAME[SIZE]; + variable CTYPE STARRED`
  `= DEFAULT; + variable CTYPE STARREDNAME[SIZE] = DEFAULT;`

  Оголосіть змінну *STARREDNAME* типу *CTYPE* для кожного екземпляра, опціонально як масив елементів *SIZE* і опціонально з типовим значенням *DEFAULT*. Елементи без *DEFAULT* ініціалізуються з усіма бітами, що дорівнюють нулю. «CTYPE» — це простий однослівний тип C, такий як «float», «u32», «s32», «int» тощо. Для доступу до змінних масиву використовуються квадратні дужки.

Якщо змінна має бути типу вказівника, між символом ”*” та іменем змінної не може бути пробілу.
Тому прийнятним є наступне:

```
variable int *example;
```

Але наступні не є такими:

```
variable int* badexample;
variable int * badexample;
```

5.9.8.5 Коментарі

Однорядкові коментарі в стилі C++ (`//...`) та багаторядкові коментарі в стилі C (`/* ... */`) підтримуються в розділі оголошення.

5.9.9 Обмеження

Хоча HAL дозволяє виводу, параметру та функції мати однакову назву, «halcompile» цього не дозволяє.

Назви змінних та функцій, які не можна використовувати або які можуть спричинити проблеми, включають:

- Будь-що, що починається з _comp.
- *comp_id*
- *fperiod*
- *rtapi_app_main*
- *rtapi_app_exit*
- *extra_setup*
- *extra_cleanup*

5.9.10 Зручні макроси

На основі елементів у розділі декларації «halcompile» створює структуру C під назвою «struct *comp_state». Однак замість посилання на члени цієї структури (наприклад, «*(inst->name)»), на них зазвичай посилаються за допомогою макросів, наведених нижче. Деталі «struct* comp_state» та цих макросів можуть змінюватися від однієї версії «halcompile» до іншої.

- `FUNCTION( `__name__)`` - Використовуйте цей макрос, щоб розпочати визначення функції реального часу, яка була раніше оголошена за допомогою *function NAME*. Функція включає параметр *period*, який є цілим числом наносекунд між викликами функції. Див. також *option period* вище.
- `EXTRA_SETUP()` – Використовуйте цей макрос для початку визначення функції, яка викликається для виконання додаткового налаштування цього екземпляра. Повертає від'ємне значення UNIX *errno* для позначення помилки (наприклад, 'return -EBUSY.
- `EXTRA_CLEANUP()` - Використовуйте цей макрос, щоб розпочати визначення функції, яка викликається для виконання додаткового очищення компонента. Зверніть увагу, що ця функція повинна очистити всі екземпляри компонента, а не тільки один. Макроси «pin_name», «parameter_name» та «data» тут не можуть використовуватися.
- «pin_name» або «parameter_name» — для кожного виводу «pin_name» або параметра «parameter_name» існує макрос, який дозволяє використовувати ім'я самостійно для посилання на вивід або параметр. Коли «pin_name» або «parameter_name» є масивом, макрос має вигляд «pin_name(idx)» або «param_name(idx)», де «idx» є індексом у масиві контактів. Коли масив є масивом змінного розміру, дозволено посилатися лише на елементи до його «condsize».

  Коли елемент є умовним елементом, посилання на нього допустиме лише тоді, коли його «умова» оцінюється як ненульове значення.
- *variable_name* - Для кожної змінної *variable_name* існує макрос, який дозволяє використовувати ім'я самостійно для посилання на змінну. Коли *variable_name* є масивом, використовується звичайний індекс у стилі C: *variable_name[idx]*.

- *data* – Якщо вказано ”option data”, цей макрос дозволяє доступ до даних екземпляра.
- *fperiod* – кількість секунд у форматі з плаваючою комою між викликами цієї функції реального часу. Див. також *option period* вище.
- `FOR_ALL_INSTS() {...}` - Для компонентів, що не працюють у режимі реального часу. Цей макрос повторює всі визначені екземпляри. Усередині тіла циклу макроси «pin_name», «parameter_name» та «data» працюють так само, як і в функціях реального часу.

5.9.11 Компоненти з однією функцією

Якщо компонент має тільки одну функцію і рядок «FUNCTION» не з’являється ніде після «;;», то частина після «;;» вважається тілом єдиної функції компонента. Дивіться Simple Comp для прикладу цього.

5.9.12 Компонент особистості

Якщо компонент має будь-які виводи або параметри з «умовою if» або «[maxsize : condsize]», він називається компонентом з «особистістю». «Особистість» кожного екземпляра визначається під час завантаження модуля. «Особистість» може використовуватися для створення виводів тільки за потреби. Наприклад, особистість використовується в компоненті «логіка», щоб забезпечити змінну кількість вхідних контактів для кожного логічного вентиля і дозволити вибір будь-якої з основних булевих логічних функцій «and», «or» і «xor».

Кількість дозволених елементів «персоналізації» за замовчуванням встановлюється під час компіляції (64). Значення за замовчуванням застосовується до численних компонентів, що входять до дистрибутиву та зібрані за допомогою halcompile.

Щоб змінити дозволену кількість елементів особистості для компонентів, створених користувачем, використовуйте опцію *--personalities* з halcompile. Наприклад, щоб дозволити до 128 разів особистості:

```
[sudo] halcompile --personalities=128 --install ...
```

Під час використання компонентів з особистостями зазвичай вказують елемент особистості для **кожного** зазначеного екземпляра компонента. Приклад для 3 екземплярів логічного компонента:

```
loadrt logic names=and4,or3,nand5, personality=0x104,0x203,0x805
```

Note
Якщо рядок loadrt визначає більше екземплярів, ніж особистостей, екземплярам з невизначеними особистостями присвоюється особистість 0. Якщо запитувана кількість екземплярів перевищує кількість дозволених особистостей, особистості присвоюються шляхом індексації за модулем кількості дозволених особистостей. Виводиться повідомлення, що вказує на такі присвоєння.

5.9.13 Приклади

5.9.13.1 постійний

Зверніть увагу, що оголошення ”function _” створює функції з іменем ”constant.0” тощо. Ім’я файлу має збігатися з іменем компонента.

```
component constant;
pin out float out;
param r float value = 1.0;
option period no;
function _;
b''лb''b''ib''b''цb''b''eb''b''нb''b''зb''b''ib''b''яb'' "GPL"; // b''вb''b''кb''b''ab''b' ↩
    'зb''b''yb''b''єb'' b''нb''b''ab'' GPL b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' 2 b' ↩
    'ab''b''бb''b''ob'' b''пb''b''ib''b''зb''b''нb''b''ib''b''шb''b''ob''b''їb'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''ib''b''їb''
;;
FUNCTION(_) { out = value; }
```

5.9.13.2 синхронізація

Цей компонент обчислює синус і косинус вхідного кута в радіанах. Він має інші можливості, ніж виходи «синус» і «косинус» siggen, оскільки вхідним параметром є кут, а не «частота», що змінюється вільно.

Виводи оголошені у вихідному коді з іменами *sin_* та *cos_*, щоб вони не заважали функціям *sin()* та *cos()*. Виводи HAL все ще називаються *sincos.<num>.sin*.

```
component sincos;
pin out float sin_;
pin out float cos_;
pin in float theta;
option period no;
function _;
b''лb''b''ib''b''цb''b''eb''b''нb''b''зb''b''ib''b''яb'' "GPL"; // b''вb''b''кb''b''ab''b' ↩
    'зb''b''yb''b''єb'' b''нb''b''ab'' GPL b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' 2 b' ↩
    'ab''b''бb''b''ob'' b''пb''b''ib''b''зb''b''нb''b''ib''b''шb''b''ob''b''їb'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''ib''b''їb''
;;
#b''вb''b''кb''b''лb''b''юb''b''чb''b''ab''b''юb''b''тb''b''ьb'' <rtapi_math.h>
FUNCTION(_) { sin_ = sin(theta); cos_ = cos(theta); }
```

5.9.13.3 out8

Цей компонент є драйвером для «фіктивної» карти під назвою «out8», яка має 8 виводів цифрового виходу, що обробляються як одне 8-бітне значення. У системі може бути різна кількість таких карт, і вони можуть знаходитися за різними адресами. Контакт називається «out_», оскільки «out» є ідентифікатором, що використовується в «<asm/io.h>». Він ілюструє використання «EXTRA_SETUP» та «EXTRA_CLEANUP» для запиту області вводу-виводу, а потім її звільнення у разі помилки або коли модуль вивантажується.

```
component out8;
pin out unsigned out_ "b''Bb''b''иb''b''хb''b''ib''b''дb''b''нb''b''eb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb''; b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b' ↩
    'cb''b''тb''b''ob''b''вb''b''yb''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''лb''b''иb''b' ↩
    'шb''b''eb'' b''мb''b''ob''b''лb''b''ob''b''дb''b''шb''b''ib'' 8 b''бb''b''ib''b''тb''b' ↩
    'ib''b''вb''";
param r unsigned ioaddr;

function _;

option period no;
option count_function;
option extra_setup;
```

```
option extra_cleanup;
option constructable no;

b''лb''b''ib''b''цb''b''eb''b''нb''b''зb''b''ib''b''яb'' "GPL"; // b''вb''b''кb''b''ab''b' ↩
    'зb''b''уb''b''єb'' b''нb''b''ab'' GPL b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' 2 b' ↩
    'ab''b''бb''b''ob'' b''пb''b''ib''b''зb''b''нb''b''ib''b''шb''b''ob''b''їb'' b''вb''b' ↩
    'eb''b''pb''b''cb''b''ib''b''їb''
;;
#include <asm/io.h>

#define MAX 8
int io[MAX] = {0,};
RTAPI_MP_ARRAY_INT(io, MAX, "b''Ab''b''дb''b''pb''b''eb''b''cb''b''иb'' b''вb''b''вb''b' ↩
    'ob''b''дb''b''уb''/b''вb''b''иb''b''вb''b''ob''b''дb''b''уb'' b''пb''b''лb''b''ab''b' ↩
    'тb'' out8");

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")) {
        // b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''цb''b' ↩
            'eb''b''йb'' b''пb''b''ob''b''pb''b''тb'' b''вb''b''вb''b''ob''b''дb''b''уb''/b' ↩
            'вb''b''иb''b''вb''b''ob''b''дb''b''уb'' b''нb''b''ab'' 0, b''щb''b''ob''b''бb'' ↩
             EXTRA_CLEANUP b''нb''b''eb'' b''зb''b''вb''b''ib''b''лb''b''ьb''b''нb''b''яb''b ↩
            ''вb'' b''вb''b''вb''b''ob''b''дb''-b''вb''b''иb''b''вb''b''ob''b''дb''
        // b''пb''b''ob''b''pb''b''тb''b''иb'', b''яb''b''кb''b''ib'' b''нb''b''ib''b''кb'' ↩
            b''ob''b''лb''b''иb'' b''нb''b''eb'' b''зb''b''ab''b''пb''b''иb''b''тb''b''уb''b ↩
            ''вb''b''ab''b''лb''b''иb''b''cb''b''яb''.
        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;
```

Цей фрагмент компонента ілюструє використання префікса *hal_* в назві компонента.

`loop` є загальною назвою, а префікс `hal_` дозволяє уникнути потенційних конфліктів імен з іншим, не пов'язаним програмним забезпеченням. Наприклад, у системах реального часу RTAI код реального часу виконується в ядрі, тому якщо компонент мав би назву просто `loop`, він міг би легко конфліктувати зі стандартним модулем ядра `loop`.

Після завантаження, *halcmd show comp* покаже компонент під назвою *hal_loop*. Однак, пін, який показує *halcmd show pin*, буде *loop.0.example*, а не *hal-loop.0.example*.

5.9.13.5 демо-масиву

Цей компонент реального часу ілюструє використання масивів фіксованого розміру:

```
component arraydemo ”4-bit Shift register”;
pin in bit in;
pin out bit out-# [4];
option period no;
function _ nofp;
b’’лb’’b’’ib’’b’’цb’’b’’eb’’b’’нb’’b’’зb’’b’’ib’’b’’яb’’ ”GPL”; // b’’вb’’b’’кb’’b’’ab’’b’ ↩
    ’зb’’b’’уb’’b’’єb’’ b’’нb’’b’’ab’’ GPL b’’вb’’b’’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’ 2 b’ ↩
    ’ab’’b’’бb’’b’’ob’’ b’’пb’’b’’ib’’b’’зb’’b’’нb’’b’’ib’’b’’шb’’b’’ob’’b’’їb’’ b’’вb’’b’ ↩
    ’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’
;;
int i;
for(i=3; i>0; i--) out(i) = out(i-1);
out(0) = in;
```

5.9.13.6 ранд

Цей компонент, що не працює в реальному часі, змінює значення на своєму вихідному контакті на нове випадкове значення в діапазоні (0,1) приблизно раз на 1 мс.

```
component rand;
option userspace;

pin out float out;
b’’лb’’b’’ib’’b’’цb’’b’’eb’’b’’нb’’b’’зb’’b’’ib’’b’’яb’’ ”GPL”; // b’’вb’’b’’кb’’b’’ab’’b’ ↩
    ’зb’’b’’уb’’b’’єb’’ b’’нb’’b’’ab’’ GPL b’’вb’’b’’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’ 2 b’ ↩
    ’ab’’b’’бb’’b’’ob’’ b’’пb’’b’’ib’’b’’зb’’b’’нb’’b’’ib’’b’’шb’’b’’ob’’b’’їb’’ b’’вb’’b’ ↩
    ’eb’’b’’pb’’b’’cb’’b’’ib’’b’’їb’’
;;
#include <unistd.h>

void user_mainloop(void) {
    while(1) {
        usleep(1000);
        FOR_ALL_INSTS() out = drand48();
    }
}
```

5.9.13.7 логіка (використовуючи особистість)

Цей компонент реального часу показує, як використовувати ”персоналізацію” для створення масивів змінного розміру та додаткових виводів.

```
b’’лb’’b’’ob’’b’’гb’’b’’ib’’b’’кb’’b’’ab’’ b’’кb’’b’’ob’’b’’мb’’b’’пb’’b’’ob’’b’’нb’’b’ ↩
    ’eb’’b’’нb’’b’’тb’’b’’ab’’ ”b’’Kb’’b’’ob’’b’’мb’’b’’пb’’b’’ob’’b’’нb’’b’’eb’’b’’нb’’b’ ↩
    ’тb’’ LinuxCNC HAL, b’’щb’’b’’ob’’ b’’зb’’b’’ab’’b’’бb’’b’’eb’’b’’зb’’b’’пb’’b’’eb’’b’ ↩
    ’чb’’b’’уb’’b’’єb’’ b’’eb’’b’’кb’’b’’cb’’b’’пb’’b’’eb’’b’’pb’’b’’иb’’b’’мb’’b’’eb’’b’ ↩
    ’нb’’b’’тb’’b’’ab’’b’’лb’’b’’ьb’’b’’нb’’b’’ib’’ b’’лb’’b’’ob’’b’’гb’’b’’ib’’b’’чb’’b’ ↩
    ’нb’’b’’ib’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’ib’’b’’їb’’”;
b’’вb’’b’’иb’’b’’вb’’b’’ib’’b’’дb’’ b’’вb’’ b’’бb’’b’’ib’’b’’тb’’ in-##[16 : personality &  ↩
    0xff];
```

```
b''вb''b''иb''b''вb''b''ib''b''дb'' b''зb'' b''бb''b''ib''b''тb''b''уb'' and b''яb''b''кb'' ↩
    b''щb''b''ob'' personality & 0x100;
b''вb''b''иb''b''вb''b''ib''b''дb'' b''зb'' b''бb''b''ib''b''тb''b''уb'' or b''яb''b''кb''b ↩
    ''щb''b''ob'' personality & 0x200;
b''вb''b''иb''b''вb''b''ib''b''дb'' b''зb'' b''бb''b''ib''b''тb''b''уb'' xor b''яb''b''кb'' ↩
    b''щb''b''ob'' personality & 0x400;
b''ob''b''пb''b''цb''b''ib''b''яb'' period no;
b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''яb'' _ nofp;
b''ob''b''пb''b''иb''b''cb'' """
b''Eb''b''кb''b''cb''b''пb''b''eb''b''рb''b''иb''b''мb''b''eb''b''нb''b''тb''b''аb''b''лb'' ↩
    b''ьb''b''нb''b''иb''b''йb'' b''зb''b''аb''b''гb''b''аb''b''лb''b''ьb''b''нb''b''иb''b' ↩
    'йb'' b''кb''b''оb''b''мb''b''пb''b''оb''b''нb''b''eb''b''нb''b''тb'' «b''лb''b''оb''b' ↩
    'гb''b''ib''b''чb''b''нb''b''оb''b''їb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b' ↩
    'їb''». b''Mb''b''оb''b''жb''b''eb'' b''вb''b''иb''b''кb''b''оb''b''нb''b''уb''b''вb''b ↩
    ''аb''b''тb''b''иb'' «and», «or»
b''тb''b''аb'' «xor» b''дb''b''оb'' 16 b''вb''b''хb''b''оb''b''дb''b''ib''b''вb''. b''Bb'' ↩
    b''иb''b''зb''b''нb''b''аb''b''чb''b''тb''b''eb'' b''пb''b''рb''b''аb''b''вb''b''иb''b' ↩
    'лb''b''ьb''b''нb''b''eb'' b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'дb''b''лb''b''яb'' «personality»,
 b''дb''b''оb''b''дb''b''аb''b''вb''b''шb''b''иb'':
.IP \\(bu 4
b''Kb''b''ib''b''лb''b''ьb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''вb''b''хb''b''ib''b' ↩
    'дb''b''нb''b''иb''b''хb'' b''кb''b''оb''b''нb''b''тb''b''аb''b''кb''b''тb''b''ib''b' ↩
    'вb'', b''зb''b''аb''b''зb''b''вb''b''иb''b''чb''b''аb''b''йb'' b''вb''b''ib''b''дb'' 2 ↩
    b''дb''b''оb'' 16
.IP \\(bu
256 (0x100) b''яb''b''кb''b''щb''b''оb'' b''бb''b''аb''b''жb''b''аb''b''нb''b''иb''b''йb'' ↩
     b''вb''b''иb''b''хb''b''ib''b''дb'' «and»
.IP \\(bu
512 (0x200) b''яb''b''кb''b''щb''b''оb'' b''бb''b''аb''b''жb''b''аb''b''нb''b''иb''b''йb'' ↩
     b''вb''b''иb''b''хb''b''ib''b''дb'' «or»
.IP \\(bu
1024 (0x400) b''яb''b''кb''b''щb''b''оb'' b''бb''b''аb''b''жb''b''аb''b''нb''b''иb''b' ↩
    'йb'' b''вb''b''иb''b''хb''b''ib''b''дb'' «xor» (b''вb''b''иb''b''кb''b''лb''b''юb''b' ↩
    'чb''b''нb''b''eb'' b''аb''b''бb''b''оb'')""";
b''лb''b''ib''b''цb''b''eb''b''нb''b''зb''b''ib''b''яb'' "GPL"; // b''вb''b''кb''b''аb''b' ↩
    'зb''b''уb''b''єb'' b''нb''b''аb'' GPL v2 b''аb''b''бb''b''оb'' b''пb''b''ib''b''зb''b' ↩
    'нb''b''ib''b''шb''b''уb'' b''вb''b''eb''b''рb''b''cb''b''ib''b''юb''
;;
b''Фb''b''Уb''b''Нb''b''Кb''b''Цb''b''Ib''b''Яb''(_) {
    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;
}
```

Типова лінія навантаження для цього компонента може бути

```
loadrt logic count=3 personality=0x102,0x305,0x503
```

що створює такі піни:

- 2-входовий елемент І: `логіка 0.і`, `логіка 0.в-00`, `логіка 0.в-01`
- 5-входові логічні елементи І та АБО: `logic.1.and`, `logic.1.or`, `logic.1.in-00`, `logic.1.in-01`, `logic.1.in-02`, `logic.1.in-03`, `logic.1.in-04`,
- 3-входові елементи І та Виключаюче АБО: `logic.2.and`, `logic.2.xor`, `logic.2.in-00`, `logic.2.in-01`, `logic.2.in-02`

5.9.13.8 Загальні функції

У цьому прикладі показано, як викликати функції з функції main. Також показано, як передавати посилання на виводи HAL цим функціям.

```
b’’пb’’b’’pb’’b’’иb’’b’’кb’’b’’лb’’b’’аb’’b’’дb’’ b’’кb’’b’’оb’’b’’мb’’b’’пb’’b’’оb’’b’ ↩
    ’нb’’b’’еb’’b’’нb’’b’’тb’’b’’аb’’;
pin in s32 in;
pin out bit out1;
pin out bit out2;

b’’нb’’b’’оb’’b’’мb’’b’’еb’’b’’pb’’ b’’пb’’b’’еb’’b’’pb’’b’’іb’’b’’оb’’b’’дb’’b’’уb’’ b’ ↩
    ’оb’’b’’пb’’b’’цb’’b’’іb’’b’’їb’’;
b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’’яb’’ _;
b’’лb’’b’’іb’’b’’цb’’b’’еb’’b’’нb’’b’’зb’’b’’іb’’b’’яb’’ ”GPL”;
;;

// b’’зb’’b’’аb’’b’’гb’’b’’аb’’b’’лb’’b’’ьb’’b’’нb’’b’’иb’’b’’йb’’ b’’нb’’b’’аb’’b’’бb’’b’ ↩
    ’іb’’b’’pb’’ b’’вb’’b’’иb’’b’’вb’’b’’оb’’b’’дb’’b’’іb’’b’’вb’’, b’’фb’’b’’уb’’b’’нb’’b’ ↩
    ’кb’’b’’цb’’b’’іb’’b’’яb’’ true
void set(hal_bit_t *p){
    *p = 1;
}

// b’’зb’’b’’аb’’b’’гb’’b’’аb’’b’’лb’’b’’ьb’’b’’нb’’b’’аb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’ ↩
    ’цb’’b’’іb’’b’’яb’’ b’’вb’’b’’сb’’b’’тb’’b’’аb’’b’’нb’’b’’оb’’b’’вb’’b’’лb’’b’’еb’’b’ ↩
    ’нb’’b’’нb’’b’’яb’’ b’’вb’’b’’иb’’b’’вb’’b’’оb’’b’’дb’’b’’іb’’b’’вb’’ false
void unset(hal_bit_t *p){
    *p = 0;
}

//b’’оb’’b’’сb’’b’’нb’’b’’оb’’b’’вb’’b’’нb’’b’’аb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’ ↩
    ’іb’’b’’яb’’
FUNCTION(_) {
    if (in < 0){
        set(&out1);
        unset(&out2);
    }else if (in >0){
        unset(&out2);
        set(&out2);
    }else{
        unset(&out1);
        unset(&out2);
    }
}
```

Цей компонент використовує дві загальні функції для маніпулювання бітовим виводом HAL, на який посилається.

5.9.14 Використання командного рядка

Сторінка довідки halcompile містить детальну інформацію про виклик `halcompile`.

```
$ man halcompile
```

Короткий опис використання halcompile наведено таким чином:

```
$ halcompile --help
```

5.10 Файли HALCL

halcmd чудово підходить для визначення компонентів і з’єднань, але ці скрипти не мають обчислювальних можливостей. Як результат, INI-файли мають обмежену чіткість і стислість, які можливі при використанні мов вищого рівня.

Функція haltcl надає можливість використовувати скрипти Tcl та їхні функції для обчислень, циклів, розгалужень, процедур тощо в файлах INI. Щоб скористатися цією функцією, використовуйте мову Tcl та розширення .tcl для файлів HAL.

Розширення .tcl розуміється основним скриптом (`linuxcnc`), який обробляє INI-файли. Файли Haltcl ідентифікуються в розділі HAL INI-файлів (як і HAL-файли).

Приклад

```
[HAL]
HALFILE = conventional_file.hal
HALFILE = tcl_based_file.tcl
```

За належної обережності файли HAL та Tcl можна змішувати.

5.10.1 Сумісність

Мова halcmd, що використовується у файлах HAL, має простий синтаксис, який насправді є підмножиною потужнішої універсальної мови сценаріїв Tcl.

5.10.2 Команди Haltcl

Файли Haltcl використовують мову сценаріїв Tcl, доповнену специфічними командами рівня абстракції обладнання (HAL) LinuxCNC. Специфічні команди HAL:

```
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
```

Два особливі випадки виникають для команд «gets» та «list» через конфлікти з вбудованими командами Tcl. Для haltcl цим командам має передувати ключове слово «hal»:

```
halcmd   haltcl
------   ------
gets     hal gets
list     hal list
```

5.10.3 Змінні INI-файлу Haltcl

Змінні INI-файлу доступні як за допомогою `halcmd`, так і за допомогою `haltcl`, але з різним синтаксисом. INI-файли LinuxCNC використовують специфікатори SECTION та ITEM для ідентифікації елементів конфігурації:

```
[SECTION_A]
ITEM1 = value_1
ITEM2 = value_2
...
[SECTION_B]
...
```

Доступ до значень INI-файлу можна отримати шляхом підстановки тексту у HAL-файлах у такій формі:

```
[SECTION]ITEM
```

Ті самі значення INI-файлу доступні у Tcl-файлах за допомогою форми глобальної змінної масиву Tcl:

```
$::SECTION(ITEM)
```

Наприклад, елемент INI-файлу, такий як:

```
[JOINT_0]
MAX_VELOCITY = 4
```

виражається як `[JOINT_0]MAX_VELOCITY` у файлах HAL для halcmd
та як. `$::JOINT_0(MAX_VELOCITY)` у файлах Tcl для haltcl.

Оскільки INI-файли можуть повторювати один і той самий елемент ITEM в одному розділі SECTION кілька разів, `$::SECTION(ITEM)` насправді є списком Tcl для кожного окремого значення.

Коли є тільки одне значення і воно є простим (усі значення, що складаються тільки з літер і цифр без пробілів, належать до цієї групи), то можна розглядати `$::SECTION(ITEM)` так, ніби це не список.

Якщо значення може містити спеціальні символи (лапки, фігурні дужки, вбудовані пробіли та інші символи, що мають спеціальне значення в Tcl), то необхідно розрізняти список значень і початкове (і, можливо, єдине) значення в списку.

У Tcl це написано `[lindex $::SECTION(ITEM) 0]`.

Наприклад: враховуючи наступні значення INI

```
[HOSTMOT2]
DRIVER=hm2_eth
IPADDR="10.10.10.10"
BOARD=7i92
CONFIG="num_encoders=0 num_pwmgens=0 num_stepgens=6"
```

А ця команда loadrt:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=$::HOSTMOT2(IPADDR) config=$::HOSTMOT2(CONFIG)
```

Ось фактична команда, яка виконується:

```
loadrt hm2_eth board_ip={"10.10.10.10"} config={"num_encoders=0 num_pwmgens=0 num_stepgens ↩
    =6"}
```

Це не вдається, оскільки loadrt не розпізнає дужки.

Отже, щоб отримати значення саме так, як вони були введені у файлі INI, перепишіть рядок loadrt ось так:

```
loadrt $::HOSTMOT2(DRIVER) board_ip=[lindex $::HOSTMOT2(IPADDR) 0] config=[lindex  ↩
    $::HOSTMOT2(CONFIG) 0]
```

5.10.4 Конвертування HAL-файлів у Tcl-файли

Існуючі файли HAL можна конвертувати у файли Tcl шляхом ручного редагування, щоб адаптувати їх до вищезазначених відмінностей. Процес можна автоматизувати за допомогою скриптів, які конвертують з використанням цих підстановок.

```
[SECTION]ITEM ---> $::SECTION(ITEM)
gets          ---> hal gets
list          ---> hal list
```

5.10.5 Примітки Haltcl

У haltcl аргумент-значення для команд *sets* та *setp* неявно обробляється як вираз у мові Tcl.

Приклад

```
# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''кb''b''ob''b' ↩
    'eb''b''фb''b''ib''b''цb''b''ib''b''єb''b''нb''b''тb'' b''пb''b''ib''b''дb''b''cb''b' ↩
    'иb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''тb''b''вb''b''ob''b''pb''b''eb''b''нb''b''нb''b''яb'' b''гb''b''pb''b''ab''b' ↩
    'дb''/b''cb'' b''вb'' b''ob''b''дb''b''иb''b''нb''b''иb''b''цb''b''ib''/b''xb''b''вb'' b ↩
    ''дb''b''лb''b''яb'' b''pb''b''ab''b''дb''b''ib''b''yb''b''cb''b''ab'' JOINT_0
setp scale.0.gain 6.28/360.0*$::JOINT_0(radius)*60.0
```

Пробіли у виразі не допускаються, для цього використовуйте лапки:

```
setp scale.0.gain "6.28 / 360.0 * $::JOINT_0(radius) * 60.0"
```

В інших контекстах, таких як «loadrt», для обчислювальних виразів необхідно явно використовувати команду Tcl expr, ([expr {}]).

Приклад

```
loadrt motion base_period=[expr {500000000/$::TRAJ(MAX_PULSE_RATE)}]
```

5.10.6 Приклади Haltcl

Розглянемо тему «запас потужності stepgen». Програмне забезпечення `stepgen` найкраще працює з обмеженням прискорення, яке є «трохи вищим», ніж те, що використовується планувальником руху. Тому при використанні файлів `halcmd` ми змушуємо файли INI мати значення, обчислене вручну.

```
[JOINT_0]
MAXACCEL = 10.0
STEPGEN_MAXACCEL = 10.5
```

За допомогою `haltcl` ви можете використовувати команди Tcl для виконання обчислень та повного виключення елемента INI-файлу `STEPGEN_MAXACCEL`:

```
setp stepgen.0.maxaccel $::JOINT_0(MAXACCEL)*1.05
```

Ще однією функцією `haltcl` є циклічність і тестування. Наприклад, багато конфігурацій симуляторів використовують файли HAL «core_sim.hal» або «core_sim9.hal». Вони відрізняються між собою через необхідність підключення більшої або меншої кількості осей. Наступний код haltcl працюватиме для будь-якої комбінації осей у машині trivkins.

```
# b''Cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''cb''b''иb''b''гb''b''нb''b' ↩
    'ab''b''лb''b''иb'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'шb''b''вb''b''иb''b''дb''b''кb''b''ob''b''cb''b''тb''b''ib'' b''тb''b''ab'' b''пb''b' ↩
    'pb''b''иb''b''cb''b''кb''b''ob''b''pb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''лb''b' ↩
    'яb'' b''кb''b''ob''b''жb''b''нb''b''ob''b''їb'' b''ob''b''cb''b''ib''
set ddt 0
for {set jnum 0} {$jnum < $::KINS(JOINTS)} {incr jnum} {
  # «list pin» b''пb''b''ob''b''вb''b''eb''b''pb''b''тb''b''ab''b''єb'' b''пb''b''ob''b' ↩
      'pb''b''ob''b''жb''b''нb''b''ib''b''йb'' b''cb''b''пb''b''иb''b''cb''b''ob''b''кb'', b ↩
      ''яb''b''кb''b''щb''b''ob'' pin b''нb''b''eb'' b''ib''b''cb''b''нb''b''yb''b''єb''
  if {[hal list pin joint.${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 Інтерактивний

Команда `halrun` розпізнає файли haltcl. З параметром -T haltcl можна запускати інтерактивно як інтерпретатор Tcl. Ця можливість корисна для тестування та для автономних HAL-застосунків.

Приклад

```
$ halrun -T haltclfile.tcl
```

5.10.8 Приклади розподілу Haltcl (simulator)

Каталог configs/sim/axis/simtcl містить INI-файл, який використовує файл .tcl для демонстрації конфігурації haltcl у поєднанні з використанням двопрохідної обробки. Приклад показує використання процедур Tcl, циклів, коментарів та виведення на термінал.

5.11 Інтерфейс користувача HAL

5.11.1 Вступ

Halui — це інтерфейс користувача на основі HAL для LinuxCNC, який з'єднує контакти HAL із командами NML. Більшість функцій (кнопки, індикатори тощо), що надаються традиційним графічним інтерфейсом користувача (AXIS, GMOCCAPY, QtDragon тощо), у Halui забезпечуються контактами HAL.

Найпростіший спосіб додати halui – це додати наступний код до розділу [HAL] INI-файлу:

```
[HAL]
HALUI = halui
```

Альтернативний спосіб його виклику (особливо якщо ви генеруєте конфігурацію за допомогою StepConf) – це включити наступне у ваш файл `custom.hal`.
Переконайтеся, що ви використовуєте правильний шлях до вашого INI-файлу.

```
loadusr halui -ini /path/to/inifile.ini
```

5.11.2 MDI

Іноді користувач хоче додати складніші завдання, які будуть виконуватися активацією виводу HAL. Це можливо шляхом додавання команд MDI до INI-файлу в розділі [HALUI]. Приклад:

```
[HALUI]
MDI_COMMAND = G0 X0
MDI_COMMAND = G0 G53 Z0
MDI_COMMAND = G28
MDI_COMMAND = o<mysub>call
...
```

Коли halui запускається, він зчитує поля `MDI_COMMAND` в INI і експортує контакти від 00 до кількості `MDI_COMMAND`, знайдених в INI, максимум до 64 команд. Ці контакти можна підключати як будь-які контакти HAL. Поширеним методом є використання кнопок, що надаються віртуальними панелями керування, як показано в прикладі Приклад підключень MDI_COMMAND.

Example 5.1 Приклад для з'єднань 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
```

Мережі, що з'єднують виводи halui.mdi-command-NN, надані halui.

```
$ halcmd show pin halui.mdi
b''Bb''b''ив''b''вb''b''ob''b''дb''b''ив'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''ib''b''вb'':
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
    ...
```

Коли контакт halui MDI встановлений (імпульсний) у стан «істина», halui надсилає команду MDI, визначену в INI. Це не завжди буде успішним, залежно від поточного режиму роботи (наприклад, у режимі AUTO halui не може успішно надсилати команди MDI).

5.11.3 Приклад конфігурації

Приклад конфігурації sim (`configs/sim/axis/halui_pyvcp/halui.ini`) включено до дистрибутиву.

5.11.4 Довідка про піни Halui

Усі піни halui також задокументовані на сторінці довідки halui:

```
$ man halui
```

Or see http://linuxcnc.org/docs/devel/html/man/man1/halui.1.html

5.11.4.1 Перервати

- *halui.abort* (bit, in) - PIN-код для надсилання повідомлення про переривання (виправляє більшість помилок)

5.11.4.2 E-Стій

- *halui.estop.activate* (bit, in) - PIN-код для запиту аварійної зупинки
- *halui.estop.is-activated* (bit, out) - вказує на скидання аварійної зупинки
- *halui.estop.reset* (bit, in) - PIN-код для запиту скидання аварійної зупинки

5.11.4.3 Перевизначення каналу

- *halui.feed-override.count-enable* (bit, in) - має бути істинним, щоб «підрахунки» або «пряме значення» працювали.
- *halui.feed-override.counts* (s32, in) - кількість * шкала = відсоток FO. Можна використовувати з енкодером або «прямим значенням».
- *halui.feed-override.decrease* (bit, in) - штифт для зменшення FO (-=масштаб)
- *halui.feed-override.increase* (bit, in) - штифт для збільшення FO (+=масштаб)
- *halui.feed-override.reset* (bit, in) - контакт для скидання оптоволоконного виводу (масштаб=1.0)
- *halui.feed-override.direct-value* (bit, in) - false під час використання енкодера для зміни лічильників, true під час безпосереднього встановлення лічильників.
- *halui.feed-override.scale* (float, in) - штифт для налаштування шкали збільшення та зменшення «корекції подачі».
- *halui.feed-override.value* (float, out) - поточне значення FO

5.11.4.4 Туман

- *halui.mist.is-on* (bit, out) - вказує на туман
- *halui.mist.off* (bit, in) - шпилька для запиту на вимкнення туману
- *halui.mist.on* (bit, in) - шпилька для запиту на туман

5.11.4.5 Повінь

- *halui.flood.is-on* (bit, out) - вказує на початок повені
- *halui.flood.off* (bit, in) - PIN-код для запиту на відключення флуду
- *halui.flood.on* (bit, in) - PIN-код для запиту на флуд

5.11.4.6 Самонаведення

- *halui.home-all* (bit, in) - PIN-код для запиту переведення всіх осей у початкове положення. Цей PIN-код буде присутній лише тоді, коли у файлі INI встановлено HOME_SEQUENCE.

5.11.4.7 Машина

- *halui.machine.units-per-mm* (float out) - штифт для машинних одиниць вимірювання на мм (дюйм:1/25, мм:1) відповідно до налаштування inifile: [TRAJ]LINEAR_UNITS
- *halui.machine.is-on* (bit, out) - вказує на увімкнення машини
- *halui.machine.off* (bit, in) - PIN-код для запиту на вимкнення машини
- *halui.machine.on* (bit, in) - PIN-код для запиту на ввімкнення машини

5.11.4.8 Максимальна швидкість

Максимальну лінійну швидкість можна налаштувати від 0 до MAX_VELOCITY, встановленого в розділі [TRAJ] INI-файлу.

- *halui.max-velocity.count-enable* (bit, in) - має бути істинним, щоб «підрахунки» або «пряме значення» працювали.
- *halui.max-velocity.counts* (s32, in) - counts * шкала = відсоток MV. Може використовуватися з енкодером або «прямим значенням».
- *halui.max-velocity.direct-value* (bit, in) - false під час використання енкодера для зміни лічильників, true під час безпосереднього встановлення лічильників.
- *halui.max-velocity.decrease* (bit, in) - штифт для зменшення максимальної швидкості
- *halui.max-velocity.increase* (bit, in) - штифт для збільшення максимальної швидкості
- *halui.max-velocity.scale* (float, in) - величина, що застосовується до поточної максимальної швидкості з кожним переходом від вимкненого до ввімкненого положення штифта збільшення або зменшення в машинних одиницях за секунду.
- *halui.max-velocity.value* (float, out) - — максимальна лінійна швидкість у машинних одиницях за секунду.

5.11.4.9 MDI

- *halui.mdi-command-*<nn> (біт, вхід) - halui спробує надіслати команду MDI, визначену в INI. *<nn>* - це двозначне число, що починається з 00.
  Якщо команда виконана успішно, LinuxCNC перейде в режим MDI, а потім повернеться в ручний режим.
  Якщо в файлі ini не встановлено змінних [HALUI]MDI_COMMAND, halui не експортуватиме контакти halui.mdi-command-<nn>.
- *halui.halui-mdi-is-running* (bit, out) - Стан виконання команд MDI, надісланих halui. Стан активний навіть під час перемикання режимів. Якщо в ini-файлі не встановлено змінні [HALUI]MDI_COMMAND halui не експортуватиме ці виводи.

5.11.4.10 Суглоб

N = joint number (0 … num_joints-1)
Приклад:

- *halui.joint.*N.*select* (bit in) - штифт для вибору з’єднання *N*
- *halui.joint.*N.*is-selected* (bit out) - PIN-код стану, на якому вибрано суглоб *N*

- *halui.joint.*N.*has-fault* (bit out) - контактний індикатор стану, що вказує на несправність з’єднання *N*
- *halui.joint.*N.*home* (bit in) - штифт для опорного шарніра *N*
- *halui.joint.*N.*is-homed* (bit out) - PIN-код стану, що вказує на те, що шарнір *N* знаходиться вдома
- *halui.joint.*N.*on-hard-max-limit* (bit out) - контакт стану, що вказує, що з’єднання *N* знаходиться на позитивній апаратній межі
- *halui.joint.*N.*on-hard-min-limit* (bit out) - контакт стану, що вказує, що з’єднання *N* знаходиться на негативній апаратній межі
- *halui.joint.*N.*on-soft-max-limit* (bit out) - контакт стану, що вказує, що з’єднання *N* знаходиться на додатній програмній межі
- *halui.joint.*N.*on-soft-min-limit* (bit out) - контакт стану, що вказує, що шарнір *N* знаходиться на негативній програмній межі
- *halui.joint.*N.*override-limits* (bit out) - PIN-код стану, що вказує на тимчасове перевизначення обмежень суглоба *N*
- *halui.joint.*N.*unhome* (bit in) - штифт для з’єднання з від’єднанням *N*
- *halui.joint.selected* (u32 out) - вибраний номер суглоба (0 … кількість_суглобів-1)
- *halui.joint.selected.has-fault* (bit out) - Вибраний з’єднання має пошкодження
- *halui.joint.selected.home* (bit in) - штифт для наведення вибраного з’єднання в початкове положення
- *halui.joint.selected.is-homed* (bit out) - PIN-код стану, що вказує на те, що вибране з’єднання перебуває в головному положенні
- *halui.joint.selected.on-hard-max-limit* (bit out) - контактний індикатор стану, який вказує, що вибране з’єднання знаходиться на позитивній апаратній межі
- *halui.joint.selected.on-hard-min-limit* (bit out) - контактний індикатор стану, який вказує, що вибраний з’єднання знаходиться на негативній межі обладнання
- *halui.joint.selected.on-soft-max-limit* (bit out) - контактний індикатор стану, який вказує, що вибраний шарнір знаходиться на додатній програмній межі
- *halui.joint.selected.on-soft-min-limit* (bit out) - контактний індикатор стану, який вказує, що вибраний шарнір знаходиться на негативній програмній межі
- *halui.joint.selected.override-limits* (bit out) - PIN-код стану, що вказує на тимчасове перевизначення обмежень вибраного суглоба
- *halui.joint.selected.unhome* (bit in) - штифт для зняття з початкового положення вибраного з’єднання

5.11.4.11 Суглобовий біг

N = номер суглоба (0 … кількість_суглобів-1)

- *halui.joint.jog-deadband* (float in) - контакт для налаштування зони нечутливості аналогового режиму поштовху (аналогові входи поштовху, менші/повільніші за цю зону - за абсолютним значенням - ігноруються)
- *halui.joint.jog-speed* (float in) - штифт для налаштування швидкості штовхання для плюс/мінус штовхання.

- *halui.joint.*N.*analog* (float in) - штифт для штовхання шарніра *N* за допомогою значення з плаваючою комою (наприклад, джойстика). Значення, яке зазвичай встановлюється між 0,0 та ±1,0, використовується як множник швидкості штовхання.
- *halui.joint.*N.*increment* (float in) - штифт для встановлення кроку поперечного переміщення для з'єднання *N* при використанні кроку плюс/мінус
- *halui.joint.*N.*increment-minus* (bit in) - Наростаючий фронт змусить з'єднання *N* зміщуватися в негативному напрямку на величину приросту
- *halui.joint.*N.*increment-plus* (bit in) - Наростаючий фронт змусить з'єднання *N* рухатися в позитивному напрямку на величину приросту
- *halui.joint.*N.*minus* (bit in) - штифт для штовхаючого з'єднання *N* у негативному напрямку зі швидкістю halui.joint.jog-speed
- *halui.joint.*N.*plus* (bit in) - штифт для штовхаючого з'єднання *N* у позитивному напрямку зі швидкістю halui.joint.jog-speed
- *halui.joint.selected.increment* (float in) - штифт для встановлення кроку поперечного переміщення для вибраного з'єднання при використанні приросту плюс/мінус
- *halui.joint.selected.increment-minus* (bit in) - Наростаючий фронт призведе до зміщення вибраного з'єднання у негативному напрямку на величину приросту
- *halui.joint.selected.increment-plus* (bit in) - Наростаючий фронт призведе до того, що вибраний з'єднання зміститься в позитивному напрямку на величину приросту
- *halui.joint.selected.minus* (bit in) - штифт для штурхового переміщення вибраного з'єднання у негативному напрямку зі швидкістю halui.joint.jog-speed
- *halui.joint.selected.plus* (bit in) - штифт для штовхання вибраного з'єднання в позитивному напрямку зі швидкістю halui.joint.jog-speed

5.11.4.12 Вісь

L = axis letter (xyzabcuvw)

- *halui.axis.*L.*select* (bit) - штифт для вибору осі за літерою
- *halui.axis.*L.*is-selected* (bit out) - контакт стану, на якому вибрано вісь *L*
- *halui.axis.*L.*pos-commanded* (float out) - Задане положення осі в координатах машини
- *halui.axis.*L.*pos-feedback* float out) - Положення осі зворотного зв'язку в координатах машини
- *halui.axis.*L.*pos-relative* (float out) - Положення осі зворотного зв'язку у відносних координатах

5.11.4.13 Біг по осі

L = axis letter (xyzabcuvw)

- *halui.axis.jog-deadband* (float in) - контакт для налаштування зони нечутливості аналогового режиму поштовху (аналогові входи поштовху, менші/повільніші за цю зону (за абсолютним значенням), ігноруються)
- *halui.axis.jog-speed* (float in) - штифт для налаштування швидкості штовхання для плюс/мінус штовхання.

- *halui.axis.*L.*analog* (float in) - штифт для штовхання осі *L* за допомогою значення з плаваючою комою (наприклад, джойстика). Значення, яке зазвичай встановлюється між 0,0 та ±1,0, використовується як множник швидкості штовхання.
- *halui.axis.*L.*increment* (float in) - штифт для встановлення приросту поштовху для осі *L* при використанні приросту плюс/мінус
- *halui.axis.*L.*increment-minus* (bit in) - Наростаючий фронт змусить вісь *L* рухатися в негативному напрямку на величину приросту
- *halui.axis.*L.*increment-plus* (bit in) - Наростаючий фронт змусить вісь *L* рухатися в позитивному напрямку на величину приросту
- *halui.axis.*L.*minus* (bit in) - штифт для осі штовхання *L* у негативному напрямку зі швидкістю halui.axis.jog-speed
- *halui.axis.*L.*plus* (bit in) - штифт для осі штовхання *L* у позитивному напрямку зі швидкістю halui.axis.jog-speed
- *halui.axis.selected* (u32 out) - вибрана вісь (за індексом: 0:x 1:y 2:z 3:a 4:b 5:cr 6:u 7:v 8:w)
- *halui.axis.selected.increment* (float in) - штифт для встановлення приросту подачі для вибраної осі при використанні приросту плюс/мінус
- *halui.axis.selected.increment-minus* (bit in) - Наростаючий фронт змусить вибрану вісь рухатися в негативному напрямку на величину приросту
- *halui.axis.selected.increment-plus* (bit in) - Наростаючий фронт змусить вибрану вісь рухатися в позитивному напрямку на величину приросту
- *halui.axis.selected.minus* (bit in) - штифт для штовхання вибраної осі в негативному напрямку зі швидкістю halui.axis.jog-speed
- *halui.axis.selected.plus* (pin in) - для штовхання вибраного осьового біта в позитивному напрямку зі швидкістю halui.axis.jog-speed

5.11.4.14 Режим

- *halui.mode.auto* (bit, in) - PIN-код для запиту автоматичного режиму
- *halui.mode.is-auto* (bit, out) - вказує на те, що автоматичний режим увімкнено
- *halui.mode.is-joint* (bit, out) - вказує на те, що ввімкнено режим поштовхового переміщення з’єднання за з’єднанням
- *halui.mode.is-manual* (bit, out) - вказує на те, що ручний режим увімкнено
- *halui.mode.is-mdi* (bit, out) - вказує на те, що режим MDI увімкнено
- *halui.mode.is-teleop* (bit, out) - вказує на те, що увімкнено режим скоординованого поштовху
- *halui.mode.joint* (bit, in) - контакт для запиту режиму поштовху з’єднання за з’єднанням
- *halui.mode.manual* (bit, in) - PIN-код для виклику ручного режиму
- *halui.mode.mdi* (bit, in) - PIN-код для запиту режиму MDI
- *halui.mode.teleop* (bit, in) - контакт для запиту скоординованого режиму поштовху

5.11.4.15 Програма

- *halui.program.block-delete.is-on* (bit, out) - PIN-код стану, що вказує на те, що видалення блоку ввімкнено
- *halui.program.block-delete.off* (bit, in) - PIN-код для запиту на видалення блоку вимкнено
- *halui.program.block-delete.on* (bit, in) - PIN-код для запиту на видалення блоку ввімкнено
- *halui.program.is-idle* (bit, out) - PIN-код стану, що вказує на те, що жодна програма не запущена
- *halui.program.is-paused* (bit, out) - PIN-код стану, що вказує на те, що програма призупинена
- *halui.program.is-running* (bit, out) - PIN-код стану, що вказує на те, що програма запущена
- *halui.program.optional-stop.is-on* (bit, out) - PIN-код стану, що вказує на увімкнення додаткової зупинки
- *halui.program.optional-stop.off* (bit, in) - шпилька із запитом на вимкнення додаткової зупинки
- *halui.program.optional-stop.on* (bit, in) - шпилька із запитом на включення додаткової зупинки
- *halui.program.pause* (bit, in) - PIN-код для призупинення програми
- *halui.program.resume* (bit, in) - PIN-код для відновлення призупиненої програми
- *halui.program.run* (bit, in) - PIN-код для запуску програми
- *halui.program.step* (bit, in) - PIN-код для поетапного входу в програму
- *halui.program.stop* (bit, in) - PIN-код для зупинки програми

5.11.4.16 Швидке перевизначення

- *halui.rapid-override.count-enable* (bit in (за замовчуванням: TRUE)) - Якщо значення TRUE, змінювати швидке перевизначення при зміні підрахунку.
- *halui.rapid-override.counts* (s32 in) - counts X scale = Відсоток швидкого перевизначення. Може використовуватися з енкодером або «прямим значенням».
- *halui.rapid-override.decrease* (bit in) - штифт для зменшення швидкого керування (-=масштаб)
- *halui.rapid-override.direct-value* (bit in) - контакт для активації входу швидкого керування прямим значенням
- *halui.rapid-override.increase* (bit in) - штифт для збільшення швидкого керування (+=масштаб)
- *halui.rapid-override.scale* (float in) - штифт для налаштування шкали при зміні швидкого керування
- *halui.rapid-override.value* (float out) - поточне значення швидкого перемикання
- *halui.rapid-override.reset* (bit, in) - контакт для скидання значення швидкого керування (масштаб=1.0

5.11.4.17 Корекції шпинделя

- *halui.spindle*.N.*override.count-enable* (bit, in) - має бути істинним, щоб «підрахунки» або «пряме значення» працювали.
- *halui.spindle*.N.*override.counts* (s32, in) - counts * scale = SO відсоток. Може використовуватися з енкодером або «прямим значенням».
- *halui.spindle*.N.*override.decrease* (bit, in) - штифт для зменшення SO (-=масштаб)
- *halui.spindle*.N.*override.direct-value* (bit, in) - false під час використання енкодера для зміни лічильників, true під час безпосереднього встановлення лічильників.
- *halui.spindle*.N.*override.increase* (bit, in) - штифт для збільшення SO (+=масштаб)
- *halui.spindle*.N.*override.scale* (float, in) - штифт для налаштування шкали при зміні SO
- *halui.spindle*.N.*override.value* (float, out) - поточне значення SO
- *halui.spindle*.N.*override.reset* (bit, in) - контакт для скидання значення SO (масштаб=1.0)

5.11.4.18 Шпиндель

- *halui.spindle*.N.*brake-is-on* (bit, out) - вказує на те, що гальмо увімкнено
- *halui.spindle*.N.*brake-off* (bit, in) - штифт для деактивації шпинделя/гальма
- *halui.spindle*.N.*brake-on* (bit, in) - штифт для активації гальма шпинделя
- *halui.spindle*.N.*decrease* (bit, in) - зменшує швидкість шпинделя
- *halui.spindle*.N.*forward* (bit, in) - запускає шпиндель рухом за годинниковою стрілкою
- *halui.spindle*.N.*increase* (bit, in)- збільшує швидкість шпинделя
- *halui.spindle*.N.*is-on* (bit, out) - вказує на те, що шпиндель увімкнено (в будь-якому напрямку)
- *halui.spindle*.N.*reverse* (bit, in)- запускає шпиндель рухом проти годинникової стрілки
- *halui.spindle*.N.*runs-backward* (bit, out) - вказує на те, що шпиндель увімкнено, і обертається у зворотному напрямку
- *halui.spindle*.N.*runs-forward* (bit, out) - вказує на те, що шпиндель увімкнено та обертається вперед
- *halui.spindle*.N.*start* (bit, in) - запускає шпиндель
- *halui.spindle*.N.*stop* (bit, in) - зупиняє шпиндель

5.11.4.19 Інструмент

- *halui.tool.length-offset.a* (float out) - поточне застосоване зміщення довжини інструмента для осі A
- *halui.tool.length-offset.b* (float out) - поточне застосоване зміщення довжини інструмента для осі B
- *halui.tool.length-offset.c* (float out) - поточне застосоване зміщення довжини інструмента для осі C
- *halui.tool.length-offset.u* (float out) - поточне застосоване зміщення довжини інструменту для осі U

- *halui.tool.length-offset.v* (float out) - поточне застосоване зміщення довжини інструмента для осі V
- *halui.tool.length-offset.w* (float out) - поточне застосоване зміщення довжини інструмента для осі W
- *halui.tool.length-offset.x* (float out) - поточне застосоване зміщення довжини інструмента для осі X
- *halui.tool.length-offset.y* (float out) - поточне застосоване зміщення довжини інструмента для осі Y
- *halui.tool.length-offset.z* (float out) - поточне застосоване зміщення довжини інструмента для осі Z
- *halui.tool.diameter* (float out) - Поточний діаметр інструмента або 0, якщо інструмент не завантажено.
- *halui.tool.number* (u32, out) - вказує на поточний вибраний інструмент

5.12 Приклади залів

Щоб будь-які приклади Halui працювали, вам потрібно додати наступний рядок до розділу [HAL] INI-файлу.

```
HALUI = halui
```

5.12.1 Дистанційний запуск

Щоб підключити кнопку віддаленого запуску програми до LinuxCNC, використовуйте контакти `halui.program.run` та `halui.mode.auto`. Спочатку необхідно переконатися, що запуск можливий, використовуючи контакт `halui.mode.is-auto`. Це можна зробити за допомогою компонента `and2`. На наступному малюнку показано, як це робиться. Коли натискається кнопка віддаленого запуску, вона підключається як до `halui.mode.auto`, так і до `and2.0.in0`. Якщо автоматичний режим є прийнятним, контакт `halui.mode.is-auto` буде увімкнений. Якщо обидва входи компонента `and2.0` увімкнені, `and2.0.out` буде увімкнений, і це запустить програму.

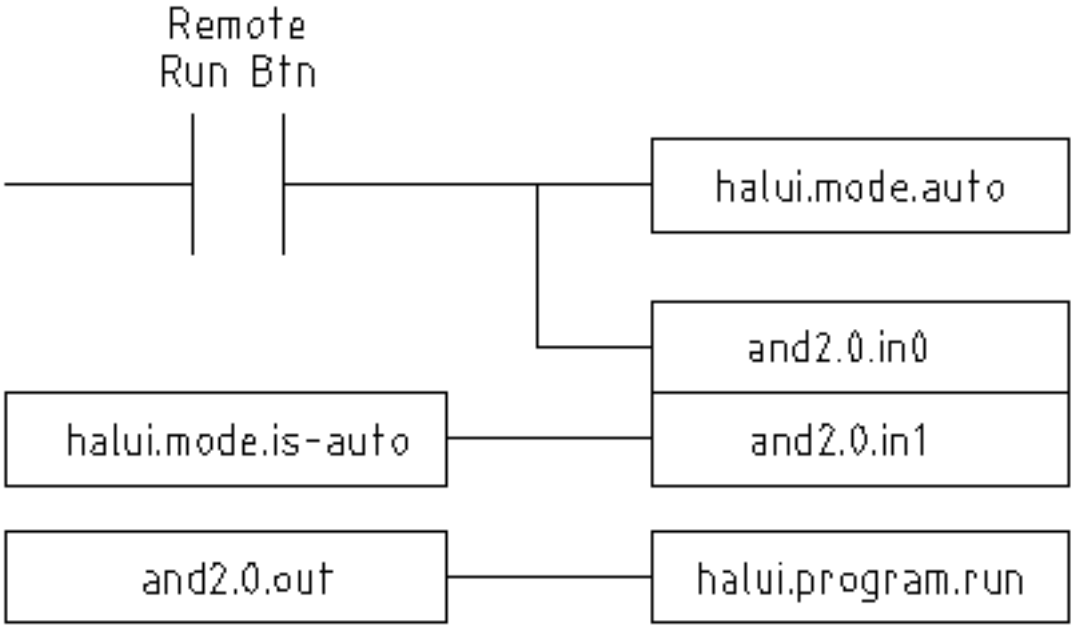


Figure 5.20: Приклад дистанційного запуску

Команди hal, необхідні для виконання вищезазначеного:

```
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
```

Зверніть увагу, що в першому рядку є два контакти зчитувача, це також можна розділити на два рядки ось так:

```
net program-start-btn halui.mode.auto <= <your input pin>
net program-start-btn and2.0.in0
```

5.12.2 Пауза та відновлення

Цей приклад було розроблено, щоб дозволити LinuxCNC переміщувати обертову вісь за сигналом від зовнішнього верстата. Координацію між двома системами забезпечуватимуть два компоненти Halui:

- `halui.program.is-paused`
- `halui.program.resume`

У вашому налаштованому HAL-файлі додайте наступні два рядки, які будуть підключені до вашого вводу/виводу, щоб увімкнути паузу програми або відновити її, коли зовнішня система захоче продовжити роботу LinuxCNC.

```
net ispaused halui.program.is paused => ”your output pin”
net resume halui.program.resume <= ”your input pin”
```

Ваші вхідні та вихідні контакти підключені до контактів, підключених до іншого контролера. Це можуть бути контакти паралельного порту або будь-які інші контакти вводу/виводу, до яких у вас є доступ.

Ця система працює наступним чином. Коли в G-коді зустрічається M0, сигнал `halui.program.is-pause` стає істинним. Це вмикає вихідний контакт, щоб зовнішній контролер знав, що LinuxCNC призупинено

Щоб відновити виконання G-коду програми LinuxCNC, зовнішній контролер, коли буде готовий, зробить свій вивід true. Це сигналізуватиме LinuxCNC про те, що слід відновити виконання G-коду.

Труднощі з визначенням часу

- Вхідний сигнал повернення ”відновлення” не повинен бути довшим за час, необхідний для повторного запуску G-коду.
- Вихід ”призупинено” не повинен бути активним до моменту закінчення сигналу ”відновити”.

Цих проблем з синхронізацією можна уникнути, використовуючи ClassicLadder для активації виходу «is-paused» за допомогою моностабільного таймера, щоб подати один вузький вихідний імпульс. Імпульс «resume» також можна отримати за допомогою моностабільного таймера.

5.13 Створення компонентів Python, що не працюють у реальном часі

У цьому розділі пояснюються принципи реалізації компонентів HAL за допомогою мови програмування Python.

5.13.1 Базовий приклад використання

Компонент, що не працює в режимі реального часу, спочатку створює свої контакти та параметри, а потім входить у цикл, який періодично передає всі вихідні дані з вхідних. Наступний компонент копіює значення, яке бачить на своєму вхідному контакті («passthrough.in»), на свій вихідний контакт («passthrough.out») приблизно раз на 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
```

Скопіюйте наведений вище список у файл з назвою ”passthrough”, зробіть його виконуваним (*chmod +x*) та помістіть його у ваш *$PATH*. Потім спробуйте:

Копія екрана з детальною інформацією про виконання щойно створеного модуля HAL для передачі даних.

```
$ halrun

halcmd: loadusr passthrough

halcmd: show pin
```

```
    b''Kb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''нb''b''ib'' b''кb''b' ↩
        'ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'':
    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

    b''Kb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''нb''b''ib'' b''кb''b' ↩
        'ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'':
    Owner Type  Dir     Value  Name
     03   float IN       3.14  passthrough.in
     03   float OUT      3.14  passthrough.out
```

5.13.2 Компоненти та затримки, що не працюють у реальному часі

Якщо ви швидко набрали «show pin», ви можете побачити, що «passthrough.out» все ще має старе значення 0. Це відбувається через виклик «time.sleep(1)», який змушує присвоювання виходу відбуватися не частіше ніж раз на секунду. Оскільки це компонент, що не працює в режимі реального часу, фактична затримка між присвоєннями може бути набагато довшою, якщо пам'ять, яка використовується компонентом passthrough, переноситься на диск, оскільки присвоєння може бути відкладено до тих пір, поки ця пам'ять не буде перенесена назад.

Таким чином, компоненти, що не працюють у режимі реального часу, підходять для інтерактивних елементів, таких як панелі керування (затримки в діапазоні мілісекунд не помітні, а більш тривалі затримки є прийнятними), але не підходять для надсилання імпульсів кроку до плати крокового драйвера (затримки завжди повинні бути в діапазоні мікросекунд, незалежно від обставин).

5.13.3 Створення контактів та параметрів

```
h = hal.component("passthrough")
```

Сам компонент створюється за допомогою виклику конструктора «hal.component». Аргументами є ім'я компонента HAL і (опціонально) префікс, що використовується для імен виводів і параметрів. Якщо префікс не вказано, використовується ім'я компонента.

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

Потім штифти створюються за допомогою викликів методів об'єкта компонента. Аргументами є: суфікс імені штифта, тип штифта та напрямок штифта. Для параметрів аргументами є: суфікс імені параметра, тип параметра та напрямок параметра.

Table 5.31: Назви опцій HAL

Типи контактів та параметрів:	HAL_BIT	HAL_FLOAT	HAL_S32	HAL_U32
HAL_S64	HAL_U64	Вказівки щодо встановлення шпильок:	HAL_IN	HAL_OUT
HAL_IO				Вказівки щодо параметрів:

Table 5.31: (continued)

HAL_RO	HAL_RW			

Повна назва виводу або параметра утворюється шляхом об'єднання префікса та суфікса за допомогою символу ".", тому в наведеному прикладі створений вивод називається «passthrough.in».

```
h.ready()
```

Після створення всіх виводів та параметрів викличте метод *.ready()*.

5.13.3.1 Зміна префікса

Префікс можна змінити, викликавши метод *.setprefix()*. Поточний префікс можна отримати, викликавши метод *.getprefix()*.

5.13.4 Читання та запис виводів і параметрів

Для виводів та параметрів, які також є власними ідентифікаторами Python, значення можна отримати або встановити за допомогою синтаксису атрибута:

```
h.out = h.in
```

Для всіх пінів, незалежно від того, чи є вони також належними ідентифікаторами Python, значення можна отримати або встановити за допомогою синтаксису нижнього індексу:

```
h['out'] = h['in']
```

Щоб побачити всі піни з їхніми значеннями, getpins повертає всі значення зі словника цього компонента.

```
h.getpins()
>>>{'in': 0.0, 'out': 0.0}
```

5.13.4.1 Виводи керування виходом (HAL_OUT)

Періодично, зазвичай у відповідь на сигнал таймера, всі виводи HAL_OUT повинні «управлятися» шляхом присвоєння їм нового значення. Це слід робити незалежно від того, чи відрізняється значення від останнього присвоєного. Коли вивід підключений до сигналу, його старе вихідне значення не копіюється в сигнал, тому правильне значення з'явиться на сигналі тільки після того, як компонент присвоїть нове значення.

5.13.4.2 Керування двонаправленими (HAL_IO) контактами

Вищезазначене правило не застосовується до двонаправлених виводів. Натомість двонаправлений вивід повинен керуватися компонентом лише тоді, коли компонент бажає змінити значення. Наприклад, у канонічному інтерфейсі енкодера компонент енкодера встановлює контакт «index-enable» лише на **FALSE** (коли бачиться імпульс індексу, а старе значення є **TRUE**), але ніколи не встановлює його на **TRUE**. Повторне керування контактом **FALSE** може спричинити те, що інший підключений компонент поводитиметься так, ніби бачився інший імпульс індексу.

5.13.5 Виходимо

Запит «halcmd unload» для компонента передається як виняток «KeyboardInterrupt». Коли надходить запит на вивантаження, процес повинен або завершитися за короткий час, або викликати метод «.exit()» на компоненті, якщо для завершення процесу вимкнення необхідно виконати значну роботу (наприклад, читання або запис файлів).

5.13.6 Корисні функції

Див. огляд доступних функцій у розділі Інтерфейс Python HAL.

5.13.7 Константи

Використовуйте їх для уточнення деталей, а не значення, яке вони містять.

- 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 Інформація про систему

Прочитайте їх, щоб отримати інформацію про систему реального часу.

- is_kernelspace
- is_rt
- is_sim
- is_userspace

5.14 Канонічні інтерфейси пристроїв

5.14.1 Вступ

У наступних розділах наведено контакти, параметри та функції, що надаються «канонічними пристроями». Усі драйвери пристроїв HAL повинні надавати однакові контакти та параметри і реалізовувати однакову поведінку.

Зверніть увагу, що для канонічного пристрою визначені лише поля `_<io-type>_` та `_<specific-name>_`. Поля `_<device-name>`, `_<device-num>_` та `_<chan-num>_` встановлюються на основі характеристик реального пристрою.

5.14.2 Цифровий вхід

Канонічний цифровий вхід (поле типу вводу/виводу: `digin`) досить простий.

5.14.2.1 Піни

(біт) **in**:: Стан апаратного входу. (біт) **in-not**:: Інвертований стан входу.

5.14.2.2 Параметри

Жоден

5.14.2.3 Функції

(funct) **read**:: Зчитування апаратного забезпечення та встановлення контактів HAL «in» та «in-not».

5.14.3 Цифровий вихід

Канонічний цифровий вихід (поле типу вводу/виводу: `digout`) також дуже простий.

5.14.3.1 Піни

(bit) **out**:: Значення, яке буде записано (можливо, інвертовано) на вихід обладнання.

5.14.3.2 Параметри

(bit) **invert**:: Якщо значення TRUE (ИСТИНА), **out** інвертується перед записом на обладнання.

5.14.3.3 Функції

(funct) **write**:: Зчитайте **out** та **invert**, і відповідно встановіть апаратний вихід.

5.14.4 Аналоговий вхід

Канонічний аналоговий вхід (тип вводу/виводу: `adcin`). Очікується, що він буде використовуватися для аналого-цифрових перетворювачів, які перетворюють, наприклад, напругу, у безперервний діапазон значень.

5.14.4.1 Піни

(float) **value**:: Показник апаратного забезпечення, масштабований відповідно до параметрів **scale** та **offset**.
value = ((вхідний показник, в одиницях, що залежать від апаратного забезпечення) * **scale**) - **offset**

5.14.4.2 Параметри

(float) **масштаб**:: Вхідна напруга (або струм) буде помножена на **масштаб**, перш ніж буде виведена у вигляді **значення**. (float) **зміщення**:: Це буде віднято від вхідної напруги (або струму) обладнання після застосування множника масштабу. (float) **біт_вага**:: Значення одного найменш значущого біта (LSB). Це фактично ступінь гранулярності вхідного зчитування. (float) **hw_offset**:: Значення, що присутнє на вході, коли на вхідний(і) контакт(и) подається 0 вольт.

5.14.4.3 Функції

(funct) **read**:: Зчитувати значення цього аналогового вхідного каналу. Це може бути використано для зчитування окремих каналів або може призвести до зчитування всіх каналів.

5.14.5 Аналоговий вихід

Канонічний аналоговий вихід (тип вводу/виводу: **adcout**). Призначений для будь-якого типу обладнання, яке може видавати більш-менш безперервний діапазон значень. Прикладами є цифро-аналогові перетворювачі або генератори PWM.

5.14.5.1 Піни

(float) **value**:: Значення, яке потрібно записати. Фактичне значення, що виводиться на апаратне забезпечення, залежатиме від параметрів масштабу та зміщення. (bit) **увімкнути**:: Якщо значення false (хибне), то вивести 0 на обладнання, незалежно від виводу **value**.

5.14.5.2 Параметри

(float) **зміщення**:: Це буде додано до **значення** перед оновленням обладнання. (float) **масштаб**:: Це слід встановити таким чином, щоб вхідний сигнал 1 на виводі **value** призводив до того, що аналоговий вихідний вивід зчитуватиме 1 вольт. (float) **high_limit** (необов'язково):: Під час обчислення значення для виведення на обладнання, якщо **value**
offset більше за **high_limit**, тоді замість нього буде використано **high_limit**. (float) **low_limit** (необов'язково):: Під час обчислення значення для виведення на обладнання, якщо **value**
offset менше за **low_limit**, тоді замість нього буде використано **low_limit**. (float) **bit_weight** (необов'язково):: Значення одного найменш значущого біта (LSB) у вольтах (або мА для струмових виходів). (float) **hw_offset** (необов'язково):: Фактична напруга (або струм), яка буде виведена, якщо в апаратне забезпечення записати 0.

5.14.5.3 Функції

(funct) **write**:: Це призводить до виведення обчисленого значення на апаратне забезпечення. Якщо enable має значення false, то вихідним значенням буде 0, незалежно від **value**, **scale** та **offset**. Значення «0» залежить від апаратного забезпечення. Наприклад, двополярний 12-бітний ADC може потребувати запису 0x1FF (середня шкала) в DAC, щоб отримати 0 вольт з виводу апаратного забезпечення. Якщо enable має значення true, зчитайте scale, offset і value та виведіть на adc (**scale** * **value**) + **offset**. Якщо enable має значення false, виведіть 0.

5.15 Інструменти HAL

5.15.1 Halcmd

`halcmd` — це інструмент командного рядка для роботи з HAL. Існує досить повна сторінка довідки для посилання:../man/man1/halcmd.1.html[`halcmd`], яка буде встановлена, якщо ви встановили LinuxCNC з вихідного коду або пакета. Сторінка довідки містить інформацію про використання:

```
man halcmd
```

Якщо ви скомпілювали LinuxCNC для ”запуску на місці”, вам потрібно знайти вихідний код скрипта rip-environment, щоб зробити сторінку довідника доступною:

```
cd toplevel_directory_for_rip_build
. scripts/rip-environment
man halcmd
```

У розділі HAL Tutorial наведено низку прикладів використання halcmd, і він є гарним навчальним посібником з `halcmd`.

5.15.2 Півметра

Halmeter — це «вольтметр» для HAL. Він дозволяє переглянути контакт, сигнал або параметр і відображає поточне значення цього елемента. Він досить простий у використанні. Запустіть його, ввівши `halmeter` у командному рядку X Windows. Halmeter — це програма з графічним інтерфейсом. Вона відкриє невелике вікно з двома кнопками «Select» (Вибрати) та «Exit» (Вийти). Вийти легко — програма закриється. Вибрати відкриває більше вікно з трьома вкладками. Одна вкладка містить список усіх контактів, які наразі визначені в HAL. Наступна вкладка містить список усіх сигналів, а остання вкладка — список усіх параметрів. Клацніть на вкладку, а потім на контакт/сигнал/параметр. Потім клацніть «OK». Списки зникнуть, а у маленькому вікні з’явиться назва та значення вибраного елемента. Дисплей оновлюється приблизно 10 разів на секунду. Якщо натиснути «Прийняти» замість «OK», у маленькому вікні з’явиться назва та значення вибраного елемента, але велике вікно залишиться на екрані. Це зручно, якщо ви хочете швидко переглянути кілька різних елементів.

Ви можете запустити кілька halmeter одночасно, якщо хочете відстежувати кілька елементів. Якщо ви хочете запустити halmeter, не займаючи вікно оболонки, введіть `halmeter &`, щоб запустити його у фоновому режимі. Ви також можете змусити halmeter негайно почати відображати певний елемент, додавши `pin|sig|par[am] _<name>_` до командного рядка. Він відображатиме pin, signal або parameter *<name>* відразу після запуску — якщо такого елемента немає, він просто запуститься у звичайному режимі. І нарешті, якщо ви вказали елемент для відображення, ви можете додати `-s` перед `pin|sig|param`, щоб halmeter використовував невелике вікно. Назва елемента буде відображатися в рядку заголовка, а не під значенням, і не буде кнопок. Це корисно, коли ви хочете розмістити багато вимірювачів на невеликій площі екрана.

Зверніться до розділу Посібник з Halmeter для отримання додаткової інформації.

`halmeter` можна завантажити з терміналу або з AXIS. `halmeter` швидше відображає значення, ніж `halshow`. `halmeter` має два вікна: одне для вибору контакту, сигналу або параметра, що потрібно контролювати, а інше для відображення значення. Одночасно можна відкрити кілька ``halmeter``. Якщо ви використовуєте скрипт для відкриття декількох ``halmeter``, ви можете встановити положення кожного з них за допомогою `-g X Y` відносно верхнього лівого кута екрана. Наприклад:

```
loadusr halmeter pin hm2.0.stepgen.00.velocity-fb -g 0 500
```

Дивіться сторінку довідки для отримання додаткових опцій та розділ Halmeter.

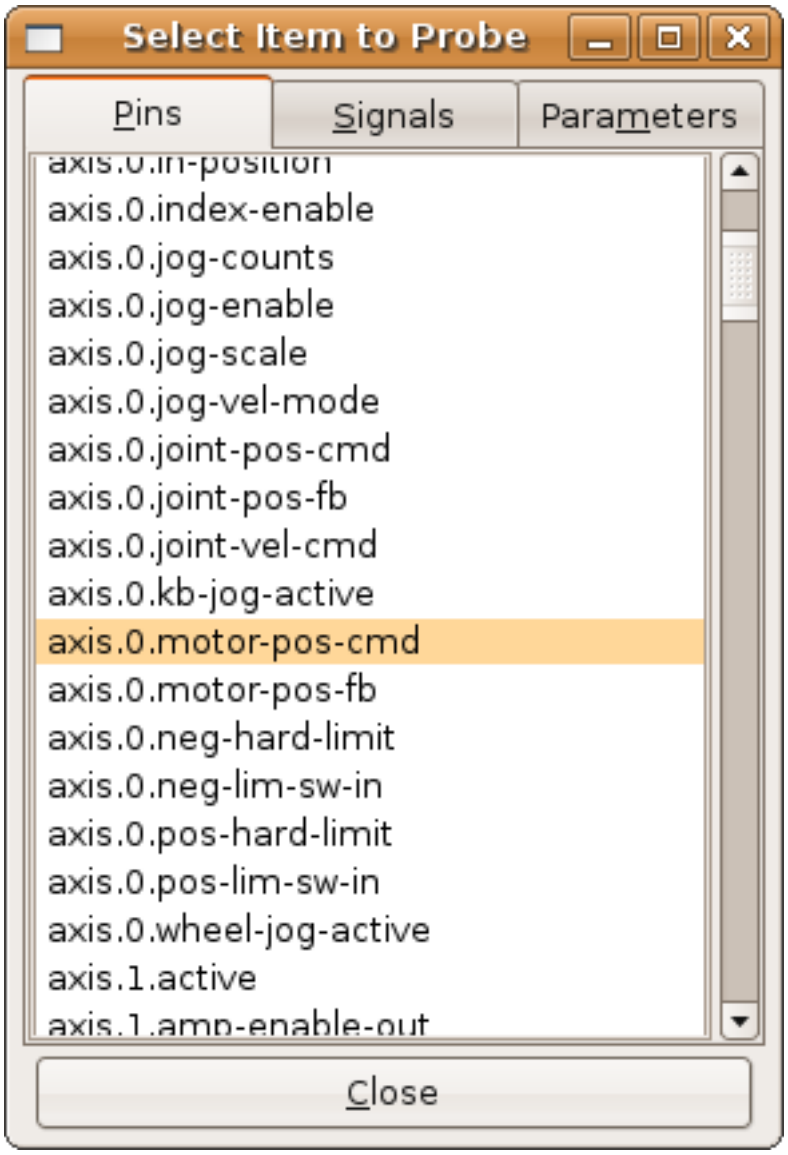


Figure 5.21: Вікно вибору півметра

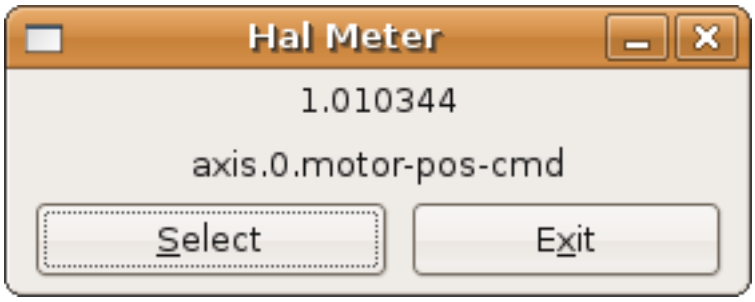


Figure 5.22: Вікно годинника Halmeter

5.15.3 Галшоу

halshow (повний опис використання) можна запустити з командного рядка, щоб відобразити детальну інформацію про вибрані компоненти, контакти, параметри, сигнали, функції та потоки працюючого HAL. Вкладка WATCH забезпечує безперервне відображення вибраних контактів, параметрів та сигналів. Меню «Файл» містить кнопки для збереження елементів спостереження у списку спостереження та завантаження існуючого списку спостереження. Елементи списку спостереження також можна завантажувати автоматично під час запуску. Для використання командного рядка:

```
halshow --help
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яв'':
  halshow [Options] [watchfile]
  b''Ob''b''пb''b''цb''b''ib''b''їb'':
           --help    (this help)
           --fformat format_string_for_float
           --iformat format_string_for_int

b''Пb''b''pb''b''ив''b''мb''b''ib''b''тb''b''кb''b''ив'':
b''Cb''b''тb''b''вb''b''ob''b''pb''b''ib''b''тb''b''ьb'' b''фb''b''ab''b''йb''b''лb'' b' ↩
    'cb''b''пb''b''ob''b''cb''b''тb''b''eb''b''pb''b''eb''b''жb''b''eb''b''нb''b''нb''b' ↩
    'яв'' b''вb'' halshow b''зb''b''ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b' ↩
    'ob''b''юв'': «b''Фb''b''ab''b''йb''b''лb''/b''Зb''b''бb''b''eb''b''pb''b''eb''b''гb''b' ↩
    'тb''b''ив'' b''cb''b''пb''b''ив''b''cb''b''ob''b''кb'' b''cb''b''пb''b''ob''b''cb''b' ↩
    'тb''b''eb''b''pb''b''eb''b''жb''b''eb''b''нb''b''нb''b''яв''».
b''Дb''b''лb''b''яв'' b''ab''b''вb''b''тb''b''ob''b''нb''b''ob''b''мb''b''нb''b''ob''b' ↩
    'гb''b''ob'' b''вb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b' ↩
    'нb''b''яв'' LinuxCNC b''мb''b''ab''b''єb'' b''бb''b''уb''b''тb''b''ив'' b''зb''b''ab''b ↩
    ''пb''b''уb''b''щb''b''eb''b''нb''b''ob''.
```

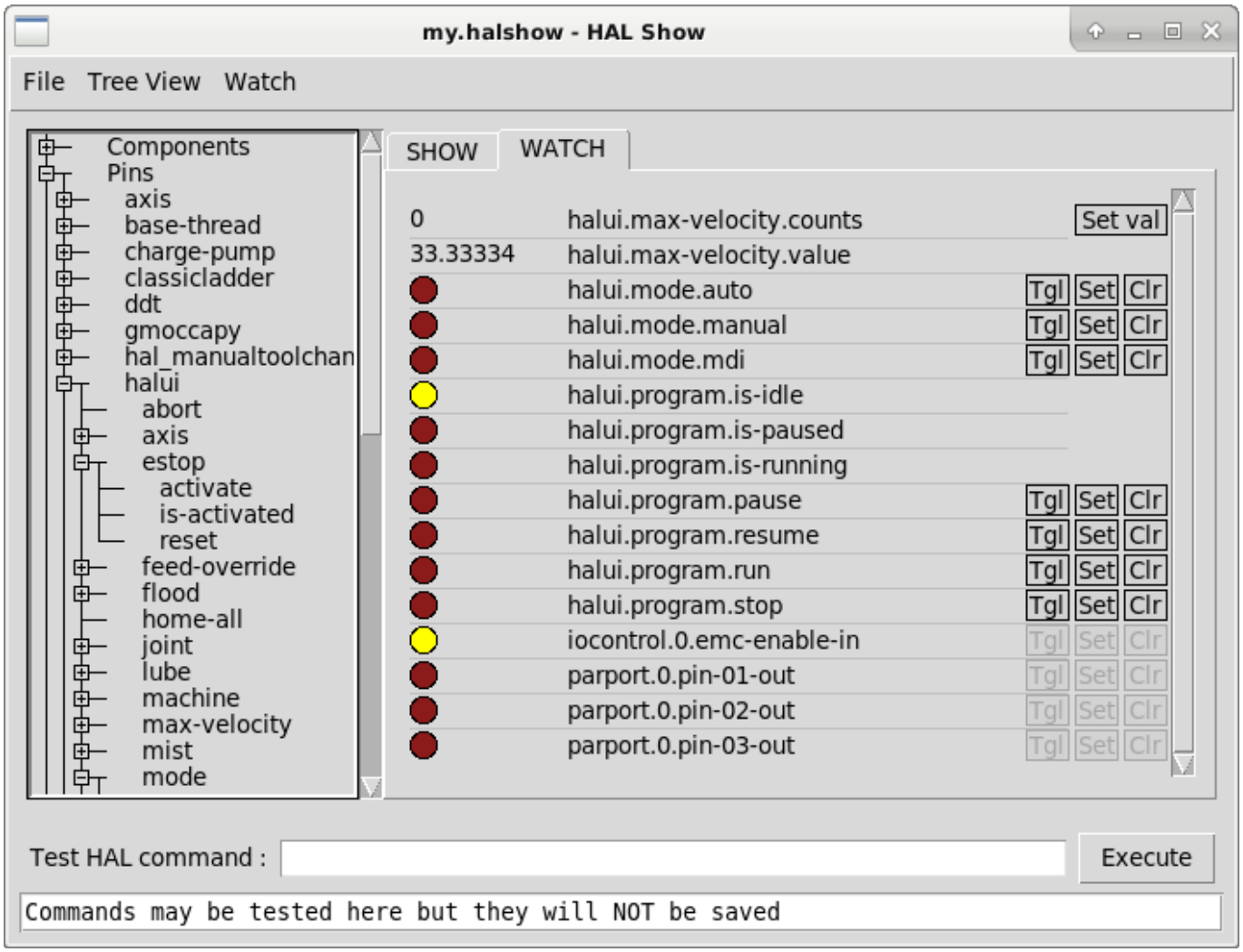


Figure 5.23: Вкладка годинника Halshow

Файл спостереження, створений за допомогою пункту меню «Файл/Зберегти список спостереження», має формат одного рядка з маркерами «pin+», «param+», «sig=+», за якими йде відповідна назва виводу, параметра або сигналу. Пари маркер-назва розділяються пробілом.

Приклад однорядкового файла спостереження

```
pin+joint.0.pos-hard-limit pin+joint.1.pos-hard-limit sig+estop-loop
```

Файл спостереження, створений за допомогою пункту меню «Файл/Зберегти список спостереження (багаторядковий)», форматується окремими рядками для кожного елемента, ідентифікованого парами токен-імена, як описано вище.

Приклад файлу спостереження з розділеними рядками

```
pin+joint.0.pos-hard-limit
pin+joint.1.pos-hard-limit
sig+estop-loop
```

Під час завантаження файлу спостереження за допомогою пункту меню «Файл/Завантажити список спостереження» пари «імена токенів» можуть відображатися як один або кілька рядків. Пусті рядки та рядки, що починаються з символу #, ігноруються.

5.15.4 Галскоп

Halscope — це «осцилограф» для HAL. Він дозволяє фіксувати значення виводів, сигналів і параметрів як функцію часу. Повна інструкція з експлуатації буде розміщена тут пізніше. Наразі дивіться розділ Halscope у розділі підручника, де пояснюються основні принципи роботи.

У меню «Файл» програми halscope є кнопки для збереження конфігурації або відкриття попередньо збереженої конфігурації. Після завершення роботи програми halscope остання конфігурація зберігається у файлі з назвою autosave.halscope.

Файли конфігурації також можна вказати під час запуску halscope з командного рядка. Використання довідки з командного рядка (-h):

```
halscope -h
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яв'':
  halscope [-h] [-i infile] [-o outfile] [num_samples]
```

5.15.5 Пін-код SIM-карти

sim_pin — це утиліта командного рядка для відображення та оновлення будь-якої кількості записуваних виводів, параметрів або сигналів.

Використання sim_pin

```
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яв'':
        sim_pin [Options] name1 [name2 ...] &

b''Ob''b''пb''b''цb''b''ib''b''їb'':
        --help                (this text)
        --title title_string  (window title, default: sim_pin)

b''Пb''b''pb''b''ив''b''мb''b''ib''b''тb''b''кb''b''ab'':  LinuxCNC (b''ab''b''бb''b''ob''  ↩
    b''ab''b''вв''b''тb''b''ob''b''нb''b''ob''b''мb''b''нb''b''ab'' b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''ab'' HAL) b''пb''b''ob''b''вв''b''ив''b''нb''b''нb''b''ab'' ↩
     b''бb''b''уb''b''тb''b''ив'' b''зb''b''ab''b''пb''b''уb''b''щb''b''eb''b''нb''b''ab''.
        b''Ib''b''мb''b''eb''b''нb''b''ob''b''вв''b''ab''b''нb''b''ив''b''йв'' b''eb''b' ↩
            'лb''b''eb''b''мb''b''eb''b''нb''b''тb'' b''мb''b''ob''b''жb''b''eb'' b''вв''b' ↩
            'кb''b''ab''b''зb''b''уb''b''вв''b''ab''b''тb''b''ив'' b''нb''b''ab'' b''кb''b' ↩
            'ob''b''нb''b''тb''b''ab''b''кb''b''тb'', b''пb''b''ab''b''pb''b''ab''b''мb''b' ↩
            'eb''b''тb''b''pb'' b''ab''b''бb''b''ob'' b''cb''b''ив''b''гb''b''нb''b''ab''b' ↩
            'лb''.
        b''Eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb'' b''пb''b''ob''b''вв''b''ив''b' ↩
            'нb''b''eb''b''нb'' b''бb''b''уb''b''тb''b''ив'' b''дb''b''ob''b''cb''b''тb''b' ↩
            'уb''b''пb''b''нb''b''ив''b''мb'' b''дb''b''лb''b''яв'' b''зb''b''ab''b''пb''b' ↩
            'ив''b''cb''b''уb'', b''нb''b''ab''b''пb''b''pb''b''ив''b''кb''b''лb''b''ab''b' ↩
            'дb'':
          b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb'':    IN b''ab''b''бb''b''ob'' I/ ↩
              O (b''ib'' b''нb''b''eb'' b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b' ↩
              'eb''b''нb''b''ив''b''йв'' b''дb''b''ob'' b''cb''b''ив''b''гb''b''нb''b''ab''b ↩
              ''лb''b''уb'' b''зb'' b''зb''b''ab''b''пb''b''ив''b''cb''b''уb''b''юb''b''чb'' ↩
              b''ив''b''мb'' b''пb''b''pb''b''ив''b''cb''b''тb''b''pb''b''ob''b''єb''b' ↩
              'мb'')
          b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'':  RW
          b''cb''b''ив''b''гb''b''нb''b''ab''b''лb'': b''пb''b''ib''b''дb''b''кb''b''лb''b' ↩
              'юb''b''чb''b''eb''b''нb''b''ив''b''йв'' b''дb''b''ob'' b''кb''b''ob''b''нb''b ↩
              ''тb''b''ab''b''кb''b''тb''b''уb'', b''дb''b''ob''b''cb''b''тb''b''уb''b''пb'' ↩
              b''нb''b''ob''b''гb''b''ob'' b''дb''b''лb''b''яв'' b''зb''b''ab''b''пb''b' ↩
              'ив''b''cb''b''уb''.
```

```
        b''Пb''b''ib''b''дb''b''тb''b''pb''b''иb''b''мb''b''уb''b''юb''b''тb''b''ьb''b' ↩
            'cb''b''яb'' b''eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb''b''иb'' HAL b' ↩
            'тb''b''иb''b''пb''b''уb'' bit, s32, u32, float.

        b''Кb''b''оb''b''лb''b''иb'' b''вb''b''кb''b''аb''b''зb''b''аb''b''нb''b''оb'' b' ↩
            'бb''b''ib''b''тb''b''оb''b''вb''b''иb''b''йb'' b''eb''b''лb''b''eb''b''мb''b' ↩
            'eb''b''нb''b''тb'', b''cb''b''тb''b''вb''b''оb''b''pb''b''юb''b''єb''b''тb''b' ↩
            'ьb''b''cb''b''яb'' b''кb''b''нb''b''оb''b''пb''b''кb''b''аb''
        b''дb''b''лb''b''яb'' b''уb''b''пb''b''pb''b''аb''b''вb''b''лb''b''ib''b''нb''b' ↩
            'нb''b''яb'' b''eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb''b''оb''b''мb'' b' ↩
            'оb''b''дb''b''нb''b''иb''b''мb'' b''ib''b''зb'' b''тb''b''pb''b''ьb''b''оb''b' ↩
            'хb'' b''cb''b''пb''b''оb''b''cb''b''оb''b''бb''b''ib''b''вb'', b''вb''b''кb''b' ↩
            'аb''b''зb''b''аb''b''нb''b''иb''b''хb''
        b''pb''b''аb''b''дb''b''ib''b''оb''-b''кb''b''нb''b''оb''b''пb''b''кb''b''аb''b' ↩
            'мb''b''иb'':
            toggle: b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''аb''b''нb''b''нb''b ↩
                ''яb'' b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b''нb''b''яb'' b''пb''b' ↩
                'pb''b''иb'' b''нb''b''аb''b''тb''b''иb''b''cb''b''кb''b''аb''b''нb''b''нb'' ↩
                b''ib'' b''кb''b''нb''b''оb''b''пb''b''кb''b''иb''
            pulse:  b''ib''b''мb''b''пb''b''уb''b''лb''b''ьb''b''cb'' b''eb''b''лb''b''eb'' ↩
                b''мb''b''eb''b''нb''b''тb''b''аb'' b''дb''b''оb'' 1 b''оb''b''дb''b''иb''b' ↩
                'нb'' b''pb''b''аb''b''зb'' b''пb''b''pb''b''иb'' b''нb''b''аb''b''тb''b' ↩
                'иb''b''cb''b''кb''b''аb''b''нb''b''нb''b''ib'' b''кb''b''нb''b''оb''b''пb'' ↩
                b''кb''b''иb''
            hold:   b''уb''b''cb''b''тb''b''аb''b''нb''b''оb''b''вb''b''кb''b''аb'' b''нb'' ↩
                b''аb'' 1 b''пb''b''ib''b''дb'' b''чb''b''аb''b''cb'' b''нb''b''аb''b''тb''b ↩
                ''иb''b''cb''b''кb''b''аb''b''нb''b''нb''b''яb'' b''кb''b''нb''b''оb''b' ↩
                'пb''b''кb''b''иb''
        b''Pb''b''eb''b''жb''b''иb''b''мb'' b''бb''b''ib''b''тb''b''оb''b''вb''b''оb''b' ↩
            'їb'' b''кb''b''нb''b''оb''b''пb''b''кb''b''иb'' b''мb''b''оb''b''жb''b''нb''b' ↩
            'аb'' b''вb''b''кb''b''аb''b''зb''b''аb''b''тb''b''иb'' b''вb'' b''кb''b''оb''b' ↩
            'мb''b''аb''b''нb''b''дb''b''нb''b''оb''b''мb''b''уb''
        b''pb''b''яb''b''дb''b''кb''b''уb'', b''вb''b''ib''b''дb''b''фb''b''оb''b''pb''b' ↩
            'мb''b''аb''b''тb''b''уb''b''вb''b''аb''b''вb''b''шb''b''иb'' b''ib''b''мb'''b' ↩
            'яb'' b''eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb''b''аb'':
            namei/mode=[toggle | pulse | hold]
        b''Яb''b''кb''b''щb''b''оb'' b''pb''b''eb''b''жb''b''иb''b''мb'' b''пb''b''оb''b' ↩
            'чb''b''иb''b''нb''b''аb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' b''вb''b' ↩
            'eb''b''лb''b''иb''b''кb''b''оb''b''їb'' b''лb''b''ib''b''тb''b''eb''b''pb''b' ↩
            'иb'', b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''аb''b''чb''b''ib''
        b''дb''b''лb''b''яb'' b''вb''b''иb''b''бb''b''оb''b''pb''b''уb'' b''ib''b''нb''b' ↩
            'шb''b''иb''b''хb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''ib''b''вb'' b''нb''b' ↩
            'eb'' b''вb''b''ib''b''дb''b''оb''b''бb''b''pb''b''аb''b''жb''b''аb''b''юb''b' ↩
            'тb''b''ьb''b''cb''b''яb''
```

Для отримання повної інформації див. сторінку довідки:

```
man sim_pin
```

Приклад sim_pin (із запущеним LinuxCNC)

```
halcmd loadrt mux2 names=example; halcmd net sig_example example.in0
sim_pin example.sel example.in1 sig_example &
```

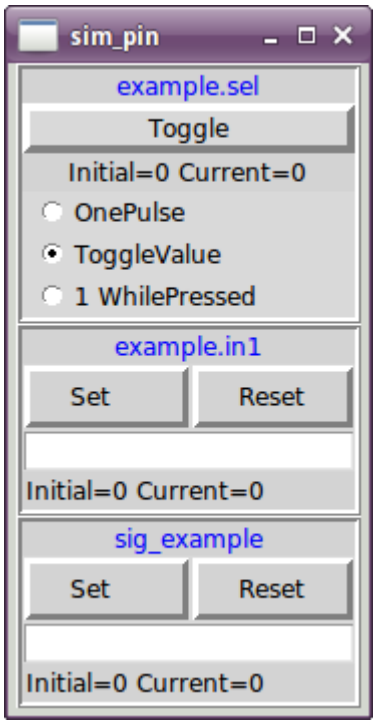


Figure 5.24: Вікно sim_pin

5.15.6 Моделювання зонда

`simulate_probe` це простий графічний інтерфейс для імітації активації штифта motion.probe-input. Використання:

```
simulate_probe &
```

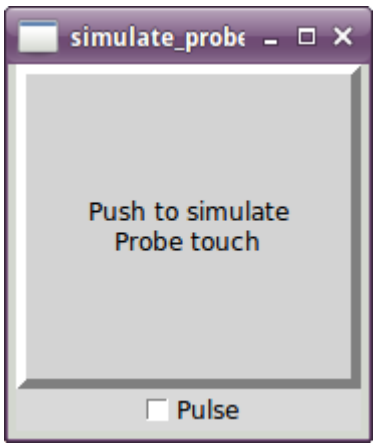


Figure 5.25: Вікно simulate_probe

5.15.7 Гістограма HAL

`hal-histogram` — це утиліта командного рядка для відображення гістограм для виводів HAL.

Використання:

```
   hal-histogram --help | -?
or
   hal-histogram [Options] [pinname]
```

Table 5.32: Опції:

Варіант	**Значення**	**Опис**
--minvalue	мінімальне значення	мінімальний контейнер, за замовчуванням: 0
--binsize	binsize	розмір бінша, за замовчуванням: 100
--nbins	nbins	кількість контейнерів, за замовчуванням: 50
--logscale	0/1	логарифмічна шкала осі Y, за замовчуванням: 1
--text	примітка	текстовий дисплей, за замовчуванням: ””
--show		показувати кількість невідображених nbin, за замовчуванням вимкнено
--verbose		прогрес і налагодження, за замовчуванням вимкнено

Нотатки:

1. LinuxCNC (або інша програма HAL) має бути запущено.
2. Якщо ім’я контакту не вказано, значення за замовчуванням: `motion-command-handler.time`.
3. Цей додаток можна відкрити за 5 позначок.
4. Підтримуються типи розпізнавальних елементів: float, s32, u32, bit.
5. Вивід має бути пов’язаний з потоком, що підтримує обчислення з плаваючою комою. Для базового потоку може знадобитися використання `loadrt motmod ... base_thread_fp=1`.

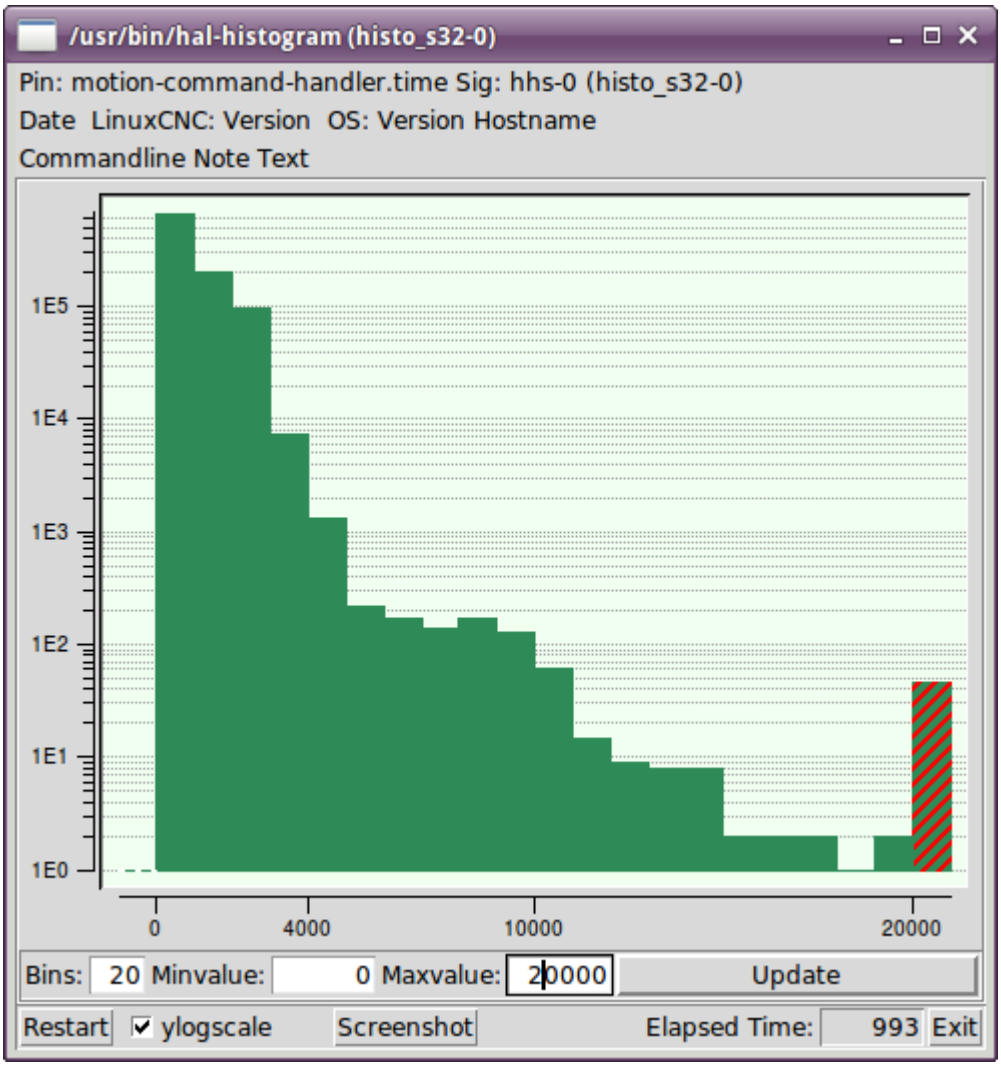


Figure 5.26: Вікно hal-гістограми

5.15.8 Halreport

`halreport` — це утиліта командного рядка, яка створює звіт про підключення HAL для запущеної програми LinuxCNC (або іншої програми HAL). У звіті відображаються всі підключення сигналів і потенційні проблеми. Включена інформація:

1. Опис системи та версія ядра.
2. Сигнали та всі підключені вихідні, вхідні та контакти.
3. Component_function, thread та addf-order кожного виводу.
4. Виводи компонентів, що не працюють у реальному часі, з невпорядкованими функціями.
5. Ідентифікація невідомих функцій для необроблених компонентів.
6. Сигнали без виходу.
7. Сигнали без вхідних даних.
8. Функції без додавання.

9. Теги попередження для компонентів, позначених як застарілі/нерекомендовані в документації.
10. Справжні імена для пінів, що використовують псевдоніми.

Звіт можна згенерувати з командного рядка та спрямувати до вихідного файлу (або stdout, якщо не вказано ім’я вихідного файлу):

Використання halreport

```
b''Bb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''аb''b''нb''b''нb''b''яb'':
  halreport -h | --help (this help)
or
  halreport [outfilename]
```

Щоб створити звіт для кожного запуску LinuxCNC, додайте halreport та назву вихідного файлу як запис [APPLICATIONS]APP до INI-файлу.

Приклад halreport

```
[APPLICATIONS]
APP = halreport /tmp/halreport.txt
```

Впорядкування функцій addf може бути важливим для сервоциклів, де важлива послідовність функцій, обчислених у кожному сервоперіоді. Зазвичай порядок такий:

1. Зчитування вхідних контактів,
2. виконувати функції обробника команд руху та контролера руху,
3. виконати обчислення pid, і нарешті
4. запис вихідних контактів.

Для кожного сигналу в критичному шляху порядок addf вихідного виводу повинен бути чисельно меншим, ніж порядок addf критичних вхідних виводів, до яких він підключений.

Для звичайних сигнальних шляхів, що обробляють входи перемикачів, нереальні виводи тощо, порядок додавання часто не є критичним. Більше того, час зміни значень нереальних виводів не можна контролювати або гарантувати в інтервалах, що зазвичай використовуються для потоків HAL.

Приклад уривків файлу звіту, що показують цикл pid для stepgen hostmot2, що працює в режимі швидкості на машині trivkins з `joint.0`, що відповідає координаті осі 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)
```

У наведеному вище прикладі HALFILE використовує псевдоніми halcmd для спрощення назв контактів плати FPGA hostmot2 за допомогою таких команд, як:

```
alias pin hm2_7i92.0.stepgen.00.position-fb h.00.position-fb
```

Note
Можливе виявлення сумнівних функцій компонентів для

1. непідтримувані (застарілі) компоненти,
2. компоненти, створені користувачем, які використовують кілька функцій або нетрадиційне найменування функцій, або
3. Компоненти, створені за допомогою графічного інтерфейсу користувача (GUI), які не працюють у реальному часі та не мають відмінних характеристик, таких як префікс на основі назви програми з графічним інтерфейсом користувача.

Сумнівні функції позначаються знаком питання ”?”.

Note
Виводи компонента, які неможливо пов’язати з відомою функцією різьби, повідомляють про функцію як ”Невідомо”.

`halreport` генерує звіт про з’єднання (без типів контактів і поточних значень) для запущеної програми HAL, щоб допомогти в проектуванні та перевірці з’єднань. Це допомагає зрозуміти, звідки береться значення контакту. Використовуйте цю інформацію з такими програмами, як `halshow`, `halmeter`, `halscope` або командою `halcmd show` у терміналі.

Chapter 6

Драйвери обладнання

6.1 Драйвер паралельного порту

Компонент hal_parport є драйвером для традиційного паралельного порту ПК. Порт має загалом 17 фізичних контактів. Оригінальний паралельний порт поділяв ці контакти на три групи: дані, керування та стан. Група даних складається з 8 вихідних контактів, група керування — з 4 контактів, а група стану — з 5 вхідних контактів.

На початку 1990-х років був представлений двонаправлений паралельний порт, який дозволяє використовувати групу даних для виводу або вводу. Драйвер HAL підтримує двонаправлений порт і дозволяє користувачеві налаштувати групу даних як ввід або вивід. Якщо порт налаштований як «вивід», він забезпечує в цілому 12 виводів і 5 вводів. Якщо порт налаштований як «ввід», він забезпечує 4 виводи і 13 вводів.

У деяких паралельних портах контакти групи управління є відкритими колекторами, які також можуть бути переведені в низький стан зовнішнім затвором. На платі з відкритими колекторними контактами управління. Якщо налаштований як «x», він забезпечує 8 виходів і 9 входів.

У деяких паралельних портах група керування має двотактні драйвери та не може використовуватися як вхід.

HAL та відкриті колектори
HAL не може автоматично визначити, чи є двонаправлені контакти режиму «x» відкритими колекторами (OC). Якщо це не так, їх не можна використовувати як входи, а спроба перевести їх у стан LOW із зовнішнього джерела може пошкодити апаратне забезпечення.
Щоб визначити, чи має ваш порт виводи «відкритого колектора», завантажте hal_parport у режимі «x». Якщо пристрій не підключено, HAL повинен зчитати вивід як TRUE. Далі підключіть резистор 470 Ом від одного з виводів керування до GND. Якщо отримана напруга на контакті управління близька до 0 В, і HAL тепер зчитує контакт як FALSE, то у вас є порт OC. Якщо отримана напруга далека від 0 В, або HAL не зчитує контакт як FALSE, то ваш порт не може бути використаний в режимі «x».
Зовнішнє обладнання, яке керує керуючими контактами, також повинно використовувати логічні елементи з відкритим колектором, наприклад, 74LS05.
На деяких комп'ютерах налаштування BIOS можуть впливати на можливість використання режиму «x». Найімовірніше, працюватиме режим «SPP».

Жодні інші комбінації не підтримуються, і порт не можна змінити з вхідного на вихідний після встановлення драйвера.

Драйвер parport може керувати до 8 портів (визначено параметром MAX_PORTS у hal_parport.c). Порти нумеруються, починаючи з нуля.

6.1.1 Завантаження

Драйвер hal_parport є компонентом реального часу, тому його необхідно завантажити в потік реального часу за допомогою команди «loadrt». Конфігураційний рядок описує паралельні порти, які будуть використовуватися, та (опціонально) їх типи. Якщо конфігураційний рядок не описує хоча б один порт, це є помилкою.

```
loadrt hal_parport cfg="port [type] [port [type] ...]"
```

Вказівка порту Числа нижче 16 відносяться до паралельних портів, виявлених системою. Це найпростіший спосіб налаштування драйвера `hal_parport`, який співпрацює з драйвером Linux parport_pc, якщо він завантажений. Порт 0 — це перший паралельний порт, виявлений у системі, 1 — наступний і так далі.

Базова конфігурація Це використовуватиме перший паралельний порт, який виявить Linux:

```
loadrt hal_parport cfg="0"
```

Використання адреси порту Натомість адресу порту можна вказати у шістнадцятковому форматі з префіксом *0x*.

Конфігураційний рядок представляє шістнадцяткову адресу порту, за якою опціонально може слідувати напрямок, і все це повторюється для кожного порту. Напрямки можуть бути *in*, *out* або *x* і визначають напрямок фізичних контактів 2–9 роз'єму D-Sub 25. Якщо напрямок не вказано, група даних за замовчуванням буде налаштована як виходи. Наприклад:

Команда для завантаження модуля реального часу hal_partport з додатковим <рядок конфігурації>, щоб вказати порт, на якому очікується підключення плати паралельного порту.

```
loadrt hal_parport cfg="0x278 0x378 in 0x20A0 out"
```

У цьому прикладі встановлюються драйвери для порту 0x0278, з виводами 2–9 як виходи (за замовчуванням, оскільки не вказано ні *in*, ні *out*), порту 0x0378, з виводами 2–9 як входи, та порту 0x20A0, з виводами 2–9, явно вказаними як виходи. Зверніть увагу, що для правильної конфігурації драйверів необхідно знати базову адресу паралельних портів. Для портів шини ISA це зазвичай не є проблемою, оскільки порти майже завжди мають відому адресу, наприклад 0x278 або 0x378, яка зазвичай конфігурується в BIOS. Адреси карт шини PCI зазвичай можна знайти за допомогою команди `lspci -v` у рядку *I/O ports* або в повідомленні ядра після виконання команди `sudo modprobe -a parport_pc`. За замовчуванням адреса не встановлюється, тому якщо `<config-string>` не містить хоча б одну адресу, це вважається помилкою.

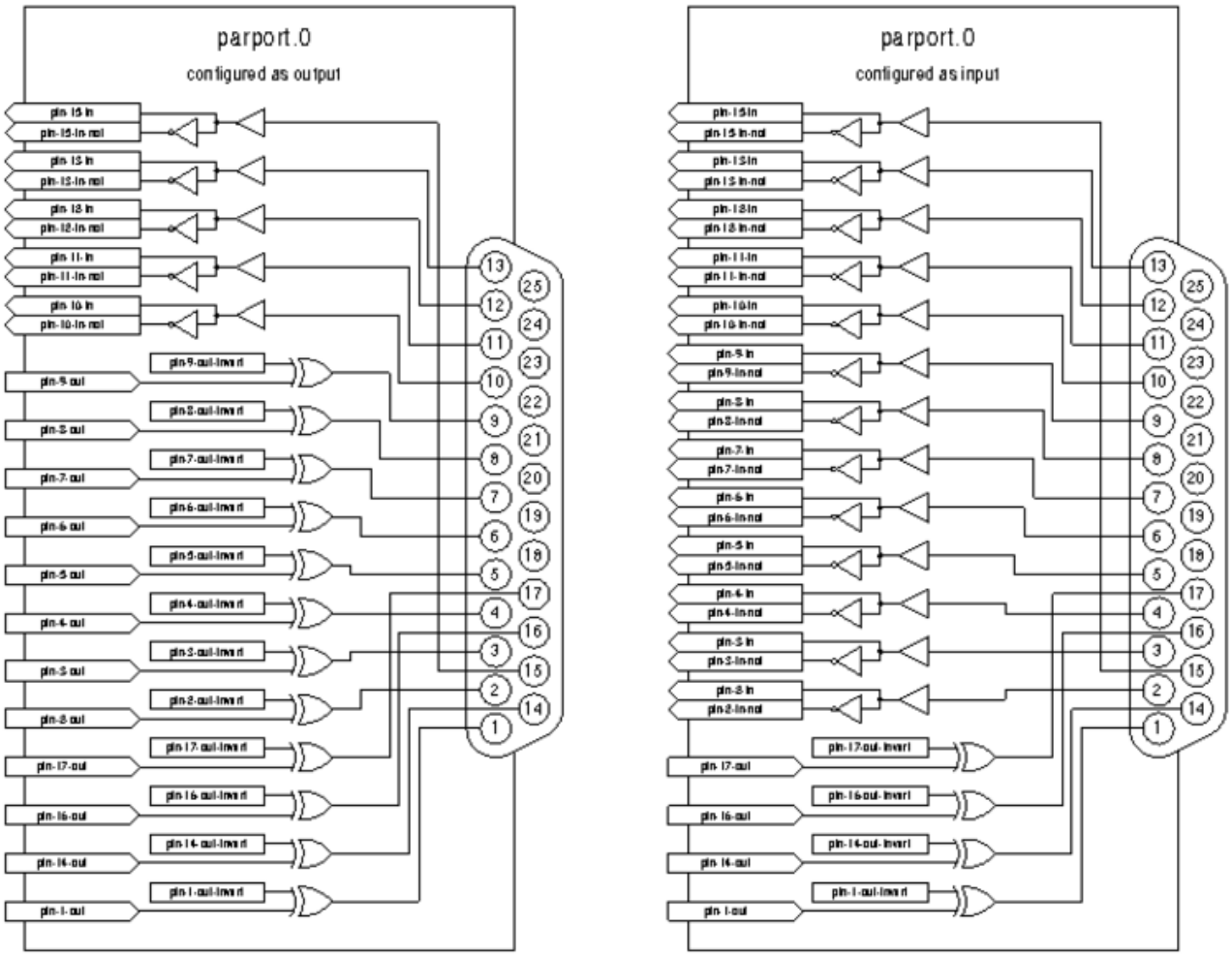


Figure 6.1: Блок-схема Парпорту

Тип Для кожного паралельного порту, що обробляється драйвером hal_parport, можна додатково вказати «тип». Тип може бути одним із «in», «out», «epp» або «x».

Table 6.1: Напрямок паралельного порту

Закріпити	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)

Закріпити	in	out/epp	x
14	out	out	in
15	in	in	in
16	out	out	in
17	out	out	in

Якщо тип не вказано, значення за замовчуванням — «out».

Тип «epp» є таким самим, як «out», але драйвер hal_parport вимагає, щоб порт перейшов у режим EPP. Драйвер hal_parport **не** використовує протокол шини EPP, але в деяких системах режим EPP змінює електричні характеристики порту таким чином, що деяке периферійне обладнання може працювати краще. Відомо, що зарядний насос Gecko G540 вимагає цього на деяких паралельних портах.

Див. примітку вище щодо режиму *x*.

Приклад з двома паралельними портами Це ввімкне два паралельні порти, виявлені системою, перший у режимі виводу, а другий у режимі вводу:

```
loadrt hal_parport cfg="0 out 1 in"
```

Функції читання/запису Parport Ви також повинні направити LinuxCNC на виконання функцій «читання» та «запису».

```
addf parport.0.read base-thread
addf parport.0.write base-thread
```

6.1.2 Адреса порту PCI

Одна хороша PCI-карта parport виготовлена на базі чіпсета Netmos 9815. Вона має хороші сигнали $+5^{-1}$ В і може постачатися з одним або двома портами.

Щоб знайти адреси вводу/виводу для PCI-карт, відкрийте вікно терміналу та скористайтеся командою list pci:

```
lspci -v
```

Знайдіть запис із написом «Netmos». Приклад 2-портової карти:

```
0000:01:0a.0 Communication controller: \
      b''Kb''b''ob''b''нb''b''тb''b''pb''b''ob''b''лb''b''eb''b''pb'' b''бb''b''ab''b''гb'' ↩
          b''ab''b''тb''b''ob''b''кb''b''ab''b''нb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
          'гb''b''ob'' b''вb''b''вb''b''ob''b''дb''b''уb''/b''вb''b''иb''b''вb''b''ob''b' ↩
          'дb''b''уb'' Netmos Technology PCI 9815 (b''вb''b''eb''b''pb''b''cb''b''ib''b' ↩
          'яb'' 01)
b''Пb''b''ib''b''дb''b''cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''ab'': LSI Logic / Symbios ↩
     Logic 2POS (2-b''пb''b''ob''b''pb''b''тb''b''ob''b''вb''b''иb''b''йb'' b''пb''b''ab''b' ↩
    'pb''b''ab''b''лb''b''eb''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''ab''b''дb''b''ab''b' ↩
    'пb''b''тb''b''eb''b''pb'')
b''Пb''b''pb''b''ab''b''пb''b''ob''b''pb''b''цb''b''ib'': b''cb''b''eb''b''pb''b''eb''b' ↩
    'дb''b''нb''b''ib''b''йb'' b''pb''b''ib''b''вb''b''eb''b''нb''b''ьb'' b''pb''b''ob''b' ↩
    'зb''b''pb''b''ob''b''бb''b''кb''b''иb'', 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 результаті експериментів я виявив, що перший порт (вбудований порт) використовує третю адресу зі списку (c000), а другий порт (той, що підключається за допомогою плоского кабелю) використовує першу адресу зі списку (b800). Наступний приклад показує вбудований паралельний порт і паралельний порт PCI, що використовують напрямок виходу за замовчуванням.

```
loadrt hal_parport cfg="0x378 0xc000"
```

Зверніть увагу, що ваші значення будуть відрізнятися. Карти Netmos є Plug-N-Play і можуть змінювати свої налаштування залежно від того, в який слот ви їх вставляєте, тому якщо ви любите «заглядати під капот» і переставляти речі, обов'язково перевірте ці значення перед запуском LinuxCNC.

6.1.3 Піни

- parport.<p>.pin-__<n>__-out (bit) Керує фізичним вихідним контактом.
- parport.<p>.pin-__<n>__-in (біт) Відстежує фізичний вхідний контакт.
- parport.<p>.pin-__<n>__-in-not (bit) Відстежує фізичний вхідний контакт, але інвертований.

Для кожного контакту <*p*> – це номер порту, а <*n*> – це фізичний номер контакту в 25-контактному D-подібному роз'ємі.

Для кожного фізичного вихідного контакту драйвер створює один контакт HAL, наприклад: parport.0

Для кожного фізичного вхідного контакту драйвер створює два контакти HAL, наприклад: parport.0.p та parport.0.pin-12-in-not.

Вивід HAL -in має значення TRUE (ІСТИНА), якщо фізичний вивід має високий рівень, і FALSE (ХИБНІСТЬ), якщо фізичний вивід має низький рівень. Вивід HAL -in-not інвертований і має значення FALSE, якщо фізичний вивід має високий рівень.

6.1.4 Параметри

- parport.__<p>__.pin-__<n>__-out-invert (біт) Інвертує вихідний контакт.
- parport.__<p>__.pin-__<n>__-out-reset (bit) ((лише для виводів -out) TRUE, якщо цей вивод має бути скинутий під час виконання функції -reset.
- parport.__<p>__.reset-time (U32) Час (у наносекундах) між виводами встановлюється опцією -write та скидається функцією -reset, якщо вона ввімкнена.

Параметр -invert визначає, чи є вихідний контакт активним у високому або низькому стані. Якщо -invert має значення FALSE, то встановлення контакту HAL -out у стан TRUE призводить до переведення фізичного контакту у високий стан, а FALSE — у низький. Якщо -invert має значення TRUE, то встановлення контакту HAL -out у стан TRUE призведе до переведення фізичного контакту у низький стан.

6.1.5 Функції

- `parport.`__<p>__.read` (funct) Зчитує фізичні вхідні контакти порту номер <*p*> та оновлює контакти HAL `-in` та `-in-not`.
- `parport.read-all` (funct) Зчитує фізичні вхідні контакти всіх портів та оновлює контакти HAL `-in` та `-in-not`.
- `parport.`__<p>__.write` (funct) Зчитує контакти HAL `-out` порту номер <*p*> та оновлює фізичні вихідні контакти цього порту.
- `parport.write-all` (funct) Зчитує HAL `-out` контакти всіх портів та оновлює всі фізичні вихідні контакти.
- `parport.`__<p>__.reset` (функція) Чекає, поки не мине час `reset-time` з моменту відповідного `write`, а потім скидає виводи до значень, вказаних у налаштуваннях `-out-invert` та `-out-invert`. `reset` повинен бути пізніше в тому ж потоці, що й `write`. Якщо `-reset` має значення TRUE, то функція `reset` встановить контакт на значення `-out-invert`. Це можна використовувати разом з *doublefreq* stepgen для створення одного кроку за період. stepgen stepspace для цього контакту має бути встановлено на 0, щоб увімкнути doublefreq.

Окремі функції передбачені для ситуацій, коли один порт потрібно оновлювати в дуже швидкому потоці, а інші порти можна оновлювати в повільнішому потоці, щоб заощадити час процесора. Ймовірно, не варто одночасно використовувати функцію `-all` та окрему функцію.

6.1.6 Поширені проблеми

Якщо завантаження звітів модуля

```
insmod: error inserting '/home/jepler/emc2/rtlib/hal_parport.ko':
-1 b''Пb''b''pb''b''иb''b''cb''b''тb''b''pb''b''ib''b''йb'' b''ab''b''бb''b''ob'' b''pb''b' ↩
    'eb''b''cb''b''yb''b''pb''b''cb'' b''зb''b''ab''b''йb''b''нb''b''яb''b''тb''b''иb''b' ↩
    'йb''
```

потім переконайтеся, що стандартний модуль ядра «parport_pc» не завантажений примітка:[У пакетах LinuxCNC для Ubuntu файл /etc/modprobe.d/emc2 зазвичай запобігає автоматичному завантаженню «parport_pc».] і що жоден інший пристрій у системі не зайняв порти вводу-виводу.

Якщо модуль завантажується, але, здається, не працює, значить, адреса порту неправильна.

6.1.7 Використання DoubleStep

Щоб налаштувати DoubleStep на паралельному порту, необхідно додати функцію parport.n.reset після parport.n.write і встановити stepspace на 0 та бажаний час скидання. Таким чином, крок може бути підтверджений у кожному періоді в HAL, а потім вимкнений parport після підтвердження протягом часу, зазначеного в `parport.`__n__.reset-time`.

Наприклад:

```
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
```

Більше інформації про DoubleStep можна знайти на вікі.

6.1.8 probe_parport

У сучасних ПК паралельні порти можуть вимагати конфігурації plug and play (PNP) перед тим, як їх можна буде використовувати. Модуль ядра *probe_parport* конфігурує всі наявні порти PNP. Його необхідно завантажити перед *hal_parport*. На машинах без порту PNP його можна завантажити, але це не матиме ніякого ефекту.

6.1.8.1 Встановлення probe_parport

Якщо, коли модуль ядра `parport_pc` завантажується командою:

```
sudo modprobe -a parport_pc; sudo rmmod parport_pc
```

Ядро Linux виводить повідомлення, подібне до:

```
b''пb''b''ab''b''pb''b''пb''b''ob''b''pb''b''тb'': b''вb''b''иb''b''яb''b''вb''b''лb''b' ↩
    'eb''b''нb''b''ob'' b''пb''b''ab''b''pb''b''пb''b''ob''b''pb''b''тb'' PnPBIOS.
```

Тоді використання цього модуля, ймовірно, буде необхідним.

Нарешті, слід завантажити компоненти HAL parport:

```
loadrt probe_parport
loadrt hal_parport ...
```

6.2 Драйвер AX5214H

Axiom Measurement & Control AX5214H — це 48-канальна цифрова плата вводу-виводу. Вона підключається до шини ISA і нагадує пару мікросхем 8255. Насправді це може бути пара мікросхем 8255, але я не впевнений. Якщо/коли хтось почне розробляти драйвер для 8255, йому слід подивитися на код ax5214, оскільки велика частина роботи вже зроблена.

6.2.1 Встановлення

```
loadrt hal_ax5214h cfg="<config-string>"
```

Конфігураційний рядок складається з шістнадцяткової адреси порту, за якою йде 8-символьний рядок з літер «I» та «O», що встановлює групи контактів як входи та виходи. Перші два символи встановлюють напрямок перших двох 8-бітних блоків контактів (0-7 та 8-15). Наступні два символи встановлюють блоки з 4 контактами (16-19 і 20-23). Потім шаблон повторюється: ще два блоки з 8 бітами (24-31 і 32-39) і два блоки з 4 бітами (40-43 і 44-47). Якщо встановлено більше однієї плати, дані для другої плати йдуть після даних для першої. Наприклад, рядок «0x220 IIIOIIOO 0x300 OIOOIOIO» встановлює драйвери для двох плат. Перша плата знаходиться за адресою 0x220 і має 36 входів (0-19 і 24-39) та 12 виходів (20-23 і 40-47). Друга плата знаходиться за адресою 0x300 і має 20 входів (8-15, 24-31 і 40-43) та 28 виходів (0-7, 16-23, 32-39 і 44-47). В одній системі можна використовувати до 8 плат.

6.2.2 Піни

- *(bit) ax5214.<номер_плати>.out-<номер_виводу>* — Керує фізичним вихідним виводом.
- *(bit) ax5214.<boardnum>.in-<pinnum>* — Відстежує фізичний вхідний контакт.
- *(bit) ax5214.<boardnum>.in-<pinnum>-not* — Відстежує фізичний вхідний контакт, інвертований.

Для кожного виводу <boardnum> – це номер плати (починається з нуля), а <pinnum> – це номер каналу вводу/виводу (від 0 до 47).

Зверніть увагу, що драйвер передбачає активні сигнали LOW. Це необхідно для коректної роботи таких модулів, як OPTO-22 (TRUE означає вихід увімкнено або вхід під напругою). Якщо сигнали використовуються безпосередньо без буферизації або ізоляції, необхідно враховувати інверсію. Контакт in- HAL має значення TRUE, якщо фізичний контакт має низький рівень (модуль OPTO-22 під напругою), і FALSE, якщо фізичний контакт має високий рівень (модуль OPTO-22 вимкнений). Контакт in-<pinnum>-not HAL інвертований — він має значення FALSE, якщо фізичний контакт має низький рівень (модуль OPTO-22 під напругою). Підключивши сигнал до одного або іншого, користувач може визначити стан входу.

6.2.3 Параметри

- *(bit) ax5214.<boardnum>.out-<pinnum>-invert* — Інвертує вихідний контакт.

Параметр -invert визначає, чи є вихідний контакт активним високим або активним низьким. Якщо -invert має значення FALSE, установка виходу HAL TRUE призводить до низького рівня фізичного виводу, вмикаючи підключений модуль OPTO-22, а FALSE призводить до високого рівня, вимикаючи модуль OPTO-22. Якщо -invert має значення TRUE, установка виходу HAL TRUE призведе до високого рівня фізичного виводу і вимкнення модуля.

6.2.4 Функції

- *(funct) ax5214.<boardnum>.read* — Зчитує всі цифрові входи на одній платі.
- *(funct) ax5214.<boardnum>.write* — Записує всі цифрові виходи на одну плату.

6.3 Загальний водій мехатроніки

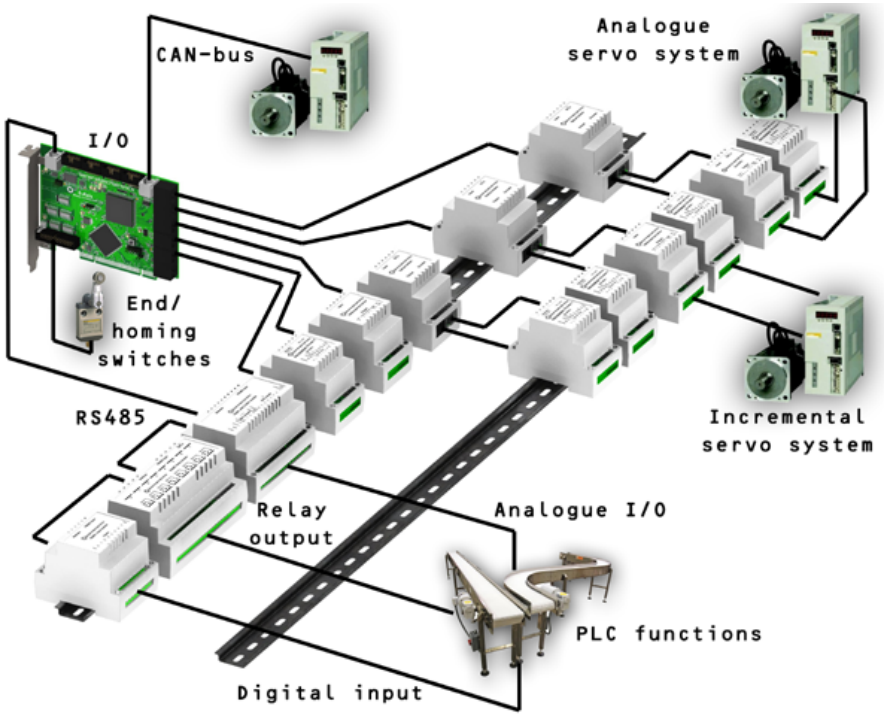
Система керування рухом на базі карти General Mechatronics GM6-PCI

Для отримання детального опису зверніться до https://www.generalmechatronics.com/data/products/-robot_controller/PCI_UserManual_eng.pdf [Посібник із системної інтеграції].

Карта управління рухом GM6-PCI базується на FPGA та інтерфейсі PCI bridge ASIC. Невелику автоматизовану виробничу комірку можна контролювати за допомогою короткої процедури інтеграції системи. На наступному малюнку показано типове підключення пристроїв, пов’язаних із системою управління:

- Він може керувати до шести осей, кожна з яких може бути кроковим двигуном, інтерфейсом шини CAN або аналоговим сервоприводом.
- GPIO: Чотири по вісім контактів вводу/виводу розміщені на стандартних плоских кабельних роз’ємах.

- Модулі розширення вводу-виводу RS485: шина RS485 була розроблена для взаємодії з компактними модулями розширення, що встановлюються на DIN-рейку. Наразі доступні 8-канальні цифрові входи, 8-канальні релейні виходи та аналогові модулі вводу-виводу (4 виходи +/-10 В та 8 входів +/-5 В). До шини можна підключити до 16 модулів.
- 20 оптично ізольованих вхідних контактів: шість по три для прямого підключення двох кінцевих вимикачів та одного датчика самонаведення для кожного шарніра. А також два оптично ізольовані входи аварійної зупинки.



Встановлення:

```
loadrt hal_gm
```

Під час завантаження (або спроби завантаження) драйвер виводить деякі корисні повідомлення про налагодження до журналу ядра, які можна переглянути за допомогою dmesg.

В одній системі можна використовувати до 3 плат.

На платі GM6-PCI можна знайти такі роз’єми:

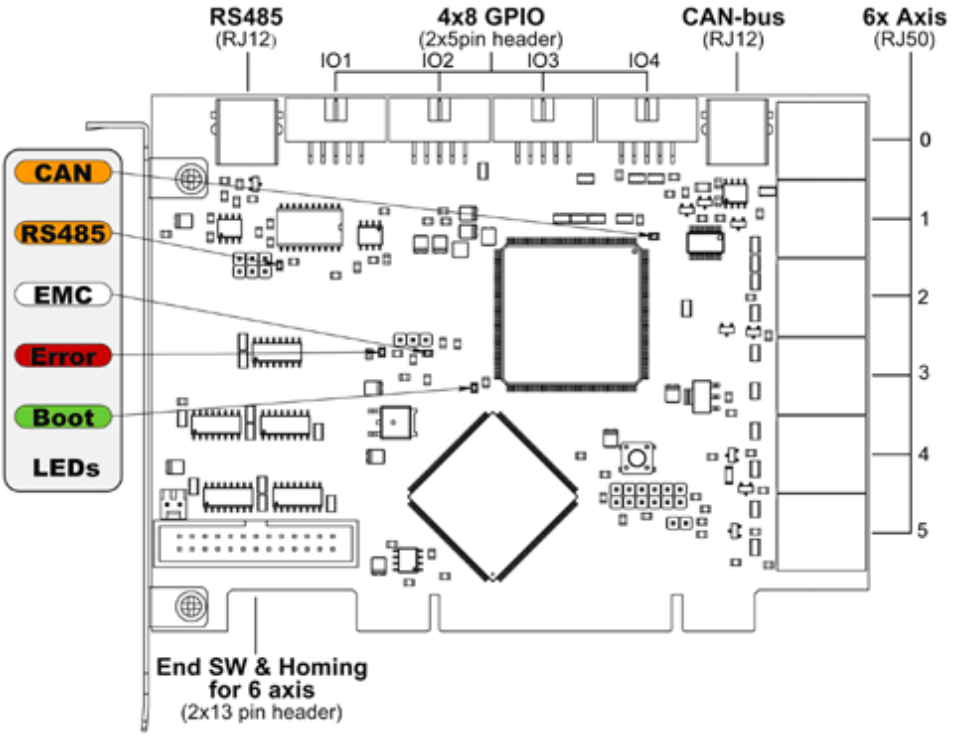


Figure 6.2: Роз’єми та світлодіоди карти GM6-PCI

6.3.1 I/O роз’єми

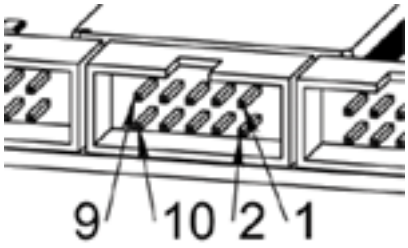


Figure 6.3: Нумерація контактів роз’ємів GPIO

Table 6.2: Розпіновка роз’ємів GPIO

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

Кожен контакт можна налаштувати як цифровий вхід або вихід. Карта керування рухом GM6-PCI має 4 роз'єми загального призначення для вводу-виводу (GPIO), кожен з яких має вісім настроюваних входів-виходів. Кожен контакт GPIO та назва параметра починаються наступним чином:

```
gm.<card_no>.gpio.<gpio_con_no>
```

де *<gpio_con_no>* становить від 0 до 3.

Стан першого контакту першого роз'єму GPIO на карті GM6-PCI.

```
gm.0.gpio.0.in-0
```

Виводи HAL оновлюються функцією

```
gm.<card_no>.read
```

6.3.1.1 Піни

Table 6.4: Контакти GPIO

Піни	Тип і напрямок	Опис піна
`.in-<0-7>`	(біт, вихід)	Вхідний контакт
`.in-not-<0-7>`	(біт, вихід)	Негативний вхідний контакт
`.out-<0-7>`	(біт, В)	Вихідний контакт. Використовується лише тоді, коли GPIO налаштовано на вихід.

6.3.1.2 Параметри

Table 6.5: Параметри GPIO

Піни	Тип і напрямок	Опис параметра
`.is-out-<0-7>`	(біт, R/W)	Коли значення True (Істина), відповідний GPIO (інтерфейс введення-виведення) встановлюється на вихід тотемного стовпа, інакше — на вхід з високим імпедансом.
`.invert-out-<0-7>`	(біт, R/W)	Якщо значення True (Істина), значення виводу буде інвертовано. Використовується, коли вивод налаштовано як вихід.

6.3.2 Роз’єми осей

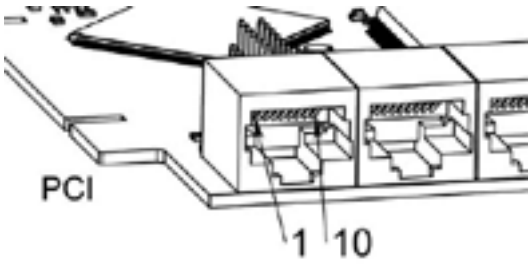


Figure 6.4: Нумерація контактів роз’ємів осей

Table 6.6: Розпіновка роз’ємів осей

1	Енкодер A
2	+5 Volt (PC)
3	Енкодер B
4	Індекс кодера
5	Розлом
6	Живлення ввімкнено
7	Крок/CCW/B
8	Напрямок/CW/A
9	Земля (PC)
10	Послідовна лінія DAC

6.3.2.1 Модулі інтерфейсу Axis

Невеликі інтерфейсні модулі, що монтуються на DIN-рейку, забезпечують простий спосіб підключення різних типів сервомодулів до роз’ємів осей. У посібнику з інтеграції систем (https://www.generalmechatr data/products/robot_controller/PCI_UserManual_eng.pdf) представлено сім різних конфігурацій систем для оцінки типових застосувань. Також у посібнику з інтеграції систем можна знайти докладний опис модулів осей.

Для оцінки відповідної структури сервоприводу модулі необхідно підключити, як показано на наступній блок-схемі:

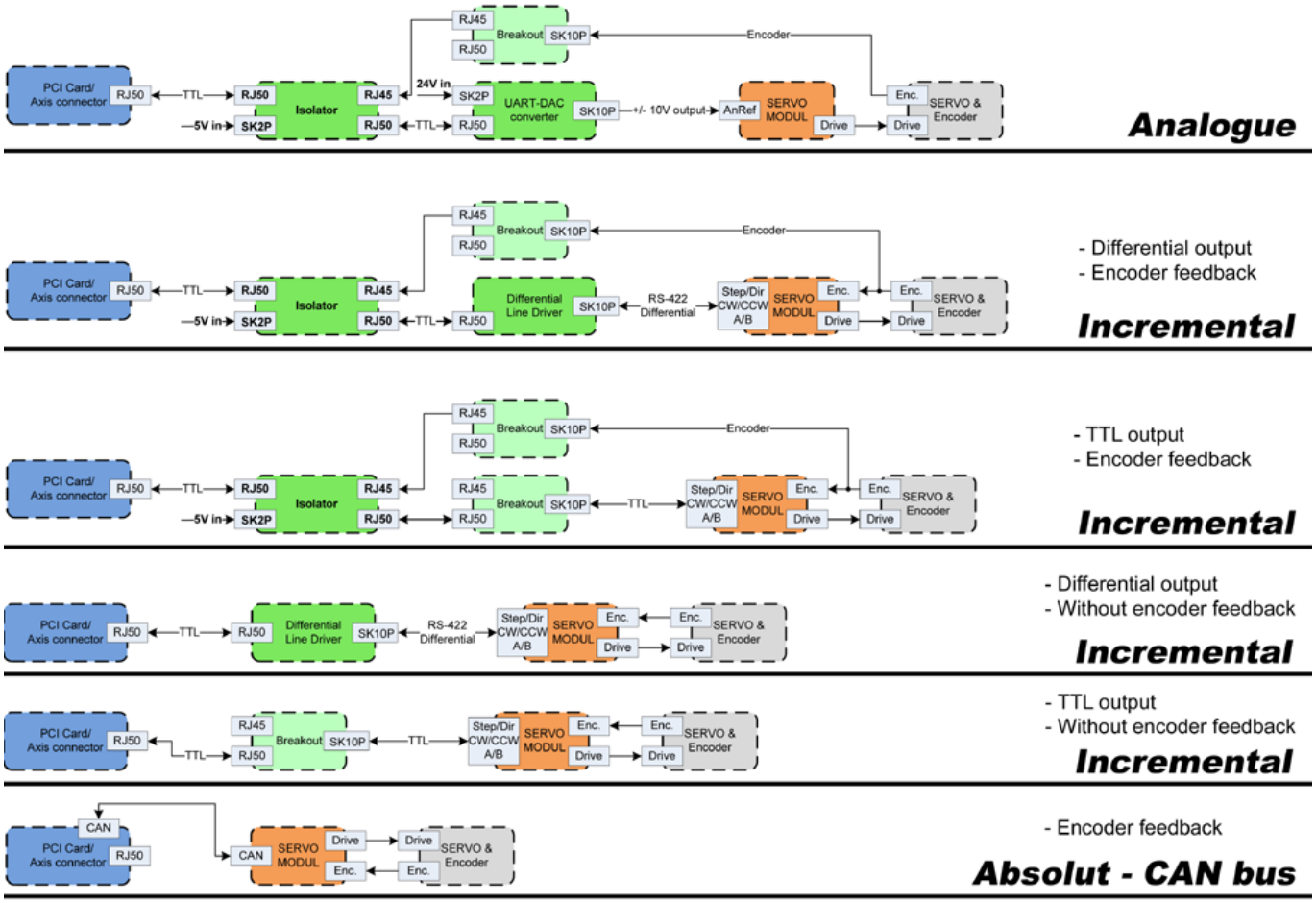


Figure 6.5: Інтерфейси сервоосі

6.3.2.2 Енкодер

Плата керування рухом GM6-PCI має шість модулів енкодера. Кожен модуль енкодера має три канали:

- Канал-A
- Канал-B
- Канал-I (індекс)

Він здатний рахувати сигнали квадратурного енкодера або сигнали кроку/напрямку. Кожен модуль енкодера підключається до входів відповідного роз'єму осі RJ50.

Кожна назва виводу та параметра енкодера починається так:

```
gm.<card_no>.encoder.<axis_no>
```

де *<номер_осі>* має значення від 0 до 5. Наприклад, `gm.0.encoder.0.position` посилається на положення модуля енкодера осі 0.

Плата GM6-PCI підраховує сигнал енкодера незалежно від LinuxCNC. Піни HAL оновлюються функцією:

```
gm.<card_no>.read
```

Table 6.7: Виводи енкодера

Піни	**Тип і напрямок**	**Опис піна**
`.reset`	(біт, B)	Якщо значення True, скидає лічильники та позицію до нуля.
`.rawcounts`	(s32, Вихід)	Необроблений підрахунок – це підрахунки, на які не впливає скидання або індексний імпульс.
`.counts`	(s32, Вихід)	Позиція в лічильниках кодера.
`.position`	(float, Вихід)	Позиція в масштабованих одиницях (=.counts/.position-scale).
`.index-enabled`	(біт, IO)	Якщо True, підрахунок і позиція округлюються або скидаються (залежно від режиму індексу) при наступному передньому фронті каналу I. Кожного разу, коли позиція скидається через індекс, вивід `index-enabled` встановлюється в 0 і залишається 0, поки підключений вивід HAL не встановить його.
`.velocity`	(float, Вихід)	Швидкість у масштабованих одиницях за секунду. Енкодер GM використовує високочастотний апаратний таймер для вимірювання часу між імпульсами енкодера з метою обчислення швидкості. Це значно зменшує квантувальний шум у порівнянні з простим диференціюванням вихідних даних про положення. Коли виміряна швидкість нижча за мінімальну оцінку швидкості, вихідні дані про швидкість дорівнюють 0.

Table 6.8: Параметри енкодера

Параметри	**Введення тексту та читання/запис**	**Опис параметра**
`.counter-mode`	(біт, R/W)	Якщо значення True, лічильник підраховує кожен передній фронт вхідного сигналу каналу A у напрямку, визначеному каналом B. Це корисно для підрахунку вихідного сигналу одноканального (неквадратурного) або датчика сигналу кроку/напрямку. Якщо значення False, підрахунок відбувається в квадратурному режимі.

Table 6.8: (continued)

Параметри	**Введення тексту та читання/запис**	**Опис параметра**
`.index-mode`	(біт, R/W)	Коли True і .index-enabled також true, .counts і .position округлюються (на основі .counts-per-rev) на передньому фронті каналу I. Це корисно для виправлення помилок декількох імпульсів, спричинених шумом. У режимі округлення важливо правильно встановити параметр .counts-per-rev. Коли .index-mode має значення False, а .index-enabled має значення true, .counts і .position скидаються при імпульсі каналу I.
`.counts-per-rev`	(s32, R/V)	Визначте, скільки імпульсів знаходиться між двома імпульсами індексу. Використовується тільки в круговому режимі, тобто коли параметри .index-enabled і .index-mode мають значення True. Енкодер GM обробляє сигнал енкодера в режимі 4x, тому, наприклад, у випадку енкодера 500 CPR його слід встановити на 2000. Цей параметр можна легко виміряти, встановивши .index-enabled True і .index-mode False (так, щоб .counts скидався на імпульсі каналу I), а потім перемістивши вісь вручну і подивившись максимальне значення .counts pin в halmeter.
`.index-invert`	(біт, R/W)	Якщо значення True, подія каналу I (скидання або округлення) відбувається на спадаючому фронті сигналу каналу I, інакше – на зростаючому фронті.
`.min-speed-estimate`	(float, R/W)	Визначте мінімальну виміряну величину швидкості, при якій швидкість буде встановлена як відмінна від нуля. Занадто низьке значення цього параметра призведе до того, що швидкість буде довго повертатися до нуля після припинення надходження імпульсів від енкодера.
`.position-scale`	(float, R/W)	Масштабування в кількості відліків на одиницю довжини. .position=.counts/.position-scale. Наприклад, якщо position-scale дорівнює 2000, то 1000 відліків кодера забезпечать позицію 0,5 одиниці.

Налаштування модуля енкодера осі 0 для отримання сигналу квадратурного енкодера 500 CPR та використання скидання в кругле положення.

```
setp gm.0.encoder.0.counter-mode 0         # 0: quad, 1: stepDir
setp gm.0.encoder.0.index-mode 1           # 0: b''cb''b''кb''b''иb''b''дb''b''ab''b''нb''b ↩
    ''нb''b''яb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb'' b''зb''b''ab'' b''іb''b' ↩
    'нb''b''дb''b''eb''b''кb''b''cb''b''ob''b''мb'', 1: b''ob''b''кb''b''pb''b''уb''b''гb''b ↩
    ''лb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb'' b' ↩
    'зb''b''ab'' b''іb''b''нb''b''дb''b''eb''b''кb''b''cb''b''ob''b''мb''
setp gm.0.encoder.0.counts-per-rev 2000    # b''Пb''b''pb''b''ob''b''цb''b''eb''b''cb''b' ↩
    'ob''b''pb'' GM b''ob''b''бb''b''pb''b''ob''b''бb''b''лb''b''яb''b''єb'' b''eb''b''нb''b ↩
    ''кb''b''ob''b''дb''b''eb''b''pb'' b''уb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''іb'' 4x ↩
    , 4x500=2000
setp gm.0.encoder.0.index-invert 0         #
setp gm.0.encoder.0.min-speed-estimate 0.1 # b''вb'' b''ob''b''дb''b''иb''b''нb''b''иb''b' ↩
    'цb''b''іb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb''/b''cb''
setp gm.0.encoder.0.position-scale 20000   # 10 b''ob''b''бb''b''eb''b''pb''b''тb''b''іb''b ↩
    ''вb'' b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb''b''ab'' b''зb''b''мb''b''уb''b' ↩
    'шb''b''уb''b''юb''b''тb''b''ьb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''уb'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''мb''b''іb''b''cb''b''тb''b''иb''b''тb''b''иb''b''cb''b''яb'' b' ↩
    'нb''b''ab'' b''ob''b''дb''b''нb''b''уb'' b''ob''b''дb''b''иb''b''нb''b''иb''b''цb''b' ↩
    'юb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb'' (10x2000)
```

Підключення положення енкодера до зворотного зв’язку щодо положення шарніра LinuxCNC

```
net Xpos-fb gm.0.encoder.0.position => joint.0.motor-pos-fb
```

6.3.2.3 Модуль StepGen

Карта управління рухом GM6-PCI має шість модулів StepGen, по одному для кожного з’єднання. Кожен модуль має два вихідні сигнали. Він може генерувати імпульси Step/Direction, Up/Down або Quadrature (A/B). Кожен модуль StepGen підключений до контактів відповідного роз’єму осі RJ50.

Кожна назва виводу та параметра StepGen починається так:

```
gm.<card_no>.stepgen.<axis_no>
```

де <*номер_осі*> має значення від 0 до 5. Наприклад, `gm.0.stepgen.0.position-cmd` посилається на команду позиціонування модуля StepGen осі 0 на картці 0.

Плата GM6-PCI генерує крокові імпульси незалежно від LinuxCNC. Виводи HAL оновлюються функцією

```
gm.<card_no>.write
```

Table 6.9: Виводи модуля StepGen

Піни	Тип і напрямок	Опис піна
`.enable`	(біт, B)	StepGen генерує імпульси лише тоді, коли цей контакт перебуває в стані ”true”.
`.count-fb`	(s32, Вихід)	Зворотній зв’язок щодо положення в одиницях відліку.

Table 6.9: (continued)

Піни	**Тип і напрямок**	**Опис піна**
`.position-fb`	(float, Вихід)	Зворотній зв’язок щодо положення в одиниці положення.
`.position-cmd`	(float, In)	Задана позиція в одиницях позиції. Використовується лише в режимі позиції.
`.velocity-cmd`	(float, In)	Задана швидкість в одиницях положення за секунду. Використовується лише в режимі швидкості.

Table 6.10: Параметри модуля StepGen

Параметри	**Введення тексту та читання/запис**	**Опис параметра**
`.step-type`	(u32, R/W)	При значенні 0 модуль генерує сигнал Step/Dir. При значенні 1 він генерує сигнали Up/Down step. А при значенні 2 він генерує квадратурні вихідні сигнали.
`.control-type`	(біт, R/W)	Якщо значення True (Так), .velocity-cmd використовується як опорне значення, а velocityvcontrol розраховує вихідну частоту імпульсів. Якщо значення False (Неправда), .position-cmd використовується як опорне значення, а position control розраховує вихідну частоту імпульсів.
`.invert-step1`	(біт, R/W)	Інвертувати вихід каналу 1 (сигнал Step у режимі StepDir)
`.invert-step2`	(біт, R/W)	Інвертувати вихід каналу 2 (сигнал Dir у режимі StepDir)
`.maxvel`	(float, R/W)	Максимальна швидкість в одиницях позиції за секунду. Якщо встановлено значення 0.0, параметр .maxvel ігнорується.
`.maxaccel`	(float, R/W)	Максимальне прискорення в одиницях положення за секунду в квадраті. Якщо встановлено значення 0.0, параметр .maxaccel ігнорується.
`.position-scale`	(float, R/W)	Масштабування в кроках на одиницю довжини.
`.steplen`	(u32, R/W)	Тривалість крокового імпульсу в наносекундах.
`.stepspace`	(u32, R/W)	Мінімальний час між двома східчастими імпульсами в наносекундах.
`.dirdelay`	(u32, R/W)	Мінімальний час між ступінчастим імпульсом та зміною напрямку в наносекундах.

Для оцінки відповідних значень дивіться часові діаграми нижче:

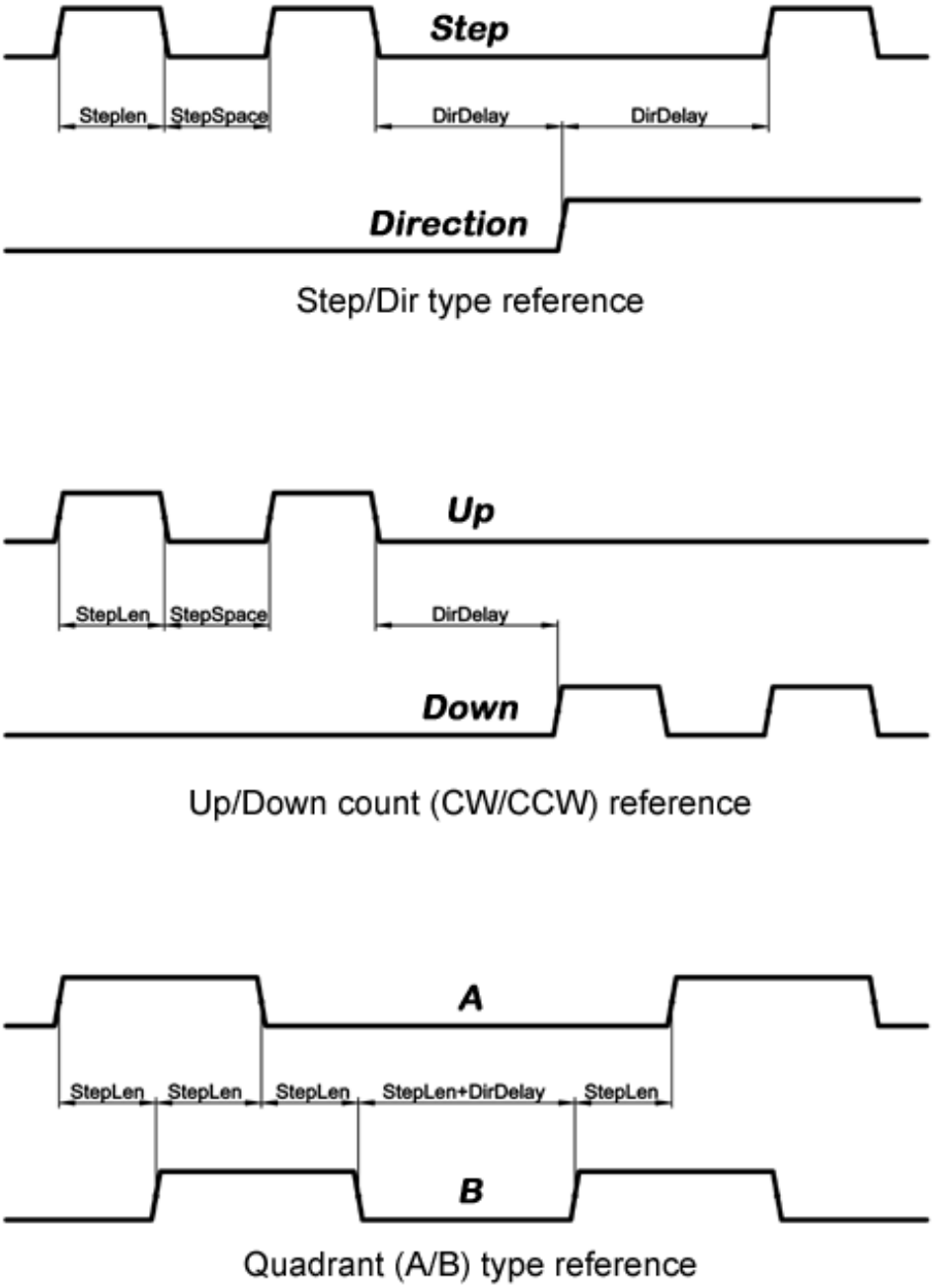


Figure 6.6: Діаграми синхронізації опорного сигналу

Налаштування модуля StepGen осі 0 для генерації 1000 крокових імпульсів на одиницю позиції

```
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            # b''нb''b''eb'' b''вb''b''cb''b''тb''b''ab''b' ↩
    'нb''b''ob''b''вb''b''лb''b''юb''b''йb''b''тb''b''eb'' maxvel b''дb''b''лb''b''яb'' b' ↩
    'гb''b''eb''b''нb''b''eb''b''pb''b''ab''b''тb''b''ob''b''pb''b''ab'' b''кb''b''pb''b' ↩
    'ob''b''кb''b''ib''b''вb'',
```

```
                                    # b''нв''b''eb''b''xb''b''ab''b''йb'' b''ib''b' ↩
                                        'нв''b''тb''b''eb''b''pb''b''пb''b''ob''b''лb''b ↩
                                        ''яb''b''тb''b''ob''b''pb'' b''кb''b''ob''b' ↩
                                        'нв''b''тb''b''pb''b''ob''b''лb''b''юb''b''єb''  ↩
                                        b''йb''b''ob''b''гb''b''ob''.
setp gm.0.stepgen.0.maxaccel 0      # b''нв''b''eb'' b''вb''b''cb''b''тb''b''ab''b' ↩
    'нв''b''ob''b''вb''b''лb''b''юb''b''йb''b''тb''b''eb'' b''мb''b''ab''b''кb''b''cb''b' ↩
    'иb''b''мb''b''ab''b''лb''b''ьb''b''нв''b''eb'' b''пb''b''pb''b''иb''b''cb''b''кb''b' ↩
    'ob''b''pb''b''eb''b''нв''b''нв''b''яb'' b''дb''b''лb''b''яb''
                                    # b''гb''b''eb''b''нв''b''eb''b''pb''b''ab''b''тb'' ↩
                                        b''ob''b''pb''b''ab'' b''кb''b''pb''b''ob''b' ↩
                                        'кb''b''ib''b''вb'', b''нв''b''eb''b''xb''b' ↩
                                        'ab''b''йb'' b''ib''b''нв''b''тb''b''eb''b''pb'' ↩
                                        b''пb''b''ob''b''лb''b''яb''b''тb''b''ob''b' ↩
                                        'pb'' b''кb''b''ob''b''нв''b''тb''b''pb''b''ob'' ↩
                                        b''лb''b''юb''b''єb'' b''йb''b''ob''b''гb''b' ↩
                                        'ob''.
setp gm.0.stepgen.0.position-scale 1000 # 1000 b''кb''b''pb''b''ob''b''кb''b''ib''b''вb''/b ↩
    ''ob''b''дb''b''иb''b''нв''b''иb''b''цb''b''яb'' b''пb''b''ob''b''зb''b''иb''b''цb''b' ↩
    'ib''b''їb''
setp gm.0.stepgen.0.steplen 1000    # 1000 b''нв''b''cb'' = 1 b''мb''b''кb''b''cb''
setp gm.0.stepgen.0.stepspace1000   # 1000 b''нв''b''cb'' = 1 b''мb''b''кb''b''cb''
setp gm.0.stepgen.0.dirdelay 2000   # 2000 b''нв''b''cb'' = 2 b''мb''b''кb''b''cb''
```

Підключіть StepGen до опорного положення осі 0 та увімкніть контакти

```
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 Сигнали ввімкнення та несправності

Плата керування рухом GM6-PCI має один вихід увімкнення та один вхід несправності HAL-контакти, обидва підключені до кожного роз’єму осі RJ50 та до роз’єму CAN.

Виводи HAL оновлюються функцією:

```
gm.<card_no>.read
```

Table 6.11: Контакти сигналів увімкнення та несправності

Піни	Тип і напрямок	Опис піна
gm.<card_no>.power-enable	(біт, B)	Якщо цей контакт має значення True, * і час дії таймера Watch Dog не минув * і немає збою живлення тоді контакти ввімкнення живлення роз’ємів Axis та CAN встановлені на високий рівень, інакше – на низький.
gm.<card_no>.power-fault	(біт, вихід)	Вхідний сигнал несправності живлення.

6.3.2.5 Вісь DAC

Карта управління рухом GM6-PCI має шість модулів драйвера DAC послідовної осі, по одному для кожного з'єднання. Кожен модуль підключений до контакту відповідного роз'єму осі RJ50. Кожен контакт DAC осі та назва параметра починаються наступним чином:

```
gm.<card_no>.dac.<axis_no>
```

де *<номер_осі>* має значення від 0 до 5. Наприклад, `gm.0.dac.0.value` посилається на вихідну напругу модуля DAC осі 0.

Виводи HAL оновлюються функцією:

```
gm.<card_no>.write
```

Table 6.12: Виводи DAC Axis

Піни	Тип і напрямок	Опис піна
`.enable`	(біт, B)	Увімкнути вихід DAC. Коли увімкнено значення «хибно», вихід DAC дорівнює 0,0 В.
`.value`	(float, In)	Значення вихідної напруги DAC у вольтах.

Table 6.13: Параметри DAC Axis

Параметри	Тип і напрямок	Опис параметра
`.offset`	(float, R/W)	Зсув додається до значення перед оновленням обладнання.
`.high-limit`	(float, R/W)	Максимальна вихідна напруга обладнання у вольтах.
`.low-limit`	(float, R/W)	Мінімальна вихідна напруга обладнання у вольтах.
`.invert-serial`	(float, R/W)	Карта GM6-PCI зв'язується з апаратним забезпеченням DAC за допомогою швидкої послідовної передачі даних, що дозволяє значно скоротити затримку в порівнянні з PWM. Модуль DAC рекомендується ізолювати, що виключає послідовну лінію зв'язку. У разі ізоляції залиште цей параметр за замовчуванням (0), а в разі відсутності ізоляції встановіть для цього параметра значення 1.

6.3.3 Сервопідсилювачі CAN-шини

Карта управління рухом GM6-PCI має модуль CAN для керування сервопідсилювачами CAN. Впровадження протоколів вищого рівня, таких як CANopen, є подальшим розвитком. Наразі підсилювачі потужності, що виробляються GM, мають драйвер вищого рівня, який експортує контакти та параметри до HAL. Вони отримують опорні значення положення та надають зворотний зв'язок від енкодера через шину CAN.

Кадри є стандартними (11-бітними) ідентифікаційними кадрами з довжиною даних 4 байти. Швидкість передачі даних становить 1 Мбіт/с. Ідентифікатори команд положення для осей 0..5 мають значення 0x10..0x15. Ідентифікатори зворотного зв'язку положення для осей 0..5 мають значення 0x20..0x25.

Ці конфігурації можна змінити шляхом модифікації hal_gm.c та перекомпіляції LinuxCNC.

Кожна назва виводу та параметра CAN починається так:

```
gm.<card_no>.can-gm.<axis_no>
```

де *<номер_осі>* має значення від 0 до 5. Наприклад, `gm.0.can-gm.0.position` посилається на вихідне положення осі 0 в одиницях вимірювання положення.

Виводи HAL оновлюються функцією:

```
gm.<card_no>.write
```

6.3.3.1 Піни

Table 6.14: Контакти CAN-модуля

Піни	**Тип і напрямок**	**Опис піна**
`.enable`	(біт, B)	Увімкнути надсилання довідок про позиції.
`.position-cmd`	(float, In)	Командна позиція в позиційних підрозділах.
`.position-fb`	(float, In)	Зворотній зв'язок щодо положення в одиницях позиції.

6.3.3.2 Параметри

Table 6.15: Параметри CAN-модуля

Параметри	**Тип і напрямок**	**Опис параметра**
`.position-scale`	(float, R/W)	Масштаб на одиницю довжини.

6.3.4 Сторожовий таймер

Сторожовий таймер скидається при виконанні функції:

```
gm.<card_no>.read
```

6.3.4.1 Піни

Table 6.16: Піни сторожового таймера

Піни	Тип і напрямок	Опис піна
gm.<*card_no*>.watchdog-expired	(bit, R/W)	Вказує на те, що час дії сторожового таймера минув.

Перевищення сторожового таймера призводить до низького рівня живлення апаратного забезпечення

6.3.4.2 Параметри

Table 6.17: Параметри сторожового таймера

Параметри	Тип і напрямок	Опис параметра
gm.<*card_no*>.watchdog-enable	(bit, R/W)	Вмикає сторожовий таймер. Настійно рекомендується вмикати сторожовий таймер, оскільки він може вимкнути всі сервопідсилювачі, знявши всі сигнали ввімкнення у разі помилки ПК.
gm.<*card_no*>.watchdog-timeout	(float, R/W)	Часовий інтервал у функції gm.<*card_no*>.read має бути виконаний. Функція gm.<*card_no*>.read зазвичай додається до servo-thread, тому тайм-аут спостереження зазвичай встановлюється на 3 періоди серво.

6.3.5 Кінцеві, вихідні та аварійні вимикачі

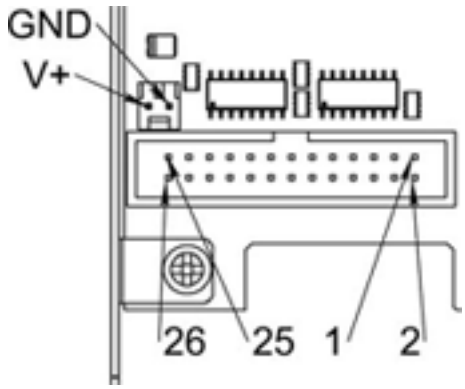


Figure 6.7: Нумерація контактів роз’єму перемикача самонаведення та кінцевого вимикача

Table 6.18: Розпіновка роз’ємів кінцевого та початкового перемикачів

25	23	21	19	17	15	13	11	9	7	5	3	1
GND		1/End-	2/End+	2/Homing	3/End-	4/End+	4/Homing	5/End-	6/End+	6/Homing	E-Стій 2	V+ (Ext.)

26	24	22	20	18	16	14	12	10	8	6	4	2
GND		1/End+	1/Homing	2/End-	3/End+	3/Homing	4/End-	5/End+	5/Homing	6/End-	E-Стій 1	V+ (Ext.)

Плата керування рухом GM6-PCI має два вхідні вимикачі кінцевого положення та один вхідний вимикач для кожного шарніра. Усі назви цих контактів починаються так:

```
gm.<card_no>.joint.<axis_no>
```

де *<axis_no>* має значення від 0 до 5. Наприклад, `gm.0.joint.0.home-sw-in` вказує на стан перемикача осі 0 ”home”.

Виводи HAL оновлюються функцією:

```
gm.<card_no>.read
```

6.3.5.1 Піни

Table 6.20: Штифти кінцевого та початкового перемикачів

Піни	Тип і напрямок	Опис піна
`.home-sw-in`	(біт, вихід)	Вхід для домашнього перемикача
`.home-sw-in-not`	(біт, вихід)	Вхідний сигнал відключення домашнього перемикача
`.neg-lim-sw-in`	(біт, вихід)	Вхід негативного кінцевого вимикача
`.neg-lim-sw-in-not`	(біт, вихід)	Вхідний сигнал негативного кінцевого вимикача
`.pos-lim-sw-in`	(біт, вихід)	Вхід позитивного кінцевого вимикача
`.pos-lim-sw-in-not`	(біт, вихід)	Вхідний сигнал кінцевого вимикача з негативним позитивним значенням

6.3.5.2 Параметри

Table 6.21: Параметри аварійного вимикача

Параметри	**Тип і напрямок**	**Опис параметра**
`gm.0.estop.0.in`	(біт, вихід)	Вхід Stop 0
`gm.0.estop.0.in-not`	(біт, вихід)	Вхід Estop 0 з запереченням
`gm.0.estop.1.in`	(біт, вихід)	Вхід Stop 1
`gm.0.estop.1.in-not`	(біт, вихід)	Вхід Estop 1 з негативним сигналом

6.3.6 Світлодіоди стану

6.3.6.1 CAN

Колір: Помаранчевий

- Блимає під час передачі даних.
- Увімкнено, коли будь-який з буферів заповнений – помилка зв’язку.
- Вимкнено, коли немає передачі даних.

6.3.6.2 RS485

Колір: Помаранчевий

- Блимає під час ініціалізації модулів на шині
- Увімкнено, коли встановлено обмін даними між усіма ініціалізованими модулями.
- Вимкнено, коли будь-який з ініціалізованих модулів вибув через помилку.

6.3.6.3 EMC

Колір: Білий

- Блимає, коли LinuxCNC працює.
- Інакше вимкнено.

6.3.6.4 Boot

Колір: Зелений

- Увімкнено, коли система успішно завантажилася.
- Інакше вимкнено.

6.3.6.5 Error

Колір: Червоний

- Вимкнено, коли в системі немає несправностей.
- Блимає, коли виникає помилка зв’язку PCI.
- Увімкнено, коли сторожовий таймер переповнений.

6.3.7 Модулі розширення вводу/виводу RS485

Ці модулі були розроблені для розширення можливостей вводу/виводу та функцій по лінії RS485 плати керування рухом GM6-PCI.

Доступні типи модулів:

- 8-канальний релейний вихідний модуль — забезпечує вісім релейних виходів NO-NC на триполюсном клемному роз’ємі для кожного каналу.
- 8-канальний модуль цифрового входу - має вісім оптично ізольованих цифрових вхідних контактів.
- 8-канальний модуль ADC та 4-канальний модуль DAC – має чотири виходи цифро-аналогового перетворювача та вісім аналого-цифрових входів. Цей модуль також оптично ізольований від карти GM6-PCI.

Автоматичне розпізнавання вузлів Кожен вузол, підключений до шини, розпізнався платою GM6-PCI автоматично. Під час запуску LinuxCNC драйвер автоматично експортує піни та параметри всіх доступних модулів.

Обробка несправностей Якщо модуль не відповідає регулярно, карта GM6-PCI відключає модуль. Якщо модуль з виходом не отримує дані з правильним CRC регулярно, модуль переходить у стан помилки (зелений світлодіод блимає) і переводить всі виходи у стан помилки.

З’єднання вузлів Модулі на шині повинні бути з’єднані послідовно з топологією, з кінцевими резисторами на кінці. Початком топології є PCI-карта, а кінцем – останній модуль.

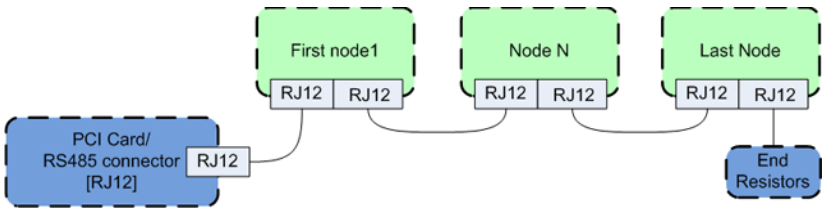


Figure 6.8: Підключення вузлів RS485 до карти GM6-PCI

Адресація Кожен вузол на шині має унікальну 4-бітну адресу, яку можна встановити за допомогою червоного DIP-перемикача.

Статус LED Зелений світлодіод показує стан модуля:

- Блимає, коли модуль лише живиться, але не ідентифікований, або коли модуль опущено.
- Вимкнено, під час ідентифікації (комп’ютер увімкнено, але LinuxCNC не запущено)
- Увімкнено, коли зв’язок здійснюється безперервно.

6.3.7.1 Модуль релейних виходів

Інформацію про роз’єм, підключення та електричні характеристики модуля див. у посібнику з інтеграції системи «https://www.generalmechatronics.com/data/products/robot_controller/PCI_UserManu

Усі виводи та параметри оновлюються наступною функцією:

```
gm.<card_no>.rs485
```

Його слід додати до потоку сервоприводу або іншого потоку з більшим періодом, щоб уникнути перевантаження процесора. Кожна назва виводу та параметра модуля RS485 починається так:

```
gm.<card_no>.rs485.<module ID>
```

де *<ідентифікатор модуля>* має значення від 00 до 15.

Table 6.22: Контакти модуля релейних виходів

Піни	Тип і напрямок	Опис піна
.relay-<0-7>	(біт, вихід)	Вихідний контакт для реле

Table 6.23: Параметри модуля релейних виходів

Параметри	Тип і напрямок	Опис параметра
.invert-relay-<0-7>	(біт, R/W)	Вихідний контакт реле заперечує

HAL приклад

```
  gm.0.rs485.0.relay-0   # b''Пb''b''eb''b''pb''b''шb''b''eb'' b''pb''b''eb''b''лb''b''eb'' ↩
      b''вb''b''yb''b''зb''b''лb''b''ab''.
# gm.0                   # b''Ib''b''дb''b''eb''b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b ↩
    ''yb''b''єb'' b''пb''b''eb''b''pb''b''шb''b''yb'' b''кb''b''ab''b''pb''b''тb''b''yb'' b' ↩
    'yb''b''пb''b''pb''b''ab''b''вb''b''лb''b''ib''b''нb''b''нb''b''яb'' b''pb''b''yb''b' ↩
    'xb''b''ob''b''мb'' GM6-PCI (b''ab''b''дb''b''pb''b''eb''b''cb''b''ab'' PCI-b''кb''b' ↩
    'ab''b''pb''b''тb''b''иb'' = 0)
#     .rs485.0           # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''вb''b''yb'' ↩
    b''зb''b''ob''b''лb'' b''зb'' b''ab''b''дb''b''pb''b''eb''b''cb''b''ob''b''юb'' 0 b' ↩
    'нb''b''ab'' b''шb''b''иb''b''нb''b''ib'' RS485
#             .relay-0   # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''пb''b''eb'' ↩
    b''pb''b''шb''b''eb'' b''pb''b''eb''b''лb''b''eb''
```

6.3.7.2 Модуль цифрового входу

Інформацію про роз’єм, підключення та електричні характеристики модуля див. у посібнику з інтеграції системи «https://www.generalmechatronics.com/data/products/robot_controller/PCI_UserManu

Усі виводи та параметри оновлюються наступною функцією:

```
gm.<card_no>.rs485
```

Його слід додати до потоку сервоприводу або іншого потоку з більшим періодом, щоб уникнути перевантаження процесора. Кожна назва виводу та параметра модуля RS485 починається так:

```
gm.<card_no>.rs485.<module ID>
```

де *<ідентифікатор модуля>* має значення від 00 до 15.

Table 6.24: Контакти модуля цифрового входу та виходу

Піни	**Тип і напрямок**	**Опис піна**
.in-*<0-7>*	(біт, вихід)	Вхідний контакт
.in-not-*<0-7>*	(біт, вихід)	Негативний вхідний контакт

HAL приклад

```
  gm.0.rs485.0.in-0   # b''Пb''b''eb''b''pb''b''шb''b''иb''b''йb'' b''вb''b''xb''b''ib''b' ↩
      'дb'' b''вb''b''yb''b''зb''b''лb''b''ab''.
# gm.0                # b''Ib''b''дb''b''eb''b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b' ↩
    'yb''b''єb'' b''пb''b''eb''b''pb''b''шb''b''yb'' b''кb''b''ab''b''pb''b''тb''b''yb'' b' ↩
    'yb''b''пb''b''pb''b''ab''b''вb''b''лb''b''ib''b''нb''b''нb''b''яb'' b''pb''b''yb''b' ↩
    'xb''b''ob''b''мb'' GM6-PCI (b''ab''b''дb''b''pb''b''eb''b''cb''b''ab'' PCI-b''кb''b' ↩
    'ab''b''pb''b''тb''b''иb'' = 0)
#     .rs485.0        # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''вb''b''yb''b' ↩
    'зb''b''ob''b''лb'' b''зb'' b''ab''b''дb''b''pb''b''eb''b''cb''b''ob''b''юb'' 0 b''нb''b ↩
    ''ab'' b''шb''b''иb''b''нb''b''ib'' RS485
#             .in-0   # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''пb''b''eb''b' ↩
    'pb''b''шb''b''иb''b''йb'' b''цb''b''иb''b''фb''b''pb''b''ob''b''вb''b''иb''b''йb'' b' ↩
    'вb''b''xb''b''ib''b''дb''b''нb''b''иb''b''йb'' b''мb''b''ob''b''дb''b''yb''b''лb''b' ↩
    'ьb''
```

6.3.7.3 Модуль DAC & ADC

Інформацію про роз'єм, підключення та електричні характеристики модуля див. у посібнику з інтеграції системи «https://www.generalmechatronics.com/data/products/robot_controller/PCI_UserManu

Усі виводи та параметри оновлюються наступною функцією:

```
gm.<card_no>.rs485
```

Його слід додати до потоку сервоприводу або іншого потоку з більшим періодом, щоб уникнути перевантаження процесора. Кожна назва виводу та параметра модуля RS485 починається так:

```
gm.<card_no>.rs485.<module ID>
```

де *<ідентифікатор модуля>* має значення від 00 до 15.

Table 6.25: Виводи модулів DAC & ADC

Піни	**Тип і напрямок**	**Опис піна**
.adc-*<0-7>*	(float, Вихід)	Значення вхідної напруги ADC у вольтах.

Table 6.25: (continued)

Піни	**Тип і напрямок**	**Опис піна**
.dac-enable-<*0-3*>	(біт, B)	Увімкнути вихід DAC. Коли значення «увімкнути» має значення «хибно», вихід DAC встановлюється на 0,0 B.
.dac-<*0-3*>	(float, In)	Значення вихідної напруги DAC у вольтах.

Table 6.26: Параметри модулів DAC & ADC

Параметри	**Тип і напрямок**	**Опис параметра**
.adc-scale-<*0-7*>	(float, R/W)	Вхідна напруга буде помножена на масштаб, перш ніж буде виведена на контакт .adc-.
.adc-offset-<*0-7*>	(float, R/W)	Зсув віднімається від вхідної напруги обладнання після застосування множника масштабу.
.dac-offset-<*0-3*>	(float, R/W)	Зсув додається до значення перед оновленням обладнання.
.dac-high-limit-<*0-3*>	(float, R/W)	Максимальна вихідна напруга обладнання у вольтах.
.dac-low-limit-<*0-3*>	(float, R/W)	Мінімальна вихідна напруга обладнання у вольтах.

HAL приклад

```
  gm.0.rs485.0.adc-0   # b''Пb''b''eb''b''pb''b''шb''b''иb''b''йb'' b''ab''b''нb''b''ab''b' ↩
      'лb''b''ob''b''гb''b''ob''b''вb''b''иb''b''йb'' b''кb''b''ab''b''нb''b''ab''b''лb'' b' ↩
      'вb''b''yb''b''зb''b''лb''b''ab''.
# gm.0                # b''Ib''b''дb''b''eb''b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b' ↩
    'yb''b''єb'' b''пb''b''eb''b''pb''b''шb''b''yb'' b''кb''b''ab''b''pb''b''тb''b''yb'' b' ↩
    'yb''b''пb''b''pb''b''ab''b''вb''b''лb''b''ib''b''нb''b''нb''b''яb'' b''pb''b''yb''b' ↩
    'xb''b''ob''b''мb'' GM6-PCI (b''ab''b''дb''b''pb''b''eb''b''cb''b''ab'' PCI-b''кb''b' ↩
    'ab''b''pb''b''тb''b''иb'' = 0)
#     .rs485.0         # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''вb''b''yb''b' ↩
    'зb''b''ob''b''лb'' b''зb'' b''ab''b''дb''b''pb''b''eb''b''cb''b''ob''b''юb'' 0 b''нb''b ↩
    ''ab'' b''шb''b''иb''b''нb''b''ib'' RS485
#             .adc-0   # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''пb''b''eb''b' ↩
    'pb''b''шb''b''иb''b''йb'' b''ab''b''нb''b''ab''b''лb''b''ob''b''гb''b''ob''b''вb''b' ↩
    'иb''b''йb'' b''вb''b''xb''b''ib''b''дb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''яb''
```

6.3.7.4 Модуль підвіски для навчання

Інформацію про роз’єм, підключення та електричні характеристики модуля див. у посібнику з інтеграції системи «https://www.generalmechatronics.com/data/products/robot_controller/PCI_UserManu

Усі виводи та параметри оновлюються наступною функцією:

```
gm.<card_no>.rs485
```

Його слід додати до потоку сервоприводу або іншого потоку з більшим періодом, щоб уникнути перевантаження процесора. Кожна назва виводу та параметра модуля RS485 починається так:

```
gm.<card_no>.rs485.<module ID>
```

де *<ідентифікатор модуля>* становить від 00 до 15. Зверніть увагу, що на модулі Teach Pendant його не можна змінити, і він попередньо запрограмований як нуль. За запитом його можна поставити з попередньо запрограмованим іншим ідентифікатором.

Table 6.27: Штифти модуля Teach Pendant

Піни	**Тип і напрямок**	**Опис піна**
.adc-*<0-5>*	(float, Вихід)	Значення вхідної напруги ADC у вольтах.
.enc-reset	(біт, В)	Якщо значення True, скидає лічильники та позицію до нуля.
.enc-counts	(s32, Вихід)	Позиція в лічильниках кодера.
.enc-rawcounts	(s32, Вихід)	Необроблений підрахунок – це підрахунки, на які не впливає скидання.
.enc-position	(float, Вихід)	Позиція в масштабованих одиницях (=.enc-counts/.enc-position-scale).
.in-*<0-7>*	(біт, вихід)	Вхідний контакт
.in-not-*<0-7>*	(біт, вихід)	Негативний вхідний контакт

Table 6.28: Параметри модуля підвіски для навчання

Параметри	**Тип і напрямок**	**Опис параметра**
.adc-scale-*<0-5>*	(float, R/W)	Вхідна напруга буде помножена на масштаб, перш ніж буде виведена на контакт .adc-.
.adc-offset-*<0-5>*	(float, R/W)	Зсув віднімається від вхідної напруги обладнання після застосування множника масштабу.
.enc-position-scale	(float, R/W)	Масштаб на одиницю довжини.

HAL приклад

```
  gm.0.rs485.0.adc-0  # b''Пb''b''eb''b''pb''b''шb''b''иb''b''йb'' b''ab''b''нb''b''ab''b' ↩
      'лb''b''ob''b''гb''b''ob''b''вb''b''иb''b''йb'' b''кb''b''ab''b''нb''b''ab''b''лb'' b' ↩
      'вb''b''уb''b''зb''b''лb''b''ab''.
# gm.0                # b''Ib''b''дb''b''eb''b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b' ↩
    'уb''b''єb'' b''пb''b''eb''b''pb''b''шb''b''уb'' b''кb''b''ab''b''pb''b''тb''b''уb'' b' ↩
    'уb''b''пb''b''pb''b''ab''b''вb''b''лb''b''ib''b''нb''b''нb''b''яb'' b''pb''b''уb''b' ↩
    'хb''b''ob''b''мb'' GM6-PCI (b''ab''b''дb''b''pb''b''eb''b''cb''b''ab'' PCI-b''кb''b' ↩
    'ab''b''pb''b''тb''b''иb'' = 0)
#     .rs485.0        # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''вb''b''уb''b' ↩
    'зb''b''ob''b''лb'' b''зb'' b''ab''b''дb''b''pb''b''eb''b''cb''b''ob''b''юb'' 0 b''нb''b ↩
    ''ab'' b''шb''b''иb''b''нb''b''ib'' RS485
#             .adc-0  # b''Bb''b''иb''b''бb''b''иb''b''pb''b''ab''b''єb'' b''пb''b''eb''b' ↩
    'pb''b''шb''b''иb''b''йb'' b''ab''b''нb''b''ab''b''лb''b''ob''b''гb''b''ob''b''вb''b' ↩
    'иb''b''йb'' b''вb''b''хb''b''ib''b''дb'' b''мb''b''ob''b''дb''b''уb''b''лb''b''яb''
```

6.3.8 Виправлення

6.3.8.1 Виправлення щодо карти GM6-PCI

Номер версії в цьому розділі стосується версії пристрою плати GM6-PCI.

Версія 1.2

- Помилка: Карта PCI не завантажується, коли перемикач Axis 1. END B активний (низький рівень). Виявлено 16 листопада 2013 року.
- Причина: Цей перемикач підключено до виводу налаштування завантаження FPGA
- Виправлення/спостереження за проблемою: Використовуйте інший контакт перемикача або підключіть до цього вхідного контакту перемикача лише нормально розімкнутий перемикач.

6.4 GS2 VFD Драйвер

Це програма HAL, яка не працює в реальному часі, для частотних перетворювачів серії GS2 від Automation Direct. Примітка: [У Європі аналогічну програму можна знайти під брендом Omron.]

Цей компонент завантажується за допомогою команди halcmd "loadusr":

```
loadusr -Wn spindle-vfd gs2_vfd -n spindle-vfd
```

Наведена вище команда говорить: loadusr, wait for named to load, component gs2_vfd, named spindle-vfd. Команда HAL `loadusr` описана в розділі loadusr.

6.4.1 Параметри командного рядка

- *-b* або *--bits* <n> (за замовчуванням: 8) Встановлює кількість бітів даних на *n*, де *n* має бути від 5 до 8 включно.
- *-d* або *--device <шлях>* (за замовчуванням: /dev/ttyS0) Встановлює шлях до файлу вузла послідовного пристрою, який потрібно використовувати.
- *-g* або *--debug* Увімкнути повідомлення налагодження. Це також встановить прапорець докладного відображення. Режим налагодження призведе до друку всіх повідомлень modbus на терміналі у шістнадцятковому форматі.
- *-n* або *--name <рядок>* (за замовчуванням: gs2_vfd) Встановлює назву модуля HAL. Назва HAL-компонента буде встановлена на <рядок>, а всі назви виводів та параметрів починатимуться з <рядка>.
- *-p* або *--parity {even,odd,none}* (за замовчуванням: odd) Встановлює парність послідовного інтерфейсу на парне, непарне або жодне.
- *-r* або *--rate* <n> (за замовчуванням: 38400) Встановлює швидкість передачі даних на *n*. Це вважається помилкою, якщо швидкість не дорівнює одному з наступних значень: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- *-s* або *--stopbits {1,2}* (за замовчуванням: 1) Встановлює стоп-біти послідовного інтерфейсу на 1 або 2
- *-t* або *--target* <n> (за замовчуванням: 1) Встановіть номер цільового пристрою MODBUS (підлеглого). Він має збігатися з номером пристрою, встановленим на GS2.
- *-v* або *--verbose* Увімкнути повідомлення налагодження.

- *-A* або *--accel-seconds <n>* (за замовчуванням: 10.0) Секунди для розгону шпинделя від 0 до максимальних обертів за хвилину.
- *-D* або *--decel-seconds <n>* (за замовчуванням: 0.0) Секунди для уповільнення шпинделя від макс. обертів за хвилину до 0. Якщо встановлено значення 0.0, шпинделю буде дозволено обертатися по інерції до повної зупинки без контрольованого уповільнення.
- *-R* або *--braking-resistor* Цей аргумент слід використовувати, коли на GS2 VFD встановлено гальмівний резистор (див. Додаток A до посібника GS2). Він вимикає запобігання зупинці через перенапругу при гальмуванні (див. параметр GS2 modbus 6.05), дозволяючи VFD продовжувати гальмування навіть у ситуаціях, коли двигун регенерує високу напругу. Регенерована напруга безпечно скидається на гальмівний резистор.

Note
Що якщо є помилки конфігурації послідовного порту, увімкнення детального режиму може призвести до потоку помилок тайм-ауту.

6.4.2 Піни

Де *<назва>* має значення ”gs2_vfd” або ім’я, задане під час завантаження з опцією `-n`:

- <назва>*.DC-bus-volts* (число з плаваючою комою, вихід) Напруга шини постійного струму частотного перетворювача
- <назва>*.at-speed* (біт, вихід), коли привід працює на заданій швидкості
- <назва>*.err-reset* (біт, вхід) помилки скидання, що надсилаються до частотно-регульованого приводу
- <назва>*.версія-прошивки* (s32, вихід) з частотно-регульованого приводу
- <назва>*.frequency-command* (число з плаваючою комою, вихід) з частотно-регульованого перетворю
- <назва>*.frequency-out* (число з плаваючою комою, вихід) з частотно-регульованого перетворювача
- <назва>*.зупинено* (біт, вихід), коли частотний перетворювач повідомляє про вихід 0 Гц
- <назва>*.load-percentage* (число з плаваючою комою, вихід) з частотно-регульованого перетворювача
- <назва>*.об/хв двигуна* (число з плаваючою комою, вихід) з частотного перетворювача
- <назва>*.output-current* (число з плаваючою комою, вихід) з частотно-регульованого перетворювача (ЧРП)
- <назва>*.output-voltage* (число з плаваючою комою, вихід) з частотного перетворювача
- <назва>*.фактор-потужності* (число з плаваючою комою, вихід) з частотного перетворювача
- <назва>*.scale-frequency* (число з плаваючою комою, вихід) з частотно-регульованого перетворювача (ЧП)
- <назва>*.command-speed* (float, in) швидкість, що надсилається до частотного перетворювача в об/хв. Надсилання швидкості, що перевищує максимальні об/хв двигуна, встановлені в частотному перетворювачі, є помилкою.
- <назва>*.spindle-fwd* (біт, вхід) 1 для FWD та 0 для REV, що надсилаються до частотного перетворювач
- <назва>*.spindle-rev* (біт, вхід) 1 для REV та 0, якщо вимкнено
- <назва>*.spindle-on* (біт, вхід) 1 для УВІМК. та 0 для ВИМК. надсилається до частотного перетворювач

- <назва>.*status-1* (s32, вихід) Стан приводу частотного перетворювача (див. посібник GS2)
- <назва>.*status-2* (s32, вихід) Стан приводу частотного перетворювача (див. посібник GS2)

Note
Значення стану – це сума всіх активних бітів. Отже, число 163, яке означає, що привід перебуває в режимі роботи, дорівнює сумі 3 (робота) + 32 (частота, встановлена послідовним портом) + 128 (операція, встановлена послідовним портом).

6.4.3 Параметри

Де *<назва>* буде gs2_vfd або назвою, заданою під час завантаження з опцією `-n`:

- <name>.*error-count* (s32, RW)
- <назва>.*loop-time* (float, RW) частота опитування Modbus (за замовчуванням: 0.1)
- <назва>.*nameplate-HZ* (число з плаваючою комою, RW) Частота двигуна в Гц (за замовчуванням: 60)
- <назва>.*nameplate-RPM* (число з плаваючою комою, RW) Обороти двигуна на паспортній табличці (за замовчуванням: 1730)
- <ім'я>.*retval* (s32, RW) повернене значення помилки в HAL
- <назва>.*tolerance* (s32, RW) допуск швидкості (за замовчуванням: 0,01)
- <назва>.*ack-delay* (s32, RW) кількість циклів читання/запису перед перевіркою на швидкості (за замовчуванням 2)

Приклад використання цього компонента для керування шпинделем див. у прикладі GS2 Spindle.

6.5 Драйвер HAL для контактів GPIO Raspberry Pi

Примітка: Цей драйвер не буде компілюватися в образи, призначені для процесорів, відмінних від ARM. Він дійсно призначений для роботи лише на Raspberry Pi. Він може працювати, а може й ні, на подібних платах або прямих клонах.

6.5.1 Мета

Цей драйвер дозволяє використовувати виводи GPIO Rapberry Pi аналогічно драйверу паралельного порту на ПК x86. Він може використовувати ті ж генератори кроків, лічильники енкодерів та подібні компоненти.

6.5.2 Застосування

```
loadrt hal_pi_gpio dir=0x13407 exclude=0x1F64BF8
```

Маска «dir» визначає, чи є контакти входами та виходами, маска виключення запобігає використанню контактів драйвером (і таким чином дозволяє використовувати їх для звичайних цілей RPi, таких як SPI або UART).

Маска може бути у десятковому або шістнадцятковому форматі (шістнадцятковий може бути простішим, оскільки не буде перенесення).

Щоб визначити значення масок, додайте шестнадцяткові/десятичні значення для всіх виводів, які повинні бути налаштовані як виходи, і аналогічно для всіх виводів, які повинні бути виключені відповідно до наведеної нижче таблиці.

Table 6.29: Маски GPIO — відповідність номерів GPIO (крайній лівий стовпець) фізичним номерам контактів, надрукованим на платі Raspberry Pi (крайній правий стовпець), та десятковим/шістнадцятковим значенням, що складають значення маски.

Номер GPIO	Десятковий	Шістнадцятковий	PIN-код
2	1	0x00000001	3
3	2	0x00000002	5
4	4	0x00000004	7
5	8	0x00000008	29
6	16	0x00000010	31
7	32	0x00000020	26
8	64	0x00000040	24
9	128	0x00000080	21
10	256	0x00000100	19
11	512	0x00000200	23
12	1024	0x00000400	32
13	2048	0x00000800	33
14	4096	0x00001000	8
15	8192	0x00002000	10
16	16384	0x00004000	36
17	32768	0x00008000	11
18	65536	0x00010000	12
19	131072	0x00020000	35
20	262144	0x00040000	38
21	524288	0x00080000	40
22	1048576	0x00100000	15
23	2097152	0x00200000	16
24	4194304	0x00400000	18
25	8388608	0x00800000	22
26	16777216	0x01000000	37
27	33554432	0x02000000	13

Примітка: При обчисленні значення маски окремого контакту використовуються його номери GPIO, значення обчислюється як 2^(*номер GPIO* - 2), тоді як при найменуванні контактів HAL використовуються номери контактів роз'єму Raspberry Pi.

Наприклад, якщо ви ввімкнете GPIO 17 як вихід (dir=0x8000), то цей вихід буде керуватися виводом hal **hal_pi_gpio.pin-11-out**.

6.5.3 Піни

- hal_pi_gpio.pin-*NN*-out

- hal_pi_gpio.pin-*NN*-in

Залежно від напрямку та виключити маски.

6.5.4 Параметри

Існують лише стандартні параметри синхронізації, створені для всіх компонентів:

- hal_pi_gpio.read.tmax
- hal_pi_gpio.read.tmax-increased
- hal_pi_gpio.write.tmax
- hal_pi_gpio.write.tmax-increased

З невідомих причин драйвер також створює HAL-виводи для індикації часу:

- hal_pi_gpio.read.time
- hal_pi_gpio.write.time

6.5.5 Функції

- `hal_pi_gpio.read` - Додайте це до базового потоку, щоб оновити значення виводів HAL відповідно до значень фізичного входу.
- `hal_pi_gpio.write` - Додайте це до базового потоку, щоб оновити фізичні контакти відповідно до значень HAL.

Зазвичай функція «read» буде на початку списку викликів, перед будь-якими лічильниками енкодера, а функція «write» буде пізніше у списку викликів, після stepgen.make-pulses.

6.5.6 Нумерація контактів

Роз’єм GPIO та розкладка виводів залишаються незмінними з 2015 року. Ці старі моделі Pi, ймовірно, є невдалим вибором для LinuxCNC. Однак цей драйвер розроблений для роботи з ними і буде виявляти та правильно налаштовувати два альтернативні варіанти розкладки виводів.

Поточне розташування контактів між номерами GPIO та номерами контактів роз’єму наведено у таблиці вище.

Зверніть увагу, що рядок конфігурації використовує номери GPIO, але після завантаження драйвера назви контактів HAL відповідають номерам контактів роз’єму.

Це може бути більш логічним, ніж здається на перший погляд. Під час налаштування потрібно сконфігурувати достатню кількість контактів кожного типу, уникаючи перезапису будь-яких інших функцій, необхідних вашій системі. Потім, після завантаження драйвера, у шарі HAL вам потрібно лише знати, куди підключити дроти для кожного контакту HAL.

6.5.7 Відомі помилки

На даний момент (16 липня 2023 р.) цей драйвер, здається, працює лише на Raspbian, оскільки загальний образ Debian не налаштовує правильні інтерфейси в /dev/gpiomem та обмежує доступ до інтерфейсу /sys/mem.

6.6 Загальний драйвер для будь-якого GPIO, що підтримується gpiod.

Цей драйвер було протестовано на Raspberry Pi, і він також має працювати на Banana Pi, Beagle-Bone, Pine64 (та ін.) та інших одноплатних комп'ютерах, а також потенційно на інших платформах.

6.6.1 Мета

Цей драйвер дозволяє використовувати виводи GPIO аналогічно драйверу паралельного порту на ПК x86. Він може використовувати ті ж генератори кроків, лічильники енкодерів та подібні компоненти.

6.6.2 Застосування

```
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
```

Цей драйвер використовує бібліотеку libgpiod-dev та пакет gpiod, який містить низку утиліт для налаштування та запиту GPIO. Назви виводів GPIO у рядку «loadrt» HAL, наведеному вище, повинні відповідати назвам, наданим командою `gpioinfo`.

Зразок виводу (скорочений):

```
$ 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
...
```

Список вхідних та/або вихідних контактів слід вказати, як показано у прикладі вище. Символ \ використовується для продовження рядка в HAL і служить для поліпшення читабельності. Імена контактів чутливі до регістру, і в рядках не повинно бути пробілів, ні між списками контактів, розділених комами, ні між знаками «=».

Додаткові модифікатори є

інвертувати
(дійсно лише для виходів). Інвертує значення фізичного виводу відносно значення в HAL.

скинути
(дійсне тільки для виходів). Якщо будь-які контакти виділені в список «reset», то буде створено параметр HAL **hal_gpio.reset_ns**. Це не матиме ніякого ефекту, якщо функція **hal_gpio.reset** не буде додана до потоку реального часу. Вона повинна бути розміщена після функції **hal_gpio.write** і повинна знаходитися в тому ж потоці. Поведінка цієї функції еквівалентна аналогічній функції в драйвері **hal_parport** і дозволяє виконувати імпульс кроку в кожному циклі потоку. Якщо час **hal_gpio.reset_ns** встановлено довше, ніж 1/4 періоду потоку, до якого він додається, то значення буде зменшено до 1/4 періоду потоку. Існує нижня межа тривалості імпульсу. Наприклад, при 8 контактах у списку виводів ширина імпульсу не може бути меншою за 5000 нс на RPi4.

Наступні функції підтримуються у всіх версіях, але працюють лише у випадку, якщо встановлено версію libgpiod_dev >= 1.6. Вони повинні використовуватися так само, як і параметри, описані вище, і змінюють електричні параметри контактів GPIO, **якщо** це підтримується апаратним забезпеченням.

opendrain

відкритий код

упередженийвимкнути

розкривний список

підтягування

Версію встановленої libgpiod-dev можна визначити командою `gpioinfo -v`

6.6.3 Піни

- hal_gpio.NAME-in - HAL_OUT Значення вхідного виводу, що подається до HAL
- hal_gpio.NAME-in-not - HAL_OUT Інвертована версія вищезазначеного, для зручності
- hal_gpio.NAME-out - HAL_IN використовує цей контакт для передачі значення біта HAL на фізичний вихід

6.6.4 Параметри

- hal_gpio.reset_ns - HAL_RW - ”setp” - цей параметр керує тривалістю імпульсу виводів, доданих до списку ”скидання”. Значення буде обмежене між 0 та періодом потоку / 4.

6.6.5 Функції

- *hal_gpio.read* – Додайте це до базового потоку, щоб оновити значення виводів HAL відповідно до значень фізичного входу.
- *hal_gpio.write* – Додайте це до базового потоку, щоб оновити фізичні піни відповідно до значень HAL.
- *hal_gpio.reset* – Експортується лише за умови, що у списку скидання визначено контакти. Його слід розмістити після функції ”write” та в тому ж потоці.

Зазвичай функція «read» буде розташована на початку списку викликів, перед будь-якими лічильникам енкодера, а функція «write» буде розташована пізніше у списку викликів, після `stepgen.make-pulses`.

6.6.6 Ідентифікація PIN-коду

Використовуйте імена контактів, повернуті утилітою `gpioinfo`. Вона використовує дані дерева пристроїв. Якщо встановлена ОС не має бази даних дерева пристроїв, то всі контакти будуть називатися «unnamed» (або подібним чином), і цей драйвер не можна буде використовувати.

Подальше оновлення цього драйвера може дозволити доступ за індексним номером, але наразі це не підтримується.

6.6.7 Вирішення проблем із дозволами.

Якщо під час завантаження драйвера з’являються повідомлення «доступ заборонено», спробуйте виконати наступні дії: (Для Raspbian це не повинно бути необхідним, а для платформ, відмінних від Pi, потрібно буде змінити ім’я мікросхеми GPIO відповідно до фактичного)

1. Створіть нову групу `gpio` за допомогою команди

```
sudo groupadd gpio
```

2. Потім, щоб налаштувати дозволи для групи ”gpio”, створіть файл під назвою `90-gpio-access` у каталозі `/etc/udev/rules.d/` з наступним вмістом (він скопійований з інсталяції Raspbian)

```
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. Додайте користувача, який запускає LinuxCNC, до групи `gpio` за допомогою

```
sudo usermod -aG gpio <username>
```

6.6.8 Автор

Енді П’ю

6.6.9 Відомі помилки

Наразі жодного.

6.7 Mesa Драйвер HostMot2

6.7.1 Вступ

HostMot2 — це конфігурація FPGA, розроблена компанією Mesa Electronics для своєї лінійки карт управління рухом «Anything I/O». Прошивка є відкритою, портативною та гнучкою. Її можна налаштувати (під час компіляції) з нульовою або більшою кількістю екземплярів (об'єкт, створений під час виконання) кожного з декількох модулів: енкодерів (квадратурних лічильників), генераторів PWM та генераторів кроку/напрямку. Прошивка може бути налаштована (під час виконання) для підключення кожного з цих екземплярів до контактів на роз'ємах вводу-виводу. Контакти вводу-виводу, що не керуються екземпляром модуля, повертаються до двонаправленого цифрового вводу-виводу загального призначення.

6.7.2 Бінарні файли прошивки

50-контактний роз'єм FPGA-карт Для різних плат вводу/виводу Anything доступні кілька попередньо скомпільованих бінарних файлів прошивки HostMot2. Цей список неповний, перевірте дистрибутив hostmot2-firmware для отримання актуальних списків прошивок.

- 3x20 (144 контакти вводу/виводу): використання модуля hm2_pci
  - 24-канальний сервопривід
  - 16-канальний сервопривід плюс 24 генератори кроків/напрямків
- 5I22 (96 контактів вводу/виводу): використання модуля hm2_pci
  - 16-канальний сервопривід
  - 8-канальний сервопривід плюс 24 генератори кроків/напрямків
- 5I20, 5I23, 4I65, 4I68 (72 контакти вводу/виводу): використання модуля hm2_pci
  - 12-канальний сервопривід
  - 8-канальний сервопривід плюс 4 генератори кроків/напрямків
  - 4-канальний сервопривід плюс 8 генераторів кроків/напрямків
- 7I43 (48 контактів вводу/виводу): використання модуля hm2_7i43
  - 8-канальний сервопривід (8 PWM-генераторів та 8 енкодерів)
  - 4-канальний сервопривід плюс 4 генератори кроків/напрямків

ПЛІС-карти DB25 Плата 5I25 Superport FPGA попередньо запрограмована під час придбання та не потребує двійкового файлу прошивки.

6.7.3 Встановлення прошивки

Залежно від того, як ви встановили LinuxCNC, можливо, вам доведеться відкрити Synaptic Package Manager (Менеджер пакетів Synaptic) з меню System (Система) і встановити пакет для вашої карти Mesa. Найшвидший спосіб знайти їх — це виконати пошук за «hostmot2» в Synaptic Package Manager. Позначте прошивку для встановлення, а потім застосуйте.

6.7.4 Завантаження HostMot2

Підтримка LinuxCNC для прошивки HostMot2 розділена на загальний драйвер під назвою «hostmot2» та два низькорівневі драйвери вводу-виводу для плат Anything I/O. Низькорівневі драйвери вводу-виводу — це «hm2_7i43» та «hm2_pci» (для всіх плат Anything I/O на базі PCI та PC-104/Plus). Драйвер hostmot2 необхідно завантажити першим, використовуючи таку команду HAL:

```
loadrt hostmot2
```

Дивіться сторінку довідки hostmot2(9) для отримання детальної інформації.

Драйвер hostmot2 сам по собі нічого не робить, йому потрібен доступ до реальних плат, на яких працює прошивка HostMot2. Цей доступ забезпечують драйвери низького рівня вводу-виводу. Драйвери низького рівня вводу-виводу завантажуються за допомогою таких команд:

```
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT
       num_encoders=3 num_pwmgens=3 num_stepgens=1"
```

Параметри конфігурації описані на сторінці довідки hostmot2.

6.7.5 Сторожовий пес

Прошивка HostMot2 може містити модуль сторожового таймера; якщо він є, драйвер hostmot2 використовуватиме його.

Сторожовий таймер повинен періодично пестити LinuxCNC, інакше він вкусить. Функція запису hm2 (див. нижче) пестить сторожовий таймер.

Коли сторожовий таймер спрацьовує, всі виводи вводу-виводу плати від'єднуються від своїх екземплярі модулів і стають входами з високим імпедансом (підтягнутими до високого рівня). Стан модулів прошивки HostMot2 не порушується (за винятком конфігурації виводів вводу/виводу). Інстанції енкодера продовжують підраховувати квадратурні імпульси, а генератори pwm і step продовжують генерувати сигнали (які не передаються на двигуни, оскільки виводи вводу/виводу стали входами).

Скидання сторожового таймера скидає налаштування контактів вводу/виводу до конфігурації, вибраної під час завантаження.

Якщо прошивка містить сторожовий таймер, будуть експортовані такі об'єкти HAL:

6.7.5.1 Піни

- *has_bit* - (біт вводу/виводу) True, якщо сторожовий таймер має біт, False, якщо сторожовий таймер не має біта. Якщо сторожовий таймер має біт, а біт has_bit має значення True, користувач може скинути його до False, щоб відновити роботу.

6.7.5.2 Параметри

- *timeout_ns* - (u32 читання/запис) Тайм-аут сторожового таймера, в наносекундах. Він ініціалізується до 5000000 (5 мілісекунд) під час завантаження модуля. Якщо між викликами функції запису hm2 пройде більше часу, ніж зазначено, сторожовий таймер спрацює.

6.7.6 Функції HostMot2

- *hm2_<ТипПлати>.<НомерПлати>.read* - Зчитати всі входи, оновити вхідні HAL-піни.
- *hm2_<ТипПлати>.<НомерПлати>.write* - Записати всі виходи.

- *hm2_<ТипПлати>.<НомерПлати>.read_gpio* – Зчитує лише вхідні контакти GPIO. (Ця функція недоступна на 7I43 через обмеження шини EPP.)
- *hm2_<ТипПлати>.<НомерПлати>.write_gpio* – Запис лише в регістри керування GPIO та вихідні контакти. (Ця функція недоступна на 7I43 через обмеження шини EPP.)

Note
Вищезазначені функції «read_gpio» та «write_gpio» зазвичай не потрібні, оскільки біти GPIO читаються та записуються разом з усім іншим у стандартних функціях «read» та «write», які зазвичай виконуються в серво-потоці.
Функції «read_gpio» та «write_gpio» були надані на випадок, якщо знадобиться дуже швидкий (часто оновлюваний) ввід-вивід. Ці функції слід запускати в базовому потоці. Якщо вам це потрібно, надішліть нам електронного листа та повідомте про це, а також про те, що це за програма.

6.7.7 Розпіновки

Драйвер hostmot2 не має певної розкладки виводів. Розкладка виводів визначається прошивкою, яку драйвер hostmot2 надсилає на плату Anything I/O. Кожна прошивка має різну розкладку виводів, яка залежить від кількості доступних енкодерів, pwmgens та stepgens, що використовуються. Щоб отримати список розкладки виводів для вашої конфігурації після завантаження LinuxCNC у вікні терміналу, введіть:

```
dmesg > hm2.txt
```

Отриманий текстовий файл міститиме багато інформації, а також розпіновку HostMot2 та будь-які повідомлення про помилки та попередження.

Щоб зменшити кількість зайвих повідомлень, очистивши буфер повідомлень перед завантаженням LinuxCNC, введіть наступну команду у вікні терміналу:

```
sudo dmesg -c
```

Тепер, коли ви запускаєте LinuxCNC, а потім виконуєте команду «dmesg > hm2.txt» у терміналі, у вашому файлі буде тільки інформація з моменту завантаження LinuxCNC разом із розкладкою виводів. Файл буде знаходитися в поточному каталозі вікна терміналу. Кожен рядок буде містити назву карти, номер карти, номер виводу вводу/виводу, роз’єм і вивід, а також використання. З цього роздрукованого документа ви дізнаєтеся про фізичні підключення до вашої карти на основі вашої конфігурації.

Приклад конфігурації 5I20:

```
[HOSTMOT2]
DRIVER=hm2_pci
BOARD=5i20
CONFIG=”firmware=hm2/5i20/SVST8_4.BIT num_encoders=1 num_pwmgens=1 num_stepgens=3”
```

Вищевказана конфігурація призвела до цього роздруківку.

```
[ 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
Що I/O Pin nnn буде відповідати номеру виводу, показаному на екрані конфігурації HAL для GPIO. Деякі StepGen, Encoder і PWMGen також будуть відображатися як GPIO на екрані конфігурації HAL.

6.7.8 PIN-файли

Розташування контактів за замовчуванням описано у файлі .PIN (текст, що читається людиною). Під час встановлення пакета прошивки файли .PIN встановлюються у

```
/usr/share/doc/hostmot2-firmware-<board>/
```

6.7.9 Прошивка

Вибране вбудоване програмне забезпечення (.BIT файл) та конфігурація завантажуються з материнськ плати ПК на материнську плату Mesa під час запуску LinuxCNC. Якщо ви використовуєте Run In Place, вам все одно необхідно встановити пакет hostmot2-firmware-<board>. Більше інформації про вбудоване програмне забезпечення та конфігурацію можна знайти в розділі «Конфігурації».

6.7.10 Піни HAL

Конфігурацію контактів HAL для кожної конфігурації можна переглянути, відкривши пункт «Show HAL Configuration» (Показати конфігурацію HAL) в меню «Machine» (Машина). Там можна знайти всі контакти та параметри HAL. На малюнку нижче показано конфігурацію 5I20, яка використовувалас вище.

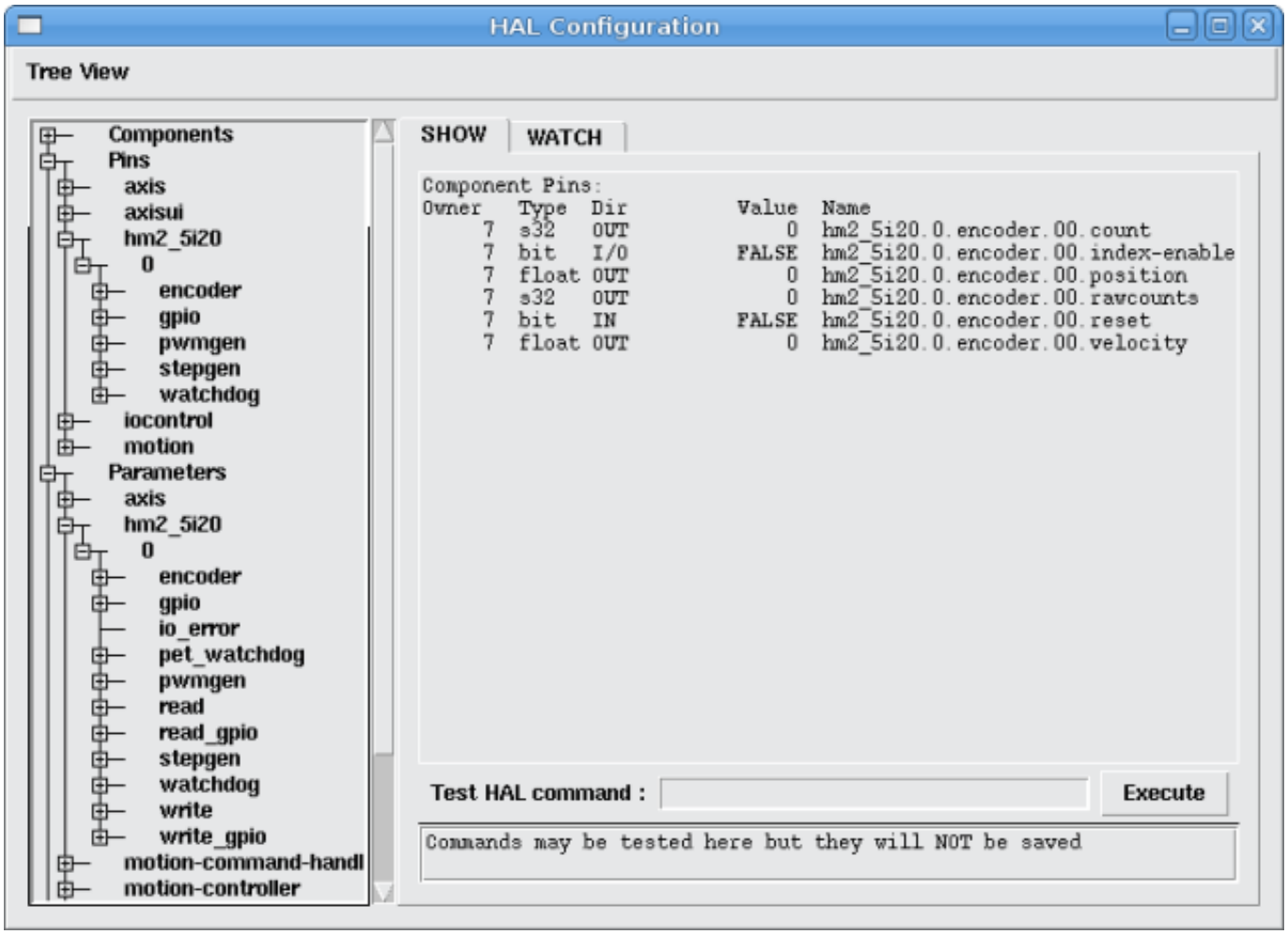


Figure 6.9: 5i20 HAL-штифти

6.7.11 Конфігурації

Прошивка Hostmot2 доступна в декількох версіях, залежно від того, що ви намагаєтеся досягти. Ви можете дізнатися, для чого призначена конкретна прошивка, подивившись на її назву. Давайте розглянемо кілька прикладів.

У 7I43 (два порти) SV8 («Servo 8») призначений для 8 сервоприводів або менше, використовуючи «класичну» 4-осьову (на порт) сервоплату 7I33. Отже, 8 сервоприводів використають усі 48 сигналів у двох портах. Але якщо вам потрібно лише 3 сервоприводи, ви можете вказати «num_encoders і «num_pwmgens=3» і відновити 5 сервоприводів по 6 сигналів кожен, отримавши таким чином 30 бітів GPIO.

Або, в 5I22 (чотири порти), SVST8_24 («Серво 8, Степпер 24») буде для 8 сервоприводів або менше (знову 7I33 x2) і 24 степперів або менше (7I47 x2). Це використає всі чотири порти. Якщо вам потрібно лише 4 сервоприводи, ви можете вказати «num_encoders=4» і «num_pwmgens=4» і звільнити 1 порт (і заощадити 7I33). А якщо вам потрібно лише 12 крокових двигунів, ви можете вказати «num_stepgens=12» і звільнити один порт (і заощадити 7I47). Таким чином, ми можемо заощадити два порти (48 бітів) для GPIO.

Ось таблиці прошивок, доступних в офіційних пакетах. На веб-сайті Mesanet.com можуть бути доступні додаткові прошивки, які ще не увійшли до офіційних пакетів прошивок LinuxCNC, тому перевірте і там.

3x20 (різні 6-портові) конфігурації за замовчуванням (3x20 постачається у версіях з вентилями 1M, 1.5M та 2M. Наразі вся прошивка доступна для вентилів усіх розмірів.)

Прошивка	Енкодер	PWMGen	StepGen	GPIO
SV24	24	24	0	0
SVST16_24	16	16	24	0

Конфігурації 5I22 (4-портовий PCI) за замовчуванням (5I22 постачається у версіях з вентилями 1M та 1.5M. Наразі вся прошивка доступна для всіх розмірів вентилів.)

Прошивка	Енкодер	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-портовий PCI) Конфігурації за замовчуванням (5I23 має 400k вентилів.)

Прошивка	Енкодер	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-портовий PCI) Конфігурації за замовчуванням (5I20 має 200k вентилів.)

Прошивка	Енкодер	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-портовий PC/104) Конфігурації за замовчуванням (4I68 має 400k вентилів.)

Прошивка	Енкодер	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-портовий PC/104) Конфігурації за замовчуванням (4I65 має 200k вентилів.)

Прошивка	Енкодер	PWM	StepGen	GPIO
SV12	12	12	0	0
SVST8_4	8	8	4	0
SVST8_4IM2	8 (+IM)	8	4	8

7I43 (2-портовий паралельний) версії з вентилем 400k, конфігурації за замовчуванням

Прошивка	Енкодер	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-портовий паралельний) версії з вентилями 200k, конфігурації за замовчуванням

Прошивка	Енкодер	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

Незважаючи на те, що кілька карт можуть мати файл .BIT з однаковою назвою, ви не можете використовувати файл .BIT, який не призначений для цієї карти. Різні карти мають різні тактові частоти, тому переконайтеся, що ви завантажуєте файл .BIT, який підходить для вашої карти. Для спеціальних застосувань можна створити власні прошивки hm2, і ви можете побачити деякі власні прошивки hm2 у каталогах із стандартними прошивками.

Коли ви завантажуєте драйвер плати (hm2_pci або hm2_7i43), ви можете вказати йому вимкнути екземпляри трьох основних модулів (pwmgen, stepgen та encoder), встановивши нижче значення лічильника. Будь-які виводи вводу-виводу, що належать до вимкнених екземплярів модулів, стають GPIO.

6.7.12 GPIO

Контакти вводу-виводу загального призначення на платі, які не використовуються екземпляром модуля, експортуються в HAL як «повні» контакти GPIO. Повні контакти GPIO можна налаштувати під час виконання як входи, виходи або відкриті стоки, і вони мають інтерфейс HAL, який забезпечує таку гнучкість. Контакти вводу-виводу, що належать активному екземпляру модуля, обмежуються вимогами модуля-власника і мають обмежений інтерфейс HAL.

GPIO мають такі імена: «hm2_<BoardType>.<BoardNum>.gpio.<IONum>». IONum — це тризначне число. Відповідність між IONum, роз'ємом і контактом на цьому роз'ємі записується в системний журнал під час завантаження драйвера і описана в посібнику Mesa для плат Anything I/O.

Представлення hm2 GPIO змодельовано за зразком цифрових входів та цифрових виходів, описаних у Canonical Device Interface (частина документа HAL General Reference).

Виводи GPIO за замовчуванням встановлені на вхід.

6.7.12.1 Піни

- *in* - (Bit, Out) Нормальний стан вхідного контакту апаратного забезпечення. Цей контакт мають як повноцінні контакти GPIO, так і контакти вводу/виводу, що використовуються як вхідні дані активними екземплярами модулів.

- *in_not* - (Біт, Вихід) Інвертований стан вхідного контакту апаратного забезпечення. Цей контакт мають як повноцінні контакти GPIO, так і контакти вводу/виводу, що використовуються як вхідні дані активними екземплярами модулів.
- *out* - (Bit, In) Значення, яке буде записано (можливо, інвертовано) на вихідний контакт апаратного забезпечення. Тільки повноцінні контакти GPIO мають цей контакт.

6.7.12.2 Параметри

- *invert_output* - (біт, RW) Цей параметр діє тільки в тому випадку, якщо параметр *is_output* має значення true. Якщо цей параметр має значення true, вихідне значення GPIO буде оберненим до значення на виводі HAL *out*. Тільки повні виводи GPIO та виводи вводу-виводу, що використовують як виходи активними екземплярами модулів, мають цей параметр. Щоб інвертувати вивід активного модуля, потрібно інвертувати вивід GPIO, а не вивід модуля.
- *is_opendrain* - (біт, RW) Цей параметр діє тільки в тому випадку, якщо параметр *is_output* має значення true. Якщо цей параметр має значення false, GPIO працює як звичайний вихідний контакт: контакт вводу-виводу на роз'ємі приводиться в стан, визначений контактом HAL *out* (можливо інвертованим), а значення контактів HAL *in* і *in_not* не визначені. Якщо цей параметр має значення true, GPIO працює як вивід з відкритим стоком. Запис 0 у вивід HAL «out» приводить вивід вводу-виводу в низький стан, запис 1 у вивід HAL «out» переводить вивід вводу-виводу у стан високого імпедансу. У цьому стані з високим імпедансом контакт вводу-виводу плаває (слабо підтягується до високого рівня), і інші пристрої можуть керувати значенням; отримане значення на контакті вводу-виводу доступне на контактах «in» і «in_not». Тільки повні контакти GPIO і контакти вводу-виводу, що використовуються як виходи активними екземплярами модулів, мають цей параметр.
- *is_output* - (біт, RW) Якщо встановлено значення 0, GPIO є входом. Контакт вводу-виводу переводиться у стан високого імпедансу (слабко підтягнутий до високого рівня), щоб керуватися іншими пристроями. Логічне значення на контакті вводу-виводу доступне на контактах HAL *in* та *in_not*. Записи в контакт HAL «out» не мають ефекту. Якщо цей параметр встановлений на 1, GPIO є виходом; його поведінка тоді залежить від параметра «is_opendrain». Тільки повні контакти GPIO мають цей параметр.

6.7.13 StepGen

StepGens мають такі імена: «hm2_<BoardType>.<BoardNum>.stepgen.<Instance>». «Instance» — це двозначне число, яке відповідає номеру екземпляра HostMot2 stepgen. Існує «num_stepgens» екземплярів, починаючи з 00.

Кожен stepgen виділяє 2-6 виводів вводу-виводу (вибираються під час компіляції прошивки), але в даний час використовує тільки два: виводи Step і Direction. Примітка: [В даний час прошивка підтримує багатофазні крокові виводи, але драйвер - ні. Зацікавлені волонтери запрошуються.]

Представлення StepGen моделюється на основі програмного компонента stepgen. За замовчуванням StepGen має активний високий вихідний сигнал (високий під час кроку, низький під час проміжку між кроками). Щоб інвертувати вихідний контакт StepGen, інвертуйте відповідний контакт GPIO, який використовується StepGen. Щоб знайти контакт GPIO, який використовується для виходу StepGen, запустіть *dmesg*, як показано вище.

Кожен екземпляр StepGen має такі контакти та параметри:

6.7.13.1 Піни

- *control-type* - (Bit, In) Перемикається між режимом керування положенням (0) та режимом керування швидкістю (1). За замовчуванням використовується керування положенням (0).

- *counts* - (s32, Out) Позиція зворотного зв’язку в лічильниках (кількість кроків).
- *enable* - (Bit, In) Вмикає вихідні кроки. Якщо значення false, кроки не генеруються.
- *position-cmd* - (Float, In) Цільова позиція руху крокового механізму, у визначених користувачем одиницях позиції.
- *position-fb* - (Float, Out) Позиція зворотного зв’язку в одиницях вимірювання позиції, визначених користувачем (counts / position_scale).
- *velocity-cmd* - (Float, In) Цільова швидкість руху крокового двигуна, у визначених користувачем одиницях позиції за секунду. Цей контакт використовується лише тоді, коли кроковий генератор перебуває в режимі керування швидкістю (control-type=1).
- *velocity-fb* - (Float, Out) Швидкість зворотного зв’язку в одиницях положення, визначених користувач за секунду.

6.7.13.2 Параметри

- *dirhold* - (u32, RW) Мінімальна тривалість стабільного сигналу напрямку після завершення кроку, у наносекундах.
- *dirsetup* - (u32, RW) Мінімальна тривалість стабільного сигналу напрямку перед початком кроку, у наносекундах.
- *maxaccel* - (Float, RW) Максимальне прискорення, в одиницях позиції за секунду за секунду. Якщо встановлено значення 0, драйвер не обмежуватиме своє прискорення.
- *maxvel* - (Float, RW) Максимальна швидкість, в одиницях положення за секунду. Якщо встановлено значення 0, драйвер вибере максимальну швидкість на основі значень steplen і stepspace (на момент, коли maxvel було встановлено на 0).
- *position-scale* - (Float, RW) Перетворює з одиниць вимірювання позиції на одиниці вимірювання позиції. position = counts / position_scale
- *step_type* - (u32, RW) Формат виводу, як modparam step_type для компонента програмного забезпечення stegen(9). 0 = Крок/Напрямок, 1 = Вгору/Вниз, 2 = Квадратура. У режимі Quadrature (step_type=2) stepgen виводить один повний цикл Грея (00 -> 01 -> 11 -> 10 -> 00) для кожного «кроку», який він робить.
- *steplen* - (u32, RW) Тривалість ступінчастого сигналу в наносекундах.
- *stepspace* - (u32, RW) Мінімальний інтервал між сигналами кроків, у наносекундах.

6.7.13.3 Вихідні параметри

Виводи Step та Direction кожного StepGen мають два додаткові параметри. Щоб знайти, який вивід вводу/виводу належить до якого виводу кроку та напрямку, виконайте команду *dmesg*, як описано вище.

- *invert_output* - (біт, RW) Цей параметр діє тільки в тому випадку, якщо параметр *is_output* має значення true. Якщо цей параметр має значення true, вихідне значення GPIO буде оберненим до значення на виводі HAL *out*.
- *is_opendrain* - (біт, RW) Якщо цей параметр має значення false, GPIO працює як звичайний вихідний контакт: контакт вводу-виводу на роз’ємі приводиться до значення, заданого контактом HAL *out* (можливо інвертованим). Якщо цей параметр має значення true, GPIO працює як контакт з відкритим стоком. Запис 0 на вивід HAL «out» приводить вивід вводу-виводу в низький стан, запис 1 на вивід HAL «out» переводить вивід вводу-виводу у стан високого імпедансу.

У цьому стані високого імпедансу вивід вводу-виводу плаває (слабо підтягується до високого рівня), і інші пристрої можуть керувати значенням; отримане значення на виводі вводу-виводу доступне на виводах «in» і «in_not». Цей параметр мають тільки повні контакти GPIO і контакти вводу-виводу, які використовуються як виходи активними екземплярами модулів.

6.7.14 PWMGen

PWMgens мають такі імена: «hm2_<BoardType>.<BoardNum>.pwmgen.<Instance>». «Instance» — це двозначне число, яке відповідає номеру екземпляра HostMot2 pwmgen. Існує «num_pwmgens» екземплярів, починаючи з 00.

У HM2 кожен pwmgen використовує три виходи I/O: Not-Enable, Out0 та Out1. Щоб інвертувати вихідний вивід PWMGen, інвертуйте відповідний вивід GPIO, який використовується PWMGen. Щоб знайти вивід GPIO, який використовується для виводу PWMGen, запустіть *dmesg*, як показано вище.

Функція контактів вводу/виводу Out0 та Out1 залежить від параметра типу виводу (див. нижче).

Представлення hm2 pwmgen подібне до програмного компонента pwmgen. Кожен екземпляр pwmgen має такі контакти та параметри:

6.7.14.1 Піни

- *enable* - (Біт, Вхід) Якщо значення true (істина), pwmgen встановить свій пін Not-Enable у значення false (неправда) та видаватиме імпульси. Якщо значення *enable* має значення false (хиба), pwmgen встановить свій пін Not-Enable у значення true та не видаватиме жодних сигналів.
- *value* - (Float, In) Поточне значення команди pwmgen у довільних одиницях.

6.7.14.2 Параметри

- *output-type* - (s32, RW) Емулює аргумент output_type load-time для програмного компонента pwmgen. Цей параметр можна змінювати під час виконання, але в більшості випадків його слід встановлювати під час запуску і не змінювати. Прийнятні значення: 1 (PWM на Out0 і Direction на Out1), 2 (Up на Out0 і Down на Out1), 3 (режим PDM, PDM на Out0 і Dir на Out1) і 4 (Direction на Out0 і PWM на Out1, «для заблокованої антифази»).
- «scale» — (Float, RW) Коефіцієнт масштабування для перетворення «value» з довільних одиниць у робочий цикл: dc = value / scale. Робочий цикл має ефективний діапазон від -1,0 до +1,0 включно, все, що виходить за межі цього діапазону, обрізається.
- *pdm_frequency* - (u32, RW) Вказує частоту PDM, в Гц, для всіх екземплярів pwmgen, що працюють у режимі PDM (режим 3). Це «частота імпульсного слота»; частота, з якою генератор pdm на платі Anything I/O вибирає, чи випромінювати імпульс, чи проміжок. Кожен імпульс (і проміжок) у послідовності імпульсів PDM має тривалість 1/pdm_frequency секунд. Наприклад, якщо встановити pdm_frequency на $2*10^6$ Hz (2 MHz) і робочий цикл на 50%, отримаємо прямокутну хвилю 1 MHz, ідентичну сигналу PWM 1 MHz з робочим циклом 50%. Ефективний діапазон цього параметра становить від приблизно 1525 Hz до трохи менше 100 MHz. Зверніть увагу, що максимальна частота визначається частотою ClockHigh плати Anything I/O; плати 5I20 і 7I43 мають тактову частоту 100 MHz, що дає максимальну частоту PDM 100 MHz. Інші плати можуть мати інші тактові частоти, що призводить до різних максимальних частот PDM. Якщо користувач спробує встановити занадто високу частоту, вона буде обмежена максимальною частотою, що підтримується платою.
- *pwm_frequency* - (u32, RW) Вказує частоту PWM, в Гц, для всіх екземплярів pwmgen, що працюють в режимах PWM (режими 1 і 2). Це частота хвилі зі змінним робочим циклом. Її ефективний

діапазон становить від 1 Гц до 193 кГц. Зверніть увагу, що максимальна частота визначається частотою ClockHigh плати Anything I/O; плати 5i20 і 7i43 мають тактову частоту 100 МГц, що дає максимальну частоту PWM 193 кГц. Інші плати можуть мати інші тактові частоти, що призводить до різних максимальних частот PWM. Якщо користувач спробує встановити занадто високу частоту, вона буде обмежена максимальною частотою, що підтримується платою. Частоти нижче приблизно 5 Гц не є надто точними, але вище 5 Гц вони досить близькі до точних.

6.7.14.3 Вихідні параметри

Вихідні контакти кожного PWM-генератора мають два додаткові параметри. Щоб знайти, який контакт вводу/виводу належить до якого виходу, запустіть `dmesg`, як описано вище.

- `invert_output` - (біт, RW) Цей параметр діє тільки в тому випадку, якщо параметр `is_output` має значення true. Якщо цей параметр має значення true, вихідне значення GPIO буде оберненим до значення на виводі HAL `out`.
- `is_opendrain` - (біт, RW) Якщо цей параметр має значення false, GPIO працює як звичайний вихідний контакт: контакт вводу-виводу на роз'ємі приводиться до значення, заданого контактом HAL `out` (можливо інвертованим). Якщо цей параметр має значення true, GPIO працює як контакт з відкритим стоком. Запис 0 у вивід HAL `out` приводить вивід вводу-виводу в низький стан, запис 1 у вивід HAL `out` переводить вивід вводу-виводу у стан високого імпедансу. У цьому стані з високим імпедансом контакт вводу/виводу плаває (слабо підтягується до високого рівня), і інші пристрої можуть керувати значенням; отримане значення на контакті вводу/виводу доступне на контактах `in` та `in_not`. Цей параметр мають лише повні контакти GPIO та контакти вводу/виводу, які використовуються як виходи активними екземплярами модулів.

6.7.15 Енкодер

Енкодери мають такі імена: `hm2_<BoardType>.<BoardNum>.encoder.<Instance>.`. `Instance` — це двозначне число, яке відповідає номеру екземпляра енкодера HostMot2. Існує *num_encoders* екземплярів, починаючи з 00.

Кожен енкодер використовує три або чотири вхідні контакти вводу-виводу, залежно від того, як було скомпільовано прошивку. Триконтактні енкодери використовують A, B та Index (іноді також відомий як Z). Чотириконтактні енкодери використовують A, B, Index та Index-mask.

Представлення кодера hm2 схоже на те, що описано в Canonical Device Interface (у документі HAL General Reference), та на компонент програмного кодера. Кожен екземпляр кодера має такі контакти та параметри:

6.7.15.1 Піни

- `count` - (s32, Out) Кількість підрахунків енкодера з моменту останнього скидання.
- `index-enable` - (біт, ввід/вивід) Коли цей вивід встановлений у стан True, лічильник (а отже, і позиція) обнуляються при наступному імпульсі індексу (фаза Z). Одночасно index-enable обнуляється, щоб вказати на те, що імпульс відбувся.
- `position` - (Float, Out) Положення енкодера в одиницях позиції (підрахунок / масштаб).
- `rawcounts` - (s32, Out) Загальна кількість підрахунків енкодера з початку, без урахування індексу або скидання.
- `reset` - (біт, вхід) Коли цей вивід має значення TRUE, виводи підрахунку та позиції встановлюються на 0. Це не впливає на значення виводу швидкості. Драйвер не скидає цей вивід на FALSE після скидання підрахунку на 0, це завдання користувача.
- `velocity` - (Float, Out) Орієнтовна швидкість енкодера в одиницях позиції за секунду.

6.7.15.2 Параметри

- `counter-mode` - (біт, RW) Встановіть значення False (за замовчуванням) для квадратури. Встановіть значення True для Up/Down або для одного входу на фазі A. Може використовуватися для перетворювача частоти в швидкість з одним входом на фазі A, якщо встановлено значення true.
- `filter` - (біт, RW) Якщо встановлено значення True (за замовчуванням), квадратурний лічильник потребує 15 тактів, щоб зареєструвати зміну на будь-якій з трьох вхідних ліній (будь-який імпульс, коротший за цей, відкидається як шум). Якщо встановлено значення False, квадратурний лічильник потребує лише 3 тактів, щоб зареєструвати зміну. Тактова частота зразка кодера становить 33 МГц на картах PCI Anything I/O і 50 МГц на 7I43.
- `index-invert` - (біт, RW) Якщо встановлено значення True, наростаючий фронт вхідного виводу Index запускає подію Index (якщо index-enable має значення True). Якщо встановлено значення False, спрацьовує спадаючий фронт.
- `index-mask` - (біт, RW) Якщо встановлено значення True, вхідний контакт Index має ефект лише тоді, коли вхідний контакт Index-Mask має значення True (або False, залежно від контакту `index-mask-invert` нижче).
- `index-mask-invert` - (Bit, RW) Якщо встановлено значення True, Index-Mask має бути False, щоб Index мав ефект. Якщо встановлено значення False, контакт `Index-Mask` має бути True.
- `scale` - (Float, RW) Перетворює одиниці виміру з «count» в одиниці виміру «position». Квадратурний енкодер зазвичай має 4 імпульси на імпульс, тому енкодер 100 PPR матиме 400 імпульсів на оберт. У режимі `.counter-mode` енкодер 100 PPR матиме 100 імпульсів на оберт, оскільки він використовує тільки передній фронт A, а напрямок - B.
- `vel-timeout` - (Float, RW) Коли енкодер рухається повільніше, ніж один імпульс за кожен раз, коли драйвер зчитує лічильник з FPGA (у функції hm2_read()), швидкість важче оцінити. Драйвер може чекати кілька ітерацій на наступний імпульс, весь час повідомляючи про верхню межу швидкості енкодера, яку можна точно вгадати. Цей параметр визначає, скільки часу чекати на наступний імпульс, перш ніж повідомити про зупинку енкодера. Цей параметр вимірюється в секундах.

6.7.16 Конфігурація 5I25

6.7.16.1 Прошивка

Прошивка 5I25 постачається попередньо завантаженою для дочірньої карти, з якою вона придбана. Тому `firmware=xxx.BIT` не є частиною рядка конфігурації hm2_pci під час використання 5I25.

6.7.16.2 Конфігурація

Приклади конфігурацій плат 5I25/7I76 та 5I25/7I77 наведено у Вибір конфігурації.

Якщо ви бажаєте налаштувати власну конфігурацію, наступні приклади показують, як завантажити драйвери у файл HAL.

5I25 + 7I76 Картка

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''уb''b' ↩
    'нb''b''іb''b''вb''b''eb''b''pb''b''cb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b' ↩
    'дb''b''pb''b''ab''b''йb''b''вb''b''eb''b''pb''
loadrt hostmot2
```

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''тb''b''eb'' b''дb''b''pb''b' ↩
    'ab''b''йb''b''вb''b''eb''b''pb'' PCI b''тb''b''ab'' b''нb''b''ab''b''лb''b''ab''b''шb'' ↩
    b''тb''b''уb''b''йb''b''тb''b''eb'' b''йb''b''ob''b''гb''b''ob''
loadrt hm2_pci config="num_encoders=1 num_stepgens=5 sserial_port_0=0XXX"
```

5I25 + 7I77 Картка

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''уb''b' ↩
    'нb''b''іb''b''вb''b''eb''b''pb''b''cb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b' ↩
    'дb''b''pb''b''ab''b''йb''b''вb''b''eb''b''pb''
loadrt hostmot2

# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''тb''b''eb'' b''дb''b''pb''b' ↩
    'ab''b''йb''b''вb''b''eb''b''pb'' PCI b''тb''b''ab'' b''нb''b''ab''b''лb''b''ab''b''шb'' ↩
    b''тb''b''уb''b''йb''b''тb''b''eb'' b''йb''b''ob''b''гb''b''ob''
loadrt hm2_pci config="num_encoders=6 num_pwmgens=6 sserial_port_0=0XXX"
```

6.7.16.3 Конфігурація SSERIAL

Конфігураційний рядок `sserial_port_0=0XXX` встановлює деякі параметри для інтелектуальної дочірньої карти послідовного порту. Ці параметри є специфічними для кожної дочірньої карти. Дивіться посібник Mesa для отримання додаткової інформації про точне використання (зазвичай у розділі під назвою РЕЖИМИ ОБРОБКИ ДАНИХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ) або дивіться сторінку посібника за посиланням:../man/man9/sserial.9.html[SSERIAL(9)].

6.7.16.4 7I77 Ліміти

Minlimit та maxlimit – це обмеження для значення виводу (у цьому випадку значення аналогового виходу), fullscalemax – це коефіцієнт масштабування.

За замовчуванням вони встановлені на аналоговий вхід або аналоговий діапазон (найімовірніше, у вольтах).

Наприклад, для аналогових виходів 7I77 +-10 В значення за замовчуванням такі:

minlimit: -10
maxlimit: +10
maxfullscale: 10

Якщо ви хочете масштабувати аналоговий вихід каналу до IPS для сервоприводу в режимі швидкості (скажімо, максимум 24 IPS), ви можете встановити обмеження ось так:

minlimit: -24
maxlimit: +24
maxfullscale: 24

Якщо ви хочете масштабувати аналоговий сигнал з каналу до обертів за хвилину для шпинделя від 0 до 6000 об/хв з керуванням 0-10 В, ви можете встановити обмеження таким чином:

minlimit: 0
maxlimit: 6000
maxfullscale: 6000
(це запобігло б встановленню небажаної негативної вихідної напруги)

6.7.17 Приклади конфігурацій

LinuxCNC містить кілька прикладів конфігурацій для апаратного забезпечення Mesa. Конфігурації знаходяться в розділах hm2-servo та hm2-stepper Configuration Selector. Зазвичай для завантаження обраної конфігурації потрібно встановити плату. Приклади є хорошим початком і допоможуть заощадити час. Просто виберіть відповідний приклад із селектора конфігурації LinuxCNC і збережіть його копію на своєму комп'ютері, щоб мати можливість редагувати. Щоб побачити точні контакти та параметри, які надала ваша конфігурація, відкрийте вікно «Показати конфігурацію HAL» у меню «Машина» або виконайте команду `dmesg`, як описано вище.

6.8 MB2HAL

6.8.1 Вступ

MB2HAL — це універсальний компонент HAL, що не працює в реальному часі, для зв'язку з одним або кількома пристроями Modbus. Наразі існує два варіанти зв'язку з пристроєм Modbus:

1. Один із варіантів — створити компонент HAL як драйвер, див. VFD Modbus.
2. Іншим варіантом є використання Classic Ladder з вбудованим Modbus, див. ClassicLadder.
3. Тепер є третій варіант, який складається з "загального" драйвера, налаштованого текстовим файлом, він називається MB2HAL.

Чому варто обрати MB2HAL? Розгляньте можливість використання MB2HAL, якщо:

- Вам потрібно написати новий драйвер, і ви нічого не знаєте про програмування.
- Для керування з'єднаннями Modbus потрібно використовувати "лише" Classic Ladder.
- Вам потрібно виявити та налаштувати перші транзакції Modbus. MB2HAL має рівні налагодження для полегшення низькорівневого налагодження протоколу.
- У вас є більше одного пристрою для підключення. MB2HAL дуже ефективно управляє декількома пристроями, транзакціями та посиланнями. В даний час я контролюю два двоосьові драйвери за допомогою порту Rs232, драйвер VFD за допомогою іншого порту Rs232 та віддалений ввід/вивід за допомогою TCP/IP.
- Вам потрібен протокол для підключення вашого Arduino до HAL. Перегляньте доданий зразок файлу конфігурації, ескізу та бібліотеки для Arduino Modbus.

6.8.2 Застосування

a. Створіть конфігураційний файл з наведеного нижче прикладу

   1. Встановити назву компонента (необов'язково)
      Отримати `HAL_MODULE_NAME=mymodule` (за замовчуванням `HAL_MODULE_NAME=mb2hal`)
   2. Завантажити компонент Modbus HAL, що не працює в режимі реального часу

b. Назва компонента за замовчуванням: `loadusr -W mb2hal config=config_file.ini`

c. Назва користувацького компонента: `loadusr -Wn mymodule mb2hal config=config_file.ini`

6.8.3 Опції

6.8.3.1 Розділ ініціалізації

[MB2HAL_INIT]

Значення	Тип	Обов’язково	Опис
INIT_DEBUG	Ціле число	Ні	Рівень налагодження ініціалізації та розбору INI-файлів. 0 = беззвучно 1 = повідомлення про помилки (за замовчуванням) 2 = повідомлення про підтвердження OK 3 = повідомлення про налагодження 4 = максимальна кількість повідомлень про налагодження (лише в транзакціях)
ВЕРСІЯ	Рядок	Ні	Номер версії у форматі N.N[NN]. За замовчуванням — 1.0.
HAL_MODULE_NAME	Рядок	Ні	Назва модуля (компонента) HAL. За замовчуванням — «mb2hal».
SLOWDOWN	Float	Ні	Вставте затримку «FLOAT секунд» між транзакціями, щоб уникнути великої кількості записів у журналі та полегшити налагодження. Корисно при використанні DEBUG=3 (НЕ INIT_DEBUG=3). Це впливає на ВСІ транзакції. Для нормальної роботи використовуйте «0.0».
TOTAL_TRANSACTIONS	Ціле число	Так	Загальна кількість транзакцій Modbus. Максимальної кількості немає..

6.8.3.2 Розділи транзакцій

Для кожної транзакції необхідний один розділ транзакції, починаючи з [TRANSACTION_00] і послідовно збільшуючи номер. Якщо є нове посилання (не транзакція), ви повинні вказати ОБОВ’ЯЗКОВІ параметри вперше. Попередження: будь-які НЕОБОВ’ЯЗКОВІ параметри, які не вказані, копіюються з попередньої транзакції.

Значення	Тип	Обов’язково	Опис
LINK_TYPE	Рядок	Так	Ви повинні вказати посилання ”serial” або ”tcp” для першої транзакції. Якщо не вказано посилання попередньої транзакції, наступні транзакції використовуватимуть його.
TCP_IP	IP-адреса	Якщо LINK_TYPE=tcp	IP-адреса веденого пристрою Modbus. Ігнорується, якщо LINK_TYPE=serial.
TCP_PORT	Ціле число	Ні	TCP-порт веденого пристрою Modbus. За замовчуванням 502. Ігнорується, якщо LINK_TYPE=serial.
SERIAL_PORT	Рядок	Якщо LINK_TYPE=serial	Послідовний порт. Наприклад, ”/dev/ttyS0”. Ігнорується, якщо LINK_TYPE=tcp.
SERIAL_BAUD	Ціле число	Якщо LINK_TYPE=serial	Швидкість передачі даних. Ігнорується, якщо LINK_TYPE=tcp.
SERIAL_BITS	Ціле число	Якщо LINK_TYPE=serial	Біти даних. Один з 5, 6, 7, 8. Ігнорується, якщо LINK_TYPE=tcp.
SERIAL_PARITY	Рядок	Якщо LINK_TYPE=serial	Парність даних. Одне з: парне, непарне, жодного. Ігнорується, якщо LINK_TYPE=tcp.
SERIAL_STOP	Ціле число	Якщо LINK_TYPE=serial	Стоп-біти. Один з 1, 2. Ігнорується, якщо LINK_TYPE=tcp.

Значення	Тип	Обов’язково	Опис
SERIAL_DELAY_MS	Ціле число	Якщо LINK_TYPE=serial	Затримка послідовного порту між транзакціями лише серіального розділу. У мс. За замовчуванням 0. Ігнорується, якщо LINK_TYPE=tcp.
MB_SLAVE_ID	Ціле число	Так	Номер веденого пристрою Modbus.
FIRST_ELEMENT	Ціле число	Так	Адреса першого елемента.
NELEMENTS	Ціле число	Якщо не вказано PIN_NAMES	Кількість елементів. Вказувати одночасно NELEMENTS та PIN_NAMES є помилкою. Назви контактів будуть послідовними номерами, наприклад, mb2hal.plcin.01.
PIN_NAMES	Список	Якщо не вказано NELEMENTS	Список імен елементів. Ці імена будуть використовуватися для імен контактів, наприклад mb2hal.plcin.cycle_start. **ПРИМІТКА:** У списку не повинно бути пробілів. Приклад: PIN_NAMES=cycle_start,stop,feed_hold
MB_TX_CODE	Рядок	Так	Код функції транзакції Modbus (див. посилання: специфікації): • fnct_01_read_coils • fnct_02_read_discrete_inputs • fnct_03_read_holding_registers • fnct_04_read_input_registers • fnct_05_write_single_coil • fnct_06_write_single_register • fnct_15_write_multiple_coils • fnct_16_write_multiple_registers
MB_RESPONSE_TIMEOUT_MS	Ціле число	Ні	Тайм-аут відповіді для цієї транзакції. У мс. За замовчуванням 500 мс. Це час очікування першого байта перед виникненням помилки.
MB_BYTE_TIMEOUT_MS	Ціле число	Ні	Тайм-аут байта для цієї транзакції. У мс. За замовчуванням 500 мс. Це час очікування від байта до байта, перш ніж виникне помилка.
HAL_TX_NAME	Рядок	Ні	Замість номера транзакції використовуйте ім’я. Приклад: mb2hal.00.01 може стати mb2hal.plcin.01. Ім’я не повинно перевищувати 28 символів. **ПРИМІТКА:** при використанні імен будьте обережні, щоб не створити дві транзакції з однаковим ім’ям.
MAX_UPDATE_RATE	float	Ні	Максимальна частота оновлення в Гц. За замовчуванням встановлено значення 0,0 (0,0 = як тільки з’явиться можливість = нескінченно). **ПРИМІТКА:** Це максимальна частота, фактична частота може бути нижчою. Якщо ви хочете обчислити її в мс, використовуйте (1000 / required_ms). Приклад: 100 мс = MAX_UPDATE_RATE=10.0, оскільки 1000,0 мс / 100,0 мс = 10,0 Гц.
НАЛАГОДЖЕННЯ	Рядок	Ні	Рівень налагодження лише для цієї транзакції. Див. параметр INIT_DEBUG вище.

6.8.3.3 Коди помилок

Під час налагодження транзакцій зверніть увагу, що повернене значення ”ret[]” відповідає:

Винятки протоколу Modbus:

- 0x01 - ILLEGAL_FUNCTION - код FUNCTION, отриманий у запиті, є недозволеним або недійсним.

- 0x02 - ILLEGAL_DATA_ADDRESS - адреса даних, отримана в запиті, не є допустимою адресою для ведомого пристрою або є недійсною.
- 0x03 - ILLEGAL_DATA_VALUE - значення, що міститься в полі запиту даних, не є допустимим або недійсним.
- 0x04 - SLAVE_DEVICE_FAILURE - ЗБІЙ ПІДЛЕЖЕНОГО (або ГОЛОВНОГО) пристрою, що неможливо відновити, під час спроби виконання запитуваної дії.
- 0x04 - SERVER_FAILURE - (див. вище).
- 0x05 - ПІДТВЕРДЖЕННЯ - Ця відповідь повертається для ЗАПОБІГАННЯ ЧАСУ ОЧИЩЕННЯ (PREVENT A TIMEOUT) у головному пристрої. Для обробки запиту у веденому пристрої потрібен тривалий час.
- 0x06 - SLAVE_DEVICE_BUSY - Ведений пристрій (або сервер) ЗАЙНЯТИЙ. Повторіть запит пізніше.
- 0x06 - SERVER_BUSY - (див. вище).
- 0x07 - NEGATIVE_ACKNOWLEDGE - Невдалий запит на програмування з використанням коду функції 13 або 14.
- 0x08 - MEMORY_PARITY_ERROR - Помилка парності SLAVE в ПАМ’ЯТІ.
- 0x0A (-10) - GATEWAY_PROBLEM_PATH - Шлях(и) шлюзу недоступні.
- 0x0B (-11) - GATEWAY_PROBLEM_TARGET - Цільовий пристрій не відповів (згенеровано головним, а не ведомим пристроєм).

Програма або з’єднання:

- 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 Приклад конфігураційного файлу

Натисніть тут для завантаження.

```
#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 Піни

Note
Жовтий = Нове в MB2HAL 1.1 (LinuxCNC 2.9). Щоб використовувати ці нові функції, потрібно встановити `VERSION = 1.1`.
m = Значення `HAL_TX_NAME`, якщо встановлено, або номер транзакції
n = Номер елемента (`NELEMENTS`) або назва з `PIN_NAMES`
Приклад:

- `mb2hal.00.01.int` (TRANSACTION_00, другий регістр)
- `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 має бути 1, або PIN_NAMES має містити лише одне ім’я.

6.8.5.6 fnct_06_write_single_register

- mb2hal.*m*.*n*.float *float in*
- mb2hal.*m*.*n*.int *s32 in*

NELEMENTS має дорівнювати 1, або PIN_NAMES має містити лише одну назву. Обидва значення виводів додаються та обмежені значенням 65535 (UINT16_MAX). Використовуйте одне, а інше розмикайте (читайте як 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*

Обидва значення виводів додаються та обмежені значенням 65535 (UINT16_MAX). Використовуйте один, а інший розімкнутий (зчитується як 0).

6.9 Драйвер частотного перетворювача Mitsub

Це нереальна програма HAL, написана на Python, для управління VFD від Mitsubishi.
Зокрема, серії A500 F500 E500 A500 D700 E700 F700 - інші можуть працювати.
mitsub_vfd підтримує послідовне управління за допомогою протоколу RS485.
Для перетворення з USB або послідовного порту в RS485 потрібне спеціальне обладнання.

Note
Оскільки це програма, що не працює в режимі реального часу, на її роботу можуть впливати завантаження комп’ютера та затримка. Можлива втрата контролю над VFD. За бажанням можна налаштувати VFD на зупинку у разі втрати зв’язку. Завжди слід мати схему Estop, яка відключає живлення пристрою в разі надзвичайної ситуації.

Цей компонент завантажується за допомогою команди halcmd ”loadusr”:

```
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01
```

Наведена вище команда говорить:
loadusr, чекати готовності контактів охолоджувальної рідини, компонент mitsub_vfd, з 2 веденими пристроями під назвою шпиндель (ведений №2) та охолоджувальна рідина (ведений №1)

6.9.1 Параметри командного рядка

Параметри командного рядка:

- *-b or --baud <rate>* : встановити швидкість передачі даних – усі мережеві частотні перетворювачі повинні бути однаковими
- *-p or --port <device path>* : встановлює порт для використання, наприклад /dev/ttyUSB0
- *<name>=<slave#>* : встановлює назву компонента/виводу HAL та номер підлеглого пристрою.

Налагодження можна ввімкнути, встановивши для виводу налагодження значення true.

Note
Увімкнення налагодження призведе до перевантаження терміналу текстом.

6.9.2 Піни

Де <n> — це `mitsub_vfd` або ім’я, задане під час завантаження.

- *<n>.fwd* (біт, вхід) True встановлює рух вперед, False — назад.
- *<n>.run* (біт, вхід) True запускає ЧРП на основі виводу .fwd.
- *<n>.debug* (біт, вхід) Виводить інформацію про налагодження в термінал.
- *<n>.alarm* (біт, вихід) сигналізує про стан тривоги частотно-регульованого приводу.
- *<n>.up-to-speed* (біт, вихід), коли привід досягає заданої швидкості (допуск швидкості встановлено на частотному перетворювачі частоти)
- *<n>.monitor* (біт, вхід) деякі моделі (наприклад, E500) не можуть контролювати стан - у цьому випадку встановіть для виводу монітора значення false, такі виводи, як up-to-speed, ampers, alarm та status, не оновлюються.
- *<n>.motor-cmd* (float, in) – команда швидкості, що передається частотному перетворювачу (за замовчуванням масштабується в герцах).
- *<n>.motor-fb* (float, out) швидкість зворотного зв’язку від частотного перетворювача (за замовчуванн масштабується в герцах).
- *<n>.motor-amps* (float, out) Поточна вихідна сила струму двигуна.
- *<n>.motor-power* (float, out) Поточна вихідна потужність двигуна.
- *<n>.scale-cmd* (float, in) Масштабує вивід motor-cmd до довільних одиниць. За замовчуванням 1 = герц.
- *<n>.scale-fb* (float, in) Масштабує вивід motor-fb до довільних одиниць. за замовчуванням 1 = герц.
- *<n>.scale-amps* (float, in) Масштабує вивід струму двигуна до довільних одиниць. за замовчуванням 1 = ampers.
- *<n>.scale-power* (float, in) Масштабує вивід живлення двигуна до довільних одиниць. за замовчуванн 1 = .
- *<n>.estop* (біт, in) переводить частотний перетворювач у стан аварійної зупинки.
- *<n>.status-bit-N* (біт, вихід) N = від 0 до 7, біти стану налаштовуються користувачем на частотному перетворювачі. Біт 3 має бути встановлений на швидкість, а біт 7 – на сигналізацію. Інші можна встановити за потреби.

6.9.3 HAL приклад

```
#
# b''пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b''вb''b''иb''b''кb''b''оb''b''pb''b' ↩
    'иb''b''сb''b''тb''b''ab''b''нb''b''нb''b''яb'' b''дb''b''pb''b''ab''b''йb''b''вb''b' ↩
    'еb''b''pb''b''ab'' b''чb''b''ab''b''сb''b''тb''b''оb''b''тb''b''нb''b''оb''b''гb''b' ↩
    'оb'' b''пb''b''еb''b''pb''b''еb''b''тb''b''вb''b''оb''b''pb''b''юb''b''вb''b''ab''b' ↩
    'чb''b''ab'' Mitsubishi
#
loadusr -Wn coolant mitsub_vfd spindle=02 coolant=01

# ********** b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''яb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''еb''b''лb''b''ьb''b''нb''b''оb''b' ↩
    'гb''b''оb'' b''чb''b''ab''b''сb''b''тb''b''оb''b''тb''b''нb''b''оb''b''гb''b''оb'' b' ↩
    'пb''b''еb''b''pb''b''еb''b''тb''b''вb''b''оb''b''pb''b''юb''b''вb''b''ab''b''чb''b' ↩
    'ab'', b''вb''b''еb''b''дb''b''еb''b''нb''b''иb''b''йb'' b''пb''b''pb''b''иb''b''сb''b' ↩
    'тb''b''pb''b''ib''b''йb'' 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

# ********** b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''яb'' b''чb''b''ab''b''сb''b''тb''b''оb''b''тb''b''нb''b''оb''b''гb''b''оb'' b' ↩
    'пb''b''еb''b''pb''b''еb''b''тb''b''вb''b''оb''b''pb''b''юb''b''вb''b''ab''b''чb''b' ↩
    'ab'' b''оb''b''xb''b''оb''b''лb''b''оb''b''дb''b''жb''b''уb''b''вb''b''ab''b''лb''b' ↩
    'ьb''b''нb''b''оb''b''їb'' b''pb''b''ib''b''дb''b''иb''b''нb''b''иb'', b''вb''b''еb''b' ↩
    'дb''b''еb''b''нb''b''иb''b''йb'' b''пb''b''pb''b''иb''b''сb''b''тb''b''pb''b''ib''b' ↩
    'йb'' 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 Налаштування частотного перетворювача Mitsubishi для послідовного використання

6.9.4.1 Підключення послідовного порту

Перетворювачі частоти Mitsubishi мають роз’єм RJ-45 для послідовної передачі даних.
Оскільки вони використовують протокол RS485, їх можна об’єднати в мережу за принципом «точка-точка».
Цей драйвер було протестовано з використанням Opto22 AC7A для перетворення RS232 в RS485.

6.9.4.2 Налаштування Modbus

Посилання на посібники:

«Довідковий посібник з опцій зв’язку» та «Технічний посібник A500» для серії 500.
«Технічний посібник Fr-A700 F700 E700 D700» для серії 700

Для послідовного зв’язку частотно-регульований перетворювач (ЧР) необхідно вручну налаштувати параметри PR.
Щоб зареєструвати деякі з них, необхідно вимикати та перемикати живлення ЧРД, наприклад, PR 79

- «PR 77» встановлено на 1 «-щоб розблокувати інші модифікації PR»
- «PR 79» встановлено на 1 або 0 «-для зв’язку через послідовний порт»
- «PR 117» встановлено на 0-31 «-номер веденого пристрою, драйвер повинен посилатися на той самий номер»
- «PR 118» протестовано зі швидкістю передачі даних 96 бод (можна встановити на 48, 96, 192 боди), якщо також встановлено драйвер.’
- *PR 119* встановлено на 0 *-стоповий біт/довжина даних (8 бітів, два стопи)*
- «PR 120» встановлено на 0 «-без парності»
- «PR 121» встановлено на 1-10 «- якщо 10 (максимум) помилок COM, то несправності частотного перетворювача»
- «PR 122» протестовано з кодом 9999 «- якщо зв’язок втрачено, частотний перетворювач не видасть помилку»
- «PR 123» встановлено на 9999 «-час очікування до кадру послідовних даних не додається»
- *PR 124* встановлено на 0 ’-без повернення каретки в кінці рядка

6.10 Водій Мотенк

Вітальні системи Motenc-100 та Motenc-LITE

Vital Systems Motenc-100 і Motenc-LITE — це 8- і 4-канальні плати сервоуправління. Motenc-100 має 8 лічильників квадратурних енкодерів, 8 аналогових входів, 8 аналогових виходів, 64 (68?) цифрових входів і 32 цифрових виходи. Motenc-LITE має лише 4 лічильники енкодерів, 32 цифрових входи і 16 цифрових виходів, але все одно має 8 аналогових входів і 8 аналогових виходів. Драйвер автоматично ідентифікує встановлену плату і експортує відповідні об’єкти HAL.

Встановлення:

```
loadrt hal_motenc
```

Під час завантаження (або спроби завантаження) драйвер виводить деякі корисні повідомлення про налагодження до журналу ядра, які можна переглянути за допомогою dmesg.

В одній системі можна використовувати до 4 плат.

6.10.1 Піни

У наступних контактах, параметрах і функціях <board> є ідентифікатором плати. Згідно з правилами іменування, перша плата завжди повинна мати ідентифікатор нуль. Однак цей драйвер встановлює ідентифікатор на основі пари перемичок на платі, тому він може бути відмінним від нуля, навіть якщо є тільки одна плата.

- *(s32) motenc.<board>.enc-<channel>-count* - Положення енкодера, в одиницях.
- *(float) motenc.<board>.enc-<channel>-position* - Положення енкодера, в одиницях користувача.
- *(bit) motenc.<board>.enc-<channel>-index* - Поточний стан вхідного індексного імпульсу.
- *(bit) motenc.<плата>.enc-<канал>-idx-latch* – Драйвер встановлює цей вивід у значення true, коли він фіксує індексний імпульс (увімкнено latch-index). Очищається очищенням latch-index.
- *(bit) motenc.<плата>.enc-<канал>-latch-index* - Якщо цей вивід має значення ”true”, драйвер скине лічильник при наступному індексному імпульсі.
- *(bit) motenc.<board>.enc-<channel>-reset-count* - Якщо цей вивід має значення «true», лічильник негайно скинеться до нуля, а вивід очиститься.
- *(float) motenc.<board>.dac-<channel>-value* - Аналогове вихідне значення для DAC (в одиницях користувача, див. -коефіцієнти посилення та -зміщення)
- *(float) motenc.<board>.adc-<channel>-value* - Аналогове вхідне значення, що зчитується ADC (в одиницях користувача, див. -коефіцієнт підсилення та -зсуву),
- *(bit) motenc.<board>.in-<channel>* - Стан цифрового вхідного виводу, див. канонічний цифровий вхід.
- *(bit) motenc.<board>.in-<channel>-not* - Інвертований стан цифрового вхідного виводу, див. канонічний цифровий вхід.
- *(bit) motenc.<board>.out-<channel>* - Значення, яке буде записано на цифровий вихід, розглядається як канонічний цифровий вихід.
- *(bit) motenc.<board>.estop-in* - Спеціальний вхід для estop, потрібні додаткові деталі.
- *(bit) motenc.<board>.estop-in-not* - Інвертований стан виділеного входу estop.
- *(bit) motenc.<board>.watchdog-reset* - Двонаправлений, - Встановіть значення TRUE для одноразового скидання сторожового таймера, автоматично очищається.

6.10.2 Параметри

- *(float) motenc.<board>.enc-<channel>-scale* - Кількість підрахунків / одиниця користувача (для перетворення з підрахунків в одиниці).
- *(float) motenc.<board>.dac-<channel>-offset* - Встановлює зміщення DAC.
- *(float) motenc.<board>.dac-<channel>-gain* - Встановлює коефіцієнт підсилення (масштабування) DAC.
- *(float) motenc.<board>.adc-<channel>-offset* - Встановлює зміщення ADC.
- *(float) motenc.<board>.adc-<channel>-gain* - Встановлює коефіцієнт підсилення (масштабування) ADC.
- *(bit) motenc.<board>.out-<channel>-invert* - Інвертує цифровий вихід, див. канонічний цифровий вихід.

- *(u32) motenc.<board>.watchdog-control* - Налаштовує сторожовий таймер. Значення може бути побітовим АБО одним із наступних значень:

Біт #	Значення	Значення
0	1	Тайм-аут становить 16 мс, якщо встановлено, 8 мс, якщо не встановлено
1	2	
2	4	Сторожовий таймер увімкнено
3	8	
4	16	Сторожовий таймер автоматично скидається записами DAC (функція запису DAC у HAL)

Зазвичай корисними значеннями є 0 (сторожовий таймер вимкнено) або 20 (сторожовий таймер 8 мс увімкнено, очищається командою dac-write).

- *(u32) motenc.<board>.led-view* - Зв’язує деякі входи/виходи з вбудованими світлодіодами.

6.10.3 Функції

- *(funct) motenc.<board>.encoder-read* - Зчитує всі лічильники енкодера.
- *(funct) motenc.<board>.adc-read* - Зчитує аналого-цифрові перетворювачі.
- *(funct) motenc.<board>.digital-in-read* - Зчитує цифрові входи.
- *(funct) motenc.<board>.dac-write* - Записує напруги на DAC.
- *(funct) motenc.<board>.digital-out-write* - Записує цифрові виходи.
- *(funct) motenc.<board>.misc-update* - Оновлення різних речей.

6.11 Драйвер Opto22

PCI AC5 АДАПТЕРНА КАРТКА / ДРАЙВЕР HAL

6.11.1 Адаптерна плата

Це карта, виготовлена Opto22 для адаптації PCI-порту до твердотільних релейних стійок, таких як їх стандартна серія або серія G4. Вона має 2 порти, які можуть керувати до 24 точками кожен, і 4 вбудовані світлодіоди. Порти використовують 50-контактні роз’єми, такі самі, як і плати Mesa. Будь-які релейні стійки/роз’ємні плати, що працюють з платами Mesa, повинні працювати з цією платою, з урахуванням того, що будь-які лічильники енкодера, PWM тощо повинні бути реалізовані в програмному забезпеченні. AC5 не має на борту жодної «розумної» логіки, це просто адаптер.

Більше інформації дивіться на веб-сайті виробника:

https://www.opto22.com/site/pr_details.aspx?cid=4&item=PCI-AC5

Я хотів би подякувати Opto22 за публікацію інформації в їхньому посібнику, що полегшило написання цього драйвера!

6.11.2 Драйвер

Цей драйвер призначений для карти PCI AC5 і не працює з картою ISA AC5. Драйвер HAL є модулем реального часу. Він підтримує 4 карти в поточній конфігурації (з можливістю підтримки більшої кількості карт після внесення змін до вихідного коду). Завантажте базовий драйвер таким чином:

```
loadrt opto_ac5
```

Це завантажить драйвер, який буде шукати максимум 4 плати. Він встановить вхід/вихід 2 портів кожної плати на стандартні налаштування. Стандартна конфігурація передбачає 12 входів і 12 виходів. Номери виводів відповідають позиції на релейній стійці. Наприклад, імена виводів для стандартних налаштувань входу/виходу порту 0 будуть такими:

- *opto_ac5.0.port0.in-00* - Вони будуть пронумеровані від 00 до 11
- *opto_ac5.0.port0.out-12* - Вони будуть пронумеровані від 12 до 23, порт 1 буде однаковим.

6.11.3 Піни

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER] OUT bit* -
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER]-not OUT bit* - Підключіть бітовий сигнал HAL до цього виводу, щоб зчитати точку вводу/виводу з плати. PINNUMBER позначає положення в релейній стійці. Наприклад, PINNUMBER 0 — це позиція 0 в релейній стійці Opto22 і буде контактом 47 на 50-контактному роз’ємі. Контакт -not інвертований, тому LOW дає TRUE, а HIGH дає FALSE.
- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER] IN bit* - Підключіть бітовий сигнал HAL до цього виводу, щоб записати дані в точку вводу-виводу карти. PINNUMBER позначає положення в релейній стійці. Наприклад, PINNUMBER 23 — це положення 23 в релейній стійці Opto22 і буде контактом 1 на 50-контактному роз’ємі.
- *opto_ac5.[BOARDNUMBER].led[NUMBER] OUT bit* - Вмикає/вимикає один із 4 вбудованих світлодіодів Світлодіоди пронумеровані від 0 до 3.

НОМЕР ПЛАТИ може бути 0-3, НОМЕР ПОРТІ може бути 0 або 1. Порт 0 розташований найближче до кронштейна плати.

6.11.4 Параметри

- *opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER]-invert W bit* - Якщо TRUE, інвертувати значення відповідного виводу -out так, щоб TRUE давало низький рівень, а FALSE - високий.

6.11.5 ФУНКЦІЇ

- *opto_ac5.0.digital-read* - Додайте це до теми, щоб прочитати всі вхідні дані.
- *opto_ac5.0.digital-write* - Додайте це до потоку, щоб записати всі вихідні точки та світлодіоди.

Наприклад, назви контактів для налаштувань вводу/виводу порту 0 за замовчуванням будуть такими:

```
opto_ac5.0.port0.in-00
```

Вони будуть пронумеровані від 00 до 11

```
opto_ac5.0.port0.out-12
```

Вони будуть пронумеровані від 12 до 23, порт 1 буде однаковим.

6.11.6 Налаштування портів вводу/виводу

Щоб змінити налаштування за замовчуванням, завантажте драйвер приблизно так:

```
loadrt opto_ac5 portconfig0=0xffff portconfig1=0xff0000
```

Звичайно, змінюючи номери відповідно до потрібних вам вводів/виводів. Кожен порт можна налаштувати по-різному.

Ось як визначити це число: номер конфігурації представляє собою 32-бітний код, який вказує карті, які точки вводу-виводу є вихідними, а які вхідними. Нижні 24 біти є точками вводу-виводу одного порту. Два найвищі біти призначені для двох вбудованих світлодіодів. Один біт у будь-якій позиції робить точку вводу-виводу вихідною. Два найвищі біти повинні бути вихідними, щоб світлодіоди працювали. Драйвер автоматично встановить два найвищі біти за вас, ми не будемо про них говорити.

Найпростіший спосіб зробити це - запустити калькулятор в розділі ПРОГРАМИ/АКСЕСУАРИ. Встановіть його в науковий режим (натисніть «Вид»). Встановіть його в ДВОЙКОВИЙ режим (перемикач «Bin»). Натисніть 1 для кожного виходу, який ви хочете, і/або 0 для кожного входу. Пам’ятайте, що контакт HAL 00 відповідає крайньому правому біту. 24 цифри представляють 24 точки вводу/виводу одного порту. Отже, для стандартних налаштувань (12 входів і 12 виходів) ви повинні натиснути 1 дванадцять разів (це виходи), а потім 0 дванадцять разів (це входи). Зверніть увагу, що перша точка вводу/виводу є найнижчим (крайнім правим) бітом. (цей біт відповідає виводу HAL 00 .дивиться назад) На екрані має бути 24 цифри. Тепер натисніть кнопку Hex. Відображене число (fff000) є номером конфігураційного порту (поставте перед ним *0x*, щоб позначити його як число HEX).

Інший приклад: встановити порт для 8 виходів та 16 входів (те саме, що й у карти Mesa). Ось 24 біти, представлені у двійковому числі. Біт 1 — це крайнє праве число:

16 нулів для 16 входів та 8 одиниць для 8 виходів

```
000000000000000011111111
```

Це перетворюється на FF на калькуляторі, тому 0xff – це число, яке слід використовувати для portconfig0 та/або portconfig1 під час завантаження драйвера.

6.11.7 Нумерація контактів

Контакт HAL 00 відповідає біту 1 (крайньому правому), який представляє позицію 0 на релейній стійці Opto22. Контакт HAL 01 відповідає біту 2 (на одне місце ліворуч від крайнього правого), який представляє позицію 1 на релейній стійці Opto22. Контакт HAL 23 відповідає біту 24 (крайньому лівому), який представляє позицію 23 на релейній стійці Opto22.

Контакт HAL 00 підключається до контакту 47 на 50-контактному роз’ємі кожного порту. Контакт HAL 01 підключається до контакту 45 на 50-контактному роз’ємі кожного порту. Контакт HAL 23 підключається до контакту 1 на 50-контактному роз’ємі кожного порту.

Зверніть увагу, що Opto22 і Mesa використовують протилежні системи нумерації: позиція 23 Opto22 = контакт 1 роз’єму, і позиція зменшується в міру збільшення номера контакту роз’єму. Позиція 1 Mesa Hostmot2 = контакт 1 роз’єму, і номер позиції збільшується в міру збільшення номера контакту роз’єму.

6.12 Драйвери Pico

Pico Systems має серію плат для аналогового сервоприводу, крокового двигуна та PWM (цифрового) сервоприводу. Плати підключаються до ПК через паралельний порт, що працює в режимі EPP. Хоча більшість користувачів підключають одну плату до паралельного порту, теоретично на одному паралельному порту можна використовувати будь-яку комбінацію до 8 або 16 плат. Один драйвер обслуговує всі типи плат. Остаточна комбінація входів/виходів залежить від підключених плат. Драйвер не розрізняє плати, а просто нумерує канали вводу/виводу (енкодери тощо), починаючи з 0 на першій платі. Драйвер має назву hal_ppmc.ko Аналоговий сервоінтерфейс також називається PPMC (Parallel Port Motion Control, паралельний порт управління рухом). Існує також універсальний кроковий контролер, скорочено USC. І універсальний PWM-контролер, або UPC.

Встановлення:

```
loadrt hal_ppmc port_addr=<addr1>[,<addr2>[,<addr3>...]]
```

Параметр «port_addr» повідомляє драйверу, які паралельні порти перевіряти. За замовчуванням «<addr1>» дорівнює 0x0378, а «<addr2>» і наступні не використовуються. Драйвер шукає в усьому адресному просторі розширених паралельних портів за адресою «port_addr» будь-які плати сімейства PPMC. Потім він експортує виводи HAL для всіх знайдених пристроїв. Під час завантаження (або спроби завантаження) драйвер виводить у журнал ядра корисні повідомлення для налагодження, які можна переглянути за допомогою команди «dmesg».

Можна використовувати до 3 шин Parkport, і кожна шина може мати до 8 (або, можливо, 16 пристроїв PPMC).

6.12.1 Параметри командного рядка

У командному рядку loadrt можна вказати кілька опцій. По-перше, USC і UPC можуть мати 8-бітний DAC, доданий для регулювання швидкості шпинделя та подібних функцій. Це можна вказати за допомогою параметра extradac=*0xnn[,0xmm]*. Частина, укладена в [ ], дозволяє вказати цю опцію на більш ніж одній платі системи. Перша шестнадцяткова цифра вказує, до якої шини EPP звертаються, вона відповідає порядку адрес портів у параметрі port_addr, де <addr1> тут буде нулем. Отже, для першої шини EPP перша плата USC або UPC буде описана як *0x00*, друга плата USC або UPC на тій самій шині буде *0x02*. (Зверніть увагу, що кожна плата USC або UPC займає дві адреси, тому якщо одна знаходиться на 00, наступна повинна бути 02.)

Крім того, 8 цифрових вивідних контактів можна використовувати як додаткові цифрові виходи, це працює так само, як описано вище, із синтаксисом: extradout=0xnn’. Опції extradac та extradout є взаємовиключними на кожній платі, можна вказати лише одну з них.

Плати кодерів UPC і PPMC можуть ставити мітку часу на прибуття лічильників кодера, щоб уточнити виведення швидкості осі. Ця виведена швидкість може бути передана до компонента PID hal для отримання більш плавного відгуку терміна D. Синтаксис: timestamp=*0xnn[,0xmm]*, це працює так само, як і вище, для вибору плати, яка конфігурується. За замовчуванням опція timestamp не ввімкнена. Якщо ви вкажете цю опцію в командному рядку, вона ввімкнеться. Перше «n» вибирає шину EPP, друге відповідає адресі плати, на якій ввімкнено цю опцію. Драйвер перевіряє рівень ревізії плати, щоб переконатися, що вона має прошивку, яка підтримує цю функцію, і видає повідомлення про помилку, якщо плата її не підтримує.

Плата енкодера PPMC має опцію вибору частоти цифрового фільтра енкодера. (UPC має таку ж можливість за допомогою DIP-перемикачів на платі.) Оскільки плата енкодера PPMC не має цих додаткових DIP-перемикачів, вибір потрібно здійснювати за допомогою опції командного рядка. За замовчуванням фільтр працює на частоті 1 МГц, що дозволяє лічильникам лічильників до приблизно 900 кГц (залежно від шуму і квадратурної точності лічильника). Опції: 1, 2,5, 5 і 10 МГц. Вони встановлюються за допомогою параметра 1,2,5 і 10 (десятичний), який вказується як шістнадцятковий символ «A». Вони вказуються аналогічно до наведених вище опцій, але з

налаштуванням частоти ліворуч від цифр шини/адреси. Отже, щоб встановити 5 МГц на платі енкодера за адресою 3 на першій шині EPP, потрібно написати: `enc_clock='0x503'`.

Нещодавно було виявлено, що деякі мікросхеми паралельного порту не працюють з драйвером ppmc. Зокрема, ця проблема стосувалася мікросхеми Oxford OXPCIe952 на картах паралельного порту SIIG PCIe. Драйвер ppmc у всіх версіях LinuxCNC, починаючи з 2.7.8, був виправлений для цієї проблеми за замовчуванням. Однак це може спричинити проблеми з дуже старим апаратним забезпеченням паралельного порту EPP, тому існує опція командного рядка, яка дозволяє повернутися до попередньої поведінки. Нова поведінка встановлюється за замовчуванням або шляхом додавання параметра epp_dir=0 у командний рядок. Щоб отримати стару поведінку, додайте epp_dir=1 до командного рядка. Всі паралельні порти, які я маю тут, працюють з новою поведінкою за замовчуванням. Як і для інших параметрів, можна вказати список, наприклад epp_dir=1,0,1, щоб встановити різні налаштування для кожного з 3 паралельних портів.

6.12.2 Піни

У наступних контактах, параметрах і функціях <порт> є ідентифікатором паралельного порту. Згідно з правилами іменування перший порт завжди повинен мати ідентифікатор нуль. Усі плати мають певний спосіб налаштування адреси на шині EPP. USC і UPC мають прості можливості для лише двох адрес, але за допомогою перемичок можна налаштувати адреси для 4 плат. Плати PPMC мають 16 можливих адрес. У всіх випадках драйвер перераховує плати за типом і експортує відповідні контакти HAL. Наприклад, кодери будуть перераховані від нуля вгору, в тому ж порядку, що і перемикачі адрес на платі. Отже, перша плата матиме кодери 0 - 3, друга плата матиме кодери 4 - 7. Перший стовпець після маркера вказує, які плати матимуть цей контакт HAL або параметр, пов'язаний з ним. «Все» означає, що цей контакт доступний на всіх трьох типах плат. «Опція» означає, що цей контакт буде експортований тільки тоді, коли ця опція буде ввімкнена опціональним параметром у команді loadrt HAL. Ці опції вимагають, щоб плата мала достатній рівень ревізії для підтримки цієї функції.

- *(All s32 output) ppmc.<port>.encoder.<channel>.count* - Положення енкодера, в одиницях.
- *(All s32 output) ppmc.<port>.encoder.<channel>.delta* - Зміна кількості підрахунків з моменту останнього зчитування, в одиницях підрахунку необроблених даних кодера.
- «(Всі вихідні дані типу float) ppmc.<порт>.encoder.<канал>.velocity» — швидкість, масштабована в одиницях користувача за секунду. На PPMC і USC вона виводиться з необроблених значень енкодера за сервоперіод, тому на неї впливає гранулярність енкодера. На платах UPC з прошивкою 8/21/09 і пізнішою версією для поліпшення плавності виведення швидкості можна використовувати оцінку швидкості за допомогою часових міток підрахунків енкодера. Цю інформацію можна передати до компонента PID HAL для отримання більш стабільної реакції сервоприводу. Цю функцію потрібно ввімкнути в командному рядку HAL, що запускає драйвер PPMC, за допомогою опції timestamp=0x00.
- *(All float output) ppmc.<port>.encoder.<channel>.position* - Положення енкодера, в одиницях користувача.
- *(All bit bidir) ppmc.<port>.encoder.<channel>.index-enable* - Підключіться до joint.#.index-enable для home-to-index. Це двонаправлений сигнал HAL. Встановлення значення true змушує апаратне забезпечення енкодера скинути лічильник до нуля при наступному імпульсі індексу енкодера. Драйвер виявить це і поверне сигнал до значення false.
- *(Вихід PPMC з плаваючою комою) ppmc.<порт>.DAC.<канал>.значення* – надсилає знакове значення до 16-бітного цифро-аналогового перетворювача на платі PPMC DAC16, керуючи аналоговою вихідною напругою цього каналу DAC.
- *(UPC bit input) ppmc.<port>.pwm.<channel>.enable* - Вмикає PWM-генератор.

- «(UPC float input) ppmc.<port>.pwm.<channel>.value» — значення, яке визначає робочий цикл імпульсних сигналів PWM. Значення ділиться на «pwm.<channel>.scale», і якщо результат дорівнює 0,6, робочий цикл буде 60 % і так далі. Від'ємні значення призводять до того, що робочий цикл базується на абсолютному значенні, а напрямок виводу встановлюється на від'ємний.
- *(USC bit input) ppmc.<port>.stepgen.<channel>.enable* - Вмикає генератор крокових імпульсів.
- *(USC float input) ppmc.<port>.stepgen.<channel>.velocity* - Значення, яке визначає частоту кроку. Значення множиться на *stepgen.<channel>.scale* , і результатом є частота в кроках за секунду. Негативні значення призводять до того, що частота базується на абсолютному значенні, а напрямок штифта встановлюється на негативний.
- *(All bit output) ppmc.<port>.din.<channel>.in* - Стан цифрового вхідного виводу, див. канонічний цифровий вхід.
- *(All bit output) ppmc.<port>.din.<channel>.in-not* - Інвертований стан цифрового вхідного виводу, див. канонічний цифровий вхід.
- *(All bit input) ppmc.<port>.dout.<channel>.out* - Значення, яке буде записано на цифровий вихід, див. канонічний цифровий вихід.
- *(Опція введення з плаваючою точкою) ppmc.<порт>.DAC8-<канал>.значення* - Значення, яке буде записано в аналоговий вихід, діапазон від 0 до 255. Це відправляє 8 бітів виходу на J8, до якого повинна бути підключена плата Spindle DAC. 0 відповідає нулю вольт, 255 відповідає 10 вольтам. Полярність виходу можна встановити на завжди мінус, завжди плюс або можна керувати станом SSR1 (плюс, коли увімкнено) та SSR2 (мінус, коли увімкнено). Ви повинні вказати extradac = 0x00 у командному рядку HAL, який завантажує драйвер PPMC, щоб увімкнути цю функцію на першій платі USC або UPC.
- «(Вхід опційного біта) ppmc.<порт>.dout.<канал>.out» — значення, яке потрібно записати в один із 8 додаткових цифрових вивідних контактів на J8. Ви повинні вказати extradout = 0x00 у командному рядку HAL, який завантажує драйвер ppmc, щоб увімкнути цю функцію на першій платі USC або UPC. extradac і extradout є взаємовиключними функціями, оскільки вони використовують ті самі сигнальні лінії для різних цілей. Ці виходи будуть перераховані після стандартних цифрових виходів плати.

6.12.3 Параметри

- *(All float) ppmc.<port>.encoder.<channel>.scale* - Кількість підрахунків / одиниця користувача (для перетворення з підрахунків в одиниці).
- *(UPC float) ppmc.<port>.pwm.<channel-range>.freq* - Носійна частота PWM, в Гц. Застосовується до групи з чотирьох послідовних генераторів ШІМ, як вказано в *<channel-range>*. Мінімальне значення - 610 Гц, максимальне - 500 кГц.
- *(PPMC float) ppmc.<порт>.DAC.<канал>.scale* - Встановлює масштаб вихідного каналу DAC16 таким чином, що вихідне значення, рівне 1/масштабу, буде генерувати вихідне значення + або - вольт. Отже, якщо параметр масштабу дорівнює 0,1, а ви надсилаєте значення 0,5, вихідне значення буде 5,0 вольт.
- *(UPC float) ppmc.<port>.pwm.<channel>.scale* - Масштабування для PWM-генератора. Якщо «масштаб» дорівнює X, то шпаруватість становитиме 100%, коли контакт «значення» має значення X (або -X).
- *(UPC float) ppmc.<port>.pwm.<channel>.max-dc* - Максимальний робочий цикл, від 0,0 до 1,0.
- *(UPC float) ppmc.<port>.pwm.<channel>.min-dc* - Мінімальний робочий цикл, від 0,0 до 1,0.
- *(UPC float) ppmc.<port>.pwm.<channel>.duty-cycle* - Фактичний робочий цикл (використовується здебільшого для усунення несправностей)

- *(UPC bit) ppmc.<port>.pwm.<channel>.bootstrap* - Якщо значення «true» (істина), PWM-генератор генеруватиме коротку послідовність імпульсів обох полярностей, коли аварійна зупинка стане хибною, щоб скинути фіксатори вимкнення на деяких PWM-сервоприводах.
- *(USC u32) ppmc.<порт>.stepgen.<діапазон каналів>.час налаштування* - Встановлює мінімальний час між зміною напрямку та імпульсом кроку в одиницях 100 нс. Застосовується до групи з чотирьох послідовних генераторів кроків, як зазначено в *<діапазон каналів>*. Можна вказати значення від 200 нс до 25,5 мкс.
- *(USC u32) ppmc.<порт>.stepgen.<діапазон каналів>.ширина імпульсу* - Встановлює ширину імпульсів кроку в одиницях 100 нс. Застосовується до групи з чотирьох послідовних генераторів кроку, як зазначено в *<діапазон каналів>*. Можна вказати значення від 200 нс до 25,5 мкс.
- *(USC u32) ppmc.<порт>.stepgen.<діапазон каналів>.pulse-space-min* - Встановлює мінімальний час між імпульсами в одиницях 100 нс. Застосовується до групи з чотирьох послідовних генераторів імпульсів, як зазначено в *<діапазон каналів>*. Можна вказати значення від 200 нс до 25,5 мкс.

  Максимальна частота кроків становить: $\frac{1}{100\text{ns} * (pulsewidth + pulsespacemin)}$
- *(USC float) ppmc.<port>.stepgen.<channel>.scale* - Масштабування для генератора крокових імпульсів. Частота кроку в Гц – це абсолютне значення «швидкості» * «масштабу».
- *(USC float) ppmc.<port>.stepgen.<channel>.max-vel* - Максимальне значення для «швидкості». Команди, більші за «max-vel», будуть фіксовані. Також застосовується до від'ємних значень. (Абсолютне значення фіксується.)
- *(USC float) ppmc.<port>.stepgen.<channel>.frequency* - Фактична частота крокових імпульсів у Гц (використовується здебільшого для усунення несправностей)
- *(Опція float) ppmc.<порт>.DAC8.<канал>.scale* - Встановлює масштаб додаткового виходу DAC таким чином, що вихідне значення, рівне масштабу, дає вихідну величину 10,0 В. (Знак виходу встановлюється за допомогою перемичок та/або інших цифрових виходів.)
- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Інвертує цифровий вихід, див. канонічний цифровий вихід.
- *(Option bit) ppmc.<port>.dout.<channel>.invert* - Інвертує цифровий вихідний контакт J8, див. канонічний цифровий вихід.

6.12.4 Функції

- *(All funct) ppmc.<port>.read* - Зчитує всі входи (цифрові входи та лічильники енкодера) на одному порту. Ці зчитування організовані в блоки суміжних регістрів для зчитування в блоці, щоб мінімізувати накладні витрати процесора.
- *(All funct) ppmc.<port>.write* - Записує всі виходи (цифрові виходи, ступінчасті генератори, PWM) на один порт. Ці записи організовані в блоки суміжних регістрів для запису в блок, щоб мінімізувати накладні витрати процесора.

6.13 Плутон P Драйвер

6.13.1 Загальна інформація

Pluto-P — це плата FPGA з мікросхемою ACEX1K від Altera.

6.13.1.1 Вимоги

1. Дошка Плуто-П
2. Паралельний порт, сумісний з EPP, налаштований на режим EPP у системному BIOS, або плата паралельного порту PCI, сумісна з EPP.

Note
Плата Pluto P вимагає режиму EPP. Мікросхеми Netmos98xx не працюють у режимі EPP. Плата Pluto P працюватиме на деяких комп'ютерах, а на інших — ні. Невідомо, які саме комп'ютери працюватимуть, а які — ні.

Для отримання додаткової інформації про карти паралельного порту, сумісні з PCI EPP, див. сторінку Підтримуване обладнання LinuxCNC на вікі.

6.13.1.2 Роз'єми

- Плата Pluto-P поставляється з попередньо припаяним лівим роз'ємом, ключ знаходиться у вказаному положенні. Інші роз'єми не заповнені. Стандартного 12-контактного роз'єму IDC, схоже, немає, але деякі контакти 16-контактного роз'єму можуть висіти з плати поруч з QA3/QZ3.
- Нижній і правий роз'єми розташовані на одній сітці 0,1 дюйма, але лівий роз'єм - ні. Якщо OUT2...OUT9 не потрібні, один роз'єм IDC може охоплювати нижній роз'єм і два нижні ряди правого роз'єму.

6.13.1.3 Фізичні піни

- Ознайомтеся з технічним описом ACEX1K для отримання інформації про порогові значення вхідної та вихідної напруги. Усі контакти налаштовані в режимі «LVTTL/LVCMOS» і загалом сумісні з логікою 5V TTL.
- Перед конфігурацією та після правильного виходу з LinuxCNC всі контакти Pluto-P перебувають у тристабільному стані зі слабкими підтягуваннями (мінімум 20 кОм, максимум 50 кОм). Якщо таймер сторожового пристрою ввімкнено (за замовчуванням), ці контакти також перебувають у тристабільному стані після переривання зв'язку між LinuxCNC і платою. Таймер сторожового пристрою активується приблизно за 6,5 мс. Однак програмні помилки в прошивці pluto_servo або LinuxCNC можуть залишити контакти Pluto-P в невизначеному стані.
- У режимі pwm+dir за замовчуванням dir має значення HIGH для від'ємних значень і LOW для додатних значень. Щоб вибрати HIGH для додатних значень і LOW для від'ємних значень, встановіть відповідний параметр dout-NN-invert на TRUE, щоб інвертувати сигнал.
- Вхід індексу спрацьовує на передньому фронті. Початкові випробування показали, що входи QZx особливо чутливі до шуму, оскільки опитуються кожні 25 нс. Було додано цифрове фільтрування для фільтрування імпульсів, коротших за 175 нс (сім разів опитування). Рекомендується додаткове зовнішнє фільтрування на всіх вхідних контактах, таке як буфер Шмітта або інвертор, RC-фільтр або диференційний приймач (якщо це можливо).
- Контакти IN1...IN7 мають послідовні резистори 22 Ом, підключені до відповідних контактів FPGA. Жодні інші контакти не мають захисту від напруги або струму, що виходять за межі технічних характеристик. Інтегратор повинен додати відповідну ізоляцію та захист. Традиційні плати оптоізоляторів паралельного порту не працюють з pluto_servo через двонаправлений характер протоколу EPP.

6.13.1.4 LED

- Коли пристрій не запрограмований, світлодіод світиться слабко. Коли пристрій запрограмований, світлодіод світиться відповідно до робочого циклу PWM0 («LED» = «UP0» «xor» «DOWN0») або STEPGEN0 («LED» = «STEP0» «xor» «DIR0»).

6.13.1.5 Потужність

- Невелика кількість струму може бути взята з VCC. Доступний струм залежить від нерегульованого вхідного постійного струму на платі. Альтернативно, регульований +3,3 В постійного струму може бути поданий на FPGA через ці контакти VCC. Необхідний струм ще не відомий, але, ймовірно, становить близько 50 мА плюс струм вводу-виводу.
- Регулятор на платі Pluto-P має низький рівень падіння напруги. Подача 5 В на роз'єм живлення дозволить регулятору працювати належним чином.

6.13.1.6 Інтерфейс ПК

- Підтримується лише одна плата pluto_servo або pluto_step.

6.13.1.7 Перезбірка прошивки FPGA

Підкаталоги «src/hal/drivers/pluto_servo_firmware/» та «src/hal/drivers/pluto_step_firmware/» містять вихідний код Verilog та додаткові файли, які використовуються Quartus для прошивок FPGA. Для перекомпіляції прошивки FPGA необхідне програмне забезпечення Altera Quartus II. Щоб перекомпілювати прошивку з файлів .hdl та інших вихідних файлів, відкрийте файл «.qpf» і натисніть CTRL-L. Потім перекомпілюйте LinuxCNC.

Як і драйвер обладнання HAL, прошивка FPGA ліцензована відповідно до умов Загальної публічної ліцензії GNU.

Безкоштовна версія Quartus II працює лише на Microsoft Windows, хоча, очевидно, існує платна версія, яка працює на Linux.

6.13.1.8 Для отримання додаткової інформації

Деяка додаткова інформація про це доступна за посиланнями KNJC LLC та блог розробника.

6.13.2 Сервопривід Плутона

Система pluto_servo підходить для управління 4-осьовим фрезерним верстатом з CNC з сервомоторами, 3-осьовим фрезерним верстатом з PWM-управлінням шпинделем, токарним верстатом з енкодером шпинделя тощо. Велика кількість входів дозволяє використовувати повний набір кінцевих вимикачів.

Цей драйвер має такі характеристики:

- 4 канали квадратури з частотою дискретизації 40 МГц. Лічильники працюють у режимі «4x». Максимальна корисна частота квадратури становить 8191 імпульсів на сервоцикл LinuxCNC або приблизно 8 МГц для стандартної частоти сервоприводу LinuxCNC 1 мс.
- 4 канали PWM, типу «вгору/вниз» або «pwm+напрямок». 4095 робочих циклів від -100% до +100%, включаючи 0%. Період PWM становить приблизно 19,5 кГц (40 МГц / 2047). Також доступний режим, подібний до PDM.

- 18 цифрових виходів: 10 виділених, 8 спільних з функціями PWM. (Приклад: Токарний верстат з односпрямованим PWM-керуванням шпинделем може використовувати загалом 13 цифрових виходів)
- 20 цифрових входів: 8 виділених, 12 спільних з функціями квадратурної корекції. (Приклад: Токарний верстат з індексними імпульсами лише на шпинделі може використовувати загалом 13 цифрових входів.)
- Зв’язок EPP з ПК. Зв’язок EPP зазвичай триває близько 100 мкс на машинах, протестованих досі, що забезпечує швидкість сервоприводу вище 1 кГц.

6.13.2.1 Розпіновка

- «UPx» — сигнал «up» (режим вгору/вниз) або «pwm» (режим pwm+напрямок) від генератора PWM X. Може використовуватися як цифровий вихід, якщо відповідний канал PWM не використовує або вихід на каналі завжди негативний. Відповідний інвертований цифровий вихід можна встановити на TRUE, щоб UPx був активним низьким, а не активним високим.
- «DNx» — сигнал «вниз» (режим вгору/вниз) або «напрямок» (режим pwm+напрямок) від генератора PWM X. Може використовуватися як цифровий вихід, якщо відповідний канал PWM не використовує або вихід на каналі ніколи не є від’ємним. Відповідний інвертований цифровий вихід можна встановити на TRUE, щоб DNx був активним низьким, а не активним високим.
- «QAx, QBx» – сигнали A та B для квадратурного лічильника X. Може використовуватися як цифровий вхід, якщо відповідний квадратурний канал не використовується.
- *QZx* – Сигнал Z (індекс) для квадратурного лічильника X. Може використовуватися як цифровий вхід, якщо функція індексу відповідного квадратурного каналу не використовується.
- «INx» – виділений цифровий вхід №x
- «OUTx» – виділений цифровий вихід #x
- «GND» – Заземлення
- *VCC* - +3,3 В регульованого постійного струму

.Розпіновка сервоприводу Плуто Розпіновка сервоприводу Плуто

Table 6.41: Функції альтернативних контактів Плуто-Серво

Основна функція	**Альтернативна функція**	**Поведінка, якщо використовуються обидві функції**
UP0	PWM0	Коли pwm-0-pwmdir має значення TRUE, цей контакт є виходом PWM
	OUT10	Виключаємо АБО з UP0 або PWM0
UP1	PWM1	Коли pwm-1-pwmdir має значення TRUE, цей контакт є виходом PWM
	OUT12	Виключає АБО з UP1 або PWM1
UP2	PWM2	Коли pwm-2-pwmdir має значення TRUE, цей контакт є виходом PWM

Table 6.41: (continued)

Основна функція	**Альтернативна функція**	**Поведінка, якщо використовуються обидві функції**
	OUT14	XOR з UP2 або PWM2
UP3	PWM3	Коли pwm-3-pwmdir має значення TRUE, цей контакт є виходом PWM
	OUT16	XOR з UP3 або PWM3
DN0	DIR0	Коли pwm-0-pwmdir має значення TRUE, цей контакт є виходом DIR
	OUT11	Вирівнювання з DN0 або DIR0 за допомогою операції XOR
DN1	DIR1	Коли pwm-1-pwmdir має значення TRUE, цей контакт є виходом DIR
	OUT13	Використано XOR з DN1 або DIR1
DN2	DIR2	Коли pwm-2-pwmdir має значення TRUE, цей контакт є виходом DIR
	OUT15	Вирівнювання з DN2 або DIR2 за допомогою операції XOR
DN3	DIR3	Коли pwm-3-pwmdir має значення TRUE, цей контакт є виходом DIR
	OUT17	Використано XOR з DN3 або DIR3
QZ0	IN8	Зчитати те саме значення
QZ1	IN9	Зчитати те саме значення
QZ2	IN10	Зчитати те саме значення
QZ3	IN11	Зчитати те саме значення
QA0	IN12	Зчитати те саме значення
QA1	IN13	Зчитати те саме значення
QA2	IN14	Зчитати те саме значення
QA3	IN15	Зчитати те саме значення
QB0	IN16	Зчитати те саме значення
QB1	IN17	Зчитати те саме значення
QB2	IN18	Зчитати те саме значення
QB3	IN19	Зчитати те саме значення

6.13.2.2 Фіксація входу та оновлення виходу

- PWM Робочі цикли для кожного каналу оновлюються в різний час.
- Цифрові виходи OUT0–OUT9 оновлюються одночасно. Цифрові виходи OUT10–OUT17 оновлюються одночасно з функцією PWM, з якою вони спільно працюють.
- Цифрові входи IN0 - IN19 фіксуються одночасно.
- Квадратурні положення для кожного каналу фіксуються в різний час.

6.13.2.3 Функції, виводи та параметри HAL

Список усіх аргументів «loadrt», назв функцій HAL, назв виводів та назв параметрів знаходиться на сторінці довідки «pluto_servo.9».

6.13.2.4 Сумісне обладнання драйвера

Схема для 2-осьової плати сервопідсилювача PWM 2A доступна на сайті (розробник програмного забезпечення). H-міст L298 може використовуватися для двигунів до 4 A (один двигун на L298) або до 2 A (два двигуни на L298) з напругою живлення до 46 В. Однак L298 не має вбудованого обмеження струму, що є проблемою для двигунів з високим струмом зупинки. Для більш високих струмів і напруг деякі користувачі повідомляють про успішне використання інтегрованих драйверів високого/низького рівня від International Rectifier.

6.13.3 Крок Плутона

Pluto-step підходить для керування 3- або 4-осьовим фрезерним верстатом з ЧПК з кроковими двигунами. Велика кількість входів дозволяє використовувати повний комплект кінцевих вимикачів.

Дошка має такі характеристики:

- 4 канали «крок+напрямок» з максимальною частотою кроку 312,5 кГц, програмованою довжиною кроку, інтервалом та часом зміни напрямку
- 14 спеціалізованих цифрових виходів
- 16 спеціальних цифрових входів
- Зв’язок EPP з ПК

6.13.3.1 Розпіновка

- *STEPx* – Вихідний сигнал *кроку* (тактової частоти) каналу stepgen *x*
- *DIRx* – Вихідний сигнал «напрямок» каналу ступінчастого генератора *x*
- «INx» – виділений цифровий вхід №x
- «OUTx» – виділений цифровий вихід #x
- «GND» – Заземлення
- *VCC* - +3,3 В регульованого постійного струму

Хоча «розширений головний роз’єм» має надмножину сигналів, які зазвичай зустрічаються на роз’ємі Step & Direction DB25 — 4 генератори кроку, 9 входів і 6 виходів загального призначення — компонування цього роз’єму відрізняється від компонування стандартного 26-контактного плоского кабелю до роз’єму DB25.

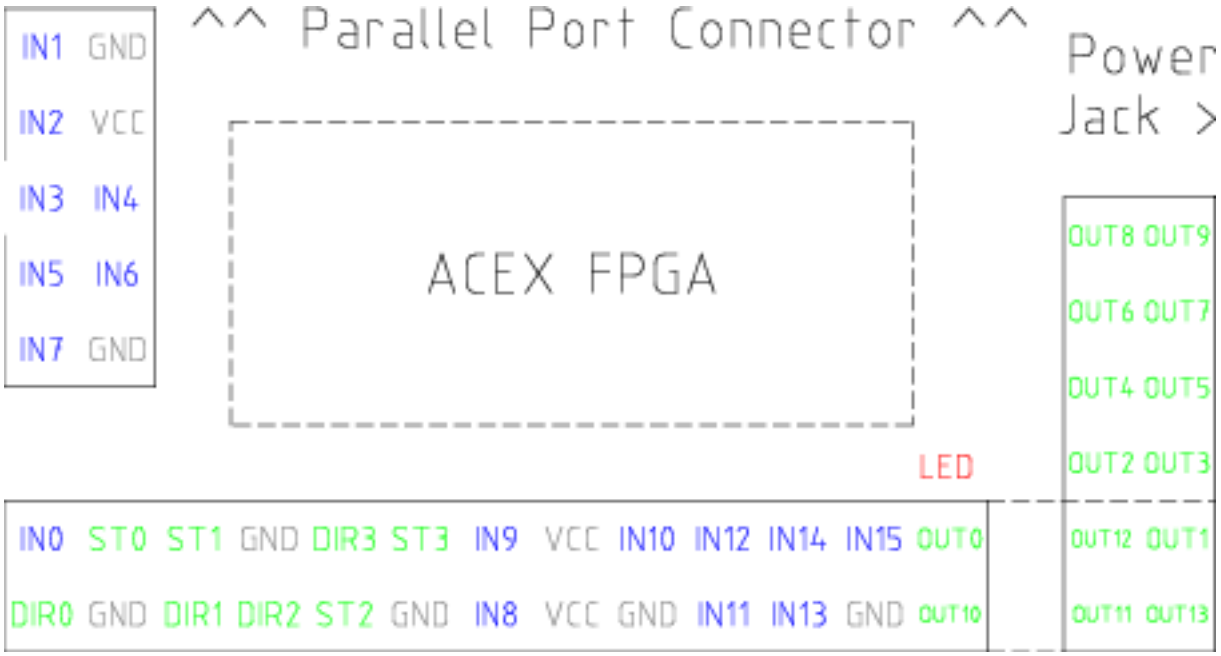


Figure 6.10: Розподілення клапанів Плуто-Ступінчастої Конфігурації

6.13.3.2 Фіксація входу та оновлення виходу

- Частоти кроків для кожного каналу оновлюються в різний час.
- Усі цифрові виходи оновлюються одночасно.
- Всі цифрові входи фіксуються одночасно.
- Положення зворотного зв’язку для кожного каналу фіксуються в різний час.

6.13.3.3 Таймінги ступінчастої форми хвилі

Прошивка та драйвер забезпечують дотримання довжини кроку, інтервалу та часу зміни напрямку. Часові параметри округлюються до найближчого кратного 1,6 мкс, максимум до 49,6 мкс. Часові параметри такі самі, як і для програмного компонента stepgen, за винятком того, що «dirhold» і «dirsetup» об’єднані в один параметр «dirtime», який повинен бути максимальним з двох, і що однакові часові параметри кроку завжди застосовуються до всіх каналів.

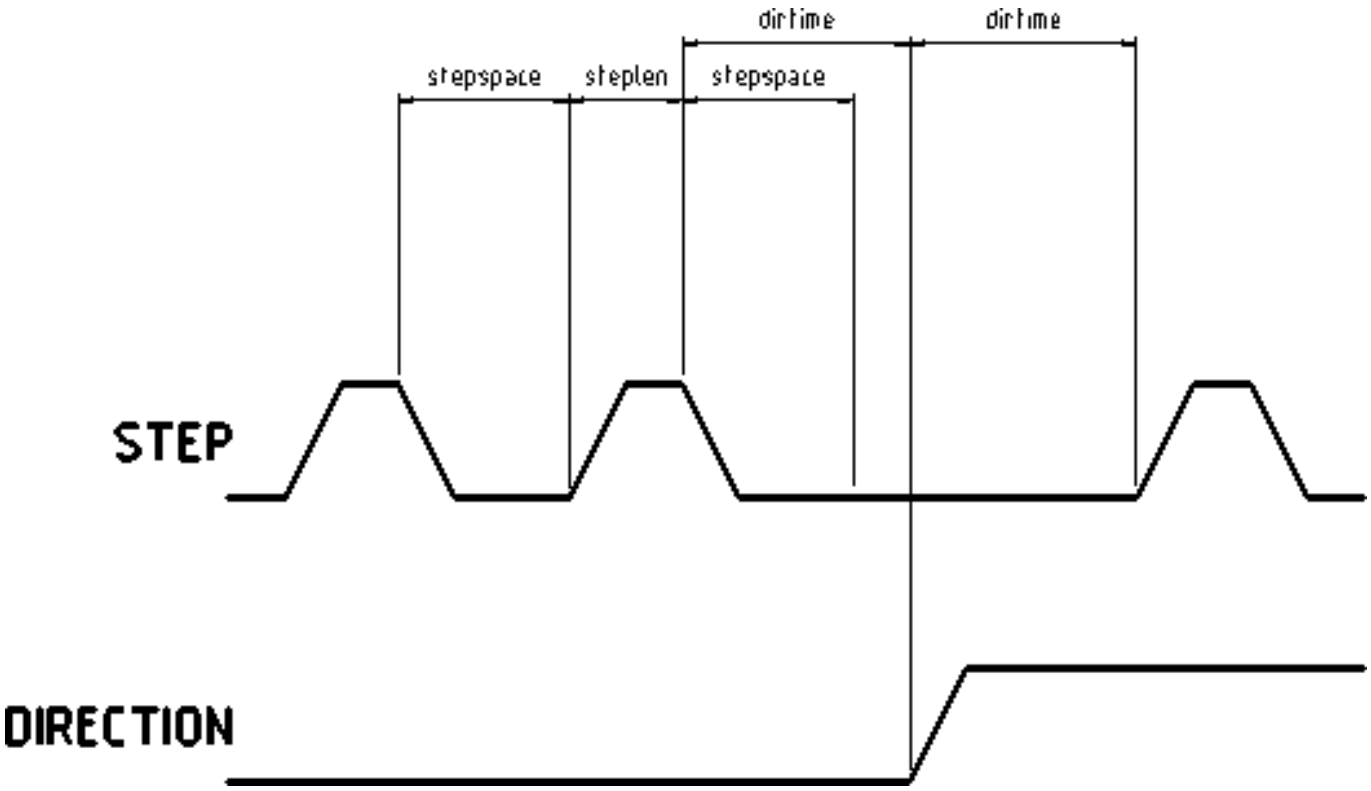


Figure 6.11: Таймінги Плуто-кроків

6.13.3.4 Функції, виводи та параметри HAL

Список усіх аргументів «loadrt», назв функцій HAL, назв виводів та назв параметрів наведено на сторінці довідки «pluto_step.9».

6.14 Драйвер Powermax Modbus

Це HAL-програма, написана на Python, яка не працює в реальному часі, для керування плазмовими різаками Hypetherm Powermax за допомогою протоколу Modbus ASCII через RS485.

Note
Оскільки це програма, що не працює в режимі реального часу, на її роботу можуть впливати завантаження комп’ютера та затримка. Можлива втрата зв’язку, про що свідчитиме зміна статусу виходу. Завжди слід мати схему Estop, яка відключає живлення пристрою в разі надзвичайної ситуації.

Цей компонент завантажується за допомогою команди halcmd ”loadusr”:

```
loadusr -Wn pmx485 pmx485 /dev/ttyUSB0
```

Це завантажить компонент pmx485, використовуючи порт /dev/ttyUSB0, та зачекає його готовності.

Необхідно назвати порт, який буде використовуватися для зв’язку.

6.14.1 Піни

- **pmx485.mode-set** (bit, in) # встановити режим різання
- **pmx485.current-set** (bit, in) # встановити струм різання
- **pmx485.pressure-set** (bit, in) # встановлений тиск газу
- **pmx485.enable** (bit, in) # увімкнути компонент
- **pmx485.mode** (bit, out) # зворотний зв’язок у режимі скорочення
- **pmx485.current** (bit, out) # зворотний зв’язок по струму скорочення
- **pmx485.pressure** (bit, out) # зворотний зв’язок по тиску газу
- **pmx485.fault** (bit, out) # Код помилки PowerMax
- **pmx485.status** (bit, out) # стан з’єднання
- **pmx485.current-min** (bit, out) # мінімально допустимий струм
- **pmx485.current-max** (bit, out) # максимально допустимий струм
- **pmx485.pressure-min** (bit, out) # мінімально допустимий тиск газу
- **pmx485.pressure-max** (bit, out) # максимально допустимий тиск газу

6.14.2 Опис

Щоб зв’язатися з Powermax, компонент спочатку потрібно ввімкнути через контакт **enable**, а потім він може ініціювати запит до Powermax, записавши дійсний рядок на такі контакти:

- **mode-set**
- **current-set**
- **pressure-set**

Note
Нульове значення **тиску** є дійсним, після чого Powermax розрахує необхідний тиск самостійно.

Зв’язок можна перевірити на дисплеї Powermax або за допомогою контакту **status**. У віддаленому режимі режим, струм і тиск можна змінювати за потреби.

Щоб припинити зв’язок, виконайте одну з наведених нижче дій:

- Встановіть усі контакти set у нульове значення: **mode-set**, **current-set** та **pressure-set**.
- Від’єднайте блок живлення Powermax від джерела живлення приблизно на 30 секунд. Після повторного ввімкнення системи вона більше не буде в режимі дистанційного керування.

6.14.3 Довідка:

- Примітка до застосування Hypertherm №807220
  ”Протокол послідовного зв’язку Powermax45 XP/65/85/105/125®”

6.15 Драйвер Servo To Go

Servo-To-Go (STG) — одна з перших карт управління рухом для ПК, що підтримується LinuxCNC. Це карта ISA, яка існує в різних варіантах (усі підтримуються цим драйвером). Плата включає до 8 каналів входу квадратурного енкодера, 8 каналів аналогового входу та виходу, 32-бітний цифровий вхід/вихід, інтервальний таймер з перериванням та сторожовий таймер.

Note
Ми отримали повідомлення про те, що операційні підсилювачі на платі Servo To Go не працюють з новими блоками живлення ATX, які використовують сучасні імпульсні перетворювачі постійного струму. Проблема полягає в тому, що плата STG видає постійну напругу незалежно від команд драйвера. Старі блоки живлення ATX з лінійними стабілізаторами напруги не мають цієї проблеми і нормально працюють з платами STG.

6.15.1 Встановлення

```
loadrt hal_stg [base=<address>] [num_chan=<nr>] [dio="<dio-string>"] \
                  [model=<model>]
```

Поле базової адреси є необов'язковим; якщо воно не вказане, драйвер намагається автоматично виявити плату. Поле num_chan використовується для вказання кількості каналів, доступних на карті; якщо воно не використовується, то передбачається 8-осьова версія. Конфігурація цифрових входів/виходів визначається конфігураційним рядком, що передається insmod під час завантаження модуля. Формат складається з чотирисимвольного рядка, що встановлює напрямок кожної групи контактів. Кожен символ рядка напрямку є або «I», або «O». Перший символ встановлює напрямок порту A (порт A - DIO.0-7), наступний - порту B (порт B - DIO.8-15), наступний - порту C (порт C - DIO.16-23), а четвертий - порту D (порт D - DIO.24-31). Поле моделі можна використовувати, якщо драйвер не визначає автоматично правильну версію карти.

ПІДКАЗКА: після запуску драйвера можна звернутися до *dmesg*, щоб отримати повідомлення, що стосуються драйвера (наприклад, автоматично визначений номер версії та базова адреса). Наприклад:

```
loadrt hal_stg base=0x300 num_chan=4 dio="IOIO"
```

У цьому прикладі встановлюється драйвер STG для карти, яка знаходиться за базовою адресою 0x300, 4 канали зворотного зв'язку кодера, DAC & ADC, а також 32 біти вводу/виводу, налаштовані таким чином: перші 8 (порт A) налаштовані як вхід, наступні 8 (порт B) налаштовані як вихід, наступні 8 (порт C) налаштовані як вхід, а останні 8 (порт D) налаштовані як вихід

```
loadrt hal_stg
```

У цьому прикладі встановлюється драйвер і здійснюється спроба автоматичного визначення адреси та моделі плати. За замовчуванням встановлюється 8 осей разом зі стандартною конфігурацією вводу-виводу: порти A і B налаштовані як вхідні, порти C і D налаштовані як вихідні.

6.15.2 Піни

- *stg.<channel>.counts* - (s32) Відстежує підраховані такти енкодера.
- *stg.<channel>.position* - (float) Виводить перетворену позицію.
- *stg.<channel>.dac-value* - (float) Визначає напругу для відповідного DAC.

- *stg.<channel>.adc-value* - (float) Відстежує виміряну напругу з відповідного ADC.
- *stg.in-<pinnum>* - (bit) Відстежує фізичний вхідний контакт.
- *stg.in-<pinnum>-not* - (bit) Відстежує фізичний вхідний контакт, але інвертований.
- *stg.out-<pinnum>* - (bit) Керує фізичним вихідним контактом

Для кожного виводу <channel> є номером осі, а <pinnum> є номером логічного виводу STG. Якщо визначено `II00`, є 16 вхідних виводів (in-00 .. in-15) і 16 вихідних виводів (out-00 .. out-15), і вони відповідають PORT ABCD (in-00 є PORTA.0, out-15 є PORTD.7).

Вивід HAL in-<pinnum> має значення TRUE, якщо фізичний вивід має високий рівень, і FALSE, якщо фізичний вивід має низький рівень. Вивід HAL in-<pinnum>-not є інвертованим — він має значення FALSE, якщо фізичний вивід має високий рівень. Підключивши сигнал до одного з них, користувач може визначити стан входу.

6.15.3 Параметри

- *stg.<channel>.position-scale* - (float) Кількість підрахунків / одиниця виміру користувача (для конвертації з підрахунків в одиниці).
- *stg.<channel>.dac-offset* - (float) Встановлює зміщення для відповідного DAC.
- *stg.<channel>.dac-gain* - (float) Встановлює коефіцієнт підсилення відповідного DAC.
- *stg.<channel>.adc-offset* - (float) Встановлює зміщення відповідного ADC.
- *stg.<channel>.adc-gain* - (float) Встановлює коефіцієнт підсилення відповідного ADC.
- *stg.out-<pinnum>-invert* - (bit) Інвертує вихідний контакт.

Параметр -invert визначає, чи є вихідний контакт активним у високому або низькому стані. Якщо -invert має значення FALSE, то встановлення HAL out- pin TRUE призводить до переведення фізичного контакту у високий стан, а FALSE — у низький. Якщо -invert має значення TRUE, то встановлення HAL out- pin TRUE призведе до переведення фізичного контакту у низький стан.

6.15.4 Функції

- *stg.capture-position* - Зчитує лічильники енкодера з осі <канал>.
- *stg.write-dacs* - Записує напруги на DAC.
- *stg.read-adcs* - Зчитує напруги з ADC.
- *stg.di-read* - Зчитує фізичні вхідні контакти всіх портів та оновлює всі контакти HAL in-<pinnum> та in-<pinnum>-not.
- *stg.do-write* - Зчитує всі виводи HAL out-<pinnum> та оновлює всі фізичні вихідні виводи.

6.16 Шатл

6.16.1 Опис

Shuttle — це компонент HAL, що не працює в реальному часі, який інтегрує пристрої ShuttleXpress, ShuttlePRO та ShuttlePRO2 від Contour Design з HAL від LinuxCNC.

Якщо драйвер запускається без аргументів командного рядка, він перевіряє всі файли пристроїв /dev/hidraw* на наявність пристроїв Shuttle і використовує всі знайдені пристрої. Якщо він запускається з аргументами командного рядка, він перевіряє тільки вказані пристрої.

ShuttleXpress має п’ять кнопок миттєвого керування, поворотне колесо з фіксаторами на 10 оборотів та пружинне зовнішнє колесо з 15 позиціями, яке повертається в центральне положення після відпускання.

ShuttlePRO має 13 кнопок миттєвого керування, поворотне колесо з фіксаторами на 10 оборотів та пружинне зовнішнє колесо з 15 положеннями, яке повертається в центральне положення після відпускання.

ShuttlePRO2 має 15 кнопок миттєвого керування, поворотне колесо з фіксаторами на 10 оборотів та пружинне зовнішнє колесо з 15 положеннями, яке повертається в центральне положення після відпускання.



Warning
Пристрої Shuttle мають внутрішній 8-бітний лічильник поточного положення колеса прокрутки. Драйвер Shuttle не може знати це значення, поки пристрій Shuttle не надішле своє перше повідомлення. Коли перше повідомлення надходить до драйвера, драйвер використовує повідомлене положення колеса прокрутки пристрою для ініціалізації лічильника до 0.
Це означає, що якщо перша подія генерується рухом поворотного колеса, цей перший рух буде втрачено.
Будь-яка взаємодія користувача з пристроєм Shuttle генерує подію, що інформує драйвер про положення колеса прокрутки. Отже, якщо ви (наприклад) натиснете одну з кнопок під час запуску, колесо прокрутки працюватиме нормально і реагуватиме на перше натискання.

6.16.2 Налаштування

Драйверу човника потрібен дозвіл на читання файлів пристрою /dev/hidraw*. Цього можна досягти, додавши файл /etc/udev/rules.d/99-shuttle.rules з таким вмістом:

```
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”
```

Пакет LinuxCNC Debian автоматично встановлює відповідний файл udev, але якщо ви створюєте LinuxCNC з вихідного коду і не використовуєте пакет Debian, вам доведеться встановити цей файл вручну. Якщо ви встановлюєте файл вручну, вам доведеться повідомити udev про необхідність перезавантаження файлів правил, виконавши команду `udevadm control --reload-rules`.

6.16.3 Піни

Усі назви контактів HAL починаються з префікса `shuttle`, після якого йде індекс пристрою (порядок, у якому драйвер їх знайшов), наприклад `shuttle.0` або `shuttle.2`.

<*Prefix*>.button-<ButtonNumber> (bit out)
Ці контакти мають стан True (1), коли кнопка натиснута.

<*Prefix*>.button-<ButtonNumber>-not (bit out)
Ці контакти мають інверсний стан до стану кнопки, тому вони мають значення True (1), коли кнопка не натиснута.

<*Prefix*>.counts (s32 out)
Накопичені показники з поворотного колеса (внутрішнього колеса).

<*Prefix*>.spring-wheel-s32 (s32 out)
Поточне відхилення пружинного колеса (зовнішнього колеса). У стані спокою воно дорівнює 0 і коливається від -7 проти годинникової стрілки до +7 за годинниковою стрілкою.

<*Prefix*>.spring-wheel-f (float out)
Поточне відхилення пружинного колеса (зовнішнього колеса). У стані спокою воно дорівнює 0,0, у крайньому положенні проти годинникової стрілки — -1,0, а у крайньому положенні за годинниковою стрілкою — +1,0. Пристрої Shuttle повідомляють про положення пружинного колеса у вигляді цілого числа від -7 до +7, тому цей вивід повідомляє лише 15 дискретних значень у своєму діапазоні.

6.17 Драйвер частотного перетворювача VFS11

Це програма HAL, що не працює в реальному часі, для керування частотними перетворювачами серії S11 від Toshiba.

vfs11_vfd підтримує послідовні та TCP-з'єднання. Послідовні з'єднання можуть бути RS232 або RS485. RS485 підтримується в повно- та напівдуплексному режимі. TCP-з'єднання можуть бути пасивними (очікування вхідного з'єднання) або активними вихідними з'єднаннями, що може бути корисно для підключення до пристроїв на базі TCP або через термінальний сервер.

Незалежно від типу підключення, vfs11_vfd працює як головний пристрій Modbus.

Цей компонент завантажується за допомогою команди halcmd "loadusr":

```
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd
```

Наведена вище команда говорить: loadusr, чекайте завантаження named, компонент vfs11_vfd, named spindle-vfd

6.17.1 Параметри командного рядка

«vfs11_vfd» здебільшого налаштовується через параметри INI-файлу. Параметри командного рядка:

- *-n or --name <halname>* : встановити назву компонента HAL
- *-I або --ini <назва_ініфайлу>*: взяти конфігурацію з цього INI-файлу. За замовчуванням використовує змінна середовища INI_ФАЙЛ_НАЗВА.
- *-S або --section <назва розділу>*: взяти конфігурацію з цього розділу в INI-файлі. За замовчуванням використовується значення *VFS11*.
- «-d або --debug» увімкне виведення повідомлень налагодження у консоль.
- *-m або --modbus-debug* увімкнути повідомлення Modbus у виводі консолі
- *-r або --report-device* повідомляє про властивості пристрою в консолі під час запуску

Налагодження можна ввімкнути або вимкнути, надіславши сигнал USR1 до процесу vfs11_vfd. Налагодження Modbus можна ввімкнути або вимкнути, надіславши сигнал USR2 до процесу vfs11_vfd (приклад: `kill -USR1 $(pidof vfs11_vfd)`).

Note
Що якщо є помилки конфігурації послідовного порту, увімкнення детального режиму може призвести до потоку помилок тайм-ауту.

6.17.2 Піни

Де <n> — це `vfs11_vfd` або ім'я, задане під час завантаження з опцією -n.

- *<n>.acceleration-pattern* (біт, in) якщо значення true, встановлює часи розгону та уповільнення, як визначено в регістрах F500 та F501 відповідно. Використовується в ПІД-контурах для вибору коротших часів наростання/зниження, щоб уникнути коливань.
- *<n>.alarm-code* (s32, out) ненульове значення, якщо накопичувач перебуває в стані тривоги. Бітове зображення, що описує інформацію про тривогу (див. опис регістра FC91). Використовуйте err-reset (див. нижче), щоб скинути тривогу.
- *<n>.at-speed* (біт, вихід), коли привід працює на заданій швидкості (див. допуск швидкості нижче)
- *<n>.current-load-percentage* (число з плаваючою комою, вихід), що надходить від частотно-регульованого перетворювача (ЧРП)
- *<n>.dc-brake* (біт, in) увімкнути гальмо постійного струму. Також вимикає шпиндель.
- *<n>.enable* (біт, вхід) увімкнення частотного перетворювача. Якщо значення false (хибність), усі робочі параметри все ще зчитуються, але керування звільняється, а керування панеллю вмикається (залежно від налаштувань частотного перетворювача).
- *<n>.err-reset* (біт, вхід) скидає помилки (тривоги, тобто стан відключення та аварійної зупинки). Скидання частотно-регульованого приводу може спричинити 2-секундну затримку до його перезавантаження та повторного запуску Modbus.
- *<n>.estop* (біт, in) переводить частотний перетворювач у стан аварійної зупинки. Робота неможлива, доки не буде очищено за допомогою скидання помилки або повторного ввімкнення/вимкнення живлення.
- *<n>.frequency-command* (float, out) поточна цільова частота в Гц, встановлена за допомогою команди швидкості (яка в обертах на хвилину) з частотно-регульованого перетворювача
- *<n>.frequency-out* (float, out) поточна вихідна частота частотного перетворювача
- *<n>.inverter-load-percentage* (float, out) звіт про поточне навантаження від частотного перетворювач
- *<n>.is-e-stopped* (біт, вихід) ЧРП перебуває в стані аварійної зупинки (миготить "E" на панелі). Використовуйте err-reset для перезавантаження ЧРП та очищення стану аварійної зупинки.
- *<n>.is-stopped* (біт, вихід) істина, коли частотний перетворювач повідомляє про вихід 0 Гц
- *<n>.max-rpm* (float, R) фактичне обмеження обертів на хвилину на основі максимальної частоти, яку може генерувати VFD, та значень, зазначених на табличці двигуна. Наприклад, якщо номінальна частота становить 50 Гц, а номінальна частота обертання — 1410 об/хв, але VFD може генерувати до 80 Гц, то max-rpm буде дорівнювати 2256 (80*1410/50). Обмеження частоти зчитується з VFD під час запуску. Щоб збільшити верхню межу частоти, необхідно змінити параметри UL і FH на панелі. Інструкції щодо налаштування максимальної частоти див. в посібнику VF-S11.

- *<n>.modbus-ok* (біт, вихід) true, коли сеанс Modbus успішно встановлено, і останні 10 транзакцій повернуто без помилок.
- *<n>.motor-RPM* (float, out) розраховане поточне значення обертів за хвилину, від частотного перетворювача
- *<n>.output-current-percentage* (float, out) від частотного перетворювача
- *<n>.output-voltage-percentage* (float, out) від частотного перетворювача
- *<n>.output-voltage* (float, out) від частотного перетворювача
- *<n>.speed-command* (float, in) швидкість, що надсилається до частотного перетворювача (ЧРП) в об/хв. Надсилання швидкості, що перевищує максимальні об/хв двигуна, встановлені в ЧРП, є помилкою
- *<n>.spindle-fwd* (bit, in) 1 для FWD та 0 для REV, надсилається до частотного перетворювача
- *<n>.spindle-on* (bit, in) 1 для УВІМК. та 0 для ВИМК. надсилається на частотно-регульований перетворювач, увімкнено лише під час роботи
- *<n>.spindle-rev* (bit, in) 1 для УВІМК. та 0 для ВИМК., увімкнено лише під час роботи
- *<n>.jog-mode* (біт, вхід) 1 для увімкнення та 0 для вимкнення, увімкнення «режиму поштовху» VF-S11. Регулювання швидкості вимкнено, а вихідна частота визначається регістром F262 (заздалегідь встановлено на 5 Гц). Це може бути корисно для орієнтації шпинделя. У звичайному режимі VFD вимикається, якщо частота падає нижче 12 Гц.
- *<n>.status* (s32, out) Стан приводу частотно-регульованого перетворювача (див. інструкцію з експлуатації комунікаційних функцій TOSVERT VF-S11, регістр FD01). Растрове зображення.
- *<n>.trip-code* (s32, out) код відключення, якщо VF-S11 перебуває у стані відключення.
- *<n>.error-count* (s32, out) кількість транзакцій Modbus, які повернули помилку
- *<n>.max-speed* (bit, in) ігноруйте параметр часу циклу та запускайте Modbus на максимальній швидкості, що призведе до збільшення навантаження на процесор. Рекомендовано використовувати під час позиціонування шпинделя.

6.17.3 Параметри

Де <n> — це `vfs11_vfd` або ім'я, задане під час завантаження з опцією -n.

- *<n>.frequency-limit* (float, RO) верхня межа, зчитана з налаштувань частотно-регульованого перетворювача.
- *<n>.loop-time* (float, RW) як часто опитується Modbus (інтервал за замовчуванням 0,1 секунди)
- *<n>.nameplate-HZ* (float, RW) Задана частота двигуна в Гц (за замовчуванням 50). Використовується для розрахунку цільової частоти (разом із значенням заданої частоти обертання двигуна) для цільового значення обертів двигуна, заданого командою швидкості.
- *<n>.nameplate-RPM* (float, RW) Оберти двигуна на заводській табличці (за замовчуванням 1410)
- *<n>.rpm-limit* (float, RW) не перевищувати програмне обмеження для обертів двигуна (за замовчуван — nameplate-RPM).
- *<n>.tolerance* (float, RW) допуск швидкості (за замовчуванням 0,01) для визначення того, чи шпиндель обертається на швидкості (0,01 означає: вихідна частота знаходиться в межах 1% від цільової частоти)

6.17.4 INI-файл конфігурації

Тут перелічено всі опції, які розуміє vfs11_vfd. Типові налаштування для RS-232, RS-485 та TCP можна знайти в *src/hal/user_comps/vfs11_vfd/*.ini*.

```
[VFS11]
# b''пb''b''оb''b''сb''b''лb''b''іb''b''дb''b''оb''b''вb''b''нb''b''еb'' b''зb'''b''єb''b' ↩
    'дb''b''нb''b''аb''b''нb''b''нb''b''яb''
TYPE=rtu

# b''пb''b''оb''b''сb''b''лb''b''іb''b''дb''b''оb''b''вb''b''нb''b''иb''b''йb'' b''пb''b' ↩
    'оb''b''рb''b''тb''
DEVICE=/dev/ttyS0

# TCP-b''сb''b''еb''b''рb''b''вb''b''еb''b''рb'' - b''оb''b''чb''b''іb''b''кb''b''уb''b' ↩
    'вb''b''аb''b''нb''b''нb''b''яb'' b''вb''b''хb''b''іb''b''дb''b''нb''b''оb''b''гb''b' ↩
    'оb'' b''зb'''b''єb''b''дb''b''нb''b''аb''b''нb''b''нb''b''яb''
TYPE=tcpserver

# b''нb''b''оb''b''мb''b''еb''b''рb'' b''пb''b''оb''b''рb''b''тb''b''уb'' tcp b''дb''b' ↩
    'лb''b''яb'' TYPE=tcpserver b''аb''b''бb''b''оb'' tcpclient
PORT=1502

# TCP-b''кb''b''лb''b''іb''b''єb''b''нb''b''тb'' b''—b'' b''аb''b''кb''b''тb''b''иb''b' ↩
    'вb''b''нb''b''еb'' b''вb''b''иb''b''хb''b''іb''b''дb''b''нb''b''еb'' b''зb''b'''b''b' ↩
    'єb''b''дb''b''нb''b''аb''b''нb''b''нb''b''яb''
TYPE=tcpclient

# b''пb''b''уb''b''нb''b''кb''b''тb'' b''пb''b''рb''b''иb''b''зb''b''нb''b''аb''b''чb''b' ↩
    'еb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b''іb''b''дb''b''кb''b''лb''b' ↩
    'юb''b''чb''b''еb''b''нb''b''нb''b''яb'', b''яb''b''кb''b''щb''b''оb'' TYPE=tcpclient
TCPDEST=192.168.1.1

#---------- meaningful only if TYPE=rtu -------
# b''дb''b''еb''b''тb''b''аb''b''лb''b''іb'' b''пb''b''оb''b''сb''b''лb''b''іb''b''дb''b' ↩
    'оb''b''вb''b''нb''b''оb''b''гb''b''оb'' b''пb''b''рb''b''иb''b''сb''b''тb''b''рb''b' ↩
    'оb''b''юb''
# 5 6 7 8
BITS= 5

# b''пb''b''аb''b''рb''b''нb''b''іb'' b''нb''b''еb''b''пb''b''аb''b''рb''b''нb''b''іb'' b' ↩
    'жb''b''оb''b''дb''b''нb''b''іb''
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
# b''цb''b''яb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''яb'' b''мb''b''оb''b''жb''b' ↩
    'еb'' b''нb''b''еb'' b''пb''b''рb''b''аb''b''цb''b''юb''b''вb''b''аb''b''тb''b''иb'' b' ↩
    'зb''b''іb'' b''сb''b''тb''b''аb''b''нb''b''дb''b''аb''b''рb''b''тb''b''нb''b''иb''b' ↩
    'мb'' b''пb''b''аb''b''кb''b''еb''b''тb''b''оb''b''мb'' Ubuntu
# libmodbus5/libmodbus-dev b''іb'' b''гb''b''еb''b''нb''b''еb''b''рb''b''уb''b''вb''b''аb'' ↩
    b''тb''b''иb'' b''пb''b''оb''b''пb''b''еb''b''рb''b''еb''b''дb''b''жb''b''еb''b''нb''b' ↩
    'нb''b''яb''.
# b''Bb''b''иb''b''кb''b''оb''b''нb''b''аb''b''нb''b''нb''b''яb'' b''пb''b''рb''b''оb''b' ↩
```

```
    'дb''b''ob''b''вb''b''жb''b''иb''b''тb''b''ьb''b''cb''b''яb'', b''яb''b''кb'' b''яb''b' ↩
    'кb''b''щb''b''ob'' b''бb'' b''бb''b''уb''b''лb''b''ob'' b''зb''b''ab''b''дb''b''ab''b' ↩
    'нb''b''ob'' RTS_MODE=up.
RTS_MODE=up
#--------------------

# b''тb''b''ab''b''йb''b''мb''b''eb''b''pb''b''иb'' Modbus b''уb'' b''cb''b''eb''b''кb''b' ↩
    'уb''b''нb''b''дb''b''ab''b''хb''
# b''мb''b''ib''b''жb''b''cb''b''иb''b''мb''b''вb''b''ob''b''лb''b''ьb''b''нb''b''иb''b' ↩
    'йb'' b''тb''b''ab''b''йb''b''мb''b''eb''b''pb''
BYTE_TIMEOUT=0.5
# b''тb''b''ab''b''йb''b''мb''b''eb''b''pb'' b''пb''b''ab''b''кb''b''eb''b''тb''b''ib''b' ↩
    'вb''
RESPONSE_TIMEOUT=0.5

# b''цb''b''ib''b''лb''b''ьb''b''ob''b''вb''b''иb''b''йb'' b''ib''b''дb''b''eb''b''нb''b' ↩
    'тb''b''иb''b''фb''b''ib''b''кb''b''ab''b''тb''b''ob''b''pb'' Modbus
TARGET=1

# b''уb'' b''pb''b''ab''b''зb''b''ib'' b''зb''b''бb''b''ob''b''юb'' b''вb''b''вb''b''ob''b' ↩
    'дb''b''уb''/b''вb''b''иb''b''вb''b''ob''b''дb''b''уb'', b''cb''b''пb''b''pb''b''ob''b' ↩
    'бb''b''уb''b''йb''b''тb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''пb''b''ib''b''дb''b' ↩
    'кb''b''лb''b''юb''b''чb''b''иb''b''тb''b''иb''b''cb''b''яb'' b''пb''b''ib''b''cb''b' ↩
    'лb''b''яb'' b''cb''b''нb''b''уb''
# b''пb''b''pb''b''ob''b''тb''b''яb''b''гb''b''ob''b''мb'' RECONNECT_DELAY b''cb''b''eb''b' ↩
    'кb''b''уb''b''нb''b''дb''
RECONNECT_DELAY=1

# b''pb''b''ib''b''зb''b''нb''b''ib'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b' ↩
    'pb''b''иb''
DEBUG=10
MODBUS_DEBUG=0
POLLCYCLES=10
```

6.17.5 HAL приклад

```
#
# b''пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b''вb''b''иb''b''кb''b''ob''b''pb''b' ↩
    'иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'' b''дb''b''pb''b''ab''b''йb''b''вb''b' ↩
    'eb''b''pb''b''ab'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''нb''b''ob''b''гb''b' ↩
    'ob'' b''пb''b''eb''b''pb''b''eb''b''тb''b''вb''b''ob''b''pb''b''юb''b''вb''b''ab''b' ↩
    'чb''b''ab'' VF-S11
#
#
loadusr -Wn spindle-vfd vfs11_vfd -n spindle-vfd

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''кb''b''ob''b' ↩
    'нb''b''тb''b''ab''b''кb''b''тb''b''иb'' b''нb''b''ab''b''пb''b''pb''b''яb''b''мb''b' ↩
    'кb''b''уb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''дb''b''ob'' b' ↩
    'чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''eb''b' ↩
    'pb''b''eb''b''тb''b''вb''b''ob''b''pb''b''юb''b''вb''b''ab''b''чb''b''ab'' (b''Чb''b' ↩
    'Пb'')
net vfs11-fwd spindle-vfd.spindle-fwd <= spindle.0.forward
net vfs11-rev spindle-vfd.spindle-rev <= spindle.0.reverse

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''ьb'' b''нb''b''ab'' b''шb''b''тb''b''иb''b''фb''b' ↩
    'тb''b''ib'' b''дb''b''ob'' VF-S11
net vfs11-run spindle-vfd.spindle-on <= spindle.0.on
```

```
# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' VF-S11 b''нb''b' ↩
    'ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b''ib'' b''дb''b''оb'' b' ↩
    'pb''b''уb''b''хb''b''уb'' b''нb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b' ↩
    'cb''b''тb''b''ib''
net vfs11-b''нb''b''ab''-b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b''ib''  ↩
    spindle.0.b''нb''b''ab''-b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b''ib'' ↩
     <= spindle-vfd.b''нb''b''ab''-b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b ↩
    ''ib''

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''ьb'' RPM b''дb''b''оb'' VF-S11
net vfs11-RPM spindle-vfd.speed-command <= spindle.0.speed-out

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''иb''b''тb''b''иb'' b''гb''b''ab''b' ↩
    'лb''b''ьb''b''мb''b''оb'' b''пb''b''оb''b''сb''b''тb''b''ib''b''йb''b''нb''b''оb''b' ↩
    'гb''b''оb'' b''сb''b''тb''b''pb''b''уb''b''мb''b''уb'' VF-S11
# b''оb''b''сb''b''кb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''вb''b''оb''b''нb''b''оb'' b' ↩
    'сb''b''пb''b''оb''b''жb''b''иb''b''вb''b''аb''b''єb'' b''eb''b''нb''b''eb''b''pb''b' ↩
    'гb''b''ib''b''юb'' b''пb''b''ib''b''дb'' b''чb''b''аb''b''сb'' b''вb''b''иb''b''мb''b' ↩
    'кb''b''нb''b''eb''b''нb''b''нb''b''яb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b' ↩
    'лb''b''яb'', b''кb''b''оb''b''нb''b''тb''b''аb''b''кb''b''тb'' b''гb''b''аb''b''лb''b' ↩
    'ьb''b''мb''b''аb'' b''пb''b''оb''b''сb''b''тb''b''ib''b''йb''b''нb''b''оb''b''гb''b' ↩
    'оb'' b''сb''b''тb''b''pb''b''уb''b''мb''b''уb''
# b''кb''b''pb''b''аb''b''щb''b''eb'' b''пb''b''pb''b''иb''b''вb''b''оb''b''дb''b''иb''b' ↩
    'тb''b''иb'' b''вb'' b''дb''b''ib''b''юb'' b''зb''b''аb'' b''дb''b''оb''b''пb''b''оb''b' ↩
    'мb''b''оb''b''гb''b''оb''b''юb'' b''мb''b''оb''b''нb''b''оb''b''фb''b''лb''b''оb''b' ↩
    'пb''b''аb'', b''яb''b''кb''b''иb''b''йb'' b''сb''b''пb''b''pb''b''аb''b''цb''b''ьb''b' ↩
    'оb''b''вb''b''уb''b''єb'' b''нb''b''аb'' b''сb''b''пb''b''аb''b''дb''b''ib'' b''фb''b' ↩
    'pb''b''оb''b''нb''b''тb''b''уb'' b''вb''b''кb''b''лb''b''юb''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb''
#net vfs11-spindle-brake spindle.N.brake => spindle-vfd.dc-brake

# b''вb''b''иb''b''кb''b''оb''b''pb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b''вb''b' ↩
    'аb''b''тb''b''иb'' b''pb''b''eb''b''жb''b''иb''b''мb'' b''пb''b''оb''b''шb''b''тb''b' ↩
    'оb''b''вb''b''хb''b''уb'' VFS11 b''дb''b''лb''b''яb'' b''оb''b''pb''b''ib''b''єb''b' ↩
    'нb''b''тb''b''аb''b''цb''b''ib''b''їb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b' ↩
    'лb''b''яb''
# b''дb''b''иb''b''вb''. orient.9 b''тb''b''ab'' motion.9
net spindle-orient spindle.0.orient spindle-vfd.max-speed spindle-vfd.jog-mode

# b''мb''b''аb''b''тb''b''иb'' b''пb''b''pb''b''ib''b''оb''b''pb''b''иb''b''тb''b''eb''b' ↩
    'тb'' b''нb''b''аb''b''дb'' b''пb''b''аb''b''нb''b''eb''b''лb''b''лb''b''юb'' b''кb''b' ↩
    'eb''b''pb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb''
setp spindle-vfd.enable 1
```

6.17.6 Робота панелі

Драйвер vfs11_vfd має пріоритет над керуванням панеллю, поки він увімкнений (див. контакт «enable»), ефективно вимикаючи панель. Скидання контакту «enable» знову вмикає панель. Контакти та параметри все ще можна налаштовувати, але вони не будуть записані у VFD, доки не буде встановлено контакт «enable». Параметри роботи все ще зчитуються, коли управління шиною вимкнено. Вихід із драйвера vfs11_vfd у контрольований спосіб звільнить VFD від шини та відновить управління панеллю.

Докладнішу інформацію див. у посібнику для інтеграторів LinuxCNC. Детальний опис регістрів частотно-регульованих приводів Toshiba див. у «Посібнику з комунікаційних функцій TOSVERT VF-S11» (номер документа Toshiba E6581222) та «Посібнику з експлуатації TOSVERT VF-S11» (номер документа Toshiba E6581158).

6.17.7 Відновлення помилок

`vfs11_vfd` відновлюється після помилок вводу/виводу наступним чином: спочатку всі виводи HAL встановлюються у значення за замовчуванням, і драйвер переходить у режим сну на `RECONNECT_DE` секунд (за замовчуванням 1 секунда).

- Послідовний режим (`TYPE=rtu`): у разі помилки закрити та знову відкрити послідовний порт.
- Режим TCP-сервера (`TYPE=tcpserver`): після втрати TCP-з'єднання драйвер знову перейде до очікування вхідних з'єднань.
- Режим TCP-клієнта (`TYPE=tcpclient`): після втрати TCP-з'єднання драйвер знову підключиться до *TCPDEST:PORTNO*.

6.17.8 Налаштування частотного перетворювача VFS11 для використання в Modbus

6.17.8.1 Підключення послідовного порту

VF-S11 має роз'єм RJ-45 для послідовної передачі даних. На жаль, він не має стандартного роз'єму RS-232 і логічних рівнів. Рекомендований Toshiba спосіб: підключіть перетворювач USB-послідовний USB001Z до приводу і підключіть USB-порт до ПК. Більш дешевою альтернативою є саморобний інтерфейс (підказки від служби підтримки Toshiba, схема підключення).

Примітка: вихід 24 В з частотного перетворювача не має захисту від короткого замикання.

Заводські налаштування послідовного порту за замовчуванням — 9600/8/1/парні, протокол за замовчуванням використовує власний «Toshiba Inverter Protocol».

6.17.8.2 Налаштування Modbus

Перед тим, як VF-S11 почне взаємодіяти з цим модулем, необхідно налаштувати кілька параметрів. Це можна зробити вручну за допомогою панелі керування або через послідовний канал зв'язку. Компанія Toshiba постачає програму для Windows під назвою «PCM001Z», яка може зчитувати/налашт параметри в VFD. Примітка: PCM001Z підтримує тільки протокол інвертора Toshiba. Отже, останнім параметром, який потрібно змінити, є протокол — встановіть його з Toshiba Inverter Protocol на Modbus; після цього програма для Windows стане непотрібною.

Щоб збільшити верхню межу частоти, потрібно змінити параметри UL та FH на панелі. Я збільшив їх з 50 до 80.

Дивіться dump-params.mio для опису нестандартних параметрів VF-S11 моєї конфігурації. Цей файл призначений для інтерактивна утиліта modio Modbus.

6.17.9 Примітка щодо програмування

Драйвер vfs11_vfd використовує бібліотеку libmodbus версії 3, яка є новішою за код версії 2, що використовується в `gs2_vfd`.

Пакет Ubuntu `libmodbus5` та `libmodbus-dev` доступні лише починаючи з Ubuntu 12 («Precise Pengolin»). Більше того, ці пакети не підтримують прапорці MODBUS_RTS_MODE_*. Тому, під час компіляції vfs11_vfd з використанням цієї бібліотеки може з'явитися попередження, якщо в файлі INI вказано RTS_MODE=.

Щоб скористатися повним функціоналом на Lucid and Precise:

- вилучити пакети libmodbus: `sudo apt-get remove libmodbus5 libmodbus-dev`

- зібрати та встановити libmodbus версії 3 з вихідного коду, як описано here.

Libmodbus не збирається на Ubuntu Hardy, тому vfs11_vfd недоступний на Hardy.

Chapter 7

Приклади обладнання

7.1 Паралельний порт PCI

Коли ви додаєте другий паралельний порт до шини PCI, вам потрібно з’ясувати його адресу, перш ніж ви зможете використовувати його з LinuxCNC.

Щоб знайти адресу вашої карти паралельного порту, відкрийте вікно терміналу та введіть

```
lspci -v
```

Ви побачите щось подібне, а також інформацію про все інше на шині PCI:

```
0000:00:10.0 b’’Kb’’b’’ob’’b’’нb’’b’’тb’’b’’pb’’b’’ob’’b’’лb’’b’’eb’’b’’pb’’ b’’зb’’b’ ↩
    ’вb’’’b’’яb’’b’’зb’’b’’кb’’b’’уb’’: \
        1-b’’пb’’b’’ob’’b’’pb’’b’’тb’’b’’ob’’b’’вb’’b’’иb’’b’’йb’’ b’’пb’’b’’ab’’b’’pb’’b’ ↩
            ’ab’’b’’лb’’b’’eb’’b’’лb’’b’’ьb’’b’’нb’’b’’иb’’b’’йb’’ b’’ab’’b’’дb’’b’’ab’’b’ ↩
            ’пb’’b’’тb’’b’’eb’’b’’pb’’ NetMos Technology PCI (b’’вb’’b’’eb’’b’’pb’’b’’cb’’b’ ↩
            ’ib’’b’’яb’’ 01)
        b’’Пb’’b’’ib’’b’’дb’’b’’cb’’b’’иb’’b’’cb’’b’’тb’’b’’eb’’b’’мb’’b’’ab’’: LSI Logic / ↩
             Symbios Logic: b’’Нb’’b’’eb’’b’’вb’’b’’ib’’b’’дb’’b’’ob’’b’’мb’’b’’иb’’b’’йb’’  ↩
            b’’пb’’b’’pb’’b’’иb’’b’’cb’’b’’тb’’b’’pb’’b’’ib’’b’’йb’’ 0010
        b’’Пb’’b’’pb’’b’’ab’’b’’пb’’b’’ob’’b’’pb’’b’’иb’’: b’’cb’’b’’eb’’b’’pb’’b’’eb’’b’ ↩
            ’дb’’b’’нb’’b’’ib’’b’’йb’’ b’’pb’’b’’ib’’b’’вb’’b’’eb’’b’’нb’’b’’ьb’’ b’’pb’’b’ ↩
            ’ob’’b’’зb’’b’’pb’’b’’ob’’b’’бb’’b’’кb’’b’’иb’’, 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]
```

У моєму випадку адреса була першою, тому я змінив свій файл .hal з

```
loadrt hal_parport cfg=0x378
```

до

```
loadrt hal_parport cfg=”0x378 0xa800 in”
```

(Зверніть увагу на подвійні лапки навколо адрес.)

а потім додав наступні рядки, щоб парпорт був прочитаний та записаний:

```
addf parport.1.read base-thread
addf parport.1.write base-thread
```

Після виконання вищезазначеного запустіть конфігурацію та перевірте, чи паралельний порт завантажився у вікні Machine/Show HAL Configuration.

7.2 Управління шпинделем

LinuxCNC може керувати до 8 шпинделів. Кількість встановлюється в файлі INI. У наведених нижче прикладах йдеться про конфігурацію з одним шпинделем, де контакти керування шпинделем мають назви на кшталт «spindle.0..». У разі верстата з декількома шпинделями змінюється лише те, що з'являються додаткові контакти з назвами на кшталт «spindle.6..».

7.2.1 Швидкість шпинделя 0-10 вольт

Якщо швидкість вашого шпинделя контролюється аналоговим сигналом (наприклад, частотним перетворювачем із сигналом від 0 В до 10 В), і ви використовуєте плату DAC, таку як m5i20, для виведення керуючого сигналу:

Спочатку потрібно визначити масштаб швидкості шпинделя для керуючого сигналу, тобто напруги. Для цього прикладу максимальна швидкість шпинделя 5000 об/хв дорівнює 10 вольтам.

$$\frac{10\,Volts}{5000\,RPM} = \frac{0.002\,Volts}{1\,RPM}$$

Нам потрібно додати компонент масштабування до HAL-файлу, щоб масштабувати *spindle.N.speed-out* до значень від 0 до 10, необхідних для частотного перетворювача, якщо ваша плата DAC не виконує масштабування.

```
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''нb''b''ab''b''зb''b''вb''b''ab'' b''вb''b''ab''b' ↩
    'шb''b''ob''b''гb''b''ob'' b''вb''b''иb''b''вb''b''ob''b''дb''b''yb'' DAC>
```

7.2.2 PWM Швидкість шпинделя

Якщо ваш шпиндель можна керувати за допомогою PWM-сигналу, використовуйте компонент *pwmgen* для створення сигналу:

```
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
# b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''мb''b''ab''b' ↩
    'кb''b''cb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''yb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b' ↩
    'лb''b''яb'' b''вb'' b''ob''b''бb''b''eb''b''pb''b''тb''b''ab''b''xb'' b''зb''b''ab'' b' ↩
    'xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''yb''
setp pwmgen.0.scale 1800
```

Це передбачає, що реакція контролера шпинделя на PWM є простою: 0% PWM дає 0 об/хв, 10% PWM дає 180 об/хв тощо. Якщо для обертання шпинделя необхідний мінімальний PWM, дотримуйтесь прикладу в зразковій конфігурації nist-lathe, щоб використовувати компонент шкали.

7.2.3 Увімкнення шпинделя

Якщо вам потрібен сигнал увімкнення шпинделя, підключіть вихідний контакт до *spindle.0.on*. Щоб підключити ці контакти до контакту паралельного порту, вставте в файл .hal щось на зразок наведеного нижче, переконавшись, що ви вибрали контакт, який підключений до вашого пристрою керування.

```
net spindle-enable spindle.0.on => parport.0.pin-14-out
```

7.2.4 Напрямок шпинделя

Якщо ви маєте контроль над напрямком обертання шпинделя, то контакти HAL «spindle.*N*.forward» та «spindle.*N*.reverse» керуються G-кодами M3 та M4. Швидкість обертання шпинделя S*n* повинна бути встановлена на позитивне значення, відмінне від нуля, щоб M3/M4 могли ввімкнути рух шпинделя.

Щоб підключити ці контакти до контакту паралельного порту, додайте щось на кшталт наступного у ваш файл .hal, переконавшись, що ви вибрали контакт, підключений до вашого пристрою керування.

```
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 Плавний пуск шпинделя

Якщо вам потрібно збільшити швидкість шпинделя, а ваша система управління не має такої функції, це можна зробити в HAL. В основному, вам потрібно перехопити вихід «spindle.*N*.speed-out» і пропустити його через компонент «limit2» зі встановленим масштабом, щоб збільшити оберти з «spindle.*N*.speed-out» до вашого пристрою, який отримує оберти. Друга частина полягає в тому, щоб повідомити LinuxCNC, коли шпиндель досягне необхідної швидкості, щоб можна було розпочати рух.

У прикладі 0-10 вольт лінія

```
net spindle-speed-scale spindle.0.speed-out => scale.0.in
```

змінюється, як показано в наступному прикладі:

Вступ до компонентів HAL limit2 та близьких до них Якщо ви раніше з ними не стикалися, ось короткий вступ до двох компонентів HAL, що використовуються в наступному прикладі.

- «Limit2» — це компонент HAL (число з плаваючою комою), який приймає вхідне значення та надає вихідне значення, обмежене діапазоном max/min, а також обмежене тим, щоб не перевищувати задану швидкість зміни.
- «Близький» — це компонент HAL (число з плаваючою комою) з двійковим виходом, який показує, чи приблизно рівні два вхідні значення.

Більше інформації можна знайти в документації до компонентів HAL або на сторінках довідника, просто скажіть «man limit2» або «man near» у терміналі.

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''тb''b''eb'' b''мb''b''ob''b' ↩
    'дb''b''уb''b''лb''b''іb'' b''рb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b' ↩
    'ob'' b''чb''b''ab''b''cb''b''уb'' limit2 b''тb''b''ab'' near b''зb'' b''іb''b''мb''b' ↩
    'eb''b''нb''b''ab''b''мb''b''иb'', b''щb''b''ob''b''бb'' b''бb''b''уb''b''лb''b''ob'' b' ↩
    'лb''b''eb''b''гb''b''шb''b''eb'' b''вb''b''іb''b''дb''b''cb''b''тb''b''eb''b''жb''b' ↩
    'уb''b''вb''b''ab''b''тb''b''иb'' b''їb''b''хb''b''нb''b''іb'' b''зb''b''вb'''b''яb''b' ↩
    'зb''b''кb''b''иb''
```

```
loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed

# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b' ↩
    'їb'' b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb''
addf spindle-ramp servo-thread
addf spindle-at-speed servo-thread

# b''вb''b''сb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b''ab''b' ↩
    'рb''b''ab''b''мb''b''eb''b''тb''b''рb'' b''дb''b''лb''b''яb'' b''мb''b''ab''b''кb''b' ↩
    'сb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ob''b''сb''b''тb''b''іb'' b''зb''b''мb''b''іb''b''нb''b''иb''
# (b''мb''b''ab''b''кb''b''сb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''eb'' b''пb''b' ↩
    'рb''b''иb''b''сb''b''кb''b''ob''b''рb''b''eb''b''нb''b''нb''b''яb''/b''сb''b''пb''b' ↩
    'ob''b''вb''b''іb''b''лb''b''ьb''b''нb''b''eb''b''нb''b''нb''b''яb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''вb'' b''ob''b''дb''b''иb''b''нb''b''иb''b' ↩
    'цb''b''яb''b''хb'' b''зb''b''ab'' b''сb''b''eb''b''кb''b''уb''b''нb''b''дb''b''уb'')
setp spindle-ramp.maxv 60

# b''вb''b''иb''b''вb''b''eb''b''сb''b''тb''b''иb'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'іb''b''сb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b' ↩
    'нb''b''ab''b''зb''b''ob''b''вb''b''нb''b''іb'' b''тb''b''ab'' b''нb''b''ab''b''пb''b' ↩
    'рb''b''ab''b''вb''b''иb''b''тb''b''иb'' b''їb''b''їb'' b''нb''b''ab'' b''рb''b''ab''b' ↩
    'мb''b''пb''b''уb'' b''нb''b''ab''b''рb''b''ob''b''сb''b''тb''b''ab''b''нb''b''нb''b' ↩
    'яb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb''
net spindle-cmd <= spindle.0.speed-out => spindle-ramp.in

# b''Bb''b''иb''b''хb''b''іb''b''дb''b''нb''b''иb''b''йb'' b''сb''b''иb''b''гb''b''нb''b' ↩
    'ab''b''лb'' b''рb''b''ab''b''мb''b''пb''b''иb'' b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'eb''b''лb''b''яb'' b''нb''b''ab''b''дb''b''сb''b''иb''b''лb''b''ab''b''єb''b''тb''b' ↩
    'ьb''b''сb''b''яb'' b''нb''b''ab'' b''шb''b''кb''b''ab''b''лb''b''уb'' b''вb''
net spindle-ramped <= spindle-ramp.out => scale.0.in

# b''щb''b''ob''b''бb'' b''зb''b''нb''b''ab''b''тb''b''иb'', b''кb''b''ob''b''лb''b''иb'' b ↩
    ''пb''b''ob''b''чb''b''иb''b''нb''b''ab''b''тb''b''иb'' b''рb''b''уb''b''хb'', b''мb''b' ↩
    'иb'' b''нb''b''ab''b''дb''b''сb''b''иb''b''лb''b''ab''b''єb''b''мb''b''ob'' b''нb''b' ↩
    'ab''b''йb''b''бb''b''лb''b''иb''b''жb''b''чb''b''иb''b''йb'' b''кb''b''ob''b''мb''b' ↩
    'пb''b''ob''b''нb''b''eb''b''нb''b''тb''
# (b''нb''b''ab''b''зb''b''вb''b''ab''b''нb''b''иb''b''йb'' spindle-at-speed) b''дb''b' ↩
    'ob'' b''зb''b''ab''b''дb''b''ab''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''сb''b''тb''b''іb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b' ↩
    'яb'' b''вb''b''іb''b''дb''
# b''сb''b''иb''b''гb''b''нb''b''ab''b''лb''b''уb'' spindle-cmd b''тb''b''ab'' b''фb''b' ↩
    'ab''b''кb''b''тb''b''иb''b''чb''b''нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''сb''b''тb''b''іb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b' ↩
    'яb'',
# b''зb''b''ab'' b''уb''b''мb''b''ob''b''вb''b''иb'', b''щb''b''ob'' b''вb''b''ab''b''шb''  ↩
   b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''ьb'' b''мb''b''ob''b''жb''b''eb'' b' ↩
    'пb''b''рb''b''иb''b''сb''b''кb''b''ob''b''рb''b''юb''b''вb''b''ab''b''тb''b''иb''b' ↩
    'сb''b''яb'' b''зb'' b''нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''вb''b''ab''b' ↩
    'нb''b''нb''b''яb''b''мb'' maxv.
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2

# b''вb''b''иb''b''хb''b''іb''b''дb''b''нb''b''иb''b''йb'' b''сb''b''иb''b''гb''b''нb''b' ↩
    'ab''b''лb'' b''вb''b''іb''b''дb'' spindle-at-speed b''нb''b''ab''b''дb''b''сb''b''иb''b ↩
    ''лb''b''ab''b''єb''b''тb''b''ьb''b''сb''b''яb'' b''дb''b''ob'' spindle.0.at-speed
# b''іb'' b''кb''b''ob''b''лb''b''иb'' b''цb''b''eb'' b''сb''b''тb''b''ab''b''нb''b''eb''b' ↩
    'тb''b''ьb''b''сb''b''яb'', b''рb''b''уb''b''хb'' b''рb''b''ob''b''зb''b''пb''b''ob''b' ↩
    'чb''b''нb''b''eb''b''тb''b''ьb''b''сb''b''яb''
net spindle-ready <= spindle-at-speed.out => spindle.0.at-speed
```

7.2.6 Зворотній зв’язок шпинделя

7.2.6.1 Синхронізований рух шпинделя

Зворотний зв’язок шпинделя необхідний LinuxCNC для виконання будь-яких координованих рухів шпинделя, таких як нарізування різьби та постійна швидкість поверхні. LinuxCNC може виконувати синхронізовані рухи та CSS з будь-яким із 8 шпинделів. Які шпинделі використовуються, контролюєтьс з G-коду. CSS можливий з декількома шпинделями одночасно.

Майстер StepConf може виконати підключення для конфігурації з одним шпинделем, якщо ви виберете Фазу енкодеру A та Індекс енкодеру як вхідні дані.

Припущення щодо апаратного забезпечення для цього прикладу:

- До шпинделя підключений енкодер, який видає 100 імпульсів на оберт у фазі A.
- Фаза A енкодера підключена до контакту 10 паралельного порту.
- Індексний імпульс енкодера підключено до виводу 11 паралельного порту.

Основні кроки для додавання компонентів та їх налаштування: примітка:[У цьому прикладі ми припустимо, що деякі енкодери вже були видані осям/з’єднанням 0, 1 та 2. Отже, наступним енкодером, який ми можемо приєднати до шпинделя, буде номер 3. Ваша ситуація може відрізнятися.] примітка:[Індекс-активація енкодера HAL є винятком із правила, оскільки він працює як вхід і вихід, детальніше див. Encoder Section] примітка:[Оскільки вище ми вибрали «неквадратурне просте підрахування...», ми можемо обійтися «квадратурним» підрахунком без будь-якого квадратурно входу B.]

```
# b''Дb''b''ob''b''дb''b''ab''b''йb''b''тb''b''eb'' b''кb''b''ob''b''дb''b''eb''b''pb'' b' ↩
    'дb''b''ob'' HAL b''тb''b''ab'' b''пb''b''pb''b''иb''b''єb''b''дb''b''нb''b''ab''b''йb'' ↩
    b''тb''b''eb'' b''йb''b''ob''b''гb''b''ob'' b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b ↩
    ''кb''b''ib''b''вb''.
loadrt b''кb''b''ob''b''дb''b''eb''b''pb'' num_chan=4
addf encoder.update-counters base-thread
addf encoder.capture-position servo-thread

# b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''eb''b''нb''b' ↩
    'кb''b''ob''b''дb''b''eb''b''pb'' HAL b''нb''b''ab'' 100 b''ib''b''мb''b''пb''b''уb''b' ↩
    'лb''b''ьb''b''cb''b''ib''b''вb'' b''нb''b''ab'' b''ob''b''бb''b''eb''b''pb''b''тb''.
setp encoder.3.position-scale 100

# b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''кb''b''ob''b' ↩
    'дb''b''eb''b''pb'' HAL b''нb''b''ab'' b''нb''b''eb''b''кb''b''вb''b''ab''b''дb''b''pb'' ↩
    b''ab''b''тb''b''уb''b''pb''b''нb''b''иb''b''йb'' b''пb''b''pb''b''ob''b''cb''b''тb''b' ↩
    'иb''b''йb'' b''pb''b''ab''b''xb''b''уb''b''нb''b''ob''b''кb'', b''вb''b''иb''b''кb''b' ↩
    'ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''юb''b''чb''b''иb'' b''лb''b' ↩
    'иb''b''шb''b''eb'' A.
setp encoder.3.counter-mode true

# b''Пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''вb''b''иb''b' ↩
    'xb''b''ob''b''дb''b''иb'' HAL-b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb''b''ab'' ↩
    b''дb''b''ob'' 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

# b''Пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''вb''b''xb''b' ↩
    'ob''b''дb''b''иb'' HAL-b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''pb''b''ab'' b''дb'' ↩
    b''ob'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''eb''b''нb''b' ↩
    'кb''b''ob''b''дb''b''eb''b''pb''b''ab''.
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 на швидкості

Щоб LinuxCNC міг чекати, поки шпиндель досягне необхідної швидкості, перш ніж виконувати серію рухів, параметр spindle.*N*.at-speed повинен стати true в момент, коли шпиндель досягне заданої швидкості. Для цього необхідна зворотний зв'язок від енкодера шпинделя. Оскільки зворотний зв'язок і задана швидкість зазвичай не збігаються «точно», слід використовувати компонент «near», щоб визначити, чи достатньо близькі ці два числа.

Необхідні з'єднання: від сигналу команди швидкості шпинделя до near.n.in1 і від швидкості шпинделя від енкодера до near.n.in2. Потім near.n.out підключається до spindle.*N*.at-speed. near.n.scale потрібно налаштувати, щоб вказати, наскільки близькими мають бути ці два числа перед увімкненням виходу. Залежно від ваших налаштувань, можливо, доведеться відрегулювати шкалу, щоб вона працювала з вашим обладнанням.

Нижче наведено типові доповнення, які необхідно внести до файлу HAL, щоб увімкнути функцію Spindle At Speed. Якщо у файлі HAL вже є near, збільште кількість і відкоригуйте код відповідно. Перевірте, чи назви сигналів у файлі HAL збігаються.

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''кb''b' ↩
    'ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'' near b''тb''b''ab'' b''пb''b' ↩
    'рb''b''иb''b''єb''b''дb''b''нb''b''ab''b''тb''b''иb'' b''йb''b''ob''b''гb''b''ob'' b' ↩
    'дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb''
loadrt near
addf near.0 servo-thread

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''ob''b''дb''b' ↩
    'иb''b''нb'' b''вb''b''хb''b''ib''b''дb'' b''дb''b''ob'' b''зb''b''ab''b''дb''b''ab''b' ↩
    'нb''b''ob''b''їb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ob''b''cb''b''тb''b''ib'' b' ↩
    'шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb''
net spindle-cmd => near.0.in1

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''ob''b''дb''b' ↩
    'иb''b''нb'' b''вb''b''хb''b''ib''b''дb'' b''дb''b''ob'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''cb''b''тb''b''ib'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b' ↩
    'яb'', b''вb''b''иb''b''мb''b''ib''b''рb''b''яb''b''нb''b''ob''b''їb'' b''eb''b''нb''b' ↩
    'кb''b''ob''b''дb''b''eb''b''рb''b''ob''b''мb''
net spindle-velocity => near.0.in2

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''вb''b''иb''b' ↩
    'хb''b''ib''b''дb'' b''дb''b''ob'' b''вb''b''хb''b''ob''b''дb''b''уb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''нb''b''ab'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''cb''b''тb''b''ib''
net spindle-at-speed spindle.0.at-speed <= near.0.out

# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''вb''b''хb''b' ↩
    'ib''b''дb''b''нb''b''ib'' b''дb''b''ab''b''нb''b''ib'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''ob''b''cb''b''тb''b''ib'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b' ↩
    'яb'' b''тb''b''ab''b''кb'', b''щb''b''ob''b''бb'' b''вb''b''ob''b''нb''b''иb'' b''уb''b ↩
    ''зb''b''гb''b''ob''b''дb''b''жb''b''уb''b''вb''b''ab''b''лb''b''иb''b''cb''b''яb'', b' ↩
    'яb''b''кb''b''щb''b''ob'' b''вb''b''ob''b''нb''b''иb'' b''зb''b''нb''b''ab''b''хb''b' ↩
    'ob''b''дb''b''яb''b''тb''b''ьb''b''cb''b''яb'' b''вb'' b''мb''b''eb''b''жb''b''ab''b' ↩
    'хb'' 1%
setp near.0.scale 1.01
```

7.3 Підвіска MPG

Цей приклад пояснює, як підключити поширені на сьогоднішньому ринку підвісні пристрої MPG. У цьому прикладі використовується підвісний пристрій MPG3 та інтерфейсна карта C22 від CNC4PC, підключена до другого паралельного порту, вставленого в слот PCI. Цей приклад надає вам 3 осі з 3-кроковими приростами 0,1, 0,01, 0,001

У файлі custom.hal або jog.hal додайте наступне, переконавшись, що mux4 або кодер ще не використовуються. Якщо використовуються, просто збільште кількість і змініть номери посилань. Більше інформації про mux4 і кодер можна знайти в посібнику HAL або на сторінці man.

Докладнішу інформацію про додавання файлу HAL див. у розділі INI HAL Section документації. Для кожного з'єднання та всіх координатних літер передбачені контакти управління рухом. У цьому прикладі для руху в режимі світового простору використовуються контакти осі руху. Машини з неідентичною кінематикою можуть потребувати додаткових з'єднань для руху в режимі з'єднання.

jog.hal

```
# b''Пb''b''ib''b''дb''b''вb''b''ib''b''cb''b''кb''b''ab'' Jog
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

# b''Яb''b''кb''b''щb''b''ob'' b''вb''b''ab''b''шb'' MPG b''вb''b''иb''b''дb''b''ab''b' ↩
    'єb'' b''кb''b''вb''b''ab''b''дb''b''pb''b''ab''b''тb''b''уb''b''pb''b''нb''b''иb''b' ↩
    'йb'' b''cb''b''иb''b''гb''b''нb''b''ab''b''лb'' b''нb''b''ab'' b''кb''b''лb''b''ab''b' ↩
    'цb''b''ab''b''нb''b''нb''b''яb'', b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b' ↩
    'ib''b''тb''b''ьb'' x4 b''нb''b''ab'' 1
# b''Яb''b''кb''b''щb''b''ob'' b''вb''b''ab''b''шb'' MPG b''вb''b''иb''b''дb''b''ab''b' ↩
    'єb'' 1 b''ib''b''мb''b''пb''b''уb''b''лb''b''ьb''b''cb'' b''нb''b''ab'' b''кb''b''лb''b ↩
    ''ab''b''цb''b''ab''b''нb''b''нb''b''яb'', b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b' ↩
    'вb''b''ib''b''тb''b''ьb'' x4 b''нb''b''ab'' 0
setp encoder.0.x4-mode 0

# b''Дb''b''лb''b''яb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''уb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''ob''b''cb''b''тb''b''ib'' b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b' ↩
    'вb''b''ib''b''тb''b''ьb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' 1.
# b''Уb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''ib'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'ob''b''cb''b''тb''b''ib'' b''вb''b''ib''b''cb''b''ьb'' b''зb''b''уb''b''пb''b''иb''b' ↩
    'нb''b''яb''b''єb''b''тb''b''ьb''b''cb''b''яb'', b''кb''b''ob''b''лb''b''иb'' b''зb''b' ↩
    'уb''b''пb''b''иb''b''нb''b''яb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''цb''b''иb''b' ↩
    'фb''b''eb''b''pb''b''бb''b''лb''b''ab''b''тb'',
# b''нb''b''ab''b''вb''b''ib''b''тb''b''ьb'' b''яb''b''кb''b''щb''b''ob'' b''цb''b''eb'' b' ↩
    'ob''b''зb''b''нb''b''ab''b''чb''b''ab''b''єb'', b''щb''b''ob'' b''зb''b''ab''b''дb''b' ↩
    'ab''b''нb''b''иb''b''йb'' b''pb''b''уb''b''хb'' b''нb''b''eb'' b''зb''b''ab''b''вb''b' ↩
    'eb''b''pb''b''шb''b''eb''b''нb''b''ob''.
# b''Дb''b''лb''b''яb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''уb'' b''пb''b''ob''b''лb''b' ↩
    'ob''b''жb''b''eb''b''нb''b''нb''b''яb'' (b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b' ↩
    'вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'') b''вb''b''cb''b''тb''b' ↩
    'ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''зb''b''нb''b''ab''b''чb''b''eb''b' ↩
    'нb''b''нb''b''яb'' 0.
# b''Уb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''ib'' b''пb''b''ob''b''лb''b''ob''b''жb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''вb''b''ib''b''cb''b''ьb'' b''pb''b''уb''b''хb''b''ab''b' ↩
    'тb''b''иb''b''мb''b''eb''b''тb''b''ьb''b''cb''b''яb'' b''тb''b''ob''b''чb''b''нb''b' ↩
    'ob'' b''нb''b''ab'' b''вb''b''eb''b''лb''b''иb''b''чb''b''иb''b''нb''b''уb'' b''шb''b' ↩
    'кb''b''ab''b''лb''b''иb'' b''ib''b''мb''b''пb''b''уb''b''лb''b''ьb''b''cb''b''нb''b' ↩
    'ob''b''гb''b''ob'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b' ↩
    'нb''b''яb''
```

```
# b''зb''b''ab'' b''кb''b''ob''b''жb''b''нb''b''иb''b''мb'' b''іb''b''мb''b''пb''b''уb''b' ↩
    'лb''b''ьb''b''cb''b''ob''b''мb'', b''нb''b''еb''b''зb''b''ab''b''лb''b''еb''b''жb''b' ↩
    'нb''b''ob'' b''вb''b''іb''b''дb'' b''тb''b''ob''b''гb''b''ob'', b''cb''b''кb''b''іb''b' ↩
    'лb''b''ьb''b''кb''b''иb'' b''чb''b''ab''b''cb''b''уb'' b''цb''b''еb'' b''зb''b''ab''b' ↩
    'йb''b''мb''b''еb''.
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# b''Цb''b''еb'' b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''юb''b''єb'' b' ↩
    'мb''b''ab''b''cb''b''шb''b''тb''b''ab''b''бb'', b''яb''b''кb''b''иb''b''йb'' b''бb''b' ↩
    'уb''b''дb''b''еb'' b''вb''b''иb''b''кb''b''ob''b''рb''b''иb''b''cb''b''тb''b''ob''b' ↩
    'вb''b''уb''b''вb''b''ab''b''тb''b''иb''b''cb''b''яb'' b''нb''b''ab'' b''ob''b''cb''b' ↩
    'нb''b''ob''b''вb''b''іb'' b''вb''b''хb''b''іb''b''дb''b''нb''b''иb''b''хb'' b''дb''b' ↩
    'ab''b''нb''b''иb''b''хb'' mux4.
setp mux4.0.in0 0.1
setp mux4.0.in1 0.01
setp mux4.0.in2 0.001

# b''Bb''b''хb''b''ob''b''дb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''еb''b' ↩
    'нb''b''тb''b''ab'' mux4
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# b''Bb''b''иb''b''хb''b''іb''b''дb''b''нb''b''иb''b''йb'' b''cb''b''иb''b''гb''b''нb''b' ↩
    'ab''b''лb'' b''зb'' mux4 b''нb''b''ab''b''дb''b''cb''b''иb''b''лb''b''ab''b''єb''b' ↩
    'тb''b''ьb''b''cb''b''яb'' b''нb''b''ab'' b''кb''b''ob''b''жb''b''нb''b''уb'' b''ob''b' ↩
    'cb''b''ьb''b''ob''b''вb''b''уb'' b''шb''b''кb''b''ab''b''лb''b''уb'' b''пb''b''ob''b' ↩
    'шb''b''тb''b''ob''b''вb''b''хb''b''іb''b''вb''.
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

# b''Bb''b''хb''b''ob''b''дb''b''иb'' MPG
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# b''Bb''b''хb''b''ob''b''дb''b''иb'' b''вb''b''иb''b''бb''b''ob''b''рb''b''уb'' b''ob''b' ↩
    'cb''b''ib''
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

# b''Bb''b''иb''b''хb''b''іb''b''дb''b''нb''b''иb''b''йb'' b''cb''b''иb''b''гb''b''нb''b' ↩
    'ab''b''лb'' b''eb''b''нb''b''кb''b''ob''b''дb''b''eb''b''рb''b''ab'' b''пb''b''іb''b' ↩
    'дb''b''рb''b''ab''b''хb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'дb''b''ob'' b''ob''b''cb''b''іb''. b''Pb''b''уb''b''хb''b''ab''b''тb''b''иb''b''мb''b' ↩
    'eb''b''тb''b''ьb''b''cb''b''яb'' b''лb''b''иb''b''шb''b''eb'' b''вb''b''иb''b''бb''b' ↩
    'pb''b''ab''b''нb''b''ab'' b''вb''b''іb''b''cb''b''ьb''.
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
```

Якщо машина здатна на високе прискорення, щоб згладити рухи для кожного клацання MPG, використовуйте компонент посилання:../man/man9/ilowpass.9.html[ilowpass] для обмеження прискоре

jog.hal з 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

# b''Яb''b''кb''b''щb''b''оb'' b''вb''b''аb''b''шb'' MPG b''вb''b''иb''b''дb''b''аb''b' ↩
    'єb'' b''кb''b''вb''b''аb''b''дb''b''рb''b''аb''b''тb''b''уb''b''рb''b''нb''b''иb''b' ↩
    'йb'' b''сb''b''иb''b''гb''b''нb''b''аb''b''лb'' b''нb''b''аb'' b''кb''b''лb''b''аb''b' ↩
    'цb''b''аb''b''нb''b''нb''b''яb'', b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b' ↩
    'іb''b''тb''b''ьb'' x4 b''нb''b''аb'' 1
# b''Яb''b''кb''b''щb''b''оb'' b''вb''b''аb''b''шb'' MPG b''вb''b''иb''b''дb''b''аb''b' ↩
    'єb'' 1 b''іb''b''мb''b''пb''b''уb''b''лb''b''ьb''b''сb'' b''нb''b''аb'' b''кb''b''лb''b ↩
    ''аb''b''цb''b''аb''b''нb''b''нb''b''яb'', b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b' ↩
    'вb''b''іb''b''тb''b''ьb'' x4 b''нb''b''аb'' 0
setp encoder.0.x4-mode 0

# b''Дb''b''лb''b''яb'' b''рb''b''еb''b''жb''b''иb''b''мb''b''уb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''оb''b''сb''b''тb''b''іb'' b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b' ↩
    'вb''b''іb''b''тb''b''ьb'' b''зb''b''нb''b''аb''b''чb''b''еb''b''нb''b''нb''b''яb'' 1.
# b''Уb'' b''рb''b''еb''b''жb''b''иb''b''мb''b''іb'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'оb''b''сb''b''тb''b''іb'' b''вb''b''іb''b''сb''b''ьb'' b''зb''b''уb''b''пb''b''иb''b' ↩
    'нb''b''яb''b''єb''b''тb''b''ьb''b''сb''b''яb'', b''кb''b''оb''b''лb''b''иb'' b''зb''b' ↩
    'уb''b''пb''b''иb''b''нb''b''яb''b''єb''b''тb''b''ьb''b''сb''b''яb'' b''цb''b''иb''b' ↩
    'фb''b''еb''b''рb''b''бb''b''лb''b''аb''b''тb'',
# b''нb''b''аb''b''вb''b''іb''b''тb''b''ьb'' b''яb''b''кb''b''щb''b''оb'' b''цb''b''еb'' b' ↩
    'оb''b''зb''b''нb''b''аb''b''чb''b''аb''b''єb'', b''щb''b''оb'' b''зb''b''аb''b''дb''b' ↩
    'аb''b''нb''b''иb''b''йb'' b''рb''b''уb''b''хb'' b''нb''b''еb'' b''зb''b''аb''b''вb''b' ↩
    'еb''b''рb''b''шb''b''еb''b''нb''b''оb''.
# b''Дb''b''лb''b''яb'' b''рb''b''еb''b''жb''b''иb''b''мb''b''уb'' b''пb''b''оb''b''лb''b' ↩
    'оb''b''жb''b''еb''b''нb''b''нb''b''яb'' (b''зb''b''аb'' b''зb''b''аb''b''мb''b''оb''b' ↩
    'вb''b''чb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb''b''мb'') b''вb''b''сb''b''тb''b' ↩
    'аb''b''нb''b''оb''b''вb''b''іb''b''тb''b''ьb'' b''зb''b''нb''b''аb''b''чb''b''еb''b' ↩
    'нb''b''нb''b''яb'' 0.
# b''Уb'' b''рb''b''еb''b''жb''b''иb''b''мb''b''іb'' b''пb''b''оb''b''лb''b''оb''b''жb''b' ↩
    'еb''b''нb''b''нb''b''яb'' b''вb''b''іb''b''сb''b''ьb'' b''рb''b''уb''b''хb''b''аb''b' ↩
    'тb''b''иb''b''мb''b''еb''b''тb''b''ьb''b''сb''b''яb'' b''тb''b''оb''b''чb''b''нb''b' ↩
    'оb'' b''нb''b''аb'' b''вb''b''еb''b''лb''b''иb''b''чb''b''иb''b''нb''b''уb'' b''шb''b' ↩
    'кb''b''аb''b''лb''b''иb'' b''іb''b''мb''b''пb''b''уb''b''лb''b''ьb''b''сb''b''нb''b' ↩
    'оb''b''гb''b''оb'' b''пb''b''еb''b''рb''b''еb''b''мb''b''іb''b''щb''b''еb''b''нb''b' ↩
    'нb''b''яb''
# b''зb''b''аb'' b''кb''b''оb''b''жb''b''нb''b''иb''b''мb'' b''іb''b''мb''b''пb''b''уb''b' ↩
    'лb''b''ьb''b''сb''b''оb''b''мb'', b''нb''b''еb''b''зb''b''аb''b''лb''b''еb''b''жb''b' ↩
    'нb''b''оb'' b''вb''b''іb''b''дb'' b''тb''b''оb''b''гb''b''оb'', b''сb''b''кb''b''іb''b' ↩
    'лb''b''ьb''b''кb''b''иb'' b''чb''b''аb''b''сb''b''уb'' b''цb''b''еb'' b''зb''b''аb''b' ↩
    'йb''b''мb''b''еb''.
setp axis.x.jog-vel-mode 0
setp axis.y.jog-vel-mode 0
setp axis.z.jog-vel-mode 0

# b''Цb''b''еb'' b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''лb''b''юb''b''єb'' b' ↩
    'мb''b''аb''b''сb''b''шb''b''тb''b''аb''b''бb'', b''яb''b''кb''b''иb''b''йb'' b''бb''b' ↩
    'уb''b''дb''b''еb'' b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b' ↩
    'вb''b''уb''b''вb''b''аb''b''тb''b''иb''b''сb''b''яb'' b''нb''b''аb'' b''оb''b''сb''b' ↩
    'нb''b''оb''b''вb''b''іb'' b''вb''b''хb''b''іb''b''дb''b''нb''b''оb''b''гb''b''оb'' b' ↩
    'сb''b''иb''b''гb''b''нb''b''аb''b''лb''b''уb'' b''нb''b''аb'' mux4
# b''Вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''аb''b''нb''b''иb''b''йb'' b' ↩
    'тb''b''уb''b''тb'' b''мb''b''аb''b''сb''b''шb''b''тb''b''аb''b''бb'' b''мb''b''аb''b' ↩
```

```
    ’єb’’ b’’6b’’b’’yb’’b’’тb’’b’’иb’’ b’’пb’’b’’ob’’b’’мb’’b’’нb’’b’’ob’’b’’жb’’b’’eb’’b’ ↩
    ’нb’’b’’иb’’b’’йb’’ b’’нb’’b’’ab’’ b’’мb’’b’’ab’’b’’cb’’b’’шb’’b’’тb’’b’’ab’’b’’6b’’  ↩
    ilowpass
setp mux4.0.in0 0.0001
setp mux4.0.in1 0.00001
setp mux4.0.in2 0.000001

# b’’Bb’’b’’xb’’b’’ob’’b’’дb’’b’’иb’’ b’’кb’’b’’ob’’b’’мb’’b’’пb’’b’’ob’’b’’нb’’b’’eb’’b’ ↩
    ’нb’’b’’тb’’b’’ab’’ mux4
net scale1 mux4.0.sel0 <= parport.1.pin-09-in
net scale2 mux4.0.sel1 <= parport.1.pin-10-in

# b’’Bb’’b’’иb’’b’’xb’’b’’ib’’b’’дb’’b’’нb’’b’’ib’’ b’’дb’’b’’ab’’b’’нb’’b’’ib’’ b’’зb’’ b’ ↩
    ’лb’’b’’ib’’b’’чb’’b’’иb’’b’’лb’’b’’ьb’’b’’нb’’b’’иb’’b’’кb’’b’’ib’’b’’вb’’ b’’кb’’b’ ↩
    ’ob’’b’’дb’’b’’yb’’b’’вb’’b’’ab’’b’’лb’’b’’ьb’’b’’нb’’b’’иb’’b’’кb’’b’’ab’’ b’’нb’’b’ ↩
    ’ab’’b’’дb’’b’’cb’’b’’иb’’b’’лb’’b’’ab’’b’’юb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’дb’’b’ ↩
    ’ob’’ ilowpass
net mpg-out ilowpass.0.in <= encoder.0.counts

# b’’Bb’’b’’иb’’b’’xb’’b’’ib’’b’’дb’’b’’нb’’b’’иb’’b’’йb’’ b’’cb’’b’’иb’’b’’гb’’b’’нb’’b’ ↩
    ’ab’’b’’лb’’ b’’зb’’ mux4 b’’нb’’b’’ab’’b’’дb’’b’’cb’’b’’иb’’b’’лb’’b’’ab’’b’’єb’’b’ ↩
    ’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’нb’’b’’ab’’ b’’кb’’b’’ob’’b’’жb’’b’’нb’’b’’yb’’ b’’ob’’b’ ↩
    ’cb’’b’’ьb’’b’’ob’’b’’вb’’b’’yb’’ b’’шb’’b’’кb’’b’’ab’’b’’лb’’b’’yb’’ b’’пb’’b’’ob’’b’ ↩
    ’шb’’b’’тb’’b’’ob’’b’’вb’’b’’xb’’b’’ib’’b’’вb’’.
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

# b’’Bb’’b’’xb’’b’’ob’’b’’дb’’b’’иb’’ MPG
net mpg-a encoder.0.phase-A <= parport.1.pin-02-in
net mpg-b encoder.0.phase-B <= parport.1.pin-03-in

# b’’Bb’’b’’xb’’b’’ob’’b’’дb’’b’’иb’’ b’’вb’’b’’иb’’b’’6b’’b’’ob’’b’’pb’’b’’yb’’ b’’ob’’b’ ↩
    ’cb’’b’’ib’’
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

# b’’Bb’’b’’иb’’b’’xb’’b’’ib’’b’’дb’’ b’’зb’’ ilowpass b’’нb’’b’’ab’’b’’дb’’b’’cb’’b’’иb’’b ↩
    ’’лb’’b’’ab’’b’’єb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’нb’’b’’ab’’ b’’кb’’b’’ob’’b’’жb’’b’ ↩
    ’нb’’b’’yb’’ b’’вb’’b’’ib’’b’’cb’’b’’ьb’’ jog count
# b’’Pb’’b’’yb’’b’’xb’’b’’ab’’b’’тb’’b’’иb’’b’’мb’’b’’eb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’ ↩
    ’лb’’b’’иb’’b’’шb’’b’’eb’’ b’’вb’’b’’иb’’b’’6b’’b’’pb’’b’’ab’’b’’нb’’b’’ab’’ b’’вb’’b’ ↩
    ’ib’’b’’cb’’b’’ьb’’.
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 Шпиндель GS2

7.4.1 Приклад

У цьому прикладі показано підключення, необхідні для використання частотного перетворювача Automation Direct GS2 для керування шпинделем. Швидкість і напрямок обертання шпинделя контролюються LinuxCNC.

Використання компонента GS2 вимагає мінімальних налаштувань. Почнемо з конфігурації, створеної за допомогою майстра StepConf Wizard. Переконайтеся, що контакти «Spindle CW» і «Spindle PWM» встановлені в положення «невикористовується» на екрані налаштувань паралельного порту.

У файлі custom.hal ми розміщуємо наступний код для підключення LinuxCNC до GS2 та керування приводом за допомогою LinuxCNC.

Приклад GS2

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''кb''b' ↩
    'ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'', b''щb''b''ob'' b''нb''b''eb'' b' ↩
    'пb''b''pb''b''ab''b''цb''b''юb''b''єb'' b''вb'' b''pb''b''eb''b''жb''b''иb''b''мb''b' ↩
    'ib'' b''pb''b''eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''чb''b''ab''b' ↩
    'cb''b''yb'', b''дb''b''лb''b''яb'' b''чb''b''ab''b''cb''b''тb''b''ob''b''тb''b''нb''b' ↩
    'иb''b''xb'' b''пb''b''eb''b''pb''b''eb''b''тb''b''вb''b''ob''b''pb''b''юb''b''вb''b' ↩
    'ab''b''чb''b''ib''b''вb'' Automation Direct GS2
loadusr -Wn spindle-vfd gs2_vfd -r 9600 -p none -s 2 -n spindle-vfd

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''шb''b''тb''b' ↩
    'иb''b''фb''b''тb'' b''нb''b''ab''b''пb''b''pb''b''яb''b''мb''b''кb''b''yb'' b''шb''b' ↩
    'пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''дb''b''ob'' GS2
net gs2-fwd spindle-vfd.spindle-fwd <= spindle.N.forward

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''ьb'' b''нb''b''ab'' b''шb''b''тb''b''иb''b''фb''b' ↩
    'тb''b''ib'' b''дb''b''ob'' GS2
net gs2-run spindle-vfd.spindle-on <= spindle.N.on

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' GS2 b''нb''b''ab'' ↩
     b''шb''b''вb''b''иb''b''дb''b''кb''b''ob''b''cb''b''тb''b''ib'' b''дb''b''ob'' b''pb''b ↩
    ''yb''b''xb''b''yb'' b''нb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ob''b''cb''b' ↩
    'тb''b''ib''
net gs2-at-speed spindle.N.at-speed <= spindle-vfd.at-speed

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''дb''b''ab''b' ↩
    'тb''b''чb''b''иb''b''кb'' b''ob''b''бb''b''eb''b''pb''b''тb''b''ib''b''вb'' b''шb''b' ↩
    'пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''дb''b''ob'' GS2
net gs2-RPM spindle-vfd.speed-command <= spindle.N.speed-out
```

Note
Швидкість передачі може бути вищою залежно від конкретних умов. Параметри диска та командного рядка повинні збігатися. Щоб перевірити наявність помилок передачі, додайте параметр командного рядка «-v» і запустіть програму з терміналу.

На самому приводі GS2 необхідно налаштувати кілька параметрів, перш ніж зв’язок Modbus почне працювати. Можливо, доведеться налаштувати й інші параметри відповідно до фізичних вимог, але це виходить за межі даного посібника. Докладнішу інформацію про параметри приводу див. у посібнику GS2, що додається до приводу.

- Перемикачі зв’язку повинні бути встановлені в положення RS-232C.
- Параметри двигуна повинні бути налаштовані відповідно до двигуна.
- P3.00 (Джерело команди керування) має бути встановлено на «Робота визначається інтерфейсом RS-485», 03 або 04.
- P4.00 (Джерело команди частоти) має бути встановлено на Частоту, визначену інтерфейсом зв’язку RS232C/RS485, 05.

- P9.01 (Швидкість передачі) має бути встановлено на 9600 бод, 01.
- P9.02 (Протокол зв’язку) має бути встановлено на «Режим Modbus RTU, 8 бітів даних, без парності, 2 стоп-біти», 03.

Панель PyVCP, заснована на цьому прикладі, має вигляд here.

Chapter 8

Класична драбина

8.1 Вступ до ClassicLadder

8.1.1 Історія

ClassicLadder — це безкоштовна реалізація інтерпретатора сходових схем, випущена під ліцензією LGPL. Її автор — Марк Ле Дуарен.

Він описує початок проєкту на своєму вебсайті:

> Я вирішив запрограмувати мову драбини спочатку тільки для тестових цілей, у лютому 2001 року. Було заплановано, що я повинен буду брати участь у створенні нового продукту після того, як залишу підприємство, на якому працював на той час. І я думав, що наявність мови Ladder у цих продуктах може бути гарним варіантом, який варто врахувати. Тож я почав писати перші рядки коду для обчислення сходинки з мінімальними елементами та динамічного відображення її в Gtk, щоб перевірити, чи працює моя перша ідея щодо реалізації всього цього.
>
> І оскільки я швидко виявив, що це досить добре просувається, я продовжив працювати зі складнішими елементами: таймером, кратними сходинками тощо…
>
> Вуаля, ось ця робота… і навіть більше: я продовжую додавати нові функції з того часу.
>
> — Марк Ле Дуарен, з ”Genesis” на вебсайті ClassicLadder

ClassicLadder було адаптовано для роботи з HAL від LinuxCNC та наразі розповсюджується разом із LinuxCNC. Якщо виникнуть /проблеми/помилки, будь ласка, повідомте про них проєкту LinuxCNC.

8.1.2 Вступ

Ляддерна логіка або мова програмування Ladder — це метод малювання електричних логічних схем. Зараз це графічна мова, дуже популярна для програмування програмованих логічних контролерів (PLCs). Спочатку вона була винайдена для опису логіки, побудованої на реле. Назва походить від того, що програми на цій мові нагадують драбини з двома вертикальними «рейками» і серією горизонтальних «сходинок» між ними. У Німеччині та інших країнах Європи рейли зазвичай малюють горизонтально вздовж верхньої та нижньої частин сторінки, а сходинки — вертикально зліва направо.

Програма в логіці драбини, також звана діаграмою драбини, схожа на схему набору релейних ланцюгів. Логіка драбини корисна тим, що завдяки своїй схожості її можуть зрозуміти і використовувати без додаткового навчання інженери та технічні фахівці різних спеціальностей.

Ланцюгова логіка широко використовується для програмування PLCs, де необхідне послідовне управління процесом або виробничою операцією. Ланцюгова логіка корисна для простих, але критично важливих систем управління або для переробки старих дротових релейних схем. У міру вдосконалення програмованих логічних контролерів вона також стала використовуватися в дуже складних системах автоматизації.

Логіку драбини можна розглядати як мову, засновану на правилах, а не як процедурну мову. «Сходинка» в драбині представляє правило. При реалізації за допомогою реле та інших електромеханіч пристроїв різні правила «виконуються» одночасно і негайно. При реалізації в програмованому логічному контролері правила зазвичай виконуються послідовно за допомогою програмного забезпече в циклі. Швидке виконання циклу, зазвичай багато разів на секунду, забезпечує ефект одночасного і негайного виконання.

Логіка сходів виконує такі загальні кроки для роботи.

- Зчитування вхідних даних
- Розв’яжіть логіку
- Оновити виходи

8.1.3 Приклад

Найпоширенішими компонентами сходової схеми є контакти (входи), зазвичай вони бувають або NC (нормально замкнуті), або NO (нормально розімкнуті), та котушки (виходи).

- нормально-поворотний контакт 
- контакт NC 
- котушка (вихід) 

Звичайно, повноцінна мова сходинок містить багато інших компонентів, але розуміння їх допоможе вам зрозуміти загальну концепцію.

Драбина складається з однієї або кількох сходинок. Ці сходинки є горизонтальними доріжками (що представляють дроти) з компонентами на них (входами, виходами та іншими), які оцінюються зліва направо.

Цей приклад є найпростішим сходинкою:



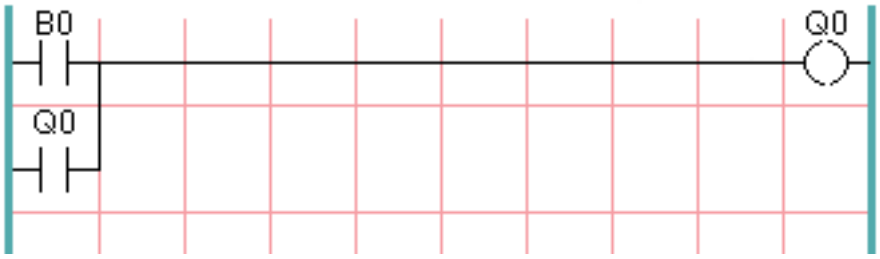
Вхід зліва, B0, нормально відкритий контакт, підключений до котушки (виходу) праворуч, Q0. Тепер уявіть, що до крайнього лівого кінця подається напруга, оскільки вхід B0 стає істинним (наприклад, вхід активовано або користувач натиснув контакт NO). Напруга має прямий шлях до котушки (виходу) праворуч, Q0. Як наслідок, котушка Q0 (вихід) перейде з 0/вимкнено/неправда до 1/увімкнено/правда. Якщо користувач відпустить B0, вихід Q0 швидко повернеться до 0/вимкнено/н

8.1.4 Базова схема фіксації ввімкнення/вимкнення

На основі наведеного вище прикладу припустимо, що ми додаємо перемикач, який замикається, коли котушка Q0 активна. Так відбувається в реле, де котушка може активувати контакти перемикача, або в контакторі, де часто є кілька невеликих допоміжних контактів на додаток до великих трифазних контактів, які є основною особливістю контактора.

Оскільки цей допоміжний перемикач приводиться в дію котушкою Q0 у нашому попередньому прикладі, ми надамо йому той самий номер, що й котушці, яка його приводить у дію. Це стандартна практика, якої дотримуються у всьому програмуванні релейних схем, хоча спочатку може здатися дивним бачити перемикач з тим самим номером, що й котушка. Тож назвемо цей допоміжний контакт Q0 і підключимо його до контакту «кнопки» B0 з нашого попереднього прикладу.

Давайте розглянемо це:

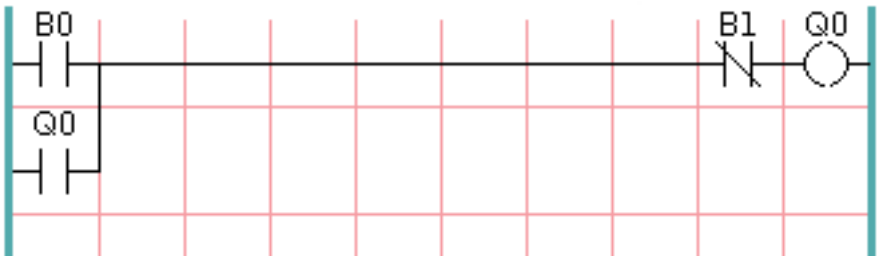


Як і раніше, коли користувач натискає кнопку B0, котушка Q0 вмикається. А коли котушка Q0 вмикається, вмикається перемикач Q0. Тепер відбувається цікава частина. Коли користувач відпускає кнопку B0, котушка Q0 не зупиняється, як це було раніше. Це відбувається тому, що перемикач Q0 цієї схеми фактично утримує кнопку користувача натиснутою. Отже, ми бачимо, що перемикач Q0 все ще утримує котушку Q0 у включеному стані після відпускання кнопки «старт».

Цей тип контакту на котушці або реле, що використовується таким чином, часто називають «утримуючим контактом», оскільки він «утримує» котушку, з якою він пов’язаний. Іноді його також називають «герметичним» контактом, і коли він активний, кажуть, що ланцюг «герметичний».

На жаль, наша схема поки що має мало практичного застосування, оскільки, хоча ми маємо кнопку «включення» або «запуску» у вигляді кнопки B0, у нас немає можливості вимкнути цю схему після її запуску. Але це легко виправити. Все, що нам потрібно, це спосіб перервати подачу живлення на котушку Q0. Тому додамо нормально закриту (NC) кнопку безпосередньо перед котушкою Q0.

Ось як це виглядатиме:



Тепер ми додали кнопку «вимкнути» або «зупинити» B1. Якщо користувач натискає її, контакт від ланки до котушки розривається. Коли котушка Q0 втрачає живлення, вона опускається до 0/вимкнено/неправда. Коли котушка Q0 вимикається, вимикається і перемикач Q0, тому «утримуючий контакт» розривається, або ланцюг «розривається». Коли користувач відпускає кнопку «стоп», контакт від щаблі до котушки Q0 відновлюється, але щабель вимкнений, тому котушка не вмикається знову.

Ця схема використовується вже протягом десятиліть практично на всіх машинах, що мають трифазний двигун, керований контактором, тому було неминуче, що її почнуть застосовувати програмісти драбинок/ПЛК. Це також дуже безпечна схема, оскільки якщо одночасно натиснути кнопки «старт» і «стоп», завжди перемагає функція «стоп».

Це основний будівельний блок більшої частини програмування рейкових схем, тому, якщо ви новачок у ньому, вам варто переконатися, що ви розумієте, як працює ця схема.

8.2 Програмування ClassicLadder

8.2.1 Концепції сходів

ClassicLadder — це тип мови програмування, спочатку реалізований на промислових PLC (він називається Ladder Programming). Він базується на концепції релейних контактів і котушок і може використовуватися для побудови логічних перевірок і функцій у спосіб, звичний для багатьох системних інтеграторів. Ladder складається з щаблів, які можуть мати розгалуження і нагадують електричну схему. Важливо знати, як оцінюються програми Ladder під час виконання.

Здається природним, що кожна рядок буде оцінюватися зліва направо, потім наступний рядок вниз і т. д., але в логіці драбини це працює не так. Логіка драбини «сканує» сходинки драбини 3 рази, щоб змінити стан виходів.

- вхідні дані зчитуються та оновлюються
- логіка зрозуміла
- виходи встановлені

Спочатку це може заплутати, якщо вихід одного рядка зчитується входом іншого рядка. Після встановлення виходу буде проведено одне сканування, перш ніж другий вхід стане істинним.

Ще одна проблема з програмуванням рейок — це правило «Останній перемагає». Якщо у вас однаковий вихід у різних місцях вашої рейок, стан останнього буде таким, яким встановлений вихід.

8.2.2 Мови

Найпоширенішою мовою, що використовується під час роботи з ClassicLadder, є «ladder». ClassicLadder також підтримує послідовні функціональні схеми (Grafcet).

8.2.3 Компоненти

ClassicLadder складається з двох компонентів.

- Модуль реального часу classicladder_rt
- Модуль classicladder, що не працює в реальному часі (включаючи графічний інтерфейс)

8.2.3.1 Файли

Зазвичай компоненти ClassicLadder розміщуються у файлі custom.hal, якщо ви працюєте з конфігураці
згенерованою StepConf. Їх не можна розміщувати у файлі custom_postgui.hal, інакше меню Ladder Editor буде неактивним.

Note
Файли сходинок (.clp) не повинні містити пробілів в назві.

8.2.3.2 Модуль реального часу

Завантаження модуля ClassicLadder реального часу (classicladder_rt) можливе з файлу HAL або безпосередньо за допомогою команди halcmd. Перший рядок завантажує модуль ClassicLadder реального часу. Другий рядок додає функцію classicladder.0.refresh до сервопотоку. Цей рядок змушує ClassicLadder оновлюватися зі швидкістю сервопотоку.

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Швидкість потоку, в якому працює ClassicLadder, безпосередньо впливає на швидкість реагування на вхідні та вихідні дані. Якщо ви можете вмикати та вимикати перемикач швидше, ніж ClassicLadder може це помітити, то, можливо, вам потрібно пришвидшити потік. Найшвидша швидкість, з якою ClassicLadder може оновлювати сходинки, становить одну мілісекунду. Ви можете помістити його в швидший потік, але він не буде оновлюватися швидше. Якщо ви помістите його в потік, повільніший за одну мілісекунду, ClassicLadder буде оновлювати щаблі повільніше. Поточний час сканування буде відображатися на дисплеї секції, він округлюється до мікросекунд. Якщо час сканування перевищує одну мілісекунду, можливо, вам доведеться скоротити драбину або помістити її в повільніший потік.

8.2.3.3 Змінні

Під час завантаження модуля ClassicLadder в режимі реального часу можна налаштувати кількість об’єктів кожного типу. Якщо ви не налаштуєте кількість об’єктів, ClassicLadder використовуватиме значення за замовчуванням.

Table 8.1: Кількість змінних за замовчуванням

Назва об’єкта	**Назва змінної**	**Значення за замовчуванням**
Кількість сходинок	(numRungs)	100
Кількість бітів	(numBits)	20
Кількість змінних у словах	(numWords)	20
Кількість таймерів	(numTimers)	10
Кількість таймерів IEC	(numTimersIec)	10
Кількість моностабільних	(numMonostables)	10
Кількість лічильників	(numCounters)	10
Кількість бітових контактів входів HAL	(numPhysInputs)	15
Кількість вихідних бітових контактів HAL	(numPhysOutputs)	15
Кількість арифметичних виразів	(numArithmExpr)	50
Кількість секцій	(numSections)	10
Кількість символів	(numSymbols)	Авто
Кількість входів S32	(numS32in)	10
Кількість виходів S32	(numS32out)	10
Кількість входів з плаваючою точкою	(numFloatIn)	10
Кількість виходів з плаваючим зчитуванням	(numFloatOut)	10

Найбільший інтерес представляють об’єкти numPhysInputs, numPhysOutputs, numS32in та numS32out.

Зміна цих чисел змінить кількість доступних бітових виводів HAL. numPhysInputs і numPhysOutputs контролюють кількість доступних бітових виводів HAL (увімкнено/вимкнено). numS32in і numS32out контролюють кількість доступних виводів HAL зі знаковими цілими числами (+- діапазон цілих чисел).

Наприклад (вам не потрібно змінювати все це, щоб змінити лише декілька):

```
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
```

Щоб завантажити кількість об’єктів за замовчуванням:

```
loadrt classicladder_rt
```

8.2.4 Завантаження модуля ClassicLadder, що не працює в реальному часі

Команди ClassicLadder HAL повинні бути виконані до завантаження графічного інтерфейсу користувач
інакше пункт меню Ladder Editor не буде працювати. Якщо ви використовували майстер налаштування
крокового двигуна, розмістіть всі команди ClassicLadder HAL у файлі custom.hal.

Щоб завантажити модуль, що не працює в реальному часі:

```
loadusr classicladder
```

Note
Можна завантажити лише один файл .clp. Якщо вам потрібно розділити драбину, використовуйте секції.

Щоб завантажити файл рейдингу:

```
loadusr classicladder myladder.clp
```

Варіанти завантаження ClassicLadder

- *--nogui* - (завантажується без редактора сходинок), зазвичай використовується після завершення налагодження.
- *--modbus_port=port* - (завантажує номер порту Modbus)
- *--modmaster* - (ініціалізує головний модуль MODBUS) має одночасно завантажувати програму рейкового шини, інакше TCP є портом за замовчуванням.
- *--modslave* - (ініціалізує ведомий пристрій MODBUS) тільки TCP

Щоб використовувати ClassicLadder з HAL без EMC:

```
loadusr -w classicladder
```

Параметр -w вказує HAL не закривати середовище HAL, доки не завершиться робота ClassicLadder.

Якщо спочатку завантажити програму логічних конструкцій з опцією *--nogui*, а потім знову завантажит
ClassicLadder без опцій, графічний інтерфейс відобразить останню завантажену програму логічних
конструкцій.

В AXIS ви можете завантажити графічний інтерфейс з редактора файлів/схем...

8.2.5 ClassicLadder GUI

Якщо ви завантажите ClassicLadder з графічним інтерфейсом, він відобразить два вікна: вікно відображення розділів та вікно керування розділами.

8.2.5.1 Менеджер розділів

Під час першого запуску ClassicLadder ви отримаєте порожнє вікно менеджера розділів.

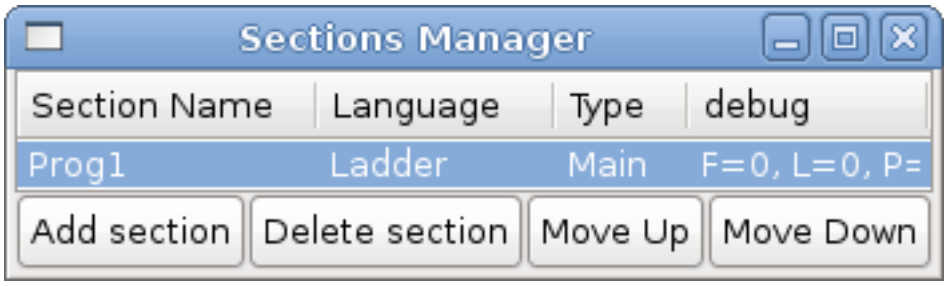


Figure 8.1: Вікно менеджера розділів за замовчуванням

Це вікно дозволяє вам називати, створювати або видаляти розділи та вибирати мову, яку використовува
цей розділ. Так само ви називаєте підпрограму для викликів котушок.

8.2.5.2 Відображення розділу

Під час першого запуску ClassicLadder ви побачите порожнє вікно відображення розділу. Відображає
один порожній рядок.

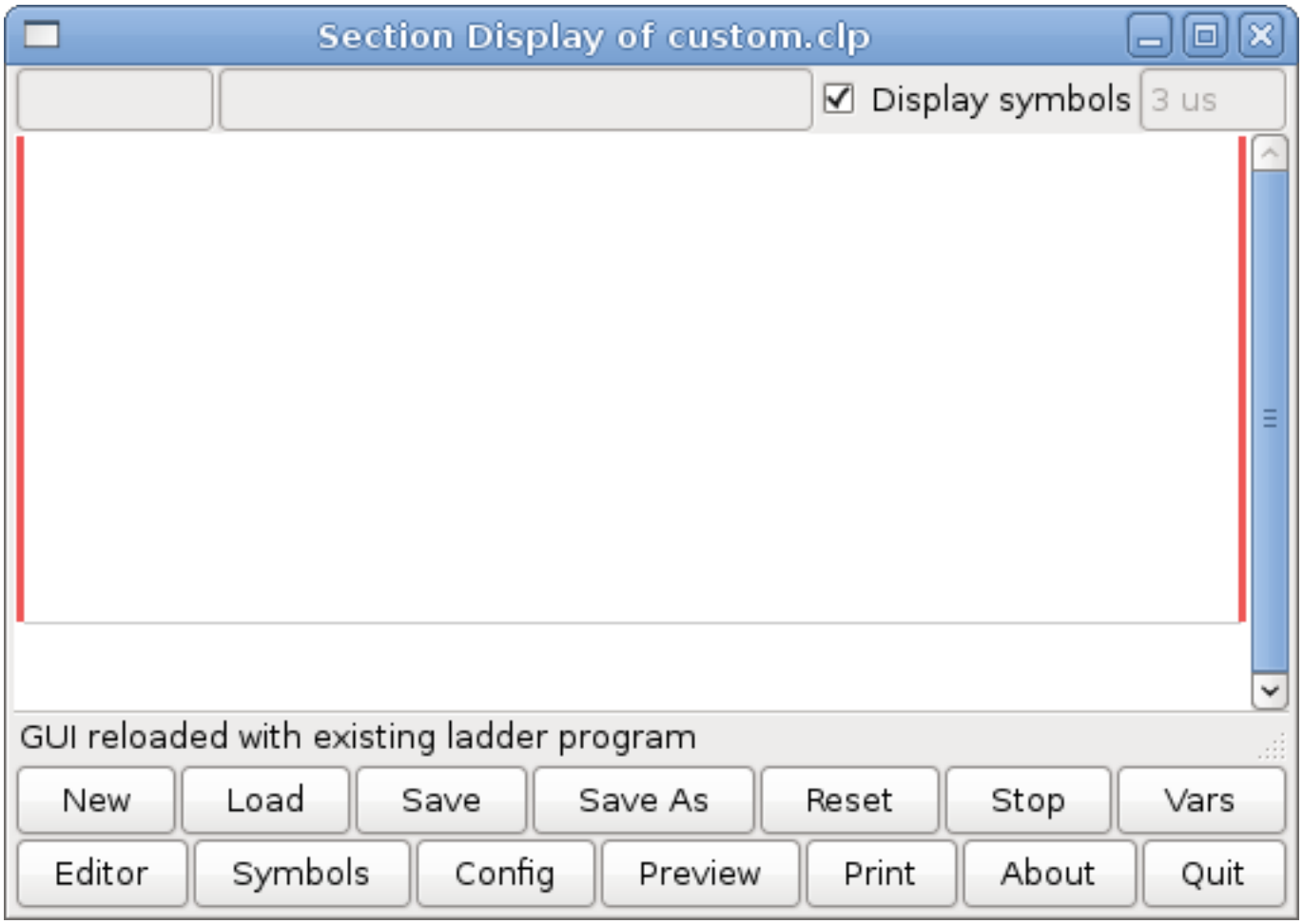


Figure 8.2: Вікно відображення розділу за замовчуванням

Більшість кнопок зрозумілі самі по собі:

Кнопка «Змінні» призначена для перегляду змінних, перемикайте її, щоб відображати одну, іншу, обидві, а потім жодної з вікон.

Кнопка «Налаштування» використовується для Modbus та показує максимальну кількість елементів рейкової логіки, які були завантажені модулем реального часу.

Кнопка «Символи» відобразить список символів для змінних, які можна редагувати (підказка: ви можете назвати входи, виходи, котушки тощо).

Кнопка «Вийти» завершить роботу програми, яка не працює в реальному часі, тобто Modbus та дисплей. Програма релейної логіки в реальному часі все ще працюватиме у фоновому режимі.

Прапорець у верхньому правому куті дозволяє вибрати, чи відображатимуться назви змінних, чи назви символів

Ви можете помітити, що під дисплеєм програми драбини є рядок із написом «Project failed to load...» (Не вдалося завантажити проект...). Це рядок стану, який надає інформацію про елементи програми драбини, на які ви натискаєте у вікні дисплея. Цей рядок стану тепер відображатиме назви сигналів HAL для змінних %I, %Q та першої %W (у рівнянні). Ви можете побачити деякі дивні мітки, такі як (103) у щаблях. Це відображається (навмисно) через стару помилку — під час стирання елементів старі версії іноді не стирали об’єкт із правильним кодом. Ви, можливо, помітили, що довга горизонтальна кнопка з’єднання іноді не працювала в старих версіях. Це було тому, що вона шукала «вільний» код, але знаходила щось інше. Число в

дужках — це нерозпізнаний код. Програма драбини все одно працюватиме належним чином, щоб виправити це, видаліть коди за допомогою редактора та збережіть програму.

8.2.5.3 Змінні вікна

Це два вікна змінних: вікно стану бітів (булеве) та вікно спостереження (ціле число зі знаком). Кнопка Vars знаходиться у вікні відображення розділу. Перемикайте кнопку Vars, щоб відобразити одне, інше, обидва або жодне з вікон змінних.

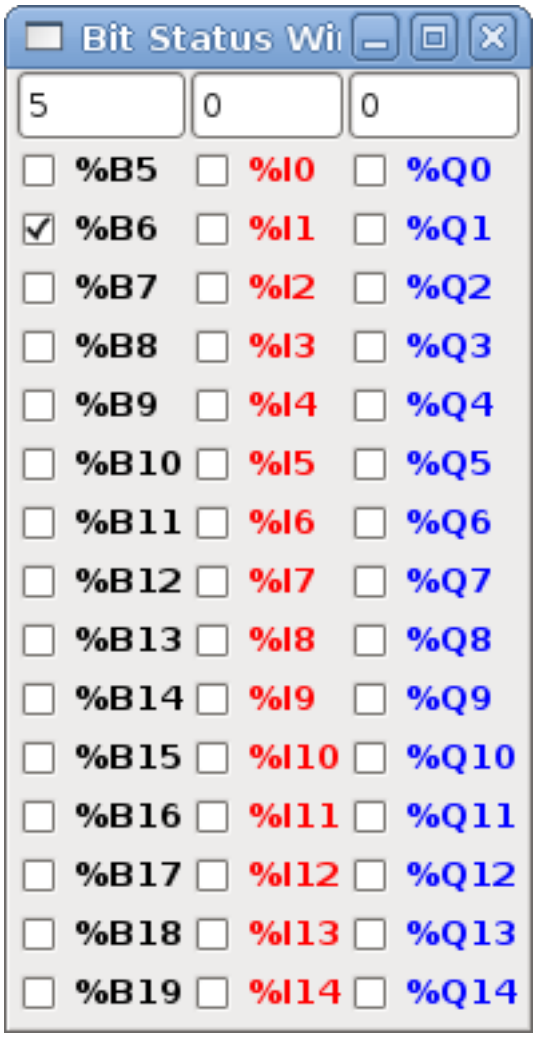


Figure 8.3: Вікно стану біта

Вікно стану бітів відображає деякі дані змінних типу «істина/хиба» (on/off). Зверніть увагу, що всі змінні починаються зі знака %. Змінні %I представляють вхідні бітові виводи HAL. Змінні %Q представляють котушку реле та вихідні бітові виводи HAL. Змінні %B представляють внутрішню котушку реле або внутрішній контакт. Три області редагування у верхній частині дозволяють вибрати 15 змінних, які будуть відображатися в кожному стовпці. Наприклад, якщо стовпець змінних %B має 15 записів, а ви ввели 5 у верхній частині стовпця, будуть відображатися змінні від %B5 до %B19. Прапорці дозволяють вручну встановлювати та скасовувати змінні %B, якщо програма релейної схеми не встановлює їх як виходи. Будь-які біти, які встановлені програмою як виходи під час роботи ClassicLadder, не можуть бути змінені і будуть відображатися як встановлені, якщо вони увімкнені, і як не встановлені, якщо вони вимкнені.

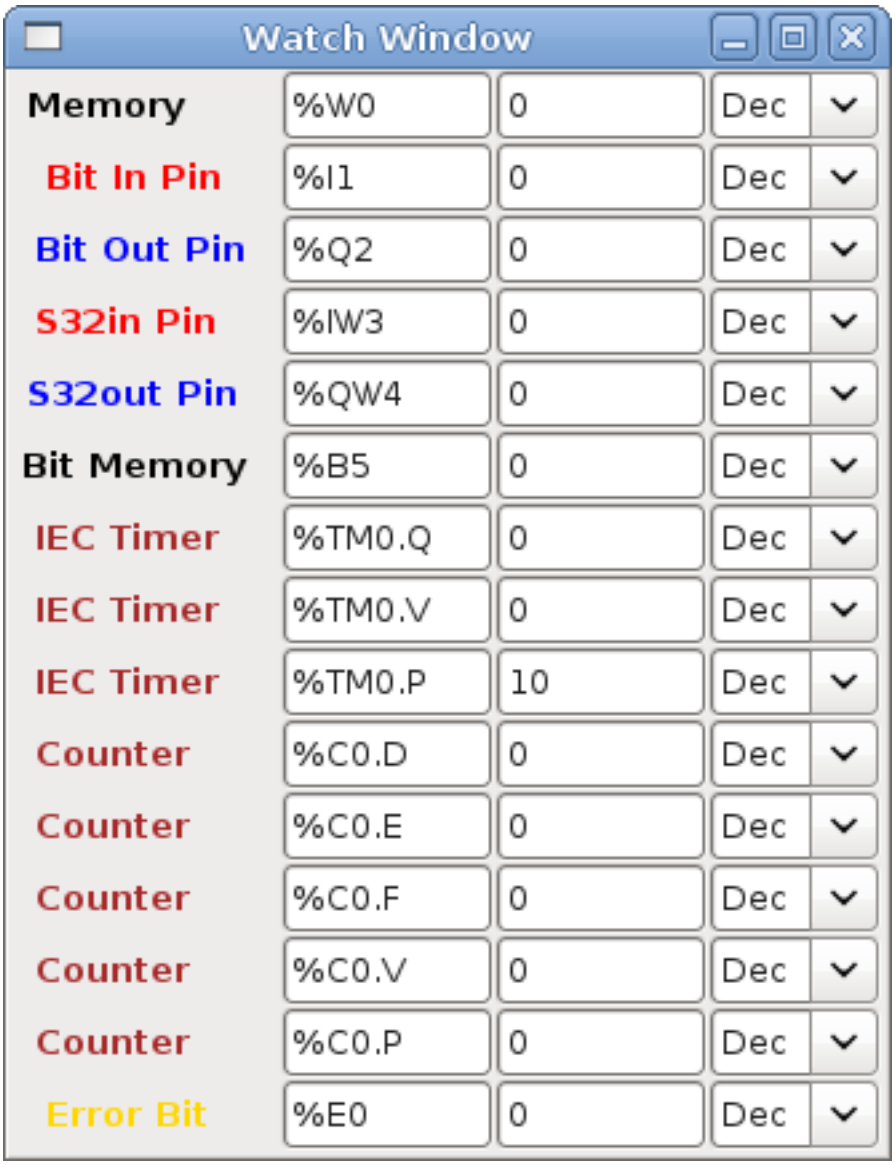


Figure 8.4: Вікно спостереження

Вікно спостереження відображає стан змінної. Поле редагування поруч із ним містить число, збережене в змінній, а випадаюче меню поруч із ним дозволяє вибрати, чи буде число відображатися в шістнадцятковій, десятковій чи двійковій системі числення. Якщо у вікні символів для змінних слів визначено імена символів і в розділі вікна відображення встановлено прапорець «відображати символи», імена символів будуть відображатися. Щоб змінити відображувану змінну, введіть номер змінної, наприклад %W2 (якщо прапорець «Відображати символи» не встановлено) або введіть ім’я символу (якщо прапорець «Відображати символи» встановлено) поверх існуючого номера/імені змінної та натисніть клавішу Enter.

8.2.5.4 Вікно символів

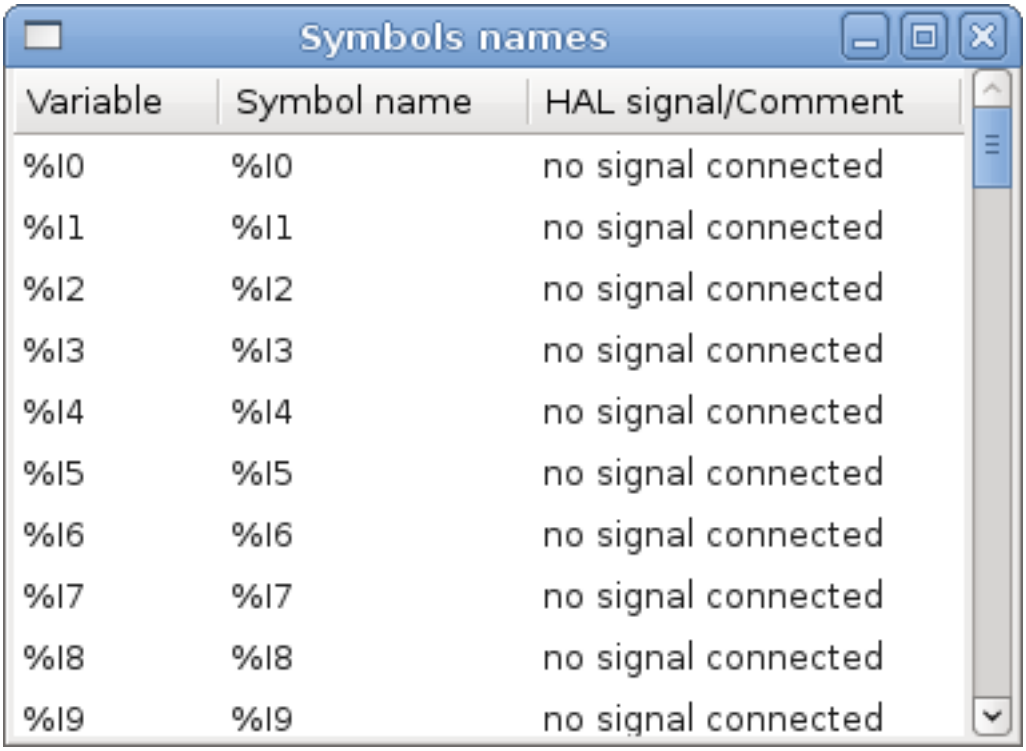


Figure 8.5: Вікно «Назви символів»

Це список імен «символів», які слід використовувати замість імен змінних, що відображаються у вікні розділу, коли встановлено прапорець «відображати символи». Ви додаєте ім'я змінної (пам'ятайте про символ «%» і великі літери), ім'я символу. Якщо до змінної може бути підключений сигнал HAL (%I, %Q і %W, якщо ви завантажили s32 pin з модулем реального часу), то в розділі коментарів буде показано поточне ім'я сигналу HAL або його відсутність. Імена символів повинні бути короткими, щоб краще відображатися. Майте на увазі, що ви можете відобразити довші імена сигналів HAL змінних %I, %Q і %W, натиснувши на них у вікні розділу. Між цими двома можна відстежувати, до чого підключена програма драбини!

8.2.5.5 Вікно редактора

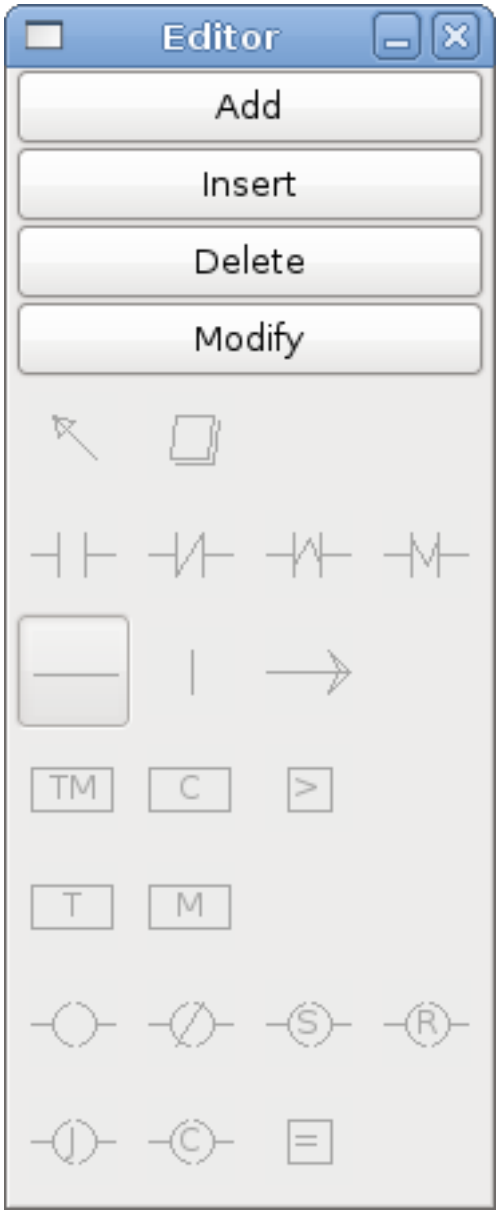


Figure 8.6: Вікно редактора

- *Add* - додає сходинку після вибраної сходинки
- *Insert* - вставляє сходинку перед вибраною сходинкою
- «Видалити» – видаляє вибраний рядок
- *Modify* - відкриває вибраний рядок для редагування

Починаючи з верхнього лівого зображення:

- Вибір об’єктів, Ластик
- Норморозімкнений вхід, нормальнорозімкнений вхід, вхід наростаючого фронту, вхід спадаючого фронту

- Горизонтальне підключення, вертикальне підключення, довге горизонтальне підключення
- Блок таймера IEC, блок лічильника, змінна порівняння
- Старий блок таймера, старий моностабільний блок (їх замінили на таймер IEC)
- КОТУШКИ - Норморозімкнений вихід, Норморозімкнений вихід, Вихід встановлення, Вихід скидання
- Котушка стрибка, котушка виклику, змінне призначення

Короткий опис кожної з кнопок:

- *Selector* - дозволяє вибирати існуючі об’єкти та змінювати інформацію про них.
- *Eraser* - стирає об’єкт.
- *N.O. Contact* - створює нормально відкритий контакт. Це може бути зовнішній контакт входу HAL-pin (%I), внутрішній бітовий контакт котушки (%B) або зовнішній контакт котушки (%Q). Контакт входу HAL-pin закритий, коли HAL-pin має значення true. Контакти котушки закриті, коли відповідна котушка активна (контакт %Q2 закривається, коли котушка %Q2 активна).
- *N.C. Contact* - створює нормально замкнутий контакт. Він такий самий, як і нормально-розмикаючий контакт, за винятком того, що контакт розімкнутий, коли контакт HAL увімкнений або котушка активна.
- *Rising Edge Contact* - створює контакт, який замикається, коли HAL-вивід переходить з положення «хибно» в «істина», або котушка переходить з неактивного в активний стан.
- *Falling Edge Contact* - створює контакт, який замикається, коли HAL-вивід переходить з активного стану в хибний або котушка переходить з активного у вимкнений.
- *Horizontal Connection* - створює горизонтальний зв’язок з об’єктами.
- *Vertical Connection* - створює вертикальне з’єднання з горизонтальними лініями.
- *Horizontal Running Connection* - створює горизонтальне з’єднання між двома об’єктами та є швидким способом з’єднання об’єктів, що знаходяться на відстані більше одного блоку один від одного.
- *IEC Timer* - створює таймер і замінює «Таймер».
- *Timer* - створює модуль таймера (замість нього використовуйте таймер IEC).
- *Monostable* - створює одноразовий моностабільний модуль
- «Лічильник» – створює модуль лічильника.
- *Compare* - створює блок порівняння для порівняння змінної зі значеннями або іншими змінними, наприклад, `%W1<=5` або `%W1=%W2`. Блок порівняння не можна розмістити в крайній правій частині відображення розділу.
- *Variable Assignment* - створює блок присвоєння, щоб ви могли присвоювати значення змінним, наприклад, `%W2=7` або `%W1=%W2`. Функції ПРИСВЯЗУВАННЯ можна розміщувати лише в крайньому правому куті відображення розділу.

8.2.5.6 Вікно конфігурації

Вікно конфігурації відображає поточний стан проекту та містить вкладки налаштування Modbus.

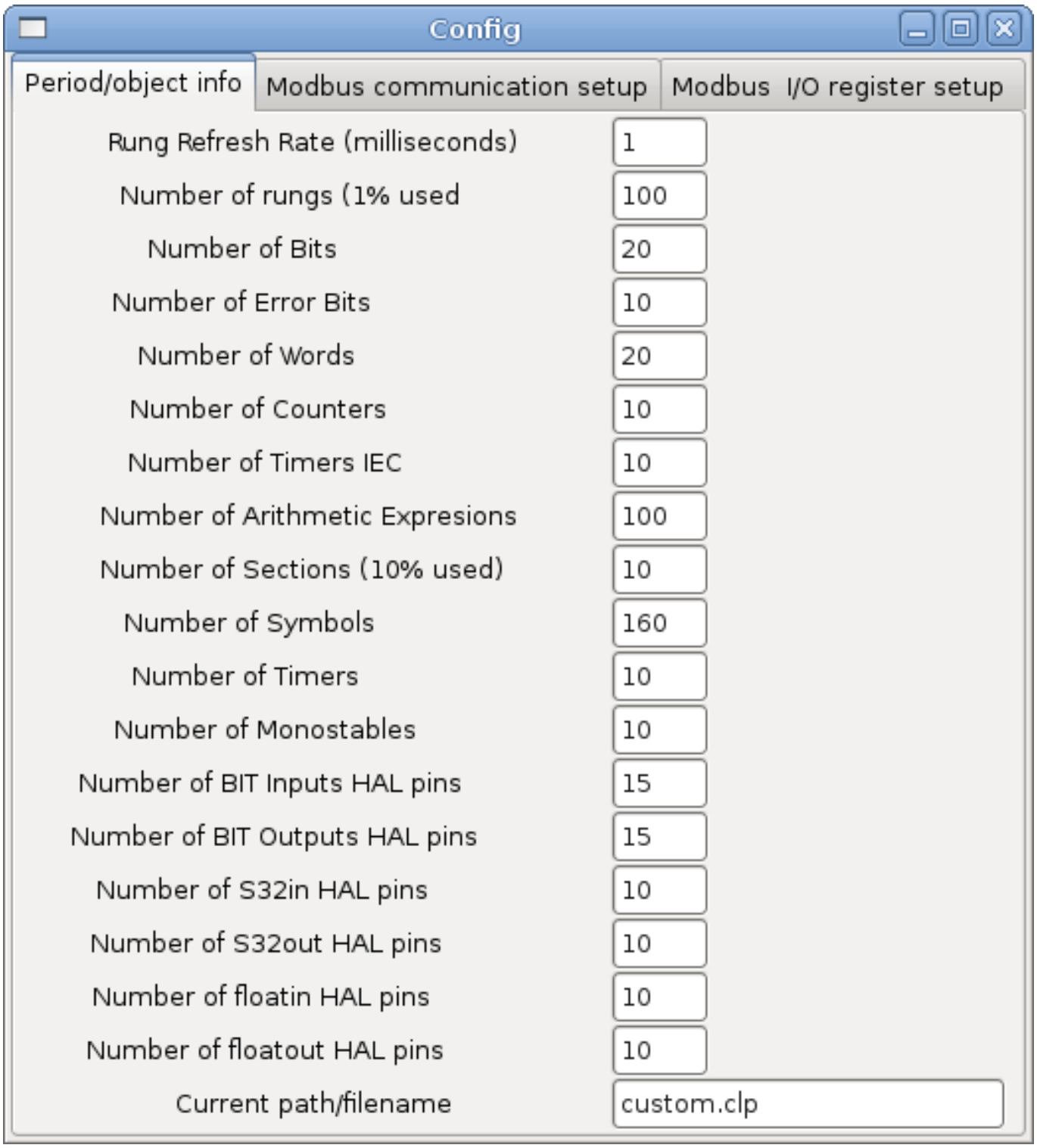


Figure 8.7: Вікно конфігурації

8.2.6 Об’єкти драбини

8.2.6.1 КОНТАКТИ

Представляють собою перемикачі або релейні контакти. Вони керуються змінною літерою та цифрою, призначеною їм.

Змінна літера може бути B, I або Q, а число може бути до трьох цифр, наприклад, `%I2`, `%Q3` або `%B123`. Змінна I контролюється вхідним контактом HAL з відповідним номером. Змінна B призначена для внутрішніх контактів, що керуються котушкою B з відповідним номером. Змінна Q керується котушкою Q з відповідним номером (як реле з декількома контактами). Наприклад, якщо контакт HAL `classicladder.0.in-00` має значення true, то контакт %I0 N.O. буде увімкнений (замкнений, true, як завгодно). Якщо котушка %B7 «під напругою» (увімкнена, істинна тощо), то контакт %B7 N.O. буде увімкнений. Якщо котушка %Q1 «під напругою», то контакт %Q1 N.O. буде увімкнений (а контакт HAL `classicladder.0.out-01` буде істинним).

- «H.O. контакт» - зображення:images/ladder_action_load.png[«Нормально розімкнутий контакт»] (Нормально розімкнутий). Коли змінна має значення «хибність», перемикач вимкнено.
- «H.З. контакт» - зображення:images/ladder_action_loadbar.png[«Нормально замкнутий контакт»] (Нормально замкнутий). Коли змінна має значення «хибність», перемикач увімкнено.
- *Rising Edge Contact* - Коли змінна змінюється з хибного на істинне, перемикач увімкнений ІМПУЛЬСНО.
- *Falling Edge Contact* - Коли змінна змінюється з істини на хибність, перемикач увімкнений ІМПУЛЬСНО.

8.2.6.2 ТАЙМЕРИ IEC

Представляють нові таймери зворотного відліку. Таймери IEC замінюють таймери та моностабільні таймери.

Таймери IEC мають 2 контакти.

- *I* - вхідний контакт
- *Q* - вихідний контакт

Є три режими - TON, TOF, TP.

- *TON* - Коли вхідний сигнал таймера має значення «true», зворотний відлік починається та триває доти, доки вхідний сигнал залишається «true». Після завершення зворотного відліку, доки вхідний сигнал таймера залишається «true», вихідний сигнал буде «true».
- *TOF* - Коли вхідний сигнал таймера має значення «true», він встановлює вихідний сигнал «true». Коли вхідний сигнал має значення «false», таймер починає зворотний відлік, а потім встановлює вихідний сигнал «false».
- *TP* - Коли вхід таймера імпульсно перетворюється на ”true” або утримується в такому положенні, таймер встановлює вихід у ”true” до закінчення зворотного відліку таймера (одноразово)

Інтервали часу можна встановлювати, кратні 100 мс, секундам або хвилинам.

Також є змінні для таймерів IEC, які можна зчитувати та/або записувати в блоках порівняння або операції.

- *%TMxxx.Q* - таймер завершено (логічне значення, читання/запис)
- *%TMxxx.P* - пресет таймера (читання/запис)
- *%TMxxx.V* - значення таймера (читання-запис)

8.2.6.3 ТАЙМЕРИ

Представляють таймери зворотного відліку. Це застаріло та замінено таймерами IEC.

Таймери мають 4 контакти.

- *E* - увімкнути (вхід) запускає таймер, коли значення true, скидає, коли значення false
- *C* - керування (вхід) має бути увімкнене для роботи таймера (зазвичай підключається до E)
- *D* - виконано (вивід) true, коли таймер вичерпає час очікування, і доки E залишається true
- *R* - running (вихід) true, коли таймер працює

Базовий час таймера може бути кратним мілісекундам, секундам або хвилинам.

Також є змінні для таймерів, які можна зчитувати та/або записувати в блоках порівняння або операції.

- *%Txx.R* - Таймер xx працює (логічне значення, лише для читання)
- *%Txx.D* - Таймер xx завершено (логічне значення, лише для читання)
- *%Txx.V* - Поточне значення таймера xx (ціле число, лише для читання)
- *%Txx.P* - Таймер xx попередньо встановлено (ціле число, читання або запис)

8.2.6.4 МОНОСТАЛИ

Представляють оригінальні одноразові таймери. Тепер це застаріло та замінено таймерами IEC.

Моностабільні реле мають 2 контакти, I та R.

- *I* - вхід (input) запустить монотаймер.
- *R* - running (вихід) буде істинним, поки працює таймер.

Контакт I чутливий до переднього фронту, тобто він запускає таймер тільки при зміні стану з false на true (або з вимкненого на увімкнений). Під час роботи таймера контакт I може змінюватися без впливу на роботу таймера. R буде true і залишатиметься true, поки таймер не дорахує до нуля. Базовий інтервал таймера може бути кратним мілісекундам, секундам або хвилинам.

Також є змінні для моностабільних об’єктів, які можна зчитувати та/або записувати в блоках порівняння або операції.

- *%Mxx.R* - Моностабільний xx виконується (логічне значення, лише для читання)
- *%Mxx.V* - Моностабільне поточне значення xx (ціле число, лише для читання)
- *%Mxx.P* - Моностабільний xx пресет (ціле число, читання або запис)

8.2.6.5 ЛІЧИЛЬНИКИ

Представляють лічильники вгору/вниз.

Є 7 контактів:

- *R* - reset (input) скине лічильник до 0.
- *P* - пресет (вхід) встановить лічильник на номер пресету, призначений у меню редагування.

- «U» – підрахунок (вхід) додасть одиницю до лічильника.
- *D* - зворотний лічильник (вхід) відніме одиницю від лічильника.
- *E* - Значення ”під потоком” (вихід) буде істинним, коли лічильник перейде від 0 до 9999.
- *D* - done (вивід) буде true, коли лічильник дорівнює попередньому значенню.
- *F* - Переповнення (вихід) буде істинним, коли лічильник перевищить значення з 9999 до 0.

Контакти для лічильника вгору та вниз чутливі до фронту, тобто вони рахують лише тоді, коли стан контакту змінюється з хибного на істинне (або з вимкненого на ввімкнене, якщо вам зручніше).

Діапазон від 0 до 9999.

Також є змінні для лічильників, які можна зчитувати та/або записувати в блоках порівняння або операції.

- ’%C’*xx*.D - Лічильник *xx* завершено (*логічне*, *лише_читання*)
- ’%C’*xx*.E - Лічильник *xx* порожній переповнення (*логічне*, *лише для читання_*)
- ’%C’*xx*.F - Лічильник *xx* повне переповнення (*логічне*, *лише_читання*)
- ’%C’*xx*.V - Поточне значення лічильника *xx* (*ціле*, *читання* або *запис*)
- ’%C’*xx*.P - Лічильник *xx* попередньо встановлено (*ціле*, *читання* або *запис*)

8.2.6.6 ПОРІВНЯТИ

Для арифметичного порівняння. Чи змінна %XXX = цьому числу (або обчисленому числу)

Блок порівняння буде істинним, коли порівняння має значення істинне. Ви можете використовувати більшість математичних символів:

- +, -, *, /, = (стандартні математичні символи)
- < (менше ніж), > (більше ніж), <= (менше або дорівнює), >= (більше або дорівнює), <> (не дорівнює)
- (,) розділяють на групи, наприклад %IF1=2,&%IF2<5 у псевдокоді перекладається як «якщо %IF1 дорівнює 2, а %IF2 менше 5, то порівняння є істинним». Зверніть увагу на кому, що розділяє дві групи порівнянь.
- ^ (показник степеня), % (модуль), & (і), | (або). -
- ABS (абсолютний), MOY (французькою – середній), AVG (середній)

Наприклад, ABS(%W2)=1, MOY(%W1,%W2)<3.

У рівнянні порівняння пробіли не допускаються. Наприклад, %C0.V>%C0.P є коректним виразом порівняння, тоді як %C0.V > %CO.P не є коректним виразом.

Внизу сторінки наведено список змінних, які можна використовувати для читання та запису в об’єкти релейно-контактної схеми. При відкритті нового блоку порівняння обов’язково видаліть символ # при введенні порівняння.

Щоб дізнатися, чи значення змінної слова №1 менше ніж у 2 рази перевищує поточне значення лічильника №0, синтаксис буде таким:

```
%W1<2*%C0.V
```

Щоб дізнатися, чи дорівнює S32in біт 2 10, синтаксис буде таким:

```
%IW2=10
```

Примітка: Порівняння використовує арифметичне дорівнювання, а не подвійне дорівнювання, до якого звикли програмісти.

8.2.6.7 ПРИЗНАЧЕННЯ ЗМІННИХ

Для присвоєння змінної, наприклад, присвоїти це число (або обчислене число) цій змінній %xxx, існують дві математичні функції MINI і MAXI, які перевіряють змінну на максимальне (0x80000000) і мінімальне значення (0x07FFFFFF) (майте на увазі значення зі знаком) і не дозволяють їм виходити за межі.

Коли відкривається новий блок присвоєння змінних, обов'язково видаліть символ # під час введення присвоєння.

Щоб призначити значення 10 попередньому налаштуванню таймера IEC Timer 0, синтаксис буде таким:

```
%TM0.P=10
```

Щоб призначити значення 12 біту 3 s32out, синтаксис буде таким:

```
%QW3=12
```

Note
Коли ви присвоюєте значення змінній за допомогою блоку присвоєння змінної, це значення зберігається доти, доки ви не присвоїте нове значення за допомогою блоку присвоєння змінної. Останнє присвоєне значення буде відновлено при запуску LinuxCNC.

На наступному малюнку показано приклад присвоєння та порівняння. %QW0 є бітом S32out, а %IW0 є бітом S32in. У цьому випадку вивід HAL `classicladder.0.s32out-00` буде встановлений на значення 5, а коли вивід HAL `classicladder.0.s32in-00` дорівнює 0, вивід HAL `classicladder.0.ou` буде встановлений на True.

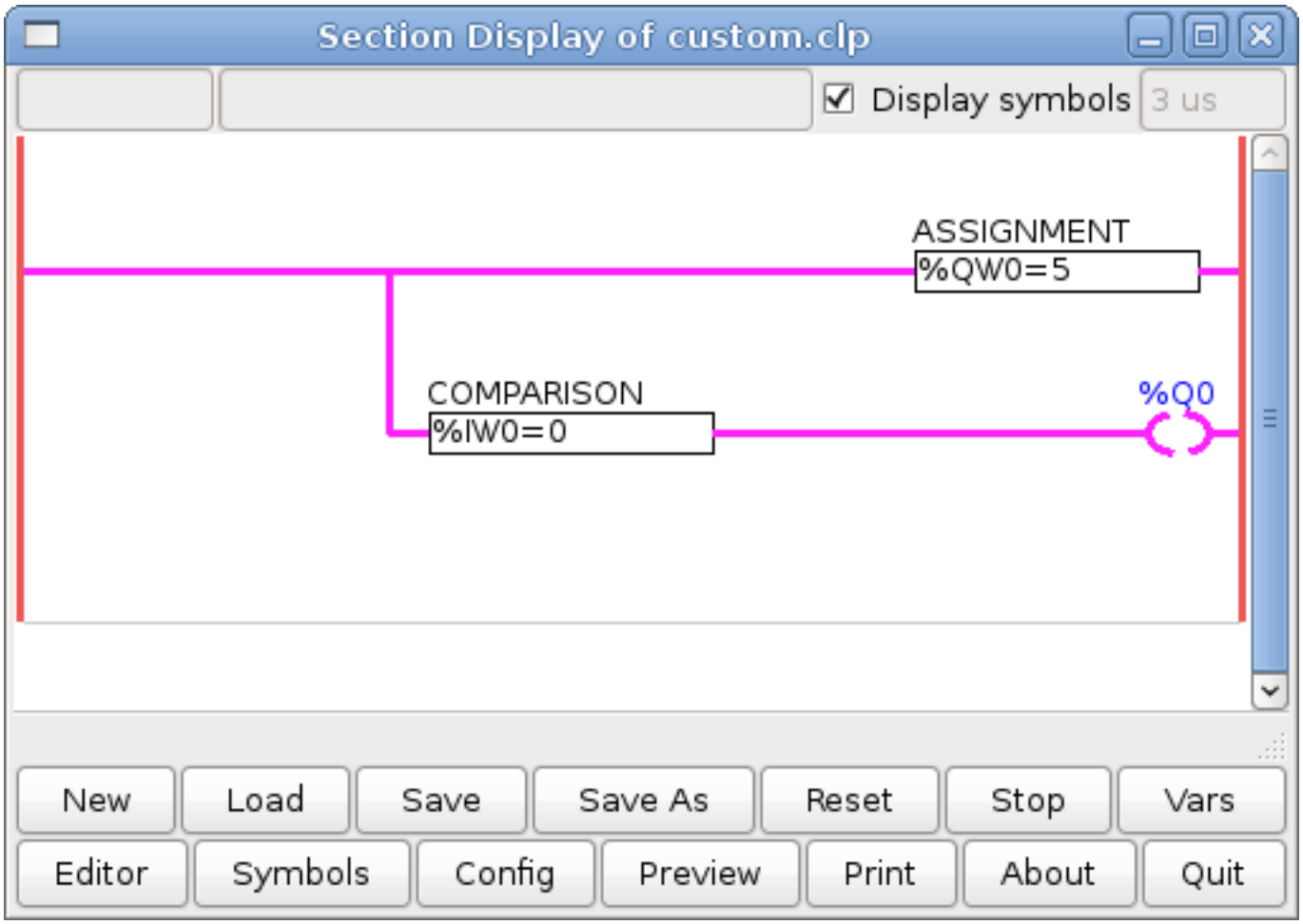


Figure 8.8: Приклад сходинок призначення/порівняння



Figure 8.9: Приклад виразу присвоєння



Figure 8.10: Приклад виразу порівняння

8.2.6.8 КОТУШКИ

Котушки представляють собою котушки реле. Вони керуються змінною літерою та цифрою, призначеною їм.

Змінною літерою може бути B або Q, а числом може бути число до трьох цифр, наприклад, %Q3 або %B123. Котушки Q керують виводами HAL, наприклад, якщо %Q15 знаходиться під напругою, то вивід HAL classicladder.0.out-15 буде істинним. Котушки B є внутрішніми котушками, що використовуються для керування потоком програми.

- *N.O. COIL* - Котушка реле: Коли котушка під напругою, її нормально розімкнутий (короткозамкнутий Н.О.) контакт буде замкнутий (увімкнений, істина тощо), і струм може проходити.
- *N.C. COIL* - Котушка реле, яка інвертує свої контакти: коли котушка під напругою, її контакт, який нормально замкнутий (короткозамкнений: Н.З.), розмикається (вимикається, хибний тощо), і потік струму переривається.
- *SET COIL* - Котушка реле з фіксуючими контактами: Коли котушка під напругою, її нормально-розмикаючий контакт буде фіксовано замкнутим.
- *RESET COIL* - Котушка реле з фіксуючими контактами: Коли котушка під напругою, її нормально-розмикаючий контакт буде розімкнутим.
- «JUMP COIL» — котушка «goto»: коли котушка знаходиться під напругою, програма решітки переходить до сходинки (у розділі CURRENT) — точки переходу позначаються міткою сходинки. (Додайте мітки сходинок у розділі дисплея, у верхньому лівому полі мітки.)
- «CALL COIL» — котушка «gosub»: коли котушка отримує живлення, програма переходить до розділу підпрограми, позначеного номером підпрограми — підпрограми позначені SR0 до SR9 (позначте їх у диспетчері розділів).



Warning
Якщо ви використовуєте нормально-розмикаючий контакт з нормально-розмикаючою котушкою, логіка працюватиме (коли котушка під напругою, контакт буде замкнутий), але це справді важко виконати!

JUMP COIL використовується для «переходу» до іншого розділу, подібно до goto в мові програмування BASIC.

Якщо ви подивитеся у верхній лівий кут вікна відображення розділів, ви побачите невелике поле для підпису та довше поле для коментарів поруч із ним. Тепер перейдіть до Редактор → Змінити, потім поверніться до маленького поля та введіть назву.

Додайте коментар у розділі коментарів. Ця назва мітки є лише назвою цієї сходинки та використовується JUMP COIL для визначення місця призначення.

Розміщуючи СТРИБКОВУ КОТУШКУ, додайте її в крайнє праве положення та змініть мітку на сходинку, на яку ви хочете СТРИБНУТИ.

CALL COIL використовується для переходу до секції підпрограми з подальшим поверненням, подібно до gosub у мові програмування BASIC.

Якщо ви перейдете до вікна менеджера розділів, натисніть кнопку «Додати розділ». Ви можете назвати цей розділ, вибрати мову, яку він використовуватиме (колонки або послідовну), та вибрати його тип (головний або підпрограма).

Виберіть номер підпрограми (наприклад, SR0). Буде відображено порожній розділ, і ви зможете створити свою підпрограму.

Коли ви це зробите, поверніться до менеджера розділів і натисніть на ваш головний розділ (назва за замовчуванням prog1).

Тепер ви можете додати котушку виклику до вашої програми. Котушки виклику слід розміщувати в крайній правій позиції сходинки.

Не забудьте змінити мітку на номер підпрограми, який ви вибрали раніше.

8.2.7 Змінні ClassicLadder

Ці змінні використовуються в інструкціях COMPARE або OPERATE для отримання інформації про об’єкти рейкової коефіцієнта або зміни їх специфікацій, таких як зміна попереднього налаштування лічильника або перевірка завершення роботи таймера.

Список змінних:

- `%B`*xxx* - Бітова пам’ять *xxx* (логічне значення)
- `%W`*xxx* - Пам’ять слів *xxx* (32 біти зі знаком цілого числа)
- `%IW`*xxx* - Пам’ять слів *xxx* (S32 на виводі)
- `%QW`*xxx* - Пам’ять слів *xxx* (вихідний контакт S32)
- `%IF`*xx* - Пам’ять слів *xx* (число з плаваючою комою в pin) (**конвертовано в S32 в ClassicLadder**)
- `%QF`*xx* - Пам’ять слів *xx* (вихідний висновок) (**конвертовано в S32 у ClassicLadder**)
- `%T __xx__.R` - Таймер *xx* працює (логічне значення, лише для читання користувачем)
- `%T __xx__.D` - Таймер *xx* завершено (логічне значення, лише для читання користувачем)
- `%T __xx__.V` - Поточне значення таймера *xx* (ціле число, лише для читання користувачем)
- `%T __xx__.P` – Попереднє налаштування таймера *xx* (ціле число)
- `%TM __xxx__.Q` - Таймер *xxx* завершено (логічне значення, читання/запис)
- `%TM __xxx__.P` - Таймер *xxx* пресет (ціле число, читання та запис)
- `%TM __xxx__.V` - Значення таймера *xxx* (ціле число, читання/запис)
- `%M __xx__.R` - Моностабільний *xx* виконується (логічне значення)

- %M `__xx__.V` - Моностабільне *xx* поточне значення (ціле число, лише для читання користувачем)
- %M `__xx__.P` - Моностабільний пресет *xx* (ціле число)
- %C `__xx__.D` - Лічильник *xx* завершено (логічне значення, лише для читання користувачем)
- %C `__xx__.E` - Лічильник *xx* порожній переповнення (логічне значення, лише для читання користувачем)
- %C `__xx__.F` - Лічильник *xx* повне переповнення (логічне значення, лише для читання користувачем
- %C `__xx__.V` - Поточне значення лічильника *xx* (ціле число)
- %C `__xx__.P` - Попередньо встановлений лічильник *xx* (ціле число)
- %I*xxx* - Фізичний вхід *xxx* (логічне значення) (вхідний біт HAL)
- %Q*xxx* - Фізичний вихід *xxx* (логічне значення) (біт виходу HAL)
- %X*xxx* - Діяльність кроку *xxx* (послідовна мова)
- %X `__xxx__.V` – Час активності в секундах кроку *xxx* (послідовна мова)
- %E*xx* - Помилки (логічні, читання та запис (будуть перезаписані))
- «Індексовані або векторизовані змінні» – це змінні, індексовані іншою змінною. Дехто може назвати це векторизованими змінними. Приклад: %W0[%W4] => якщо %W4 дорівнює 23, це відповідає %W23

8.2.8 Програмування GRAFCET (кінцевий автомат)



Warning
Це, мабуть, найменш використовувана і найменш зрозуміла функція ClassicLadder. Послідовне програмування використовується для того, щоб серія подій ладдеру завжди відбувалася у встановленому порядку. Послідовні програми не працюють самостійно. Завжди є також програма ладдеру, яка контролює змінні. Ось основні правила, що регулюють послідовні програми:

- Правило 1: Початкова ситуація - Початкова ситуація характеризується початковими кроками, які за визначенням перебувають в активному стані на початку операції. Повинно бути принаймні один початковий крок.
- Правило 2: R2, Очищення переходу - Перехід може бути увімкненим або вимкненим. Він вважається увімкненим, коли всі безпосередньо попередні кроки, пов’язані з відповідним символом переходу, є активними, в іншому випадку він вимкнений. Перехід не може бути очищений, якщо він не увімкнений і пов’язана з ним умова переходу не є істинною.
- Правило 3: R3, Еволюція активних кроків - Очищення переходу одночасно призводить до активного стану наступного кроку (кроків) і до неактивного стану попереднього кроку (кроків).
- Правило 4: R4, Одночасне очищення переходів - Усі одночасно очищені переходи очищуються одночасно.
- Правило 5: R5, Одночасна активація та деактивація кроку - Якщо під час роботи крок одночасно активується та деактивується, пріоритет надається активації.

Це вікно редактора SEQUENTIAL. (Починаючи з верхнього лівого кута):
Стрілка селектора, Ластик
Звичайний крок, Початковий (стартовий) крок
Перехід, Крок і перехід
Перехід-нижній зв’язок, Перехід-верхній зв’язок
Прохідний зв’язок-нижній, Прохідний зв’язок-верхній Стрибок
Зв’язок, Поле для коментарів

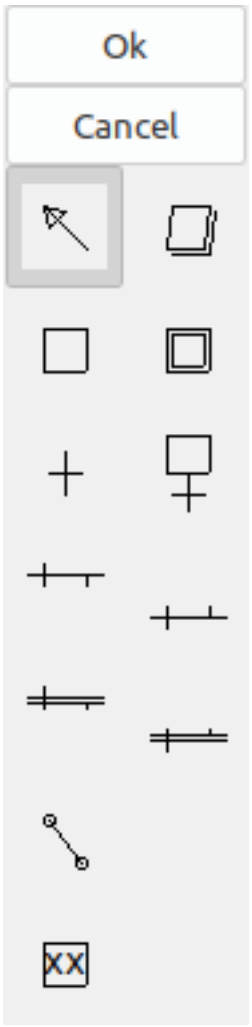


Figure 8.11: Вікно редактора послідовностей

- *ORDINARY STEP* - має унікальний номер для кожного
- *STARTING STEP* - Послідовна програма повинна мати його. Саме тут програма почне свою роботу.
- *TRANSITION* - показує змінну, яка має бути істинною, щоб керування перейшло до наступного кроку.
- «КРОК І ПЕРЕХІД» – поєднані для зручності
- *TRANSITION LINK-DOWNSIDE* - розділяє логічний потік на одну з двох можливих ліній залежно від того, який із наступних кроків є істинним першим (логіка «Подумай АБО»)
- *TRANSITION LINK=UPSIDE* - об’єднує дві логічні лінії (АБО) назад в одну

- *PASS-THROUGH LINK-DOWNSIDE* - розділяє логічний потік на два рядки, ОБИДВА з яких мають бути істинними для продовження (думка I логіка)
- *PASS-THROUGH LINK-UPSIDE* - об’єднує дві паралельні логічні лінії (логіка I) знову разом
- *JUMP LINK* - з’єднує кроки, які не розташовані один під одним, наприклад, з’єднує останній крок з першим
- *COMMENT BOX* - використовується для додавання коментарів

Щоб використовувати посилання, у вас вже мають бути розміщені кроки. Виберіть тип посилання, а потім по черзі вибирайте два кроки або транзакції. Це вимагає практики!

При послідовному програмуванні: змінна `%X`__xxx__ (наприклад, `%X5`) використовується для перевірки, чи активний крок. Змінна `%X`__xxx__.V (наприклад, `%X5.V`) використовується для перевірки, як довго крок був активним. Змінні %X і %X.v використовуються в логіці LADDER. Змінні, призначені переходам (наприклад, %B), контролюють, чи логіка перейде до наступного кроку. Після того, як крок став активним, змінна переходу, яка спричинила його активацію, більше не контролює його. Останній крок повинен JUMP LINK повернутися тільки до початкового кроку.

8.2.9 Modbus

Що слід врахувати:

- Modbus — це програма, яка не працює в реальному часі, тому на сильно завантаженому комп’ютері можуть виникати проблеми із затримкою.
- Modbus насправді не підходить для подій у реальному часі, таких як керування положенням двигунів або керування аварійною зупинкою.
- Для роботи Modbus має бути запущений графічний інтерфейс ClassicLadder.
- Modbus ще не повністю завершений, тому він не виконує всі функції Modbus.

Щоб ініціалізувати MODBUS, потрібно вказати це під час завантаження програми ClassicLadder, яка не працює в реальному часі.

Завантаження Modbus

```
loadusr -w classicladder --modmaster myprogram.clp
```

Параметр `-w` змушує HAL чекати, поки ви закриєте ClassicLadder, перш ніж закривати сеанс реального часу. ClassicLadder також завантажує підлеглий сервер TCP Modbus, якщо ви додасте `--modserver` у командному рядку.

Функції Modbus

- *1* - котушки для зчитування
- *2* - читання вхідних даних
- *3* - читання регістрів зберігання
- *4* - читання вхідних регістрів
- *5* - написати окремі котушки
- *6* - запис в один регістр
- *8* - ехо-тест

- *15* - написати кілька котушок
- *16* - запис кількох регістрів

Якщо ви не вкажете *--modmaster* під час завантаження програми ClassicLadder, яка не працює в реальному часі, ця сторінка не відображатиметься.

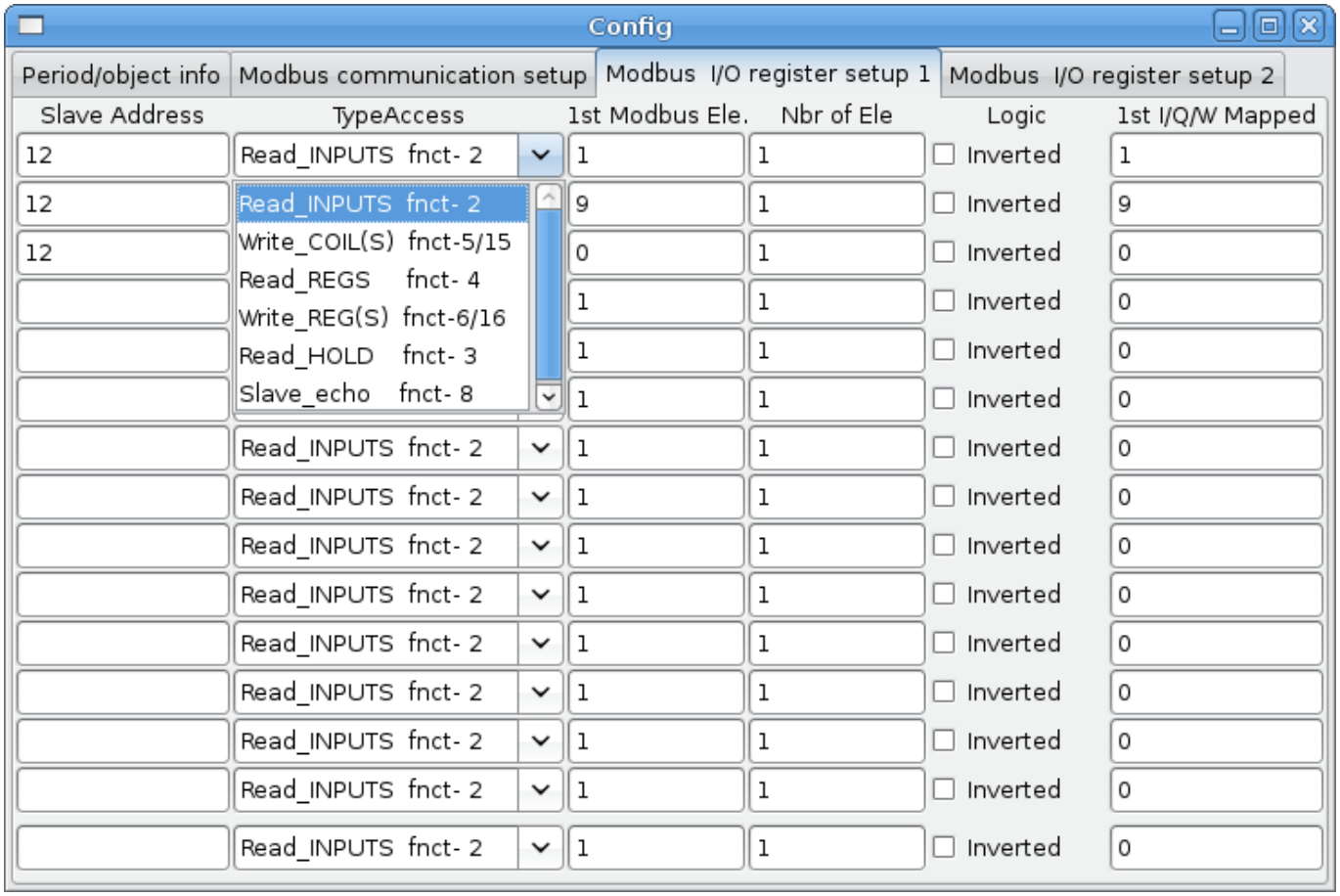


Figure 8.12: Конфігурація вводу/виводу Modbus

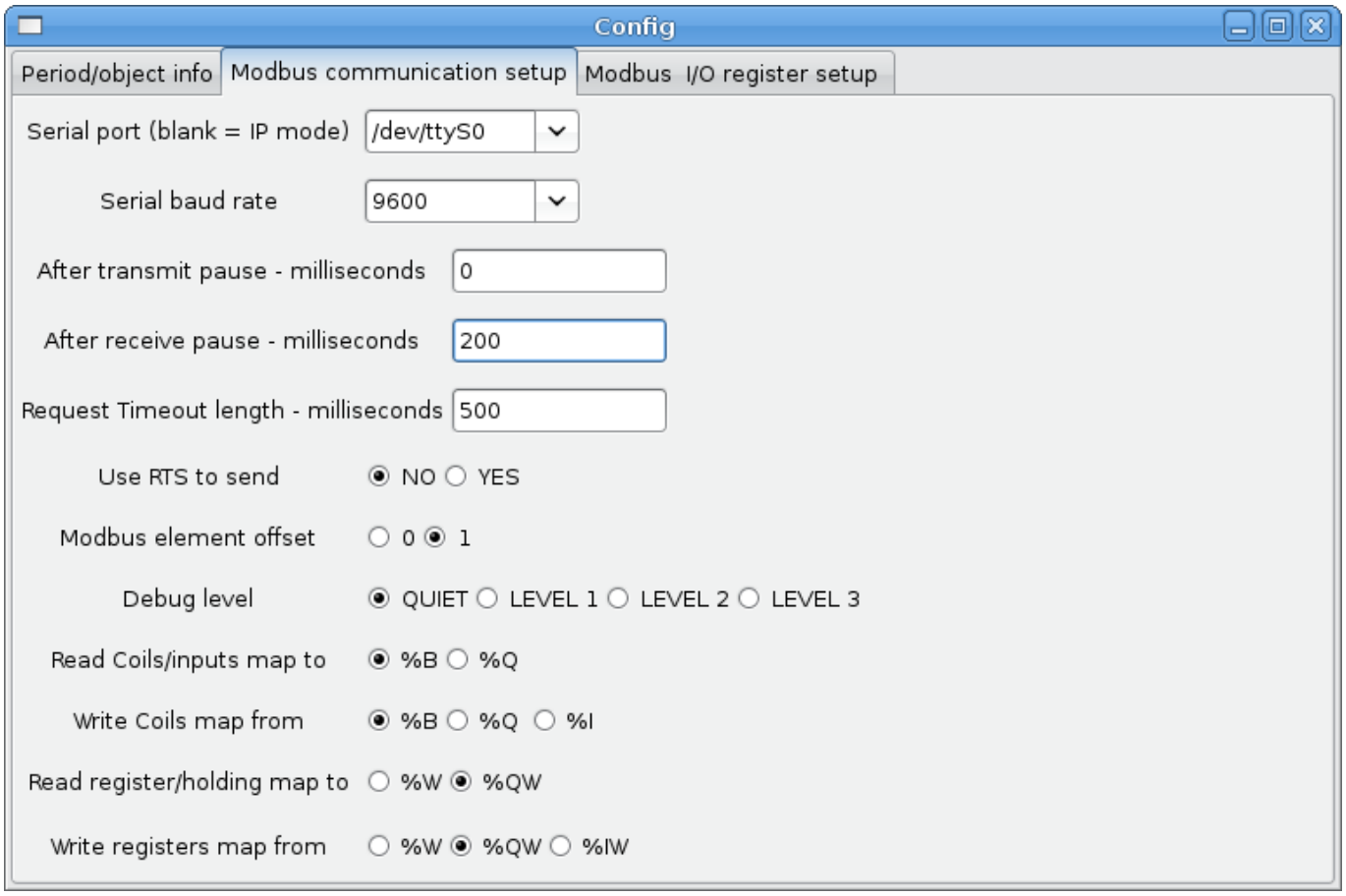


Figure 8.13: Конфігурація зв’язку Modbus

- «SERIAL PORT» – для IP-адреси порожнє поле. Для послідовного порту – розташування/назва драйвера послідовного порту, наприклад, /dev/ttyS0 (або /dev/ttyUSB0 для перетворювача USB-послідовний порт).
- *SERIAL SPEED* - Слід встановити швидкість, на яку встановлено ведений пристрій - підтримуються 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- *PAUSE AFTER TRANSMIT* - Пауза (мілісекунди) після передачі та перед отриманням відповіді, деяким пристроям потрібен довше часу (наприклад, перетворювачі USB-послідовний порт).
- *PAUSE INTER-FRAME* - Пауза (мілісекунди) після отримання відповіді від підлеглого пристрою. Це встановлює робочий цикл запитів (це пауза для КОЖНОГО запиту).
- *REQUEST TIMEOUT LENGTH* - Тривалість (мілісекунди), перш ніж ми вирішимо, що підлеглий пристрій не відповів.
- *MODBUS ELEMENT OFFSET* - використовується для зміщення номерів елементів на 1 (для розбіжностей у нумерації виробників).
- *DEBUG LEVEL* - Встановіть це значення на 0-3 (0, щоб припинити виведення інформації про налагодження, окрім помилок відсутності відповіді).
- *READ COILS/INPUTS MAP TO* - Виберіть, які змінні, що зчитують котушки/входи, будуть оновлюватис (B або Q).
- *WRITE COILS MAP TO* - Виберіть, з яких змінних, що записують котушки, будуть оновлюватися (B, Q або I).

- *READ REGISTERS/HOLDING* - Виберіть, які змінні, що зчитують регістри, будуть оновлюватися (W або QW).
- *WRITE REGISTERS MAP TO* - Виберіть, з яких змінних, що зчитують регістри, будуть оновлюватися (W, QW або IW).
- *SLAVE ADDRESS* - Для послідовного порту ідентифікаційний номер веденого пристрою зазвичай встановлюється на веденому пристрої (зазвичай 1-256). Для IP-адреси веденого пристрою та, за бажанням, номер порту.
- *TYPE ACCESS* - Це вибирає код функції MODBUS для надсилання до веденого пристрою (наприклад, який тип запиту).
- *COILS / INPUTS* - Входи та котушки (біти) зчитуються/записуються в змінні I, B або Q (вибір користувача).
- *REGISTERS (WORDS)* - Регістри (слова/числа) відповідають змінним IW, W або QW (вибір користувача
- *1st MODBUS ELEMENT* - Адреса (або номер регістра) першого елемента в групі (не забудьте правильно встановити ЗМІЩЕННЯ ЕЛЕМЕНТА MODBUS).
- *NUMBER OF ELEMENTS* - Кількість елементів у цій групі.
- *LOGIC* - Тут можна перевернути логіку.
- *1st%I%Q IQ WQ MAPPED* - Це початкова кількість змінних %B, %I, %Q, %W, %IW або %QW, які відображаються на/з групи елементів modbus (починаючи з першого номера елемента modbus).

У наведеному вище прикладі: номер порту - для мого комп’ютера /dev/ttyS0 був моїм послідовним портом.

Швидкість послідовного порту встановлена на рівні 9600 бод.

Адреса веденого пристрою встановлена на 12 (на моєму частотному перетворювачі я можу встановити її від 1 до 31, тобто я можу взаємодіяти максимум з 31 частотним перетворювачем в одній системі).

Перший рядок налаштовано на 8 вхідних бітів, починаючи з першого номера регістра (регістр 1). Таким чином, номери регістрів 1-8 відображаються на змінні %B ClassicLadder, починаючи з %B1 і закінчуючи %B8.

Другий рядок встановлений для 2 вихідних бітів, починаючи з дев’ятого номера регістра (регістр 9), тому номери регістрів 9-10 відображаються на змінні %Q ClassicLadder, починаючи з %Q9 і закінчуючи %Q10.

Третій рядок призначений для запису 2 регістрів (по 16 біт кожен), починаючи з 0-го регістра (регістр 0), тому номери регістрів 0-1 відображаються на змінні %W ClassicLadder, починаючи з %W0 і закінчуючи %W1.

Легко зробити помилку «off-by-one», оскільки іноді елементи Modbus посилаються, починаючи з одиниці, а не з 0 (насправді, згідно зі стандартом, так і повинно бути!). Для цього можна скористатися перемикачем «Modbus element offset» (Зсув елемента Modbus).

У документації до вашого ведомого пристрою Modbus буде вказано, як налаштовано регістри – стандартного способу немає.

Рівень ПОСЛІДОВНИЙ ПОРТ, ШВИДКІСТЬ ПОРТІВ, ПАУЗА та НАЛАГОДЖЕННЯ можна редагувати (значення застосовуються після закриття вікна конфігурації, хоча радіокнопки застосовуються негайно).

Щоб скористатися функцією echo, виберіть функцію echo та додайте номер підлеглого пристрою, який ви хочете перевірити. Вам не потрібно вказувати жодних змінних.

Число 257 буде надіслано на вказаний вами номер підлеглого пристрою, і підлеглий пристрій має надіслати його назад. Щоб побачити повідомлення, вам потрібно запустити ClassicLadder у терміналі.

8.2.10 Налаштування MODBUS

Серійний:

- ClassicLadder використовує протокол RTU (не ASCII).
- 8 бітів даних, парність не використовується, а 1 стоп-біт також відомий як 8-N-1.
- Швидкість передачі даних має бути однаковою для веденого та головного пристроїв. ClassicLadder може мати лише одну швидкість передачі даних, тому всі ведені пристрої повинні бути налаштовані на однакову швидкість.
- Міжкадрова пауза – це час для паузи після отримання відповіді.
- MODBUS_TIME_AFTER_TRANSMIT — це тривалість паузи після надсилання запиту та до отримання відповіді (це, очевидно, допомагає з USB-конвертерами, які є повільними).

8.2.10.1 Інформація про MODBUS

- ClassicLadder може використовувати розподілені входи/виходи на модулях, що використовують протокол Modbus (”головний”: опитування підлеглих пристроїв).
- Підлеглі пристрої та їхні входи/виходи можна налаштувати у вікні конфігурації.
- Доступні 2 ексклюзивні режими: Ethernet з використанням Modbus/TCP та послідовний з використанням Modbus/RTU.
- Парність не використовується.
- Якщо ім’я порту для послідовного порту не вказано, буде використано режим TCP/IP...
- Адреса веденого пристрою – це адреса веденого пристрою (Modbus/RTU) або IP-адреса.
- IP-адресу можна вказати за номером порту (xx.xx.xx.xx:pppp), інакше за замовчуванням використову порт 9502.
- Для тестів було використано 2 продукти: Modbus/TCP (Adam-6051, https://www.advantech.com) та послідовний Modbus/RTU (https://www.ipac.ws).
- Див. приклади: adam-6051 and modbus_rtu_serial.
- Веб-посилання: https://www.modbus.org та це цікаве: https://www.iatips.com/modbus.html
- СЕРВЕР MODBUS TCP ВКЛЮЧЕНО
- ClassicLadder має інтегрований сервер Modbus/TCP. Порт за замовчуванням – 9502 (попередній стандарт 502 вимагав, щоб застосунок запускався з правами root).
- Список підтримуваних кодів функцій Modbus: 1, 2, 3, 4, 5, 6, 15 та 16.
- Таблиця відповідності бітів та слів Modbus насправді не є параметричною та відповідає безпосереднь змінним %B та %W.

Більше інформації про протокол Modbus можна знайти в інтернеті.

https://www.modbus.org/

8.2.10.2 Помилки зв’язку

Якщо трапиться помилка зв’язку, з’явиться вікно з попередженням (якщо працює графічний інтерфейс) і %E0 буде істинним. Modbus продовжить спроби зв’язку. %E0 можна використовувати для прийняття рішення на основі помилки. Таймер можна використовувати для зупинки машини в разі закінчення часу очікування тощо.

8.2.11 Налагодження проблем Modbus

Хороший довідник по протоколу: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. Якщо ви запускаєте linuxcnc/classicladder з терміналу, він буде виводити команди Modbus і відповіді ведених пристроїв.

Тут ми налаштовуємо ClassicLadder на запит до веденого пристрою 1, щоб прочитати регістри утримання (код функції 3), починаючи з адреси 8448 (0x2100). Ми запитуємо повернення 1 елемента даних (шириною 2 байти). Ми відображаємо його на змінну ClassicLadder, починаючи з 2.

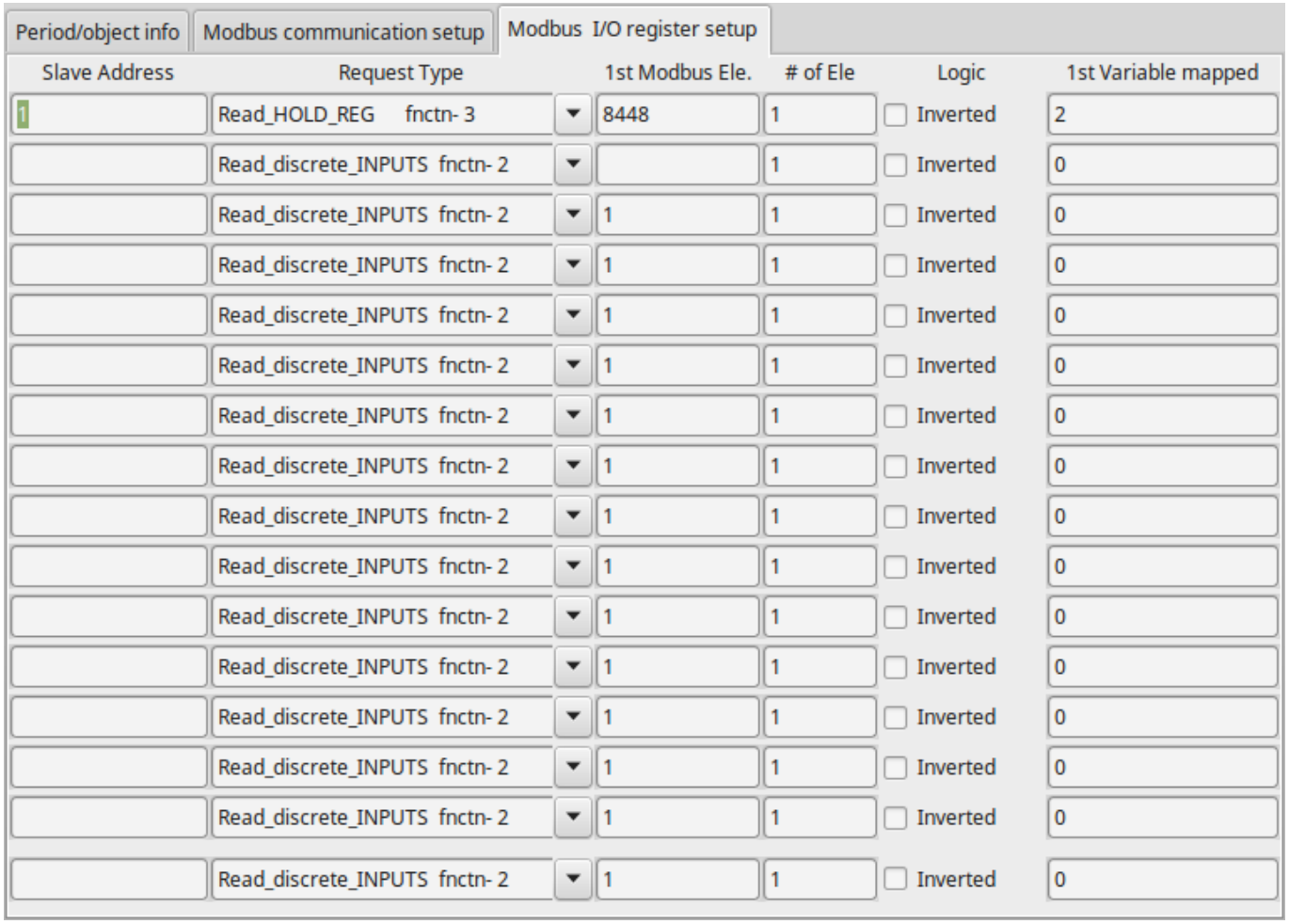


Figure 8.14: Налаштування регістрів вводу/виводу Modbus

Зверніть увагу, що на цьому зображенні ми встановили рівень налагодження на 1, тому повідомлення Modbus виводяться на термінал. Ми зіставили наші регістри читання та запису зі змінними %W ClassicLadder, тому наші повернуті дані будуть у %W2, як на іншому зображенні, де ми зіставили дані, починаючи з 2-го елемента.

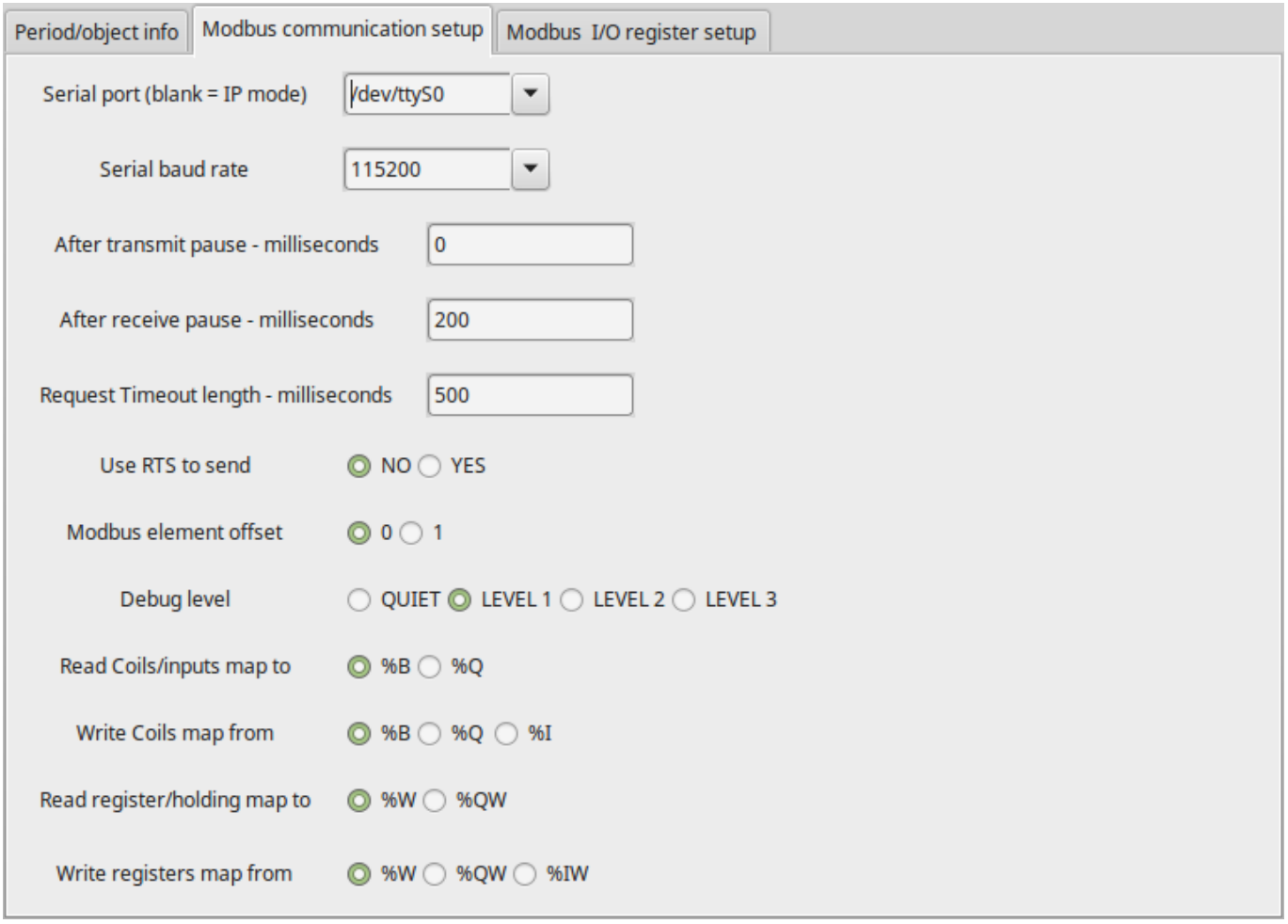


Figure 8.15: Налаштування зв’язку Modbus

8.2.11.1 Запит

Розглянемо приклад читання одного регістра зберігання з номером 8448 Decimal (0x2100 Hex). Дивлячись у довіднику протоколу Modbus:

Table 8.2: Запит на читання регістра зберігання

Ім’я	**кількість байтів**	**Значення (hex)**
Код функції	(1 байт)	3 (0x03)
Початкова адреса	(2 байти)	0 - 65535 (0x0000 до 0xFFFF)
Кількість регістрів	(2 байти)	1 до 125 (0x7D)
Контрольна сума	(2 байти)	Розраховується автоматично

Ось приклад надісланої команди, як її надруковано в терміналі (усі шістнадцяткові):

```
INFO CLASSICLADDER   b''Mb''b''ob''b''дb''b''yb''b''лb''b''ьb'' b''вb''b''вb''b''ob''b' ↩
    'дb''b''yb''/b''вb''b''иb''b''вb''b''ob''b''дb''b''yb'' Modbus b''дb''b''лb''b''яb'' b' ↩
    'нb''b''аb''b''дb''b''cb''b''иb''b''лb''b''аb''b''нb''b''нb''b''яb'':   Lgt=8 <- b''Ab'' ↩
    b''дb''b''pb''b''eb''b''cb''b''аb'' b''вb''b''eb''b''дb''b''eb''b''нb''b''ob''b''гb''b' ↩
    'ob'' b''пb''b''pb''b''иb''b''cb''b''тb''b''pb''b''ob''b''юb''-1.  b''Kb''b''ob''b''дb'' ↩
     b''фb''b''yb''b''нb''b''кb''b''цb''b''іb''b''їb''-3 b''Дb''b''аb''b''нb''b''іb''-21 0 0 ↩
     1 8E 36
```

Значення (Hex):

- Lgt = 8 = повідомлення має довжину 8 байтів, включаючи номер підлеглого пристрою та номер контрольної суми
- Номер веденого пристрою = 1 (0x1) = Адреса веденого пристрою 1
- Код функції = 3 (0x3) = читання регістра зберігання
- Початок з адреси = старший байт 33 (0x21), молодший байт 0 (0x00) = комбінована адреса = 8448 (0x2100)
- Кількість регістрів = 1 (0x1) = повернути 1 2-байтовий регістр (регістри зберігання та читання завжди мають ширину 2 байти)
- Контрольна сума = старший байт 0x8E, молодший байт 0x36 = (0x8E36)

8.2.11.2 Відповідь на помилку

Якщо є відповідь про помилку, вона надсилає код функції плюс 0x80, код помилки та контрольну суму. Отримання відповіді про помилку означає, що ведений пристрій бачить команду запиту, але не може надати дійсні дані. Дивіться довідник протоколу Modbus:

Table 8.3: Повернуто помилку для коду функції 3 (зчитування регістра зберігання)

Ім’я	**Кількість байтів**	**Значення (hex)**
Код помилки	1 байт	131 (0x83)
Код винятку	1 байт	1-4 (0x01 до 0x04)
Контрольна сума	(2 байти)	Розраховується автоматично

Значення коду винятку:

- 1 - недопустима функція
- 2 - незаконна адреса даних
- 3 - недопустиме значення даних
- 4 - збій веденого пристрою

Ось приклад отриманої команди, надрукованої в терміналі (усі шістнадцяткові):

```
INFO CLASSICLADDER-   Modbus I/O module received: Lgt=5 ->   (Slave address-1  Function  ↩
    code-83 ) 2 C0 F1
```

Значення (шістнадцяткове):

- Номер веденого пристрою = 1 (0x1) = Адреса веденого пристрою 1
- Код функції = 131 (0x83) = помилка під час читання регістра зберігання
- Код помилки = 2 (0x2) = запитується недійсна адреса даних
- Контрольна сума = (0x8E36)

8.2.11.3 Відповідь на дані

Шукаємо відповідь у довіднику протоколу Modbus:

Table 8.4: Відповідь на дані для коду функції 3 (зчитування регістра зберігання)

Ім’я	**кількість байтів**	**Значення (hex)**
Код функції	1 байт	3 (0x03)
Кількість байтів	1 байт	2 x N*
Значення реєстру	N* x 2 байти	повернене значення запитуваної адреси
Контрольна сума	(2 байти)	розраховується автоматично

*N = Кількість регістрів

Ось приклад отриманої команди, надрукованої в терміналі (усі шістнадцяткові):

```
INFO CLASSICLADDER-   b’’Ob’’b’’тb’’b’’pb’’b’’иb’’b’’мb’’b’’ab’’b’’нb’’b’’ob’’ b’’мb’’b’ ↩
    ’ob’’b’’дb’’b’’yb’’b’’лb’’b’’ьb’’ b’’вb’’b’’вb’’b’’ob’’b’’дb’’b’’yb’’/b’’вb’’b’’иb’’b’ ↩
    ’вb’’b’’ob’’b’’дb’’b’’yb’’ Modbus: Lgt=7 -> (b’’Ab’’b’’дb’’b’’pb’’b’’eb’’b’’cb’’b’’ab’’  ↩
    b’’вb’’b’’eb’’b’’дb’’b’’eb’’b’’нb’’b’’ob’’b’’гb’’b’’ob’’ b’’пb’’b’’pb’’b’’иb’’b’’cb’’b’ ↩
    ’тb’’b’’pb’’b’’ob’’b’’юb’’-1 b’’Kb’’b’’ob’’b’’дb’’ b’’фb’’b’’yb’’b’’нb’’b’’кb’’b’’цb’’b’ ↩
    ’ib’’b’’їb’’-3 2 0 0 B8 44)
```

значення (Hex):

- Номер веденого пристрою = 1 (0x1) = Адреса веденого пристрою 1
- Запитаний код функції = 3 (0x3) = запитується читання регістра зберігання
- кількість байтових регістрів = 2 (0x1) = повернути 2 байти (кожне значення регістра має ширину 2 байти)
- значення старшого байта = 0 (0x0) = значення старшого байта адреси 8448 (0x2100)

- значення молодшого байта = 0 (0x0) = значення старшого байта адреси 8448 (0x2100)
- Контрольна сума = (0xB844)

(високі та низькі байти об’єднуються для створення 16-бітного значення, а потім передаються до змінної ClassicLadder.) Реєстри читання можуть бути зіставлені з %W або %QW (внутрішня пам’ять або виходи HAL). Реєстри запису можуть бути відображені з %W, %QW або %IW (внутрішня пам’ять, виходи HAL або входи HAL). Номер змінної починається з номера, введеного в стовпці сторінки налаштування реєстру вводу-виводу Modbus: «Перша відображена змінна». Якщо в одному читанні/записі запитується кілька регістрів, то номери змінних йдуть послідовно після першого.

8.2.11.4 Помилки MODBUS

- У блоках порівняння функція %W=ABS(%W1-%W2) приймається, але обчислюється неправильно. Наразі допустимою є лише %W0=ABS(%W1).
- Під час завантаження програми релейної логіки вона завантажить інформацію про Modbus, але не накаже ClassicLadder ініціалізувати Modbus. Ви повинні ініціалізувати Modbus під час першого завантаження графічного інтерфейсу, додавши *--modmaster*.
- Якщо менеджер розділів розмістити поверх відображення розділу, через смугу прокручування та натиснути кнопку «Вихід», програма, яка не працює в реальному часі, аварійно завершить роботу.
- При використанні *--modmaster* необхідно одночасно завантажувати програму рейдерів, інакше працюватиме лише TCP.
- Читання/запис кількох регістрів у Modbus має помилки контрольної суми.

8.2.12 Налаштування ClassicLadder

У цьому розділі ми розглянемо кроки, необхідні для додавання ClassicLadder до конфігурації, згенерованої майстром StepConf. На сторінці розширених параметрів конфігурації майстра StepConf поставте позначку навпроти ”Включити” ClassicLadder PLC”.

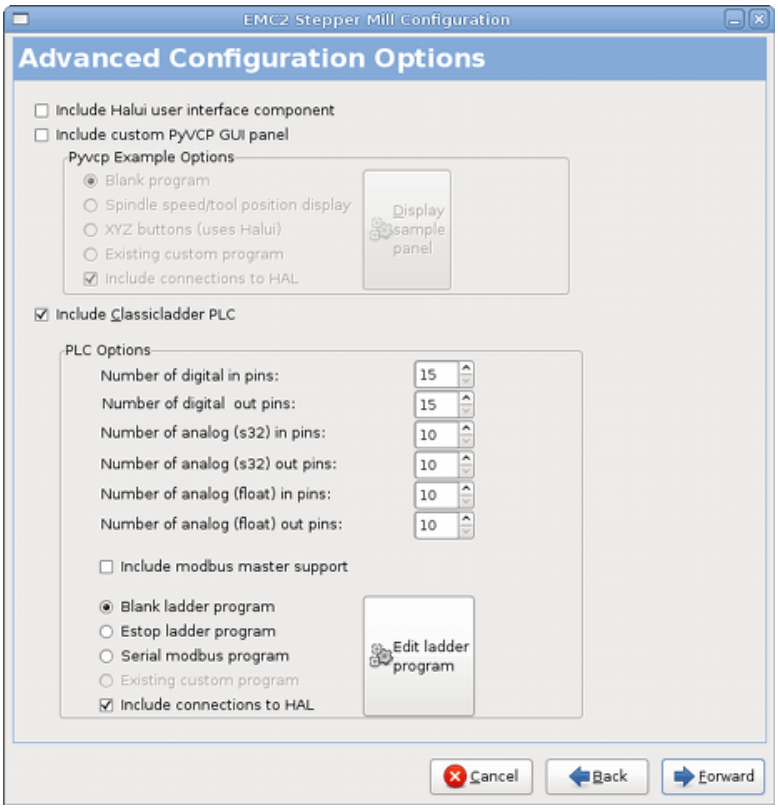


Figure 8.16: StepConf ClassicLadder

8.2.12.1 Додайте модулі

Якщо ви використовували майстер StepConf для додавання ClassicLadder, ви можете пропустити цей крок.

Щоб вручну додати ClassicLadder, спочатку потрібно додати модулі. Це робиться шляхом додавання кількох рядків до файлу custom.hal.

Цей рядок завантажує модуль реального часу:

```
loadrt classicladder_rt
```

Цей рядок додає функцію ClassicLadder до потоку серво:

```
addf classicladder.0.refresh servo-thread
```

8.2.12.2 Додавання логіки сходів

Запустіть конфігурацію та виберіть «Файл/Редактор драбини», щоб відкрити графічний інтерфейс ClassicLadder. Ви побачите порожнє вікно «Відображення розділів» та «Менеджер розділів», як показано вище. У вікні Section Display відкрийте редактор. У вікні редактора виберіть Modify. Тепер з’явиться вікно Properties, а Section Display покаже сітку. Сітка є одним щаблем драбини. Щіль може містити гілки. Простий щабель має один вхід, лінію з’єднання та один вихід. Щіль може мати до шести горизонтальних гілок. Хоча в одному прогоні може бути більше одного контуру, результати непередбачувані.

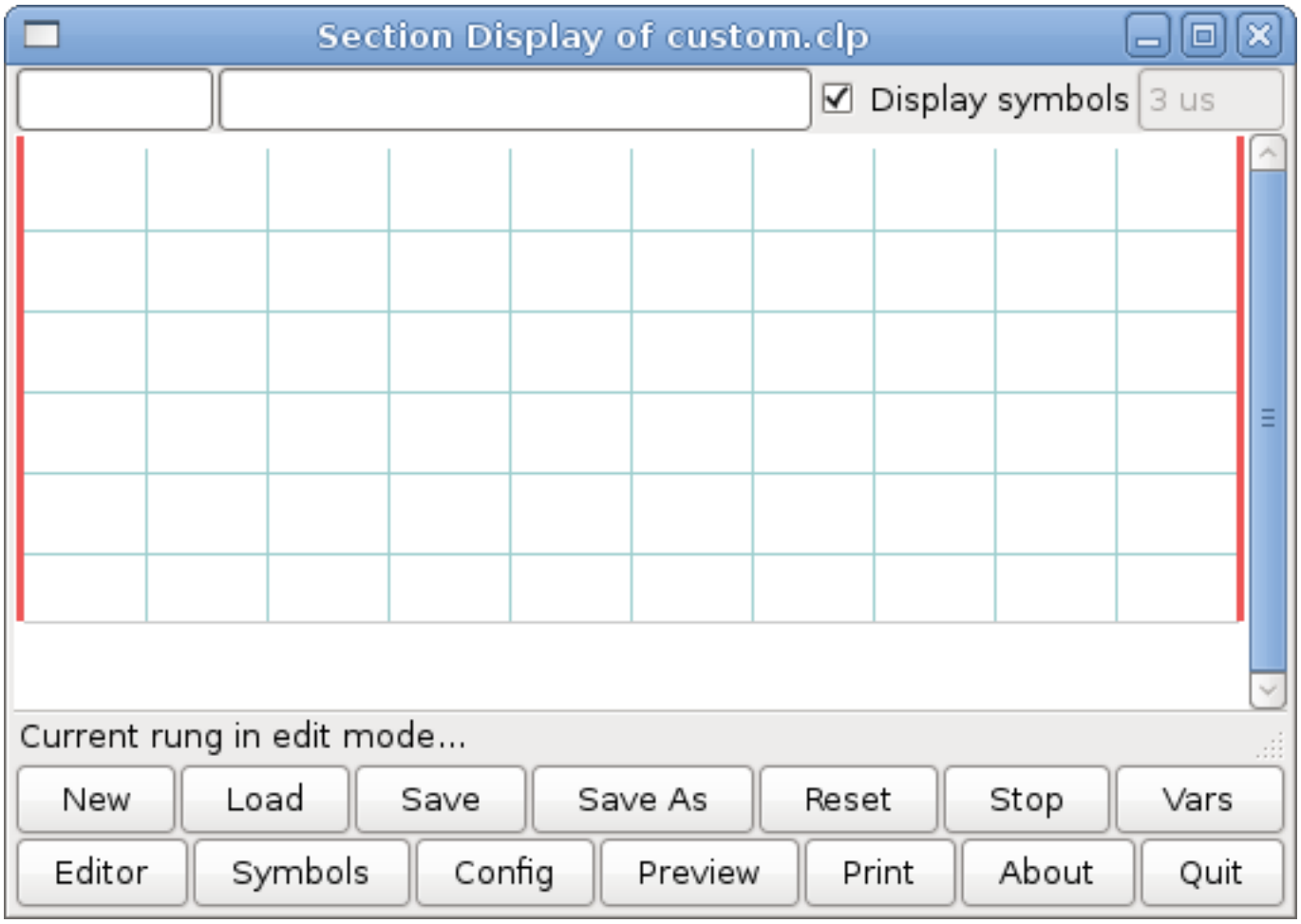


Figure 8.17: Відображення розділу з сіткою

Тепер натисніть на поле N.O. у вікні редактора.

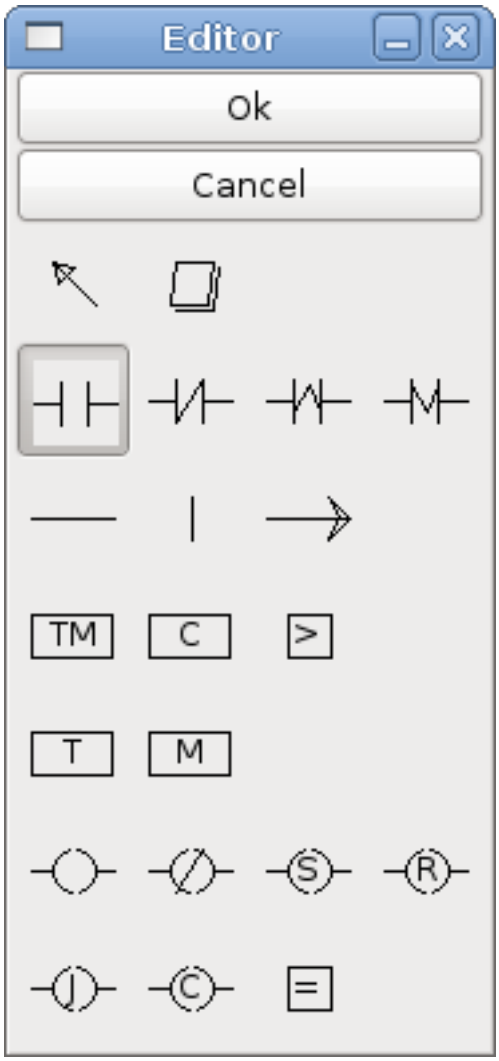


Figure 8.18: Вікно редактора

Тепер клацніть у верхній лівій сітці, щоб розмістити вхідний сигнал N.O. у сходинці.

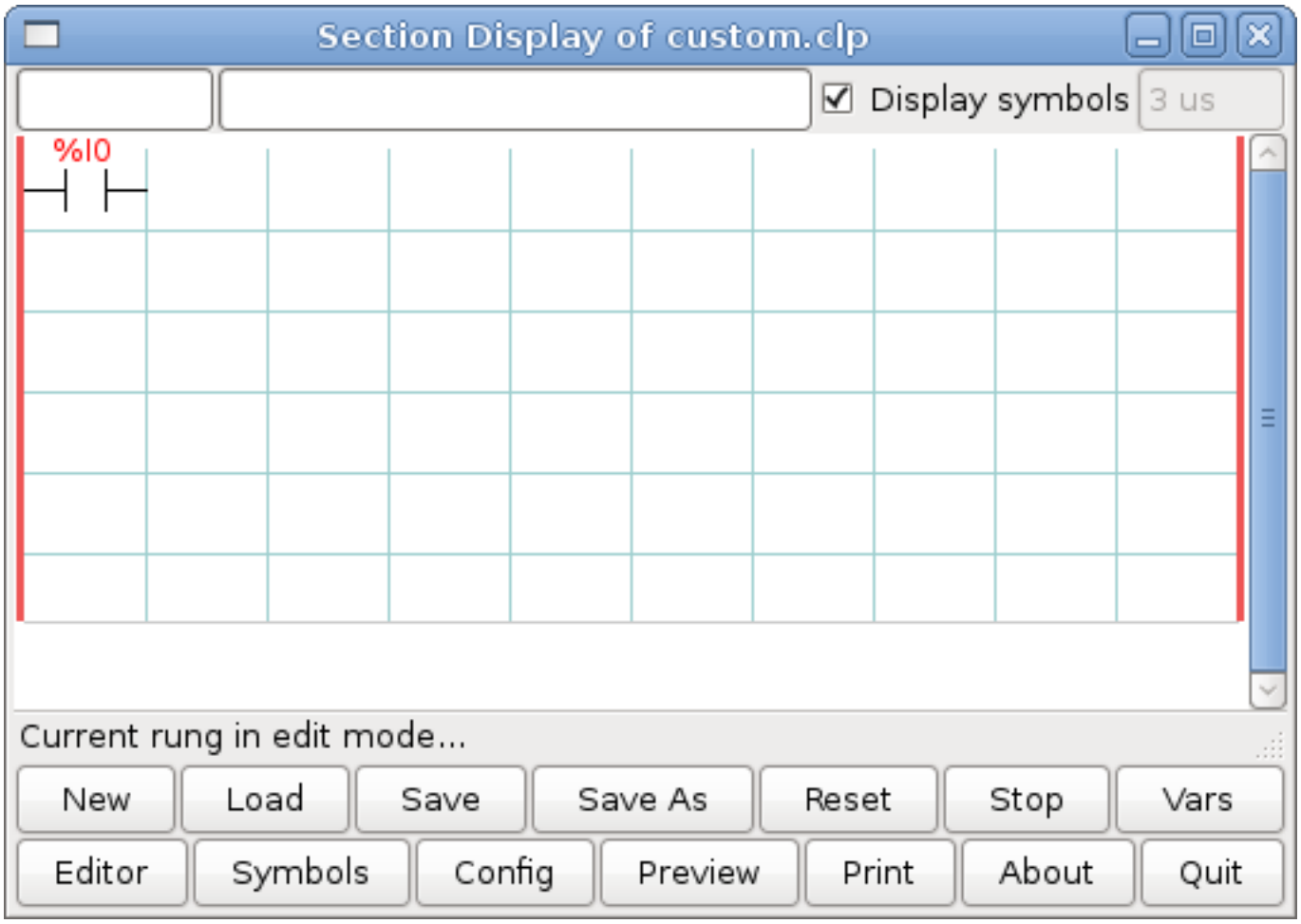


Figure 8.19: Відображення розділу з введенням даних

Повторіть вищевказані кроки, щоб додати вихід N.O. у верхній правій сітці, і використовуйте горизонтальне з’єднання, щоб з’єднати їх. Результат повинен виглядати так, як показано нижче. Якщо ні, використовуйте ластик, щоб видалити непотрібні ділянки.

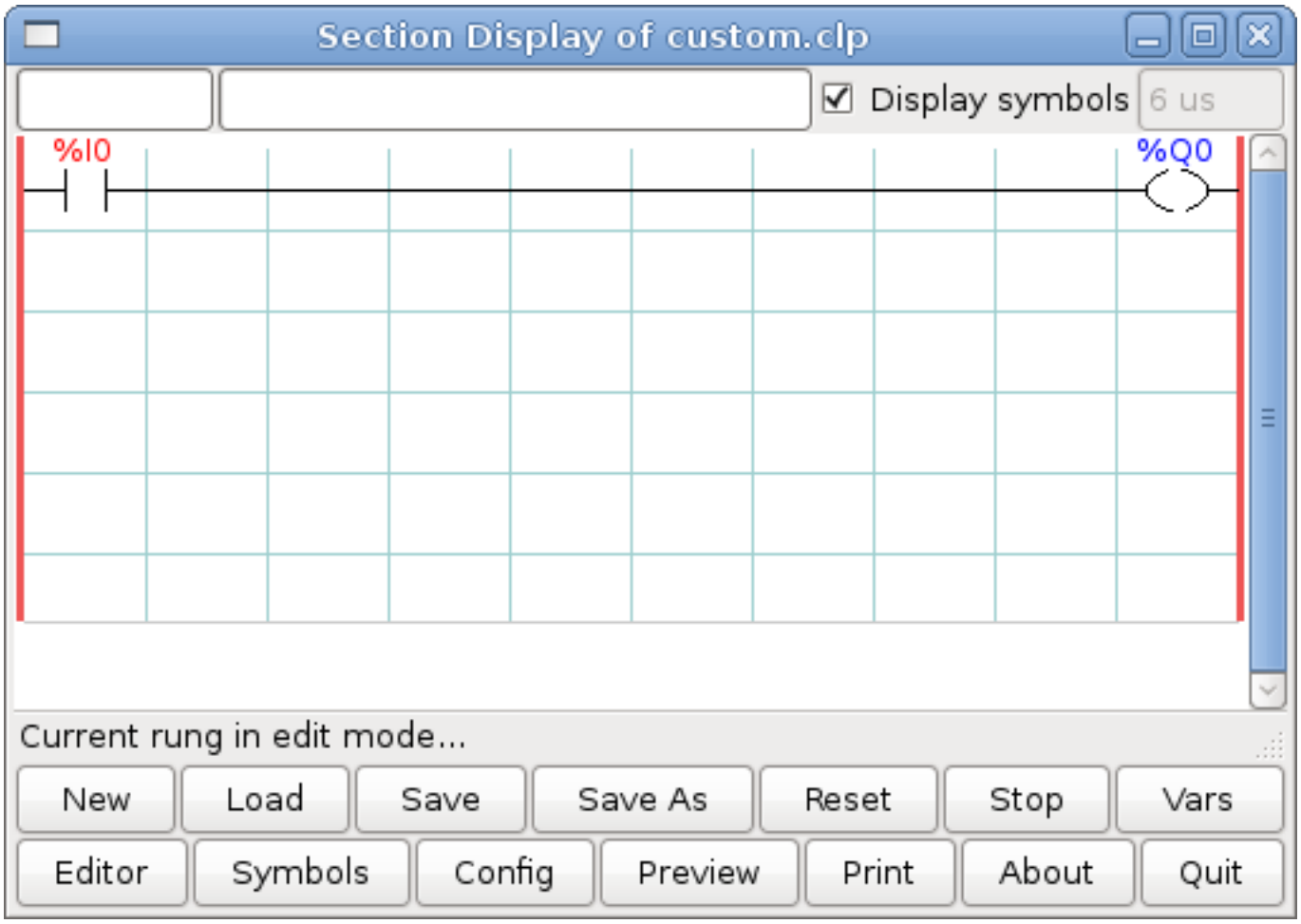


Figure 8.20: Відображення розділу з сходинкою

Тепер натисніть кнопку OK у вікні редактора. Тепер ваше відображення розділу має виглядати так:

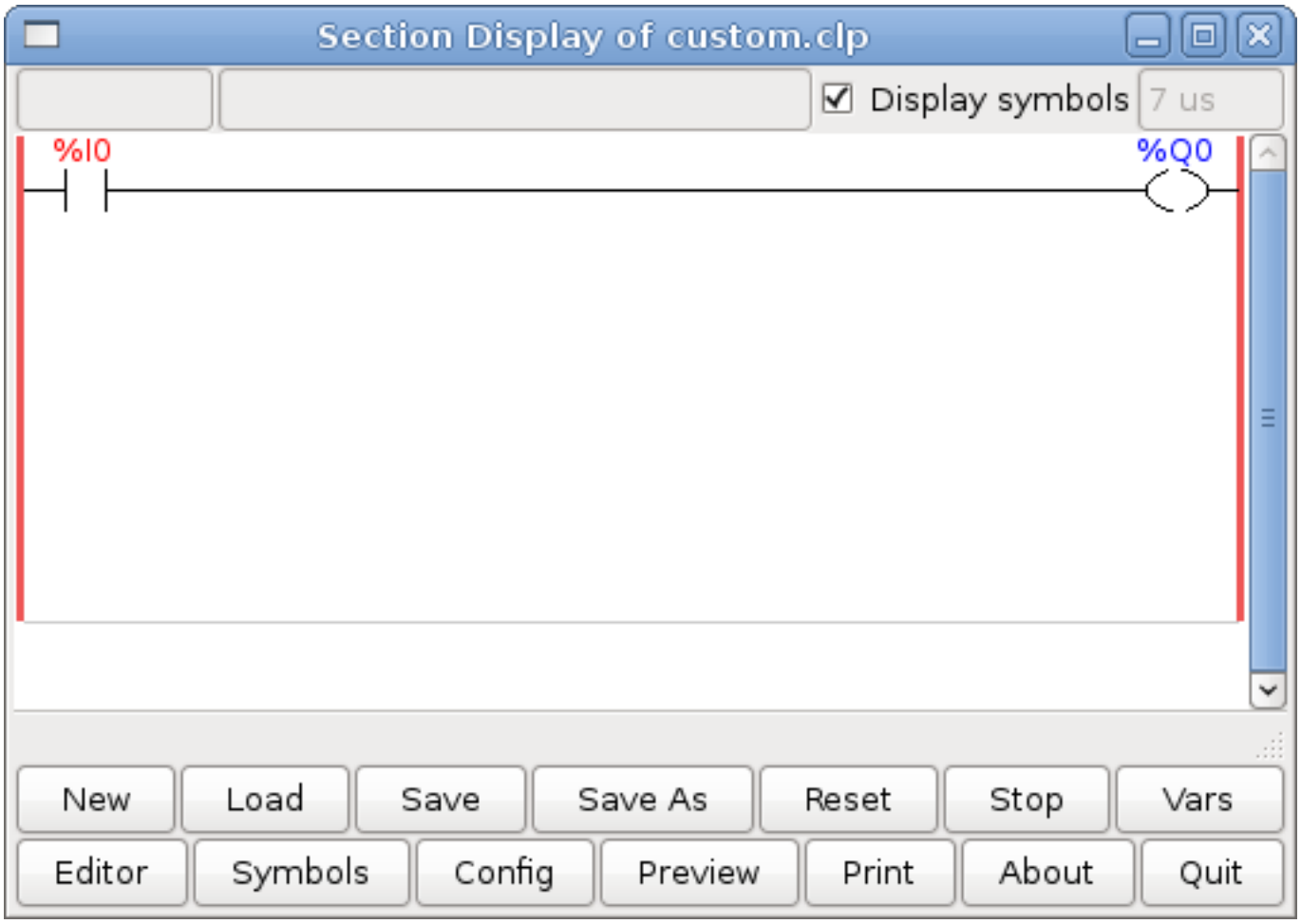


Figure 8.21: Відображення розділу завершено

Щоб зберегти новий файл, виберіть «Зберегти як» та введіть йому ім’я. Розширення .clp буде додано автоматично. За замовчуванням для його збереження має бути використано каталог запущеної конфігурації.

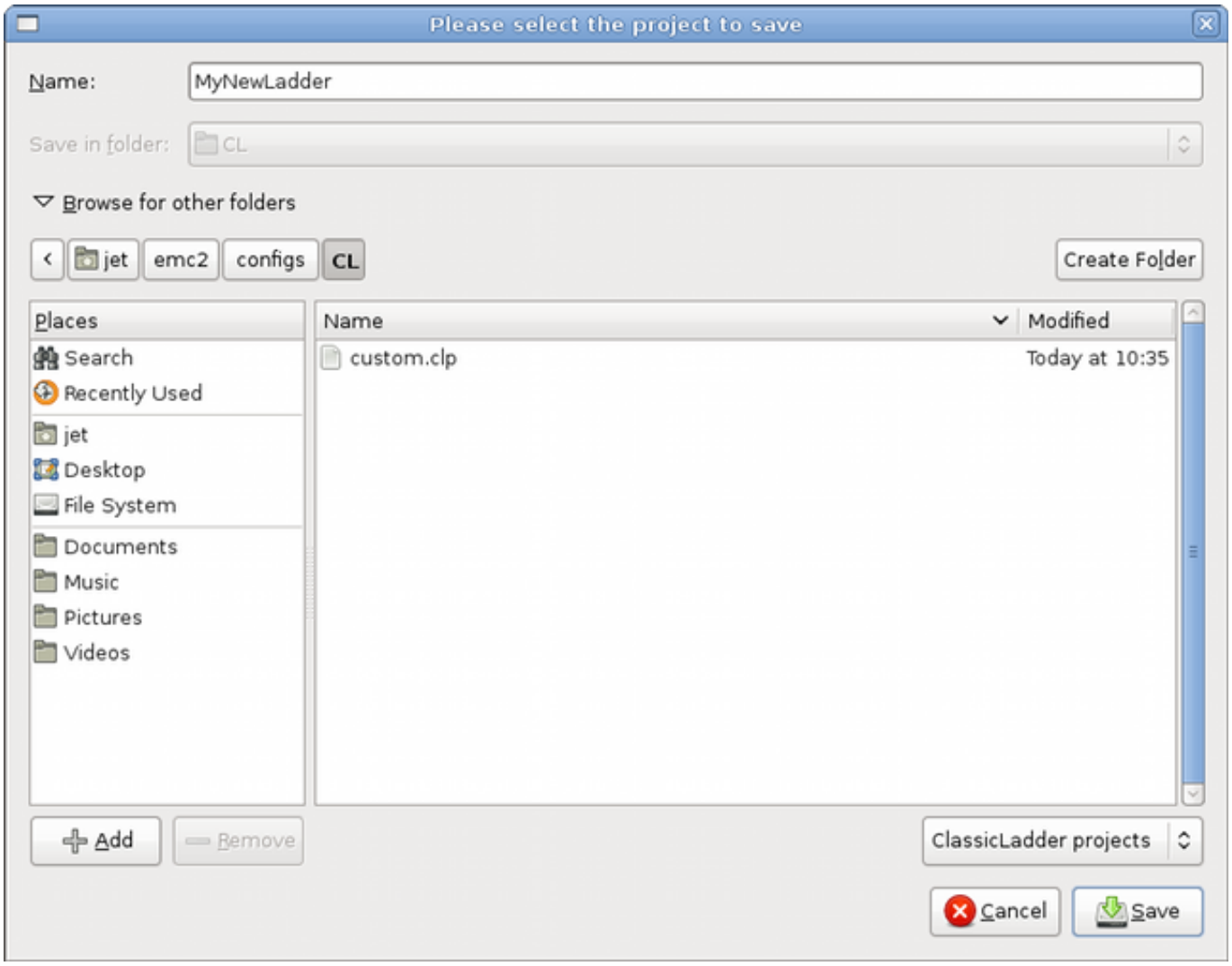


Figure 8.22: Діалогове вікно «Зберегти як»

Знову ж таки, якщо ви використовували майстер StepConf для додавання ClassicLadder, ви можете пропустити цей крок.

Щоб вручну додати сходинку, вам потрібно додати рядок до вашого файлу custom.hal, який завантажить ваш файл сходинки. Закрийте сеанс LinuxCNC та додайте цей рядок до вашого файлу custom.hal.

```
loadusr -w classicladder --nogui MyLadder.clp
```

Тепер, якщо ви запустите конфігурацію LinuxCNC, ваша програма рейдингу також буде працювати. Якщо ви виберете ”Файл/Редактор рейдингів”, створена вами програма відобразиться у вікні відображення розділу.

8.3 Приклади ClassicLadder

8.3.1 Лічильник упаковки

Щоб отримати лічильник, який «обертається», необхідно використовувати вивід попереднього встановлення та вивід скидання. При створенні лічильника встановіть попереднє значення на число, яке ви хочете досягти перед поверненням до 0. Логіка полягає в тому, що якщо значення лічильника перевищує попереднє значення, то лічильник скидається, а якщо підтікання увімкнено, то значення лічильника встановлюється на попереднє значення. Як ви можете бачити в прикладі, коли значення лічильника перевищує попередньо встановлене значення, запускається скидання лічильника, і значення стає 0. Вихід нижньої межі %Q2 встановить значення лічильника на попередньо встановлене значення при зворотному відліку.

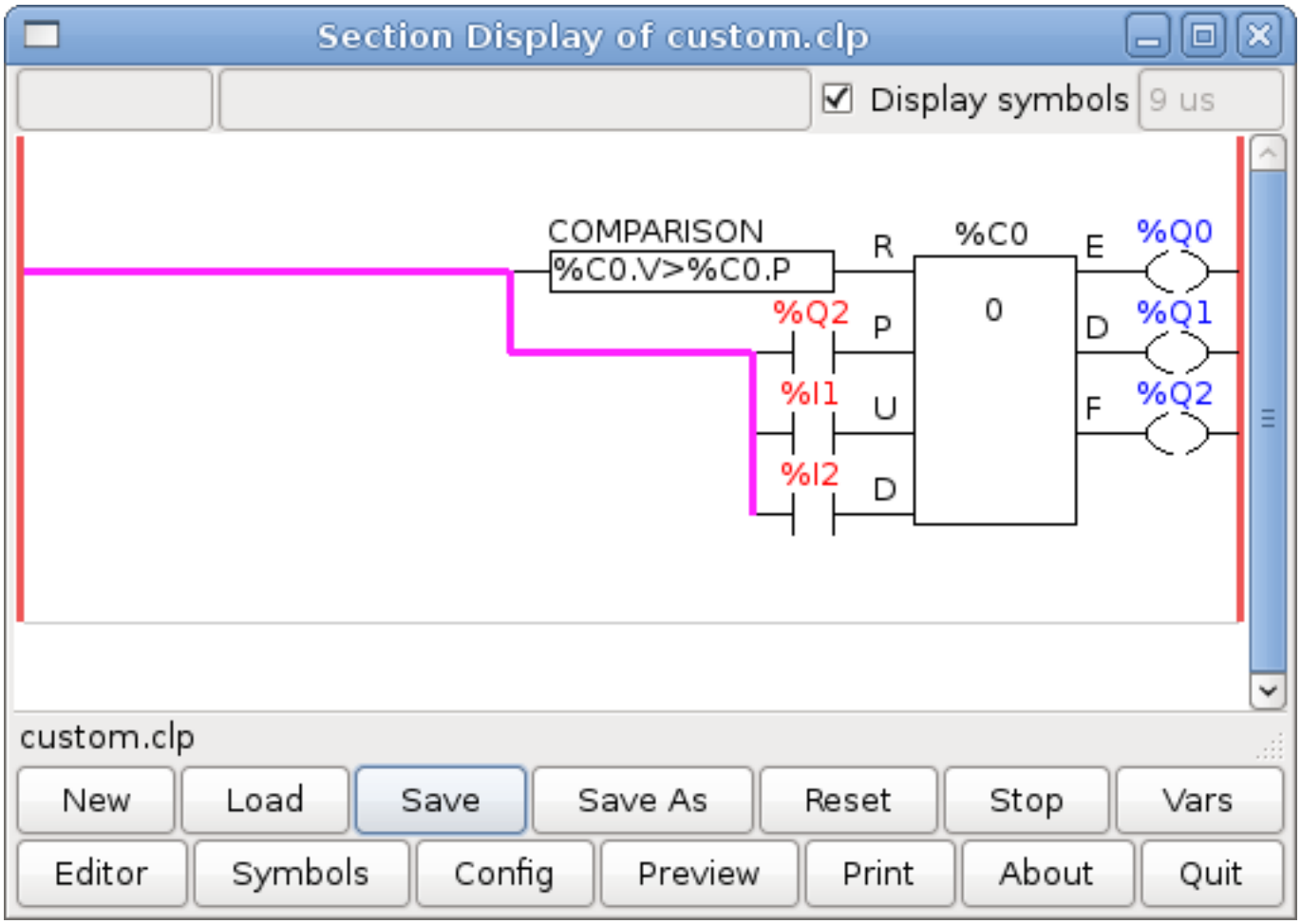


Figure 8.23: Лічильник упаковки

8.3.2 Відхилити зайві імпульси

Цей приклад показує, як відхилити зайві імпульси з входу. Припустимо, що вхідний імпульс %I0 має неприємну властивість видавати зайвий імпульс, який псує нашу логіку. TOF (Timer Off Delay) запобігає надходженню зайвого імпульсу до нашого очищеного виходу %Q0. Це працює таким чином: коли таймер отримує вхідний сигнал, вихід таймера залишається увімкненим протягом заданого часу. Використовуючи нормально закритий контакт %TM0.Q, вихід таймера

блокує будь-які подальші вхідні сигнали, що надходять до нашого виходу, до закінчення часу очікування.

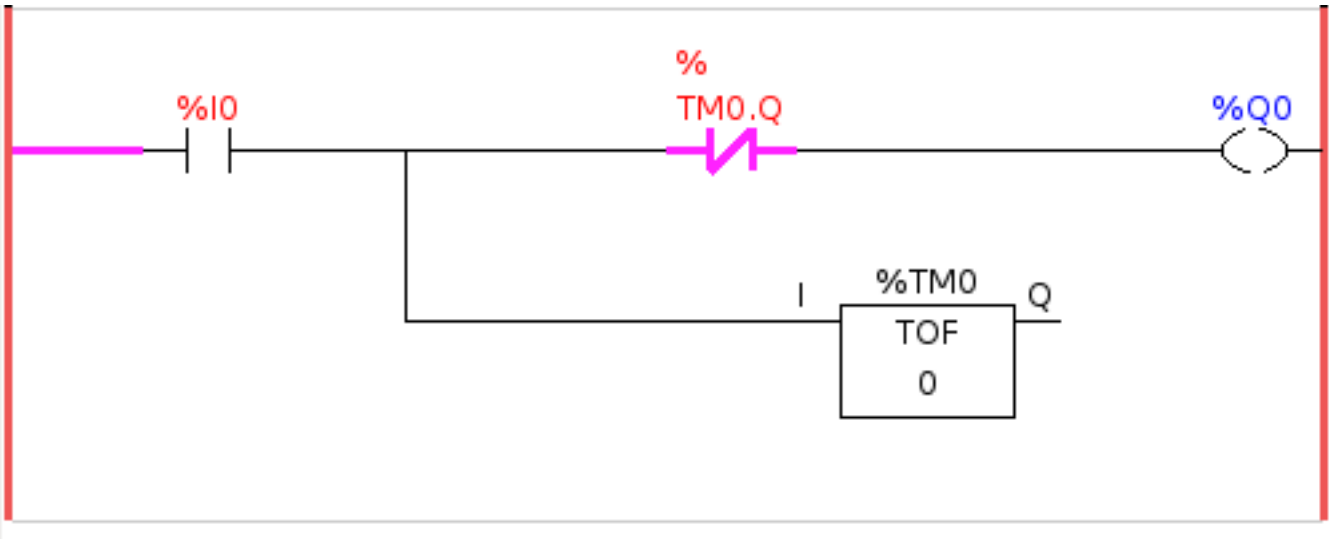


Figure 8.24: Відхилити додатковий імпульс

8.3.3 Зовнішній аварійний зупинник

Приклад зовнішньої аварійної зупинки знаходиться в папці /config/classicladder/cl-estop. Він використовує панель PyVCP для імітації зовнішніх компонентів.

Для підключення зовнішнього аварійного зупинника до LinuxCNC та забезпечення його спільної роботи із внутрішнім аварійним зупинником, потрібне кілька з'єднань через ClassicLadder.

Спочатку нам потрібно відкрити цикл E-Stop у головному файлі HAL, закоментувавши його, додавши знак фунта, як показано, або видаливши наступні рядки.

```
# net estop-out <= iocontrol.0.user-enable-out
# net estop-out => iocontrol.0.emc-enable-in
```

Далі ми додаємо ClassicLadder до нашого файлу custom.hal, додавши ці два рядки:

```
loadrt classicladder_rt
addf classicladder.0.refresh servo-thread
```

Далі ми запускаємо нашу конфігурацію та будуємо сходи, як показано тут.

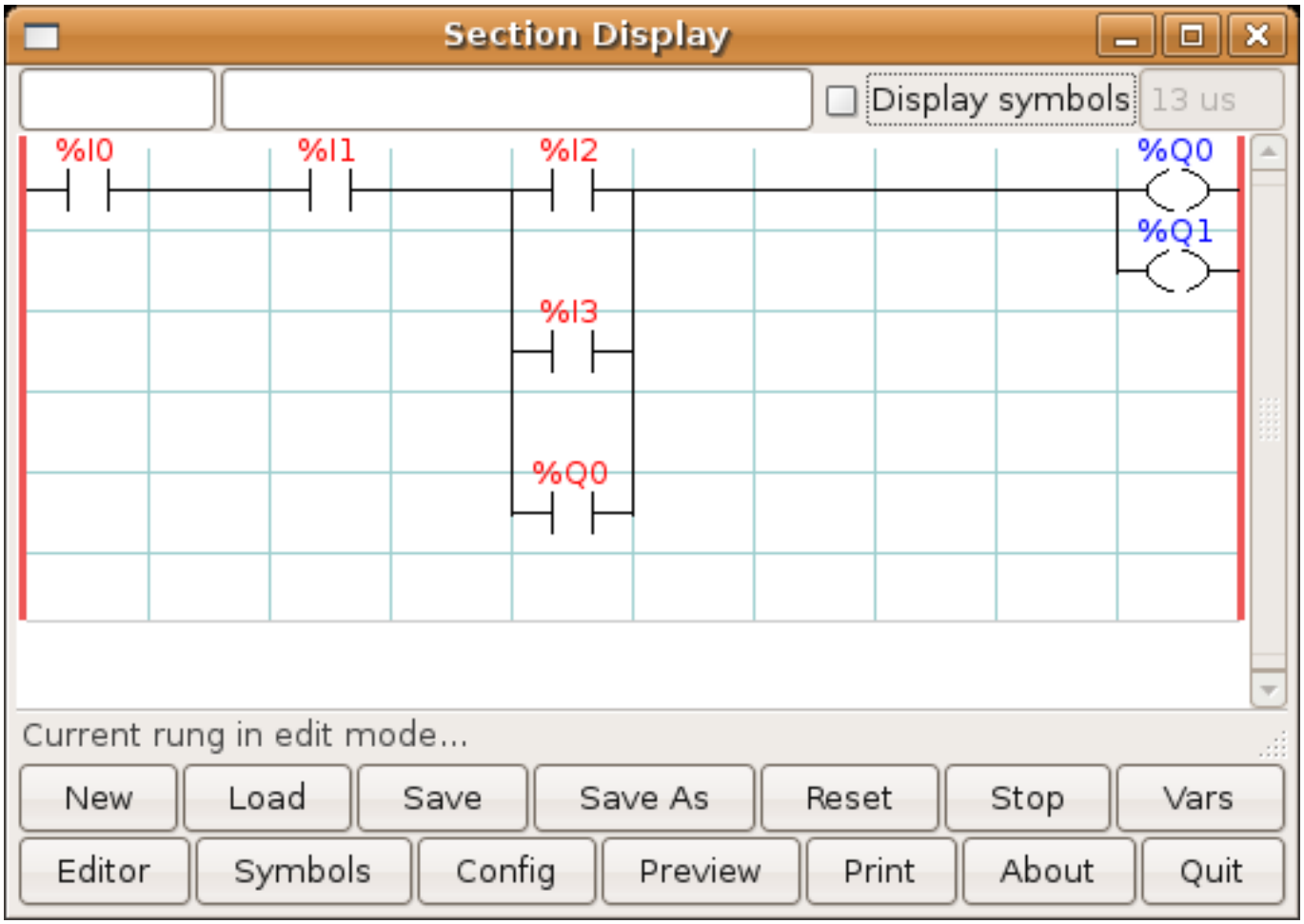


Figure 8.25: Дисплей секції аварійної зупинки

Після побудови драбини виберіть «Зберегти як» та збережіть драбину як estop.clp

Тепер додайте наступний рядок до вашого файлу custom.hal.

```
# b''3b''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''дb''b' ↩
    'pb''b''ab''b''бb''b''иb''b''нb''b''yb''
loadusr classicladder --nogui estop.clp
```

I/O завдання

- %I0 = Вхідні дані з панелі PyVCP, що імітує аварійну зупинку (прапорець)
- %I1 = Вхідні дані від аварійної зупинки LinuxCNC
- %I2 = Вхідні дані від імпульсу скидання аварійної зупинки LinuxCNC
- %I3 = Вхідні дані з кнопки скидання панелі PyVCP
- %Q0 = Вивід до LinuxCNC для активації
- %Q1 = Вихід на контакт увімкнення зовнішньої плати драйвера (використовуйте вихід N/C, якщо ваша плата має контакт вимикання)

Далі ми додаємо наступні рядки до файлу custom_postgui.hal

```
# b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b''ab''b''вb''b''ab''b''pb''b''ib''b' ↩
    'йb''b''нb''b''ob''b''їb'' b''зb''b''уb''b''пb''b''иb''b''нb''b''кb''b''иb'' b''зb'' b' ↩
    'вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb''b' ↩
    'мb'' b''кb''b''нb''b''ob''b''пb''b''ob''b''кb'' PyVCP b''дb''b''лb''b''яb'' b''ib''b' ↩
    'мb''b''ib''b''тb''b''ab''b''цb''b''ib''b''їb'' b''зb''b''ob''b''вb''b''нb''b''ib''b' ↩
    'шb''b''нb''b''ib''b''хb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b' ↩
    'тb''b''ib''b''вb''

# b''Kb''b''нb''b''ob''b''пb''b''кb''b''ab'' b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b' ↩
    'pb''b''кb''b''иb'' PyVCP b''ib''b''мb''b''ib''b''тb''b''уb''b''єb'' b''нb''b''ob''b' ↩
    'pb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ob'' b''зb''b''ab''b''мb''b''кb''b''нb''b' ↩
    'уb''b''тb''b''иb''b''йb'' b''зb''b''ob''b''вb''b''нb''b''ib''b''шb''b''нb''b''ib''b' ↩
    'йb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b''иb''b''йb'' b''cb''b''тb''b' ↩
    'ob''b''пb''
net ext-estop classicladder.0.in-00 <= pyvcp.py-estop

# b''3b''b''ab''b''пb''b''иb''b''тb'' b''нb''b''ab'' b''вb''b''вb''b''ib''b''мb''b''кb''b' ↩
    'нb''b''eb''b''нb''b''нb''b''яb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b' ↩
    'ob''b''їb'' b''зb''b''уb''b''пb''b''иb''b''нb''b''кb''b''иb'' b''вb''b''ib''b''дb''  ↩
    LinuxCNC
net estop-all-ok iocontrol.0.emc-enable-in <= classicladder.0.out-00

# b''3b''b''ab''b''пb''b''иb''b''тb'' b''нb''b''ab'' b''вb''b''вb''b''ib''b''мb''b''кb''b' ↩
    'нb''b''eb''b''нb''b''нb''b''яb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b' ↩
    'ob''b''їb'' b''зb''b''уb''b''пb''b''иb''b''нb''b''кb''b''иb'' b''вb''b''ib''b''дb''  ↩
    PyVCP b''ab''b''бb''b''ob'' b''зb''b''ob''b''вb''b''нb''b''ib''b''шb''b''нb''b''ьb''b' ↩
    'ob''b''гb''b''ob'' b''дb''b''жb''b''eb''b''pb''b''eb''b''лb''b''ab''
net ext-estop-reset classicladder.0.in-03 <= pyvcp.py-reset

# b''Цb''b''eb''b''йb'' b''pb''b''яb''b''дb''b''ob''b''кb'' b''cb''b''кb''b''иb''b''дb''b' ↩
    'ab''b''єb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b''уb'' b''зb''b''уb''b' ↩
    'пb''b''иb''b''нb''b''кb''b''уb'' b''зb'' LinuxCNC
net emc-reset-estop iocontrol.0.user-request-enable => classicladder.0.in-02

# b''Цb''b''eb''b''йb'' b''pb''b''яb''b''дb''b''ob''b''кb'' b''дb''b''ob''b''зb''b''вb''b' ↩
    'ob''b''лb''b''яb''b''єb'' LinuxCNC b''pb''b''ob''b''зb''b''бb''b''лb''b''ob''b''кb''b' ↩
    'уb''b''вb''b''ab''b''тb''b''иb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b' ↩
    'иb''b''йb'' b''cb''b''тb''b''ob''b''пb'' b''уb'' ClassicLadder
net emc-estop iocontrol.0.user-enable-out => classicladder.0.in-01

# b''Цb''b''яb'' b''лb''b''ib''b''нb''b''ib''b''яb'' b''вb''b''мb''b''иb''b''кb''b''ab''b' ↩
    'єb'' b''зb''b''eb''b''лb''b''eb''b''нb''b''иb''b''йb'' b''ib''b''нb''b''дb''b''иb''b' ↩
    'кb''b''ab''b''тb''b''ob''b''pb'' b''пb''b''ib''b''cb''b''лb''b''яb'' b''вb''b''иb''b' ↩
    'хb''b''ob''b''дb''b''уb'' b''зb'' b''ab''b''вb''b''ab''b''pb''b''ib''b''йb''b''нb''b' ↩
    'ob''b''їb'' b''зb''b''уb''b''пb''b''иb''b''нb''b''кb''b''иb''.
net estop-all-ok => pyvcp.py-es-status
```

Далі ми додаємо наступні рядки до файлу panel.xml. Зверніть увагу, що його потрібно відкрити за допомогою текстового редактора, а не за допомогою програми для перегляду html за замовчуванням.

```
<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>
```

Тепер запустіть вашу конфігурацію, і вона має виглядати ось так.

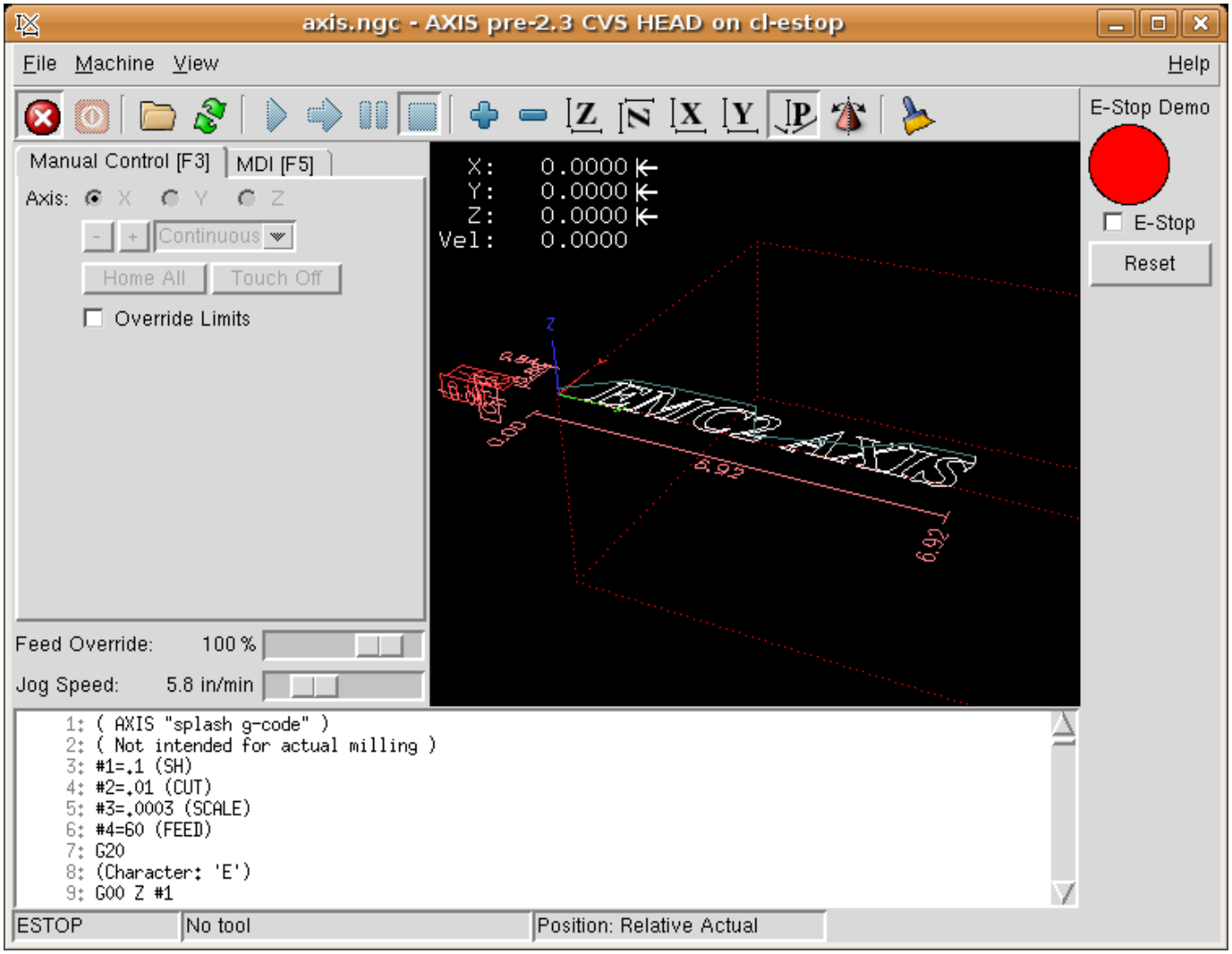


Figure 8.26: Екстрена зупинка AXIS

Зверніть увагу, що в цьому прикладі, як і в реальному житті, ви повинні скинути дистанційну аварійну зупинку (імітовану прапорцем) перед тим, як аварійна зупинка AXIS або зовнішній скид переведуть вас у режим ВИМКНЕНО. Якщо на екрані AXIS була натиснута аварійна зупинка, ви повинні натиснути її ще раз, щоб скинути. Ви не можете виконати скид із зовнішнього пристрою після того, як ви виконали аварійну зупинку в AXIS.

8.3.4 Приклад таймера/операції

У цьому прикладі ми використовуємо блок Operate для призначення значення попередньому налаштуванню таймера залежно від того, чи вхід увімкнено чи вимкнено.

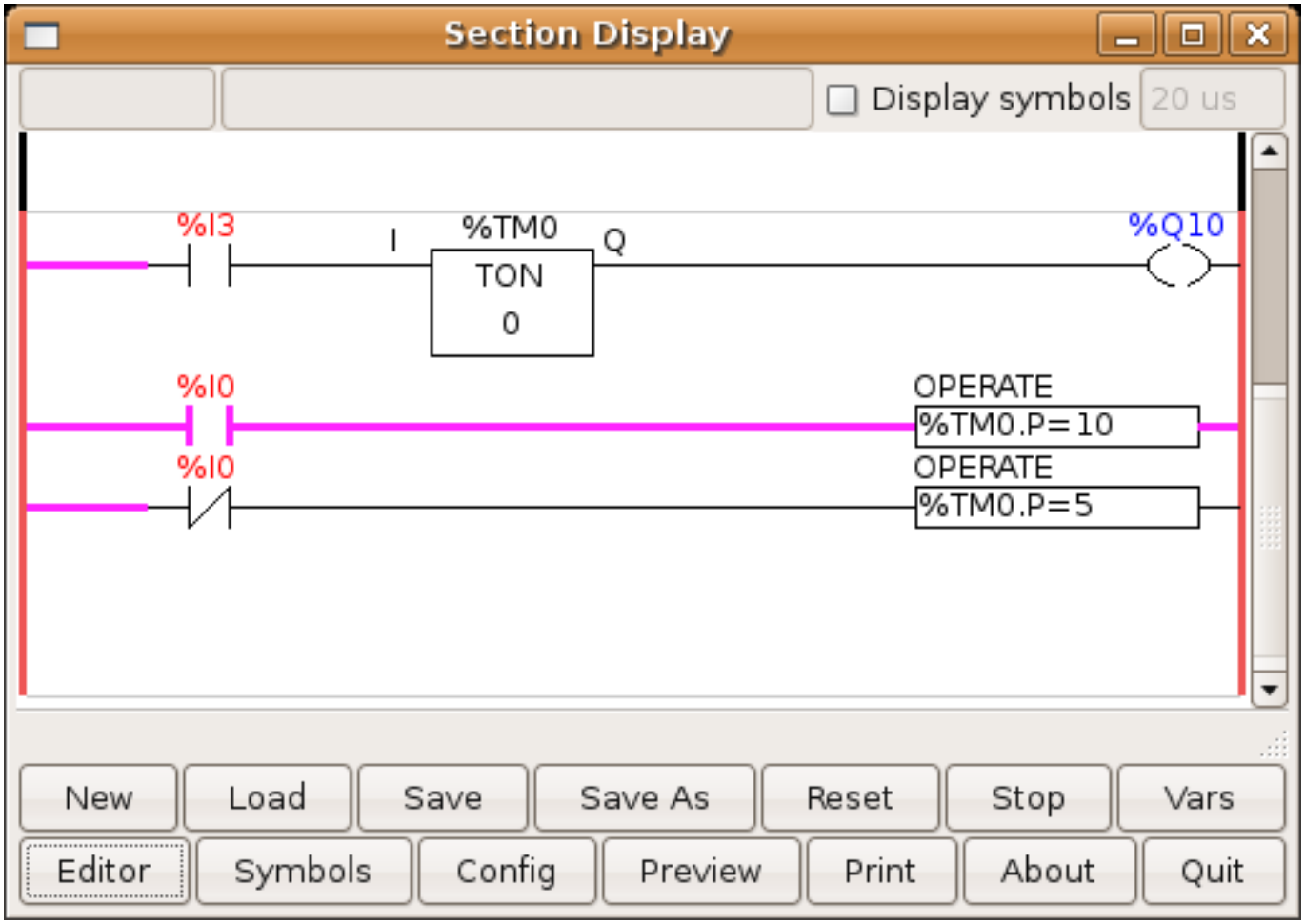


Figure 8.27: Приклад таймера/операції

У цьому випадку %I0 має значення true, тому значення попереднього налаштування таймера дорівнює 10. Якщо %I0 має значення false, попередньо налаштоване значення таймера дорівнюватиме 5.

Chapter 9

Розширені теми

9.1 Кінематика

9.1.1 Вступ

Коли ми говоримо про верстати з CNC, зазвичай маємо на увазі машини, яким дають команду переміститися в певне місце і виконати різні завдання. Щоб мати єдине уявлення про простір верстата і пристосувати його до людського сприйняття тривимірного простору, більшість верстатів (якщо не всі) використовують загальну систему координат, яка називається декартовою системою координат.

Декартова система координат складається з трьох осей (X, Y, Z), кожна з яких перпендикулярна до двох інших. Примітка: [Слово «осі» також часто (і неправильно) використовується при обговоренні верстатів з CNC та згадці про напрямки руху верстата.].

Коли ми говоримо про програму G-коду (RS274/NGC), ми маємо на увазі низку команд (G0, G1 тощо), які мають позиції як параметри (X- Y- Z-). Ці позиції точно відповідають декартовим позиціям. Частина контролера руху LinuxCNC відповідає за перетворення цих позицій у позиції, які відповідають кінематиці верстата примітка:[Кінематика: двостороння функція перетворення з декартового простору в простір з’єднання.].

9.1.1.1 Суглоби проти осей

Суглоб верстата з CNC є одним із фізичних ступенів свободи верстата. Він може бути лінійним (гвинтові шнеки) або обертовим (поворотні столи, суглоби маніпулятора робота). На даному верстаті може бути будь-яка кількість суглобів. Наприклад, один популярний робот має 6 суглобів, а типовий простий фрезерний верстат має лише 3.

Існують певні машини, в яких з’єднання розташовані відповідно до кінематичних осей (з’єднання 0 вздовж осі X, з’єднання 1 вздовж осі Y, з’єднання 2 вздовж осі Z), і ці машини називаються декартовими машинами (або машинами з тривіальною кінематикою). Це найпоширеніші машини, що використовуються у фрезеруванні, але вони не дуже поширені в інших сферах управління машинами (наприклад, зварювання: роботи типу puma).

LinuxCNC підтримує осі з назвами: X Y Z A B C U V W. Осі X Y Z зазвичай відносяться до звичайних декартових координат. Осі A B C відносяться до координат обертання навколо осей X Y Z відповідно. Осі U V W відносяться до додаткових координат, які зазвичай є колінеарними до осей X Y Z відповідно.

9.1.2 Тривіальна кінематика

Найпростіші машини — це ті, в яких кожен шарнір розміщений вздовж однієї з декартових осей. На цих машинах відображення з декартового простору (програма G-коду) на простір шарнірів (фактичні приводи машини) є тривіальним. Це просте відображення 1:1:

```
pos->tran.x = joints[0];
pos->tran.y = joints[1];
pos->tran.z = joints[2];
```

У наведеному вище фрагменті коду можна побачити, як здійснюється відображення: положення по осі X ідентичне суглобу 0, положення по осі Y — суглобу 1 тощо. Вищезазначене стосується прямої кінематики (один напрямок перетворення). Наступний фрагмент коду стосується оберненої кінематики (або оберненого напрямку перетворення):

```
joints[0] = pos->tran.x;
joints[1] = pos->tran.y;
joints[2] = pos->tran.z;
```

У LinuxCNC кінематика ідентичності реалізована за допомогою кінематичного модуля «trivkins» і розширена до 9 осей. Стандартні співвідношення між координатами осей і номерами з’єднань такі: примітка:[Якщо верстат (наприклад, токарний) оснащений тільки осями X, Z і A, а файл INI LinuxCNC містить тільки визначення цих 3 з’єднань, то попереднє твердження є хибним. Оскільки наразі ми маємо (joint0=X, joint1=Z, joint2=A), що передбачає joint1=Y. Щоб це працювало в LinuxCNC, просто визначте всі осі (XYZA), тоді LinuxCNC використовуватиме простий цикл в HAL для невикористаної осі Y.] Примітка: [Інший спосіб зробити це — змінити відповідний код і перекомпілювати програмне забезпечення.]

```
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];
```

Аналогічно, співвідношення за замовчуванням для оберненої кінематики тривкінів такі:

```
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;
```

Перетворення для тривіальної машини ”кінсів” (кінематика ”трівкінсів”) або декартової машини легко виконати за умови, що немає пропусків у використаних літерах осей.

Ситуація дещо ускладнюється, якщо в машині відсутня одна або кілька літер осей. Проблема пропущених літер осей вирішується за допомогою параметра модуля «coordinates=» з модулем trivkins. Номери з’єднань присвоюються послідовно кожній вказаній координаті. Токарний верстат можна описати за допомогою «coordinates=xz». Тоді призначення з’єднань будуть такими:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.z
```

Використання параметра «coordinates=» рекомендується для конфігурацій, в яких не вказані літери осей. Примітка: Історично модуль trivkins не підтримував параметр «coordinates=», тому конфігурації токарних верстатів часто налаштовувалися як машини XYZ. Невикористана вісь Y налаштовувалася на 1) негайне повернення в початкове положення, 2) використання простого зворотного зв’язку для підключення виводу HAL команди положення до виводу HAL зворотного зв’язку положення, і 3) приховування в графічному інтерфейсі. Численні конфігурації sim використову ці методи для спільного використання загальних файлів HAL.]

Модуль кінематики «trivkins» також дозволяє вказати одну і ту ж координату для декількох з’єднань. Ця функція може бути корисною на таких машинах, як портал з двома незалежними двигунами для координати y. Така машина може використовувати «coordinates=xyyz», що призведе до призначення з’єднань:

```
joints[0] = pos->tran.x
joints[1] = pos->tran.y
joints[2] = pos->tran.y
joints[3] = pos->tran.z
```

Дивіться сторінки довідки trivkins для отримання додаткової інформації.

9.1.3 Нетривіальна кінематика

Існує чимало типів налаштувань машин (роботи: puma, scara; гексаподи тощо). Кожен з них налаштовується за допомогою лінійних і поворотних з’єднань. Ці з’єднання зазвичай не відповідають декартовим координатам, тому нам потрібна кінематична функція, яка виконує перетворення (насправді 2 функції: пряма і обернена кінематична функція).

Для ілюстрації вищесказаного розглянемо просту кінематику під назвою сошка (спрощена версія штатива, яка є спрощеною версією гексапода).

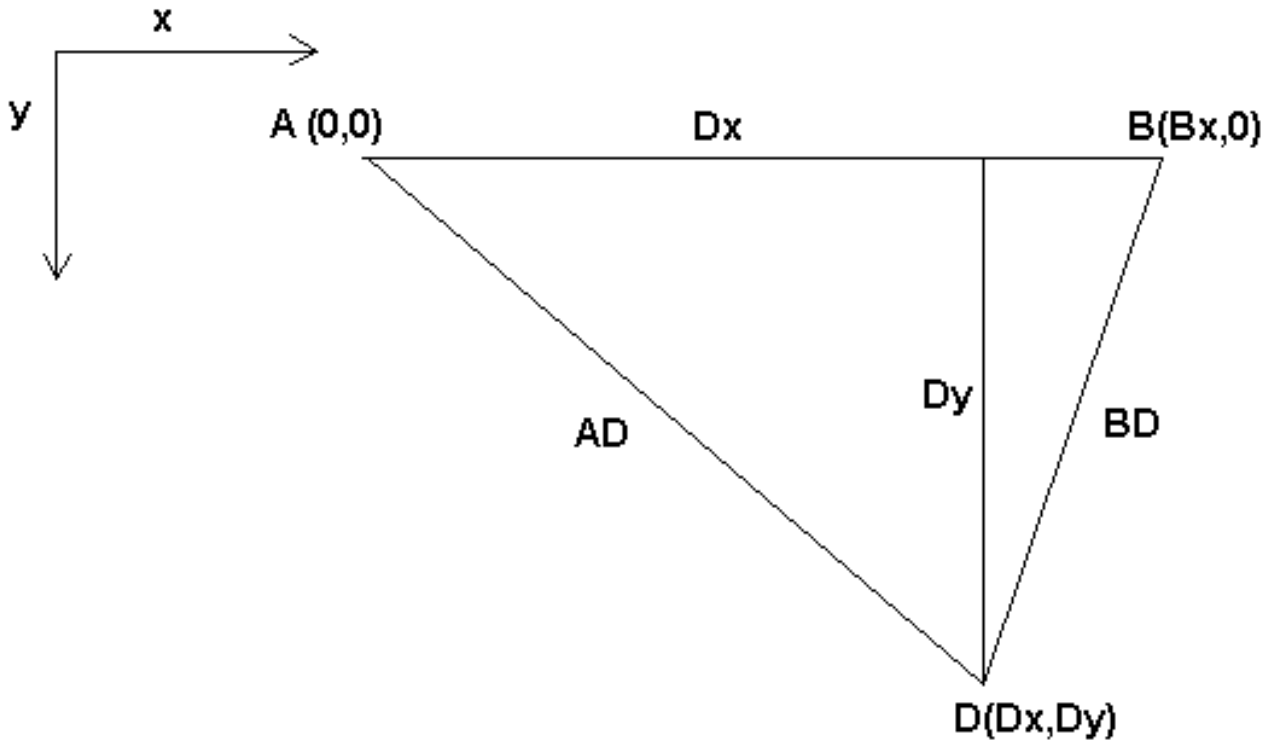


Figure 9.1: Налаштування сошок

Двонога, про яку ми говоримо, — це пристрій, що складається з 2 двигунів, розміщених на стіні, до яких за допомогою дроту підвішено пристрій. У цьому випадку суглобами є відстані від двигунів до пристрою (на малюнку позначені AD і BD).

Положення двигунів фіксується за угодою. Двигун A знаходиться в точці (0,0), що означає, що його координата X дорівнює 0, а координата Y також дорівнює 0. Двигун B розміщений в точці (Bx, 0), що означає, що його координата X дорівнює Bx.

Наша підказка буде в точці D, яка визначається відстанями AD та BD, а також декартовими координатами Dx, Dy.

Завдання кінематики полягає в перетворенні довжин суглобів (AD, BD) на декартові координати (Dx, Dy) і навпаки.

9.1.3.1 Пряма трансформація

Для перетворення з простору суглобів у декартовий простір ми скористаємося деякими правилами тригонометрії (прямокутні трикутники, визначені точками (0,0), (Dx,0), (Dx,Dy) та трикутниками (Dx,0), (Bx,0) та (Dx,Dy)).

Ми можемо легко побачити, що:

$$AD^2 = x^2 + y^2 \; BD^2 = (Bx - x)^2 + y^2$$

так само:

$$BD^2 = (Bx - x)^2 + y^2$$

Якщо відняти одне від іншого, то отримаємо:

$$AD^2 - BD^2 = x^2 + y^2 - x^2 + 2 * x * Bx - Bx^2 - y^2$$

і тому:

$$x = \frac{AD^2 - BD^2 + Bx^2}{2 * Bx}$$

Звідти ми обчислюємо:

$$y = \sqrt{AD2 - x2}$$

Зверніть увагу, що обчислення для y включає квадратний корінь з різниці, що може не давати дійсного числа. Якщо для цього положення з'єднання немає єдиної декартової координати, то таке положення називається сингулярністю. У цьому випадку пряма кінематика повертає -1.

Перекладено у фактичний 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 Зворотне перетворення

Зворотна кінематика в нашому прикладі набагато простіше, оскільки ми можемо записати її безпосередньо:

$$AD = \sqrt{x^2 + y^2}$$

$$BD = \sqrt{(Bx - x)^2 + y^2}$$

або переведено у фактичний 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 Деталі впровадження

Кінематичний модуль реалізовано як компонент HAL і має дозвіл на експорт виводів та параметрів. Він складається з кількох функцій "C" (на відміну від функцій HAL):

```
int kinematicsForward(const double *joint, EmcPose *world,
const KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Реалізує функцію forward kinematics.

```
int kinematicsInverse(const EmcPose * world, double *joints,
const KINEMATICS_INVERSE_FLAGS *iflags,
KINEMATICS_FORWARD_FLAGS *fflags)
```

Реалізує обернену кінематичну функцію.

```
KINEMATICS_TYPE kinematicsType(void)
```

Повертає ідентифікатор типу кінематики, типовий варіант *KINEMATICS_BOTH*:

1. KINEMATICS_IDENTITY (кожен номер шарніра відповідає літері осі)
2. KINEMATICS_BOTH (надаються прямі та зворотні кінематичні функції)
3. KINEMATICS_FORWARD_ONLY
4. KINEMATICS_INVERSE_ONLY

Note
Графічні інтерфейси можуть інтерпретувати KINEMATICS_IDENTITY, щоб приховати відмінності між номерами шарнірів та літерами осей у режимі шарніра (зазвичай перед переміщенням до початкової точки).

```
int kinematicsSwitchable(void)
int kinematicsSwitch(int switchkins_type)
KINS_NOT_SWITCHABLE
```

Функція kinematicsSwitchable() повертає 1, якщо підтримується кілька типів кінематики. Функція kinematicsSwitch() вибирає тип кінематики. Див. Switchable Kinematitcs.

Note
Більшість наданих кінематичних модулів підтримують один тип кінематики і використовують директиву "**KINS_NOT_SWITCHABLE**" для надання значень за замовчуванням для необхідних функцій kinematicsSwitchable() і kinematicsSwitch().

```
int kinematicsHome(EmcPose *world, double *joint,
KINEMATICS_FORWARD_FLAGS *fflags,
KINEMATICS_INVERSE_FLAGS *iflags)
```

Функція кінематики вихідного положення встановлює всі свої аргументи на відповідні значення у відомому вихідному положенні. При виклику ці значення повинні бути встановлені, якщо вони відомі, на початкові значення, наприклад, з файлу INI. Якщо кінематика вихідного положення може приймати довільні початкові точки, слід використовувати ці початкові значення.

```
int rtapi_app_main(void)
void rtapi_app_exit(void)
```

Це стандартні функції налаштування та демонтажу модулів RTAPI.

Коли кінематичні модулі містяться в одному вихідному файлі, їх можна скомпілювати та встановити за допомогою «halcompile». Див. сторінку довідки «halcompile(1)» або посібник HAL для отримання додаткової інформації.

9.1.4.1 Кінематичний модуль з використанням шаблону userkins.comp

Інший спосіб створити власний кінематичний модуль – це адаптувати компонент HAL «userkins». Цей шаблонний компонент може бути змінений локально користувачем і може бути зібраний за допомогою halcompile.

Дивіться сторінки довідки userkins для отримання додаткової інформації.

Зауважте, що для створення перемиканих кінематичних модулів необхідні модифікації дещо складніші.

Див. «millturn.comp» як приклад перемиканого кінематичного модуля, створеного за допомогою шаблону «userkins.comp».

9.2 Налаштування ”модифікованих” параметрів Денавіта-Харте (DH) для ”генсеркінів”

9.2.1 Прелюдія

LinuxCNC підтримує низку кінематичних модулів, включаючи один, який підтримує узагальнений набір послідовної кінематики, що зазвичай визначається за допомогою параметрів Денавіта-Хартенберга.

У цьому документі показано метод налаштування DH-параметрів для Mitsubishi RV-6SDL у LinuxCNC з використанням кінематики «genserkins».

Note
Цей документ не охоплює створення моделі «vismach», яка, хоча і є безперечно дуже корисною, вимагає настільки ж ретельного моделювання, якщо вона має відповідати моделі «genserkins», виведеній у цьому документі.

Note
Можливі помилки та/або недоліки – використовуйте на свій страх і ризик!

9.2.2 Загальне

З поширенням промислових роботів зростає інтерес до управління вживаними роботами за допомогою LinuxCNC. Поширеним типом роботів, що використовуються в промисловості та виробництві, є «послідовний маніпулятор», спроектований як серія моторизованих шарнірів, з’єднаних жорсткими ланками. Послідовні роботи часто мають шість шарнірів, необхідних для шести ступенів свободи, потрібних як для позиціонування (XYZ), так і для орієнтації (ABC або крен, рискання) об’єкта в просторі. Часто ці роботи мають конструкцію руки, яка простягається від основи до кінцевого виконавчого механізму.

Управління таким серійним роботом вимагає обчислення положення та орієнтації кінцевого виконавчого механізму відносно опорної системи координат, коли кути згину суглобів відомі (**пряма кінематика**), а також більш складного зворотного обчислення необхідних кутів згину суглобів для заданого положення та орієнтації кінцевого виконавчого механізму відносно опорної системи координат (**обернена кінематика**). Стандартними математичними інструментами, що використовуються для цих розрахунків, є матриці, які в основному являють собою таблиці параметрів і формул, що полегшують обробку обертань і перетворень, які використовуються в розрахунках прямої та зворотної кінематики.

Детальне знання математики не є необхідним для серійного робота, оскільки LinuxCNC надає кінематичний модуль, який реалізує алгоритм під назвою «genserkins» для обчислення прямої та оберненої кінематики для загального серійного робота. Для управління конкретним серійним роботом «genserkins» необхідно надати дані, щоб він міг побудувати математичну модель механічної структури робота і, таким чином, виконати обчислення.

Необхідні дані повинні бути в стандартизованій формі, яка була введена Жаком Денавітом і Річардом Хартенбергом ще в п’ятдесятих роках і називається DH-параметрами. Денавіт і Хартенберг використовували чотири параметри для опису того, як один суглоб пов’язаний з наступним. Ці параметри описують в основному два обертання («альфа» і «тета») і два перенесення («a» і «d»).

9.2.3 Змінені DH-параметри

Як це часто буває, цей «стандарт» був модифікований іншими авторами, які ввели «модифіковані параметри DH», і потрібно бути дуже обережним, оскільки «genserkins» використовує «модифіковані параметри DH», як описано в публікації «Вступ до робототехніки, механіки та управління» Джона Дж. Крейга. Зверніть увагу, що про «параметри DH» можна знайти багато інформації, але автори рідко вказують, яка саме конвенція використовується. Крім того, деякі люди вважають за необхідне змінити параметр з «a» на «r», що ще більше ускладнює ситуацію. Цей документ дотримується конвенції, викладеної у вищезгаданій публікації Крейга, з тією різницею, що нумерація суглобів і параметрів починається з цифри «0», щоб відповідати «genserkins» і його контактам HAL.

Стандартні та модифіковані параметри DH складаються з чотирьох числових значень для кожного суглоба («a», «d», «alpha» та «theta»), які описують, як система координат (CS), що знаходиться в одному суглобі, повинна бути переміщена та обертана, щоб вирівнятися з наступним суглобом. Вирівнювання означає, що вісь Z нашої CS збігається з віссю обертання суглоба і вказує в позитивному напрямку таким чином, що, використовуючи правило правої руки, коли великий палець вказує в позитивному напрямку осі Z, пальці вказують в позитивному напрямку обертання суглоба. Стає зрозуміло, що для цього необхідно визначити позитивні напрямки всіх суглобів, перш ніж починати виводити параметри!

Різниця між «стандартними» та «модифікованими» позначеннями полягає в тому, як параметри розподіляються між посиланнями. Використання «стандартних» параметрів DH у «genserkins» **не** дасть правильної математичної моделі.

9.2.4 Модифіковані DH-параметри, що використовуються в ”genserkins”

Зверніть увагу, що «genserkins» не обробляє зміщення до значень тета — тета є змінною суглоба, яка **контролюється** LinuxCNC. Коли CS вирівняно з суглобом, обертання навколо його осі Z ідентичне обертанню, яке LinuxCNC командує цьому суглобу. Це унеможливлює довільне визначення положення 0° суглобів наших роботів.

Три налаштовувані параметри:

1. **alpha** : додатне або від’ємне обертання (у радіанах) навколо осі X ”поточної системи координат”
2. **a** : додатна відстань вздовж осі X між двома осями з’єднання, задана в «машинних одиницях» (мм або дюймах), визначених у системному INI-файлі.
3. **d**: додатна або від’ємна довжина вздовж Z (також у «машинних одиницях»)

Набори параметрів завжди виводяться в одному і тому ж порядку, а набір завершується встановленням параметра d. Це не залишає вісь Z нашої CS вирівняною з наступним з’єднанням! Це може здатися заплутаним, але дотримання цього правила дасть робочий набір параметрів. Після встановлення параметра **d** вісь X нашої CS повинна вказувати на вісь наступного з’єднання.

9.2.5 Нумерація суглобів та параметрів

Перший шарнір в LinuxCNC - це joint-0 (оскільки в програмному забезпеченні нумерація починається з 0), тоді як у більшості публікацій нумерація починається з цифри «1». Це стосується також усіх параметрів. Тобто нумерація починається з a-0, alpha-0, d-0 і закінчується a-5, alpha-5 і d-5. Майте це на увазі, коли слідуєте публікації для налаштування параметрів «genserkins».

9.2.6 Як почати

Домовленість полягає в тому, щоб почати з розміщення опорної комп’ютерної точки (CS) в основі робота так, щоб її вісь Z збігалася з віссю першого суглоба, а вісь X була спрямована в бік осі наступного суглоба.

Це також призведе до того, що значення DRO в LinuxCNC будуть посилатися на цю точку. Після цього a-0 і alpha-0 встановлюються на 0. У вищезгаданій публікації (Крейг) також встановлюється d-0 на 0, що викликає плутанину, коли для отримання опорної CS внизу основи необхідне зміщення. Встановлення d-0 = на зміщення дає правильні результати. Таким чином, перший набір параметрів становить alpha-0 = 0, a-0 = 0, d0 = зміщення, а вісь X CS вказує на вісь наступного з’єднання (joint-1).

Виведення чистого набору (альфа-1, a-1, d-1) відбувається далі — завжди використовуючи ту саму послідовність аж до шостого набору (альфа-5, a-5, d-5).

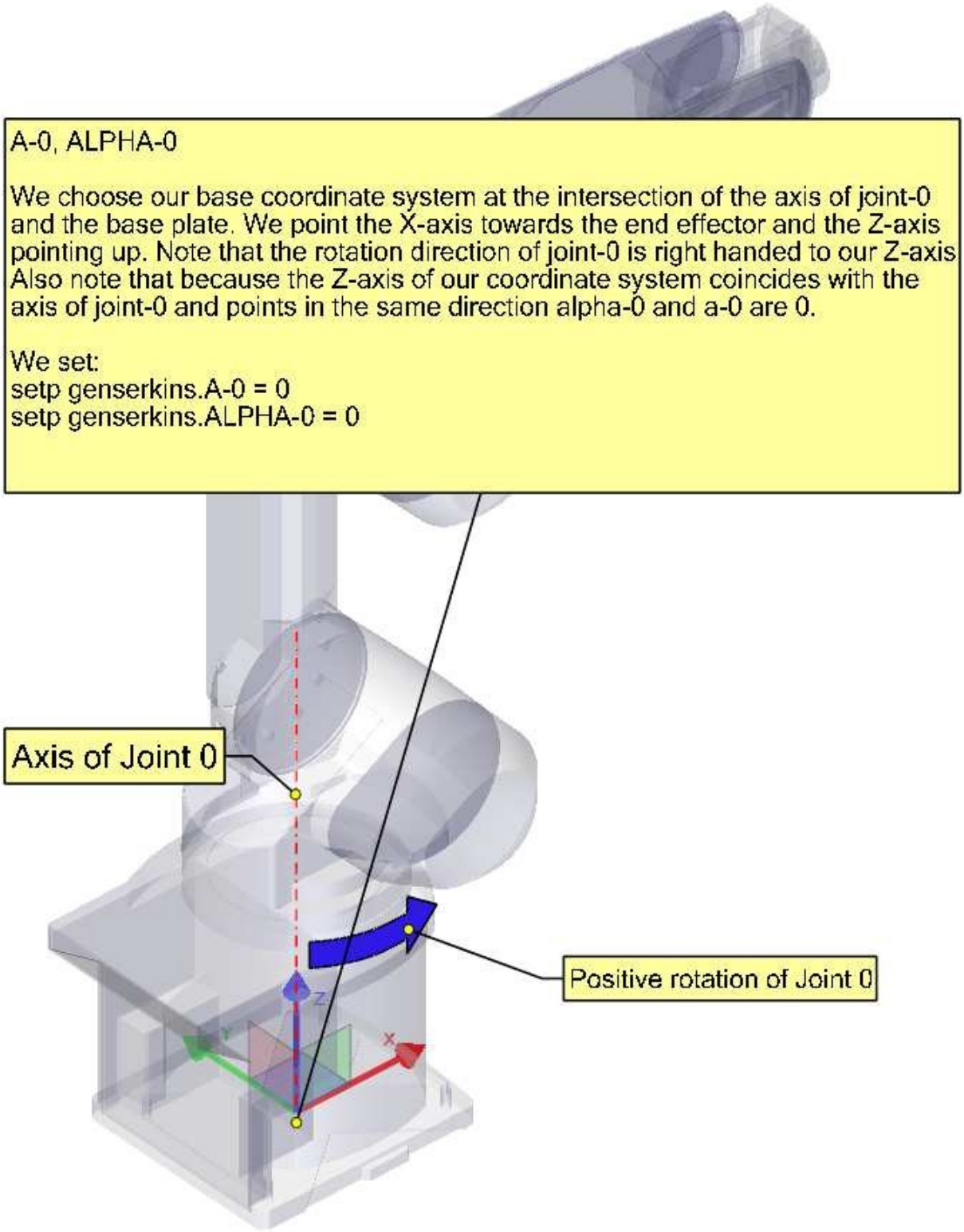
Таким чином, TCP-CS кінцевого ефектора знаходиться в центрі фланця руки.

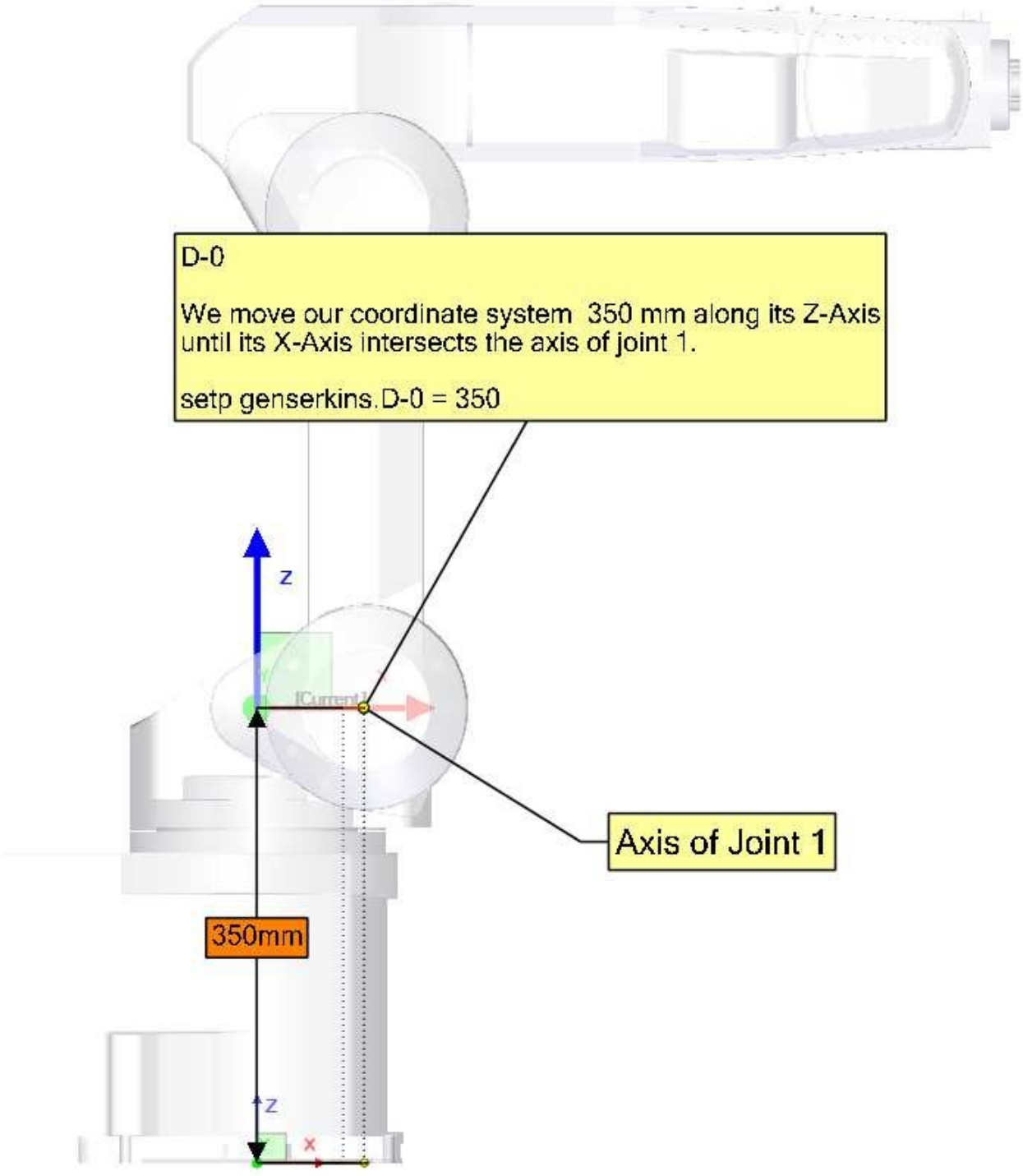
9.2.7 Особливі випадки

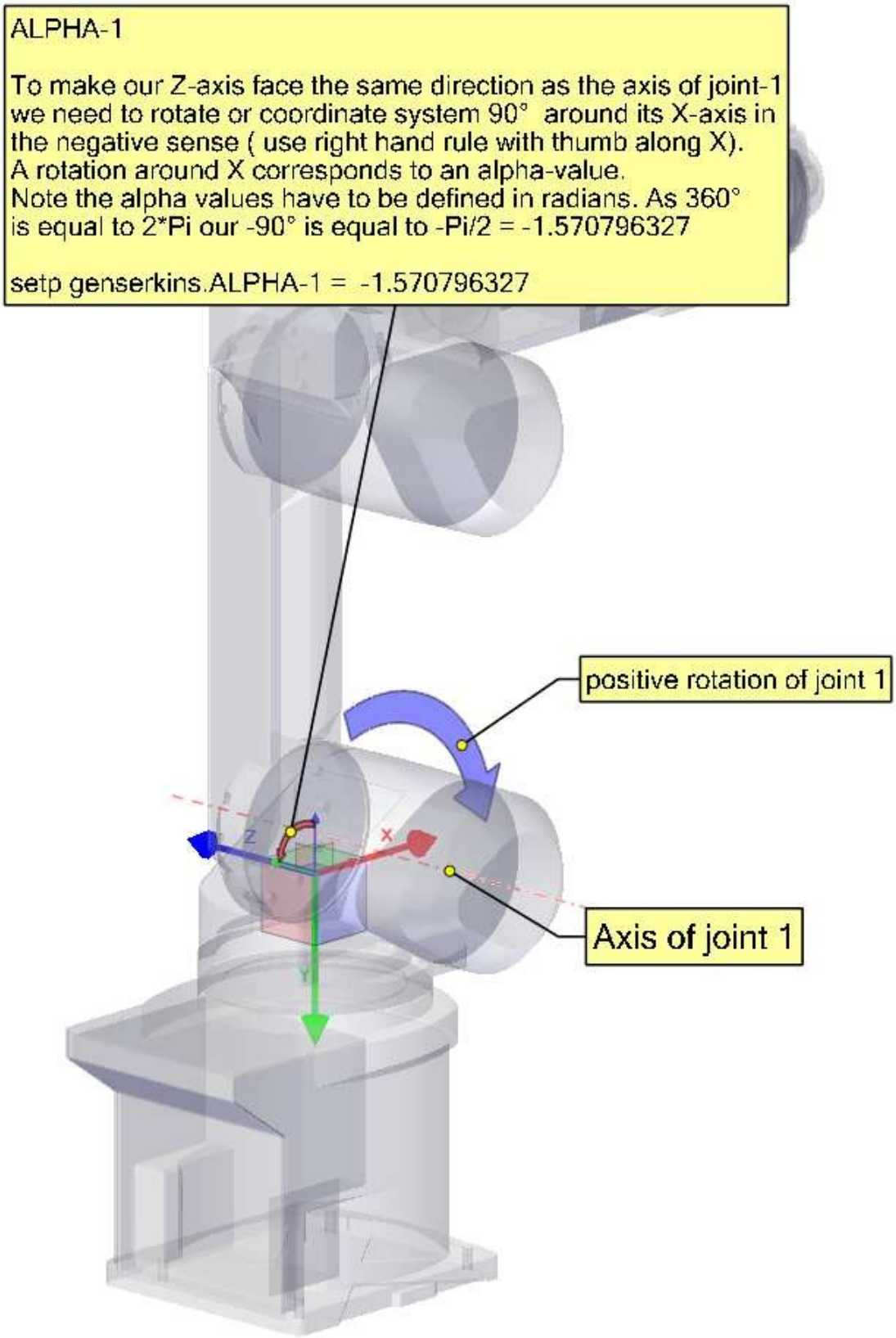
Якщо наступна вісь суглоба паралельна попередній, то можна довільно вибрати значення для d-параметра, але немає сенсу встановлювати його, відмінне від 0.

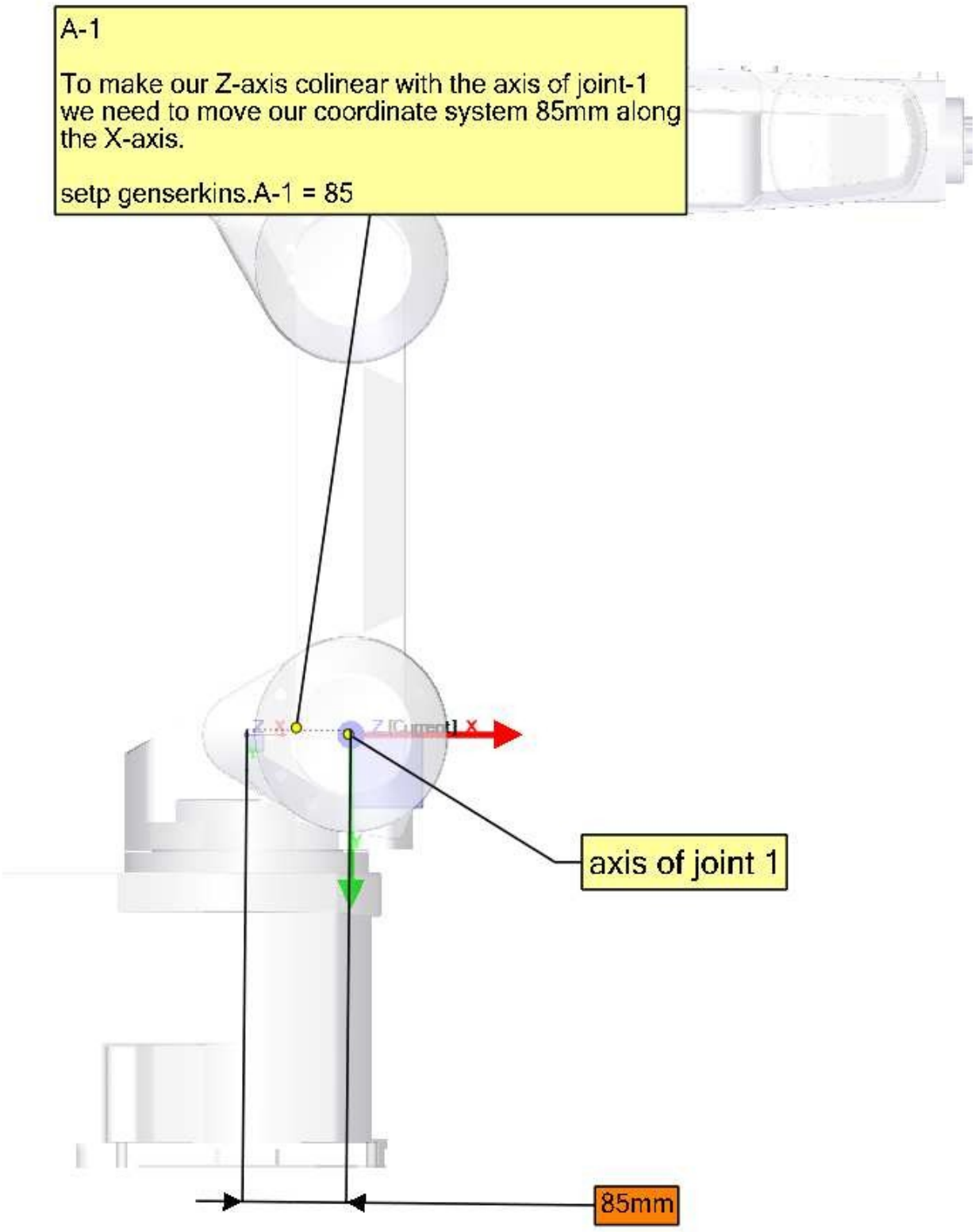
9.2.8 Детальний приклад (RV-6SL)

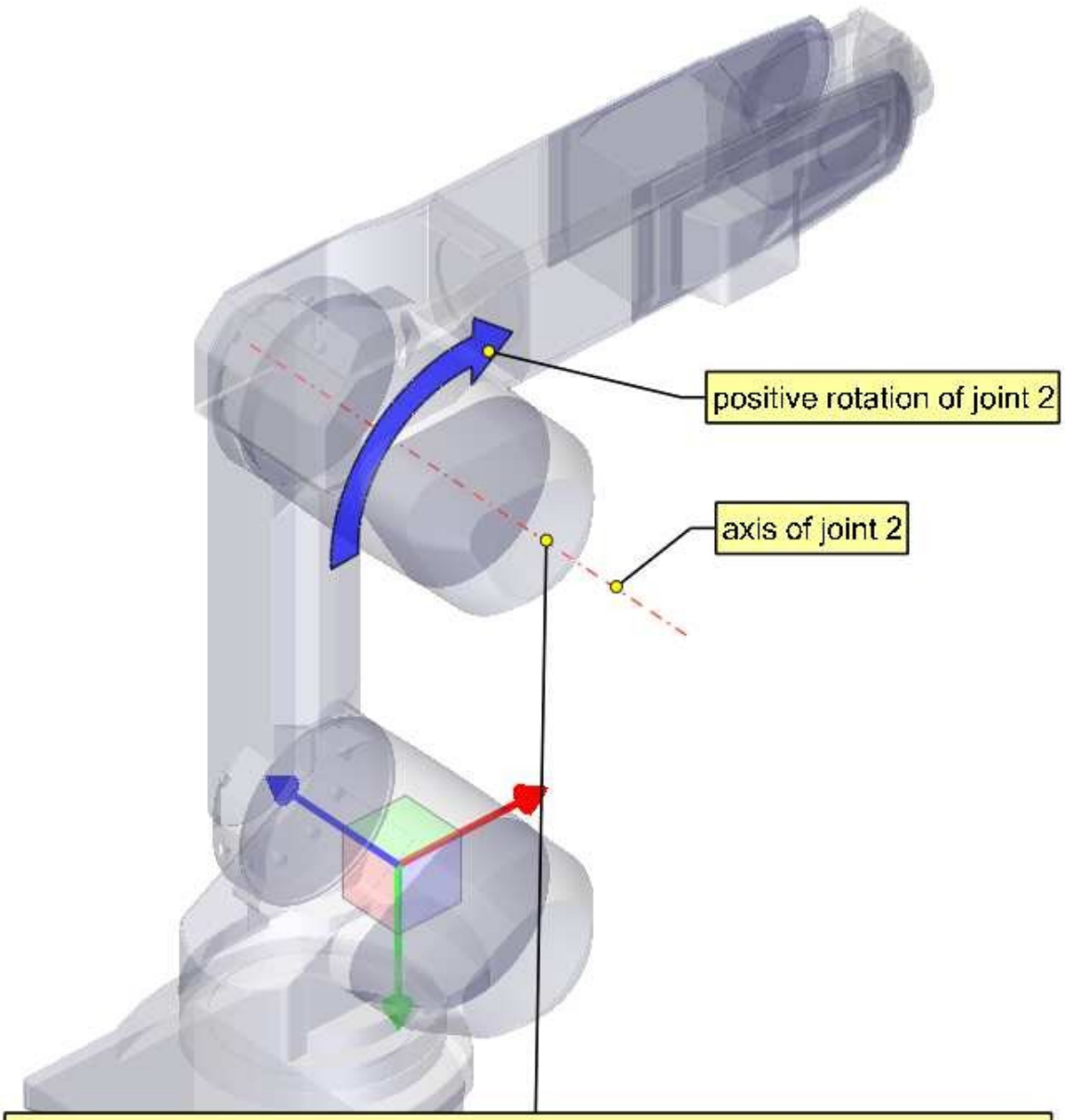
Нижче описано метод отримання необхідних «модифікованих параметрів DH» для Mitsubishi RV-6SDL та спосіб налаштування параметрів у файлі HAL для використання з кінематикою «genserkins» у LinuxCNC. Необхідні розміри найкраще брати з креслення, наданого виробником робота.


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

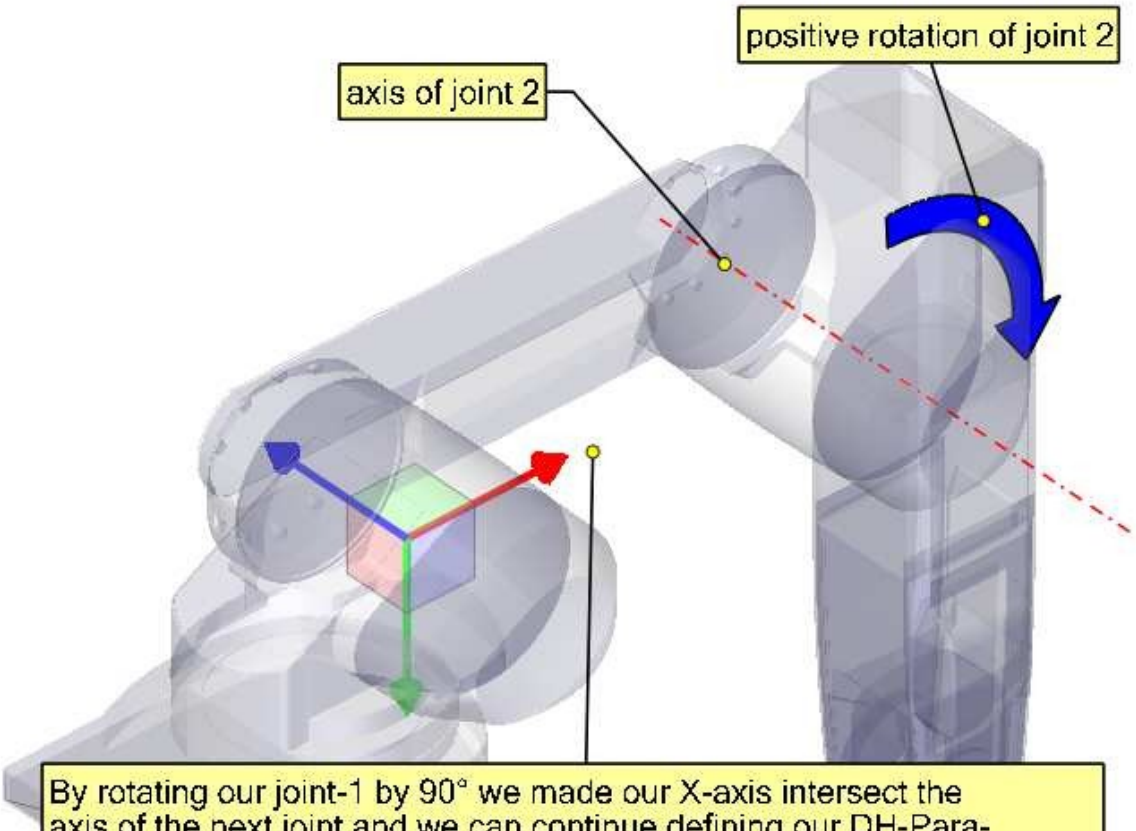

D-0
We move our coordinate system 350 mm along its Z-Axis until its X-Axis intersects the axis of joint 1.
setp genserkins.D-0 = 350
Z
Axis of Joint 1
350mm
Z
X


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

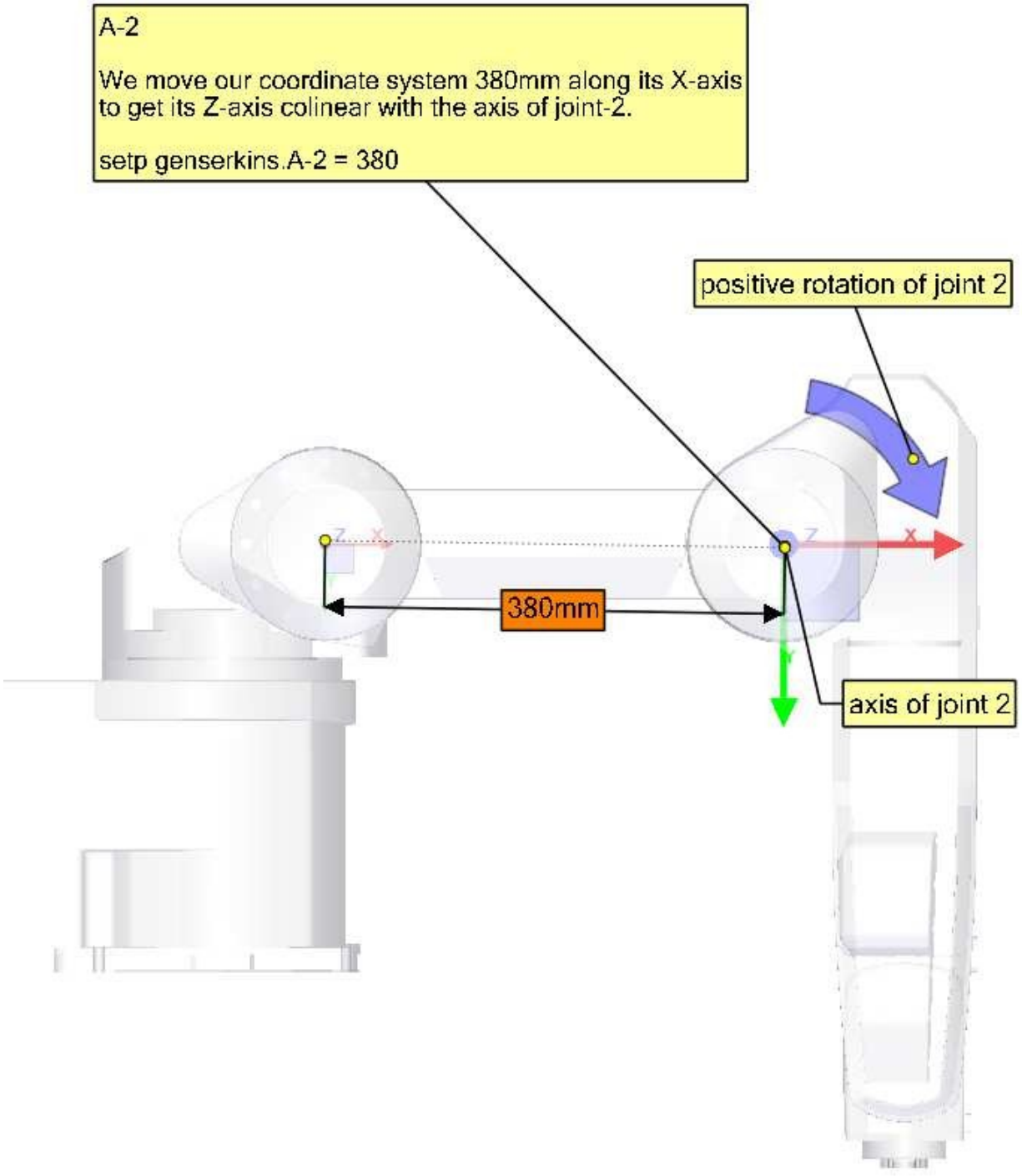
Since, after we rotated joint 1, our X-axis already intersects the axis of joint-2 we do not need to move our coordinate system along the Z-axis.

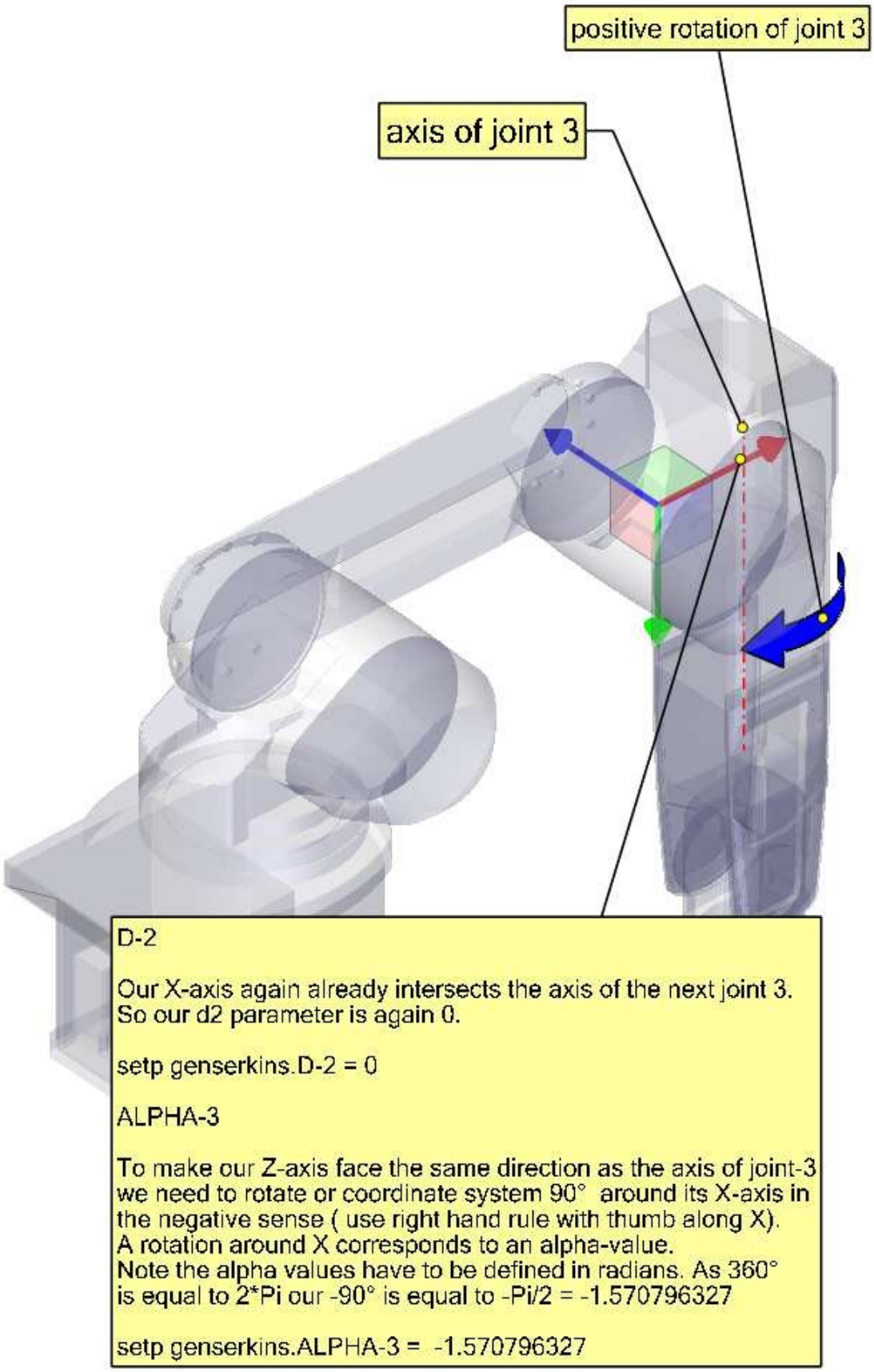
setp genserkins.D-1 = 0

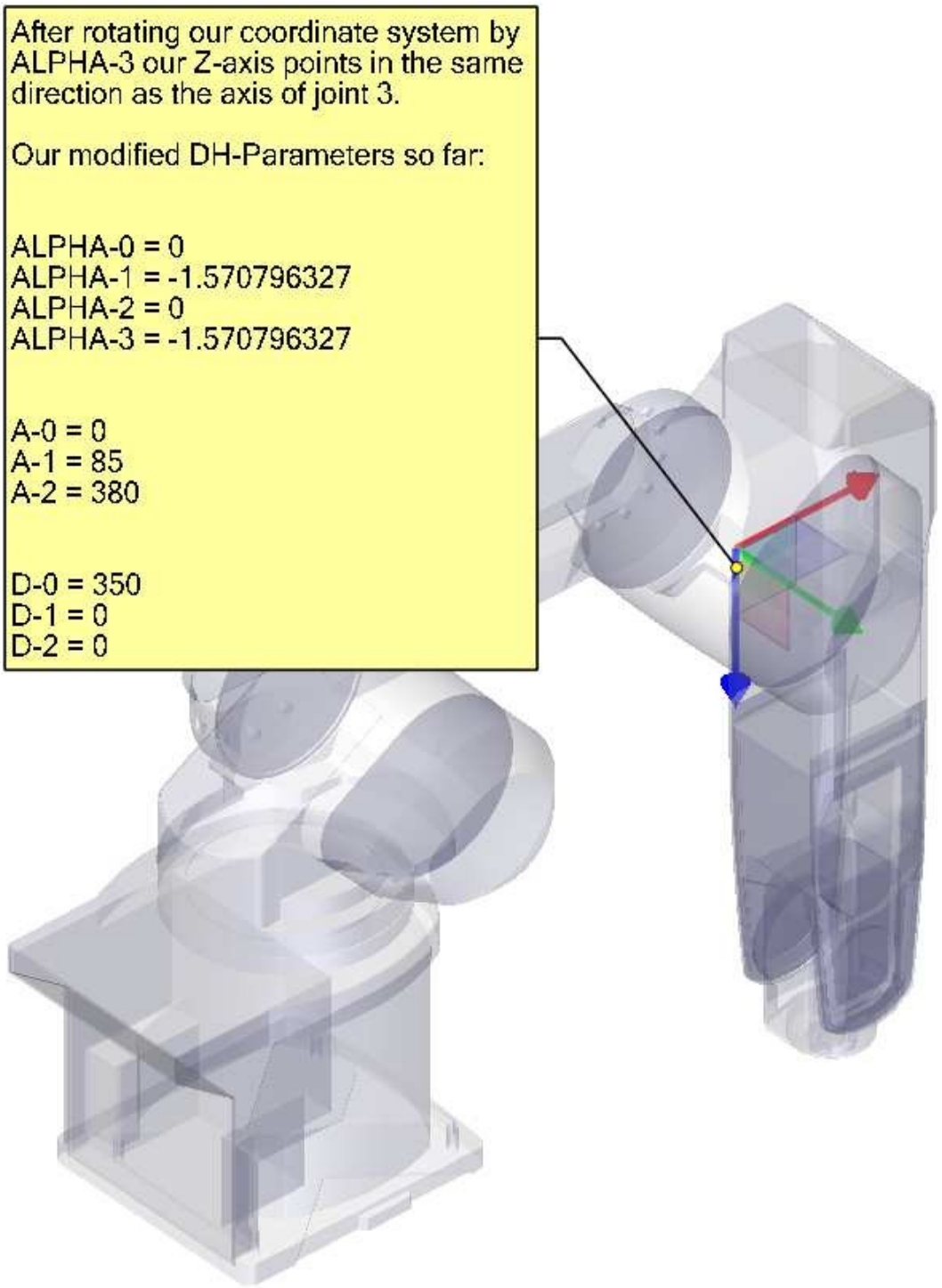
ALPHA-2

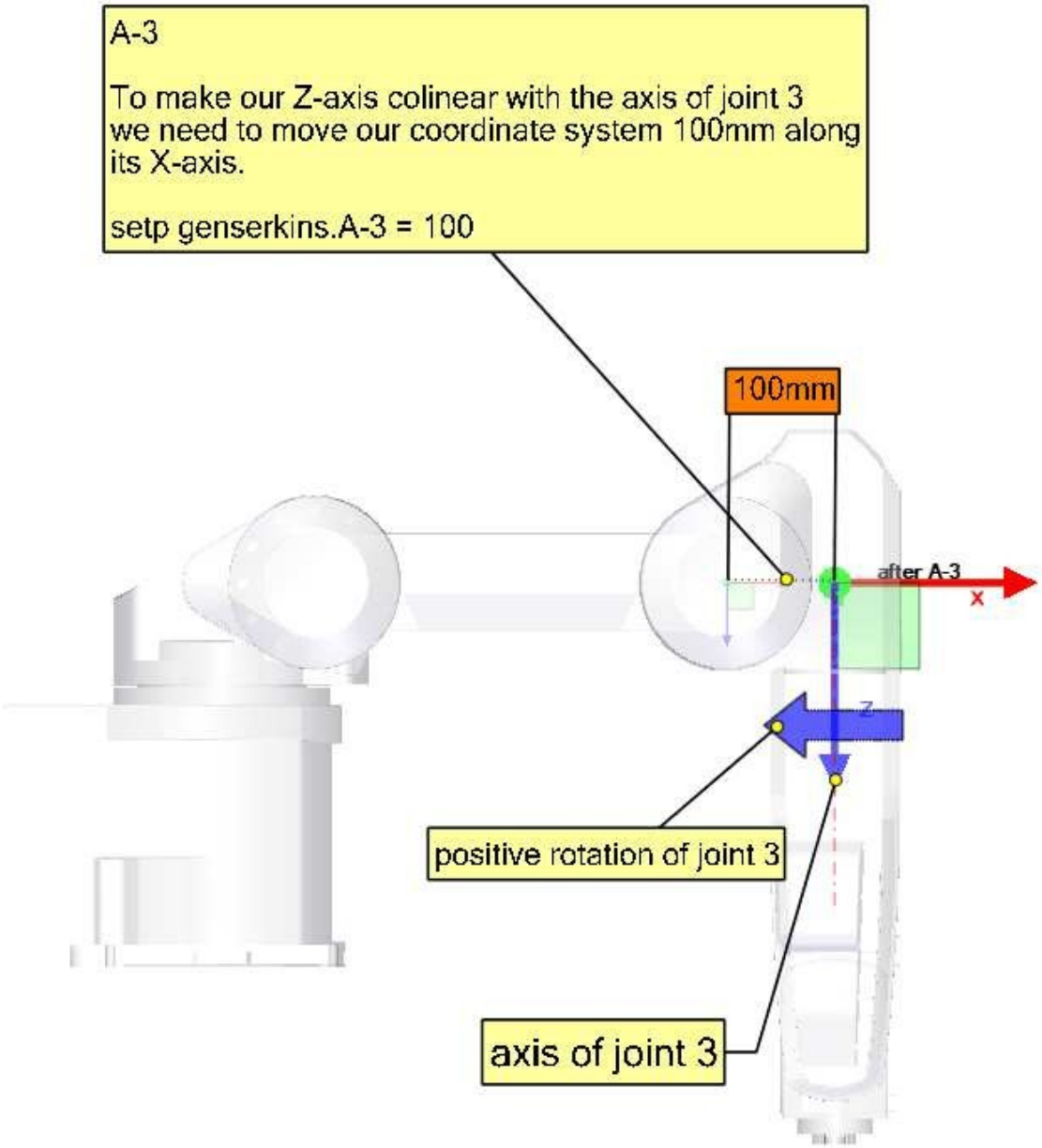
The axis of joint -2 is parallel to the axis of joint -1 and points in the same direction. Thus we do not need to rotate our Z-axis.

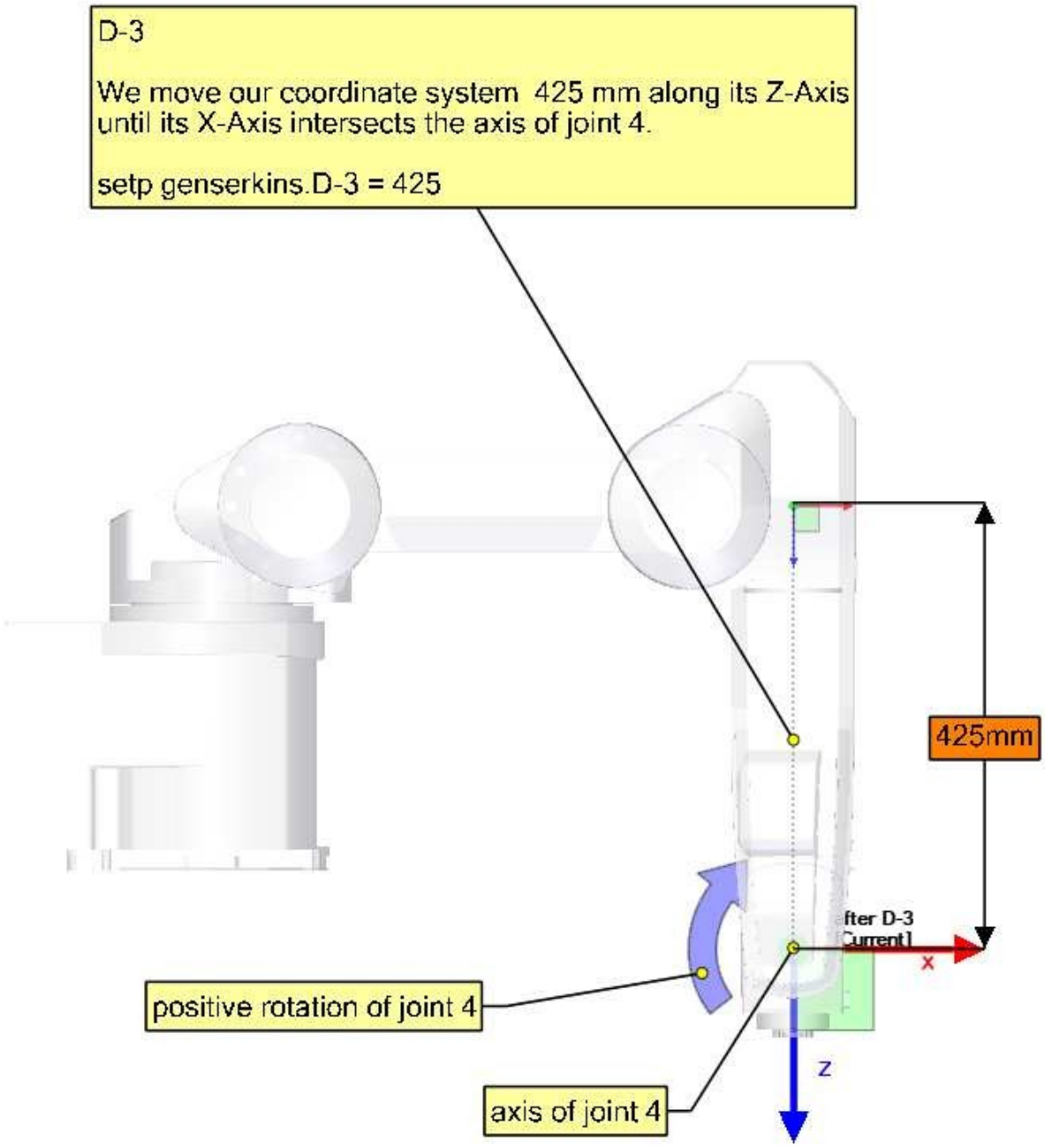
setp genserkins.ALPHA-2 = 0

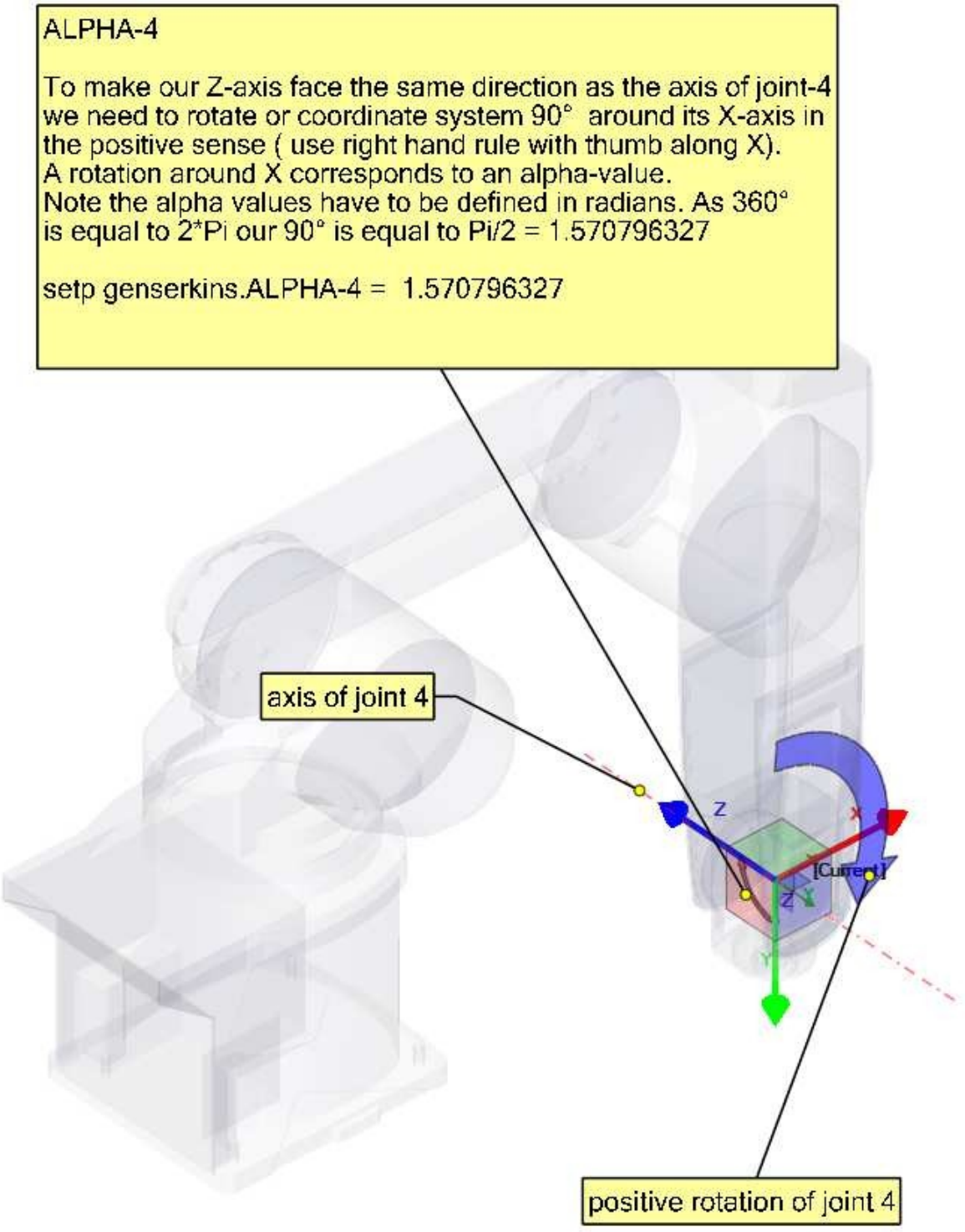

A-2
We move our coordinate system 380mm along its X-axis to get its Z-axis colinear with the axis of joint-2.
setp genserkins.A-2 = 380
positive rotation of joint 2
380mm
axis of joint 2

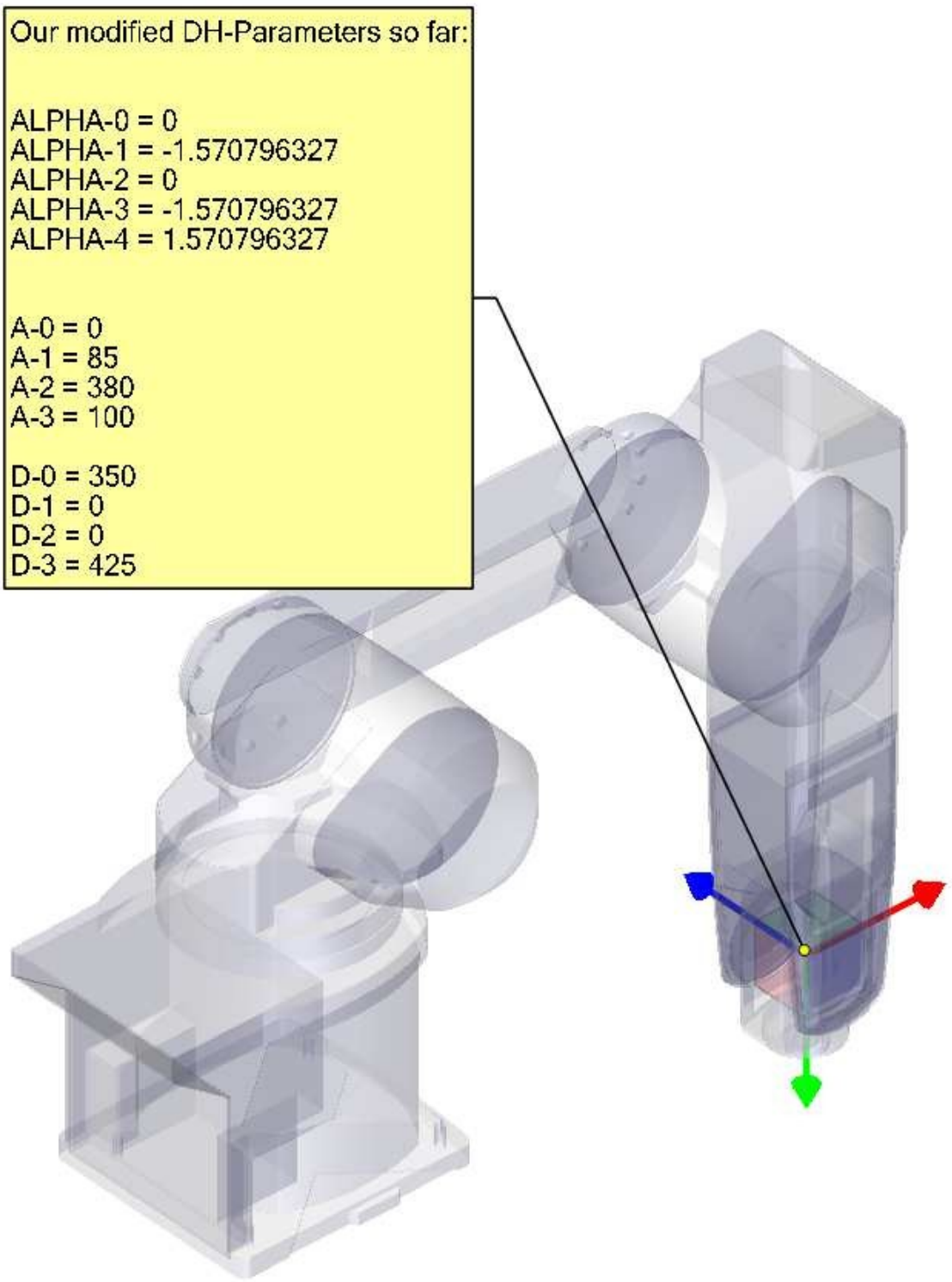

positive rotation of joint 3
axis of joint 3
D-2
Our X-axis again already intersects the axis of the next joint 3.
So our d2 parameter is again 0.
setp genserkins.D-2 = 0
ALPHA-3
To make our Z-axis face the same direction as the axis of joint-3
we need to rotate or coordinate system 90° around its X-axis 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

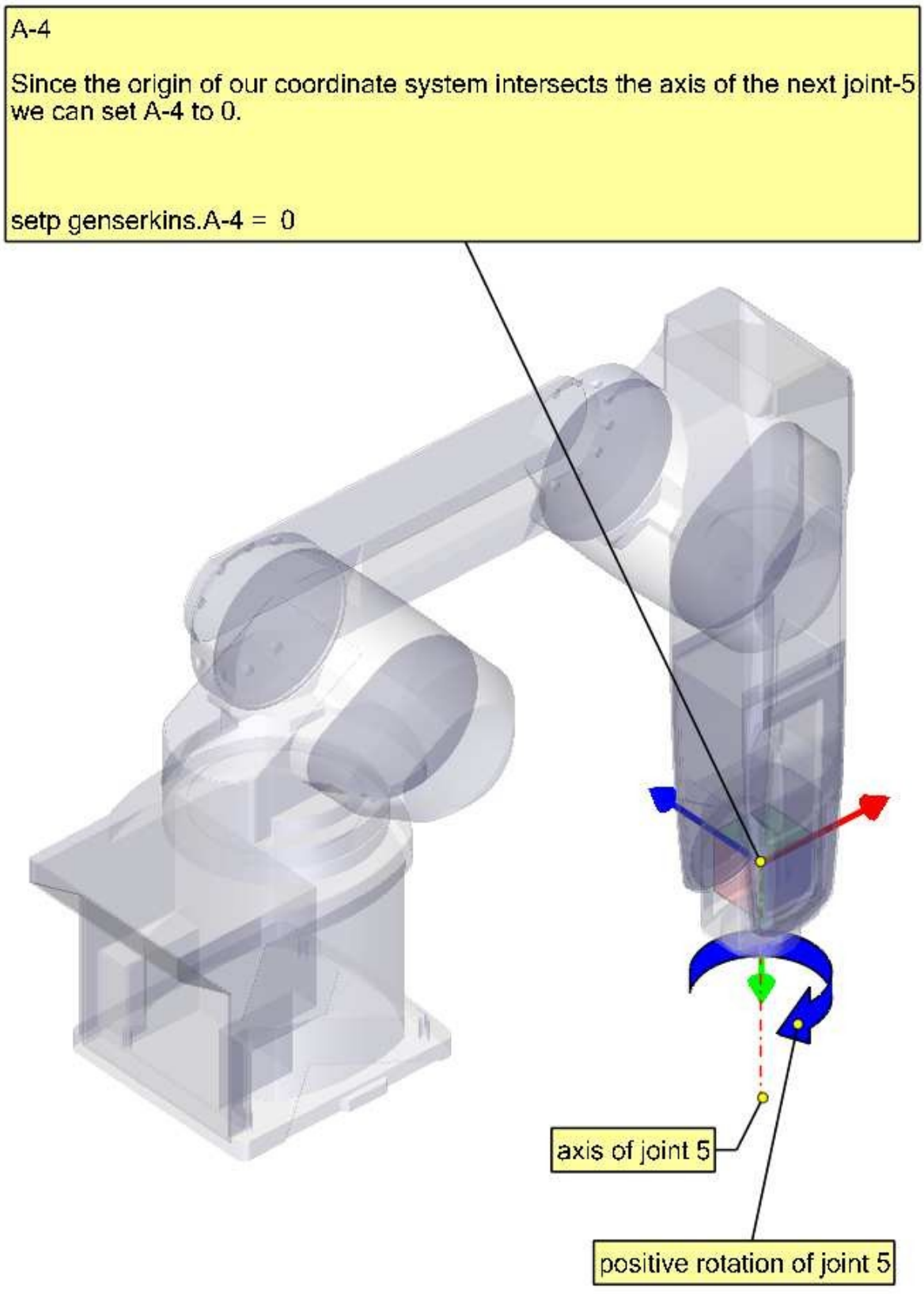

After rotating our coordinate system by ALPHA-3 our Z-axis points in the same direction as the axis of joint 3.
Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
D-0 = 350
D-1 = 0
D-2 = 0


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

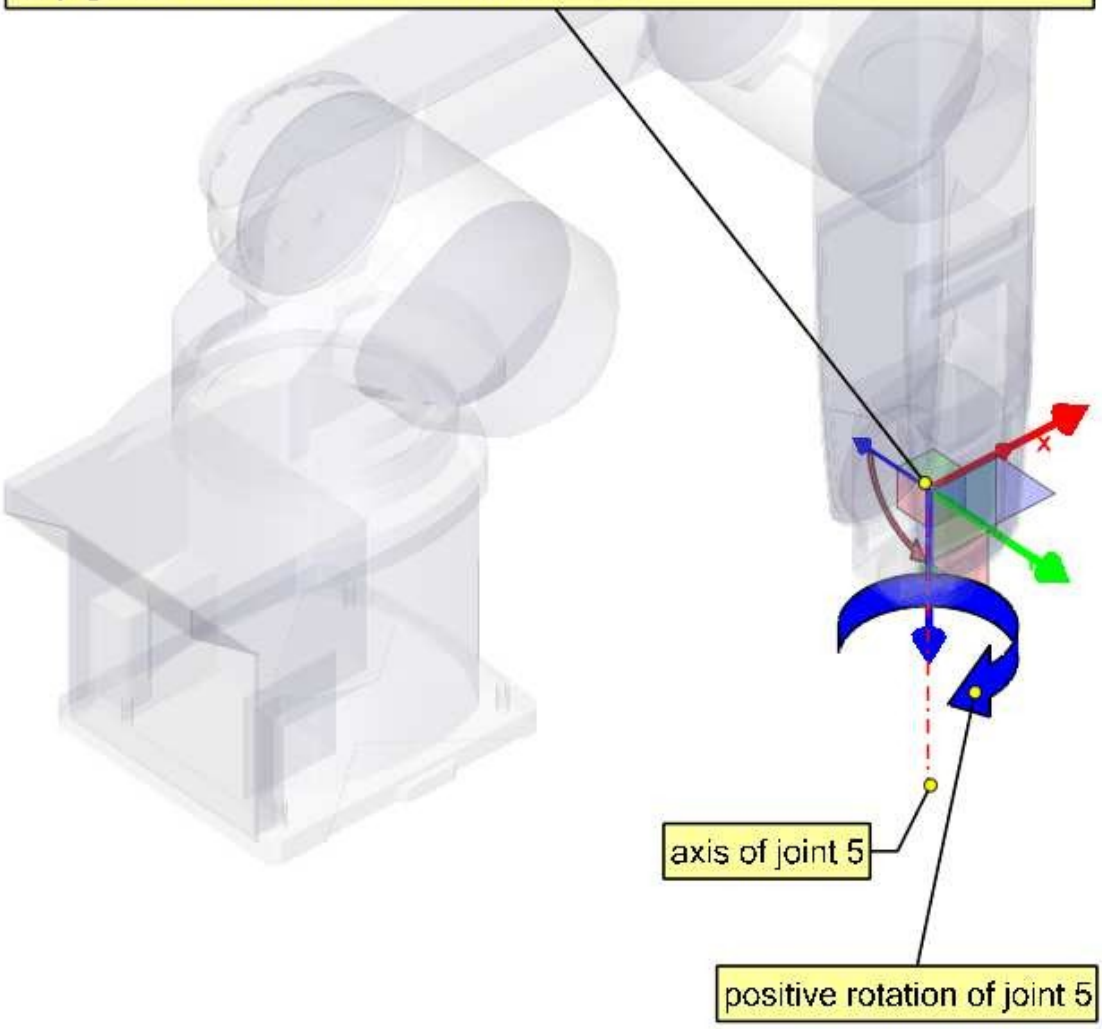
Since the origin of our coordinate system lies on the axis of the next joint our d4 parameter is also 0.

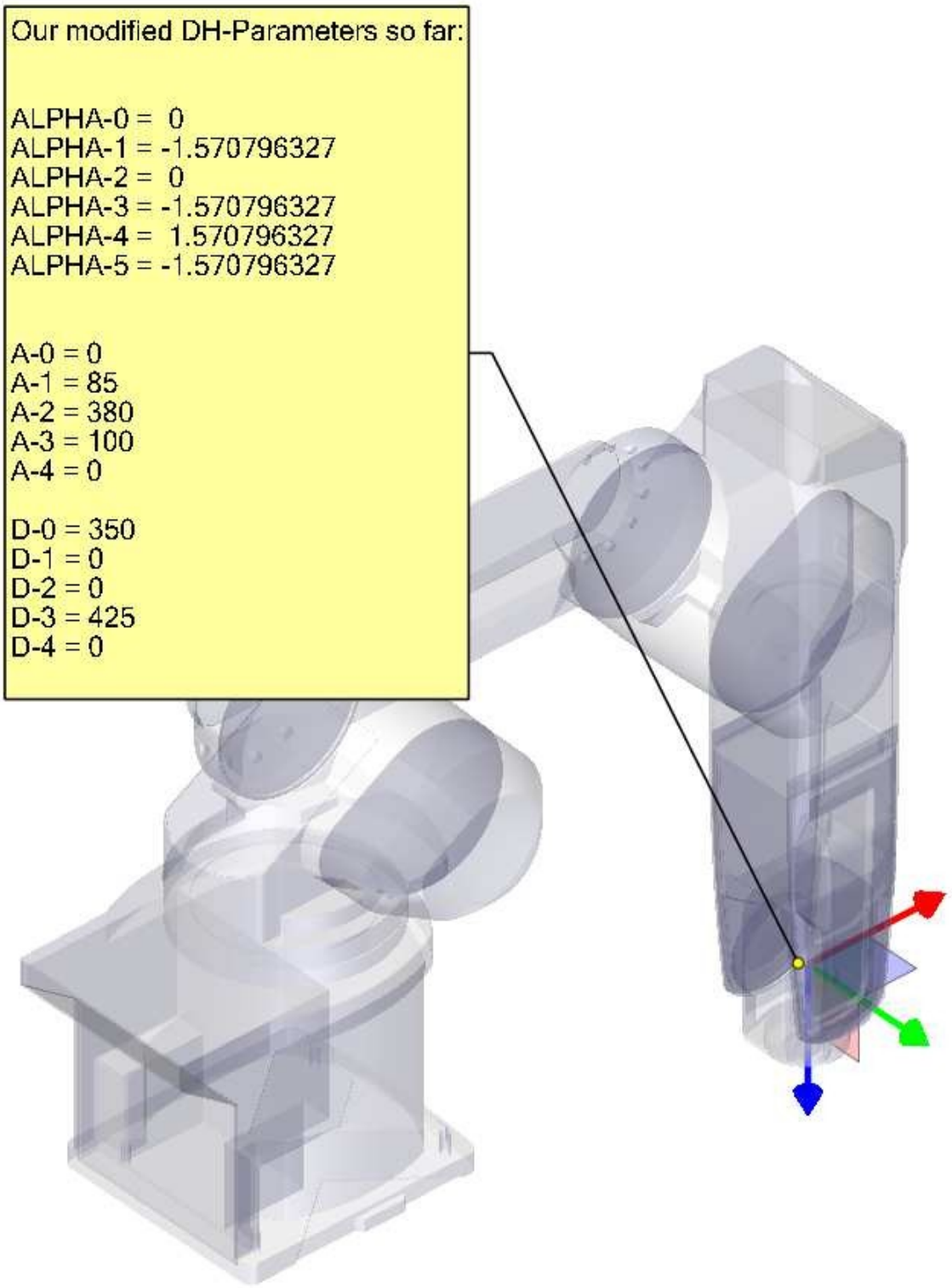
setp genserkins.D-4 = 0

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



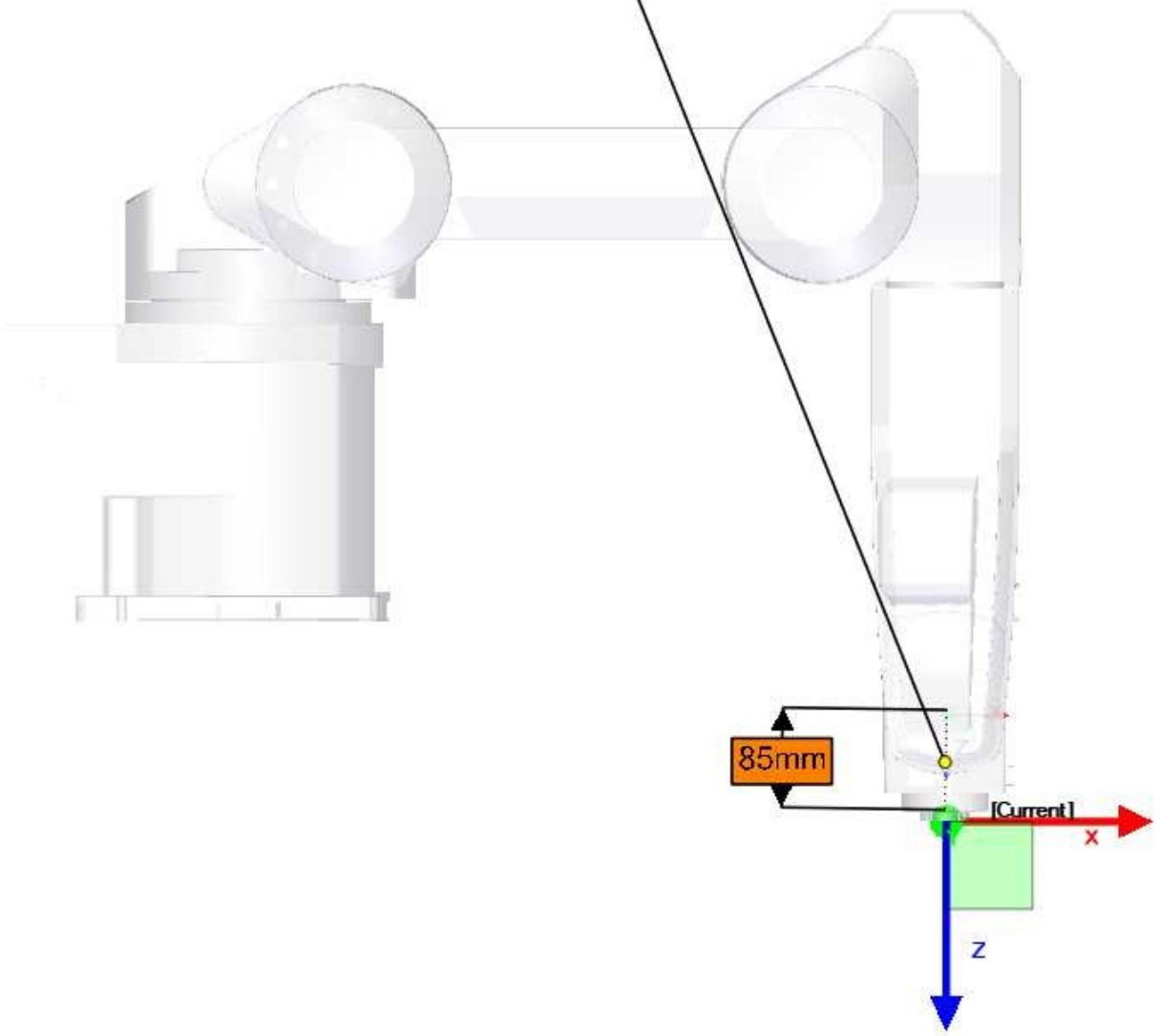

Our modified DH-Parameters so far:
ALPHA-0 = 0
ALPHA-1 = -1.570796327
ALPHA-2 = 0
ALPHA-3 = -1.570796327
ALPHA-4 = 1.570796327
ALPHA-5 = -1.570796327
A-0 = 0
A-1 = 85
A-2 = 380
A-3 = 100
A-4 = 0
D-0 = 350
D-1 = 0
D-2 = 0
D-3 = 425
D-4 = 0

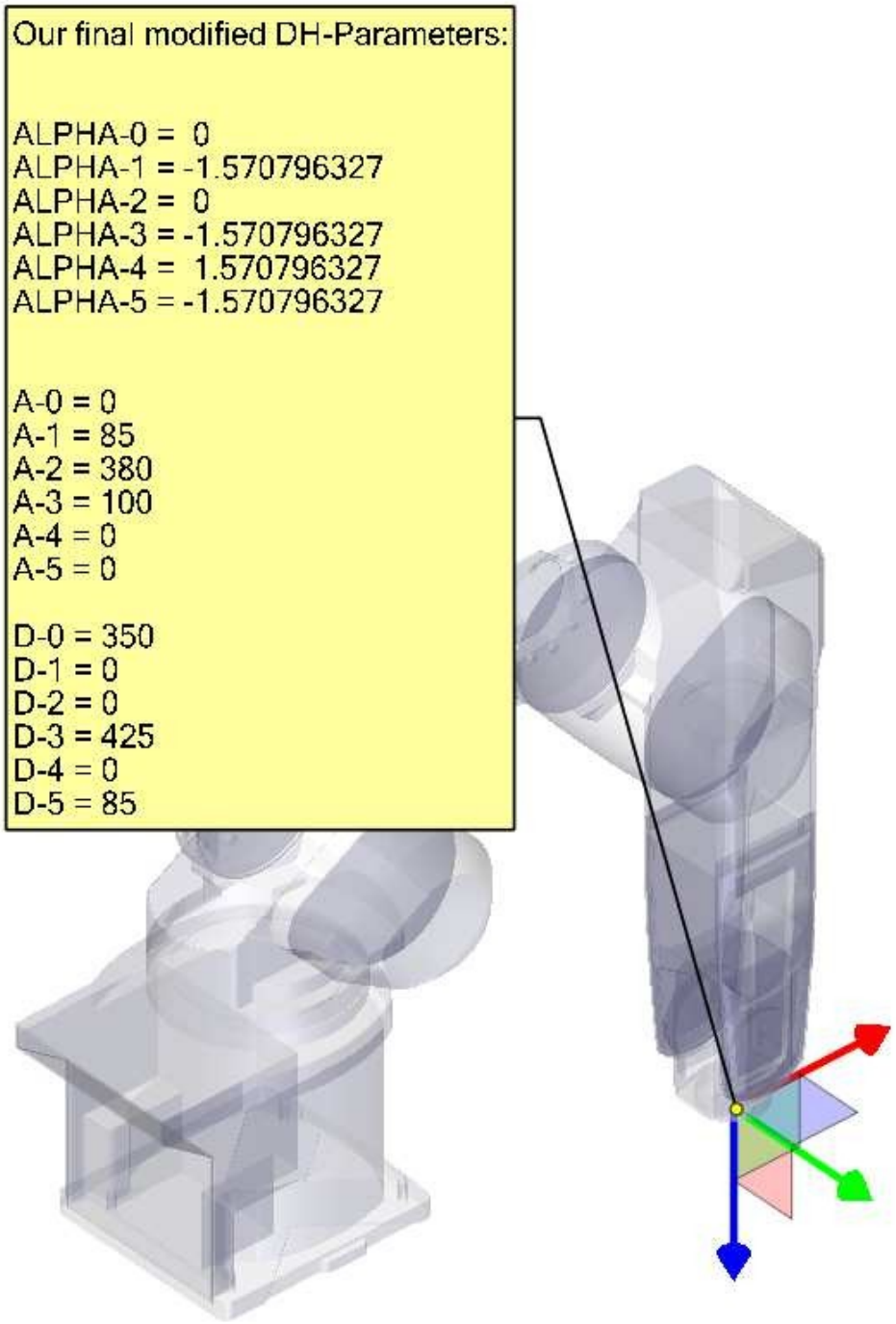
D-5

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 Кредити

Дякуємо користувачеві Aciera за весь текст та графіку для робота RV-6SL!

9.3 5-осьова кінематика

9.3.1 Вступ

Координовані багатоосьові верстати з CNC, що керуються за допомогою LinuxCNC, вимагають спеціального кінематичного компонента для кожного типу верстата. У цьому розділі описано деякі з найпопулярніших конфігурацій 5-осьових верстатів, а потім розроблено прямі (від робочих до спільних координат) та обернені (від спільних до робочих) перетворення в загальному математичном процесі для двох типів верстатів.

Наведено кінематичні компоненти, а також моделі вісмаш-симуляції для демонстрації їхньої поведінки на екрані комп'ютера. Також наведено приклади даних HAL-файлу.

Зверніть увагу, що за цієї кінематики осі обертання рухаються у протилежному напрямку відносно прийнятого. Див. розділ ["осі обертання"](https://linuxcnc.org/docs/html/gcode/machining-center.html#_ для деталей.

9.3.2 Конфігурації 5-осьових верстатів

У цьому розділі ми розглянемо типові 5-осьові фрезерні або фрезерні верстати з п'ятьма шарнірами або ступенями вільності, які керуються скоординованими рухами.

3-осьові верстати не можуть змінювати орієнтацію інструменту, тому 5-осьові верстати використовуют дві додаткові осі для встановлення ріжучого інструменту у відповідній орієнтації для ефективної обробки поверхонь довільної форми.

Типові конфігурації 5-осьових верстатів показано на рис. 3, 5, 7 та 9-11 [1,2] у розділі Рисунки.

Кінематика 5-осьових верстатів набагато простіша, ніж у 6-осьових серійних роботів з маніпулятором, оскільки 3 осі зазвичай є лінійними осями, а лише дві – осями обертання.

9.3.3 Орієнтація та розташування інструменту

Системи CAD/CAM зазвичай використовуються для створення 3D-моделей заготовки в CAD, а також даних CAM для введення в 5-осьовий верстат з CNC. Дані про розташування інструменту або різака (CL) складаються з положення кінчика різака та орієнтації різака відносно системи координат заготовки. Ця інформація міститься у двох векторах, які генеруються більшістю систем CAM і показані на рис. 1:

$$K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} \text{ orientation vector;} \qquad Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \text{ position vector} \qquad (1)$$

Вектор K еквівалентний 3-му вектору з матриці пози E_6, яка використовувалася в 6-осьовій кінематиці робота [3], а вектор Q еквівалентний 4-му вектору E_6. Наприклад, у MASTERCAM ця інформація міститься в проміжному вихідному файлі ".nci".

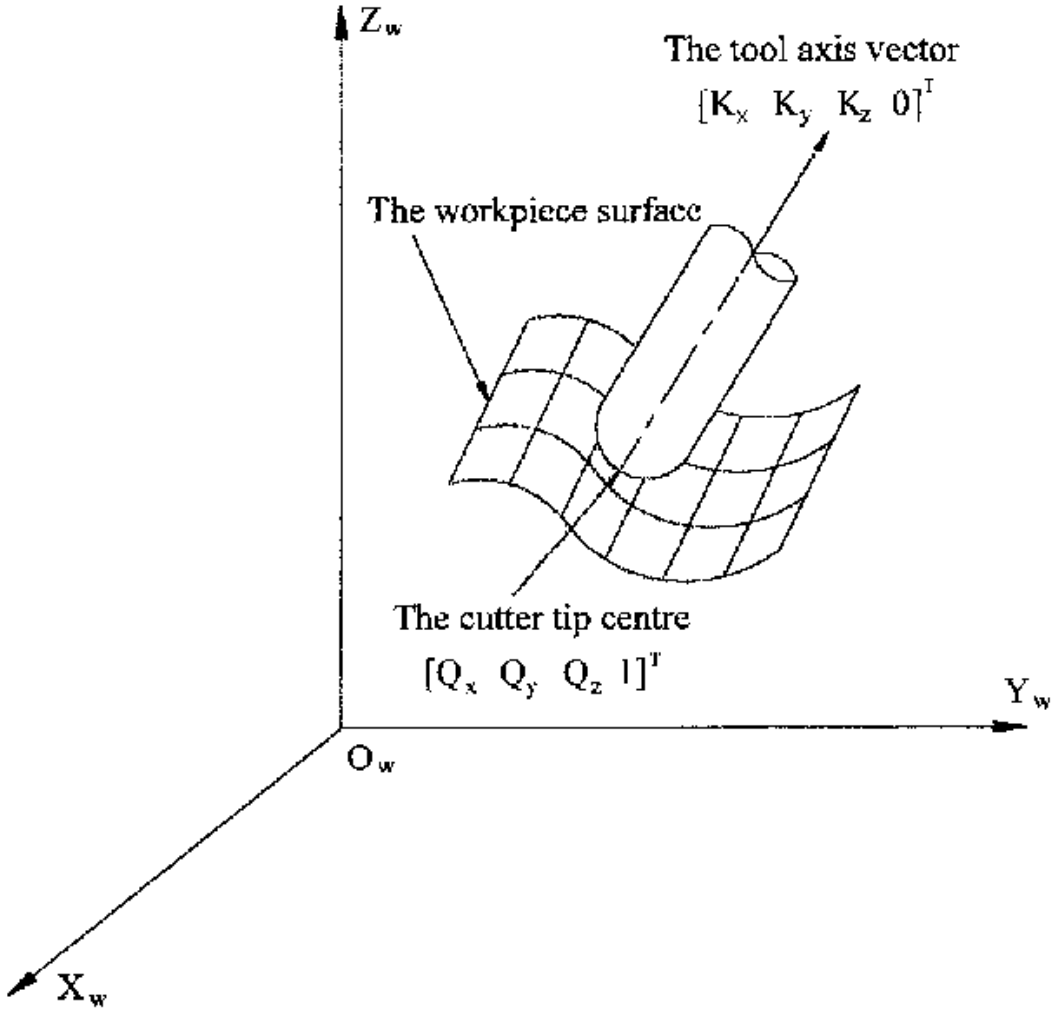


Figure 9.2: Дані про місцезнаходження різака

9.3.4 Матриці переміщення та обертання

Однорідні перетворення забезпечують простий спосіб опису математики багатоосьової кінематики верстатів. Перетворення простору H є матрицею 4x4 і може представляти перетворення зсуву та обертання. Якщо задано точку x,y,x, описану вектором u = $\{x,y,z,1\}^T$, то її перетворення v представлено матричним добутком

$$v = H \cdot u$$

Існує чотири фундаментальні матриці перетворення, на яких може базуватися 5-осьова кінематика:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

$$R(Y,\theta) = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R(Z,\theta) = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

Матриця T(a,b,c) означає зсув у напрямках координат X, Y, Z на величини a, b, c відповідно. Матриці R означають обертання на кут тета навколо осей координат X, Y і Z відповідно. Символи «C» і «S» позначають відповідно косинусну і синусну функції.

9.3.5 Поворотні/нахилені 5-осьові конфігурації столу

У цих верстатах дві осі обертання кріпляться на робочому столі верстата. Зазвичай використовуються дві форми:

- Поворотний стіл, що обертається навколо вертикальної осі Z (поворот C, вторинний), встановлений на нахильному столі, що обертається навколо осі X або Y (поворот A або B, первинний). Заготовка встановлюється на поворотний стіл.
- Нахильний стіл, що обертається навколо осі X або Y (обертання A або B, вторинне), встановлений на поворотному столі, що обертається навколо осі Z (обертання C, первинне), з заготовкою на нахильному столі.

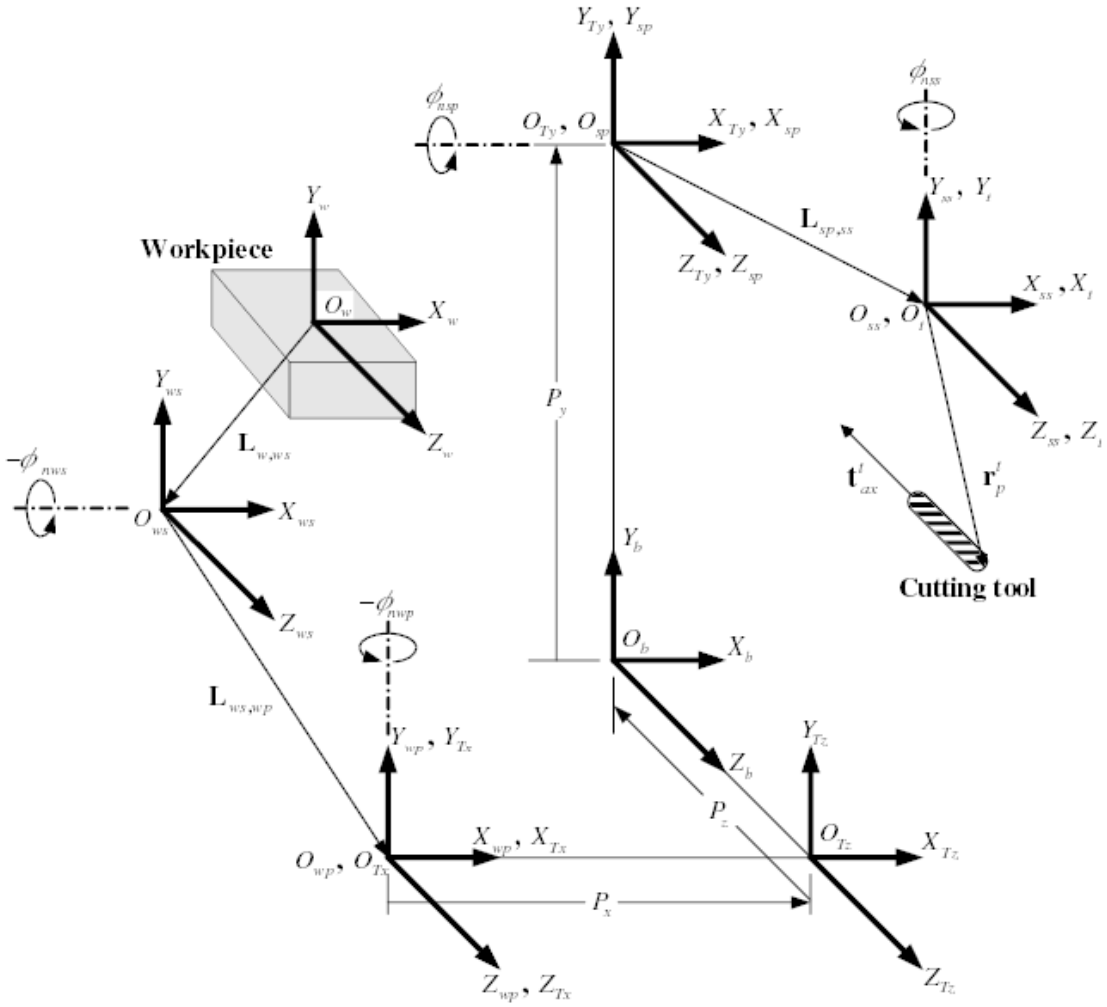


Figure 9.3: Загальна конфігурація та системи координат

Багатоосьову машину можна розглядати як сукупність ланок, з'єднаних між собою шарнірами. Вбудувавши систему координат у кожну ланку машини та використовуючи однорідні перетворення, ми можемо описати відносне положення та орієнтацію між цими системами координат

Нам потрібно описати взаємозв'язок між системою координат заготовки та системою координат інструменту. Це можна визначити за допомогою матриці перетворення «${}^{w}A_{t}$», яку можна знайти за допомогою послідовних перетворень між різними структурними елементами або ланками верстата, кожна з яких має свою власну систему координат. Загалом таке перетворення може виглядати наступним чином:

$$ {}^{w}A_{t} = {}^{w}A_{1} \cdot {}^{1}A_{2} \cdot {}^{2}A_{3} \cdots {}^{n}A_{t} \qquad (4) $$

де кожна матриця ${}^{i-1}A_{j}$ є матрицею зсуву T або матрицею обертання R виду (2,3).

Множення матриць — це простий процес, при якому елементи кожного рядка лівої матриці A множаться на елементи кожного стовпця правої матриці B і підсумовуються для отримання елемента в матриці результатів C, тобто.

$$ C_{ij} = \sum_{k=1,n}^{n} A_{ik} B_{kj}; \quad i = 1, n; \quad j = 1, n $$

На рис. 2 показано загальну конфігурацію з системами координат [4]. Вона включає осі обертання/нах столу, а також осі обертання/нахилу шпинделя. У верстаті фактично використовуються тільки дві осі обертання.

Спочатку ми розробимо перетворення для першого типу конфігурації, згаданого вище, тобто типу нахилу/обертання столу (trt) без зміщення осі обертання. Ми можемо назвати її конфігурацією xyzac-trt.

Ми також розробляємо перетворення для того ж типу (xyzac-trt), але зі зміщеннями осей обертання.

Потім ми розробляємо перетворення для конфігурації xyzbc-trt зі зміщеннями осей обертання.

9.3.5.1 Трансформації для верстата xyzac-trt зі зміщеннями робочої точки

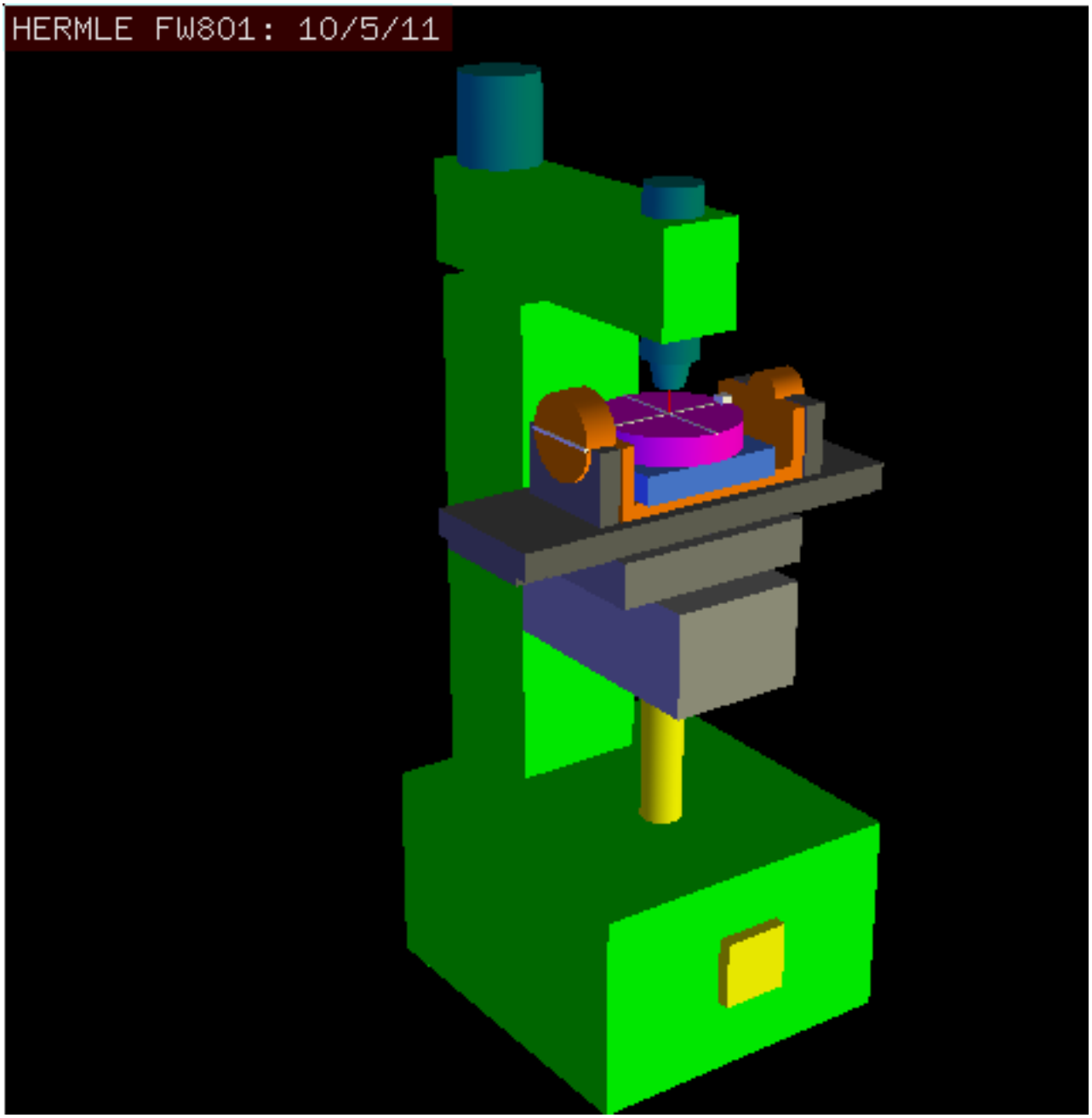


Figure 9.4: Вісмаш-модель xyzac-trt зі збігаючими осями обертання

Тут ми маємо справу зі спрощеною конфігурацією, в якій вісь нахилу і вісь обертання перетинаються в точці, яка називається точкою обертання, як показано на рис. 4. Тому дві системи координат «O_{ws}» і «O_{wp}» на рис. 2 збігаються.

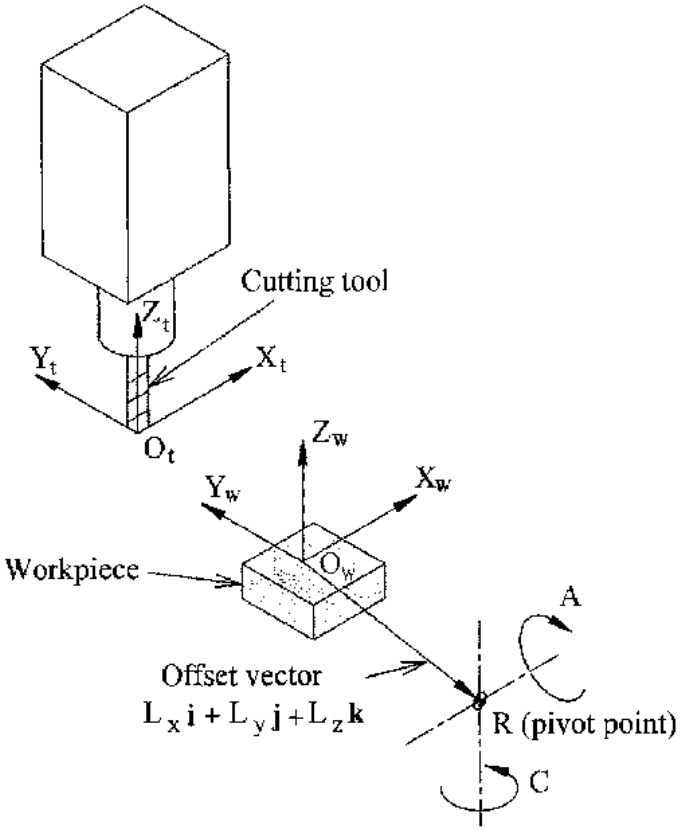


Figure 9.5: Конфігурація нахилу/повороту столу

Перетворення можна визначити послідовним множенням матриць:

$$ {}^{w}A_t = {}^{w}A_C \cdot {}^{C}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (5) $$

з матрицями, побудованими наступним чином:

$$ {}^{w}A_C = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{C}A_A = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (7) $$

У цих рівняннях L_x, L_y, L_z визначають зміщення точки обертання двох обертових осей A і C відносно початку координат заготовки. Крім того, P_x, P_y, P_z є відносними відстанями точки

обертання до положення кінчика різака, які також можна назвати «координатами з'єднання» точки обертання. Точка обертання знаходиться на перетині двох осей обертання. Знаки членів S_A і S_C відрізняються від знаків у [2,3], оскільки там обертання столу є від'ємними відносно осей координат заготовки (зауважте, що sin(-theta) = -sin(theta), cos(-theta) = cos(theta)).

При множенні відповідно до (5) отримуємо:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ 0 & -S_A & C_A & -S_A P_y + C_A P_z + L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8) $$

Тепер ми можемо прирівняти третій стовпець цієї матриці до нашого заданого вектора орієнтації інструменту K, тобто:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (9) $$

З цих рівнянь ми можемо знайти кути повороту $\theta^{-9}A^{-}$, $\theta^{-}C^{-}$. З третього рядка знаходимо:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (10) $$

і, поділивши перший рядок на другий рядок, знаходимо:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (11) $$

Ці співвідношення зазвичай використовуються в постпроцесорі CAM для перетворення векторів орієнтації інструменту в кути повороту.

Прирівнюючи останній стовпець (8) до вектора положення інструменту Q, ми можемо записати:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ -S_A P_y + C_A P_z + L_z \\ 1 \end{bmatrix} \qquad (12) $$

Вектор у правій частині також можна записати як добуток матриці та вектора, що дає:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & L_x \\ -S_C & C_C C_A & C_C S_A & L_y \\ 0 & -S_A & C_A & L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (13) $$

Це можна розширити, щоб надати

$$ \begin{aligned} Q_x &= C_c P_x + S_C C_A P_y + S_C S_A P_z + L_x \\ Q_y &= -S_C P_x + C_C C_A P_y + C_C S_A P_z + L_y \\ Q_z &= -S_A P_y + C_A P_z + L_z \end{aligned} \qquad (14) $$

що є «прямим перетворенням» кінематики.

Ми можемо вирішити P з рівняння (13) як «P = $({}^{Q}A_P)^{-1}$ * Q». Зазначимо, що квадратна матриця є однорідною матрицею 4x4, що містить матрицю обертання R і вектор перенесення q, для яких обернену матрицю можна записати як:

$$ {}^{q}A_{p} = \begin{bmatrix} R & q \\ 0 & 1 \end{bmatrix} \qquad ({}^{Q}A_{P})^{-1} = \begin{bmatrix} R^{T} & -R^{T}q \\ 0 & 1 \end{bmatrix} \qquad (15) $$

де R^T — транспонування R (рядки та стовпці поміняні місцями). Таким чином, отримуємо:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & -C_C L_x + S_C L_y \\ S_C C_A & C_C C_A & -S_A & -S_C C_A L_x - C_C C_A L_y + S_A L_z \\ S_C S_A & C_C S_A & C_A & -S_C S_A L_x - C_C S_A L_y - C_A L_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (16) $$

Таким чином, бажані рівняння для «зворотного перетворення» кінематики можна записати так:

$$ \begin{aligned} P_x &= C_c(Q_x - L_x) - S_C(Q_y - L_y) \\ P_y &= S_C C_A(Q_x - L_x) + C_C C_A(Q_y - L_y) - S_A(Q_z - L_z) \\ P_z &= S_C S_A(Q_x - L_x) + C_C S_A(Q_y - L_y) + C_A(Q_z - L_z) \end{aligned} \qquad (17) $$

9.3.5.2 Перетворення для верстата xyzac-trt зі зміщеннями поворотних осей

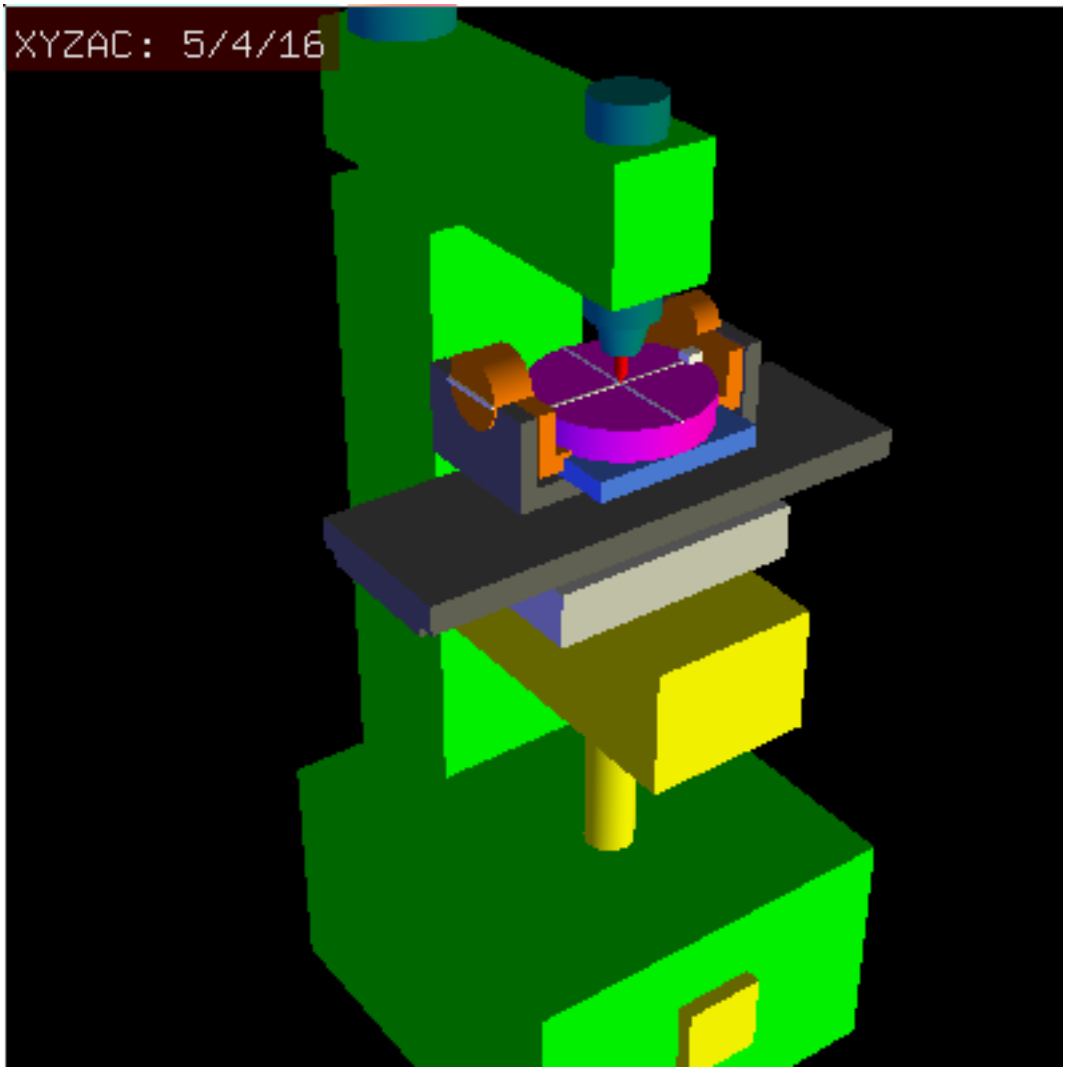


Figure 9.6: Модель vismach xyzac-trt зі зміщеннями осі обертання (додатними)

Тут ми маємо справу з розширеною конфігурацією, в якій вісь нахилу і вісь обертання не перетинаються в одній точці, а мають зміщення D_y. Крім того, між двома системами координат «O_{ws}» і «O_{wp}» на рис. 2 також існує зміщення по осі z, яке називається D_z. Модель vismach показана на рис. 5, а зміщення показані на рис. 6 (у цьому прикладі зміщення є додатними). Для спрощення конфігурації зміщення L_x, L_y, L_z з попереднього випадку не включені. Вони, ймовірно, не потрібні, якщо використовувати зміщення G54 в LinuxCNC за допомогою функції «touch of».

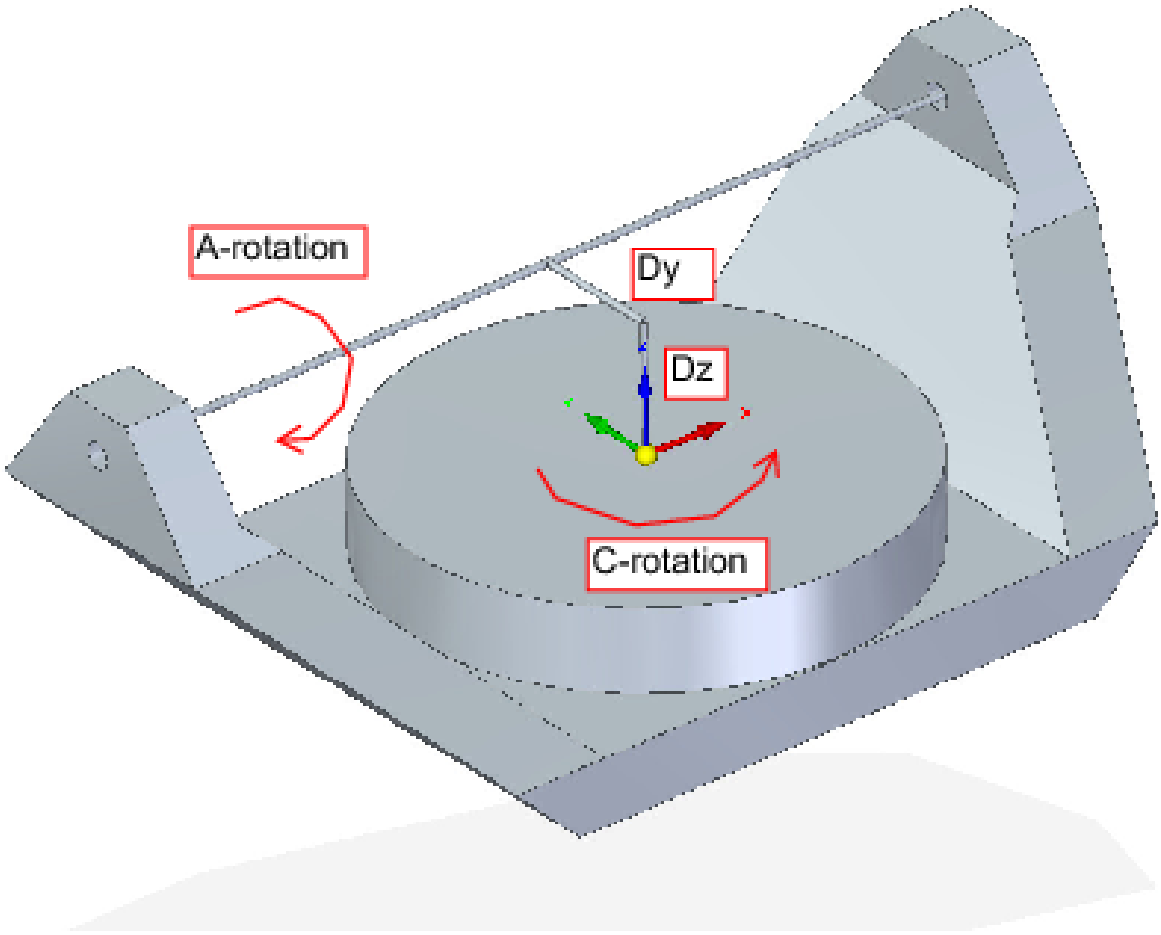


Figure 9.7: Конфігурація нахилу/повороту столу xyzac-trt зі зміщеннями осей

Перетворення можна визначити послідовним множенням матриць:

$$ {}^{w}A_t = {}^{w}A_O \cdot {}^{O}A_A \cdot {}^{A}A_P \cdot {}^{P}A_t \qquad (18) $$

з матрицями, побудованими наступним чином:

$$ {}^{w}A_O = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (19) $$

$$ {}^{A}A_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_A & S_A & 0 \\ 0 & -S_A & C_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_t = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y - D_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (20) $$

У цих рівняннях D_y, D_z визначають зміщення точки обертання обертових осей A відносно початку координат системи координат заготовки. Крім того, P_x, P_y, P_z є відносними відстанями точки обертання до положення кінчика різака, які також можна назвати «координатами з'єднання» точки обертання. Точка обертання знаходиться на обертовій осі A.

При множенні відповідно до (18) отримуємо:

$$ {}^{w}A_t = \begin{bmatrix} C_C & S_C C_A & S_C S_A & C_C P_x + S_C C_A(P_y - D_y) + S_C S_A(P_z - D_z) + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -S_C P_x + C_C C_A(P_y - D_y) + C_C S_A(P_z - D_z) + C_C D_y \\ 0 & -S_A & C_A & -S_A(P_y - D_y) + C_A(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (21) $$

Тепер ми можемо прирівняти третій стовпець цієї матриці до нашого заданого вектора орієнтації інструменту K, тобто:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} S_C S_A \\ C_C S_A \\ C_A \\ 0 \end{bmatrix} \qquad (22) $$

З цих рівнянь ми можемо знайти кути повороту $\theta^{-9}A^{-}$, $\theta^{-}C^{-}$. З третього рядка знаходимо:

$$ \theta_A = cos^{-1}(K_z) \qquad (0 < \theta_A < \pi) \qquad (23) $$

і, поділивши другий рядок на перший рядок, знаходимо:

$$ \theta_C = tan2^{-1}(K_x, K_y) \qquad (-\pi < \theta_C < \pi) \qquad (24) $$

Ці співвідношення зазвичай використовуються в постпроцесорі CAM для перетворення векторів орієнтації інструменту в кути повороту.

Прирівнюючи останній стовпець (21) до вектора положення інструменту Q, ми можемо записати:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C P_x + S_C C_A(P_y - D_y) + S_C S_A(P_z - D_z) + S_C D_y \\ -S_C P_x + C_C C_A(P_y - D_y) + C_C S_A(P_z - D_z) + C_C D_y \\ -S_A(P_y - D_y) + C_A(P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (25) $$

Вектор у правій частині також можна записати як добуток матриці та вектора, що дає:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & S_C C_A & S_C S_A & -S_c C_A D_y - S_C S_A D_z + S_C D_y \\ -S_C & C_C C_A & C_C S_A & -C_C C_A D_y - C_C S_A D_z + C_C D_y \\ 0 & -S_A & C_A & S_A D_y - C_A D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (26) $$

що є «прямим перетворенням» кінематики.

Ми можемо розв'язати для P рівняння (25) як $P = ({}^{Q}A_P)^{-1} * Q$, використовуючи (15), як і раніше. Таким чином, ми отримуємо:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C & -S_C & 0 & 0 \\ S_C C_A & C_C C_A & -S_A & -C_A D_y + S_A D_z + D_y \\ S_C S_A & C_S S_A & C_A & -S_A D_y - C_A D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (27) $$

Таким чином, бажані рівняння для «зворотного перетворення» кінематики можна записати так:

$$ \begin{aligned} P_x &= C_c Q_x - S_C Q_y \\ P_y &= S_C C_A Q_x + C_C C_A Q_y - S_A Q_z - C_A D_y + S_A D_z + D_y \\ P_z &= S_C S_A Q_x + C_S S_A Q_y + C_A Q_z - S_A D_y - C_A D_z + D_z \end{aligned} \qquad (28) $$

9.3.5.3 Перетворення для верстата xyzbc-trt зі зміщеннями поворотних осей

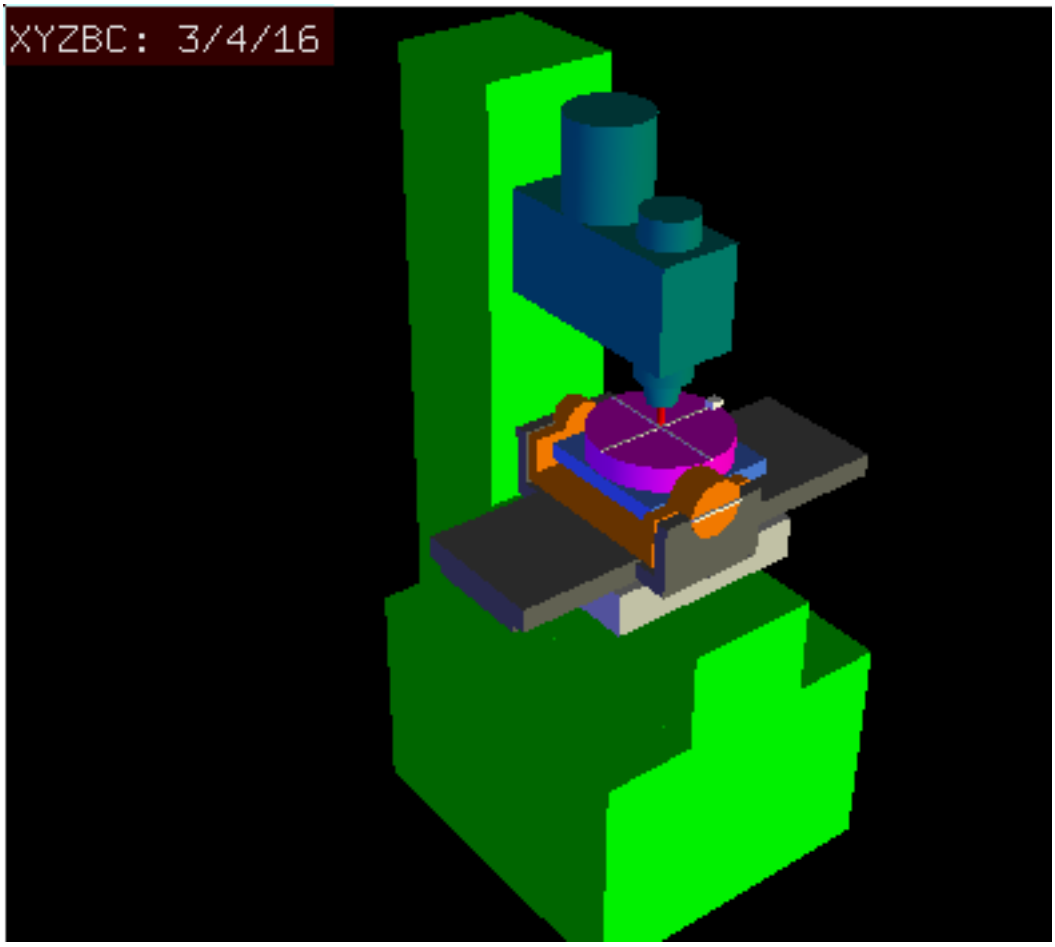


Figure 9.8: Модель vismach xyzbc-trt зі зміщеннями осі обертання (від’ємними)

Тут ми знову маємо справу з розширеною конфігурацією, в якій вісь нахилу (навколо осі y) і вісь обертання не перетинаються в одній точці, а мають зміщення D_x. Крім того, між двома системами координат «O_{ws}» і «O_{wp}» на рис. 2 також існує зсув по осі z, який називається D_z. Модель Vismach показана на рис. 7 (у цьому прикладі зсуви від’ємні), а позитивні зсуви показані на рис. 8.

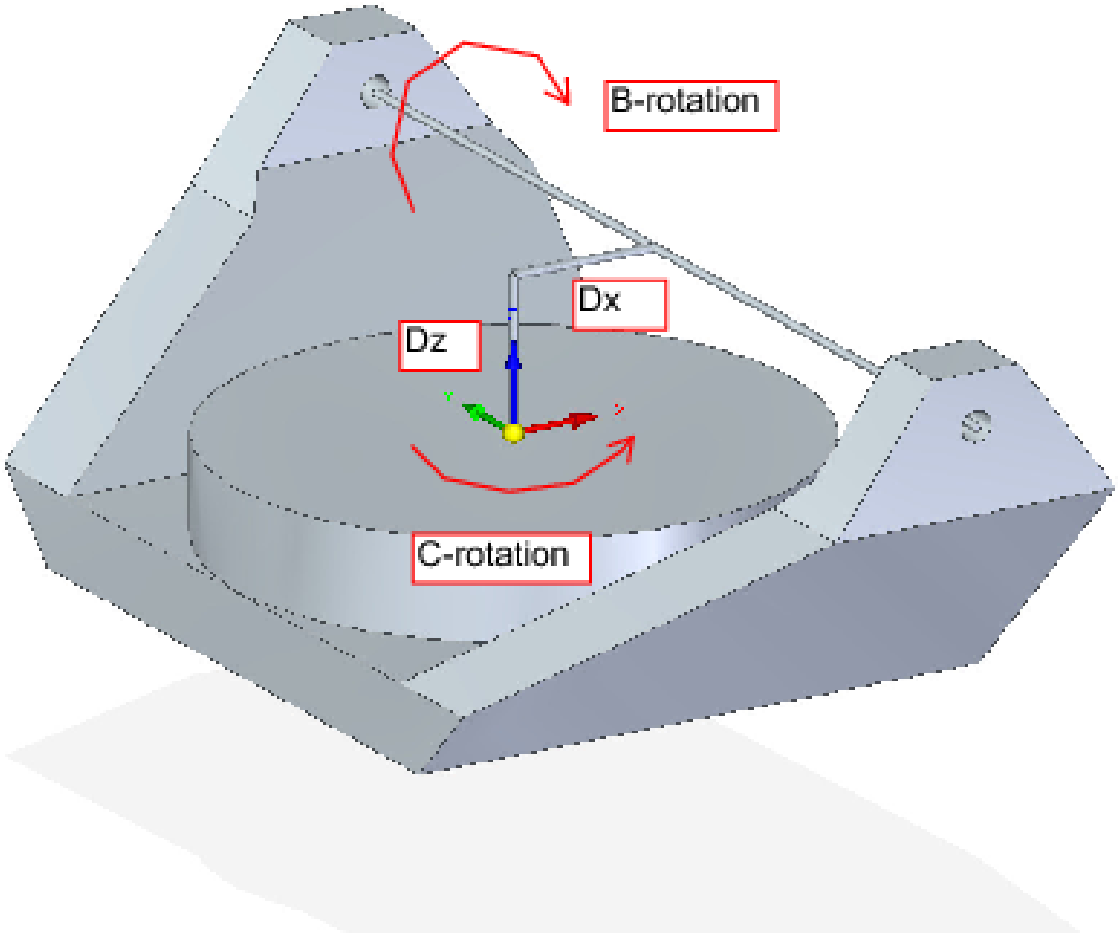


Figure 9.9: Конфігурація нахилу/повороту столу xyzbc-trt зі зміщеннями осей

Перетворення можна визначити послідовним множенням матриць:

$$ {}^{w}A_{t} = {}^{w}A_{O} \cdot {}^{O}A_{B} \cdot {}^{B}A_{P} \cdot {}^{P}A_{t} \qquad (29) $$

з матрицями, побудованими наступним чином:

$$ {}^{w}A_{O} = \begin{bmatrix} C_C & S_C & 0 & 0 \\ -S_C & C_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{O}A_{B} = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (30) $$

$$ {}^{B}A_{P} = \begin{bmatrix} C_B & 0 & -S_B & 0 \\ 0 & 1 & 0 & 0 \\ S_B & 0 & C_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{P}A_{t} = \begin{bmatrix} 1 & 0 & 0 & P_x - D_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z - D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (31) $$

У цих рівняннях D_x, D_z визначають зміщення точки обертання обертових осей B відносно початку координат системи координат заготовки. Крім того, P_x, P_y, P_z є відносними відстанями точки обертання до положення кінчика різака, які також можна назвати «координатами з'єднання» точки обертання. Точка обертання знаходиться на обертовій осі B.

При множенні відповідно до (29) отримуємо:

$$ {}^{w}A_t = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B & C_C & S_C S_B & -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B & 0 & C_B & S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (32) $$

Тепер ми можемо прирівняти третій стовпець цієї матриці до нашого заданого вектора орієнтації інструменту K, тобто:

$$ K = \begin{bmatrix} K_x \\ K_y \\ K_z \\ 0 \end{bmatrix} = \begin{bmatrix} -C_C S_B \\ S_C S_B \\ C_B \\ 0 \end{bmatrix} \qquad (33) $$

З цих рівнянь ми можемо знайти кути повороту θ⁻⁹B⁻, θ⁻C⁻. З третього рядка знаходимо:

$$ \theta_B = cos^{-1}(K_z) \qquad (0 < \theta_B < \pi) \qquad (34) $$

і, поділивши другий рядок на перший рядок, знаходимо:

$$ \theta_C = tan2^{-1}(K_y, K_x) \qquad (-\pi < \theta_C < \pi) \qquad (35) $$

Ці співвідношення зазвичай використовуються в постпроцесорі CAM для перетворення векторів орієнтації інструменту в кути повороту.

Прирівнюючи останній стовпець (32) до вектора положення інструменту Q, ми можемо записати:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B(P_x - D_x) + S_C P_y - C_C S_B(P_z - D_z) + C_C D_x \\ -S_C C_B(P_x - D_X) + C_C P_y + S_C S_B(P_z - D_z) - S_C D_x \\ S_B(P_x - D_x) + C_B(P_z - D_z) + D_z \\ 1 \end{bmatrix} \qquad (36) $$

Вектор у правій частині також можна записати як добуток матриці та вектора, що дає:

$$ Q = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & S_C & -C_C S_B & -C_c C_B D_x + C_C S_B D_z + C_C D_x \\ -S_C C_B & C_C & S_C S_B & S_C C_B D_x - S_C S_B D_z - S_C D_x \\ S_B & 0 & C_B & -S_B D_x - C_B D_z + D_z \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = {}^{Q}A_P \cdot P \qquad (37) $$

що є «прямим перетворенням» кінематики.

Ми можемо знайти P з рівняння (37) як $P = ({}^{Q}A_P)^{-1} * Q$.

Використовуючи той самий підхід, що й раніше, отримуємо:

$$ \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} C_C C_B & -S_C C_B & S_B & -C_B D_x - S_B D_z + D_x \\ S_C & C_C & 0 & 0 \\ -C_C S_B & S_C S_B & C_B & S_B D_x - C_B D_z + D_z \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} \qquad (38) $$

Таким чином, бажані рівняння для «зворотного перетворення» кінематики можна записати так:

$$ \begin{aligned} P_x &= C_c C_B Q_x - S_C C_B Q_y + S_B Q_z - C_B D_x - S_B D_z + D_x \\ P_y &= S_C Q_x + C_C Q_y \\ P_z &= -C_C S_B Q_x + S_C S_B Q_y + C_B Q_z + S_B D_x - C_B D_z + D_z \end{aligned} \qquad (39) $$

9.3.6 Приклади повороту/нахилу столу

LinuxCNC включає кінематичні модулі для топологій «xyzac-trt» та «xyzbc-trt», описаних у математичн розрахунках, наведених вище. Для зацікавлених користувачів вихідний код доступний у дереві git у каталозі «src/emc/kinematics/».

Приклади конфігурацій моделювання xyzac-trt та xyzbc-trt знаходяться в каталозі Зразки конфігурацій (*configs/sim/axis/vismach/5axis/table-rotary-tilting/*).

Приклади конфігурацій містять необхідні INI-файли та підкаталог з прикладами файлів G-коду (NGC). Ці конфігурації симулятора викликають реалістичну тривимірну модель за допомогою засобу LinuxCNC vismach.

9.3.6.1 Моделі моделювання Vismach

Vismach — це бібліотека процедур Python для відображення динамічної симуляції верстата з ЧПК на екрані ПК. Скрипт Python для конкретного верстата завантажується в HAL, а дані передаються через з'єднання контактів HAL. Модель Vismach, що не працює в режимі реального часу, завантажуєть за допомогою команди HAL, наприклад:

```
loadusr -W xyzac-trt-gui
```

а з'єднання встановлюються за допомогою команд HAL, таких як:

```
net  :table-x   joint.0.pos-fb xyzac-trt-gui.table-x
net  :saddle-y  joint.1.pos-fb xyzac-trt-gui.saddle-y
...
```

Дивіться INI-файли симуляції для отримання детальної інформації про HAL-з'єднання, що використову для моделі vismach.

9.3.6.2 Компенсація довжини інструменту

Для послідовного використання інструментів з таблиці інструментів з автоматичною компенсацією довжини інструменту необхідне додаткове зміщення по осі Z. Для інструменту, який довше за «основний» інструмент, який зазвичай має довжину інструменту нуль, LinuxCNC має змінну під назвою «motion.tooloffset.z». Якщо ця змінна передається до кінематичного компонента (і скрипту vismach python), то необхідне додаткове зміщення по осі Z для нового інструменту можна врахувати, додавши оператор компонента, наприклад:

$$D_z = D_z + \text{tool-offset}$$

Необхідне HAL-з'єднання (для xyzac-trt):

```
net :tool-offset motion.tooloffset.z xyzac-trt-kins.tool-offset
```

де:

```
:tool-offset ---------------- b''нb''b''ab''b''зb''b''вb''b''ab'' b''cb''b''иb''b''гb''b' ↩
    'нb''b''ab''b''лb''b''yb''
motion.tooloffset.z --------- b''вb''b''иb''b''вb''b''ib''b''дb'' HAL-b''вb''b''иb''b''вb'' ↩
    b''ob''b''дb''b''yb'' b''зb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''яb'' b''рb''b''yb''b ↩
    ''xb''b''yb'' LinuxCNC
xyzac-trt-kins.tool-offset -- b''вb''b''вb''b''eb''b''дb''b''eb''b''нb''b''нb''b''яb'' HAL- ↩
    b''вb''b''иb''b''вb''b''ob''b''дb''b''yb'' b''дb''b''ob'' xyzac-trt-kins
```

9.3.7 Компоненти кінематики на замовлення

LinuxCNC реалізує кінематику за допомогою компонента HAL, який завантажується під час запуску LinuxCNC. Найпоширеніший кінематичний модуль, «trivkins», реалізує ідентичну (тривіальну) кінематику, де існує однозначна відповідність між літерою координати осі та з'єднанням двигуна. Доступні додаткові модулі кінематики для більш складних систем (включаючи «xyzac-trt» і «xyzbc-trt», описані вище).

Дивіться сторінку довідки kins (**\$ man kins**) для отримання коротких описів доступних модулів кінематики.

Модулі кінематики, що надаються LinuxCNC, зазвичай написані мовою C. Оскільки використовується стандартна структура, створення власного модуля кінематики спрощується шляхом копіювання існуючого вихідного файлу в файл користувача з новою назвою, його модифікації та подальшої інсталяції.

Встановлення виконується за допомогою halcompile:

```
sudo halcompile --install kinsname.c
```

де «kinsname» — це ім'я, яке ви надаєте своєму компоненту. Для його встановлення необхідний префікс sudo, і вам буде запропоновано ввести пароль адміністратора. Докладнішу інформацію див. на сторінці довідки halcompile (**\$ man halcompile**)

Після компіляції та встановлення ви можете звернутися до нього в конфігурації вашої машини. Це робиться у файлі INI вашого каталогу конфігурації. Наприклад, загальна специфікація INI:

```
[KINS]
KINEMATICS = trivkins
```

замінюється на

```
[KINS]
KINEMATICS = kinsname
```

де «kinsname» — це назва вашої програми kins. Модуль може створити додаткові контакти HAL для змінних елементів конфігурації, таких як D_x, D_y, D_z, tool-offset, що використовуються в модулі кінематики xyzac-trt. Ці контакти можна підключити до сигналу для динамічного керування або встановити один раз за допомогою підключень HAL, наприклад:

```
# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b''ab''b' ↩
    'pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''зb''b''мb''b''ib''b''щb''b''eb''b' ↩
    'нb''b''нb''b''яb''
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 Цифри

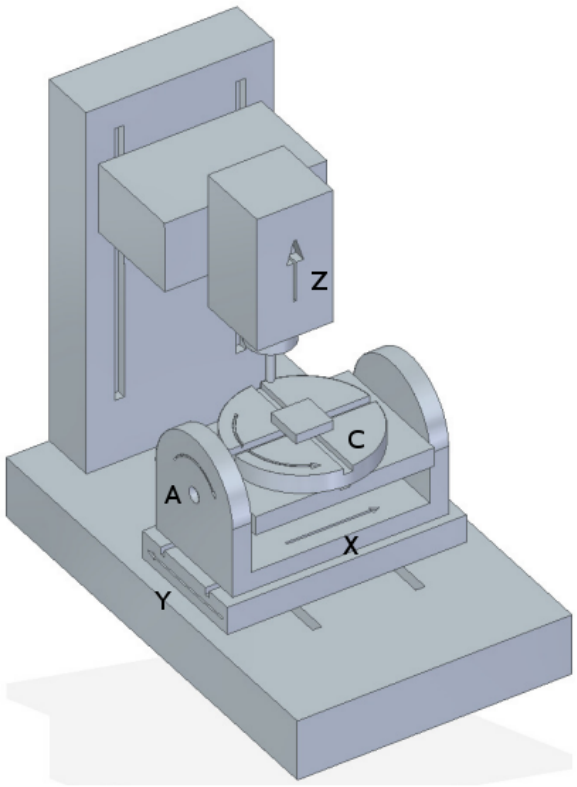


Figure 9.10: Конфігурація нахилу/повороту столу

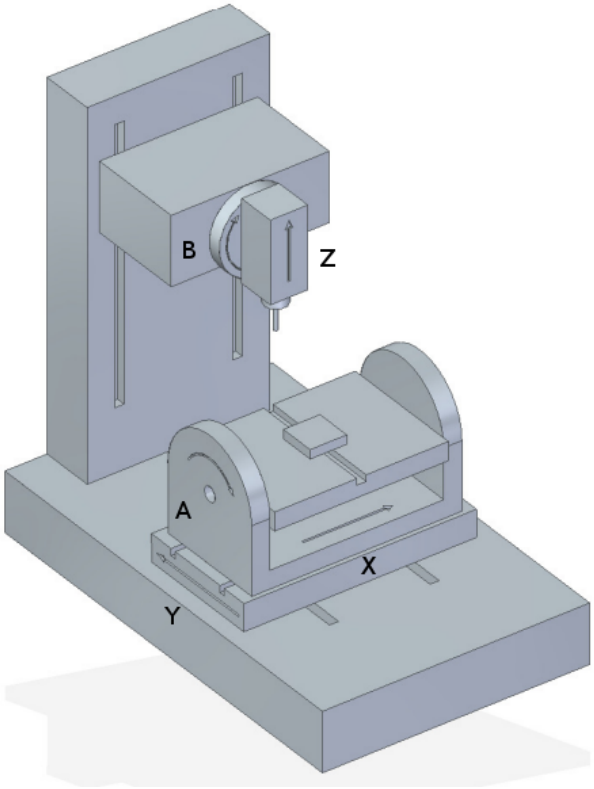


Figure 9.11: Конфігурація нахилу шпинделя/столу

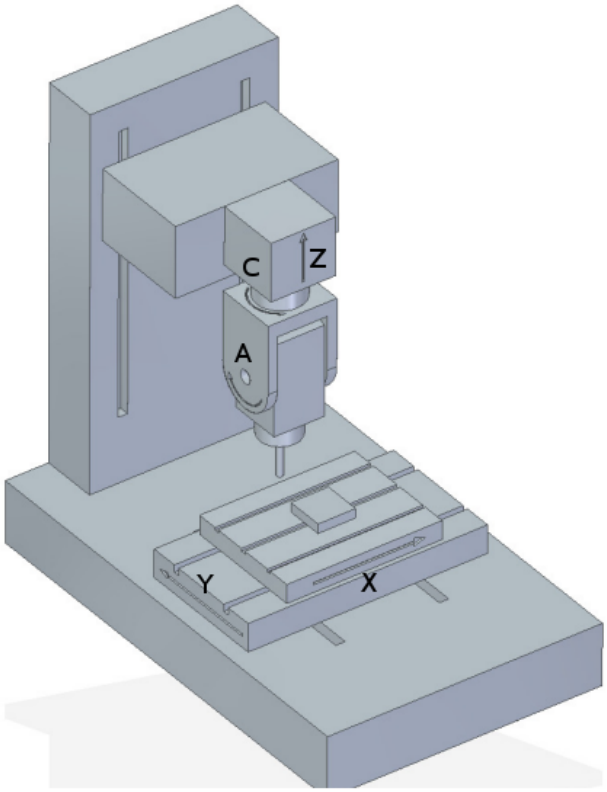


Figure 9.12: Конфігурація нахилу/повороту шпинделя

9.3.9 ПОСИЛАННЯ

1. ОСЕВІ ВЕРСТАТИ: Кінематика та реалізація Vismach у LinuxCNC, Р. Дж. дю Пріз, SA-CNC-CLUB, 7 квітня 2016 р.
2. Постпроцесор на основі кінематичної моделі для загальних п’ятиосьових верстатів: C-H She, R-S Lee, J Manufacturing Processes, V2 N2, 2000.
3. Постпроцесор з NC для 5-осьового фрезерування з поворотним/нахиленим столом: YH Jung, DW Lee, JS Kim, HS Mok, J Materials Processing Technology, 130-131 (2002) 641-646.
4. Кінематика 3D-робота з 6 ступенями свободи на серійній основі, RJ du Preez, SA-CNC-CLUB, 5 грудня 2013 р.
5. Проектування типового п’ятиосьового постпроцесора на основі узагальненої кінематичної моделі верстата: C-H She, C-C Chang, Int. J Machine Tools & Manufacture, 47 (2007) 537-545.

9.4 Перемикальна кінематика (перемикачі)

9.4.1 Вступ

Ряд кінематичних модулів підтримують перемикання кінематичних розрахунків. Ці модулі підтримуют
стандартний кінематичний метод (тип 0), другий вбудований метод (тип 1) і (опціонально) кінематичний
метод, наданий користувачем (тип 2). Для методу типу 1 зазвичай використовується ідентична
кінематика.

Функціональність switchkins може використовуватися для верстатів, де під час налаштування необхідне управління суглобами після повернення в початкове положення або для уникнення руху поблизу сингулярностей з G-коду. Такі верстати використовують специфічні кінематичні розрахунки для більшості операцій, але можуть перемикатися на ідентичну кінематику для управління окремими суглобами після повернення в початкове положення.

Тип кінематики вибирається за допомогою контакту модуля руху HAL, який можна оновити за допомогою програми G-коду або інтерактивних команд MDI. Функції halui для активації команд MDI можна використовувати для вибору типу кінематики за допомогою апаратних елементів керування або віртуальної панелі (PyVCP, GladeVCP тощо).

При зміні типу кінематики G-код також повинен видавати команди для **примусової синхронізації** інтерпретатора та рухомих частин LinuxCNC. Зазвичай для примусової синхронізації відразу після зміни керуючого контакту HAL використовується команда «читання» контакту HAL (M66 E0 L0).

9.4.2 Перемикані кінематичні модулі

Наступні кінематичні модулі підтримують перемикану кінематику:

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)

Модулі xyz[ab]c-trt-kins за замовчуванням використовують type0==xyz[ab]c-trt-kins для зворотної сумісності. Надані конфігурації sim змінюють угоду type0/type1, примусово застосовуючи type0==iden kinematics за допомогою параметра модуля *sparm* з налаштуванням INI-файлу, наприклад:

```
[KINS]
KINEMATICS = xyzac-trt-kins sparm=identityfirst
...
```

9.4.2.1 Призначення ідентифікаційних листів

Під час використання кінематики типу **identity** параметр модуля *coordinates* може бути використаний для призначення літер суглобам у довільному порядку з набору дозволених літер координат. Приклади:

```
[KINS]
JOINTS = 6

# b''зb''b''вb''b''иb''b''чb''b''аb''b''йb''b''нb''b''еb'' b''вb''b''пb''b''оb''b''рb''b' ↩
    'яb''b''дb''b''кb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb'' b''тb''b''оb''b''тb''b' ↩
    'оb''b''жb''b''нb''b''оb''b''сb''b''тb''b''ib'': joint0==x, joint1==y, ...
KINEMATICS = genhexkins coordinates=xyzabc
```

```
# b''зb''b''ab''b''мb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''ib''b''нb''b' ↩
    'дb''b''иb''b''вb''b''ib''b''дb''b''уb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''xb'' b' ↩
    'ib''b''дb''b''eb''b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b''ab''b''цb''b''ib''b' ↩
    'йb''b''нb''b''иb''b''xb'' b''дb''b''ab''b''нb''b''иb''b''xb'': joint0==c, joint1==b,  ↩
    ...
# KINEMATICS = genhexkins coordinates=cbazyx
```

Note
Якщо параметр coordinates= пропущено, призначення ідентифікаторів літер за замовчуванням буде joint0==x,joint1=y,...

Спільні завдання, передбачені для кінематики **identity** при використанні параметра координат, ідентичні тим, що передбачені для модуля trivkins. Однак дублювання літер осей для призначення декількох з'єднань для літери координати, як правило, не застосовується для послідовної або паралельної кінематики (такої як genserkins, pumakins, genhexkins тощо), де немає простого взаємозв'язку між з'єднаннями та координатами.

Дублювання літер координат осей підтримується в кінематичних модулях xyzac-trt-kins, xyzbc-trt-kins та 5axiskins (bridgemill). Типовими застосуваннями для дублювання координат є портальні верстати, де два двигуни (з'єднання) використовуються для поперечної осі.

9.4.2.2 Зворотна сумісність

Перемикається кінематика ініціалізується з motion.switchkins-type==0, реалізуючи однойменний метод кінематики. Якщо контакт motion.switchkins-type не підключений, як у старих конфігураціях, доступний тільки тип кінематики за замовчуванням.

9.4.3 Піни HAL

Перемикання кінематики контролюється вхідним контактом HAL модуля руху **motion.switchkins-type**. Значення контакту з плаваючою комою обрізається до цілого числа і використовується для вибору одного з наданих типів кінематики. Нульове значення запуску вибирає тип кінематики за замовчуванням type0.

Note
Вхідний контакт motion.switchkins-type є плаваючим, щоб полегшити підключення до вихідних контактів модуля руху, таких як motion.analog-out-0n, які можна керувати за допомогою стандартних M-кодів (зазвичай M68EnL0).

Вихідні контакти HAL призначені для інформування графічних інтерфейсів користувача про поточний тип кінематики. Ці контакти також можуть бути підключені до цифрових входів, які зчитуються програмами G-коду, щоб увімкнути або вимкнути поведінку програми відповідно до активного типу кінематики.

9.4.3.1 Зведення про піни 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 Застосування

9.4.4.1 З’єднання HAL

Функціональність Switchkins активується виводом **motion.switchkins-type**. Зазвичай цей вивід живиться від аналогового вихідного виводу, такого як motion.analog-out-03, тому його можна встановити командами M68. Наприклад:

```
net :kinstype-select <= motion.analog-out-03
net :kinstype-select => motion.switchkins-type
```

9.4.4.2 Команди G-/M-кодів

Вибір типу роду здійснюється за допомогою послідовностей G-коду, таких як:

```
...
M68 E3 Q1 ;b’’ob’’b’’нb’’b’’ob’’b’’вb’’b’’лb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ analog-out-03 b’ ↩
    ’дb’’b’’лb’’b’’яb’’ b’’вb’’b’’иb’’b’’бb’’b’’ob’’b’’pb’’b’’yb’’ kinstype 1
M66 E0 L0 ;b’’cb’’b’’иb’’b’’нb’’b’’xb’’b’’pb’’b’’ob’’b’’нb’’b’’ib’’b’’зb’’b’’ab’’b’’цb’’b’ ↩
    ’ib’’b’’яb’’ interp-motion...
...
       ;G-b’’кb’’b’’ob’’b’’дb’’ b’’кb’’b’’ob’’b’’pb’’b’’иb’’b’’cb’’b’’тb’’b’’yb’’b’’вb’’b’ ↩
           ’ab’’b’’чb’’b’’ab’’...

M68 E3 Q0 ;b’’ob’’b’’нb’’b’’ob’’b’’вb’’b’’лb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ analog-out-03 b’ ↩
    ’дb’’b’’лb’’b’’яb’’ b’’вb’’b’’иb’’b’’бb’’b’’ob’’b’’pb’’b’’yb’’ kinstype 0
M66 E0 L0 ;b’’cb’’b’’иb’’b’’нb’’b’’xb’’b’’pb’’b’’ob’’b’’нb’’b’’ib’’b’’зb’’b’’ab’’b’’цb’’b’ ↩
    ’ib’’b’’яb’’ interp-motion
...
```

Note
Команда M66 «wait-on-input» оновлює змінну #5399. Якщо поточне значення цієї змінної потрібне для подальших цілей, його слід скопіювати до додаткової змінної перед викликом M66.

Ці послідовності команд G-коду зазвичай реалізуються в підпрограмах G-коду як перепризначені M-коди або за допомогою звичайних скриптів M-коду.

Рекомендовані коди (як використовуються в конфігураціях SIM-карти):

Звичайні M-коди користувача:

1. M128 Виберіть тип кінематики 0 (запуск за замовчуванням)
2. M129 Виберіть тип кінематики 1 (зазвичай ідентична кінематика)
3. M130 Виберіть тип кінематики 2 (кінематика, задана користувачем)

Перепризначені M-коди:

1. M428 Виберіть кінематику 0 (кінематика запуску за замовчуванням)
2. M429 Виберіть тип кінематики 1 (зазвичай ідентична кінематика)
3. M430 Виберіть тип кінематики 2 (кінематика, задана користувачем)

Note
Звичайні M-коди користувача (в діапазоні M100-M199) знаходяться в модальній групі 10. Перепризначені M-коди (в діапазоні M200-M999) можуть вказувати модальну групу. Додаткову інформацію див. в документації щодо перепризначення.

9.4.4.3 Налаштування обмежень для INI-файлів

Планування траєкторії LinuxCNC використовує обмеження для положення (min, max), швидкості та прискорення для кожної відповідної літери координати, зазначеної в конфігураційному файлі INI. Приклад для літери L (у наборі «XYZABCUVW»):

```
[AXIS_L]
MIN_LIMIT =
MAX_LIMIT =
MAX_VELOCITY =
MIN_ACCELERATION =
```

Вказані обмеження файлу INI застосовуються до типу кінематики за замовчуванням типу 0, який активується під час запуску. Ці обмеження можуть **не** застосовуватися при переході на альтернативну кінематику. Однак, оскільки при переході на іншу кінематику необхідна синхронізація інтерпретатора та руху, для налаштування обмежень для очікуваного типу кінематики можна використовувати контакти INI-HAL.

Note
Виводи INI-HAL зазвичай не розпізнаються під час виконання G-коду, якщо не видано команду синхронізації (вимкнення черги). Див. сторінку довідки milltask для отримання додаткової інформації ($ man milltask).

Відповідні контакти INI-HAL для номера з'єднання (*N*):

```
ini.N.min_limit
ini.N.max_limit
ini.N.max_acceleration
ini.N.max_velocity
```

Відповідні контакти INI-HAL для координати осі (*L*):

```
ini.L.min_limit
ini.L.max_limit
ini.L.max_velocity
ini.L.max_acceleration
```

Note
Загалом, між номерами суглобів і літерами координат осей немає фіксованих відповідників. Для деяких кінематичних модулів, особливо тих, що реалізують ідентичну кінематику (trivkins), можуть існувати специфічні відповідники. Докладнішу інформацію див. на сторінці довідки kins ($ man kins).

M-код, наданий користувачем, може змінювати будь-які або всі межі координат осей перед зміною руху.switchkins-type pin і синхронізацією інтерпретатора та частин руху LinuxCNC. Наприклад, скрипт bash, що викликає halcmd, може бути «жорстко закодований» для встановлення будь-якої кількості контактів HAL:

```
#!/bin/bash
halcmd -f <<EOF
setp ini.x.min_limit -100
setp ini.x.max_limit  100
# ... b''пb''b''ob''b''вb''b''тb''b''ob''b''pb''b''ib''b''тb''b''ьb'' b''дb''b''лb''b''яb'' ↩
    b''ib''b''нb''b''шb''b''иb''b''xb'' b''гb''b''pb''b''ab''b''нb''b''иb''b''чb''b''нb''b' ↩
   'иb''b''xb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ib''b''вb''
EOF
```

Такі скрипти можна викликати як M-код користувача і використовувати **до** M-коду перемикання kins, який оновлює HAL-контакт `motion.switchkins-type` і примусово виконує синхронізацію interp-motion. Зазвичай для кожного типу kinstype (0,1,2) використовуються окремі скрипти.

Коли кінематика ідентичності надається як засіб управління окремими шарнірами, може бути зручно встановити або відновити обмеження, як зазначено в системному файлі INI. Наприклад, робот починає роботу зі складною (неідентичною) кінематикою (тип 0) після повернення в початкове положення. Система налаштована таким чином, що її можна переключити на ідентичну кінематику (тип 1) для маніпулювання окремими суглобами за допомогою звичайних літер з набору «XYZ-ABCUVW». Налаштування файлу INI ([AXIS_L]) **не** застосовуються при роботі з ідентичною кінематикою (тип 1). Для вирішення цього випадку використання скрипти M-коду користувача можна розробити наступним чином:

M129 (Перейти до типу ідентифікації 1)

1. читати та аналізувати INI-файл
2. HAL: встановити граничні контакти INI-HAL для кожної літери осі ([AXIS_L]) відповідно до налаштування INI-файлу номера з'єднання, на яке посилаються з ідентифікатором ([JOINT_N])
3. HAL: `setp motion.switchkins-type 1`
4. MDI: виконати синхронізуючий G-код (M66E0L0)

M128 (відновити кінематику робота за замовчуванням (тип 0)

1. читати та аналізувати INI-файл
2. HAL: встановити граничні контакти INI-HAL для кожної літери осі ([AXIS_L]) згідно з відповідним налаштуванням INI-файлу ([AXIS_L])
3. HAL: `setp motion.switchkins-type 0`
4. MDI: виконати синхронізуючий G-код (M66E0L0)

Note
Конфігурації симуляції vismach для робота puma демонструють сценарії M-коду (M128, M129, M130) для цього прикладу використання.

9.4.4.4 Міркування щодо зміщення системи координат

Як і налаштування обмежень файлу INI, зміщення координатної системи (G92, G10L2, G10L20, G43 тощо) зазвичай застосовуються лише для типу кінематики за замовчуванням 0. При перемиканні типів кінематики може бути **важливо** або скинути всі зміщення перед перемиканням, або оновити зміщення відповідно до вимог конкретної системи.

9.4.4.5 Міркування щодо зовнішнього зміщення

Зовнішні зміщення (встановлені на вісь (*L*) через axis.*L*.eoffset-request) зберігаються під час перемикан кінематики. Коли зміщення активне на осі перед перемиканням (видиме в axis.*L*.eoffset), планувальни траєкторії зберігає те саме зміщення після перемикання, подібно до того, як він зберігає задане положення з G-коду. Це забезпечує стабільну роботу машини незалежно від активної кінематики.

Якщо підтримка зміщення буде проблемою через зміни обмежень осей або інші проблеми, обов'язково очистіть і, можливо, вимкніть eoffset, перш ніж перемикати кінематику.

9.4.5 Конфігурації симуляції

Конфігурації симуляції (що не потребують апаратного забезпечення) надаються з ілюстративними дисплеями vismach у підкаталогах 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 Положення щодо кінематики користувача

Настроювана кінематика може бути закодована і протестована на збірках Run-In-Place («RIP»). У дистрибутиві надається файл-шаблон src/emc/kinematics/userkfuncs.c. Цей файл можна скопіювати/пе в каталог користувача і відредагувати, щоб надати настроювану кінематику з kinstype==2.

Файл користувацької кінематики можна скомпілювати з позадеревних джерел для реалізацій rt-preempt або шляхом заміни шаблонного файлу в дереві (src/emc/kinematics/userkfuncs.c) для систем rtai.

Приклад створення Preempt-rt:

```
$ userkfuncs=/home/myname/kins/mykins.c make && sudo make setuid
```

9.4.7 Попередження

Неочікувана поведінка може виникнути, якщо програму G-коду випадково запустити з несумісним типом кінематики. Небажаної поведінки в програмах G-коду можна уникнути шляхом:

1. Підключення відповідних контактів HAL kintype.is.N до контактів цифрового входу (наприклад, motion.digital-in-0m).
2. Зчитування цифрового вхідного виводу (M66 E0 Pm) на початку програми G-коду
3. Переривання (M2) програми G-коду з повідомленням (DEBUG, problem_message), якщо тип роду не підходить.

Під час інтерактивного використання засобів бігової роботи або команд MDI оператору потрібна обережність. GuiDesign повинен містити індикатори для відображення поточного типу кінематики.

Note
Зміна кінематики може спричинити істотні операційні зміни, що вимагають ретельного проектування, тестування та навчання для впровадження. Управління зміщеннями координат, компенсацією інструменту та обмеженнями файлів INI може вимагати складних і нестандартних операційних протоколів.

9.4.8 Примітки до коду

Кінематичні модулі, що забезпечують функціональність switchkins, пов'язані з об'єктом switchkins.o (switchkins.c), який забезпечує роботу головної програми модуля (rtapi_app_main()) та пов'язаних з нею функцій. Ця головна програма зчитує (опціональні) параметри командного рядка модуля (координати, sparm) і передає їх до функції switchkinsSetup(), що надається модулем.

Функція switchkinsSetup() визначає специфічні для kintype процедури налаштування та функції для прямого оберненого обчислення для кожного kintype (0,1,2) і встановлює ряд параметрів конфігурації.

Після виклику switchkinsSetup(), rtapi_app_main() перевіряє надані параметри, створює компонент HAL, а потім викликає процедуру налаштування, визначену для кожного типу kin (0,1,2).

Кожна процедура налаштування kinstype (0,1,2) може (за бажанням) створювати контакти HAL і встановлювати для них значення за замовчуванням. Коли всі процедури налаштування завершені, rtapi_app_main() видає hal_ready() для компонента, щоб завершити створення модуля.

9.5 Налаштування PID-регулятора

9.5.1 PID Контролер

Пропорційно-інтегрально-диференціальний контролер (PID-контролер) є поширеним компонентом контуру зворотного зв'язку в промислових системах управління. Примітка: [Цей підрозділ взято з набагато більш розгорнутої статті, розміщеної за адресою https://en.wikipedia.org/wiki/PID_controller]

Контролер порівнює виміряне значення з процесу (зазвичай промислового процесу) з еталонним значенням заданого параметра. Різниця (або сигнал «помилки») потім використовується для обчислення нового значення для маніпульованого вхідного сигналу процесу, що повертає виміряне значення процесу до бажаного заданого параметра.

На відміну від простіших алгоритмів управління, PID-регулятор може коригувати вихідні параметри процесу на основі історії та швидкості зміни сигналу помилки, що забезпечує більш точне та стабільне управління. (Математично можна довести, що PID-контур забезпечить точне та стабільне управління в тих випадках, коли просте пропорційне управління призвело б до стаціонарної помилки або спричинило б коливання процесу).

9.5.1.1 Основи контуру керування

Інтуїтивно, PID-контур намагається автоматизувати те, що зробив би розумний оператор, маючи в своєму розпорядженні вимірювальний прилад і ручку регулювання. Оператор зчитував би показники вимірювального приладу, що відображають вихідні параметри процесу, і за допомогою ручки регулювання коригував би вхідні параметри процесу («дію») доти, доки вихідні параметри процесу не стабілізуються на бажаному рівні, що відображається на вимірювальному приладі.

У старій літературі з питань управління цей процес регулювання називається дією «перезавантаження». Положення стрілки на манометрі є «вимірюванням», «значенням процесу» або «змінною процесу». Бажане значення на манометрі називається «заданим значенням» (також «встановленим значенням»). Різниця між стрілкою манометра і заданим значенням є «похибкою».

Контур керування складається з трьох частин:

1. Вимірювання датчиком, підключеним до процесу (наприклад, енкодером),
2. Рішення в елементі контролера,
3. Дія через вихідний пристрій, такий як двигун.

Коли контролер зчитує дані з датчика, він віднімає це значення від «заданого значення», щоб визначити «похибку». Потім він використовує похибку для обчислення корекції вхідної змінної процесу («дія»), так що ця корекція усуне похибку з вимірювання вихідних даних процесу.

У PID-контурі корекція розраховується на основі помилки трьома способами: безпосереднє скасування поточної помилки (пропорційний), час, протягом якого помилка залишалася невиправленою (інтегральний), та передбачення майбутньої помилки на основі швидкості її зміни з часом (похідний).

PID-регулятор може використовуватися для регулювання будь-якої вимірюваної змінної, на яку можна вплинути шляхом маніпулювання іншою змінною процесу. Наприклад, його можна використов для регулювання температури, тиску, швидкості потоку, хімічного складу, швидкості або інших змінних. Круїз-контроль в автомобілях є прикладом процесу поза промисловістю, в якому використову примітивний PID-регулятор.

Деякі системи управління організовують PID-регулятори в каскади або мережі. Тобто «головний» регулятор генерує сигнали, які використовуються «підлеглими» регуляторами. Типовою ситуацією є управління двигунами: часто потрібно, щоб двигун мав регульовану швидкість, причому «підлеглий» контролер (часто вбудований у привід із змінною частотою) безпосередньо керує швидкістю на основі пропорційного вхідного сигналу. Цей «підлеглий» вхідний сигнал подається з виходу «головного» контролера, який здійснює керування на основі відповідної змінної.

9.5.1.2 Теорія

«PID» названо на честь трьох коригувальних обчислень, які додаються до регульованої величини та коригують її. Ці додавання насправді є «відніманнями» похибки, оскільки пропорції зазвичай є від'ємними:

Пропорційний Для обробки поточного значення помилка множиться на (від'ємну) константу P (від «пропорційна») і додається до (віднімається від) регульованої величини. P дійсна тільки в діапазоні, в якому вихідний сигнал регулятора пропорційний помилці системи. Зверніть увагу, що коли помилка дорівнює нулю, вихідний сигнал пропорційного регулятора дорівнює нулю.

Інтеграл Щоб вчитися на минулому, похибка інтегрується (сумується) за певний проміжок часу, а потім множиться на (від'ємну) константу I (утворюючи середнє значення) і додається до (віднімається від) регульованої величини. I усереднює виміряну похибку, щоб знайти середню похибку вихідних даних процесу від заданого значення. Проста пропорційна система або коливається, рухаючись вперед і назад навколо заданого значення, оскільки немає нічого, що могло б усунути помилку, коли вона перевищує задане значення, або коливається і/або стабілізується на занадто низькому або занадто високому значенні. Додаючи від'ємну частку (тобто віднімаючи частину) середньої помилки від вхідного значення процесу, середня різниця між вихідним значенням процесу і заданим значенням завжди зменшується. Тому, зрештою, вихідний сигнал добре налагодженого PID-контуру стабілізується на заданому значенні.

Похідна Для обробки майбутнього значення обчислюється перша похідна (нахил похибки) з часом, яка множиться на іншу (від'ємну) константу D, а також додається до (віднімається від) керованої величини. Похідна величина контролює реакцію на зміну в системі. Чим більша похідна величина, тим швидше контролер реагує на зміни в вихідних даних процесу.

Більш технічно, PID-контур можна охарактеризувати як фільтр, що застосовується до складної системи частотного діапазону. Це корисно для розрахунку того, чи дійсно він досягне стабільного значення. Якщо значення обрані неправильно, вхідний сигнал керованого процесу може коливатися, і вихідний сигнал процесу може ніколи не залишатися на заданому рівні.

9.5.1.3 Налаштування петлі

«Налаштування» контуру регулювання — це регулювання його параметрів (коефіцієнт підсилення/про смуга, інтегральний коефіцієнт підсилення/скидання, похідний коефіцієнт підсилення/швидкість) до оптимальних значень для бажаної реакції системи регулювання. Оптимальна поведінка при

зміні процесу або зміні заданого значення залежить від застосування. Деякі процеси не повинні допускати перевищення змінної процесу над заданим значенням. Інші процеси повинні мінімізувати енергію, що витрачається на досягнення нової заданої величини. Як правило, необхідна стабільність реакції, і процес не повинен коливатися для будь-якої комбінації умов процесу та заданих величин.

Налаштування контурів ускладнюється часом відгуку процесу; для досягнення стабільного ефекту від зміни заданого значення може знадобитися кілька хвилин або годин. Деякі процеси мають певний ступінь нелінійності, тому параметри, які добре працюють в умовах повного навантаження, не працюють, коли процес запускається з нульового навантаження. У цьому розділі описано деякі традиційні ручні методи налаштування контурів.

Існує кілька методів налаштування PID-контуру. Вибір методу буде значною мірою залежати від того, чи можна вивести контур з експлуатації для налаштування, а також від швидкості реакції системи. Якщо систему можна вивести з експлуатації, найкращий метод налаштування часто полягає в тому, щоб піддати систему стрибкоподібній зміні вхідного сигналу, виміряти вихідний сигнал як функцію часу і використовувати цю реакцію для визначення параметрів регулювання.

Простий метод Якщо система повинна залишатися в режимі онлайн, одним із методів налаштування є спочатку встановити значення I та D на нуль. Збільшуйте P, поки вихід контуру не почне коливатися. Потім збільшуйте I, поки коливання не припиняться. Нарешті, збільшуйте D, поки контур не досягне прийнятної швидкості для досягнення свого опорного значення. Швидке налаштування контуру PID зазвичай трохи перевищує задане значення, щоб швидше досягти заданої точки; однак деякі системи не можуть прийняти перевищення.

Параметр	Час наростання	Перевищення	Час затвердіння	Помилка стаціонарного стану
P	Зменшення	Збільшення	Дрібні зміни	Зменшення
I	Зменшення	Збільшення	Збільшення	Усунути
D	Дрібні зміни	Зменшення	Зменшення	Дрібні зміни

Вплив збільшення параметрів

Метод Циглера-Ніколса Інший метод налаштування офіційно відомий як «Метод Зіглера-Ніколса», запропонований Джоном Г. Зіглером і Натаніелем Б. Ніколсом у 1942 році. Примітка: [Зіглер, Дж. Г. і Ніколс, Н. Б. (1942), «Оптимальні налаштування для автоматичних контролерів», Transactions of the ASME, посилання: DOI 10.1115/1.2899060 та посилання:https://web.archive.org/web/ Інтернет-архів.]. Починається так само, як і описаний раніше метод: спочатку встановіть коефіцієнти підсилення I і D на нуль, а потім збільште коефіцієнт підсилення P і піддайте контур зовнішньому впливу, наприклад, стукаючи по осі двигуна, щоб вивести його з рівноваги, щоб визначити критичний коефіцієнт підсилення і період коливань, поки вихід контуру не почне коливатися. Запишіть критичне посилення (K_c) і період коливань виходу (P_c). Потім відрегулюйте регулятори P, I і D, як показано в таблиці:

Тип керування	P	I	D
P	$.5K_c$		
PI	$.45K_c$	$P_c/1.2$	
PID	$.6K_c$	$P_c/2$	$P_c/8$

Заключні кроки Після налаштування осі перевірте наступну помилку за допомогою Halscope, щоб переконатися, що вона відповідає вимогам вашого верстата. Більше інформації про Halscope можна знайти в посібнику користувача HAL.

9.5.1.4 Автоматичне налаштування PID-регулятора

Починаючи з версії LinuxCNC 2.9, компонент pid підтримує автоматичне налаштування за допомогою методу реле. Примітка: [Åström, Karl Johan і Hägglund, Tore (1984), «Автоматичне налаштування

простих регуляторів із специфікаціями щодо фазових і амплітудних запасів», посилання: DOI 10.1016/0005-1098(84)90014-1]. Це заміна для компонента at_pid, який зараз видалено і який є застарілим.

Компонент pid використовує кілька констант для обчислення вихідного значення на основі поточного та бажаного стану, найважливішими з яких є Pgain', *Igain*, *Dgain*, *bias*, *FF0*, *FF1*, *FF2* та *FF3*. Усі вони повинні мати розумне значення, щоб контролер працював належним чином.

Поточна реалізація автоматичного налаштування використовує два різних алгоритми, які вибираються за допомогою контакту tune-type. Коли tune-type дорівнює нулю, це впливає на *Pgain*, *Igain* і *Dgain*, а *FF0*, *FF1* і *FF2* встановлюються на нуль. Якщо tune-type дорівнює 1, він впливає на «Pgain», «Igain» і «FF1», встановлюючи «Dgain», «FF0» і «FF2» на нуль. Тип 1 вимагає встановлення масштабування, щоб вихід був у одиницях користувача на секунду.

При автоналагодженні двигуна з типом налаштування 0 алгоритм буде генерувати прямокутну хвилю, центровану навколо значення «зсуву» на вихідному контакті PID-регулятора, рухаючись від позитивного до негативного краю діапазону виходу. Це можна побачити за допомогою HAL Scope, що надається LinuxCNC. Для контролера двигуна, який приймає +-10 В як сигнал управління, це може прискорити двигун до повної швидкості в одному напрямку на короткий період, перш ніж дати йому команду рухатися з повною швидкістю в протилежному напрямку. Переконайтеся, що з обох боків від вихідної позиції є достатньо місця, і почніть з низького значення `tune-effort`, щоб обмежити використовувану швидкість. Значення `tune-effort` визначає граничне значення `output`, яке використовується, тому якщо `tune-effort` дорівнює 1, значення `output` під час налаштуван буде змінюватися від 1 до -1. Іншими словами, граничні значення хвильового патерну контролюються контактом «tune-effort». Використання занадто високого значення «tune-effort» може призвести до перевантаження драйвера двигуна.

Кількість циклів у патерні налаштування контролюється контактом «tune-cycles». Звичайно, спроба миттєво змінити напрямок фізичного об'єкта (як у випадку переходу від позитивного напруги до еквівалентного негативного напруги в контролері двигуна) не змінює швидкість миттєво, і об'єкту потрібно деякий час, щоб сповільнитися і рухатися в протилежному напрямку. Це призводить до більш плавної форми хвилі на контакті положення, оскільки відповідна вісь вібрувала вперед і назад. Коли вісь досягала цільової швидкості в протилежному напрямку, автотюнер знову змінював напрямок. Після декількох таких змін середня затримка між «піками» і «долинами» цього графіка руху використовується для обчислення запропонованих значень для `Pgain`, `Igain` і `Dgain` і вставляється в модель HAL для використання контролером pid. Автоматично налаштовані параметри не є ідеальними, але можуть стати хорошим відправним пунктом для подальшого налаштування параметрів.

FIXME: Автор цих інструкцій не тестував автоматичне налаштування з типом tune-type, встановленим на 1, тому цей підхід потребує документування.

Озброївшись цими знаннями, можна перейти до розгляду того, як виконати налаштування. Припустим що конфігурація HAL завантажує компонент PID для X, Y і Z таким чином, використовуючи імена виводів замість *count=3*:

```
loadrt pid names=pid.x,pid.y,pid.z
```

Якщо компонент використовував *count=3*, то всі використання pid.x, pid,y та pid.z потрібно змінити відповідно на pid.1, pid.2 та pid.3. Щоб почати налаштування осі X, перемістіть вісь до середини її діапазону, щоб переконатися, що вона ні про що не зачепиться, коли почне рухатися вперед і назад. Також слід розширити межу похибки осі (похибка слідування), щоб LinuxCNC приймав більшу відхилення положення під час налаштування. Розумна межа похибки залежить від машини та налаштувань, але 1 дюйм або 20 мм можуть бути корисними вихідними точками. Далі встановіть початкове значення «tune-effort» на низьке число в діапазоні вихідних значень, наприклад 1/100 від максимального вихідного значення, і повільно збільшуйте його, щоб отримати більш точні значення налаштування. Присвойте значення «tune-mode» 1. Зверніть увагу, що це вимкне частину PID-регулювання і подасть значення «bias» на вихідний контакт, що може спричинити значне відхилення. Може бути доцільним налаштувати драйвер двигуна, щоб забезпечити, що нульова вхідна напруга не спричиняє обертання двигуна, або відрегулювати значення «bias» для

досягнення того самого ефекту. Нарешті, після налаштування «tune-mode», встановіть «tune-start» на 1, щоб активувати автоматичне налаштування. Якщо все пройде добре, ваша вісь буде вібрувати і рухатися вперед і назад протягом декількох секунд, а коли це буде зроблено, нові значення для Pgain, Igain і Dgain будуть активними. Щоб перевірити їх, змініть «tune-mode» назад на 0. Зверніть увагу, що повернення «tune-mode» до нуля може спричинити раптове ривкування осі, оскільки вона повертається до заданого положення, від якого вона могла відхилитися під час налаштування. Підсумовуючи, ось інструкції halcmd, які потрібно виконати для автоматичного налаштування:

```
setp pid.x.tune-effort 0.1
setp pid.x.tune-mode 1
setp pid.x.tune-start 1
# b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'' b''зb''b''ab''b''вb''b' ↩
    'eb''b''pb''b''шb''b''eb''b''нb''b''нb''b''яb'' b''нb''b''ab''b''лb''b''ab''b''шb''b' ↩
    'тb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''
setp pid.x.tune-mode 0
```

Скрипт, що допомагає виконати автоматичне налаштування, надається в репозиторії коду LinuxCNC як «scripts/run-auto-pid-tuner». Це забезпечить увімкнення машини та її готовність до роботи, повернення всіх осей у вихідне положення, якщо це ще не зроблено, перевірку наявності додаткових налаштувальних контактів, переміщення осі в її середню точку, виконання автоматичного налаштування та повторне увімкнення pid-регулятора після його завершення. Його можна запускати кілька разів.

9.6 Перепризначення розширення G-коду

9.6.1 Вступ: Розширення інтерпретатора RS274NGC шляхом перепризначенн кодів

9.6.1.1 Визначення: Перепризначення кодів

Під «перепризначенням кодів» ми маємо на увазі одне з наступного:

1. Визначити семантику нових, тобто наразі нерозподілених, M- або G-кодів
2. Перевизначити семантику – наразі обмеженого – набору існуючих кодів.

9.6.1.2 Навіщо розширювати інтерпретатор RS274NGC?

Набір кодів (M, G, T, S, F), які наразі розуміє інтерпретатор RS274NGC, є фіксованим і не може бути розширений за допомогою параметрів конфігурації.

Зокрема, деякі з цих кодів реалізують фіксовану послідовність кроків, які необхідно виконати. Хоча деякі з них, наприклад M6, можна частково налаштувати, активуючи або пропускаючи деякі з цих кроків за допомогою опцій файлу INI, загалом поведінка є досить жорсткою. Отже, якщо вас влаштовує така ситуація, то цей розділ посібника не для вас.

У багатьох випадках це означає, що підтримка нестандартної конфігурації або машини є або громіздкою, або неможливою, або вимагає звернення до змін на рівні мови «C/C+\+». Останнє є непопулярним з поважних причин — зміна внутрішніх компонентів вимагає глибокого розуміння внутрішніх компонентів інтерпретатора, а крім того, створює власний набір проблем із підтримкою. Хоча можна припустити, що певні виправлення можуть потрапити до основного дистрибутива LinuxCNC, результатом такого підходу є мішанина рішень для особливих випадків.

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.

У найпростішому вигляді перемапований код є не більше ніж спонтанним викликом процедури O-word. Це відбувається за лаштунками — процедура видима на рівні конфігурації, але не на рівні програми NGC.

Зазвичай, поведінку перепризначеного коду можна визначити такими способами:

- Ви визначаєте підпрограму на літеру O, яка реалізує бажану поведінку
- Або ж ви можете використовувати функцію Python, яка розширює поведінку інтерпретатора.

Як склеїти речі разом M- та G-коди, а також виклики підпрограм на основі O-слів мають дещо інший синтаксис.

Наприклад, процедури O-word приймають позиційні параметри зі специфічним синтаксисом, наприклад:

```
o<test> call [1.234] [4.65]
```

тоді як M- або G-коди зазвичай приймають обов'язкові або додаткові параметри типу «слово». Наприклад, G76 (різьблення) вимагає слів P, Z, I, J та K, а також додатково приймає слова R, Q, H, E та L.

Тому недостатньо просто сказати «коли ви зустрінете код X, запустіть процедуру Y» — необхідно принаймні перевірити та перетворити параметри. Для цього потрібен «клейовий код» між новим кодом та відповідною процедурою NGC, який виконується перед передачею контролю процедурі NGC.

Цей код-сполучна неможливо написати як процедуру O-word, оскільки мова RS274NGC не має інтроспективних можливостей та доступу до внутрішніх структур даних інтерпретатора для досягненн необхідного ефекту. Виконання коду-сполучної знову в «C/C+\+» було б негнучким і, отже, незадовільним рішенням.

Як вбудований Python вписується Щоб зробити просту ситуацію легкою, а складну – вирішувальною проблему з клеєм вирішують наступним чином:

- Для простих ситуацій вбудована процедура склеювання (`argspec`) охоплює найпоширеніші вимоги до передачі параметрів.
- Для перепризначення T, M6, M61, S, F існує стандартний клей Python, який має охопити більшість ситуацій, див. Standard Glue.
- Для складніших ситуацій можна написати власний клейовий агент Python для реалізації нової поведінки.

Вбудовані функції Python в інтерпретаторі спочатку були лише допоміжним кодом, але виявилися дуже корисними і в інших сферах. Користувачам, знайомим з Python, буде простіше писати перемаповані коди, допоміжний код, процедури O-word тощо в чистому Python, не вдаючись до дещо громіздкої мови RS274NGC.

Кілька слів про вбудований Python Багато людей знайомі з «розширенням» інтерпретатора Python за допомогою модулів «C/C++», і це широко використовується в LinuxCNC для доступу до внутрішніх компонентів Task, HAL та Interpreter зі скриптів Python. «Розширення Python» в основному означає: ваш скрипт Python виконується «як у водійському сидінні» і може отримувати

доступ до коду, що не належить до Python, шляхом імпортування та використання модулів розширення написаних на «C/C+\+». Прикладами цього є модулі LinuxCNC `hal`, `gcode` та `emc`.

Вбудований Python дещо відрізняється і є менш відомим: основна програма написана на C/C++ і може використовувати Python як підпрограму. Це потужний механізм розширення і основа для «розширень скриптів», які можна знайти в багатьох успішних програмних пакетах. Вбудований код Python може отримувати доступ до змінних і функцій «C/C+\+» за допомогою подібного методу модуля розширення.

9.6.2 Початок роботи

Визначення коду включає такі кроки:

- Виберіть код – або використовуйте нерозподілений код, або перевизначте існуючий код.
- Визначте, як обробляються параметри.
- Вирішіть, чи будуть оброблені результати і як.
- Визначтеся з послідовністю виконання.

9.6.2.1 Вбудовані перепризначення

Зверніть увагу, що наразі можна перевизначити лише деякі існуючі коди, тоді як існує багато «вільних» кодів, які можна перепризначити. При розробці перевизначеного існуючого коду рекомендується почати з не призначеного коду G або M, щоб можна було використовувати як існуючу, так і нову поведінку. Після завершення перевизначте існуючий код, щоб використовувати вашу конфігурацію перепризначення.

- Поточний набір невикористаних M-кодів, доступних для визначення користувачем, можна знайти в розділі unallocated M-codes.
- Щодо G-кодів див. unallocated G-codes.
- Існуючі коди, які можна перепризначити, перелічені в розділі remappable codes.

Наразі у stdglue.py доступні два повні перепризначення лише для Python:

- ignore_m6
- index_lathe_tool_with_wear

Вони призначені для використання з токарним верстатом. Токарні верстати не використовують M6 для індексації інструментів, вони використовують команду T.

Це перемапування також додає зносові зміщення до зміщення інструменту, наприклад, T201 буде індексувати інструмент 2 (зі зміщенням інструменту 2) і додає зносове зміщення 1. У таблиці інструментів номери інструментів вище 10000 є зносовими зміщеннями, наприклад, у таблиці інструментів інструмент 10001 буде зносовим зміщенням 1.

Ось що потрібно в INI-файлі для їх використання:

```
[RS274NGC]
REMAP=T python=index_lathe_tool_with_wear
REMAP=M6 python=ignore_m6

[PYTHON]
# b''дb''b''eb'' b''зb''b''нb''b''аb''b''йb''b''тb''b''иb'' b''кb''b''оb''b''дb'' Python:
```

```
# b''кb''b''ob''b''дb'', b''cb''b''пb''b''eb''b''цb''b''иb''b''фb''b''ib''b''чb''b''нb''b' ↩
    'иb''b''йb'' b''дb''b''лb''b''яb'' b''цb''b''ib''b''єb''b''їb'' b''кb''b''ob''b''нb''b' ↩
    'фb''b''ib''b''гb''b''уb''b''pb''b''ab''b''цb''b''ib''b''їb''
PATH_PREPEND=./

# b''зb''b''ab''b''гb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''кb''b''ob''b''дb'' b' ↩
    'пb''b''ib''b''дb''b''тb''b''pb''b''иb''b''мb''b''кb''b''иb'' b''—b'' b''пb''b''eb''b' ↩
    'pb''b''eb''b''кb''b''ob''b''нb''b''ab''b''йb''b''тb''b''eb''b''cb''b''яb'', b''щb''b' ↩
    'ob'' b''вb''b''ib''b''нb'' b''дb''b''ib''b''йb''b''cb''b''нb''b''ob'' b''вb''b''кb''b' ↩
    'ab''b''зb''b''уb''b''єb'' b''нb''b''ab'' Python-stdglue
PATH_APPEND=../../nc_files/remap_lib/python-stdglue/

# b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''уb''b''йb''b''тb''b''eb'' b''нb''b''ab''b' ↩
    'cb''b''тb''b''уb''b''пb''b''нb''b''иb''b''йb'' b''мb''b''ob''b''дb''b''уb''b''лb''b' ↩
    'ьb'' Python
TOPLEVEL=toplevel.py

# b''чb''b''иb''b''мb'' b''вb''b''иb''b''щb''b''eb'', b''тb''b''иb''b''мb'' b''дb''b''eb''b ↩
    ''тb''b''ab''b''лb''b''ьb''b''нb''b''ib''b''шb''b''eb'' b''тb''b''pb''b''ab''b''cb''b' ↩
    'уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''пb''b''лb''b''ab''b''гb''b''ib''b''нb''b' ↩
    'ab'' Python
LOG_LEVEL = 0
```

Ви також повинні додати необхідний файл Python до папки конфігурації.

Оновлення існуючої конфігурації

9.6.2.2 Вибір коду

Зверніть увагу, що на даний момент можна перевизначити лише декілька існуючих кодів, тоді як існує багато «вільних» кодів, які можна зробити доступними шляхом перепризначення. При розробці перевизначеного існуючого коду, можливо, варто почати з нерозподіленого коду G або M, щоб можна було реалізувати як існуючу, так і нову поведінку. Після завершення перевизначте існуючий код, щоб використовувати ваші налаштування перепризначення.

- Поточний набір невикористаних M-кодів, відкритих для визначення користувачем, можна знайти тут.
- Нерозподілені G-коди перелічені тут.
- Існуючі коди, які можна перепризначити, перелічені тут.

9.6.2.3 Обробка параметрів

Припустимо, що новий код буде визначений процедурою NGC і потребує деяких параметрів, деякі з яких можуть бути обов'язковими, а інші — необов'язковими. Ми маємо такі варіанти для введення значень у процедуру:

1. Вилучення слів з поточного блоку та передача їх процедурі як параметрів (наприклад, `X22.34` або `P47`),
2. посилаючись на змінні файлу INI,
3. посилання на глобальні змінні (наприклад, `#2200 = 47.11` або `#<_global_param> = 315.2`).

Перший метод є кращим для параметрів динамічного характеру, таких як позиції. Вам потрібно визначити, які слова в поточному блоці мають якесь значення для вашого нового коду, і вказати,

як це передається до процедури NGC. Найпростіший спосіб — це використання оператора argspec. Налаштований пролог може надавати кращі повідомлення про помилки.

Використання змінних INI-файлу є найбільш корисним для посилання на інформацію про налаштуванн вашої машини, наприклад, на фіксоване положення, таке як положення датчика довжини інструменту Перевага цього методу полягає в тому, що параметри є фіксованими для вашої конфігурації, незалежно від того, який NGC-файл ви виконуєте в даний момент.

Звернення до глобальних змінних завжди можливе, але їх легко пропустити.

Зверніть увагу, що кількість слів, які можна використовувати як параметри, обмежена, тому, якщо потрібно багато параметрів, може знадобитися повернутися до другого та третього методів.

9.6.2.4 Обробка результатів

Ваш новий код може працювати або не працювати, наприклад, якщо передано недійсну комбінацію параметрів. Або ви можете вирішити «просто виконати» процедуру і не зважати на результати, і в цьому випадку роботи буде небагато.

Обробники епілогів допомагають в обробці результатів процедур перепризначення - див. розділ з посиланнями.

9.6.2.5 Послідовність виконання

Виконувані слова G-коду класифікуються на modal groups, що також визначає їхню відносну поведінку під час виконання.

Якщо блок G-коду містить кілька виконуваних слів у рядку, ці слова виконуються у заздалегідь визначеному порядку order of execution, а не в порядку, в якому вони з'являються в блоці.

Коли ви визначаєте новий виконуваний код, інтерпретатор ще не знає, де ваш код вписується в цю схему. З цієї причини вам потрібно вибрати відповідну модальну групу для виконання вашого коду.

9.6.2.6 Мінімальний приклад переробленого коду

Щоб ви могли уявити, як всі елементи поєднуються між собою, давайте розглянемо досить мінімальне, але повне визначення перемапованого коду. Ми вибираємо нерозподілений M-код і додаємо до файлу INI наступну опцію:

```
[RS274NGC]
REMAP=M400  modalgroup=10 argspec=Pq ngc=myprocedure
```

Коротко кажучи, це означає:

- Код `M400` приймає обов'язковий параметр `P` та необов'язковий параметр `Q`. Інші слова в поточному блоці ігноруються стосовно коду `M400`. Якщо слово `P` відсутнє, виконання завершується з помилкою.
- Коли зустрічається код `M400`, виконайте `myprocedure.ngc` разом з іншими 10 M-кодами modal group відповідно до order of execution.
- Значення `P` та `Q` доступні в процедурі як локальні іменовані параметри. На них можна посилатися як `#<P>` та `#<Q>`. Процедура може перевірити, чи слово `Q` було присутнє за допомогою вбудованої функції EXISTS.

Очікується, що файл `myprocedure.ngc` знаходиться в каталозі `[DISPLAY]NC_FILES` або `[RS274NGC]SUBRO`

Детальний опис параметрів REMAP можна знайти в розділі з посиланнями нижче.

9.6.3 Налаштування перепризначення

9.6.3.1 Заява REMAP

Щоб перепризначити код, визначте його за допомогою опції REMAP у розділі RS274NG вашого INI-файлу. Використовуйте один рядок REMAP на кожен перепризначений код.

Синтаксис REMAP такий:

REMAP=*<code> <options>*
де *<код>* може бути одним із T,M6,M61,S,F (існуючі коди) або будь-яким із нерозподілених M-codes або G-codes.

Пропускати параметр *<code>* є помилкою.

Опції оператора REMAP розділені пробілами. Опції є парами ключове слово-значення і наразі мають такий вигляд:

modalgroup=*<modal group>*

G-коди
Єдина підтримувана наразі модальна група — 1, яка також є значенням за замовчуванням, якщо група не вказана. Група 1 означає «виконувати разом з іншими G-кодами».

M-коди
Наразі підтримуються такі модальні групи: 5, 6, 7, 8, 9, 10. Якщо модальна група не вказана, значення за замовчуванням дорівнює 10 («виконувати після всіх інших слів у блоці»).

T,S,F
для них модальна група фіксована, а будь-який параметр modalgroup= ігнорується.

argspec=*<argspec>*
Див. опис опцій параметра argspec. Необов'язково.

ngc=*<ngc_basename>*
Базова назва файлу підпрограми O-word. Не вказуйте розширення .ngc. Шукається в каталогах, вказаних у каталозі, вказаному в [DISPLAY]PROGRAM_PREFIX, а потім в [RS274NGC]SUBROUTINE_PATH Взаємовиключне з python=. Опущення як ngc=, так і python= є помилкою.

python=*<Назва функції Python>*
Замість виклику процедури ngc O-word, викличте функцію Python. Очікується, що функція буде визначена в модулі module_basename.oword. Взаємно виключні з ngc=.

prolog=*<Назва функції Python>*
Перед виконанням процедури ngc викличте цю функцію Python. Очікується, що функція буде визначена в модулі module_basename.remap. Необов'язково.

epilog=*<Назва функції Python>*
Після виконання процедури ngc викличте цю функцію Python. Очікується, що функція буде визначена в модулі module_basename.remap. Необов'язково.

Для опцій python, prolog та epilog необхідний плагін Python Interpreter, який має бути configured, а також відповідні функції Python, які мають бути визначені там, щоб їх можна було використовувати з цими опціями.

Синтаксис для визначення нового коду та перевизначення існуючого коду ідентичний.

9.6.3.2 Корисні комбінації опцій REMAP

Зверніть увагу, що хоча можливі багато комбінацій параметрів argspec, не всі вони мають сенс. Наступні комбінації є корисними ідіомами:

argspec=<*words*> ngc=<*procname*> modalgroup=_<*group*>
: Рекомендований спосіб виклику процедури NGC із стандартним перетворенням параметра argspec. Використовується, якщо argspec є достатнім. Зверніть увагу, що цього недостатньо для перепризначення кодів зміни інструменту `Tx` та `M6`/`M61`.

prolog=<*pythonprolog*> ngc=<*procname*> epilog=<*pythonepilog*> modalgroup=<*group*>
: Викличте функцію прологу Python, щоб виконати будь-які попередні кроки, а потім викличте процедуру NGC. Після завершення викличте функцію епілогу Python, щоб виконати будь-які роботи з очищення або вилучення результатів, які не можуть бути оброблені в G-коді. Це найгнучкіший спосіб перепризначення коду до процедури NGC, оскільки майже всі внутрішні змінні інтерпретатора та деякі внутрішні функції можуть бути доступні з обробників прологу та епілогу. Крім того, це довша мотузка, на якій ви можете повіситися.

python=<*pythonfunction*> modalgroup=<*group*>
: Безпосередньо викликайте функцію Python без будь-якого перетворення аргументів. Найпотужніш спосіб перемапування коду і переходу безпосередньо до Python. Використовуйте це, якщо вам не потрібна процедура NGC або NGC просто заважає вам.

argspec=<*words*> python=<*pythonfunction*> modalgroup=<*group*>
: Перетворіть слова з argspec та передайте їх функції Python як словник аргументів ключових слів. Використовуйте це, коли вам ліньки самостійно досліджувати слова, передані в блоці.

Зверніть увагу, що якщо все, що ви хочете досягти, це викликати певний код Python з G-коду, існує дещо простіший спосіб виклику функцій Python.

9.6.3.3 Параметр argspec

Специфікація аргументу (ключове слово `argspec`) описує обов'язкові та додаткові слова, які потрібно передати процедурі ngc, а також додаткові передумови для виконання цього коду.

Специфікація аргументів складається з 0 або більше символів класу `[@A-KMNP-Za-kmnp-z^>]`. Вона може бути порожньою (як `argspec=`).

Порожній аргумент argspec або його відсутність взагалі означає, що перепризначений код не отримує жодних параметрів з блоку. Він ігноруватиме будь-які додаткові наявні параметри.

Зверніть увагу, що правила RS274NGC все ще застосовуються — наприклад, ви можете використовувати слова осей (наприклад, `X`, `Y`, `Z`) лише в контексті G-коду.

Слова осей також можна використовувати, лише якщо вісь увімкнена. Якщо увімкнено лише XYZ, ABCUVW не буде доступним для використання в argspec.

Слова «F», «S» і «T» (скорочено «FST») матимуть звичайні функції, але будуть доступні як змінні в перепризначеній функції. «F» встановлюватиме швидкість подачі, «S» встановлюватиме оберти шпинделя, «T» запускатиме функцію підготовки інструменту. Слова «FST» не слід використовувати, якщо така поведінка не є бажаною.

Слова `DEIJKPQR` не мають попередньо визначеної функції та рекомендуються для використання як параметри argspec.

ABCDEFHIJKPQRSTUVWXYZ
: Визначає обов'язковий параметр слова: велика літера вказує, що відповідне слово **повинно** бути присутнім у поточному блоці. Значення слова буде передано як локальний іменований параметр з відповідною назвою. Якщо в argspec присутній символ `@`, він буде переданий як позиційний параметр, див. нижче.

abcdefhijkpqrstuvwxyz
Визначає необов’язковий параметр слова: мала літера вказує, що відповідне слово **може** бути присутнім у поточному блоці. Якщо слово присутнє, його значення буде передано як локальний іменований параметр. Якщо в argspec присутній символ `@`, він буде переданий як позиційний параметр, див. нижче.

@
Символ `@` (знак @) вказує argspec передавати слова як позиційні параметри в порядку, визначеному після опції `@`. Зверніть увагу, що при використанні передачі позиційних параметрів процедура не може визначити, чи було слово присутнє, чи ні, див. приклад нижче.

Tip
це допомагає упакувати існуючі процедури NGC як перемаповані коди. Існуючі процедури очікують позиційних параметрів. За допомогою опції `@` ви можете уникнути їх перезапису для посилання на локальні іменовані параметри.

---

^
Символ ^ (вставка) вказує, що поточна швидкість шпинделя має бути більшою за нуль (шпиндель працює), інакше код завершується невдачею з відповідним повідомленням про помилку.

>
The > (Символ «більше ніж») вказує, що поточна подача має бути більшою за нуль, інакше код завершиться невдало з відповідним повідомленням про помилку.

n
Символ n (більше ніж) вказує на передачу поточного номера рядка в локальному іменованому параметрі n.

За замовчуванням параметри передаються як локальні іменовані параметри до процедури NGC. Ці локальні параметри відображаються як «вже встановлені» під час запуску процедури, що відрізняється від існуючої семантики (локальні змінні починають з значення 0,0 і потребують явного присвоєння значення).

Наявність додаткових параметрів слова можна перевірити за допомогою ідіоми `EXISTS(#<слово>)`.

Приклад передачі іменованого параметра до процедур NGC Припустимо, що код визначено як

```
REMAP=M400 modalgroup=10 argspec=Pq ngc=m400
```

а `m400.ngc` виглядає так:

```
o<m400> sub
(P b''єb'' b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb''b''ob''b''вb''b''иb''b''мb'',  ↩
    b''ob''b''cb''b''кb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''вb'' argspec b''вb''b''ob''b ↩
    ''нb''b''ob'' b''нb''b''ab''b''пb''b''иb''b''cb''b''ab''b''нb''b''ob'' b''вb''b''eb''b' ↩
    'лb''b''иb''b''кb''b''иb''b''мb''b''иb'' b''лb''b''ib''b''тb''b''eb''b''pb''b''ab''b' ↩
    'мb''b''иb'')
(debug, P word=#<P>)
(argspec q b''єb'' b''нb''b''eb''b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb''b''ob''b ↩
    ''вb''b''иb''b''мb'', b''ob''b''cb''b''кb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''вb''  ↩
    argspec b''вb''b''ob''b''нb''b''ob'' b''нb''b''ab''b''пb''b''иb''b''cb''b''ab''b''нb''b' ↩
    'ob'' b''мb''b''ab''b''лb''b''иb''b''мb''b''иb'' b''лb''b''ib''b''тb''b''eb''b''pb''b' ↩
    'ab''b''мb''b''иb''. b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b' ↩
    'вb''b''yb''b''йb''b''тb''b''eb'' b''нb''b''ab''b''cb''b''тb''b''yb''b''пb''b''нb''b' ↩
    'иb''b''мb'' b''чb''b''иb''b''нb''b''ob''b''мb'':)
o100 if [EXISTS[#<q>]]
    (debug, Q word set: #<q>)
o100 endif
o<m400> endsub
M2
```

- Виконання `M400` завершиться невдачею з повідомленням `визначений користувачем M400: відсутні P`.
- Виконання команди `M400 P123` відобразить `P word=123.000000`.
- Виконання команди `M400 P123 Q456` відобразить `P word=123.000000` та `Q word set: 456.000000`.

Приклад передачі позиційних параметрів до процедур NGC Припустимо, що код визначено як

```
REMAP=M410 modalgroup=10 argspec=@PQr ngc=m410
```

а `m410.ngc` виглядає наступним чином:

```
o<m410> sub
(debug, [1]=#1 [2]=#2 [3]=#3)
o<m410> endsub
M2
```

- Виконання команди `M410 P10` відобразить `m410.ngc: [1]=10.000000 [2]=0.000000`.
- Виконання `M410 P10 Q20` will display `m410.ngc: [1]=10.000000 [2]=20.000000`.

Note
ви втрачаєте можливість розрізняти більше одного слова додаткового параметра, і не можете визначити, чи був присутній необов'язковий параметр, але мав значення 0, чи взагалі був відсутній.

Простий приклад передачі іменованого параметра до функції Python Можливо визначити нові коди *без* будь-якої процедури NGC. Ось перший простий приклад, складніший можна знайти в наступному розділі.

Припустимо, що код визначено як

```
REMAP=G88.6 modalgroup=1 argspec=XYZp python=g886
```

Це вказує інтерпретатору виконати функцію Python `g886` у модулі `module_basename.remap`, яка може виглядати так:

```
b''зb'' b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b''pb''b''eb''b''тb''b''ab''b''тb''b' ↩
    'ob''b''pb''b''ab'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' INTERP_OK
b''зb'' emccanon b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' 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
```

Спробуйте це без використання: g88.6 x1 y2 z3 g88.6 x1 y2 z3 p33 (коментар тут)

Ви помітите поступове впровадження вбудованого середовища Python — детальніше див. here. Зверніть увагу, що з функціями перепризначення Python немає сенсу використовувати функції прологу та епілогу Python, оскільки спочатку виконується функція Python.

Розширений приклад: Переназначені коди на чистому Python Модулі `interpreter` та `emccanon` розкривають доступ до більшої частини Інтерпретатора та деяких внутрішніх функцій Canon, тому багато речей, які досі вимагали написання на *C/C+\+*, тепер можна зробити на Python.

Наступний приклад базується на скрипті `nc_files/involute.py`, але записаний у вигляді G-коду з деякими витягненнями та перевірками параметрів. Він також демонструє рекурсивне виклик інтерпретатора (див. `self.execute()`).

Припускаючи таке визначення (Примітка: тут не використовується argspec):

```
REMAP=G88.1 modalgroup=1 py=involute
```

Функція `involute` у файлі `python/remap.py`, наведена нижче, виконує вилучення всіх слів безпосередн з поточного блоку. Зверніть увагу, що помилки інтерпретатора можуть бути перетворені на винятки Python. Пам'ятайте, що це «час попереднього читання» — помилки часу виконання не можуть бути перехоплені таким чином.

```
import sys
import traceback
b''зb'' b''мb''b''аb''b''тb''b''еb''b''мb''b''аb''b''тb''b''иb''b''чb''b''нb''b''оb''b' ↩
    'гb''b''оb'' b''іb''b''мb''b''пb''b''оb''b''рb''b''тb''b''уb'' sin,cos

from interpreter import *
from emccanon import MESSAGE
from util import lineno, call_pydevd
# b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''єb'' InterpreterException, b''яb''b' ↩
    'кb''b''щb''b''оb'' execute() b''аb''b''бb''b''оb'' read() b''зb''b''аb''b''вb''b''еb''b ↩
    ''рb''b''шb''b''уb''b''єb''b''тb''b''ьb''b''сb''b''яb'' b''нb''b''еb''b''вb''b''дb''b' ↩
    'аb''b''чb''b''еb''b''юb''
throw_exceptions = 1

def involute(self, **words):
""" b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''яb'' b''пb''b''еb''b''рb''b''еb''b''пb''b' ↩
    'рb''b''иb''b''зb''b''нb''b''аb''b''чb''b''еb''b''нb''b''нb''b''яb'' b''зb'' b''нb''b' ↩
    'еb''b''оb''b''бb''b''рb''b''оb''b''бb''b''лb''b''еb''b''нb''b''иb''b'мb'' b''дb''b' ↩
    'оb''b''сb''b''тb''b''уb''b''пb''b''оb''b''мb'' b''дb''b''оb'' b''вb''b''нb''b''уb''b' ↩
    'тb''b''рb''b''іb''b''шb''b''нb''b''іb''b''хb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
    'іb''b''йb'' b''іb''b''нb''b''тb''b''еb''b''рb''b''пb''b''рb''b''еb''b''тb''b''аb''b' ↩
    'тb''b''оb''b''рb''b''аb'' """

    if self.debugmask & 0x20000000: call_pydevd() # b''Пb''b''рb''b''аb''b''пb''b''оb''b' ↩
        'рb''b''еb''b''цb''b''ьb'' b''нb''b''аb''b''лb''b''аb''b''гb''b''оb''b''дb''b''жb''b ↩
        ''еb''b''нb''b''нb''b''яb'' USER2

    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 # b''яb''b''кb''b''щb''b''оb'' b''бb''b''уb''b''лb''b''оb'' b''зb''b''аb'' ↩
        b''дb''b''аb''b''нb''b''оb'' b''сb''b''лb''b''оb''b''вb''b''оb'' Z, b''зb''b''аb''b' ↩
        'нb''b''уb''b''рb''b''еb''b''нb''b''нb''b''яb'' - b''зb''b''іb'' b''зb''b''мb''b' ↩
        'еb''b''нb''b''шb''b''еb''b''нb''b''оb''b''юb'' b''пb''b''оb''b''дb''b''аb''b''чb''b ↩
        ''еb''b''юb''

    # b''пb''b''еb''b''рb''b''еb''b''вb''b''іb''b''рb''b''иb''b''тb''b''иb'' b''кb''b''оb'' ↩
        b''нb''b''тb''b''рb''b''оb''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''бb''b''лb''b' ↩
        'оb''b''кb'' b''нb''b''аb'' b''нb''b''аb''b''яb''b''вb''b''нb''b''іb''b''сb''b''тb'' ↩
        b''ьb'' b''вb''b''іb''b''дb''b''пb''b''оb''b''вb''b''іb''b''дb''b''нb''b''иb''b' ↩
        'хb'' b''сb''b''лb''b''іb''b''вb''
    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")  # b''вb''b''иb''b''кb''b''лb''b''иb''b''чb''b''eb'' b''вb'' ↩
            b''иb''b''нb''b''яb''b''тb''b''оb''b''кb'' 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: # b''пb''b''оb''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb'' b' ↩
            'нb''b''ab'' b''пb''b''оb''b''чb''b''ab''b''тb''b''кb''b''оb''b''вb''b''yb'' b' ↩
            'вb''b''иb''b''cb''b''оb''b''тb''b''yb''
            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
```

Приклади, описані досі, можна знайти у файлі *configs/sim/axis/remap/getting-started* з повними робочими конфігураціями.

9.6.4 Оновлення існуючої конфігурації для перепризначення

Мінімальні передумови для використання операторів REMAP такі:

- Плагін Python потрібно активувати, вказавши `[PYTHON]TOPLEVEL=<шлях-до-скрипта-топловня>` у INI-файлі.
- Скрипт верхнього рівня має імпортувати модуль `remap`, який спочатку може бути порожнім, але імпорт має бути виконаний.
- Інтерпретатор Python повинен знайти модуль remap.py вище, тому шлях до каталогу, де знаходяться ваші модулі Python, потрібно додати за допомогою `[PYTHON]PATH_APPEND=<path-to-your-local-Pyth`
- Рекомендується: імпортувати обробники `stdglue` у модуль `remap`. У цьому випадку Python також повинен знайти файл `stdglue.py` — ми просто копіюємо його з дистрибутива, щоб ви могли внести необхідні локальні зміни. Залежно від вашої інсталяції шлях до файлу `stdglue.py` може відрізнятися.

Припускаючи, що ваша конфігурація знаходиться в `/home/user/xxx`, а INI-файл — `/home/user/xxx/xxx` виконайте наступні команди.

```
$ 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
```

Тепер відредагуйте ``/home/user/`*xxx*``/`*xxx*``.ini`` та додайте наступне:

```
[PYTHON]
TOPLEVEL=/home/user/xxx/python/toplevel.py
PATH_APPEND=/home/user/xxx/python
```

Тепер перевірте, чи LinuxCNC не видає жодних повідомлень про помилки — у вікні терміналу виконайте:

```
$ cd /home/user/xxx
$ linuxcnc xxx.ini
```

9.6.5 Коди, пов'язані зі зміною інструменту переналаштування: T, M6, M61

9.6.5.1 Огляд

Якщо ви не знайомі з внутрішніми механізмами LinuxCNC, спочатку прочитайте розділ Як зараз працює зміна інструменту (важко, але необхідно).

Зверніть увагу, що під час перепризначення існуючого коду ми повністю вимикаємо вбудовану функціональність цього коду інтерпретатора.

Отже, наш перероблений код повинен буде робити дещо більше, ніж просто генерувати команди для переміщення машини так, як нам потрібно — він також повинен буде повторювати ті кроки з цієї послідовності, які необхідні для забезпечення нормальної роботи інтерпретатора та `task`.

Однак це **не** впливає на обробку команд, пов'язаних із заміною інструменту, в `task` та `iocontrol`. Це означає, що коли ми виконуємо step 6b, це все одно призведе до того, що iocontrol виконає свою дію.

Рішення, рішення:

- Чи хочемо ми використовувати процедуру на літеру O, чи робити все це кодом на Python?
- Чи достатньо послідовності HAL «iocontrol» (контакти tool-prepare/tool-prepared та tool-change/tool-changed), чи нам потрібна інша взаємодія HAL для нашого пристрою зміни інструментів (наприклад: більше контактів HAL з іншою послідовністю взаємодії)?

Залежно від відповіді, ми маємо чотири різні сценарії:

- Під час використання процедури на літеру O-слово нам потрібні функції прологу та епілогу.
- Якщо використовується лише код Python і жодної процедури на O-слові, достатньо функції Python.
- Під час використання виводів `iocontrol` наша процедура на O-слові або код Python міститиме здебільшого переміщення.
- Коли нам потрібна більш складна взаємодія, ніж та, що пропонує `iocontrol`, ми повинні повністю визначити нашу власну взаємодію, використовуючи контакти `motion.digital*` та `motion.analog*`, і, по суті, ігнорувати контакти `iocontrol`, зациклюючи їх.

Note
Якщо ви не любите процедури O-word і любите Python, ви можете зробити все в Python, і в цьому випадку вам достатньо буде вказати python=<*function*> spec в операторі REMAP. Але, припускаючи, що більшість людей будуть зацікавлені у використанні процедур O-word, оскільки вони більш звичні для них, ми зробимо це першим прикладом.

Отже, загальний підхід для нашого першого прикладу буде таким:

1. Ми хотіли б зробити якомога більше за допомогою G-коду в процедурі O-word для більшої гнучкості. Це включає всю взаємодію HAL, яка зазвичай обробляється за допомогою `iocontrol`, оскільки ми хотіли б робити розумні речі з переміщеннями, зондами, входами/виходами HAL і так далі.
2. Ми спробуємо мінімізувати код Python настільки, наскільки це необхідно, щоб інтерпретатор був задоволений, і змусити `task` виконувати будь-які дії. Це буде передано функціям Python `prolog` та `epilog`.

9.6.5.2 Розуміння ролі `iocontrol` з перепризначеними кодами зміни інструменту

`iocontrol` надає дві послідовності взаємодії HAL, які ми можемо використовувати, а можемо й не використовувати:

- Коли виконується повідомлення NML, поставлене в чергу командою SELECT_TOOL() canon, це запускає послідовність HAL «підняти tool-prepare і чекати, поки tool-prepared не стане високим» в `iocontrol`, крім встановлення контактів *XXXX*
- Коли виконується повідомлення NML, поставлене в чергу командою CHANGE_TOOL(), це запускає послідовність HAL «підняти зміну інструменту і чекати, поки інструмент не зміниться на високий» в `iocontrol`, крім установки контактів *XXXX*

Вам потрібно вирішити, чи достатньо існуючих послідовностей HAL `iocontrol` для керування вашим перетворювачем. Можливо, вам потрібна інша послідовність взаємодії — наприклад, більше контактів HAL або, можливо, більш складна взаємодія. Залежно від відповіді, ми можемо продовжувати використовувати існуючі послідовності HAL `iocontrol` або визначити власні.

Для документації ми вимкнемо ці послідовності `iocontrol` і створимо власні — результат буде виглядати і відчуватися як існуюча взаємодія, але тепер ми маємо повний контроль над ними, оскільки вони виконуються в нашій власній процедурі O-word.

Отже, ми будемо використовувати деякі контакти `motion.digital-*` та `motion.analog-*`, а також відповідні команди `M62 .. M68` для взаємодії з HAL у нашій процедурі O-word, і вони ефективно замінять послідовності `iocontrol` *tool-prepare/tool-prepared* та *tool-change/tool-changed*. Отже, ми визначимо наші контакти, які функціонально замінять існуючі контакти `iocontrol`, і продовжимо, зробивши взаємодію `iocontrol` циклом. У нашому прикладі ми будемо використовувати таку відповідність:

`iocontrol` відповідність контактів у прикладах

iocontrol.0 шпилька	**motion шпилька**
tool-prepare	digital-out-00
підготовлений інструмент	digital-in-00
tool-change	digital-out-01
зміна інструменту	digital-in-01
tool-prep-number	analog-out-00
tool-prep-pocket	analog-out-01

iocontrol.0 шпилька	**motion шпилька**
tool-number	analog-out-02

Припустимо, ви хочете перевизначити команду M6 та замінити її процедурою з літерою O, але в іншому все «має продовжувати працювати».

Отже, наша процедура O-word замінить кроки описані тут. Переглянувши ці кроки, ви побачите, що код NGC можна використовувати для більшості з них, але не для всіх. Тому те, що NGC не може обробити, буде виконано в пролозі та епілозі Python.

9.6.5.3 Визначення заміни M6

Щоб передати ідею, ми просто замінюємо вбудовану семантику M6 нашою власною. Як тільки це спрацює, ви можете продовжувати та розміщувати будь-які дії, які вважаєте за потрібне, у процедурі O-word.

Виконуючи кроки Дія інтерпретатора з командою M6, ми знаходимо:

1. перевірити, чи вже виконано команду T - **виконати в пролозі Python**
2. перевірити, чи активна компенсація різця - **виконати у пролозі Python**
3. зупинити шпиндель, якщо потрібно - **можна зробити в NGC**
4. перо вгору - **можна зробити в NGC**
5. якщо було встановлено TOOL_CHANGE_AT_G30:
   a. перемістити індексатори A, B та C, якщо це можливо - **можна зробити в NGC**
   b. генерувати швидкий рух до позиції G30 - **можна виконати в NGC**
6. надіслати команду CHANGE_TOOL Canon до `task` - **виконати в епілозі Python**
7. встановіть пронумеровані параметри 5400-5413 відповідно до нового інструменту - **виконайте в епілозі Python**
8. сигнал до `task` про припинення виклику інтерпретатора для попереднього зчитування до завершення зміни інструменту - **виконати в епілозі Python**

Отже, нам потрібен пролог та епілог. Припустимо, що наш INI-файл з інкантацією перепризначення M6 виглядає так:

```
REMAP=M6   modalgroup=6  prolog=change_prolog ngc=change epilog=change_epilog
```

Отже, пролог, що охоплює кроки 1 і 2, виглядатиме так: ми вирішуємо передати кілька змінних до процедури перемапування, де їх можна перевірити та змінити або використати в повідомленні. Це такі змінні: `tool_in_spindle`, `selected_tool` (номери інструментів) та відповідні індекси даних інструментів `current_pocket` і `selected_pocket`:

Note
Старі імена **selected_pocket** та **current_pocket** насправді посилаються на послідовний індекс tooldata для елементів інструменту, завантажених із таблиці інструментів ([EMCIO]TOOL_TABLE) або через базу даних tooldata ([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 "b''Нb''b''eb''b''мb''b''ob''b''жb''b''лb''b''иb''b''вb''b''ob'' b''зb'' ↩
                b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''ib''b''нb''b''cb''b''тb''b' ↩
                'pb''b''уb''b''мb''b''eb''b''нb''b''тb''b''иb'', b''яb''b''кb''b''щb''b' ↩
                'ob'' b''вb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob'' b' ↩
                'кb''b''ob''b''мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''юb'' ↩
                b''pb''b''ab''b''дb''b''ib''b''уb''b''cb''b''ab'' b''pb''b''ib''b''зb''b' ↩
                'цb''b''яb''"

        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)
```

Ви побачите, що більшість функцій прологу виглядають дуже схожими:

1. Спочатку перевірте, чи виконуються всі передумови для виконання коду, потім
2. підготувати середовище — ввести змінні та/або виконати будь-які підготовчі кроки обробки, які неможливо легко виконати в коді NGC;
3. потім передайте до процедури NGC, повернувши INTERP_OK.

Наша перша ітерація процедури O-word не є цікавою - просто перевірте, чи правильно ми вказали параметри, і сигналізуйте про успіх, повертаючи позитивне значення; кроки 3-5 будуть зрештою висвітлені тут (див. here для змінних, що відносяться до налаштувань файлу INI):

```
O<change> sub
(b''нb''b''ab''b''лb''b''ab''b''rb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'зb''b''мb''b''ib''b''нb''b''ab'': b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''иb''b' ↩
    'йb''_b''ib''b''нb''b''cb''b''тb''b''pb''b''уb''b''мb''b''eb''b''нb''b''тb''=#<b''пb''b' ↩
    'ob''b''тb''b''ob''b''чb''b''нb''b''иb''b''йb''_b''ib''b''нb''b''cb''b''тb''b''pb''b' ↩
    'уb''b''мb''b''eb''b''нb''b''тb''>)
(b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'зb''b''мb''b''ib''b''нb''b''ab'': b''вb''b''иb''b''бb''b''pb''b''ab''b''нb''b''ab''_b' ↩
    'кb''b''иb''b''шb''b''eb''b''нb''b''яb''=#<b''вb''b''иb''b''бb''b''pb''b''ab''b''нb''b' ↩
    'ab''_b''кb''b''иb''b''шb''b''eb''b''нb''b''яb''>)
;
; b''вb''b''cb''b''тb''b''ab''b''вb''b''тb''b''eb'' b''cb''b''юb''b''дb''b''иb'' b''бb''b' ↩
    'уb''b''дb''b''ьb''-b''яb''b''кb''b''иb''b''йb'' G-b''кb''b''ob''b''дb'', b''яb''b''кb'' ↩
    b''иb''b''йb'' b''вb''b''вb''b''ab''b''жb''b''ab''b''єb''b''тb''b''eb'' b''зb''b''ab'' b ↩
    ''пb''b''ob''b''тb''b''pb''b''ib''b''бb''b''нb''b''eb'', b''нb''b''ab''b''пb''b''pb''b' ↩
    'иb''b''кb''b''лb''b''ab''b''дb'':
; G0  #<_ini[setup]tc_x>  #<_ini[setup]tc_y>  #<_ini[setup]tc_z>
;
O<change> endsub [1]
m2
```

Якщо припустити успішне виконання `change.ngc`, нам потрібно виконати кроки 6-8:

```
def change_epilog(self, **words):
    try:
        if self.return_value > 0.0:
```

```
            # b''зb''b''ab''b''фb''b''ib''b''кb''b''cb''b''yb''b''вb''b''ab''b''тb''b''иb'' ↩
                 b''зb''b''мb''b''ib''b''нb''b''иb''
            self.selected_pocket =  int(self.params["selected_pocket"])
            emccanon.CHANGE_TOOL(self.selected_pocket)
            # b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''тb''b''иb'' b''cb''b' ↩
                'иb''b''нb''b''xb''b''pb''b''ob''b''нb''b''ib''b''зb''b''ab''b''цb''b''ib''b ↩
                ''юb''()
            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)
```

Цей замінник M6 сумісний із вбудованим кодом, за винятком того, що кроки 3-5 потрібно заповнити вашим кодом NGC.

Знову ж таки, більшість епілогів мають спільну схему:

1. Спочатку визначте, чи все пройшло правильно під час процедури перепризначення,
2. потім виконайте будь-які дії з фіксації та очищення, які неможливо виконати в коді NGC.

9.6.5.4 Налаштування `iocontrol` з перепризначеним M6

Зверніть увагу, що послідовність операцій змінилася: ми виконуємо все необхідне в процедурі O-word, включаючи будь-яке налаштування/зчитування контактів HAL для запуску змінювача та підтвердження зміни інструменту, ймовірно, за допомогою контактів вводу-виводу `motion.digital-*` та `motion-analog-*`. Коли ми нарешті виконуємо команду `CHANGE_TOOL()`, всі рухи та взаємодії HAL вже завершені.

Зазвичай тільки тепер `iocontrol` виконує свою роботу, як описано в here. Однак нам більше не потрібно, щоб HAL коливався — все, що залишається зробити `iocontrol`, це прийняти, що ми закінчили підготовку та зміни.

Це означає, що відповідні виводи `iocontrol` більше не мають функцій. Тому ми налаштовуємо `iocontrol` на негайне підтвердження змін, виконавши такі дії:

```
# b''cb''b''иb''b''гb''b''нb''b''ab''b''лb''b''иb'' b''зb''b''мb''b''ib''b''нb''b''иb'' b' ↩
    'цb''b''иb''b''кb''b''лb''b''yb'' b''пb''b''ib''b''дb'' b''чb''b''ab''b''cb'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb'' M6
net tool-change-loop iocontrol.0.tool-change iocontrol.0.tool-changed
```

Якщо ви з якоїсь причини хочете перепризначити Tx (підготувати), відповідні виводи `iocontrol` також потрібно зациклити.

9.6.5.5 Написання змін та підготовка процедур, що передбачають букву «O»

Стандартні прологи та епілоги, що знаходяться у файлі `ncfiles/remap_lib/python-stdglue/stdglue.p` передають кілька «відкритих параметрів» процедурі перепризначення.

«Відкритий параметр» — це іменована локальна змінна, видима в процедурі перемапування, яка відповідає внутрішній змінній інтерпретатора, що має значення для поточного перемапування. Відкриті параметри налаштовуються у відповідному пролозі та перевіряються в епілозі. Їх можна змінювати в процедурі перемапування, і ці зміни будуть відображені в епілозі. Відкриті параметри для вбудованих кодів, що підлягають перемапуванню, такі:

- T (prepare_prolog): #<tool> , #<pocket>
- M6 (change_prolog): #<інструмент_в_шпинделі>, #<вибраний_інструмент>, #<поточна_кишеня>, #<вибрана_кишеня>
- M61 (settool_prolog): #<tool> , #<pocket>
- S (setspeed_prolog): #<speed>
- F (setfeed_prolog): #<feed>

Якщо у вас є конкретні потреби в додаткових параметрах, які будуть видимими, їх можна просто додати до прологу — практично всі внутрішні функції інтерпретатора видимі для Python.

9.6.5.6 Внесення мінімальних змін до вбудованих кодів, включаючи M6

Пам’ятайте, що зазвичай перепризначення коду повністю вимикає всю внутрішню обробку для цього коду.

Однак у деяких ситуаціях може бути достатньо додати кілька кодів навколо існуючої вбудованої реалізації M6, наприклад, зонд довжини інструменту, але крім цього зберегти поведінку вбудованої функції M6.

Оскільки це може бути типовим сценарієм, вбудована поведінка перепризначених кодів стала доступною в процедурі перепризначення. Інтерпретатор виявляє, що ви посилаєтеся на перепризначе код у процедурі, яка повинна перевизначити його поведінку. У цьому випадку використовується вбудована поведінка — наразі вона ввімкнена для набору: M6, M61,T, S, F. Зверніть увагу, що в іншому випадку посилання на код у власній процедурі перепризначення буде помилкою — рекурсією перепризначення.

Легке повертання вбудованого елемента виглядатиме ось так (у випадку M6):

```
REMAP=M6   modalgroup=6  ngc=mychange
```

```
o<mychange> sub
M6 (use built in M6 behavior)
(.. b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’йb’’b’’дb’’b’’іb’’b’’тb’’b’’ьb’’ b’’дb’’b’’ob’’ b’’пb’’b ↩
    ’’eb’’b’’pb’’b’’eb’’b’’мb’’b’’иb’’b’’кb’’b’’ab’’b’’чb’’b’’ab’’ b’’дb’’b’’ob’’b’’вb’’b’ ↩
    ’жb’’b’’иb’’b’’нb’’b’’иb’’ b’’іb’’b’’нb’’b’’cb’’b’’тb’’b’’pb’’b’’yb’’b’’мb’’b’’eb’’b’ ↩
    ’нb’’b’’тb’’b’’ab’’, b’’вb’’b’’иb’’b’’мb’’b’’іb’’b’’pb’’b’’яb’’b’’йb’’b’’тb’’b’’eb’’ b’ ↩
    ’дb’’b’’ob’’b’’вb’’b’’жb’’b’’иb’’b’’нb’’b’’yb’’ b’’іb’’b’’нb’’b’’cb’’b’’тb’’b’’pb’’b’ ↩
    ’yb’’b’’мb’’b’’eb’’b’’нb’’b’’тb’’b’’ab’’ b’’тb’’b’’ab’’ b’’вb’’b’’cb’’b’’тb’’b’’ab’’b’ ↩
    ’нb’’b’’ob’’b’’вb’’b’’іb’’b’’тb’’b’’ьb’’ b’’їb’’b’’їb’’..)
o<mychange> endsub
m2
```



Caution
Під час перевизначення вбудованого коду **не вказуйте жодних початкових нулів у G- або M-кодах** — наприклад, скажімо REMAP=M1 .., а не REMAP=M01 ....

Дивіться каталог configs/sim/axis/remap/extend-builtins для отримання повної конфігурації, яка є рекомендованою відправною точкою для власної роботи з розширення вбудованого коду.

9.6.5.7 Визначення заміни T (підготовка)

Якщо ви впевнені в default implementation, вам не потрібно це робити. Але перепризначення також є способом обійти недоліки поточної реалізації, наприклад, щоб не блокувати доти, доки не буде встановлено контакт «tool-prepared».

Наприклад, що ви могли б зробити: - У перепризначеному T просто встановіть еквівалент виводу `tool-prepare`, але **не** чекайте тут на `tool-prepared`. - У відповідному перепризначеному M6 зачекайте на `tool-prepared` на самому початку процедури O-word.

Знову ж таки, контакти `iocontrol` tool-prepare/tool-prepared не використовуватимуться та будуть замінені контактами `motion.*`, тому ці контакти потрібно буде зациклити:

```
# b''cb''b''ив''b''rb''b''нb''b''ab''b''лb''b''ив'' b''пb''b''ib''b''дb''b''rb''b''ob''b' ↩
    'тb''b''ob''b''вb''b''кb''b''ив'' b''цb''b''ив''b''кb''b''лb''b''yb'' b''пb''b''ib''b' ↩
    'дb'' b''чb''b''ab''b''cb'' b''пb''b''eb''b''pb''b''eb''b''пb''b''pb''b''ив''b''зb''b' ↩
    'нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' T
net tool-prep-loop iocontrol.0.tool-prepare iocontrol.0.tool-prepared
```

Отже, ось налаштування для перепризначеного 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

        # b''цb''b''ib'' b''зb''b''мb''b''ib''b''нb''b''нb''b''ib'' b''бb''b''yb''b''дb''b' ↩
            'yb''b''тb''b''ьb'' b''вb''b''ив''b''дb''b''ив''b''мb''b''ib'' b''вb'' b''пb''b' ↩
            'ib''b''дb''b''cb''b''лb''b''ob''b''вb''b''ib'' ngc O-word
        # b''яb''b''кb'' b''лb''b''ob''b''кb''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''зb''b' ↩
            'мb''b''ib''b''нb''b''нb''b''ib'' #<b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b' ↩
            'мb''b''eb''b''нb''b''тb''> b''тb''b''ab'' #<b''кb''b''ив''b''шb''b''eb''b''нb'' ↩
            b''яb''>, b''ib'' b''мb''b''ob''b''жb''b''yb''b''тb''b''ьb'' b''бb''b''yb''b' ↩
            'тb''b''ив''
        # b''зb''b''мb''b''ib''b''нb''b''eb''b''нb''b''ob'' b''тb''b''ab''b''мb'' - b''eb'' ↩
            b''пb''b''ib''b''лb''b''ob''b''rb'' b''ob''b''тb''b''pb''b''ив''b''мb''b''ab''b' ↩
            'єb'' b''зb''b''мb''b''ib''b''нb''b''eb''b''нb''b''ib'' b''дb''b''ab''b''нb''b' ↩
            'ib''
        # b''цb''b''ib''b''нb''b''нb''b''ob''b''cb''b''тb''b''ib''
        self.params["tool"] = tool
        self.params["pocket"] = pocket

        return INTERP_OK
    except Exception, e:
        return "T%d/prepare_prolog: %s" % (int(words['t']), e)
```

Мінімальна процедура підготовки ngc знову виглядає так:

```
o<prepare> sub
```

```
; b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ob''b' ↩
    'зb''b''иb''b''тb''b''иb''b''вb''b''нb''b''ob''b''гb''b''ob'' b''зb''b''нb''b''ab''b' ↩
    'чb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''фb''b''іb''b''кb''b''cb''b' ↩
    'ab''b''цb''b''ib''b''їb'':
o<prepare> endsub [1]
m2
```

І епілог:

```
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)
```

Функції «prepare_prolog» та «prepare_epilog» є частиною стандартного модуля «nc_files/remap_lib/pytho
stdglue/stdglue.py». Цей модуль призначений для загального вирішення більшості стандартних ситуацій перепризначення.

9.6.5.8 Обробка помилок: обробка переривання

Вбудована процедура зміни інструменту має деякі запобіжні заходи на випадок переривання програми, наприклад, натискання клавіші Escape в AXIS під час зміни. Ваша перепризначена функція не має нічого подібного, тому може знадобитися явне очищення, якщо перепризначений код буде перервано. Зокрема, процедура перепризначення може встановити модальні налаштування, які не бажано залишати активними після переривання. Наприклад, якщо ваша процедура перепризна має коди руху (G0, G1, G38..) і перепризначення переривається, то останній модальний код залишиться активним. Однак, швидше за все, ви захочете, щоб будь-який модальний рух був скасований при перериванні перепризначення.

Для цього використовується функція `[RS274NGC]ON_ABORT_COMMAND`. Ця опція INI визначає виклик процедури O-word, яка виконується, якщо `task` з якоїсь причини переріває виконання програми. `on_abort` отримує один параметр, що вказує причину виклику процедури переривання, який може бути використаний для умовного очищення.

Причини визначено у файлі 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,        // b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b' ↩
    'pb''b''eb''b''тb''b''ab''b''тb''b''ob''b''pb'' b''нb''b''eb'' b''вb''b''дb''b''ab''b' ↩
    'вb''b''cb''b''яb'' b''пb''b''іb''b''дb'' b''чb''b''ab''b''cb'' b''пb''b''ob''b''пb''b' ↩
    'eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob''b''гb''b''ob'' b''зb''b''чb''b''иb''b' ↩
    'тb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''
EMC_ABORT_INTERPRETER_ERROR_MDI = 10, // b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b''pb''b ↩
    ''eb''b''тb''b''ab''b''тb''b''ob''b''pb'' b''нb''b''eb'' b''вb''b''дb''b''ab''b''лb''b' ↩
    'ob''b''cb''b''яb'' b''вb''b''иb''b''кb''b''ob''b''нb''b''ab''b''тb''b''иb'' MDI
```

```
EMC_ABORT_USER = 100  // b''кb''b''ob''b''дb''b''иb'' b''пb''b''eb''b''pb''b''eb''b''pb''b' ↩
    'иb''b''вb''b''ab''b''нb''b''нb''b''яb'', b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''ib'' b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b''ab''b' ↩
    'чb''b''eb''b''мb'', b''пb''b''ob''b''чb''b''иb''b''нb''b''ab''b''юb''b''тb''b''ьb''b' ↩
    'cb''b''яb'' b''тb''b''уb''b''тb''
```

```
[RS274NGC]
ON_ABORT_COMMAND=O <on_abort> call
```

Запропонована процедура on_abort виглядатиме так (адаптуйте до своїх потреб):

```
o<on_abort> sub

G54 (b''зb''b''cb''b''уb''b''вb''b''иb'' b''вb''b''ib''b''дb'' b''пb''b''ob''b''чb''b''ab'' ↩
    b''тb''b''кb''b''уb'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' b' ↩
    'вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''ib'' b''зb''b' ↩
    'ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b' ↩
    'яb''b''мb'')
G17 (b''вb''b''иb''b''6b''b''ib''b''pb'' b''пb''b''лb''b''ob''b''щb''b''иb''b''нb''b''иb''  ↩
    XY)
G90 (b''ab''b''6b''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'')
G94 (b''pb''b''eb''b''жb''b''иb''b''мb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'': b' ↩
    'ob''b''дb''b''иb''b''нb''b''иb''b''цb''b''ib''/b''хb''b''вb''b''иb''b''лb''b''иb''b' ↩
    'нb''b''ab'')
M48 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b ↩
    ''пb''b''eb''b''pb''b''eb''b''вb''b''иb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''пb''b' ↩
    'ob''b''дb''b''ab''b''чb''b''ib'' b''тb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'ob''b''cb''b''тb''b''ib'')
G40 (b''кb''b''ob''b''мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''яb'' b''pb''b ↩
    ''ib''b''зb''b''ab''b''кb''b''ab'' b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b' ↩
    'ab'')
M5  (b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''ьb'' b''вb''b''иb''b''мb''b''кb''b ↩
    ''нb''b''eb''b''нb''b''ob'')
G80 (b''cb''b''кb''b''ab''b''cb''b''уb''b''вb''b''ab''b''тb''b''иb'' b''мb''b''ob''b''дb''b ↩
    ''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''pb''b''уb''b''хb'')
M9  (b''тb''b''уb''b''мb''b''ab''b''нb'' b''ib'' b''ob''b''хb''b''ob''b''лb''b''ob''b''дb'' ↩
    b''жb''b''уb''b''юb''b''чb''b''ab'' b''pb''b''ib''b''дb''b''иb''b''нb''b''ab'' b''вb''b' ↩
    'иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''ib'')

o100 b''яb''b''кb''b''щb''b''ob'' [#1 b''дb''b''ob''b''pb''b''ib''b''вb''b''нb''b''юb''b' ↩
    'єb'' 5]
    (b''мb''b''ab''b''шb''b''иb''b''нb''b''ab'' b''уb''b''вb''b''ib''b''мb''b''кb''b''нb''b ↩
        ''eb''b''нb''b''ab'')
o100 b''ib''b''нb''b''ab''b''кb''b''шb''b''eb'' b''яb''b''кb''b''щb''b''ob'' [#1 b''дb''b' ↩
    'ob''b''pb''b''ib''b''вb''b''нb''b''юb''b''єb'' 6]
    (b''мb''b''ab''b''шb''b''иb''b''нb''b''ab'' b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b ↩
        ''нb''b''ab'')
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
Ніколи не використовуйте M2 у підпрограмі O-word, включаючи цю. Це призведе до важко виявляємих помилок. Наприклад, використання M2 у підпрограмі не призведе до правильного завершення підпрограми і залишить відкритим файл NGC підпрограми, а не вашу основну програму.

Переконайтеся, що on_abort.ngc знаходиться вздовж шляху пошуку інтерпретатора (рекомендоване розташування: SUBROUTINE_PATH, щоб не захаращувати каталог NC_FILES внутрішніми процедурами).

Оператори в цій процедурі зазвичай гарантують, що будь-який стан після переривання був очищений, наприклад, правильне скидання виводів HAL. Див. приклад. configs/sim/axis/remap/rack-toolchang

Зверніть увагу, що завершення перепризначеного коду шляхом повернення INTERP_ERROR з епілогу (див. попередній розділ) також призведе до виклику процедури on_abort.

9.6.5.9 Обробка помилок: невдача процедури NGC перепризначеного коду

Якщо у вашій процедурі обробника ви визначили, що сталася якась помилка, не використовуйте M2 для завершення обробника - див. вище:

Якщо достатньо відобразити повідомлення про помилку оператора та зупинити поточну програму, використовуйте функцію (abort, `__<message>__), щоб завершити обробник із повідомленням про помилку. Зверніть увагу, що ви можете замінити пронумеровані, іменовані, INI та HAL параметри в тексті, як у цьому прикладі (див. також tests/interp/abort-hot-comment/test.n

```
o100 if [..] (b''пb''b''eb''b''вb''b''нb''b''ab'' b''yb''b''мb''b''ob''b''вb''b''ab'' b' ↩
    'пb''b''ob''b''мb''b''иb''b''лb''b''кb''b''иb'')
     (abort, Bad Things! p42=#42 q=#<q> INI=#<_ini[a]x> pin=#<_hal[component.pin])
o100 endif
```

Note
Розширення змінних INI та HAL є необов'язковим і може бути вимкнено у файлі INI

Якщо потрібні більш детальні дії з відновлення, використовуйте ідіому, викладену в попередньому прикладі:

- Визначте функцію епілогу, навіть якщо вона просто сигналізує про помилку,
- передати від'ємне значення з обробника, щоб сигналізувати про помилку,
- перевірити значення, що повертається у функції епілогу,
- вжити будь-яких необхідних заходів щодо відновлення,
- повернути рядок повідомлення про помилку з обробника, який встановить повідомлення про помилку інтерпретатора та перерве програму (майже як abort, message=).

Це повідомлення про помилку буде відображено в інтерфейсі користувача, а повернення INTERP_ERROR призведе до обробки цієї помилки як будь-якої іншої помилки під час виконання.

Зверніть увагу, що як (abort, <*msg*>), так і повернення INTERP_ERROR з епілогу призведуть до виклику будь-якого обробника ON_ABORT, якщо він визначений (див. попередній розділ).

9.6.6 Перепризначення інших існуючих кодів:

9.6.6.1 Автоматичний вибір передачі при перепрограмуванні S (встановлення швидкості шпинделя)

Потенційним застосуванням перепрограмованого коду S може бути «автоматичний вибір передачі» залежно від швидкості. У процесі перепрограмування перевіряється, чи можна досягти бажаної швидкості з урахуванням поточного положення передачі, і, якщо ні, то передача змінюється відповідним чином.

9.6.6.2 Налаштування поведінки M0, M1

Прикладом використання перепризначення M0/M1 може бути налаштування поведінки існуючого коду. Наприклад, може бути бажано вимкнути шпиндель, туман і заливку під час паузи програми M0 або M1, а потім знову увімкнути ці налаштування після відновлення програми.

Повний приклад саме цього дивіться у `configs/sim/axis/remap/extend-builtins/`, де M1 адаптується як зазначено вище.

9.6.6.3 Налаштування поведінки M7, M8, M9

Прикладом перепризначення вбудованої поведінки M7/M8/M9 є можливість передачі додаткових аргументів, таких як P-слово, для складнішого керування охолоджуючою рідиною (наприклад, через інструмент проти зовнішнього потоку охолоджуючої рідини).

Див. `configs/sim/axis/remap/extend-builtins/` для прикладу такого розширення вбудованої поведінки для M7, M8 та M9.

9.6.7 Створення нових циклів G-коду

Цикл G-коду, як він використовується тут, має працювати наступним чином:

- Під час першого виклику збираються пов’язані слова та виконується цикл G-коду.
- Якщо наступні рядки просто продовжують слова параметрів, що застосовуються до цього коду, але не додають нового G-коду, попередній G-код виконується повторно з відповідно зміненими параметрами.

Приклад: Припустимо, що у вас є `G84.3`, визначений як перепризначений цикл G-коду з наступним сегментом INI (див. тут для детального опису `cycle_prolog` та `cycle_epilog`):

```
[RS274NGC]
# b''Цb''b''иb''b''кb''b''лb'' b''зb'' b''пb''b''рb''b''оb''b''цb''b''еb''b''дb''b''уb''b' ↩
    'рb''b''оb''b''юb'' b''нb''b''аb'' b''бb''b''уb''b''кb''b''вb''b''уb'' b''Оb'': G84.3 <X ↩
    - Y- Z- Q- P->
REMAP=G84.3 argspec=xyzabcuvwpr prolog=cycle_prolog ngc=g843 epilog=cycle_epilog modalgroup ↩
    =1
```

Виконання наступних рядків:

```
g17
(1)   g84.3 x1 y2 z3  r1
(2)   x3 y4 p2
(3)   x6 y7 z5
(4)   G80
```

спричиняє наступне (зверніть увагу, що *R* є липким, а *Z* також липким, оскільки площина є *XY*):

1. `g843.ngc` викликається зі словами x=1, y=2, z=3, r=1
2. `g843.ngc` викликається зі словами x=3, y=4, z=3, p=2, r=1
3. `g843.ngc` викликається зі словами x=6, y=7, z=3, r=1
4. Цикл `G84.3` скасовано.

Окрім створення нових циклів, це забезпечує простий метод перепакування існуючих G-кодів, які не працюють як цикли. Наприклад, код жорсткого нарізування різьби `G33.1` не працює як цикл. За допомогою такого обгортника можна легко створити новий код, який використовує `G33.1`, але працює як цикл.

Дивіться *configs/sim/axis/remap/cycle* для повного прикладу цієї функції. Вона містить два цикли, один з процедурою NGC, як вище, та приклад циклу, використовуючи лише Python.

9.6.8 Налаштування вбудованого Python

Плагін Python обслуговує як інтерпретатор, так і `task`, якщо так налаштовано, і тому має власний розділ `PYTHON` у INI-файлі.

9.6.8.1 Плагін Python: конфігурація INI-файлу

[PYTHON]

TOPLEVEL = *<filename>*
Ім’я файлу початкового скрипта Python, який буде виконано під час запуску. Цей скрипт відповідає за налаштування структури назви пакета, див. нижче.

PATH_PREPEND = *<directory>*
Додайте цей каталог до `PYTHON_PATH`. Повторювана група.

PATH_APPEND = *<directory>*
Додати цей каталог до `PYTHON_PATH`. Група, що повторюється.

LOG_LEVEL = *<integer>*
Рівень реєстрації дій, пов’язаних із плагінами. Збільште цей рівень, якщо підозрюєте проблеми. Може бути дуже детальним.

RELOAD_ON_CHANGE = [0|1]
Перезавантажте скрипт «TOPLEVEL», якщо файл було змінено. Зручно для налагодження, але наразі створює деякі накладні витрати під час виконання. Вимкніть цю опцію для робочих конфігурацій.

9.6.8.2 Виконання інструкцій Python з інтерпретатора

Для виконання команд ad-hoc було додано «гарячий коментар» Python. Вихідні дані Python за замовчуванням надходять до stdout, тому для перегляду результатів необхідно запустити LinuxCNC з терміналу. Приклад для вікна MDI:

```
;py,print(2*3)
```

Зверніть увагу, що екземпляр інтерпретатора доступний тут як `this`, тому ви також можете виконати:

```
;py,print(this.tool_table[0].toolno)
```

Ось підхід до використання підпрограми O-word для зчитування запису файлу налаштувань та додавання його як параметра G-коду.

```
(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 Програмування вбудованого Python в інтерпретаторі RS274NGC

9.6.9.1 Простір імен плагінів Python

Очікується, що простір імен буде структуровано наступним чином:

oword
: Будь-які виклики в цьому модулі є кандидатами для процедур Python O-word. Зверніть увагу, що модуль Python `oword` перевіряється **перед** тестуванням процедури NGC з тим самим іменем — фактично імена в `oword` приховують файли NGC з тим самим базовим іменем.

remap
: Очікується, що тут будуть знаходитися виклики Python, на які посилаються в опціях `prolog`, `epilog` або `python` специфікації argspec.

namedparams
: Функції Python у цьому модулі розширюють або перевизначають простір імен попередньо визначених іменованих параметрів, див. adding predefined parameters.

9.6.9.2 Інтерпретатор з точки зору Python

Інтерпретатор є існуючим класом C++ («Interp»), визначеним у «src/emc/rs274ngc». Концептуально всі виклики Python `oword.<function>` та `remap.<function>` є методами цього класу Interp, хоча явного визначення цього класу в Python немає (це екземпляр обгортки «Boost.Python»), і тому вони отримують як перший параметр `self`, який можна використовувати для доступу до внутрішніх компонентів.

9.6.9.3 Функції інтерпретатора __init__ та __delete__

Якщо модуль TOPLEVEL визначає функцію __init__, вона буде викликана після повного налаштування інтерпретатора (зчитування INI-файлу та синхронізації стану з моделлю світу).

Якщо модуль TOPLEVEL визначає функцію __delete__, вона буде викликана один раз перед завершення роботи інтерпретатора та після того, як постійні параметри будуть збережені у PARAMETER_FILE.

Примітка: на даний момент обробник __delete__ не працює для екземплярів інтерпретатора, створених шляхом імпорту модуля gcode. Якщо вам потрібна еквівалентна функціональність (що досить малоймовірно), розгляньте можливість використання модуля Python atexit.

```
# b''цb''b''eb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''зb''b''нb''b''аb''b''чb''b' ↩
    'eb''b''нb''b''оb'' b''вb'' b''мb''b''оb''b''дb''b''уb''b''лb''b''іb'' TOPLEVEL

def __init__(self):
    # b''дb''b''оb''b''дb''b''аb''b''йb''b''тb''b''eb'' b''сb''b''юb''b''дb''b''иb'' b' ↩
        'бb''b''уb''b''дb''b''ьb''-b''яb''b''кb''b''уb'' b''оb''b''дb''b''нb''b''оb''b''рb'' ↩
       b''аb''b''зb''b''оb''b''вb''b''уb'' b''іb''b''нb''b''іb''b''цb''b''іb''b''аb''b' ↩
        'лb''b''іb''b''зb''b''аb''b''цb''b''іb''b''юb''
    if self.task:
  # b''цb''b''eb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb'' milltask ↩
      interp
  pass
    else:
  # b''цb''b''eb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb'' b''іb''b ↩
     ''нb''b''тb''b''eb''b''рb''b''пb''b''рb''b''eb''b''тb''b''аb''b''цb''b''іb''b''їb'', b ↩
     ''щb''b''оb'' b''нb''b''eb'' b''єb'' milltask
       pass

def __delete__(self):
    # b''дb''b''оb''b''дb''b''аb''b''йb''b''тb''b''eb'' b''сb''b''юb''b''дb''b''иb'' b' ↩
        'бb''b''уb''b''дb''b''ьb''-b''яb''b''кb''b''іb'' b''дb''b''іb''b''їb'' b''зb'' b' ↩
        'оb''b''чb''b''иb''b''щb''b''eb''b''нb''b''нb''b''яb''/b''зb''b''бb''b''eb''b''рb''b ↩
        ''eb''b''жb''b''eb''b''нb''b''нb''b''яb'' b''сb''b''тb''b''аb''b''нb''b''уb''
    if self.task: # b''яb''b''кb'' b''вb''b''иb''b''щb''b''eb''
  pass
    else:
        pass
```

Ця функція може використовуватися для ініціалізації будь-яких атрибутів на стороні Python, які можуть знадобитися пізніше, наприклад, у функціях remap або O-word, а також для збереження або відновлення стану, що виходить за межі того, що надає PARAMETER_FILE.

Якщо є дії з налаштування або очищення, які повинні відбуватися тільки в екземплярі інтерпретатора milltask (на відміну від екземпляра інтерпретатора, який знаходиться в модулі Python gcode і служить для попереднього перегляду/відображення прогресу, але ні для чого іншого), це можна перевірити за допомогою evaluating self.task.

Приклад використання __init__ та __delete__ можна знайти в configs/sim/axis/remap/cycle/pytho
де ініціалізуються атрибути, необхідні для обробки циклів у ncfiles/remap_lib/python-stdglue/stdgl
(та імпортованих у файл configs/sim/axis/remap/cycle/python/remap.py).

9.6.9.4 Умови виклику: NGC до Python

Код Python викликається з NGC у таких ситуаціях:

- під час звичайного виконання програми:
  - коли виконується виклик O-слова, наприклад O<proc> call, і ім'я oword.proc визначено та доступне для виклику

- коли виконується коментар типу `;py,<оператор Python>`.
- під час виконання перепризначеного коду: будь-які обробники `prolog=`, `python=` та `epilog=`.

Виклик підпрограм Python на O-word

Аргументи:

self
: Екземпляр інтерпретатора.

***args**
: Список фактичних позиційних параметрів. Оскільки кількість фактичних параметрів може змінюватися, найкраще використовувати такий стиль оголошення:

```
# b''цb''b''eb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''зb''b''нb''b''аb''b''чb''b' ↩
    'eb''b''нb''b''оb'' b''вb'' b''мb''b''оb''b''дb''b''уb''b''лb''b''ib'' oword
def mysub(self, *args):
    print("number of parameters passed:", len(args))
    for a in args:
        print(a)
```

Повернення значень підпрограм Python типу O-word Так само, як процедури NGC можуть повертати значення, так само можуть повертати значення і підпрограми Python, що використовують літеру O-word. Очікується, що вони або повертатимуть значення

- немає значення (немає оператора `return` або значення `None`),
- значення типу "float" або "integer",
- рядок, це означає «це повідомлення про помилку, перервати програму». Працює як `(abort, msg)`.

Будь-який інший тип повертаного значення викличе виняток Python.

У викликаючому середовищі NGC доступні такі попередньо визначені іменовані параметри:

#<value>
: Значення, повернене останньою викликаною процедурою. Ініціалізується як 0.0 під час запуску. Відображається в інтерпретації як `self.return_value` (число з плаваючою комою).

#<value_returned>
: Вказує, що остання викликана процедура виконала `return` або `endsub` з явним значенням. 1.0, якщо true. Встановлюється на 0.0 для кожного `виклику`. В інтерпретації було виявлено `self.value_returned` (int).

Див. також приклад `tests/interp/value-returned`.

Домовленості щодо виклику підпрограм 'prolog=" та 'epilog=" Аргументи:

self
: Екземпляр інтерпретатора.

words
: Словник параметрів ключових слів. Якщо був присутній argspec, слова збираються з поточного блоку відповідно і передаються в словник для зручності (слова також можна було б отримати безпосередньо з блоку виклику, але це вимагає більш глибоких знань про внутрішню структуру інтерпретатора). Якщо argspec не був переданий або були вказані тільки опціональні значення, і жодне з них не було присутнє в блоці виклику, цей словник буде порожнім. Імена слів перетворюються в нижній регістр.

Приклад дзвінка:

```
def minimal_prolog(self, **words): # b''yb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''ib'' b' ↩
    'пb''b''eb''b''pb''b''eb''b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b' ↩
    'нb''b''нb''b''яb''
    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
```

Повернуті значення:

INTERP_OK
: Повернути це у разі успіху. Вам потрібно імпортувати це з `interpreter`.

текст повідомлення
: Повернення рядка з обробника означає «це повідомлення про помилку, перервіть програму». Працює як (`abort, <msg>`).

Домовленості щодо виклику підпрограм 'python=” Аргументи:

self
: Екземпляр інтерпретатора.

words
: Словник параметрів ключових слів. Той самий словник kwargs, що й для прологів та епілогів (див. вище).

Приклад мінімальної функції`python=”:

```
def useless(self,  **words): # b''yb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''ib'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb''
    return INTERP_OK
```

Повернуті значення:

INTERP_OK
: Повернути це у разі успіху

текст повідомлення
: Повернення рядка з обробника означає «це повідомлення про помилку, перервіть програму». Працює як (`abort, <msg>`).

Якщо обробнику потрібно виконати «операцію зупинки черги» (зміна інструменту, зондування, зчитування виводу HAL), тоді він повинен призупинити виконання за допомогою наступного оператора:

yield INTERP_EXECUTE_FINISH
: Це сигналізує `task` зупинити попереднє читання, виконати всі операції в черзі, виконати операцію «queue-buster», синхронізувати стан інтерпретатора зі станом машини, а потім сигналізувати інтерпретатору продовжувати роботу. На цьому етапі функція відновлюється з оператора, що йде за оператором `yield ...`.

Робота з вимкненням черги: зонд, зміна інструменту та очікування виводу HAL Queue busters переривають процедуру в точці, де викликається така операція, тому процедуру потрібно перезапустити після синхронізації інтерпретатора (synch()). Коли це відбувається, процедура повинна знати, чи вона перезапущена, і де продовжувати. Для перезапуску процедури використовуєть метод генератора Python.

Це демонструє продовження виклику з єдиною точкою перезапуску:

```
def read_pin(self,*args):
    # b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'' 5 b''cb''b''eb''b' ↩
        'кb''b''уb''b''нb''b''дb'', b''пb''b''ob''b''кb''b''иb'' b''цb''b''иb''b''фb''b' ↩
        'pb''b''ob''b''вb''b''иb''b''йb'' b''вb''b''хb''b''iв''b''дb'' 00 b''пb''b''eb''b' ↩
        'pb''b''eb''b''йb''b''дb''b''eb'' b''уb'' b''вb''b''иb''b''cb''b''ob''b''кb''b''иb'' ↩
       b''йb'' b''pb''b''iв''b''вb''b''eb''b''нb''b''ьb''
    emccanon.WAIT(0,1,2,5.0)
    # b''пb''b''eb''b''pb''b''eb''b''дb''b''ab''b''тb''b''иb'' b''кb''b''eb''b''pb''b''уb'' ↩
       b''вb''b''ab''b''нb''b''нb''b''яb'' b''пb''b''iв''b''cb''b''лb''b''яb'' b''вb''b' ↩
        'иb''b''кb''b''ob''b''нb''b''ab''b''нb''b''нb''b''яb'' b''чb''b''eb''b''pb''b''гb''b ↩
        ''иb''-b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''нb''b''яb'':
    yield INTERP_EXECUTE_FINISH
    # b''Bb''b''иb''b''кb''b''ob''b''нb''b''ab''b''нb''b''нb''b''яb'' post-sync() b''вb''b' ↩
        'iв''b''дb''b''нb''b''ob''b''вb''b''лb''b''юb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b ↩
        ''тb''b''уb''b''тb'':
    pin_status = emccanon.GET_EXTERNAL_DIGITAL_INPUT(0,0);
    print("pin status=",pin_status)
```



Warning
Функція «yield» є крихкою. На використання «yield INTERP_EXECUTE_FINISH» поширюються такі обмеження:

- Код Python, який виконує *yield INTERP_EXECUTE_FINISH*, має бути частиною процедури перепризна Yield не працює в процедурі Python oword.
- Підпрограма перепризначення Python, що містить оператор *yield INTERP_EXECUTE_FINISH*, може не повертати значення, як це відбувається зі звичайними операторами yield Python.
- Код, що слідує за yield, не може рекурсивно викликати інтерпретатор, як у випадку з `self.execute("command>")`. Це архітектурне обмеження інтерпретатора, яке неможливо виправити без серйозного перепроектування.

9.6.9.5 Умовні позначення викликів: Python до NGC

Код NGC виконується з Python, коли

- виконується метод `self.execute(<код NGC>[,<номер рядка>])`, або
- Під час виконання перепризначеного коду, якщо визначено функцію `prolog=`, процедура NGC, задана в `ngc=`, виконується одразу після цього.

Обробник прологу не викликає обробник, але готує середовище його виклику, наприклад, встановлюю попередньо визначені локальні параметри.

Вставка параметрів у пролог та їх отримання в епілог Концептуально пролог та епілог виконуються на одному рівні виклику, як і процедура O-word, тобто після встановлення виклику підпрограми та до завершення підпрограми endsub або return.

Це означає, що будь-яка локальна змінна, створена в пролозі, буде локальною змінною в процедурі O-word, а будь-які локальні змінні, створені в процедурі O-word, залишаються доступними під час виконання епілогу.

Масив `self.params` обробляє читання та встановлення пронумерованих і іменованих параметрів. Якщо іменований параметр починається з _ (підкреслення), він вважається глобальним параметром; якщо ні, він є локальним для процедури виклику. Крім того, пронумеровані параметри в діапазоні 1..30 обробляються як локальні змінні; їхні початкові значення відновлюються при поверненні/заверше процедури O-word.

Ось приклад переробленого коду, що демонструє вставку та вилучення параметрів у/з процедури O-word:

```
REMAP=m300 prolog=insert_param ngc=testparam epilog=retrieve_param modalgroup=10
```

```
def insert_param(self, **words): # b''yb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''ib'' b' ↩
    'пb''b''eb''b''pb''b''eb''b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b' ↩
    'нb''b''нb''b''яb''
    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>)
; b''cb''b''пb''b''pb''b''ob''b''бb''b''yb''b''йb''b''тb''b''eb'' b''зb''b''ab''b''кb''b' ↩
    'ob''b''мb''b''eb''b''нb''b''тb''b''yb''b''вb''b''ab''b''тb''b''иb'' b''нb''b''ab''b' ↩
    'cb''b''тb''b''yb''b''пb''b''нb''b''иb''b''йb'' b''pb''b''яb''b''дb''b''ob''b''кb'' b' ↩
    'ib'' b''зb''b''ab''b''пb''b''yb''b''cb''b''тb''b''ib''b''тb''b''ьb'' b''щb''b''eb'' b' ↩
    'pb''b''ab''b''зb''
#<result> = [#<myname> * 3]
#1 = [#1 * 5]
#2 = [#2 * 3]
o<testparam> endsub
m2
```

`self.params()` повертає список усіх імен змінних, визначених на даний момент. Оскільки `myname` є локальною, вона зникає після завершення епілогу.

Виклик інтерпретатора з Python Ви можете рекурсивно викликати інтерпретатор з коду Python наступним чином:

```
self.execute(<NGC code>[,<line number>])
```

Приклади:

```
  self.execute("G1 X%f Y%f" % (x,y))
  self.execute("O <myprocedure> call", currentline)
```

Можливо, вам варто перевірити, чи повертається значення < `INTERP_MIN_ERROR`. Якщо ви використову багато операторів execute(), ймовірно, легше перехопити InterpreterException, як показано нижче.



Caution
Метод вставки/отримання параметрів, описаний у попередньому розділі, у цьому випадку не працює. Його достатньо лише для

- виконання простих команд NGC або виклику процедури та
- поглиблене самоаналізування процедури, та
- передача локальних іменованих параметрів не потрібна.

Функція рекурсивного виклику є крихкою.

Виняток інтерпретатора під час виконання() Якщо `interpreter.throw_exceptions` не дорівнює нулю (за замовчуванням 1), і self.execute() повертає помилку, викликається виняток `InterpreterExcep` InterpreterException має такі атрибути:

`line_number`
де сталася помилка

`line_text`
оператор NGC, що спричиняє помилку

`error_message`
повідомлення про помилку інтерпретатора

Помилки можна перехопити наступним пайтонським способом:

```
import interpreter
interpreter.throw_exceptions = 1
   ...
   try:
        self.execute("G3456")  #  b''пb''b''ib''b''дb''b''нb''b''яb''b''тb''b''иb'' b''вb'' ↩
            b''иb''b''нb''b''яb''b''тb''b''оb''b''кb'' InterpreterException

   except InterpreterException,e:
        msg = "%d: '%s' - %s" % (e.line_number,e.line_text, e.error_message)
        return msg  # b''зb''b''ab''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''вb''b' ↩
            'бb''b''уb''b''дb''b''оb''b''вb''b''ab''b''нb''b''eb'' b''пb''b''оb''b''вb''b' ↩
            'ib''b''дb''b''оb''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''рb''b' ↩
            'оb'' b''пb''b''оb''b''мb''b''иb''b''лb''b''кb''b''уb''
```

Канон Шар канону практично повністю складається з вільних функцій. Приклад:

```
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
```

Фактичні функції канону оголошені в `src/emc/nml_intf/canon.hh` та реалізовані в `src/emc/task/emcca` Реалізацію функцій Python можна знайти в `src/emc/rs274ncg/canonmodule.cc`.

9.6.9.6 Вбудовані модулі

Вбудовані такі модулі:

interpreter
Відкриває внутрішні механізми класу Interp. Див. src/emc/rs274ngc/interpmodule.cc та регресійний тест tests/remap/introspect.

emccanon
Виявляє більшість дзвінків src/emc/task/emccanon.cc.

9.6.10 Додавання попередньо визначених іменованих параметрів

Інтерпретатор постачається з набором попередньо визначених іменованих параметрів для доступу до внутрішнього стану з рівня мови NGC. Ці параметри доступні лише для читання та є глобальними, тому їм не можна призначити значення.

Додаткові параметри можна додати, визначивши функцію в модулі namedparams. Ім'я функції визначає ім'я нового попередньо визначеного іменованого параметра, на який тепер можна посилатися в довільних виразах.

Щоб додати або перевизначити іменований параметр:

- Додайте модуль namedparams, щоб інтерпретатор міг його знайти,
- визначте нові параметри за допомогою функцій (див. нижче). Ці функції отримують self (екземпляр інтерпретатора) як параметр і тому можуть отримати доступ до довільного стану. Для повернення значення можна використовувати довільні можливості Python.
- Імпортуйте цей модуль зі скрипта TOPLEVEL.

```
# namedparams.py
# b''тb''b''рb''b''иb''b''вb''b''іb''b''аb''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''пb''b' ↩
    'рb''b''иb''b''кb''b''лb''b''аb''b''дb''
def _pi(self):
    return 3.1415926535
```

```
#<circumference> = [2 * #<radius> * #<_pi>]
```

Очікується, що функції в namedparams.py повертатимуть значення типу float або int. Якщо повертається рядок, це встановлює повідомлення про помилку інтерпретатора та перериває виконання.

Тільки функції з початковим символом підкреслення додаються як параметри, оскільки це конвенція RS274NGC для глобальних змінних.

Можна перевизначити існуючий попередньо визначений параметр, додавши функцію Python з такою ж назвою до модуля namedparams. У цьому випадку під час запуску генерується попередження.

Хоча наведений вище приклад не є надто корисним, зверніть увагу, що практично весь внутрішній стан інтерпретатора доступний з Python, тому довільні предикати можуть бути визначені таким чином. Трохи більш просунутий приклад дивіться в tests/remap/predefined-named-params.

9.6.11 Стандартні процедури клею

Оскільки багато завдань перемапування дуже схожі, я почав збирати робочі процедури прологу та епілогу в одному модулі Python. Наразі їх можна знайти в «ncfiles/remap_lib/python-stdglue/stdglue.py де вони надають такі процедури:

9.6.11.1 T: prepare_prolog і prepare_epilog

Вони завершують процедуру NGC для підготовки інструменту T*x*.

Дії prepare_prolog Наступні параметри стають видимими для процедури NGC:

- #<tool> - параметр слова на літеру T
- #<pocket> - відповідну кишеню

Якщо запитується інструмент номер нуль (тобто вивантаження інструменту), відповідне гніздо передається як -1.

Це помилка, якщо:

- Номер інструменту не вказано як параметр T,
- Інструмент не знайдено в таблиці інструментів.

Зверніть увагу, що якщо ви не встановите параметр [EMCIO] RANDOM_TOOLCHANGER=1, номер інструмент та гнізда будуть ідентичними, а номер гнізда з таблиці інструментів ігнорується. Наразі це обмеження.

Дії prepare_epilog

- Очікується, що процедура NGC поверне додатне значення, інакше видається повідомлення про помилку, що містить повернене значення, і інтерпретатор перериває роботу.
- У разі, якщо процедура NGC виконала команду T (яка потім посилається на вбудовану поведінку T), ніяких подальших дій не виконується. Це можна використовувати, наприклад, для мінімального коригування вбудованої поведінки, додаючи перед нею або після неї деякі інші оператори.
- В іншому випадку параметри #<tool> та #<pocket> витягуються з простору параметрів підпрограми. Це означає, що процедура NGC може змінювати ці значення, і епілог враховує змінені значення.
- Потім виконується команда Canon SELECT_TOOL(#<інструмент>).

9.6.11.2 M6: change_prolog і change_epilog

Це завершує процедуру NGC для зміни інструменту M6.

Дії change_prolog

- Якщо попередньої команди T, яка б спричинила вибір кишені, не було, пролог переривається з повідомленням про помилку.
- Якщо компенсація радіуса різця увімкнена, пролог переривається з повідомленням про помилку.

Потім, такі параметри експортуються до процедури NGC:

- #<tool_in_spindle> : номер інструменту, що наразі завантажений
- #<вибраний_інструмент>: номер вибраного інструменту
- #<selected_pocket> : індекс tooldata вибраного інструменту

Дії +change_epilog

- Очікується, що процедура NGC поверне додатне значення, інакше видається повідомлення про помилку, що містить повернене значення, і інтерпретатор перериває роботу.

- У разі, якщо процедура NGC виконала команду M6 (яка потім посилається на вбудовану поведінку M6), ніяких подальших дій не виконується. Це можна використовувати, наприклад, для мінімального коригування вбудованої поведінки, додаючи перед нею або після неї деякі інші оператори.
- В іншому випадку параметр #<selected_pocket> витягується з простору параметрів підпрограми і використовується для встановлення змінної інтерпретатора current_pocket. Знову ж таки, процедура може змінити це значення, і епілог враховує змінене значення.
- Потім виконується команда Canon CHANGE_TOOL(#<вибрана_кишеня>).
- Нові параметри інструменту (зміщення, діаметр тощо) встановлено.

9.6.11.3 Цикли G-коду: cycle_prolog and cycle_epilog

Вони обгортають процедуру NGC, щоб вона могла діяти як цикл, тобто код руху зберігається після завершення виконання. Якщо наступний рядок містить тільки слова-параметри (наприклад, нові значення X, Y), код виконується знову з новими словами-параметрами, об'єднаними в набір параметрів, заданих у першому виклику.

Ці процедури призначені для роботи у поєднанні з параметром argspec=<words>. Хоча це просте у використанні, у реальному сценарії слід уникати argspec і проводити більш ретельне дослідження блоку вручну, щоб отримати кращі повідомлення про помилки.

Запропонована специфікація аргументів виглядає наступним чином:

```
REMAP=G<somecode> argspec=xyzabcuvwqplr prolog=cycle_prolog ngc=<ngc procedure> epilog= ↩
    cycle_epilog modalgroup=1
```

Це дозволить cycle_prolog визначити сумісність будь-яких слів осі, наданих у блоці, див. нижче.

Дії cycle_prolog

- Визначте, чи відповідають слова, передані з поточного блоку, умовам, описаним у Помилки стандартного циклу.
  - Експортувати слова осі як <x>, #<y> тощо; не вдасться, якщо слова осі з різних груп (XYZ) (UVW) використовуються разом або якщо задано будь-яке з (ABC).
  - Експортувати *L*- як #<l>; за замовчуванням 1, якщо не вказано.
  - Експортувати *P*- як #<p>; невдача, якщо p менше 0.
  - Експортувати *R*- як #<r>; невдача, якщо r не вказано, або менше дорівнює 0, якщо вказано.
  - Помилка, якщо швидкість подачі дорівнює нулю, або якщо увімкнено зворотну подачу за часом чи компенсацію різця.
- Визначте, чи це перший виклик G-коду циклу, якщо так:
  - Додайте передані слова (згідно з argspec) до набору фіксованих параметрів, які зберігаються протягом кількох викликів.
- Якщо ні (рядок продовження з новими параметрами), то
  - об'єднати передані слова з існуючим набором закріплених параметрів.
- Експортуйте набір фіксованих параметрів до процедури NGC.

Дії cycle_epilog

- Визначити, чи справді поточний код був циклом, якщо так, то
  - зберегти поточний режим руху, тому рядок продовження без коду руху виконає той самий код руху.

9.6.11.4 S (Встановити швидкість): `setspeed_prolog` та `setspeed_epilog`

TBD

9.6.11.5 F (Встановити потік): `setfeed_prolog` та `setfeed_epilog`

TBD

9.6.11.6 M61 Встановити номер інструменту: `settool_prolog` та `settool_epilog`

TBD

9.6.12 Перепризначене виконання коду

9.6.12.1 Середовище виклику процедур NGC під час перепризначення

Зазвичай процедура O-word викликається з позиційними параметрами. Ця схема є дуже обмеженою, особливо в разі наявності необов'язкових параметрів. Тому угода про виклик була розширена, щоб використовувати щось віддалено схоже на модель ключових аргументів Python.

Див. LINKTO Підпрограми G-коду/основного типу: sub, endsub, return, call.

9.6.12.2 Вкладені перепризначені коди

Перепризначені коди можуть бути вкладеними так само, як і виклики процедур, тобто перепризначени код, процедура NGC якого посилається на якийсь інший перепризначений код, виконається належним чином.

Максимальна кількість перепризначень рівня вкладеності наразі становить 10.

9.6.12.3 Порядковий номер під час перепризначення

Номери послідовностей поширюються та відновлюються так само, як і у викликах O-word. Дивіться `tests/remap/nested-remaps/word` для регресійного тесту, який показує відстеження номерів послідовностей під час вкладених перемаплень на трьох рівнях глибини.

9.6.12.4 Прапорці налагодження

Наступні прапорці стосуються перепризначення та виконання, пов'язаного з Python:

`EMC_DEBUG_OWORD`	`0x00002000`	відстежує виконання підпрограм типу O-word
`EMC_DEBUG_REMAP`	`0x00004000`	відстежує виконання коду, пов'язаного з перепризначенням
`EMC_DEBUG_PYTHON`	`0x00008000`	виклики плагіна Python
`EMC_DEBUG_NAMEDPARAM`	`0x00010000`	доступ до іменованих параметрів трасування
`EMC_DEBUG_USER1`	`0x10000000`	визначений користувачем - не інтерпретується LinuxCNC

EMC_DEBUG_USER2	0x20000000	визначений користувачем - не інтерпретується LinuxCNC

або ці прапорці у змінну `[EMC]DEBUG` за потреби. Поточний список прапорців налагодження див. у *src/emc/nml_intf/debugflags.h*.

9.6.12.5 Налагодження вбудованого коду Python

Налагодження вбудованого коду Python складніше, ніж налагодження звичайних скриптів Python, і існує лише обмежена кількість налагоджувачів. Працюючим рішенням на основі відкритого коду є використання Eclipse IDE [https://www.eclipse.org] та плагіна Eclipse https://www.pydev.org [PydDev] і його функції віддаленого налагодження https://pydev.org/manual_adv_remote_debugger.html [remote debugging feature].

Щоб скористатися цим підходом:

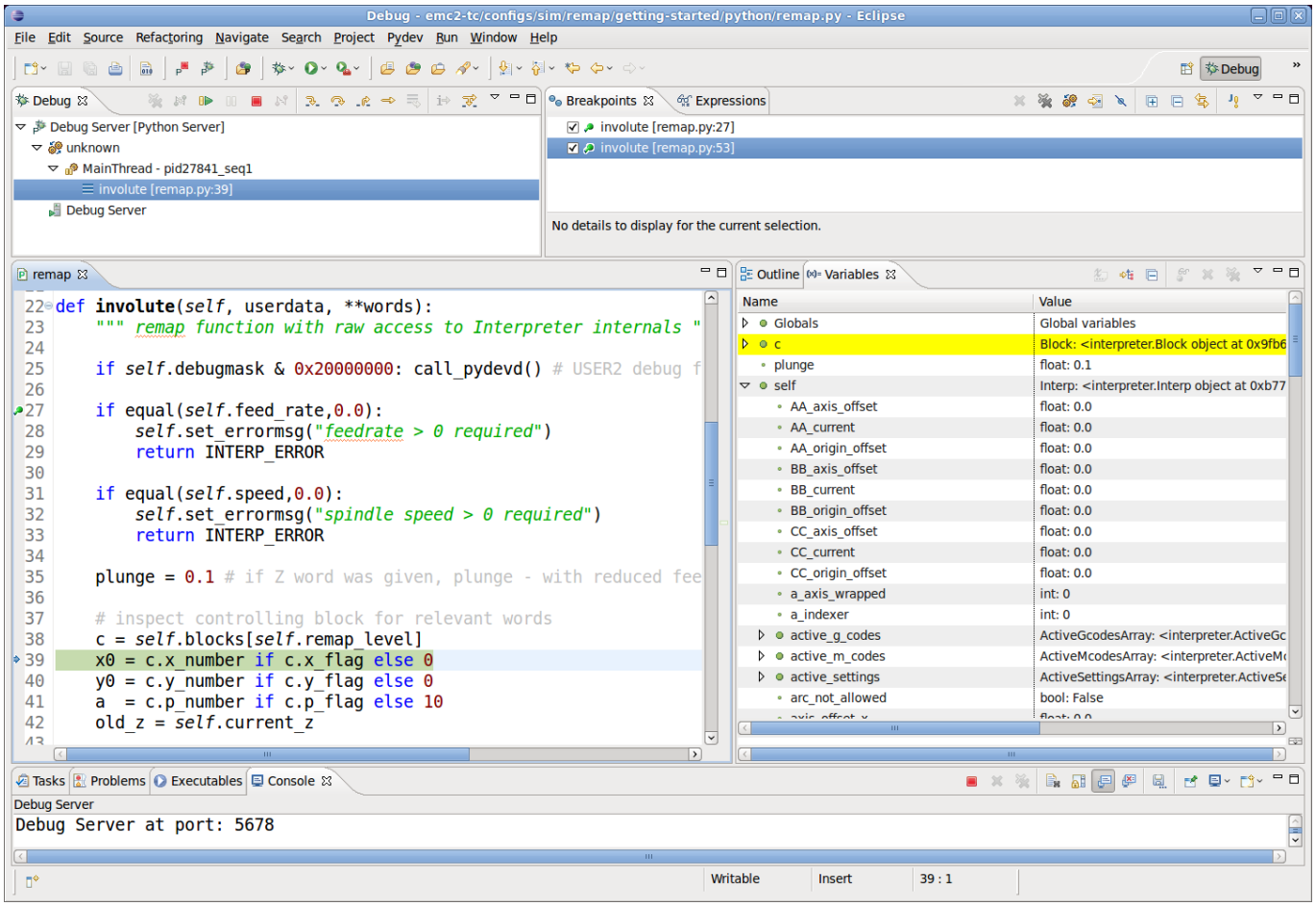
- Встановіть Eclipse через «Центр програмного забезпечення Ubuntu» (виберіть перший варіант).
- Встановіть плагін PyDev з сайт оновлень Pydev.
- Налаштуйте дерево вихідного коду LinuxCNC як проект Eclipse.
- Запустіть сервер налагодження Pydev в Eclipse.
- Переконайтеся, що вбудований код Python може знайти модуль `pydevd.py`, який поставляється разом із цим плагіном — він захований десь глибоко в каталозі інсталяції Eclipse. Встановіть змінну `pydevd` у файлі `util.py`, щоб відобразити розташування цього каталогу.
- Додайте `import pydevd` до вашого модуля Python - див. приклади `util.py` та `remap.py`.
- Викличте `pydevd.settrace()` у вашому модулі в певний момент, щоб підключитися до сервера налагодження Eclipse Python — тут ви можете встановлювати точки зупинки у вашому коді, перевіряти змінні, кроки тощо, як завжди.



Caution
`pydevd.settrace()` блокуватиме виконання, якщо Eclipse та сервер налагодження Pydev не запущено.

Щоб охопити останні два кроки: процедура `o<pydevd>` допомагає перейти в відладчик з режиму MDI. Дивіться також функцію `call_pydevd` у файлі `util.py` та її використання у файлі `remap.involute` для встановлення точки зупинки.

Ось скріншот налагодження процедури `involute` в Eclipse/PyDev, як показано вище:



Дивіться код Python у `configs/sim/axis/remap/getting-started/python` для отримання детальної інформації.

9.6.13 Попередній перегляд осі та виконання перепризначеного коду

Для повного попереднього перегляду траєкторії інструменту перемальованого коду необхідно вжити деяких запобіжних заходів. Щоб зрозуміти, що відбувається, давайте розглянемо процес попереднього перегляду та виконання (це стосується випадку AXIS, але інші випадки є подібними):

Спочатку зверніть увагу, що є **два** незалежні екземпляри інтерпретатора:

- Один приклад у програмі milltask, яка виконує програму після натискання кнопки «Пуск» і фактично змушує машину рухатися.
- Другий екземпляр в інтерфейсі користувача, основним призначенням якого є створення попередньо перегляду траєкторії інструменту. Він «виконує» програму після її завантаження, але фактично не призводить до рухів верстата.

Тепер припустимо, що ваша процедура перемапування містить операцію зонда G38, наприклад, як частину зміни інструменту з автоматичним відключенням довжини інструменту. Якщо зонд не працює, це буде явною помилкою, тому ви відобразите повідомлення і перервете програму.

А що щодо попереднього перегляду цієї процедури? Під час попереднього перегляду, звичайно, невідомо, чи зонд досягне успіху чи зазнає невдачі, але ви, ймовірно, захочете побачити максимальну глибину зонда і припустити, що він досягне успіху, і продовжити виконання, щоб переглянути подальші рухи. Крім того, немає сенсу відображати повідомлення «зонд зазнав невдачі» і припиняти виконання **під час попереднього перегляду**.

Щоб вирішити цю проблему, потрібно перевірити, чи виконується ваша процедура в режимі попереднього перегляду чи виконання. Це можна перевірити, протестувавши #<_task> predefined named parameter — під час фактичного виконання значення буде 1, а під час попереднього перегляду — 0. Повний приклад використання див. у файлі *configs/sim/axis/remap/manual-toolchange-with-tool-length-switch/nc_subroutines/manual_change.ngc*.

У Embedded Python екземпляр `task` можна перевірити, перевіривши *self.task* — це буде 1 в екземплярі milltask та 0 в екземплярі(ах) попереднього перегляду.

9.6.14 Перепризначені коди

9.6.14.1 Існуючі коди, які можна перепризначити

Поточний набір **існуючих** кодів, відкритих для перевизначення, такий:

- T*x* (Підготуйтеся)
- M6 (Змінити інструмент)
- M61 (Встановити номер інструменту)
- M0 (тимчасово призупинити запущену програму)
- M1 (тимчасово призупинити запущену програму, якщо увімкнено додатковий вимикач зупинки)
- M7 (активувати туман охолоджувальної рідини)
- M8 (активувати залив охолоджувальної рідини)
- M9 (деактивація туману та затоплення охолоджувальної рідини)
- M60 (заміна піддонних човників та тимчасова зупинка запущеної програми)
- M62 .. M65 (керування цифровим виходом)
- M66 (очікування введення)
- M67, M68 (керування аналоговим виходом)
- S (встановлена швидкість шпинделя)
- F (встановити канал)

9.6.14.2 Наразі нерозподілені G-коди:

Наразі нерозподілені G-коди (для перепризначення) необхідно вибирати з порожніх полів наступних таблиць. Усі перелічені G-коди вже визначені в поточній реалізації LinuxCNC і **не** можуть використовуватися для перепризначення нових G-кодів. (Розробникам, які додають нові G-коди до LinuxCNC, рекомендується також додавати свої нові G-коди до цих таблиць.)

Table 9.5: Таблиця розподілених G-кодів 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									

Table 9.5: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
05	G05	G05.1	G05.2	G05.3						
06										
07	G07									
08	G08									
09										

Table 9.6: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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										

Table 9.8: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
37										
38										
39										

Table 9.9: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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										

Table 9.11: (continued)

#	Gxx	Gxx.1	Gxx.2	Gxx.3	Gxx.4	Gxx.5	Gxx.6	Gxx.7	Gxx.8	Gxx.9
69										

Table 9.12: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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: Таблиця розподілених G-кодів 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 Наразі нерозподілені M-коди:

Ці M-коди наразі не визначені в поточній реалізації LinuxCNC і можуть бути використані для визначення нових M-кодів. (Розробникам, які визначають нові M-коди в LinuxCNC, рекомендується вилучити їх із цієї таблиці.)

Table 9.15: Таблиця нерозподілених M-кодів 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

Усі M-коди від `M100` до `M199` вже є користувацькими M-кодами, і їх не слід перепризначати.

Усі M-коди від `M200` до `M999` доступні для перепризначення.

9.6.15 Короткий огляд виконання програми LinuxCNC

Щоб зрозуміти перепризначення кодів, може бути корисним розглянути виконання `task` та інтерпрета стосовно перепризначення.

9.6.15.1 Стан інтерпретатора

Концептуально, стан інтерпретатора складається зі змінних, які поділяються на такі категорії:

1. *Інформація про конфігурацію* (зазвичай з INI-файлу)
2. *«Світова модель»* – представлення фактичного стану машини
3. *Модальний стан і налаштування* - відноситься до стану, який «переноситься» між виконанням окремих кодів NGC - наприклад, після увімкнення шпинделя і встановлення швидкості, він залишається в цьому налаштуванні до вимкнення. Те саме стосується багатьох кодів, таких як подача, одиниці виміру, режими руху (подача або швидкий) і так далі.
4. *Стан виконання інтерпретатора* - Містить інформацію про блок, що виконується в даний момент, про те, чи перебуваємо ми в підпрограмі, про змінні інтерпретатора тощо. Більша частина цього стану агрегується в - досить несистематичній - `структурі _setup` (див. interp_internals.hh).

9.6.15.2 Взаємодія завдання та інтерпретатора, черга та попереднє читання

Частина `task` в LinuxCNC відповідає за координацію фактичних команд машини - рух, взаємодію з HAL тощо. Вона сама по собі не обробляє мову RS274NGC. Для цього `task` викликає інтерпретатор для аналізу та виконання наступної команди - або з MDI, або з поточного файлу.

Виконання інтерпретатора генерує канонічні операції машини, які фактично переміщують щось. Однак вони не виконуються відразу, а ставляться в чергу. Фактичне виконання цих кодів відбувається в частині `task` LinuxCNC: канонічні команди витягуються з черги інтерпретатора і виконуються, що призводить до фактичних рухів машини.

Це означає, що зазвичай інтерпретатор значно випереджає фактичне виконання команд — розбір програми може бути завершений ще до того, як почнуться будь-які помітні рухи. Така поведінка називається «попереднім читанням».

9.6.15.3 Прогнозування положення машини

Щоб заздалегідь обчислити канонічні машинні операції під час попереднього зчитування, інтерпретато повинен мати можливість передбачити позицію машини після кожного рядка G-коду, а це не завжди можливо.

Розглянемо простий приклад програми, яка виконує відносні переміщення (G91), і припустимо, що верстат починає роботу з координат x=0, y=0, z=0. Відносні переміщення означають, що результат наступного переміщення залежить від положення попереднього:

```
N10 G91
N20 G0 X10 Y-5 Z20
N30 G1 Y20 Z-5
N40 G0 Z30
N50 M2
```

Тут інтерпретатор може чітко передбачити позиції машини для кожного рядка:

Після N20: x=10 y=-5 z=20; після N30: x=10 y=15 z=15; після N40: x=10 y=15 z=45 і таким чином можна розібрати всю програму та згенерувати канонічні операції заздалегідь.

9.6.15.4 Засоби запобігання черзі порушують прогнозування позиції

Однак повне попереднє читання можливе лише тоді, коли інтерпретатор може заздалегідь передбачит вплив позиції для **кожного** рядка в програмі. Розглянемо модифікований приклад:

```
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
```

Щоб заздалегідь обчислити рух в N90, інтерпретатор повинен знати, де знаходиться верстат після рядка N80, а це залежить від того, чи була команда зонда успішною, що невідомо до її фактичного виконання.

Отже, деякі операції несумісні з подальшим попереднім читанням. Вони називаються *виключателями черги*, і вони:

- Зчитування значення виводу HAL за допомогою M66: значення виводу HAL непередбачуване.
- Завантаження нового інструменту за допомогою M6: геометрія інструменту непередбачувана.
- Виконання зонду з G38.*n*: кінцеве положення та успіх/невдача непередбачувані.

9.6.15.5 Як поводяться з тими, хто не пропускає черги

Щоразу, коли інтерпретатор зустрічає помилку, що обходить чергу, йому потрібно зупинити попереднє читання та чекати, поки не стане доступним відповідний результат. Це працює так:

- Коли зустрічається такий код, інтерпретатор повертає спеціальний код повернення до `task` (*INTERP_EXECUTE_FINISH*).
- Цей код повернення сигналізує `task`, щоб на даний момент припинити попереднє читання, виконати всі накопичені до цього моменту канонічні команди в черзі (включаючи останню, яка є черговою), а потім «синхронізувати стан інтерпретатора з моделлю світу». Технічно це означає оновлення внутрішніх змінних для відображення значень виводів HAL, перезавантаження геометрії інструменту після M6 і передачу результатів зондування.
- Метод інтерпретатора *synch()* викликається `task` і робить саме це — зчитує всі значення *factual* моделі світу, які є релевантними для подальшого виконання.
- У цей момент `task` продовжується та викликає інтерпретатор для подальшого читання — доки програма не завершиться або не буде виявлено інший обхідник черги.

9.6.15.6 Порядок слів та порядок виконання

Одне або декілька «слів» можуть бути присутніми в «блоці» NGC, якщо вони сумісні (деякі з них є взаємовиключними і повинні бути в різних рядках). Однак модель виконання передбачає суворе дотримання порядку виконання кодів, незалежно від їхнього розташування в вихідному рядку (G-code Order of Execution).

9.6.15.7 Розбір

Після зчитування рядка (в режимі MDI або з поточного файлу NGC) він аналізується, а прапорці та параметри встановлюються в «структурному блоці» (struct _setup, member block1). Ця структура містить всю інформацію про поточний рядок джерела, але незалежно від різного порядку кодів у поточному рядку: якщо кілька кодів сумісні, будь-який порядок джерела призведе до встановлення однакових змінних у структурному блоці. Відразу після розбору всі коди в блоці перевіряються на сумісність.

9.6.15.8 Виконання

Після успішного розбору блок виконується за допомогою execute_block(), і тут різні елементи обробляються відповідно до порядку виконання.

Якщо знайдено «черговий руйнівник», у стані інтерпретатора встановлюється відповідний прапорець (toolchange_flag, input_flag, probe_flag), і інтерпретатор повертає значення INTERP_EXECUTE_FINISH, сигналізуючи викликанню («`task`») про «тимчасове припинення попереднього читання та повторну синхронізацію». Якщо після виконання всіх елементів не виявлено жодного порушника черги, повертається INTERP_OK, що сигналізує про можливість продовження попереднього читання.

Коли читання вперед продовжується після синхронізації, `task` знову починає виконувати операції інтерпретатора read(). Під час наступної операції читання перевіряються вищезазначені прапорці та встановлюються відповідні змінні (оскільки щойно було виконано synch(), значення тепер є поточними). Це означає, що наступна команда вже виконується в правильно встановленому контексті змінних.

9.6.15.9 Виконання процедури

Процедури O-word дещо ускладнюють обробку queue buster. Queue buster може бути знайдений десь у вкладеній процедурі, що призводить до напівзавершеного виклику процедури, коли повертаєтьс INTERP_EXECUTE_FINISH. Task забезпечує синхронізацію моделі світу та продовжує аналіз і виконання, доки виконується процедура (call_level > 0).

9.6.15.10 Як зараз працює зміна інструменту

Дії, що відбуваються в LinuxCNC, дещо складні, але необхідно отримати загальне уявлення про те, що відбувається на даний момент, перш ніж ви почнете адаптувати ці механізми до власних потреб.

Зверніть увагу, що перепризначення існуючого коду повністю вимикає всю внутрішню обробку для цього коду. Це означає, що крім бажаної поведінки (яка, ймовірно, описана за допомогою NGC O-word або процедури Python), вам потрібно відтворити внутрішні дії інтерпретатора, що в сукупності призведе до повної заміни існуючого коду. Це можна зробити в коді прологу та епілогу.

Як передається інформація про інструмент Кілька процесів «цікавляться» інформацією про інструменти: `task` та його інтерпретатор, а також інтерфейс користувача. Також процес `halui`.

Інформація про інструмент зберігається в структурі «emcStatus», яка є спільною для всіх сторін. Одним з її полів є масив «toolTable», який містить опис, завантажений з файлу таблиці інструментів (номер інструменту, діаметр, кут нахилу передньої частини, кут нахилу задньої частини та орієнтація для токарного верстата, інформація про зміщення інструменту).

Авторитетним джерелом і єдиним процесом, який фактично «встановлює» інформацію про інструмент в цій структурі, є процес «iocontrol». Всі інші процеси лише звертаються до цієї структури. Інтерпретатор фактично містить локальну копію таблиці інструментів.

Для цікавих, поточна структура emcStatus доступна за допомогою Python statements. Інтерпретатор сприймає інструмент, завантажений на даний момент, наприклад, доступний за допомогою:

```
;py,from interpreter import *
;py,print(this.tool_table[0])
```

Щоб побачити результати, потрібно запустити LinuxCNC з вікна терміналу.

9.6.15.11 Як працює T*x* (інструмент підготовки)

Дія інтерпретатора для команди Tx

Все, що робить інтерпретатор, це оцінює параметр toolnumber, шукає відповідний індекс tooldata, запам'ятовує його в змінній `selected_pocket` для подальшого використання та ставить в чергу команду canon (SELECT_TOOL). Див. *Interp::convert_tool_select* в *src/emc/rs274/interp_execute.cc*.

Дія завдання на SELECT_TOOL Коли `task` починає обробку SELECT_TOOL, він надсилає повідомлен EMC_TOOL_PREPARE процесу `iocontrol`, який обробляє більшість дій, пов'язаних з інструментами в LinuxCNC.

У поточній реалізації `task` фактично чекає, поки `iocontrol` завершить операцію позиціонування змінювача, що, на мій погляд, не є необхідним, оскільки це суперечить ідеї, що підготовка змінювача та виконання коду можуть відбуватися паралельно.

Дія Iocontrol для EMC_TOOL_PREPARE Коли `iocontrol` бачить команду вибору кишені, він виконує відповідне коливання контакту HAL - він встановлює контакт «tool-prep-number», щоб вказати, який інструмент буде наступним, піднімає контакт «tool-prepare» і чекає, поки контакт «tool-prepared» не стане високим.

Коли змінник відповідає повідомленням «інструмент готовий», він вважає етап підготовки завершеним і сигналізує завданню про продовження. Знову ж таки, це «очікування», на мою думку, не є абсолютно необхідним.

Побудова прологу та епілогу для Tx Дивіться функції Python prepare_prolog та prepare_epilog у nc_files/remap_lib/python-stdglue/stdglue.py.

9.6.15.12 Як працює M6 (інструмент зміни)

Ви повинні повністю це зрозуміти, перш ніж зможете адаптувати це. Це дуже важливо для написання прологу та епілогу для перепрограмованого M6. Перепрограмування існуючих кодів означає, що ви вимикаєте внутрішні кроки, які виконуються зазвичай, і повторюєте їх настільки, наскільки це необхідно для ваших цілей.

Навіть якщо ви не знайомі з C, я пропоную вам переглянути код *Interp::convert_tool_change* у *src/emc/rs274/interp_convert.cc*.

Дія інтерпретатора з командою M6

Коли перекладач бачить M6, він:

1. перевіряє, чи вже було виконано команду T (перевірте, чи *settings->selected_pocket* має бути >= 0), і, якщо ні, видає повідомлення *Need tool prepared -Txx- for toolchange*.
2. Перевірте, чи активна компенсація на радіус різця, і якщо так, видасть повідомлення про помилку «Неможливо змінити інструменти, якщо ввімкнено компенсацію радіуса різця».
3. зупинити шпиндель, окрім випадків, коли встановлено опцію INI ”TOOL_CHANGE_WITH_SPINDLE
4. генерувати швидкий рух «Z вгору», якщо встановлено опцію INI «TOOL_CHANGE_QUILL_UP».
5. якщо було встановлено TOOL_CHANGE_AT_G30:
   a. перемістити індексатори A, B та C, якщо це можливо
   b. генерувати швидкий рух до позиції G30
6. виконати канонну команду CHANGE_TOOL з вибраною кишенею як параметром. CHANGE_TOOL виконає:
   a. генерувати швидкий перехід до TOOL_CHANGE_POSITION, якщо так встановлено в INI
   b. поставити в чергу NML-повідомлення EMC_TOOL_LOAD для task.
7. встановити параметри нумератора 5400-5413 відповідно до нового інструменту
8. сигналізувати task про припинення виклику інтерпретатора для попереднього зчитування, повертаючи INTERP_EXECUTE_FINISH, оскільки M6 є обробником черги.

Що робить task, коли бачить команду CHANGE_TOOL Знову ж таки, не набагато більше, ніж перекладання відповідальності на iocontrol шляхом надсилання йому повідомлення EMC_TOOL_LOAD та очікування, поки iocontrol зробить свою справу.

Дія Iocontrol для EMC_TOOL_LOAD

1. він стверджує штифт ”зміни інструменту”
2. він очікує, поки штифт ”зміна інструменту” стане активним
3. коли це сталося:
   a. дессерт ”зміна інструменту”

b. встановіть контакти ”tool-prep-number” та ”tool-prep-pocket” на нуль
c. виконайте функцію *load_tool()* з кишенею як параметром.

Останній крок фактично встановлює записи таблиці інструментів у структурі «emcStatus». Фактичні дії залежать від того, чи була встановлена опція RANDOM_TOOLCHANGER INI, але в кінці процесу «toolTable[0]» відображає інструмент, який наразі знаходиться у шпинделі.

Коли це сталося:

1. `iocontrol` сигналізує `task` про необхідність виконання.
2. `task` повідомляє інтерпретатору про необхідність виконати операцію *synch()*, щоб побачити, що змінилося.
3. Інтерпретатор *synch()* отримує всю необхідну інформацію з моделі світу, зокрема й таблицю змінених інструментів.

З цього моменту інтерпретатор має повне уявлення про модель світу та продовжує читати далі.

Побудова прологу та епілогу для M6 Дивіться функції Python `change_prolog` та `change_epilog` у `nc_files/remap_lib/python-stdglue/stdglue.py`.

9.6.15.13 Як працює M61 (Зміна номера інструменту)

`M61` потрібен невід’ємний параметр `Q` (номер інструмента). Якщо дорівнює нулю, це означає «вивантажити інструмент», інакше «встановити поточний номер інструмента на Q».

Створення заміни для M61 Приклад перевизначення Python для `M61` можна знайти у функції `set_tool_number` у `nc_files/remap_lib/python-stdglue/stdglue.py`.

9.6.16 Статус

1. Функція RELOAD_ON_CHANGE досить погана. Перезапустіть після зміни файлу Python.

9.6.17 Зміни

- Раніше для повернення повідомлень про помилки та завершення роботи з помилкою використовував метод «self.set_errormsg(text)», за яким слідував «return INTERP_ERROR». Зараз його замінено на просте повернення рядка з обробника Python або підпрограми oword. Це встановлює повідомленн про помилку та припиняє роботу програми. Раніше не було чіткого способу припинення роботи підпрограми Python O-word.

9.6.18 Налагодження

У розділі *[EMC]* INI-файлу параметр *DEBUG* можна змінити, щоб отримувати різні рівні налагоджувальн повідомлень під час запуску LinuxCNC з терміналу.

```
b''Pb''b''ib''b''вb''b''eb''b''нb''b''ьb'' b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b' ↩
    'дb''b''жb''b''eb''b''нb''b''нb''b''яb'', 0 b''ob''b''зb''b''нb''b''ab''b''чb''b''ab''b' ↩
    'єb'' b''вb''b''ib''b''дb''b''cb''b''yb''b''тb''b''нb''b''ib''b''cb''b''тb''b''ьb'' b' ↩
    'пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''ьb''. b''Ib''b' ↩
    'нb''b''шb''b''ib'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''дb''b' ↩
    'иb''b''вb''. b''yb'' src/emc/nml_intf/debugflags.h.
DEBUG = 0x00000002 # b''кb''b''ob''b''нb''b''фb''b''ib''b''гb''b''yb''b''pb''b''ab''b''цb'' ↩
    b''ib''b''яb''
```

```
DEBUG = 0x7FFFDEFF # b''бb''b''eb''b''зb'' b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b' ↩
    'pb''b''eb''b''тb''b''ab''b''цb''b''ib''b''їb'', oword
DEBUG = 0x00008000 # b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' py
DEBUG = 0x0000E000 # py + remap + Oword
DEBUG = 0x0000C002 # py + remap + config
DEBUG = 0x0000C100 # py + remap + b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b''pb''b''eb''b ↩
    ''тb''b''ab''b''тb''b''ob''b''pb''
DEBUG = 0x0000C140 # py + remap + b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b''pb''b''eb''b ↩
    ''тb''b''ab''b''тb''b''ob''b''pb'' + b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b' ↩
    'лb''b''eb''b''нb''b''нb''b''яb'' NML
DEBUG = 0x0000C040 # py + remap + NML
DEBUG = 0x0003E100 # py + remap + b''ib''b''нb''b''тb''b''eb''b''pb''b''пb''b''pb''b''eb''b ↩
    ''тb''b''ab''b''тb''b''ob''b''pb'' + oword + b''cb''b''иb''b''гb''b''нb''b''ab''b''лb''b ↩
    ''иb'' + namedparams
DEBUG = 0x10000000 # EMC_DEBUG_USER1 - b''тb''b''pb''b''ab''b''cb''b''yb''b''вb''b''ab''b' ↩
    'нb''b''нb''b''яb'' b''ob''b''пb''b''eb''b''pb''b''ab''b''тb''b''ob''b''pb''b''ib''b' ↩
    'вb''
DEBUG = 0x20000000 # EMC_DEBUG_USER2 - b''пb''b''eb''b''pb''b''eb''b''xb''b''ob''b''пb''b' ↩
    'лb''b''eb''b''нb''b''нb''b''яb'' b''вb'' b''дb''b''eb''b''бb''b''ab''b''гb''b''eb''b' ↩
    'pb'' Python
DEBUG = 0x10008000 # USER1, PYTHON
DEBUG = 0x30008000 # USER1,USER2, PYTHON # USER2 b''зb''b''мb''b''yb''b''cb''b''иb''b''тb'' ↩
    b''ьb'' involute b''cb''b''пb''b''pb''b''ob''b''бb''b''yb''b''вb''b''ab''b''тb''b''иb''  ↩
    b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''иb''b''тb''b''иb''b''cb''b''яb'' b' ↩
    'дb''b''ob'' pydev
DEBUG = 0x7FFFFFFF # b''Уb''b''cb''b''ib'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b' ↩
    'мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''pb''b''ob'' b''нb''b''ab''b''лb''b' ↩
    'ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb''
```

9.7 Компонент Moveoff

Компонент HAL moveoff — це метод, доступний лише в HAL, для реалізації зміщень. ВАЖЛИВІ обмеження та попередження див. на сторінці довідки (*$ man moveoff*).

Компонент moveoff використовується для зміщення положень суглобів за допомогою спеціальних з'єднань HAL. Реалізація функції зміщення під час паузи програми підтримується за допомогою відповідних з'єднань для вхідних контактів. Підтримується дев'ять суглобів.

Значення штифтів зміщення осі (offset-in-M) безперервно застосовуються (з дотриманням обмежень щодо значення, швидкості та прискорення) до вихідних штифтів (offset-current-M, pos-plusoffset-M, fb-minusoffset-M), коли обидва вхідні штифти активації (apply-offsets та move-enable) мають значення TRUE. Два входи активації об'єднуються внутрішньо. Якщо контакт apply-offsets деактивован під час застосування зміщень, встановлюється «контакт попередження» і видається повідомлення. Контакт попередження залишається TRUE, доки зміщення не будуть видалені або не буде встановлено контакт apply-offsets.

Зазвичай контакт, що вмикає рух, підключається до зовнішніх елементів керування, а контакт, що застосовує зміщення, підключається до halui.program.is-paused (для зміщень тільки під час паузи) або встановлюється на TRUE (для постійно застосовуваних зміщень).

Застосовані зміщення «автоматично повертаються» до нуля (з дотриманням обмежень), коли будь-який із активаційних входів деактивовано. Допуск нульового значення визначається значенням вхідного виводу епсилон.

Точки маршруту записуються, коли компонент moveoff увімкнено. Точки маршруту керуються за допомогою контактів waypoint-sample-secs та waypoint-threshold. Коли контакт backtrack-enable має значення TRUE, шлях автоматичного повернення слідує за записаними точками маршруту. Коли пам'ять, доступна для точок маршруту, вичерпана, зміщення заморожуються і виводиться сигнал waypoint-limit. Це обмеження застосовується незалежно від стану виводу backtrack-enable.

Вивід, що вмикає, повинен бути вимкнений, щоб дозволити повернення до початкового (незміщеного) положення.

Повернення назад через контрольні точки призводить до «уповільнення» швидкості руху, оскільки переміщення відбуваються від точки до точки з дотриманням налаштувань швидкості та прискорення. Штифти обмеження швидкості та прискорення можна динамічно регулювати, щоб постійно контролюв зміщення.

Коли backtrack-enable має значення FALSE, рух автоматичного повернення **НЕ** координується, кожна вісь повертається до нуля зі своєю швидкістю. Якщо в цьому стані потрібно контролювати траєкторію, кожну вісь слід вручну повернути до нуля перед відключенням виводу активації.

Виводи waypoint-sample-secs, waypoint-threshold та epsilon оцінюються лише тоді, коли компонент неактивний.

Вихідний контакт із застосованими зміщеннями призначений для індикації поточного стану в графічному інтерфейсі користувача, щоб можна було керувати відновленням програми. Якщо зміщення не дорівнюють нулю, коли контакт застосування зміщень не активований (наприклад, при відновленні програми під час зміщення в режимі паузи), зміщення повертаються до нуля (з урахуванням обмежень) і видається повідомлення «Помилка».



Caution
Якщо зміщення ввімкнено та застосовано, а машина з будь-якої причини вимкнена, будь-яка «зовнішня» логіка HAL, яка керує контактами ввімкнення та входами зміщення в M, відповідає за їхній стан, коли машина згодом знову вмикається.

Цей спосіб зміщення, доступний тільки в HAL, зазвичай не відомий LinuxCNC і недоступний в попередньому перегляді GUI. **Не забезпечується захист** для переміщень з зміщенням, які перевищують м’які обмеження, що управляються LinuxCNC. Оскільки м’які обмеження не дотримуються переміщення з зміщенням може зіткнутися з жорсткими обмеженнями (або **ЗАВАНТАЖЕННЯМ**, якщо немає кінцевих вимикачів). Рекомендується використовувати входи offset-min-M та offset-max-M для обмеження переміщення. Спрацьовування жорсткого обмеження вимкне машину — див. **Застереження** вище.

Значення зміщення в M можна встановити за допомогою налаштувань файлу INI, керувати ними за допомогою графічного інтерфейсу користувача або управляти ними за допомогою інших компонентів HAL та з’єднань. Фіксовані значення можуть бути доречними в простих випадках, коли напрямок і величина зміщення чітко визначені, але для повернення зміщення до нуля необхідний метод управління для деактивації виводу активації. Графічні інтерфейси можуть надавати користувачам засоби для встановлення, збільшення, зменшення та накопичення значень зміщення для кожної осі, а також можуть встановлювати значення зміщення в M до нуля перед деактивацією виводу активації.

Значення за замовчуванням для accel, vel, min, max, epsilon, waypoint-sample-secs та waypoint-threshold можуть бути непридатними для певних застосувань. Цей компонент HAL не враховує обмеження, встановлені в інших місцях LinuxCNC. Користувачі повинні перевірити його використання в симуляторі та ознайомитися з усіма ризиками перед використанням на апаратному забезпеченні.

Конфігурації симулятора, що демонструють компонент та графічний інтерфейс (moveoff_gui), розташовані тут:

- configs/sim/axis/moveoff (axis-ui)
- configs/sim/touchy/ngcgui (touchy-ui)

9.7.1 Зміна існуючої конфігурації

Для адаптації існуючої конфігурації до використання компонента moveoff можна використовувати файл HAL, що надається системою (LIB:hookup_moveoff.tcl). Додаткові налаштування файлу INI підтримують використання простого графічного інтерфейсу (moveoff_gui) для керування зміщеннями.

Коли системний HAL-файл (LIB:hookup_moveoff.tcl) правильно вказано у файлі конфігурації INI, він:

1. Від’єднайте оригінальні з’єднання контактів joint.N.motor-pos-cmd та joint.N.motor-pos-fb
2. Завантажити (loadrt) компонент moveoff (використовуючи ім’я mv) з налаштуванням особистості, щоб врахувати всі осі, визначені у файлі INI
3. Додати (addf) функції компонента moveoff у потрібній послідовності
4. Знову підключіть контакти joint.N.motor-pos-cmd та joint.N.motor-pos-fb, щоб використовувати компонент moveoff
5. Встановіть робочі параметри та обмеження компонента Moveoff для кожної осі відповідно до додаткових налаштувань INI-файлу

Примітка: Програма moveoff_gui підтримує конфігурації, які використовують відомі кінематичні модулі з KINEMATICS_TYPE=KINEMATICS_IDENTITY. Підтримувані модулі включають: trivkins. З ідентифікаційними кінематичними модулями moveoff_gui присвоює кожній осі, вказаній за допомогою параметра командного рядка «-axes axisnames», відповідне з’єднання.

Змініть існуючу конфігурацію наступним чином:

Переконайтеся, що в файлі INI є запис для `[HAL]HALUI`, і створіть новий запис `[HAL] HALFILE` для LIB:hookup_moveoff.tcl. Запис для LIB:hookup_moveoff.tcl повинен слідувати за всіма записами `HALFILE=` для файлів HAL, які з’єднують контакти для `joint.N.motor-pos-cmd`, `joint.N.motor-pos-fb` та будь-яких компонентів, підключених до цих контактів (наприклад, компонентів PID та енкодера в сервосистемі).

```
[HAL]
HALUI   = halui
HALFILE = existing_configuration_halfile_1
...
HALFILE = existing_configuration_halfile_n
HALFILE = LIB:hookup_moveoff.tcl
```

Додайте записи INI-файлу для налаштувань кожної осі, що використовується (якщо запис не визначений, буде використано відповідний запис із розділу [AXIS_n], а якщо запис не знайдено, буде використано значення за замовчуванням компонента moveoff).

Note
Використання значень компонентів за замовчуванням або значень розділу [AXIS_n] для налаштувань зміщення для кожної осі НЕ рекомендується.

```
[MOVEOFF_n]
MAX_LIMIT =
MIN_LIMIT =
MAX_VELOCITY =
MAX_ACCELERATION =
```

Додати записи INI-файлу для налаштувань компонента Moveoff (не використовувати значення Moveoff за замовчуванням):

```
[MOVEOFF]
EPSILON =
WAYPOINT_SAMPLE_SECS =
WAYPOINT_THRESHOLD =
```

Moveoff_gui використовується для встановлення додаткових необхідних підключень та надання спливаючого графічного інтерфейсу для:

1. Надайте кнопку-перемикач для ввімкнення/вимкнення зміщень
2. Надайте кнопку-перемикач для ввімкнення/вимкнення зворотного відстеження
3. Забезпечити кнопки керування для збільшення/зменшення/обнулення кожного зміщення осі
4. Відображення поточного значення зміщення кожної осі
5. Відображення поточного стану зміщення (вимкнено, активне, видалено тощо)

Надані кнопки управління є опціональними залежно від стану виводу move-enable компонента moveoff. Дисплей та елементи управління для ввімкнення зміщення надаються, якщо вивід mv.move-enable НЕ підключений під час запуску moveoff_gui. У цьому випадку moveoff_gui керує контактом move-enable компонента moveoff (названого mv.move-enable), а також зміщеннями (mv.move-offset-in-M) і увімкненням зворотного відстеження (mv.backtrack-enable)

Якщо контакт mv.move-enable підключений під час запуску moveoff_gui, moveoff_gui буде відображати інформацію, але НЕ буде виконувати функції керування. Цей режим підтримує конфігурації, що використовують колесо прокрутки або інші методи керування вхідними даними зміщення та контактами увімкнення (mv.offset-in-M, mv.move-enable, mv.backtrack-enable).

moveoff_gui встановлює необхідні з'єднання для виводів компонента moveoff: mv.power_on та mv.apply-offsets. Вивід mv.power_on підключається до виводу motion.motion-enabled (за необхідності автоматично створюється новий сигнал). Контакт mv.apply-offsets підключається до halui.program.is-paused або встановлюється на 1 залежно від параметра командного рядка -mode [onpause | always]. За необхідності автоматично створюється новий сигнал.

Щоб використовувати moveoff_gui, додайте запис у розділі [APPLICATIONS] INI-файлу наступним чином:

```
[APPLICATIONS]
# b''Пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'': b''мb''b''ob''b''жb''b''eb'' b' ↩
    'зb''b''нb''b''ab''b''дb''b''ob''b''бb''b''иb''b''тb''b''иb''b''cb''b''яb'' b''зb''b' ↩
    'ab''b''тb''b''pb''b''иb''b''мb''b''кb''b''ab'' (b''вb''b''кb''b''ab''b''зb''b''ab''b' ↩
    'нb''b''ab'' b''вb'' b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''ab''b''хb''), b''яb''b ↩
    ''кb''b''щb''b''ob'' b''зb''b'''b''b''єb''b''дb''b''нb''b''ab''b''нb''b''нb''b''яb''
# b''cb''b''тb''b''вb''b''ob''b''pb''b''юb''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb''b' ↩
    'ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' HAL-b''фb''b''ab'' ↩
    b''йb''b''лb''b''ib''b''вb'' PostGui ([HAL]POSTGUI_HALFILE=)
DELAY = 0
APP = moveoff_gui option1 option2 ...
```

Коли файл HAL LIB:hookup_moveoff.tcl використовується для завантаження та підключення компонент moveoff, контакт mv.move-enable не буде підключений, і будуть використовуватися локальні елементи керування, що надаються moveoff_gui. Це найпростіший метод тестування або демонстрації компонента moveoff під час модифікації існуючої конфігурації INI.

Щоб увімкнути зовнішні елементи керування під час використання дисплея moveoff_gui для значень зміщення та стану, файли HAL, що відповідають LIB:hookup_moveoff.tcl, повинні мати додаткові з'єднання. Наприклад, надані демонстраційні конфігурації (configs/sim/axis/moveoff/*.ini) використовують простий системний файл HAL (з назвою LIB:moveoff_external.hal) для підключення контактів mv.move-enable, mv.offset-in-M та mv.bactrack-enable до сигналів:

```
[HAL]
HALUI = halui
...
HALFILE = LIB:hookup_moveoff.tcl
HALFILE = LIB:moveoff_external.hal
```

З'єднання, що здійснюються LIB:moveoff_external.hal (для тривісної конфігурації), такі:

```
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
```

Ці сигнали (external_enable, external_offset_M, external_backtrack_en) можуть управлятися наступними HALFILES (включаючи POSTGUI_HALFILEs) для забезпечення індивідуального контролю компонента під час використання дисплея moveoff_gui для поточних значень зміщення та стану зміщення.

Moveoff_gui налаштовується за допомогою параметрів командного рядка. Докладніше про роботу moveoff_gui див. на сторінці довідки:

```
$ man moveoff_gui
```

Щоб отримати короткий перелік параметрів командного рядка для moveoff_gui, скористайтеся параметром довідки командного рядка:

```
$ moveoff_gui --help

b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
moveoff_gui [Options]

b''Ob''b''пb''b''цb''b''ib''b''їb'':
    [--help | -? | -- -h ]  (b''Цb''b''eb''b''йb'' b''тb''b''eb''b''кb''b''cb''b''тb'')

    [-mode [onpause | always]]  (b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b ↩
        ''yb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'': b''нb''b''ab'' b''пb''b''ab''b' ↩
        'yb''b''зb''b''ib'')
                                    (onpause: b''пb''b''ob''b''кb''b''ab''b''зb''b''yb''b' ↩
                                        'вb''b''ab''b''тb''b''иb'' b''гb''b''pb''b''ab''b' ↩
                                        'фb''b''ib''b''чb''b''нb''b''иb''b''йb'' b''ib''b' ↩
                                        'нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b' ↩
                                        'cb'', b''кb''b''ob''b''лb''b''иb'' b''пb''b''pb''b' ↩
                                        'ob''b''гb''b''pb''b''ab''b''мb''b''ab'' b''пb''b' ↩
                                        'pb''b''иb''b''зb''b''yb''b''пb''b''иb''b''нb''b''eb'' ↩
                                        b''нb''b''ab'')
                                    (b''зb''b''ab''b''вb''b''жb''b''дb''b''иb'': b''пb''b' ↩
                                        'ob''b''кb''b''ab''b''зb''b''yb''b''вb''b''ab''b''тb'' ↩
                                        b''иb'' b''гb''b''pb''b''ab''b''фb''b''ib''b''чb''b' ↩
                                        'нb''b''иb''b''йb'' b''ib''b''нb''b''тb''b''eb''b' ↩
                                        'pb''b''фb''b''eb''b''йb''b''cb'' b''зb''b''ab''b' ↩
                                        'вb''b''жb''b''дb''b''иb'')

    [-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))
```

```
b''Bb''b''ab''b''pb''b''ib''b''ab''b''нb''b''тb''b''иb'' b''дb''b''лb''b''яb'' b''ob''b' ↩
    'cb''b''ob''b''бb''b''лb''b''иb''b''вb''b''иb''b''xb'' b''вb''b''иb''b''пb''b''ab''b' ↩
    'дb''b''кb''b''ib''b''вb'':
    [-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))

b''Пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'': b''Яb''b''кb''b''щb''b''ob'' b' ↩
    'кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb'' b''уb''b''вb''b''ib''b''мb''b''кb''b' ↩
    'нb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b' ↩
    'eb''b''нb''b''нb''b''яb'' (mv.move-enable) b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b' ↩
    'чb''b''eb''b''нb''b''ob'', b''кb''b''ob''b''лb''b''иb''
      moveoff_gui b''зb''b''ab''b''пb''b''уb''b''щb''b''eb''b''нb''b''ob'', b''пb''b''ob''b ↩
          ''тb''b''pb''b''ib''b''бb''b''нb''b''ib'' b''зb''b''ob''b''вb''b''нb''b''ib''b' ↩
          'шb''b''нb''b''ib'' b''eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb''b''иb'' b' ↩
          'кb''b''eb''b''pb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'', b''ib'' b''лb''b' ↩
          'иb''b''шb''b''eb''
      b''пb''b''eb''b''pb''b''eb''b''дb''b''бb''b''ab''b''чb''b''eb''b''нb''b''ib'' b''дb'' ↩
          b''иb''b''cb''b''пb''b''лb''b''eb''b''їb''.
```

9.8 Автономний перекладач

Окремий інтерпретатор RS274 доступний для використання через командний рядок.

9.8.1 Застосування

```
b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'': rs274 ↩
     [-p interp.so] [-t tool.tbl] [-v var-file.var] [-n 0|1|2]
          [-b] [-s] [-g] [input file [output file]]

    -p: b''Bb''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''ib''b''нb''b''тb''b''eb''b' ↩
        'pb''b''пb''b''pb''b''eb''b''тb''b''ab''b''тb''b''ob''b''pb'', b''яb''b''кb''b''иb'' ↩
        b''йb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b' ↩
        'cb''b''тb''b''ob''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb''b''cb''b''яb''.
    -t: b''Bb''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''фb''b''ab''b''йb''b''лb'' .tbl ↩
         (b''тb''b''ab''b''бb''b''лb''b''иb''b''цb''b''яb'' b''ib''b''нb''b''cb''b''тb''b' ↩
        'pb''b''уb''b''мb''b''eb''b''нb''b''тb''b''ib''b''вb''), b''яb''b''кb''b''иb''b' ↩
        'йb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb'' ↩
        b''тb''b''ob''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb''b''cb''b''яb''.
    -v: b''Bb''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''фb''b''ab''b''йb''b''лb'' .var ↩
         (b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb''), b''яb''b''кb''b' ↩
        'иb''b''йb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb'' ↩
        b''cb''b''тb''b''ob''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb''b''cb''b''яb''.
    -n: b''Bb''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''pb''b''eb''b''жb''b''иb''b' ↩
        'мb'' b''пb''b''pb''b''ob''b''дb''b''ob''b''вb''b''жb''b''eb''b''нb''b''нb''b''яb'':
           0: b''пb''b''pb''b''ob''b''дb''b''ob''b''вb''b''жb''b''иb''b''тb''b''иb''
```

```
          1: b''пв''b''eb''b''pb''b''eb''b''йв''b''тв''b''ив'' b''вв'' b''pb''b''eb''b' ↩
              'жв''b''ив''b''мв'' MDI
          2: b''зв''b''yв''b''пв''b''ив''b''нв''b''ив''b''тв''b''ив'' (b''зв''b''ab'' b' ↩
              'зв''b''ab''b''мв''b''ob''b''вв''b''чв''b''yв''b''вв''b''ab''b''нв''b''нв''b' ↩
              'яв''b''мв'')
    -b: b''Ув''b''вв''b''iв''b''мв''b''кв''b''нв''b''yв''b''тв''b''ив''/b''вв''b''ив''b' ↩
        'мв''b''кв''b''нв''b''yв''b''тв''b''ив'' b''пв''b''pb''b''ab''b''пв''b''ob''b''pb''b ↩
        ''eb''b''цв''b''ьв'' «b''вв''b''ив''b''дв''b''ab''b''лв''b''eb''b''нв''b''нв''b' ↩
        'яв'' b''бв''b''лв''b''ob''b''кв''b''yв''» (b''зв''b''ab'' b''зв''b''ab''b''мв''b' ↩
        'ob''b''вв''b''чв''b''yв''b''вв''b''ab''b''нв''b''нв''b''яв''b''мв'': b''Bb''b''Ив'' ↩
        b''Mb''b''Kb''b''Hb''b''Eb''b''Hb''b''Ob'')
    -s: b''Ув''b''вв''b''iв''b''мв''b''кв''b''нв''b''yв''b''тв''b''ив''/b''вв''b''ив''b' ↩
        'мв''b''кв''b''нв''b''yв''b''тв''b''ив'' b''пв''b''pb''b''ab''b''пв''b''ob''b''pb''b ↩
        ''eb''b''цв''b''ьв'' «b''дв''b''pb''b''yв''b''кв'' b''cв''b''тв''b''eb''b''кв''b' ↩
        'ab''» (b''зв''b''ab'' b''зв''b''ab''b''мв''b''ob''b''вв''b''чв''b''yв''b''вв''b' ↩
        'ab''b''нв''b''нв''b''яв''b''мв'': b''Bb''b''Ив''b''Mb''b''Kb''b''Hb''b''Eb''b''Hb'' ↩
        b''Ob'')
    -g: b''yв''b''вв''b''iв''b''мв''b''кв''b''нв''b''yв''b''тв''b''ив''/b''вв''b''ив''b' ↩
        'мв''b''кв''b''нв''b''yв''b''тв''b''ив'' b''пв''b''pb''b''ab''b''пв''b''ob''b''pb''b ↩
        ''eb''b''цв''b''ьв'' «b''пв''b''eb''b''pb''b''eb''b''йв''b''тв''b''ив'' (b''пв''b' ↩
        'ab''b''кв''b''eb''b''тв''b''нв''b''ив''b''йв'' b''pb''b''eb''b''жв''b''ив''b''мв'') ↩
        » (b''зв''b''ab'' b''зв''b''ab''b''мв''b''ob''b''вв''b''чв''b''yв''b''вв''b''ab''b' ↩
        'нв''b''нв''b''яв''b''мв'': b''Bb''b''Ив''b''Mb''b''Kb''b''Hb''b''Eb''b''Hb''b' ↩
        'Ob'')
    -i: b''вв''b''кв''b''ab''b''зв''b''ab''b''тв''b''ив'' b''фв''b''ab''b''йв''b''лв'' .ini ↩
         (b''зв''b''ab'' b''зв''b''ab''b''мв''b''ob''b''вв''b''чв''b''yв''b''вв''b''ab''b' ↩
        'нв''b''нв''b''яв''b''мв'': b''фв''b''ab''b''йв''b''лв'' ini b''вв''b''iв''b''дв''b' ↩
        'cв''b''yв''b''тв''b''нв''b''iв''b''йв'')
    -T: b''вв''b''ив''b''кв''b''лв''b''ив''b''кв''b''ab''b''тв''b''ив'' task_init()
    -l: b''вв''b''кв''b''ab''b''зв''b''ab''b''тв''b''ив'' b''pb''b''iв''b''вв''b''eb''b' ↩
        'нв''b''ьв'' b''жв''b''yв''b''pb''b''нв''b''ab''b''лв''b''yв'' (b''зв''b''ab'' b' ↩
        'зв''b''ab''b''мв''b''ob''b''вв''b''чв''b''yв''b''вв''b''ab''b''нв''b''нв''b''яв''b' ↩
        'мв'': -1)
```

9.8.2 Приклад

Щоб побачити результат роботи циклу, наприклад, ми можемо запустити rs274 на наступному файлі і побачити, що цикл ніколи не закінчується. Щоб вийти з циклу, використовуйте Ctrl Z. Для запуску прикладу необхідні наступні два файли.

test.ngc

```
#<test> = 123.352

o101 while [[#<test> MOD 60 ] NE 0]
(debug,#<test>)
    #<test> = [#<test> + 1]
o101 endwhile'

M2
```

test.tbl

```
T1 P1 Z0.511 D0.125 ;1/8 end mill
T2 P2 Z0.1 D0.0625 ;1/16 end mill
T3 P3 Z1.273 D0.201 ;#7 tap drill'
```

команда

```
rs274 -g test.ngc -t test.tbl
```

9.9 Зміщення зовнішніх осей

Зовнішні зміщення осей підтримуються під час телеопераційних (світових) переміщень і скоординован
(G-код) рухів. Зсуви зовнішніх осей вмикаються для кожної осі окремо за допомогою налаштувань
файлу INI і динамічно контролюються вхідними контактами INI. Інтерфейс INI схожий на той,
що використовується для поступального руху колеса. Цей тип інтерфейсу зазвичай реалізується
за допомогою ручного генератора імпульсів (mpg), підключеного до компонента INI енкодера,
який підраховує імпульси.

9.9.1 Налаштування INI-файлу

Для кожної літери осі (**L** у xyzabcuvw):

```
[AXIS_L]OFFSET_AV_RATIO = value (b''кb''b''eb''b''pb''b''yb''b''вb''b''аb''b''нb''b''нb''b' ↩
    'яb'' b''пb''b''pb''b''иb''b''cb''b''кb''b''оb''b''pb''b''eb''b''нb''b''нb''b''яb''b' ↩
    'мb''/b''шb''b''вb''b''иb''b''дb''b''кb''b''іb''b''cb''b''тb''b''юb'' b''дb''b''лb''b' ↩
    'яb'' b''зb''b''оb''b''вb''b''нb''b''іb''b''шb''b''нb''b''іb''b''хb'' b''зb''b''мb''b' ↩
    'іb''b''щb''b''eb''b''нb''b''ьb'')
```

1. Допустимі значення: 0 <= значення <= 0,9
2. Заборонені значення замінюються на 0.1 з повідомленням на стандартний вивід
3. Значення за замовчуванням: 0 (вимикає зовнішнє зміщення).
   Наслідок: пропущено. [AXIS_L]OFFSET_AV_RATIO вимикає зовнішнє зміщення для осі.
4. Якщо не дорівнює нулю, OFFSET_AV_RATIO (**r**) коригує звичайну (планову) максимальну швидкість та прискорення, щоб зберегти обмеження [AXIS_L]:

```
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 Піни HAL

9.9.2.1 Штифти HAL для руху по осях

Для кожної літери осі (***L*** у xyzabcuvw)

1. **axis.*L*.eoffset-enable** Input(bit): увімкнути
2. **axis.*L*.eoffset-scale** Input(float): коефіцієнт масштабування
3. **axis.*L*.eoffset-counts** Вхід(s32): вхід до регістра підрахунків
4. **axis.*L*.eoffset-clear** Input(bit): очистити запитуваний зсув
5. **axis.*L*.eoffset** Output(float): поточне зовнішнє зміщення
6. **axis.*L*.eoffset-request** Output(float): запитуваний зовнішній зсув

9.9.2.2 Інші контакти HAL для руху

1. **motion.eoffset-active** Вихід (біт): застосовано ненульові зовнішні зміщення
2. **motion.eoffset-limited** Вихід (біт): рух заборонено через програмне обмеження

9.9.3 Застосування

Вхідні контакти HAL осі (enable, scale, counts) аналогічні контактам, що використовуються для джогінгу колеса.

9.9.3.1 Обчислення зміщення

У кожному сервоперіоді контакт «axis.*L*.eoffset-counts» порівнюється з його значенням у попередньому періоді. Збільшення або зменшення (позитивне або негативне дельта) виводу «axis.*L*.eoffset-counts» множиться на поточне значення виводу «axis.*L*.eoffset-scale». Цей добуток накопичується у внутрішньому регістрі та експортується до виводу HAL «axis.*L*.eoffset-request». Регістр накопичення обнуляється при кожному ввімкненні машини.

Запитане значення зміщення використовується для планування руху для зміщення, яке застосовується до координати «*L*» і представлене контактом HAL «axis.*L*.eoffset». Запланований рух відповідає виділеним обмеженням швидкості та прискорення і може бути обмежений, якщо чистий рух (зміщення плюс телеоператорне переміщення або скоординований рух) досягає м'якого обмеження для координати «*L*».

Для багатьох застосувань контакт «axis.*L*.eoffset-scale» є постійним, а відповідь мережі «axis.*L*.eoffset-request» на «axis.*L*.eoffset-counts» еквівалентна добутку накопиченого значення «axis.*L*.eoffset-counts» та (постійних) значень виводу «axis.*L*.eoffset-scale».

9.9.3.2 Вимкнення/увімкнення машини

Коли машина вимкнена, **поточна позиція** із **зовнішніми зміщеннями зберігається**, щоб не було неочікуваного руху під час вимкнення або ввімкнення.

Під час кожного запуску (увімкнення машини) внутрішній регістр лічильників для кожного виводу HAL *axis*.L.*eoffset-counts* обнуляється, а відповідний вихідний вивод HAL *axis*.L.*eoffset* скидається до нуля.

Іншими словами, зовнішні зміщення **визначаються як НУЛЬ при кожному запуску** (увімкненні машини) незалежно від значення контактів «axis.*L*.eoffset-counts». Щоб уникнути плутанини, рекомендується встановлювати всі контакти «axis.*L*.eoffset-counts» на нуль, коли машина вимкнена.

9.9.3.3 М'які обмеження

Зовнішні переміщення з зміщенням осі плануються незалежно з налаштуваннями швидкості та прискорення, що вказані в *[AXIS_L]OFFSET_AV_RATIO*. Переміщення зі зміщенням не координується з телеоперативними імпульсами та скоординованими (G-код) переміщеннями. Під час телеопераційно переміщення та скоординованого (G-код) руху м'які обмеження осі («[AXIS_*L*]MIN_LIMIT,MAX_LIMIT») обмежують рух осі.

Коли застосовуються зовнішні зміщення і рух досягає м'якого обмеження (завдяки збільшенню зовнішнього зміщення, телеоперативному переміщенню або скоординованому руху), виводиться сигнал HAL-виводу «motion.eoffset-limited», і значення осі утримується номінально до м'якого обмеження. Цей HAL-вивід може використовуватися пов'язаною логікою HAL для обрізання додаткових значень eoffset або для зупинки машини (наприклад, підключення до «halui.machine.off»). Якщо вісь переміщується в межах м'якої межі, контакт «motion.eoffset-limited» скидається.

При роботі з м’яким обмеженням під час скоординованого руху, який продовжує змінювати заплановане значення осі, вихідний контакт HAL «axis.*L*.eoffset» буде вказувати поточне зміщення — відстань, необхідну для досягнення обмеження, замість обчисленого запиту на зміщення. Це вказане значення буде змінюватися у міру зміни запланованого значення осі.

Контакт HAL «axis.*L*.eoffset-request» вказує поточне запитуване зміщення, яке є добутком внутрішньог регістра підрахунку та масштабу eoffset. Як правило, значення виводу «axis.*L*.eoffset» відстає від значення «axis.*L*.eoffset-request», оскільки зовнішнє зміщення підлягає обмеженню прискорення. При роботі в режимі м’якого обмеження додаткові оновлення «axis.*L*.eoffset-counts» продовжуватимуть впливати на запитуване зовнішнє зміщення, що відображається у виводі HAL «axis.*L*.eoffset-request».

При телеуправлінні з увімкненими зовнішніми зміщеннями **та** застосованими ненульовими значенням досягнення м’якого обмеження зупинить рух у відповідній осі **без інтервалу уповільнення**. Аналогічно, під час скоординованого руху з увімкненими зовнішніми зміщеннями, досягнення м’якого обмеження зупинить рух без фази уповільнення. У цьому випадку не має значення, чи зміщення дорівнюють нулю.

Коли рух зупиняється без фази уповільнення, **можуть бути порушені обмеження прискорення системи**, що може призвести до: 1) помилки слідування (та/або удару) для системи сервомотора, 2) втрати кроків для системи крокового двигуна. Загалом рекомендується застосовувати зовнішні зміщення таким чином, щоб уникнути наближення до м’яких обмежень.

9.9.3.4 Нотатки

Зовнішні зміщення застосовуються до літер координат осей (xyzabcuvw). Усі з’єднання мають бути виведені в початкове положення, перш ніж будуть враховані зовнішні зміщення осей.

Якщо вивід HAL «axis.*L*.eoffset-enable» скидається, коли його зміщення не дорівнює нулю, зміщення зберігається. Зміщення можна очистити за допомогою:

1. перемикач «Вимкнено/Увімкнено машину»
2. повторна активація виводу ввімкнення та збільшення/зменшення значення виводу HAL *axis*.L.*eoffs counts* для повернення зміщення до нуля.
3. імпульсація виводу HAL *axis*.L.*eoffset-clear*

Зовнішні зміщення призначені для використання з «малими» зміщеннями, що застосовуються в межах м’яких меж.

М’які обмеження дотримуються як для телеопераційного переміщення, так і для скоординованого руху, коли застосовуються зовнішні зміщення. Однак, коли під час скоординованого руху досягається м’яке обмеження, зменшення зовнішнього зміщення, що порушує обмеження, **може не віддалити** від м’якого обмеження, **якщо запланований рух продовжується в тому ж напрямку**. Така ситуація може виникнути, оскільки швидкість виправлення зміщення (встановлена параметром «[AXIS_L]OFFSET_AV_RATIO») може бути меншою за протилежну заплановану швидкість руху. У таких випадках **призупинення** (або зупинка) запланованого скоординованого руху дозволить відійти від м’якого обмеження, коли будуть внесені виправлення у зовнішнє зміщення, що спричиняє порушення.

9.9.3.5 УВАГА

Використання зовнішніх зміщень може істотно вплинути на рух машини. Управління зовнішніми зміщеннями за допомогою компонентів і з’єднань HAL, а також будь-яких пов’язаних з ними користувацьких інтерфейсів, повинно бути ретельно розроблено і протестовано перед впровадженням.

9.9.4 Пов’язані компоненти HAL

9.9.4.1 eoffset_per_angle.comp

Компонент для обчислення зовнішнього зміщення від функції на основі виміряного кута (поворотна координата або шпиндель). Див. сторінку довідки для отримання детальної інформації (**$ man eoffset_per_angle**).

9.9.5 Тестування

Можливість зміщення зовнішньої осі вмикається шляхом додавання налаштування *[AXIS_L]* для кожної осі-кандидата. Наприклад:

```
[AXIS_Z]
OFFSET_AV_RATIO = 0.2
```

Для тестування зручно імітувати інтерфейс джог-колеса за допомогою графічного інтерфейсу **sim_pin**. Наприклад, у терміналі:

```
$ sim_pin axis.z.eoffset-enable axis.z.eoffset-scale axis.z.eoffset-counts
```

Використання зовнішніх зміщень полегшується завдяки відображенню інформації, пов’язаної з поточними зміщеннями: поточне значення eoffset і запитуване значення eoffset, pos-cmd осі та (для машини з ідентичною кінематикою) відповідне pos-cmd з’єднання двигуна і зміщення двигуна. Надана конфігурація sim (див. нижче) демонструє приклад панелі PyVCP для графічного інтерфейсу AXIS.

За відсутності власного дисплея, **halshow** можна запустити як допоміжну програму з власним списком спостереження.

Приклад налаштувань INI-файлу для імітації з’єднань eoffset контактів HAL та відображення інформації про eoffset для осі z (для тотожної кінематики з 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
```

Де знаходиться файл z.halshow (у каталозі конфігурації):

```
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 Приклади

Надані конфігурації моделювання демонструють використання зовнішніх зміщень, щоб забезпечити відправну точку для налаштування користувачем реального обладнання.

Конфігурації симулятора використовують налаштування INI «[HAL]HALFILE = LIB:basic_sim.tcl» для конфігурації всіх стандартних підключень HAL для осей, зазначених у налаштуванні INI

«[TRAJ]COORDINATES=». Логіка HAL, необхідна для демонстрації функціональності зовнішнього зміщення, та підключення виводів GUI HAL для панелі PyVCP виконані в окремих файлах HAL. Конфігурація, що не є симуляцією, повинна замінити елемент «LIB:basic_sim.tcl» HALFILEs, відповідни для машини. Надані файли PyVCP (.hal та .xml) можуть бути відправною точкою для інтерфейсів GUI, специфічних для конкретних додатків.

9.9.6.1 eoffsets.ini

Конфігурація sim, *sim/configs/axis/external_offsets/eoffsets.ini* демонструє декартову систему координат XYZ з елементами керування для ввімкнення зовнішніх зміщень на будь-якій осі.

Передбачено дисплеї для відображення всіх важливих значень положення та зміщення.

Графічний інтерфейс sim_pin надає елементи керування для виводів зміщення осей: eoffset-scale та eoffset-counts (через сигнал e:<L>counts), eoffset-clear (через сигнал e:clearall)

Скрипт (eoffsets_monitor.tcl) використовується для встановлення виводів *axis.L.counts* у нуль під час вимкнення машини.

9.9.6.2 jwp_z.ini

Конфігурація симулятора *sim/configs/axis/external_offsets/jwp_z.ini* демонструє можливість поетапного переміщення під час паузи для однієї координати (Z):

На панелі передбачені світлодіодні індикатори для відображення важливих елементів стану.

Передбачено елементи керування для встановлення коефіцієнта масштабування eoffset та для збільшення/зменшення/очистки лічильників eoffset.

9.9.6.3 dynamic_offsets.ini

Цей конфігураційний файл симулятора *sim/configs/axis/external_offsets/dynamic_offsets.ini* демонструє динамічно застосовувані зміщення шляхом підключення синусоїдальної форми хвилі до входів зовнішнього зміщення координати z.

На панелі передбачені світлодіодні індикатори для відображення важливих елементів стану.

Передбачено елементи керування для зміни налаштувань INI-файлу для максимальної швидкості та максимального прискорення осі Z.

Передбачено елементи керування для налаштування параметрів генератора сигналів.

Запускається застосунок halscope для відображення застосованої форми хвилі, реакції зміщення та реакції на команду двигуна.

Note
Зміни максимального прискорення та максимальної швидкості координати z не підтверджуються під час виконання програми.

9.9.6.4 opa.ini (eoffset_per_angle)

Конфігурація opa.ini використовує INI-компонент eoffset_per_angle (**$ man eoffset_per_angle**) для демонстрації машини XZC з функціональними зміщеннями, обчисленими з координати C (кут) і застосованими до поперечної координати (X). Обчислення зміщення базуються на заданому опорному радіусі, який зазвичай встановлюється програмою (або MDI) командою M68 для керування виводом **motion.analog-out-NN**.

На панелі передбачені світлодіодні індикатори для відображення важливих елементів стану.

Надаються функції для внутрішніх і зовнішніх багатокутників (nsides >= 3), синусоїдальних і прямокутних хвиль. Частоту функцій можна помножити за допомогою виводу fmul, а амплітуду змінити за допомогою виводу rfrac (частка опорного радіуса).

Передбачено елементи керування для запуску/зупинки зміщених осцилограм та для встановлення типу функції та її параметрів.

9.10 Інтерфейс бази даних інструментів

Дані про інструменти зазвичай описуються у файлі таблиці інструментів, який визначається налаштуванням inifile: [EMCIO]TOOL_TABLE=tooltable_filename. Файл таблиці інструментів складається з текстового рядка для кожного доступного інструменту, в якому описуються параметри інструменту, див. Формат таблиці інструментів.

Інтерфейс бази даних інструментів надає альтернативний метод отримання даних про інструменти через окрему програму, яка керує базою даних інструментів.

9.10.1 Інтерфейс

9.10.1.1 Налаштування INI-файлу

Налаштування INI-файлу дозволяють (необов'язково) роботу програми бази даних інструментів, наданої користувачем:

```
[EMCIO]
DB_PROGRAM = db_program [args]
```

Якщо його включено, **db_program** визначає шлях до виконуваного файлу програми, наданого користувачем, яка надає дані інструментів. Можна включити до 10 аргументів, розділених пробілами, та передати їх до **db_program** під час запуску.

Note
Налаштування INI-файлу для [EMCIO]TOOL_TABLE ігноруються, якщо вказано **db_program**.

Note
db_program може бути реалізований на будь-якій мові, яка в даний час підтримується в LinuxCNC (наприклад, скрипти BASH, скрипти Python або Tcl, програми C/C++), якщо він відповідає повідомленням інтерфейсу, отриманим на stdin і відповіли на stdout. **db_program** може управляти даними з плоского файлу, реляційної бази даних (наприклад, SQLite) або інших джерел даних.

9.10.1.2 db_program операція (v2.1)

Коли вказано **db_progam**, операції виконуються наступним чином:

1. Під час запуску LinuxCNC запускає **db_program** та підключається до її stdin та stdout.
2. **db_program** повинен відповісти, написавши однорядкове підтвердження, що складається з рядка версії (наприклад, «v2.1»). Жодні інструменти не будуть доступні, якщо версія не сумісна з версією інтерфейсу бази даних LinuxCNC.

3. Після успішного підтвердження LinuxCNC видає команду «g» (**get**) для запиту всіх інструментів. **db_program** повинен відповісти послідовністю відповідей, щоб ідентифікувати кожен доступний інструмент. Формат текстової відповіді ідентичний формату текстового рядка, що використовується в звичайних файлах таблиць інструментів. Остаточна відповідь «FINI» завершує відповідь.
4. Потім **db_program** переходить у цикл очікування події, щоб отримати команди, які вказують на те, що дані інструменту були змінені LinuxCNC. Зміни даних інструменту включають:
   - a) spindle loading(T*n* M6)/unloading(T0 M6)
   - b) зміни параметрів інструменту (наприклад, G10L1P*n*)
   - c) заміни інструментів (M61Q*n*).

Коли відбуваються зміни даних інструменту, LinuxCNC надсилає команду до **db_program**, що складається з ідентифікаційної літери команди, за якою йде повний або скорочений рядок даних інструменту. **db_program** повинен відповісти, щоб підтвердити отримання. Якщо відповідь містить текст «NAK», на stdout виводиться повідомлення, але виконання продовжується. Повідомлення «NAK» означає відсутність синхронізації між **db_program** і LinuxCNC — супровідний текст повинен вказувати на причину несправності.

Команди, що видаються для зміни даних інструменту:

- ”p” вводить зміни даних, спричинені G-кодами G10L1, G10L10, G10L11. Рядок даних інструменту включатиме всі елементи текстового рядка таблиці інструментів.
- ”l” spindle_load (T*n*M6). Рядок даних інструменту містить лише елементи *T* та *P*, що визначають відповідний номер інструмента та номер гнізда.
- ”u” розвантаження_шпинделя (T0M6). Рядок даних інструменту містить лише елементи *T* та *P*, що ідентифікують відповідний номер інструмента та номер гнізда.

Note
Коли за допомогою [EMCIO]RANDOM_TOOL_CHANGER=0 (за замовчуванням) задано змінювач інструментів NON_RANDOM, команда spindle_load, що видається для T*n*M6 (або M61Q*n*), має вигляд: «l T*n* P0» (кишеня 0 є шпинделем). Команда spindle_unload, що видається для T0M6, є *u T0 P0*.

Note
Коли за допомогою [EMCIO]RANDOM_TOOL_CHANGER=1 задано випадковий змінювач інструментів, при кожній заміні інструменту видається пара команд spindle_unload/spindle_load. Пара команд, що видаються для T*n*M6 (або M61Q*n*), має вигляд «u T*u* P*m*», за яким слідує «l T*n* P0», де *u* — це поточний інструмент, який потрібно відправити в кишеню *m*, а *n* — це новий інструмент, який потрібно завантажити в шпиндель (кишеня «0»). За угодою, номер інструменту 0 використовується для позначення порожнього інструменту,

---

9.10.1.3 Застосування

Використання **db_program** не змінює спосіб роботи LinuxCNC, але забезпечує підтримку нових функцій бази даних для керування інструментами.

Наприклад, програма **db_program** може вести облік робочого часу всіх інструментів, відстежуючи кожне завантаження/розвантаження інструменту. Тоді верстат може мати три 6-міліметрові кінцеві фрези в кишенях 111, 112 і 113, а програма бази даних запрограмована так, щоб присвоїти номер 110 6-міліметровій кінцевій фрезі з найменшою кількістю годин роботи. Тоді, коли програма

LinuxCNC запитує інструмент 110, база даних вказує відповідну кишеню на основі історії використання інструменту.

Зміни даних інструменту, внесені в LinuxCNC (команди «p», «u», «l»), негайно передаються в **db_program**, який повинен синхронізувати свої вихідні дані. За замовчуванням, запити LinuxCNC на дані інструменту (команди *g*) виконуються тільки під час запуску. Програма бази даних може постійно оновлювати дані про використання інструменту, тому довготривалі програми LinuxCNC можуть отримати перевагу від оновлення даних інструменту, що надаються **db_program**. Команда G-коду **G10L0** може використовуватися для запиту перезавантаження даних інструменту (команда «g») з програм G-коду або за допомогою MDI. Операція перезавантаження також зазвичай забезпечується графічним інтерфейсом користувача (GUI), що дозволяє запитувати перезавантаження за запитом. Наприклад, програма GUI на Python може використовувати:

```
#!/usr/bin/env python3
from  linuxcnc import command
command().load_tool_table()
```

Крім того, **db_program** може передавати свої локальні зміни даних для синхронізації з LinuxCNC за допомогою команди інтерфейсу load_tool_table(). Команди, які передають зміни до LinuxCNC, можуть бути відхилені, якщо працює інтерпретатор. Стан інтерпретатора можна перевірити перед видачею команди load_tool_table(). Приклад:

```
#! /usr/bin/env python3
import linuxcnc
s = linuxcnc.stat()
s.poll()
if s.interp_state == linuxcnc.INTERP_IDLE:
    linuxcnc.command().load_tool_table()
else: # b''вb''b''ib''b''дb''b''кb''b''лb''b''ab''b''cb''b''тb''b''иb'' b''зb''b''ab''b' ↩
    'вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'', b''дb''b''ob''b' ↩
    'кb''b''иb'' b''ib''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b''cb'' b''нb''b' ↩
    'eb'' b''пb''b''eb''b''pb''b''eb''b''йb''b''дb''b''eb'' b''вb'' b''pb''b''eb''b''жb''b' ↩
    'иb''b''мb'' b''ob''b''чb''b''ib''b''кb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''
   ...
```

Якщо програма бази даних додає або видаляє інструменти після ініціалізації, необхідно виконати виклик tooldb_tools() з оновленим списком user_tools. Оновлений список інструментів буде використов в наступних командах get або запитах load_tool_table().

Note
Вилучення номера інструменту слід виконувати лише тоді, коли номер інструменту наразі не завантажено в шпиндель.

Експорт змінної середовища DB_SHOW дозволяє LinuxCNC виводити (на стандартний вивід) дані інструменту, отримані з **db_program**, під час запуску та під час наступного перезавантаження даних інструменту.

Експорт змінної середовища DB_DEBUG дозволяє LinuxCNC виводити (на стандартний вивід) додаткову інформацію про налагодження активності інтерфейсу.

9.10.1.4 Приклад програми

Приклад **db_program** (реалізований як скрипт Python) надається разом із прикладами моделювання. Програма демонструє необхідні операції для:

1. підтвердити версію запуску

2. отримання запитів на дані інструменту: *g* (команда **get**)
3. отримувати оновлення даних інструменту: *p* (команда **put**)
4. отримувати оновлення завантаження інструменту: *l* (**load_spindle**
5. отримувати оновлення вивантаження інструменту: *u* (команда **unload_spindle**)

9.10.1.5 Модуль Python toolbd

У прикладі програми використовується модуль Python («tooldb»), наданий LinuxCNC, який управляє низькорівневими деталями для зв’язку та перевірки версії. Цей модуль використовує функції зворотного виклику, визначені **db_program**, для відповіді на команду «g» (get) та команди, що вказують на зміни даних інструменту («p», «l», «u»).

Програма **db_program** використовує модуль *tooldb*, реалізуючи наступний код Python:

```
user_tools = list(...)   # b''cb''b''пb''b''иb''b''cb''b''ob''b''кb'' b''дb''b''ob''b''cb'' ↩
    b''тb''b''yb''b''пb''b''нb''b''иb''b''хb'' b''нb''b''ob''b''мb''b''eb''b''pb''b''ib''b' ↩
    'вb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''ib''b' ↩
    'вb''

def user_get_tool(toolno):
    # b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''яb'' b''дb''b''лb''b''яb'' b''вb''b' ↩
        'ib''b''дb''b''пb''b''ob''b''вb''b''ib''b''дb''b''ib'' b''нb''b''ab'' b''кb''b''ob'' ↩
        b''мb''b''ab''b''нb''b''дb''b''иb'' 'g' (get)
    # b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''тb''b''ьb''b''cb''b''яb''  ↩
        b''ob''b''дb''b''иb''b''нb'' b''pb''b''ab''b''зb'' b''дb''b''лb''b''яb'' b''кb''b' ↩
        'ob''b''жb''b''нb''b''ob''b''гb''b''ob'' toolno b''вb'' user_tools
    ...
def user_put_tool(toolno,params):
    # b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''яb'' b''дb''b''лb''b''яb'' b''вb''b' ↩
        'ib''b''дb''b''пb''b''ob''b''вb''b''ib''b''дb''b''ib'' b''нb''b''ab'' b''кb''b''ob'' ↩
        b''мb''b''ab''b''нb''b''дb''b''иb'' «p» (put)
    ...
def user_load_spindle(toolno,params):
    # b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''яb'' b''дb''b''лb''b''яb'' b''вb''b' ↩
        'ib''b''дb''b''пb''b''ob''b''вb''b''ib''b''дb''b''ib'' b''нb''b''ab'' b''кb''b''ob'' ↩
        b''мb''b''ab''b''нb''b''дb''b''иb'' «l» (put)
    ...
def user_unload_spindle(toolno,params):
    # b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''яb'' b''дb''b''лb''b''яb'' b''вb''b' ↩
        'ib''b''дb''b''пb''b''ob''b''вb''b''ib''b''дb''b''ib'' b''нb''b''ab'' b''кb''b''ob'' ↩
        b''мb''b''ab''b''нb''b''дb''b''иb'' «u» (put)
    ...

#-----------------------------------------------------------
# b''Пb''b''ob''b''чb''b''ab''b''тb''b''ob''b''кb'':
from tooldb import tooldb_tools     # b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b''иb''b' ↩
    'тb''b''иb'' b''вb''b''ib''b''дb''b''ob''b''мb''b''ib'' b''ib''b''нb''b''cb''b''тb''b' ↩
    'pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''иb''
from tooldb import tooldb_callbacks # b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b''иb''b' ↩
    'тb''b''иb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb''
from tooldb import tooldb_loop      # b''гb''b''ob''b''лb''b''ob''b''вb''b''нb''b''иb''b' ↩
    'йb'' b''цb''b''иb''b''кb''b''лb''

tooldb_tools(user_tools)
tooldb_callbacks(user_get_tool,
                 user_put_tool,
                 user_load_spindle,
                 user_unload_spindle,
                )
```

```
tooldb_loop()
```

Note
Використання «tooldb» не є обов’язковим — він надається як демонстрація необхідного інтерфейсу та для зручності реалізації програм на основі Python, які взаємодіють із зовнішньою базою даних.

9.10.2 Конфігурації симуляції

Конфігурації моделювання за допомогою графічного інтерфейсу AXIS:

1. configs/sim/axis/db_demo/**db_ran**.ini (random_toolchanger)
2. configs/sim/axis/db_demo/**db_nonran**.ini (nonrandom_toolchanger)

Кожна конфігурація симулятора імітує **db_program**, що реалізує базу даних з 10 інструментами, пронумерованими від 10 до 19.

db_program надається одним скриптом (db.py) та символічними посиланнями на нього для альтернати використання: db_ran.py та db_nonran.py. За замовчуванням скрипт реалізує функціональність random_toolchanger. Невипадкові функції зміни інструменту замінюються, якщо ім’я посилання містить текст «nonran».

Конфігурації симулятора демонструють використання модуля інтерфейсу Python «tooldb» та реалізують базову базу даних у вигляді плоского файлу, яка відстежує час використання інструментів для декількох інструментів, що мають однаковий діаметр. Правила бази даних підтримують вибір інструменту з найменшим часом роботи.

Конфігурації симулятора використовують основне завдання для моніторингу та реагування на оновлення інструментів, ініційовані зсередини LinuxCNC. Періодичне завдання оновлює час використа інструменту через регулярні проміжки часу. Окремі, паралельні завдання реалізовані як потоки для демонстрації коду, необхідного при ініціюванні змін **db_program**, та демонстрації методів синхронізації внутрішніх даних інструментів LinuxCNC. Приклади включають:

1. оновлення параметрів інструменту
2. додавання та вилучення номерів інструментів

Блокування взаємного виключення використовується для захисту даних від невідповідностей через умови змагання між оновленнями інструментальних даних LinuxCNC та оновленнями програми бази даних.

9.10.2.1 Нотатки

Коли **db_program** використовується разом із випадковим змінювачем інструментів ([EMCIO]RANDOM_ LinuxCNC підтримує файл («db_spindle.tbl» у каталозі конфігурації), який складається з одного рядка таблиці інструментів, що ідентифікує поточний інструмент у шпинделі.

Part II

Застосування

Chapter 10

Інтерфейс користувача

10.1 AXIS GUI

10.1.1 Вступ

AXIS — це графічний інтерфейс для LinuxCNC, який має попередній перегляд у реальному часі та відображення графіка. Він написаний на Python та використовує Tk та OpenGL для відображення інтерфейсу користувача.

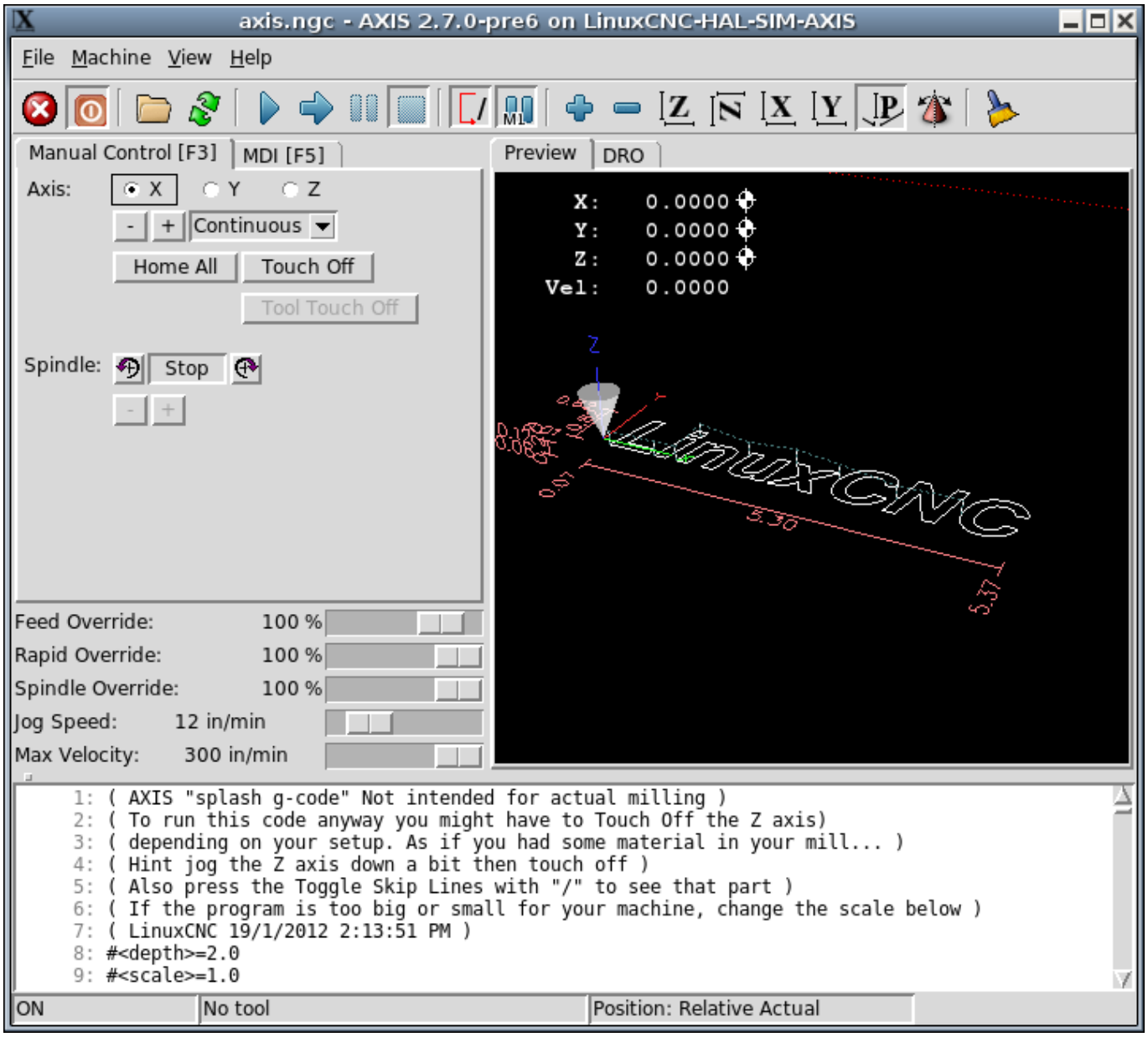


Figure 10.1: Вікно AXIS

10.1.2 Початок роботи

Якщо ваша конфігурація наразі не налаштована на використання AXIS, ви можете змінити це, відредагувавши файл .ini (INI-файл). У розділі `[DISPLAY]` змініть рядок `[DISPLAY]` на `DISPLAY = axis`.

Зразок конфігурації *sim/axis.ini* вже налаштований на використання AXIS як інтерфейсу.

Після запуску AXIS відкривається вікно, подібне до того, що показано на рисунку Figure 10.1 вище.

10.1.2.1 Налаштування INI

Для отримання додаткової інформації про налаштування файлу INI, які можуть змінити роботу AXIS, див. розділи Display Section та Axis Section глави «Налаштування INI».

- *CYCLE_TIME* - Налаштування швидкості відгуку графічного інтерфейсу користувача в мілісекундах. Типове значення 100, діапазон використання 50 - 200.
  (з міркувань сумісності з попередніми версіями приймається час у секундах (.05 -.2), але для узгодження з іншими екранами краще використовувати мілісекунди).

```
[DISPLAY]
CYCLE_TIME = 100
```

- *PREVIEW_TIMEOUT* - Встановити час очікування в секундах для завантаження попереднього перегляду G-коду. Якщо аналіз G-коду триває довше, ніж це, з'являється повідомлення і на графічному дисплеї відображається тільки початкова частина програми. Вказавши 0 або залишивши налаштування без змін, час очікування не встановлюється.

```
[DISPLAY]
PREVIEW_TIMEOUT = 5
```

10.1.2.2 Типова сесія

1. Запустіть LinuxCNC та виберіть файл конфігурації.
2. Відпустіть кнопку АВАРІЙНОЇ СТОПИ (F1) та увімкніть живлення машини (F2).
3. Головна сторінка всіх осей.
4. Завантажте файл G-коду.
5. Використайте графік попереднього перегляду, щоб перевірити правильність програми.
6. Завантажте матеріал.
7. Встановіть правильне зміщення для кожної осі за допомогою штовхання та використання кнопки «Touch Off» за потреби.
8. Запустіть програму.
9. Щоб знову обробити той самий файл, поверніться до кроку 6. Щоб обробити інший файл, поверніться до кроку 4.
10. Після завершення завдання вийдіть з AXIS.

Note
Повторне виконання тієї ж програми залежить від ваших налаштувань і вимог. Можливо, вам доведеться завантажити більше матеріалу і встановити зміщення або переміститися і встановити зміщення, а потім знову запустити програму. Якщо ваш матеріал закріплений, можливо, вам доведеться лише знову запустити програму. Дивіться Machine Menu для отримання додаткової інформації про команду запуску.

10.1.3 Вікно AXIS

Вікно AXIS містить такі елементи:

- Область відображення, яка відображає одне з наступного:
  - Попередній перегляд завантаженого файлу (у цьому випадку «axis.ngc»), а також поточне місцезнаходження «контрольованої точки» верстата з ЧПК. Пізніше в цій області буде відображено шлях, яким рухався верстат з ЧПК, який називається «backplot».
  - Велике зчитування, що показує поточне положення та всі зміщення.
- Панель меню та панель інструментів, які дозволяють виконувати різні дії
- «Вкладка ручного керування» – дозволяє рухати верстат, вмикати або вимикати шпиндель, а також вмикати або вимикати подачу охолоджувальної рідини, якщо це включено до INI-файлу.
- «Вкладка MDI» – тут можна вводити G-коди вручну, по одному рядку за раз. Тут також відображаються «Активні G-коди», які показують, які модальні G-коди активні.
- «Feed Override» (Перевищення швидкості подачі) — дозволяє змінювати швидкість запрограмованих рухів. Максимальне значення за замовчуванням становить 120 % і може бути змінено в файлі INI. Докладнішу інформацію див. у розділі Display Section файлу INI.
- «Перевірка швидкості шпинделя» – дозволяє збільшувати або зменшувати швидкість шпинделя.
- «Швидкість поштовхового переміщення» – дозволяє встановити швидкість поштовхового переміщення в межах, встановлених у файлі INI. Див. розділ Display файлу INI для отримання додаткової інформації.
- «Максимальна швидкість» – дозволяє обмежити максимальну швидкість усіх запрограмованих рухів (крім синхронізованого зі шпинделем руху).
- Область відображення тексту, яка показує завантажений G-код.
- Смуга стану, яка показує стан верстата. На цьому знімку екрана верстат увімкнено, інструмент не вставлено, а відображуване положення — «Відносне» (показує всі зміщення) та «Фактичне» (показує положення зворотного зв’язку).

10.1.3.1 Пункти меню

Деякі пункти меню можуть бути неактивними залежно від того, як налаштовано ваш INI-файл. Для отримання додаткової інформації про налаштування див. INI Chapter.

- «Відкрити…» — відкриває стандартне діалогове вікно для відкриття файлу G-коду, який потрібно завантажити в AXIS. Якщо ви налаштували LinuxCNC для використання програми-фільтра, ви також можете її відкрити. Докладнішу інформацію див. у розділі FILTER Section конфігураційного файлу INI.
- «Останні файли» – відображає список нещодавно відкритих файлів.
- «Редагувати…» — відкрийте поточний файл G-коду для редагування, якщо у вашому файлі INI налаштований редактор. Докладнішу інформацію про вказання редактора, який слід використовувати див. у розділі DISPLAY Section.
- «Перезавантажити» — перезавантажити поточний файл G-коду. Якщо ви його редагували, необхідно перезавантажити, щоб зміни набули чинності. Якщо ви зупинили файл і хочете почати з початку, перезавантажте файл. Перезавантаження на панелі інструментів відбувається так само, як і в меню.
- «Зберегти G-код як…» – Зберегти поточний файл з новим ім’ям.

- «Властивості» – сума швидкого переміщення та подачі. Не враховує прискорення, змішування або режим траєкторії, тому час, що повідомляється, ніколи не буде меншим за фактичний час виконання.
- «Редагувати таблицю інструментів…» – те саме, що й «Редагувати», якщо ви визначили редактор, ви можете відкрити таблицю інструментів та редагувати її.
- «Перезавантажити таблицю інструментів» – після редагування таблиці інструментів її необхідно перезавантажити.
- «Редактор сходів» – якщо ви завантажили ClassicLadder, ви можете редагувати його звідси. Див. розділ ClassicLadder для отримання додаткової інформації.
- *Quit* - Завершує поточний сеанс LinuxCNC.

- «Увімкнути/вимкнути аварійну зупинку F1» – Змінити стан аварійної зупинки.
- «Перемикач живлення машини F2» – Змінити стан живлення машини, якщо аварійна зупинка не ввімкнена.
- «Запустити програму» – Запустити поточну завантажену програму з початку.
- «Виконати від вибраного рядка» — виберіть рядок, з якого потрібно почати. Використовуйте цю функцію з обережністю, оскільки вона спочатку перемістить інструмент у потрібне положення перед рядком, а потім виконає решту коду.



Warning
Не використовуйте команду «Запустити з вибраного рядка», якщо ваша G-програма містить підпрограми.

- *Step* - Один крок через програму.
- *Pause* - Призупинити програму.
- *Resume* - Відновіть біг після паузи.
- «Зупинити» – Зупинити запущену програму. Якщо після зупинки вибрати «Запустити», програма почнеться з початку.
- «Зупинити на лінії M1» – якщо досягнуто лінії M1 і це позначено, виконання програми зупиниться на лінії M1. Натисніть «Відновити», щоб продовжити.
- «Пропускати рядки з символом «/»» – Якщо рядок починається з символу «/» і цей параметр позначено, рядок буде пропущено.
- «Очистити історію MDI» – очищає вікно історії MDI.
- «Копіювати з історії MDI» – копіює історію MDI в буфер обміну
- «Вставити в історію MDI» – Вставити з буфера обміну у вікно історії MDI
- «Калібрування» — запускає помічник калібрування (emccalib.tcl). Калібрування зчитує файл HAL і для кожного «setp», що використовує змінну з файлу INI, який знаходиться в розділі [AXIS_L], [JOINT_N], [SPINDLE_S] або [TUNE], створює запис, який можна редагувати та тестувати.
- «Показати конфігурацію HAL» – відкриває вікно конфігурації HAL, де можна відстежувати компоненти HAL, контакти, параметри, сигнали, функції та потоки.
- «HAL Meter» – відкриває вікно, де можна контролювати окремий контакт HAL, сигнал або параметр.

- «HAL Scope» – відкриває віртуальний осцилограф, який дозволяє будувати графік залежності значень HAL від часу.
- «Показати стан LinuxCNC» – відкриває вікно, що відображає стан LinuxCNC.
- «Встановити рівень налагодження» – відкриває вікно, де можна переглянути рівні налагодження, а деякі з них – встановити.
- *Homing* - Додому одну або всі осі.
- *Unhoming* - Зняти з базового положення одну або всі осі.
- «Нульова система координат» – встановити всі зміщення рівними нулю у вибраній системі координат.
- Інструмент дотику вимкнено
  - «Дотик інструменту до заготовки» — під час виконання дотику введене значення є відносним до поточної системи координат заготовки («G5x»), зміненої зміщенням осі («G92»). Після завершення дотику відносна координата для вибраної осі стане введеним значенням. Див. G10 L10 у розділі «G-код».
  - «Зупинка інструменту на кріпленні» — під час виконання зупинки інструменту введене значення є відносним до дев'ятої («G59.3») системи координат, при цьому зміщення осі («G92») ігнорується. Це корисно, коли на верстаті є пристрій для зняття інструменту з фіксованим розташуванням, а дев'ята система координат («G59.3») налаштована таким чином, що кінчик інструменту нульової довжини знаходиться в початку координат пристрою, коли відносні координати дорівнюють 0. Див. G10 L11 у розділі «G-код».
- «Вид зверху» – вид зверху (або вид Z) відображає G-код вздовж осі Z від позитивного до негативного напрямку. Цей вид найкраще підходить для перегляду X та Y.
- «Повернений вигляд зверху» — Повернений вигляд зверху (або повернений вигляд по осі Z) також відображає G-код вздовж осі Z від позитивного до негативного значення. Але іноді зручно відображати осі X і Y, повернені на 90 градусів, щоб краще відповідати екрану. Цей вигляд також найкраще підходить для перегляду X і Y.
- «Вид збоку» – Вигляд збоку (або вигляд X) відображає G-код вздовж осі X від позитивного до негативного положення. Цей вигляд найкраще підходить для перегляду осей Y та Z.
- «Вид спереду» – Вид спереду (або вид Y) відображає G-код вздовж осі Y від негативного до позитивного. Цей вид найкраще підходить для перегляду X та Z.
- «Перспективний вигляд» — Перспективний вигляд (або P-вигляд) відображає G-код, дивлячись на деталь з регульованої точки зору, за замовчуванням X+, Y-, Z+. Положення можна регулювати за допомогою миші та селектора перетягування/обертання. Цей вигляд є компромісним, і хоча він добре показує три (до дев'яти!) осі на двовимірному дисплеї, часто бувають елементи, які важко побачити, що вимагає зміни точки зору. Цей вигляд найкраще підходить, коли ви хочете побачити всі три (до дев'яти) осі одночасно.

Точка зору

Меню вибору відображення AXIS «View» (Вид) відноситься до видів «Top» (Зверху), «Front» (Спереду) та «Side» (Збоку). Ці терміни є правильними, якщо вертикаль осі Z верстата з ЧПК є вертикальною, а позитивне значення Z знаходиться вгорі. Це справедливо для вертикальних фрезерних верстатів, які, ймовірно, є найпопулярнішим видом обладнання, а також для майже всіх верстатів для електророзрядної обробки та навіть вертикальних токарних верстатів з револьверною головкою, де деталь обертається під інструментом.

Терміни «верх», «перед» і «бік» можуть викликати плутанину в інших верстатах з ЧПК, таких як стандартний токарний верстат, де вісь Z є горизонтальною, або горизонтальний фрезерний верстат, де вісь Z також є горизонтальною, або навіть вертикальний токарний верстат з поворотним револьвером, де деталь обертається над інструментом, а позитивний напрямок осі Z спрямований вниз!

Просто пам’ятайте, що додатна вісь Z (майже) завжди віддалена від деталі. Тому ознайомтеся з конструкцією вашого верстата та інтерпретуйте відображення за потреби.

- «Відображення в дюймах» – встановлення масштабування відображення AXIS у дюймах.
- «Відображення в міліметрах» – встановити масштаб відображення AXIS у міліметрах.
- «Показати програму» – за потреби попередній перегляд завантаженої програми G-коду можна повністю вимкнути.
- «Показати програму швидкого переміщення» — на попередньому перегляді завантаженої програми G-коду швидкість подачі (G1, G2, G3) завжди відображається білим кольором. Але відображення швидких переміщень (G0) блакитним кольором можна вимкнути за бажанням.
- «Програма з альфа-змішеним зображенням» – цей параметр спрощує попередній перегляд складних програм, але може призвести до повільнішого відображення попереднього перегляду.
- «Показати графік у реальному часі» – підсвічування траєкторій подачі (G1, G2, G3) під час руху інструменту можна вимкнути за потреби.
- «Показати інструмент» – за потреби відображення конуса/циліндра інструменту можна вимкнути.
- «Показати межі» – відображення меж (максимального переміщення в кожному напрямку осі) завантаженої програми G-коду можна вимкнути за потреби.
- «Показати зміщення» — вибране зміщення кріплення (G54-G59.3) можна відобразити у вигляді набору трьох ортогональних ліній: червоної, синьої та зеленої. Цей показ зміщення (або нульової точки кріплення) можна вимкнути за бажанням.
- «Показати межі машини» — максимальні межі переміщення машини для кожної осі, встановлені в файлі INI, відображаються у вигляді прямокутного поля, обведеного червоними пунктирними лініями. Це корисно під час завантаження нової програми G-коду або під час перевірки, яке зміщення кріплення потрібно, щоб програма G-коду не виходила за межі переміщення вашої машини. Цю функцію можна вимкнути, якщо вона не потрібна.
- «Показати швидкість» – відображення швидкості іноді корисне, щоб побачити, наскільки близько ваша машина працює до своїх розрахункових швидкостей. За бажанням його можна вимкнути.
- «Показати відстань до кінця» — відстань до кінця є дуже корисною інформацією, яку варто знати під час першого запуску невідомої програми G-коду. У поєднанні з елементами керування швидким переходом і швидкістю подачі можна уникнути небажаного пошкодження інструменту та верстата. Після налагодження програми G-коду та її безперебійного виконання відображення відстані до кінця можна вимкнути за бажанням.
- «Координати великим шрифтом…» – Координати осей та швидкість, задані заздалегідь, відображати великим шрифтом у вигляді траєкторії інструменту.

- «Очистити живий графік» — під час переміщення інструменту на дисплеї AXIS виділяється траєкторія G-коду. Щоб повторити програму або краще роздивитися область, що вас цікавить, можна очистити раніше виділені траєкторії.
- «Показати командну позицію» – це позиція, в яку LinuxCNC спробує перейти. Після зупинки руху LinuxCNC спробує утримати цю позицію.
- «Показати фактичне положення» — фактичне положення — це виміряне положення, зчитане з енкодерів системи або змодельоване генераторами кроків. Воно може дещо відрізнятися від заданого положення з багатьох причин, включаючи налаштування PID, фізичні обмеження або квантування положення.
- «Показати положення машини» – це положення в незміщених координатах, встановлене за допомогою функції «Перенаправлення».
- «Показати відносне положення» – це положення машини, змінене за допомогою зміщень «G5x», «G92» та «G43».
- «Про AXIS» – Ми всі знаємо, що це таке.
- «Короткий довідник» – показує комбінації клавіш.

10.1.3.2 Кнопки панелі інструментів

Зліва направо на дисплеї AXIS кнопки панелі інструментів (сполучення клавіш показано [у дужках]):

-  Перемикач аварійної зупинки [F1] (також називається E-Stop)
-  Перемикач живлення машини [F2]
-  Відкрити файл G-коду [O]
-  Перезавантажити поточний файл [Ctrl-R]
-  Почати виконання поточного файлу [R]
-  Виконати наступний рядок [T]
-  Призупинити виконання [P] Відновити виконання [S]
-  Зупинити виконання програми [ESC]
-  Перемикання пропуску рядків за допомогою ”/” [Alt-M-/]
-  Увімкнути/вимкнути необов’язкову паузу [Alt-M-1]
-  Збільшити

-  Зменшити масштаб
-  Вид зверху
-  Повернутий вид зверху
-  Вид збоку
-  Вид спереду
-  Перспективний вигляд
-  Перемикання між режимом перетягування та обертання [D]
-  Очистити живу задню діаграму [Ctrl-K]

10.1.3.3 Область графічного відображення

Відображення координат У верхньому лівому куті екрана програми відображається координатна позиція кожної осі. Праворуч від числа відображається символ початку координат: images/axis-homed.png [”символ початку координат відображається, якщо вісь була встановлена в початкове положення”], якщо вісь була встановлена в початкове положення.

Символ обмеження  відображається праворуч від номера координатної позиції, якщо вісь знаходиться на одному зі своїх кінцевих вимикачів.

Щоб правильно інтерпретувати координатні номери позиції, зверніться до індикатора «Позиція:» у рядку стану. Якщо позиція є «Фактичною машиною», то відображуване число знаходиться в системі координат машини. Якщо це «Відносна фактична», то відображуване число знаходиться в системі координат зміщення. Коли відображувані координати є відносними і встановлено зміщення, на дисплеї з’явиться блакитний маркер machine origin  .

Якщо положення є «Заданим», то відображається точна координата, вказана в команді G-коду. Якщо воно є «Фактичним», то це положення, до якого фактично перемістилася машина. Ці значення можуть відрізнятися від заданого положення через наступні помилки, мертву зону, роздільну здатність енкодера або розмір кроку. Наприклад, якщо ви задаєте переміщення до X 0,0033 на фрезерному верстаті, але один крок крокового двигуна або один імпульс енкодера дорівнює 0,00125, то «задане» положення може бути 0,0033, але «фактичне» положення буде 0,0025 (2 кроки) або 0,00375 (3 кроки).

Попередній перегляд сюжету Коли файл завантажується, його попередній перегляд відображаєтьс в області відображення. Швидкі переміщення (наприклад, ті, що виконуються командою «G0») відображаються у вигляді блакитних ліній. Переміщення зі швидкістю подачі (наприклад, ті, що виконуються командою «G1») відображаються у вигляді суцільних білих ліній. Затримки (наприклад, ті, що виконуються командою «G4») відображаються у вигляді маленьких рожевих знаків «X».

Рухи G0 (швидкі) перед рухом подачі не відображаються на попередньому перегляді. Швидкі рухи після T<n> (заміна інструменту) не відображаються на попередньому перегляді до першого

руху подачі. Щоб вимкнути будь-яку з цих функцій, запрограмуйте G1 без будь-яких рухів перед рухами G0.

Обсяги програми По кожній осі показано «обсяги» програми. На кінцях вказані найменше та найбільше значення координат. Посередині показано різницю між координатами.

Коли деякі координати перевищують «м'які обмеження» в файлі INI, відповідний розмір відображаєтьс іншим кольором і обводиться рамкою. На малюнку нижче максимальне м'яке обмеження перевищено на осі X, про що свідчить рамка навколо значення координати. Мінімальний хід по осі X програми становить -1,95, максимальний хід по осі X становить 1,88, а програма вимагає ходу по осі X 3,83 дюйма. Щоб перемістити програму так, щоб вона знаходилася в межах переміщення верстата в цьому випадку, перемістіть її вліво і знову виконайте Touch Off X.



Figure 10.2: M'які обмеження

Конус інструменту Коли інструмент не завантажено, розташування кінчика інструмента позначаєть «конусом інструмента». «Конус інструмента» не надає вказівок щодо форми, довжини чи радіуса інструмента.

Коли інструмент завантажується (наприклад, за допомогою команди MDI *T1 M6*), конус змінюється на циліндр, який показує діаметр інструмента, вказаний у файлі таблиці інструментів.

Задній план Коли машина рухається, вона залишає слід, який називається «бекплот». Колір лінії вказує на тип руху: жовтий для ривків, блідо-зелений для швидких рухів, червоний для прямих рухів зі швидкістю подачі та пурпуровий для кругових рухів зі швидкістю подачі.

Сітка AXIS може опціонально відображати сітку в ортогональних видах. Увімкніть або вимкніть сітку за допомогою меню «Сітка» в розділі «Вид». Коли сітка увімкнена, вона відображається у верхньому та оберненому верхньому видах; коли система координат не обертається, сітка відображається також у передньому та бічному видах. Попередні налаштування в меню «Сітка» контролюються елементом INI-файлу `[DISPLAY]GRIDS`. Якщо не вказано, за замовчуванням використов значення `10 мм, 20 мм, 50 мм, 100 мм, 1 дюйм, 2 дюйми, 5 дюймів, 10 дюймів`.

Вказівка дуже дрібної сітки може знизити продуктивність.

Взаємодія Клацнувши лівою кнопкою миші на частині діаграми попереднього перегляду, лінія буде виділена як у графічному, так і в текстовому режимах. Клацнувши лівою кнопкою миші на порожній області, виділення буде знято.

Перетягуванням з натиснутою лівою кнопкою миші діаграма попереднього перегляду буде зміщена (панорамована).

Перетягуючи з натиснутою клавішею Shift і лівою кнопкою миші або перетягуючи з натиснутим колесом миші, графік попереднього перегляду буде обертатися. Коли лінія виділена, центром обертання є центр лінії. В іншому випадку центром обертання є центр всієї програми.

Обертаючи колесо миші, або перетягуючи вказівник з натиснутою правою кнопкою миші, або перетягуючи вказівник з натиснутою клавішею Control та лівою кнопкою миші, можна збільшити або зменшити масштаб діаграми попереднього перегляду.

Натиснувши один із значків «Попередньо встановлений вигляд» або клавішу «V», можна вибрати кілька попередньо встановлених виглядів.

10.1.3.4 Область відображення тексту

Клацнувши лівою кнопкою миші на рядку програми, цей рядок буде виділено як у графічному, так і в текстовому екранах.

Під час роботи програми рядок, що виконується, виділяється червоним кольором. Якщо користувач не вибрав жодного рядка, текстовий дисплей автоматично прокручується, щоб відобразити поточний рядок.

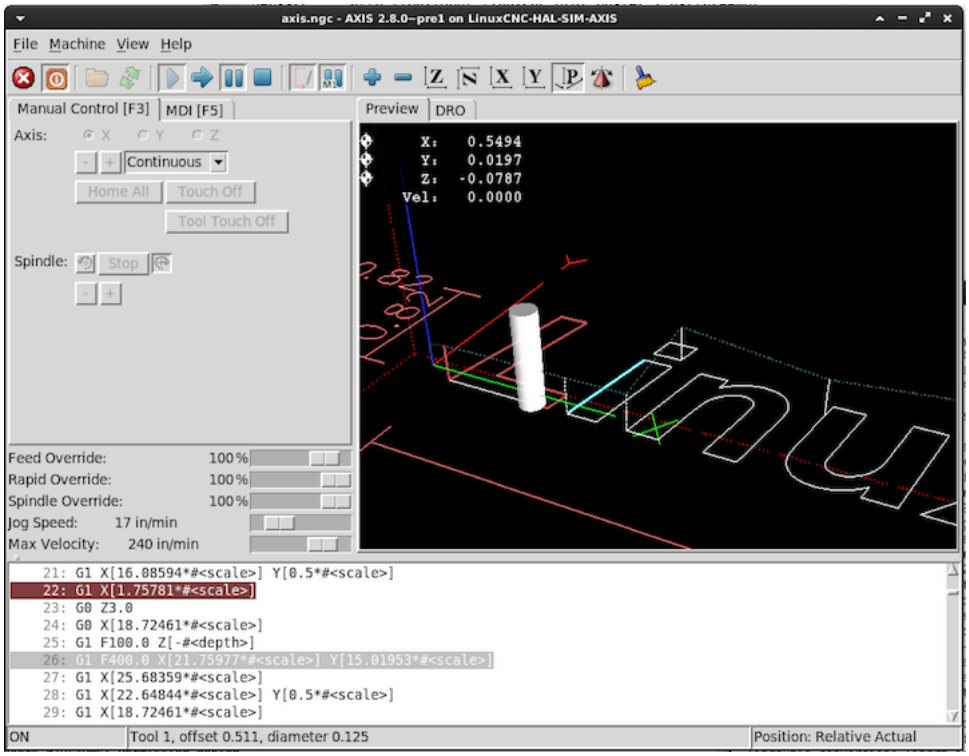


Figure 10.3: Поточні та вибрані лінії

10.1.3.5 Ручне керування

Коли верстат увімкнено, але програма не виконується, елементи на вкладці «Ручне керування» можна використовувати для переміщення верстата або керування його шпинделем та охолоджуючою рідиною.

Коли машина вимкнена або коли програма виконується, ручне керування недоступне.

Багато з описаних нижче елементів не є корисними на всіх машинах. Коли AXIS виявляє, що певний контакт не підключений в HAL, відповідний елемент на вкладці «Ручне керування» видаляється. Наприклад, якщо контакт HAL «spindle.0.brake» не підключений, кнопка «Brake» не з’явиться на екрані. Якщо встановлено змінну середовища «AXIS_NO_AUTOCONFIGURE», ця функція вимикається і всі елементи з’являться.

Група Осі «AXIS» дозволяє вручну переміщати верстат. Ця дія називається «поступальним рухом». Спочатку виберіть вісь, яку потрібно перемістити, натиснувши на неї. Потім натисніть і утримуйте кнопку «+» або «-» залежно від бажаного напрямку руху. Перші чотири осі також можна переміщати за допомогою клавіш зі стрілками (X і Y), клавіш PAGE UP і PAGE DOWN (Z) та клавіш [і] (A).

Якщо вибрано «Безперервно», рух триватиме доти, доки натиснуто кнопку або клавішу. Якщо вибрано інше значення, машина буде рухатися на відстань, що відображається, кожного разу,

коли натискається кнопка або клавіша. За замовчуванням доступні значення «0,1000; 0,0100; 0,0010; 0,0001».

Див. розділ DISPLAY для отримання додаткової інформації про встановлення приростів.

Самонаведення (кінематика ідентичності) Параметр INI-файлу [KINS]JOINTS визначає загальну кількість з'єднань для системи. З'єднання може бути налаштоване з перемикачем початкового положення або для «негайного» повернення в початкове положення. З'єднання можуть визначати послідовність початкового положення, яка організовує порядок повернення груп з'єднань у початкове положення.

Якщо **всі** з'єднання налаштовані для повернення в початкове положення і мають дійсні послідовності повернення, кнопка повернення в початкове положення буде показувати «Повернути все». Натискання кнопки «Повернути все» (або клавіші Ctrl-HOME) ініціює повернення всіх з'єднань у початкове положення за допомогою визначених послідовностей повернення. Натискання клавіші HOME поверне у початкове положення з'єднання, що відповідає поточній вибраній осі, навіть якщо послідовність повернення не визначена.

Якщо не всі осі мають дійсні послідовності додому, кнопка переведення в початкове положення відображатиме «Додому вісь» і переведе з'єднання в початкове положення лише для поточної вибраної осі. Кожну вісь потрібно вибрати та перевести в початкове положення окремо.

Випадаюче меню «Машина/Повернення до початкового положення» пропонує альтернативний метод повернення осей до початкового положення. Випадаюче меню «Машина/Зняття з початкового положення» надає засоби для повернення осей до початкового положення.

Якщо у вашій машині в конфігурації не визначено перемикачі початкового положення, кнопка «Home» (Початкове положення) встановить поточне положення вибраної осі як абсолютне положення 0 для цієї осі та встановить біт «is-homed» (початкове положення) для цієї осі.

Див. розділ Налаштування самоналаштування для отримання додаткової інформації.

Самонаведення (кінематика без ідентичності) Функціонування аналогічне функції «Ідентифікація кінематики», але перед поверненням у вихідне положення за допомогою перемикачів вибору вибираються з'єднання за номером. Кнопка повернення у вихідне положення відображатиме «Повернути все», якщо всі з'єднання налаштовані для повернення у вихідне положення і мають дійсні послідовності повернення. В іншому випадку кнопка повернення у вихідне положення відображатиме «Повернути з'єднання».

Див. розділ Налаштування самоналаштування для отримання додаткової інформації.

Дотик вимкнено

Натисканням кнопки «Touch Off» або клавіші END змінюється «G5x offset» для поточної осі, так що поточне значення осі буде відповідати заданому значенню. Вирази можна вводити, використовуючи правила для програм rs274ngc, за винятком того, що не можна посилатися на змінні. Отримане значення відображається у вигляді числа.

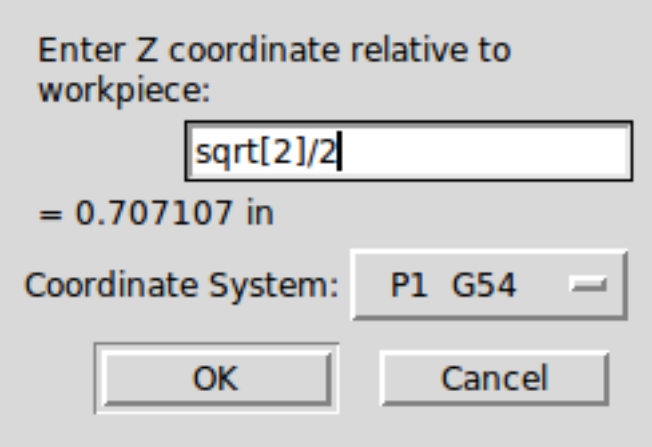


Figure 10.4: Вікно вимкнення дотику

Див. також пункти меню «Машина»: «Сенсорна деталь» та «Тримач сенсорної деталі».

Фактичне положення Дотик Вимкнено У файлі .INI можна налаштувати вісь, щоб включити фактичне значення положення осі в розрахунок відхилення, додаючи або віднімаючи це значення. Це в першу чергу корисно для верстатів, які мають немоторизовану вісь, наприклад, піноль з енкодером. Коли ця функція ввімкнена для осі, у заголовку вікна відхилення буде вказано **(system ACTUAL)**.

Див. Section 10.1.12.11 для отримання додаткової інформації.

Інструмент дотику вимкнено Натисканням кнопки «Tool Touch Off» (Вимкнення дотику інструмента довжина інструменту та зміщення поточного завантаженого інструменту будуть змінені таким чином, щоб поточне положення кінчика інструмента відповідало введеній координаті.

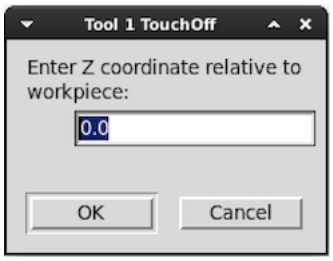


Figure 10.5: Вікно вимкнення дотику інструмента

Див. також опції «Торкання інструменту до заготовки» та «Торкання інструменту до пристосування» в меню «Верстат».

Замінити обмеження Натиснувши кнопку «Override Limits» (Перевищення обмежень), машина тимчасово отримає дозвіл на переміщення за межі фізичного кінцевого вимикача. Цей прапорець доступний тільки в разі спрацьовування кінцевого вимикача. Перевищення обмежень скидається після одного переміщення. Якщо вісь налаштована з окремими позитивними та негативними кінцевими вимикачами, LinuxCNC дозволить переміщення тільки в правильному напрямку. «Override Limits» (Перевищення обмежень) не дозволить переміщення за межі м'якого обмеження. Єдиний спосіб вимкнути м'яке обмеження на осі — це «Unhome» (Скинути нуль)._

Група «Шпиндель» Кнопки в першому рядку вибирають напрямок обертання шпинделя: проти годинникової стрілки, зупинка, за годинниковою стрілкою. Напрямок проти годинникової стрілки відображається тільки в тому випадку, якщо в файлі HAL є контакт «spindle.0.reverse» (це може

бути «net trick-axis spindle.0.reverse»). Кнопки в наступному рядку збільшують або зменшують швидкість обертання. Поле для позначки в третьому рядку дозволяє ввімкнути або вимкнути гальмо шпинделя. Залежно від конфігурації вашої машини, не всі елементи в цій групі можуть бути відображені. Натискання кнопки запуску шпинделя встановлює швидкість «S» на 1.

Група охолоджувальної рідини Дві кнопки дозволяють вмикати та вимикати охолоджувальні рідини «Розпилення» та «Потік». Залежно від конфігурації вашої машини, можуть відображатися не всі елементи цієї групи.

10.1.3.6 MDI

MDI дозволяє вводити команди G-коду вручну. Коли верстат вимкнено або коли програма виконується, елементи керування MDI недоступні.

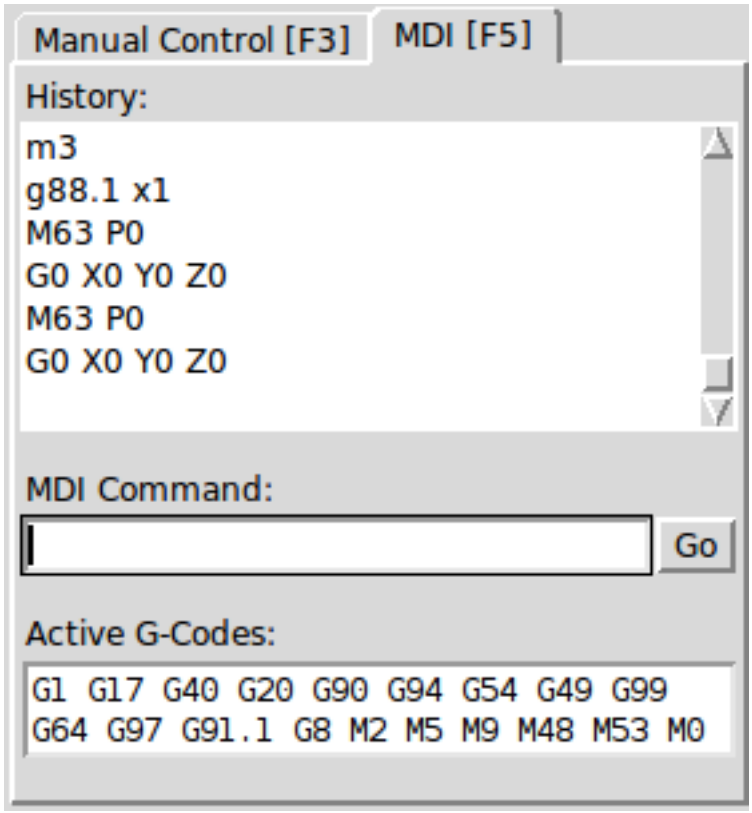


Figure 10.6: Вкладка MDI

- «Історія» – тут відображаються команди MDI, які були введені раніше в цьому сеансі.
- «Команда MDI» – дозволяє ввести команду G-коду для виконання. Виконайте команду, натиснувши Enter або натиснувши «Перейти».
- «Активні G-коди» — показує «модальні коди», які активні в інтерпретаторі. Наприклад, «G54» вказує, що «зсув G54» застосовується до всіх введених координат. У режимі «Авто» активні G-коди представляють коди після будь-якого попереднього зчитування інтерпретатором.

10.1.3.7 Перевизначення каналу

Переміщуючи цей повзунок, можна змінити запрограмовану швидкість подачі. Наприклад, якщо програма запитує «F60», а повзунок встановлено на 120%, то результуюча швидкість подачі становитиме 72.

10.1.3.8 Коригування швидкості шпинделя

Переміщуючи цей повзунок, можна змінювати запрограмовану швидкість шпинделя. Наприклад, якщо програма вимагає S8000, а повзунок встановлений на 80%, то швидкість шпинделя буде 6400. Цей елемент з'являється тільки тоді, коли підключений вивід HAL «spindle.0.speed-out».

10.1.3.9 Швидкість штовхання

Переміщуючи цей повзунок, можна змінювати швидкість переміщення. Наприклад, якщо повзунок встановлено на 1 дюйм/хв, то переміщення на 0,01 дюйма буде виконано приблизно за 0,6 секунди, або 1/100 хвилини. Ближче до лівого краю (повільні переміщення) значення розташовані близько одне до одного, а ближче до правого краю (швидкі переміщення) вони розташовані набагато далі одне від одного, що дозволяє використовувати широкий діапазон швидкостей переміщення з точним контролем, коли це найбільш важливо.

На верстатах з поворотною віссю відображається другий повзунок швидкості поштовхового переміщен
Цей повзунок встановлює швидкість поштовхового переміщення для поворотних осей (A, B та C).

10.1.3.10 Максимальна швидкість

Переміщуючи цей повзунок, можна встановити максимальну швидкість. Це обмежує максимальну швидкість для всіх запрограмованих рухів, окрім рухів, синхронізованих зі шпинделем.

10.1.4 Елементи керування клавіатурою

Майже всі дії в AXIS можна виконати за допомогою клавіатури. Повний список клавіатурних скорочень можна знайти в довіднику AXIS Quick Reference, який можна відкрити, вибравши Help > Quick Reference. Багато скорочень недоступні в режимі MDI.

10.1.4.1 Клавіші перевизначення подачі

Note
Щоб дізнатися більше про розкладку іспанської клавіатури, ознайомтеся з перекладеною документацією.

Клавіші Feed Override поводяться по-різному в ручному режимі. Клавіші «12345678» вибирають вісь, якщо вона запрограмована. Якщо у вас є 3 осі, то «» вибере вісь 0, «1» вибере вісь 1, а «2» вибере вісь 2. Решта цифрових клавіш як і раніше встановлюють Feed Override. Під час виконання програми «1234567890» встановлює перекриття подачі від 0% до 100%.

Найчастіше використовувані комбінації клавіш наведено в наступній таблиці:

Table 10.1: Найпоширеніші комбінації клавіш

Натискання клавіші	Вжиті заходи	Режим
F1	Увімкнути/вимкнути аварійну зупинку	Будь-який
F2	Увімкнення/вимкнення машини	Будь-який

Table 10.1: (continued)

Натискання клавіші	Вжиті заходи	Режим
`, 1 .. 9, 0	Встановити перевизначення подачі від 0% до 100%	Варіюється
X, `	Активувати першу вісь	Ручний
Y, 1	Активувати другу вісь	Ручний
Z, 2	Активувати третю вісь	Ручний
A, 3	Активувати четверту вісь	Ручний
I	Виберіть крок поштовху	Ручний
C	Безперервний біг підтюпцем	Ручний
Control-Home	Виконайте послідовність самонаведення	Ручний
Кінець	Touch off: Встановити зміщення G5x для активної осі	Ручний
Ліворуч, праворуч	Поштовховий рух першої осі	Ручний
Вгору, вниз	Поворот другої осі	Ручний
Сторінка вгору, сторінка вниз	Поворот третьої осі	Ручний
[,]	Поворот четвертої осі	Ручний
O	Відкрити файл	Ручний
Control-R	Перезавантажити файл	Ручний
R	Запустити файл	Ручний
P	Призупинити виконання	Авто
S	Виконання резюме	Авто
ESC	Зупинити виконання	Авто
Control-K	Очистити задній план	Автоматичний/Ручний
V	Перемикання між попередньо встановленими режимами перегляду	Автоматичний/Ручний
Shift-Left,Right	Швидка вісь X	Ручний
Shift-Up,Down	Швидка вісь Y	Ручний
Shift-PgUp, PgDn	Швидка вісь Z	Ручний
@	Перемикання між фактичним/заданим значенням	Будь-який
#	Перемикання між відносним/машинним	Будь-який

10.1.5 Показати стан LinuxCNC (`linuxcnctop`)

AXIS містить програму під назвою «linuxcnctop», яка показує деякі деталі стану LinuxCNC. Ви можете запустити цю програму, викликавши Machine > Show LinuxCNC Status

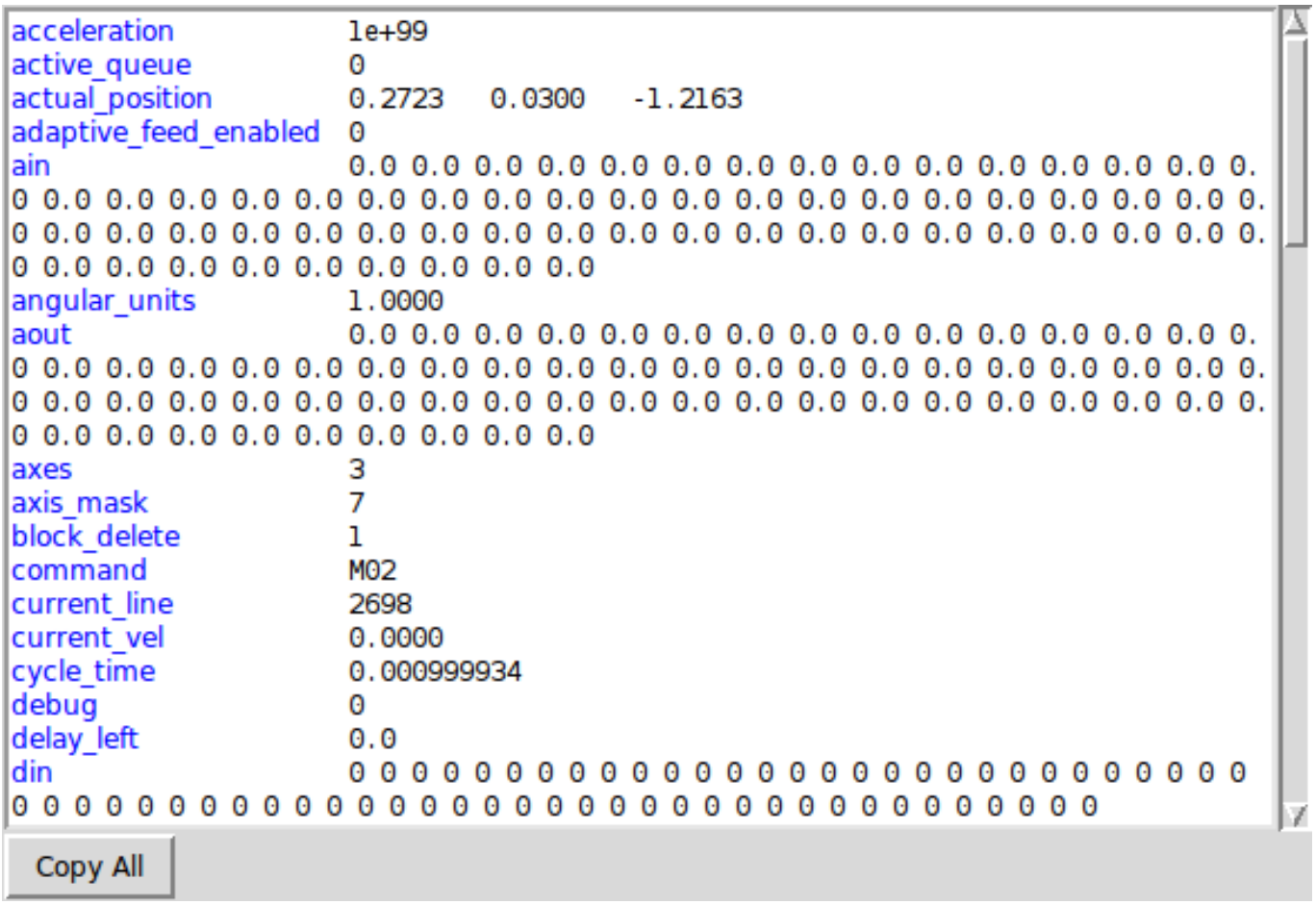


Figure 10.7: Вікно стану LinuxCNC

Назва кожного елемента відображається в лівому стовпці. Поточне значення відображається в правому стовпці. Якщо значення нещодавно змінилося, воно відображається на червоному фоні.

10.1.6 MDI-інтерфейс

AXIS включає програму під назвою «mdi», що є скороченням від «manual data input» (ручне введення даних), яка дозволяє вводити команди MDI в текстовому режимі в запущену сесію LinuxCNC. Ви можете запустити цю програму безпосередньо з командного рядка UNIX, відкривши термінал і ввівши:

```
mdi
```

Після запуску відображається запит «MDI>». Якщо введено порожній рядок, відображається поточна позиція верстата. Коли введено команду, вона надсилається на виконання до LinuxCNC.

Це приклад сеансу mdi:

```
$ mdi
MDI>
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI> G1 F5 X1
MDI>
(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI>
```

```
(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)
```

10.1.7 axis-remote

AXIS включає програму під назвою «axis-remote», яка може надсилати певні команди до запущеного AXIS. Доступні команди відображаються при запуску «axis-remote --help» і включають перевірку, чи працює AXIS («--ping»), завантаження файлу за назвою, перезавантаження поточного завантаженої файлу («--reload») та виведення AXIS («--quit»).

10.1.8 Ручна зміна інструменту

LinuxCNC включає в себе нереальний компонент HAL під назвою «hal_manualtoolchange», який відображає вікно з підказкою про те, який інструмент очікується при видачі команди «M6». Після натискання кнопки «OK» виконання програми продовжиться.

Компонент hal_manualtoolchange містить контакт HAL для кнопки, яку можна підключити до фізичної кнопки, щоб виконати зміну інструменту та прибрати вікно запиту (hal_manualtoolchange.cha

У файлі конфігурації HAL «lib/hallib/axis_manualtoolchange.hal» наведено команди HAL, необхідні для використання цього компонента.

hal_manualtoolchange можна використовувати, навіть коли AXIS не використовується як графічний інтерфейс. Цей компонент найбільш корисний, якщо у вас є попередньо налаштовані інструменти та ви використовуєте таблицю інструментів.

Note
Важлива примітка: після видачі команди T<n> пороги не відображатимуться на попередньому перегляді до наступного переміщення подачі після M6. Це може бути дуже заплутаним для більшості користувачів. Щоб вимкнути цю функцію для поточної програми зміни інструменту, введіть G1 без переміщення після T<n>.

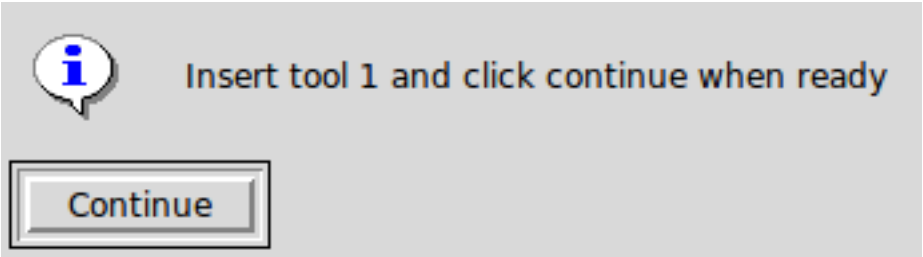


Figure 10.8: Вікно ручної зміни інструменту

10.1.9 Модулі Python

AXIS містить кілька модулів Python, які можуть бути корисними для інших. Щоб отримати додаткову інформацію про один із цих модулів, скористайтеся командою *pydoc <назва модуля>* або прочитайте вихідний код. Ці модулі включають:

- «emc» надає доступ до каналів команд, стану та помилок LinuxCNC
- «gcode» надає доступ до інтерпретатора rs274ngc

- «rs274» надає додаткові інструменти для роботи з файлами rs274ngc
- «hal» дозволяє створювати компоненти HAL, написані на Python, які не працюють у реальному часі
- «_togl» надає віджет OpenGL, який можна використовувати в застосунках Tkinter

Щоб використовувати ці модулі у власних скриптах, необхідно переконатися, що каталог, в якому вони знаходяться, знаходиться в шляху модулів Python. При запуску встановленої версії LinuxCNC це повинно відбуватися автоматично. При запуску «на місці» це можна зробити за допомогою «scripts/rip-environment».

10.1.10 Використання AXIS у режимі токарного верстата

Включення рядка «LATHE = 1» у розділ [DISPLAY] файлу INI дозволяє AXIS вибрати режим токарного верстата. Вісь «Y» не відображається в показаннях координат, вигляд змінюється, щоб показати вісь Z, що простягається вправо, і вісь X, що простягається до нижньої частини екрана, а також видаляються деякі елементи керування (наприклад, для попередньо встановлених видів). Показання координат для X замінюються діаметром і радіусом.

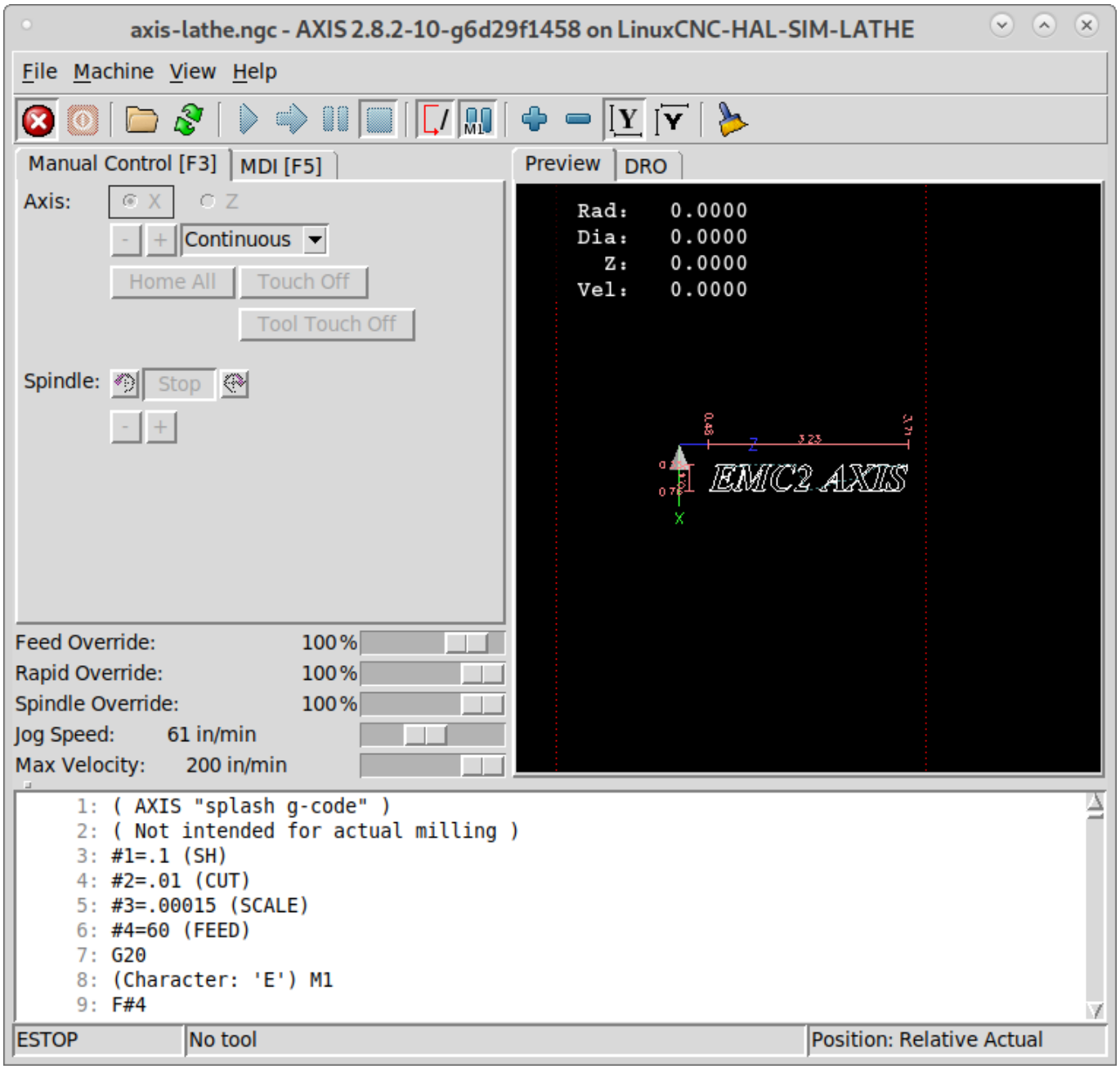


Figure 10.9: Режим токарного верстата AXIS

Натискання клавіші «V» зменшує масштаб, показуючи весь файл, якщо він завантажений.

У режимі токарного верстата відображається форма завантаженого інструменту (якщо такий є).

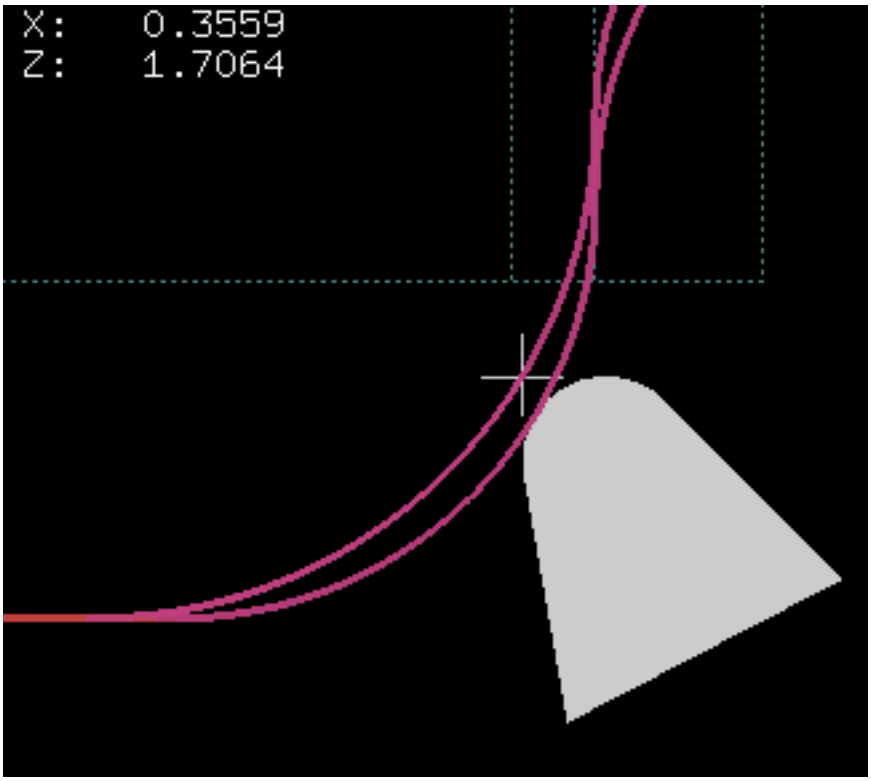


Figure 10.10: Форма токарного інструменту

Щоб змінити відображення на токарний верстат із задньою обробкою, у розділі [DISPLAY] потрібно вказати значення «LATHE = 1» та «BACK_TOOL_LATHE = 1». Це переверне зображення та розмістить інструмент на задній стороні осі Z.

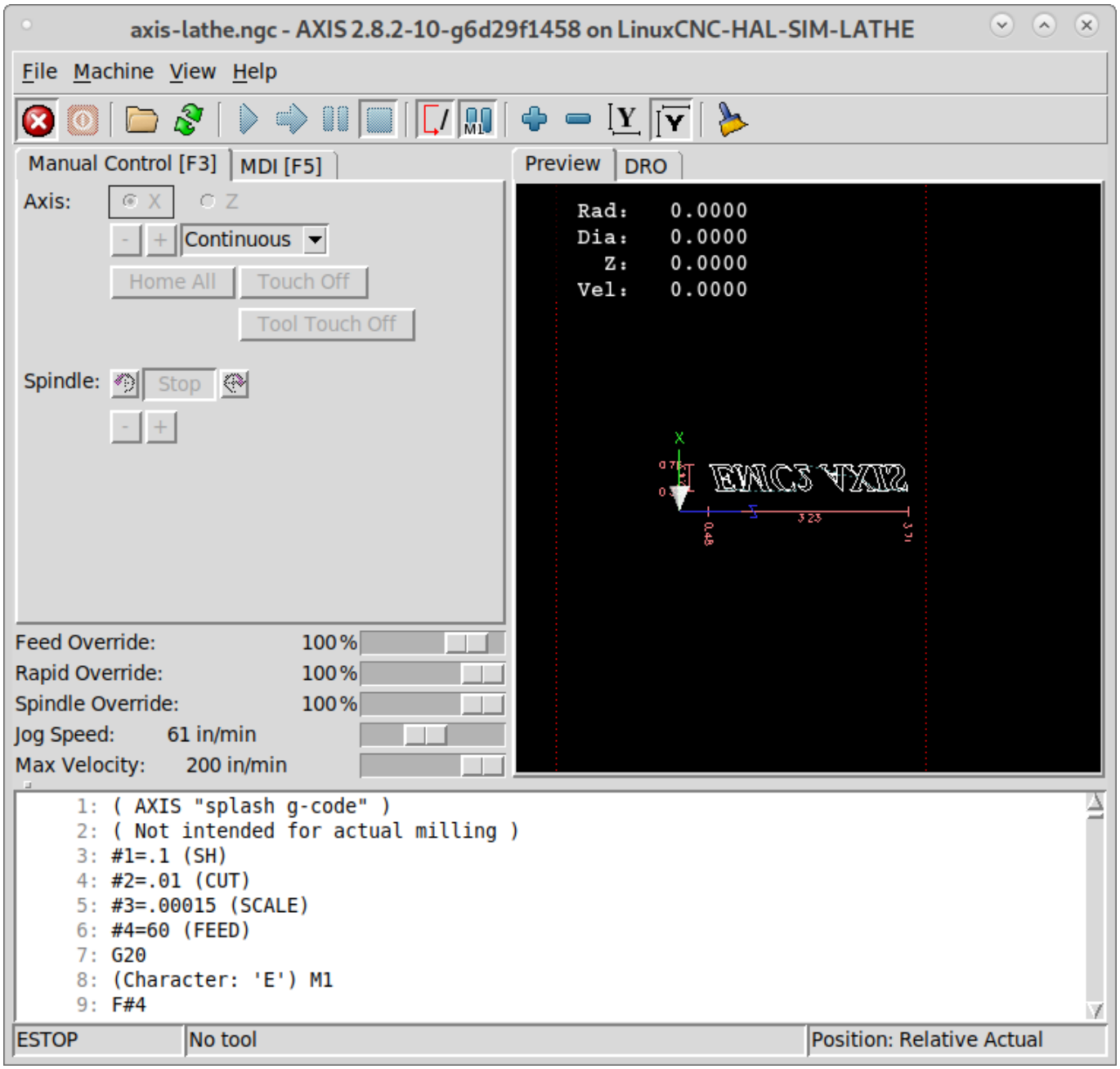


Figure 10.11: Форма заднього інструменту токарного верстата

10.1.11 Використання AXIS у режимі різання пінопласту

Включення рядка «FOAM = 1» у розділ [DISPLAY] файлу INI дозволяє AXIS вибрати режим різання пінопласту. У попередньому перегляді програми рухи XY відображаються в одній площині, а рухи UV — в іншій. У реальному графіку лінії малюються між відповідними точками на площині XY і площині UV. Спеціальні коментарі (XY_Z_POS) і (UV_Z_POS) встановлюють координати Z цих площин, які за замовчуванням дорівнюють 0 і 1,5 одиниць машини.

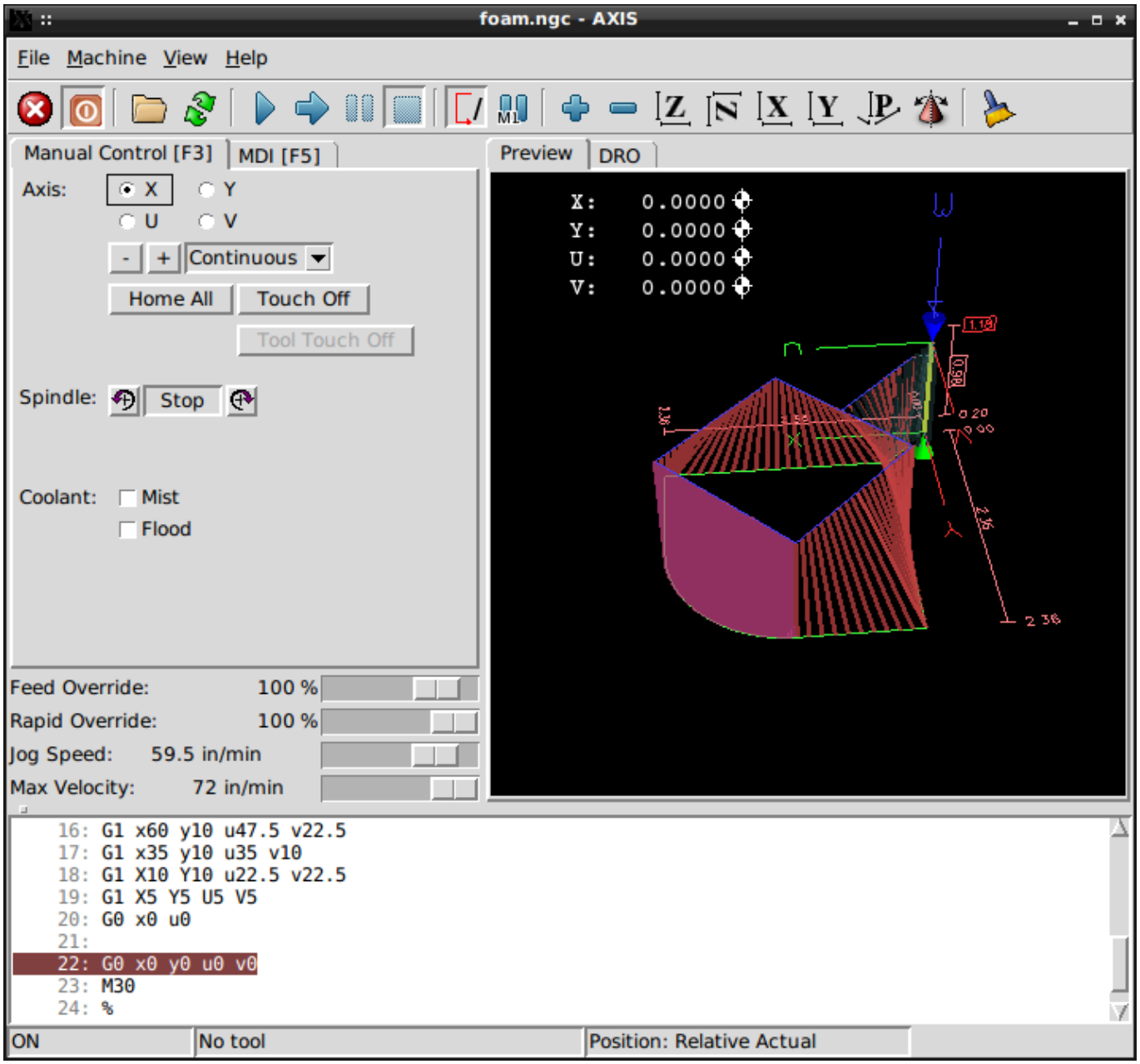


Figure 10.12: Режим різання піни

10.1.12 Розширена конфігурація

При запуску AXIS створює контакти HAL для графічного інтерфейсу користувача, а потім виконує файл HAL, названий у файлі INI: «[HAL]POSTGUI_HALFILE=<ім'я файлу>». Зазвичай «<filename>» буде базовим іменем конфігурації «_postgui» + «.hal», наприклад «lathe_postgui.hal», але може бути будь-яким допустимим іменем файлу. Ці команди виконуються після побудови екрана, гарантуючи доступність контактів HAL віджета. У файлі INI може бути кілька рядків «POSTGUI_HALFILE=<ім'я файлу>». Кожен з них буде виконуватися по черзі в порядку їх появи.

Для отримання додаткової інформації про налаштування INI-файлу, які можуть змінити спосіб роботи AXIS, див. розділ Display розділу про конфігурацію INI.

10.1.12.1 Фільтри програми

AXIS має можливість надсилати завантажені файли через «програму фільтрації». Цей фільтр може виконувати будь-яке бажане завдання: від такого простого, як перевірка, чи закінчується файл на «M2», до такого складного, як генерація G-коду з зображення.

Розділ «[FILTER]» файлу INI контролює роботу фільтрів. Спочатку для кожного типу файлу напишіть рядок «PROGRAM_EXTENSION». Потім вкажіть програму, яку потрібно виконати для кожного типу файлу. Ця програма отримує ім'я вхідного файлу як перший аргумент і повинна записувати код rs274ngc у стандартний вивід. Цей вивід буде відображатися в текстовій області, попередньо переглядатися в області відображення та виконуватися LinuxCNC при натисканні «Run». Наступні рядки додають підтримку конвертера «image-to-gcode», що входить до складу LinuxCNC:

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

Також можна вказати інтерпретатора:

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

Таким чином, можна відкрити будь-який скрипт Python, а його вихідні дані будуть оброблятися як G-код. Один із таких прикладів скрипту доступний за адресою «nc_files/holecircle.py». Цей скрипт створює G-код для свердління низки отворів по колу.

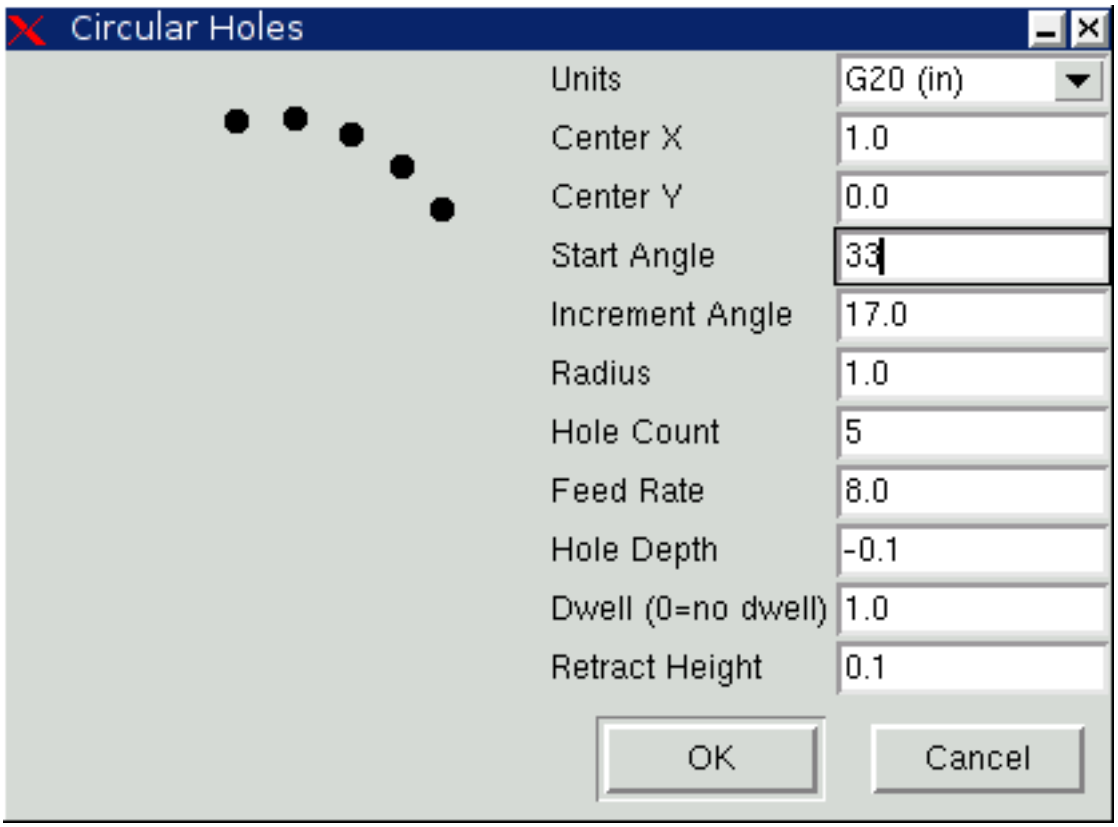


Figure 10.13: Круглі отвори

Якщо встановлено змінну середовища AXIS_PROGRESS_BAR, то рядки, що записуються в stderr, мають вигляд

```
FILTER_PROGRESS=%d
```

встановить індикатор виконання AXIS на заданий відсоток. Цю функцію слід використовувати будь-яким фільтром, який працює протягом тривалого часу.

10.1.12.2 База даних ресурсів X

Кольори більшості елементів інтерфейсу користувача AXIS можна налаштувати за допомогою бази даних X Resource Database. Зразок файлу «axis_light_background» змінює кольори вікна backplot на схему «темні лінії на білому тлі», а також служить довідником для настроюваних елементів в області відображення. Зразок файлу «axis_big_dro» змінює розмір шрифту для відображенн позиції на більший. Щоб використовувати ці файли:

```
xrdb -merge /usr/share/doc/emc2/axis_light_background

xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

Щоб отримати інформацію про інші елементи, які можна налаштувати в програмах Tk, див. сторінки довідки Tk.

Оскільки сучасні настільні середовища автоматично встановлюють деякі параметри в базі даних ресурсів X, які негативно впливають на AXIS, за замовчуванням ці параметри ігноруються. Щоб елементи бази даних ресурсів X замінили стандартні параметри AXIS, додайте наступний рядок у ваші ресурси X:

```
*AXIS*optionLevel: widgetDefault
```

це призводить до створення вбудованих опцій на рівні опцій *widgetDefault*, щоб ресурси X (які мають рівень *userDefault*) могли їх перевизначити.

10.1.12.3 Джоґвіст

Для поліпшення взаємодії AXIS з фізичним джог-колесом поточна активна вісь, вибрана в графічному інтерфейсі користувача, також повідомляється на «контакті HAL» з назвою типу «axisui.jog.x». За винятком короткого проміжку часу після зміни поточної осі, тільки один з цих контактів одночасно має значення «TRUE», інші залишаються «FALSE».

Після того, як AXIS створив ці «піни HAL», він запускає HAL-файл, оголошений як: [HAL]POSTGUI_HAL Це відрізняється від [HAL]HALFILE, який можна використовувати лише один раз.

10.1.12.4 ~/.axisrc

Якщо він існує, вміст `~/.axisrc` виконується як вихідний код Python безпосередньо перед відображенн графічного інтерфейсу AXIS. Деталі того, що може бути записано в `~/.axisrc`, можуть змінюватися протягом циклу розробки.

Наступний код додає Control-Q як комбінацію клавіш для виходу.

Приклад файлу .axisrc

```
root_window.bind("<Control-q>", "destroy .")
help2.append(("Control-Q", "Quit"))
```

Наступна команда зупиняє діалогове вікно «Ви дійсно хочете вийти?».

```
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

10.1.12.5 USER_COMMAND_FILE

Файл Python для конкретної конфігурації можна вказати за допомогою параметра INI-файлу «[DISPLAY]USER_COMMAND_FILE=filename.py». Як і файл «~/.axisrc», цей файл завантажується безпосередньо перед відображенням графічного інтерфейсу AXIS. Цей файл стосується саме конфігурації INI-файлу, а не домашнього каталогу користувача.

10.1.12.6 user_live_update()

GUI AXIS включає функцію no-op (заповнювач) під назвою «user_live_update()», яка виконується після завершення функції update() класу LivePlotter. Ця функція може бути реалізована в скрипті Python `~/.axisrc` або скрипті Python «[DISPLAY]USER_COMMAND_FILE» для виконання періодичних дій, визначених користувачем. Деталі того, що можна виконати в цій функції, залежать від реалізації AXIS GUI і можуть змінюватися протягом циклу розробки.

10.1.12.7 user_hal_pins()

GUI AXIS включає функцію no-op (заповнювач) під назвою «user_hal_pins()».
Вона виконується відразу після виклику файлу .axisrc і безпосередньо перед ініціалізацією будь-яких панелей GladeVCP / вбудованих вкладок.
Ця функція може бути реалізована в скрипті Python `~/.axisrc` або скрипті Python «[DISPLAY]USER_CO
для створення власних контактів HAL, які використовують префікс «axisui.».
Використовуйте «comp» як посилання на екземпляр компонента HAL.
HAL comp.ready() викликається відразу після повернення цієї функції.

10.1.12.8 Зовнішній редактор

Опції меню Файл > Редагувати... та Файл > Редагувати таблицю інструментів... стають доступними після визначення редактора в розділі INI [DISPLAY]. Корисні значення включають `EDITOR=gedit` та `EDITOR=gnome-terminal -e vim`. Для отримання додаткової інформації див. розділ Display Section глави «Конфігурація INI».

10.1.12.9 Віртуальна панель керування

AXIS може відображати настроювану віртуальну панель керування у правій колонці або в нижньому рядку. Крім того, одна або кілька панелей можуть відображатися у вигляді вбудованих вкладок. Ви можете програмувати кнопки, індикатори, відображення даних тощо. Для отримання додаткової інформації див. розділи PyVCP та GladeVCP.

10.1.12.10 Контроль попереднього перегляду

У файл G-коду можна вставляти спеціальні коментарі, щоб контролювати поведінку попереднього перегляду AXIS. Якщо ви хочете обмежити відображення попереднього перегляду, використовуйте ці спеціальні коментарі. Все, що знаходиться між (AXIS,hide) і (AXIS,show), не буде відображатися під час попереднього перегляду. (AXIS,hide) і (AXIS,show) повинні використовуватися парами, причому (AXIS,hide) має бути першим. Все, що знаходиться після (AXIS,stop), не буде відображатися під час попереднього перегляду.

Ці коментарі корисні для розвантаження попереднього перегляду (наприклад, під час налагодження більшого файлу G-коду можна вимкнути попередній перегляд для певних деталей, які вже працюють нормально).

- (AXIS,hide) Зупиняє попередній перегляд (має бути першим)
- (AXIS,показати) Відновлює попередній перегляд (має слідувати за приховуванням)
- (AXIS,стоп) Зупиняє попередній перегляд звідси і до кінця файлу.
- (AXIS,notify,the_text) Відображає the_text як інформаційний дисплей

Це відображення може бути корисним у попередньому перегляді AXIS, коли коментарі (debug,message не відображаються.

10.1.12.11 Доторкніться до місця призначення, використовуючи фактичне положення

Функція Touch Off може опціонально включати фактичне значення положення осі в розрахунок зміщення. Це в основному використовується в випадках, коли немоторизована вісь, така як піноль у фрезерному верстаті, надає зворотний зв'язок LinuxCNC через енкодер, але немає двигуна для управління рухом. Це дозволяє AXIS надавати дисплей DRO для такої осі з робочою функцією touch off.

Ця функція вмикається на осі шляхом зміни відповідного розділу «[AXIS_x]» у файлі .INI. Додайте опцію з назвою «TOUCHOFF_ACTUAL» і встановіть значення «PLUS» або «MINUS» залежно від того, як ви хочете застосувати фактичне положення до зміщення.

Приклад:

```
[AXIS_Z]
TOUCHOFF_ACTUAL = MINUS
```

Зазвичай для встановлення цього зміщення використовується тільки задане положення осі, що означає, що воно не працює належним чином, оскільки немоторизовані осі ніколи не отримують команди на рух, а отже, їхнє задане положення завжди дорівнює 0.

Функція Touch off надсилає команду «G10 L20» до MDI для встановлення нового значення зміщення. Застосовуване значення зазвичай є тим самим значенням, яке введено у діалоговому вікні. Коли ця функція увімкнена, вона додає або віднімає значення поточного положення від значення, введеного у діалоговому вікні, залежно від того, як вона налаштована.

10.1.13 Axisui

Для поліпшення взаємодії AXIS з фізичними джог-колесами, вісь, яка в даний момент вибрана в графічному інтерфейсі, також повідомляється на контакті з назвою типу «axisui.jog.x». Один з цих контактів одночасно має значення «TRUE», а решта — «FALSE». Вони призначені для керування контактами, що вмикають джог-режим руху.

Піни Axisui AXIS має контакти HAL, які вказують, яка радіокнопка поштовхового переміщення вибрана на вкладці «Ручне керування».

```
b''Bb''b''вb''b''eb''b''дb''b''ib''b''тb''b''ьb'' b''ib''b''мb'''b''яb'' b''дb''b''иb''b' ↩
    'рb''b''eb''b''кb''b''тb''b''ob''b''pb''b''ib''b''їb''
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 має контакт HAL для позначення кроку поштовху, вибраного на вкладці «Вручну».

```
b''Bb''b''вb''b''eb''b''дb''b''іb''b''тb''b''ьb'' b''іb''b''мb'''b''яb'' b''дb''b''иb''b' ↩
    'рb''b''eb''b''кb''b''тb''b''оb''b''рb''b''іb''b''їb''
float OUT axisui.jog.increment
```

AXIS має вихідний контакт HAL, який вказує на переривання. Контакт «axisui.abort» матиме значення «TRUE» та повернеться до значення «FALSE» через 0,3 мс.

```
Type  Dir    Name
bit   OUT    axisui.abort
```

AXIS має вихідний контакт HAL, який вказує на виникнення помилки. Контакт «axisui.error» залишатиметься в положенні «TRUE», доки всі сповіщення про помилки не будуть відхилені.

```
Type  Dir    Name
bit   OUT    axisui.error
```

AXIS має вхідні контакти HAL для очищення спливаючих повідомлень про помилки та інформацію.

```
b''Bb''b''вb''b''eb''b''дb''b''іb''b''тb''b''ьb''.  b''іb''b''мb'''b''яb''     b''дb''b' ↩
    'иb''b''рb''b''eb''b''кb''b''тb''b''оb''b''рb''b''іb''b''їb''
bit   IN     axisui.notifications-clear
bit   IN     axisui.notifications-clear-error
bit   IN     axisui.notifications-clear-info
```

AXIS має вхідний контакт HAL, який вмикає/вимикає функцію «Пауза/Відновлення».

```
Type  Dir    Name
bit   IN     axisui.resume-inhibit
```

10.1.14 Підказки щодо налаштування AXIS

AXIS є досить великою і складною для проникнення кодовою базою, що допомагає підтримувати стабільність коду, але ускладнює його налаштування.
Тут ми покажемо фрагменти коду для зміни поведінки або візуального вигляду екрану. Майте на увазі, що внутрішній код AXIS може час від часу змінюватися.
Не гарантується, що ці фрагменти коду будуть продовжувати працювати — вони можуть потребувати коригування.

10.1.14.1 Функція оновлення

У AXIS є функція під назвою user_live_update, яка викликається щоразу, коли AXIS оновлюється. Ви можете використовувати її для оновлення власних функцій.

```
# b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''яb'' b''пb''b''оb''b''сb''b''тb''b''іb''b' ↩
    'йb''b''нb''b''оb''b''гb''b''оb'' b''оb''b''нb''b''оb''b''вb''b''лb''b''eb''b''нb''b' ↩
    'нb''b''яb''
def user_live_update():
    print('i am printed every update...')
```

10.1.14.2 Вимкнути діалогове вікно закриття

```
# b''вb''b''иb''b''мb''b''кb''b''нb''b''іb''b''тb''b''ьb'' b''дb''b''іb''b''аb''b''лb''b' ↩
    'оb''b''гb''b''оb''b''вb''b''eb'' b''вb''b''іb''b''кb''b''нb''b''оb'' "b''3b''b''аb''b' ↩
    'кb''b''рb''b''иb''b''тb''b''иb''"
root_window.tk.call("wm","protocol",".","WM_DELETE_WINDOW","destroy .")
```

10.1.14.3 Зміна шрифту тексту

```
# b''зb''b''мb''b''іb''b''нb''b''иb''b''тb''b''иb'' b''шb''b''рb''b''иb''b''фb''b''тb''

font = 'sans 11'
fname,fsize = font.split()
root_window.tk.call('font','configure','TkDefaultFont','-family',fname,'-size',fsize)

# b''пb''b''еb''b''рb''b''еb''b''рb''b''оb''b''бb''b''иb''b''тb''b''иb'' b''тb''b''еb''b' ↩
    'кb''b''сb''b''тb'' b''уb'' b''вb''b''кb''b''лb''b''аb''b''дb''b''кb''b''аb''b''хb'', b' ↩
    'щb''b''оb''b''бb'' b''вb''b''оb''b''нb''b''иb'' b''зb''b''мb''b''іb''b''нb''b''юb''b' ↩
    'вb''b''аb''b''лb''b''иb'' b''рb''b''оb''b''зb''b''мb''b''іb''b''рb'' b''вb''b''іb''b' ↩
    'дb''b''пb''b''оb''b''вb''b''іb''b''дb''b''нb''b''оb'' b''дb''b''оb'' b''нb''b''оb''b' ↩
    'вb''b''оb''b''гb''b''оb'' b''шb''b''рb''b''иb''b''фb''b''тb''b''уb'' b''зb''b''аb'' b' ↩
    'зb''b''аb''b''мb''b''оb''b''вb''b''чb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb''b' ↩
    'мb''

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 ')


# b''Шb''b''рb''b''иb''b''фb''b''тb'' G-b''кb''b''оb''b''дb''b''уb'' b''єb'' b''нb''b''еb'' ↩
    b''зb''b''аb''b''лb''b''еb''b''жb''b''нb''b''иb''b''мb''

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 Зміна швидкості переміщення за допомогою комбінацій клавіш

```
# b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b''йb''b' ↩
    'тb''b''еb'' control + ' b''аb''b''бb''b''оb'' 1-0 b''яb''b''кb'' b''кb''b''оb''b''мb''b ↩
    ''бb''b''іb''b''нb''b''аb''b''цb''b''іb''b''їb'' b''кb''b''лb''b''аb''b''вb''b''іb''b' ↩
    'шb'' b''дb''b''лb''b''яb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''їb'' b''шb''b' ↩
    'вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b''іb'' b''тb''b''аb'' keep ' b''аb''b' ↩
    'бb''b''оb'' 1-0 b''дb''b''лb''b''яb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb'' ↩
    b''тb''b''іb'' b''пb''b''оb''b''дb''b''аb''b''чb''b''іb''
# b''тb''b''аb''b''кb''b''оb''b''жb'' b''дb''b''оb''b''дb''b''аb''b''єb'' b''тb''b''еb''b' ↩
    'кb''b''сb''b''тb'' b''дb''b''лb''b''яb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b' ↩
    'гb''b''оb'' b''пb''b''оb''b''сb''b''иb''b''лb''b''аb''b''нb''b''нb''b''яb'' b''вb'' b' ↩
    'дb''b''оb''b''вb''b''іb''b''дb''b''цb''b''іb''

help1.insert(10,("Control+ ',1..9,0", _("b''Bb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b ↩
    ''іb''b''тb''b''ьb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''еb'' b''пb''b''еb''b''рb''b' ↩
    'еb''b''мb''b''иb''b''кb''b''аb''b''нb''b''нb''b''яb'' b''вb''b''іb''b''дb'' 0% b''дb''b ↩
    ''оb'' 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 Прочитайте INI-файл

```
# b''пb''b''pb''b''ob''b''чb''b''иb''b''тb''b''ab''b''тb''b''иb'' b''eb''b''лb''b''eb''b' ↩
    'мb''b''eb''b''нb''b''тb'' INI-b''фb''b''ab''b''йb''b''лb''b''уb''
machine = inifile.find('EMC','MACHINE')
print('machine name =',machine)
```

10.1.14.6 Зчитування статусу LinuxCNC

```
# b''Cb''b''тb''b''ab''b''нb'' LinuxCNC b''мb''b''ob''b''жb''b''нb''b''ab'' b''пb''b''pb''b ↩
    ''ob''b''чb''b''иb''b''тb''b''ab''b''тb''b''иb'' b''зb'' s.
print(s.actual_position)
print(s.paused)
```

10.1.14.7 Змінити поточний вигляд

```
# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''вb''b''иb''b' ↩
    'гb''b''лb''b''яb''b''дb'' b''пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b' ↩
    'ьb''b''ob''b''гb''b''ob'' b''пb''b''eb''b''pb''b''eb''b''гb''b''лb''b''яb''b''дb''b' ↩
    'уb''
# b''дb''b''ib''b''йb''b''cb''b''нb''b''ib'' b''вb''b''иb''b''гb''b''лb''b''яb''b''дb''b' ↩
    'иb'': view_x, view_y, view_y2, view_z, view_z2, view_p
commands.set_view_z()
```

10.1.14.8 Створення нових пінів AXISUI HAL

```
def user_hal_pins():
    comp.newpin('my-new-in-pin', hal.HAL_BIT, hal.HAL_IN)
    comp.ready()
```

10.1.14.9 Створення нового компонента та виводів HAL

```
# b''cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b' ↩
    'ob''b''нb''b''eb''b''нb''b''тb''

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()

# b''зb''b''єb''b''дb''b''нb''b''аb''b''тb''b''иb'' b''кb''b''оb''b''нb''b''тb''b''аb''b' ↩
    'кb''b''тb''b''иb''

hal.new_sig('idle-led',hal.HAL_BIT)
hal.connect('halui.program.is-idle','idle-led')
hal.connect('my_component.idle-led','idle-led')

# b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' b''шb''b''пb''b' ↩
    'иb''b''лb''b''ьb''b''кb''b''уb''

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 Перемикання вкладок за допомогою контактів HAL

```
# b''Пb''b''іb''b''нb''b''иb'' HAL b''зb'' b''пb''b''аb''b''нb''b''еb''b''лb''b''іb''  ↩
    GladeVCP b''нb''b''еb'' b''бb''b''уb''b''дb''b''уb''b''тb''b''ьb'' b''гb''b''оb''b''тb'' ↩
    b''оb''b''вb''b''іb'' b''дb''b''оb'' b''вb''b''иb''b''кb''b''оb''b''нb''b''аb''b''нb''b' ↩
    'нb''b''яb'' user_live_update
# b''щb''b''оb''b''бb'' b''їb''b''хb'' b''пb''b''рb''b''оb''b''чb''b''иb''b''тb''b''аb''b' ↩
    'тb''b''иb'', b''пb''b''оb''b''тb''b''рb''b''іb''b''бb''b''нb''b''оb'' b''пb''b''оb''b' ↩
    'мb''b''іb''b''сb''b''тb''b''иb''b''тb''b''иb'' b''їb''b''хb'' b''уb'' b''бb''b''лb''b' ↩
    'оb''b''кb'' try/except

# b''Уb'' b''нb''b''аb''b''сb''b''тb''b''уb''b''пb''b''нb''b''оb''b''мb''b''уb'' b''пb''b' ↩
    'рb''b''иb''b''кb''b''лb''b''аb''b''дb''b''іb'' b''пb''b''еb''b''рb''b''еb''b''дb''b' ↩
    'бb''b''аb''b''чb''b''аb''b''єb''b''тb''b''ьb''b''сb''b''яb'' b''нb''b''аb''b''яb''b' ↩
    'вb''b''нb''b''іb''b''сb''b''тb''b''ьb'' 5 b''кb''b''нb''b''оb''b''пb''b''оb''b''кb''  ↩
    HAL b''нb''b''аb'' b''пb''b''аb''b''нb''b''еb''b''лb''b''іb'' GladeVCP, b''яb''b''кb''b' ↩
    'іb'' b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'юb''b''тb''b''ьb''b''сb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b''еb''b''рb''b''еb''b' ↩
    'мb''b''иb''b''кb''b''аb''b''нb''b''нb''b''яb''
# b''вb''b''кb''b''лb''b''аb''b''дb''b''оb''b''кb'' b''нb''b''аb'' b''еb''b''кb''b''рb''b' ↩
    'аb''b''нb''b''іb'' AXIS.
# b''Нb''b''аb''b''зb''b''вb''b''иb'' b''кb''b''нb''b''оb''b''пb''b''оb''b''кb'': «manual- ↩
    tab», «mdi-tab», «preview-tab», «dro-tab», «user0-tab».
# b''Вb''b''кb''b''лb''b''аb''b''дb''b''кb''b''аb'' user_0, b''яb''b''кb''b''щb''b''оb'' b' ↩
    'вb''b''оb''b''нb''b''аb'' b''іb''b''сb''b''нb''b''уb''b''єb'', b''бb''b''уb''b''дb''b' ↩
    'еb'' b''пb''b''еb''b''рb''b''шb''b''оb''b''юb'' b''вb''b''бb''b''уb''b''дb''b''оb''b' ↩
    'вb''b''аb''b''нb''b''оb''b''юb'' b''вb''b''кb''b''лb''b''аb''b''дb''b''кb''b''оb''b' ↩
    'юb'' GladeVCP.

# b''дb''b''лb''b''яb'' b''гb''b''іb''b''лb''b''кb''b''иb'' LinuxCNC 2.8+

def user_live_update():
    try:
        if hal.get_value('gladevcp.manual-tab'):
            root_window.tk.call('.pane.top.tabs','raise','manual')
        elif hal.get_value('gladevcp.mdi-tab'):
            root_window.tk.call('.pane.top.tabs','raise','mdi')
        elif hal.get_value('gladevcp.preview-tab'):
            root_window.tk.call('.pane.top.right','raise','preview')
        elif hal.get_value('gladevcp.numbers-tab'):
```

```
            root_window.tk.call('.pane.top.right','raise','numbers')
        elif hal.get_value('gladevcp.user0-tab'):
            root_window.tk.call('.pane.top.right','raise','user_0')
    except:
        pass
```

10.1.14.11 Додати кнопку «Перейти на головну»

```
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)

# b''cb''b''tb''b''вb''b''ob''b''pb''b''иb''b''tb''b''иb'' b''кb''b''нb''b''ob''b''пb''b' ↩
    'кb''b''yb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''іb''b''щb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''пb''b''ob'' b''ob''b''cb''b''іb'' Y
root_window.tk.call('button','.pane.top.tabs.fmanual.homey','-text','Home Y','-command',' ↩
    goto_home Y','-height','2')

# b''пb''b''ob''b''мb''b''іb''b''cb''b''тb''b''іb''b''тb''b''ьb'' b''кb''b''нb''b''ob''b' ↩
    'пb''b''кb''b''yb''
root_window.tk.call('grid','.pane.top.tabs.fmanual.homey','-column','1','-row','7','- ↩
    columnspan','2','-padx','4','-sticky','w')

# b''бb''b''yb''b''дb''b''ьb''-b''яb''b''кb''b''yb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ↩
    'іb''b''юb'', b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''нb''b''yb'' b''зb''  ↩
    Tcl, b''пb''b''ob''b''тb''b''pb''b''іb''b''бb''b''нb''b''ob'' b''дb''b''ob''b''дb''b' ↩
    'аb''b''тb''b''иb'' b''дb''b''ob'' TclCommands
TclCommands.goto_home = goto_home
commands = TclCommands(root_window)
```

10.1.14.12 Додати кнопку до ручної рамки

```
# b''cb''b''tb''b''вb''b''ob''b''pb''b''иb''b''tb''b''иb'' b''нb''b''ob''b''вb''b''yb'' b' ↩
    'кb''b''нb''b''ob''b''пb''b''кb''b''yb'' b''tb''b''аb'' b''пb''b''ob''b''мb''b''іb''b' ↩
    'cb''b''tb''b''иb''b''tb''b''иb'' b''їb''b''їb'' b''вb'' b''pb''b''yb''b''чb''b''нb''b' ↩
    'yb'' b''pb''b''аb''b''мb''b''кb''b''yb''

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')

# b''вb''b''иb''b''щb''b''eb''b''зb''b''аb''b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b' ↩
    'eb'' b''нb''b''аb''b''дb''b''cb''b''иb''b''лb''b''аb''b''єb'' b''кb''b''ob''b''мb''b' ↩
    'аb''b''нb''b''дb''b''yb'' «mybutton_clicked» b''пb''b''pb''b''иb'' b''нb''b''аb''b' ↩
    'tb''b''иb''b''cb''b''кb''b''аb''b''нb''b''нb''b''іb''
# b''іb''b''нb''b''шb''b''іb'' b''ob''b''пb''b''цb''b''іb''b''їb'' b''дb''b''ob''b''зb''b' ↩
    'вb''b''ob''b''лb''b''яb''b''юb''b''tb''b''ьb'' b''пb''b''pb''b''иb''b''вb''b''яb''b' ↩
    'зb''b''аb''b''tb''b''иb'' b''дb''b''ob'' b''кb''b''нb''b''ob''b''пb''b''кb''b''иb'' b' ↩
    'кb''b''ob''b''мb''b''аb''b''нb''b''дb''b''иb'' b''нb''b''аb''b''tb''b''иb''b''cb''b' ↩
    'кb''b''аb''b''нb''b''нb''b''яb'' b''аb''b''бb''b''ob'' b''вb''b''іb''b''дb''b''пb''b' ↩
    'yb''b''cb''b''кb''b''аb''b''нb''b''нb''b''яb'' (b''аb''b''бb''b''ob'' b''ob''b''бb''b' ↩
    'иb''b''дb''b''вb''b''іb'')
```

```
# b''вb''b''оb''b''нb''b''иb'' b''мb''b''оb''b''жb''b''уb''b''тb''b''ьb'' b''бb''b''уb''b' ↩
    'тb''b''иb'' b''дb''b''оb''b''дb''b''аb''b''тb''b''кb''b''оb''b''вb''b''иb''b''мb''b' ↩
    'иb'' b''дb''b''оb'' b''кb''b''оb''b''мb''b''аb''b''нb''b''дb''b''иb'' b''нb''b''аb''b' ↩
    'тb''b''иb''b''сb''b''кb''b''аb''b''нb''b''нb''b''яb'' b''аb''b''бb''b''оb'' b''зb''b' ↩
    'аb''b''мb''b''іb''b''нb''b''юb''b''вb''b''аb''b''тb''b''иb'' b''їb''b''їb''
# b''яb''b''кb''b''щb''b''оb'' b''зb''b''аb''b''мb''b''іb''b''сb''b''тb''b''ьb'' b''цb''b' ↩
    'ьb''b''оb''b''гb''b''оb'', b''вb''b''иb''b''дb''b''аb''b''лb''b''іb''b''тb''b''ьb'' «- ↩
    command','mybutton_clicked» b''зb'' b''пb''b''еb''b''рb''b''шb''b''оb''b''гb''b''оb'' b' ↩
    'рb''b''яb''b''дb''b''кb''b''аb''

# b''Кb''b''нb''b''оb''b''пb''b''кb''b''аb'' 1 = b''лb''b''іb''b''вb''b''аb'' b''кb''b' ↩
    'нb''b''оb''b''пb''b''кb''b''аb'' b''мb''b''иb''b''шb''b''іb'', 2 = b''пb''b''рb''b' ↩
    'аb''b''вb''b''аb'' b''аb''b''бb''b''оb'' 3 = b''сb''b''еb''b''рb''b''еb''b''дb''b''нb'' ↩
    b''яb''

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')

# b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''їb'', b''щb''b''оb'' b''вb''b''иb''b''кb''b' ↩
    'лb''b''иb''b''кb''b''аb''b''юb''b''тb''b''ьb''b''сb''b''яb'' b''зb'' b''кb''b''нb''b' ↩
    'оb''b''пb''b''оb''b''кb''

def mybutton_clicked():
    print('mybutton was clicked')
def mybutton_pressed():
    print('mybutton was pressed')
def mybutton_released():
    print('mybutton was released')

# b''бb''b''уb''b''дb''b''ьb''-b''яb''b''кb''b''уb'' b''фb''b''уb''b''нb''b''кb''b''цb''b' ↩
    'іb''b''юb'', b''щb''b''оb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''єb''b' ↩
    'тb''b''ьb''b''сb''b''яb'' b''зb'' Tcl, b''пb''b''оb''b''тb''b''рb''b''іb''b''бb''b' ↩
    'нb''b''оb'' b''дb''b''оb''b''дb''b''аb''b''тb''b''иb'' b''дb''b''оb'' TclCommands

TclCommands.mybutton_clicked = mybutton_clicked
TclCommands.mybutton_pressed = mybutton_pressed
TclCommands.mybutton_released = mybutton_released
commands = TclCommands(root_window)
```

10.1.14.13 Читання внутрішніх змінних

```
# b''Нb''b''аb''b''сb''b''тb''b''уb''b''пb''b''нb''b''іb'' b''зb''b''мb''b''іb''b''нb''b' ↩
    'нb''b''іb'' b''мb''b''оb''b''жb''b''нb''b''аb'' b''зb''b''чb''b''иb''b''тb''b''аb''b' ↩
    'тb''b''иb'' b''зb'' b''еb''b''кb''b''зb''b''еb''b''мb''b''пb''b''лb''b''яb''b''рb''b' ↩
    'аb'' vars

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 Приховати віджети

```
# b''пb''b''pb''b''иb''b''хb''b''ob''b''вb''b''ab''b''тb''b''иb'' b''вb''b''ib''b''дb''b' ↩
    'жb''b''eb''b''тb''
# b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''вb''b' ↩
    'ab''b''тb''b''иb'' 'grid' b''ab''b''бb''b''ob'' 'pack' b''зb''b''ab''b''лb''b''eb''b' ↩
    'жb''b''нb''b''ob'' b''вb''b''ib''b''дb'' b''тb''b''ob''b''гb''b''ob'', b''яb''b''кb'' b ↩
```

```
    ’’вb’’b’’іb’’b’’нb’’ b’’бb’’b’’уb’’b’’вb’’ b’’сb’’b’’пb’’b’’оb’’b’’чb’’b’’аb’’b’’тb’’b’ ↩
    ’кb’’b’’уb’’ b’’рb’’b’’оb’’b’’зb’’b’’мb’’b’’іb’’b’’щb’’b’’еb’’b’’нb’’b’’иb’’b’’йb’’
root_window.tk.call(’grid’,’forget’,’.pane.top.tabs.fmanual.jogf.zerohome.tooltouch’)
```

10.1.14.15 Змінити мітку

```
# b’’зb’’b’’мb’’b’’іb’’b’’нb’’b’’иb’’b’’тb’’b’’иb’’ b’’мb’’b’’іb’’b’’тb’’b’’кb’’b’’уb’’ b’ ↩
    ’вb’’b’’іb’’b’’дb’’b’’жb’’b’’еb’’b’’тb’’b’’аb’’
root_window.tk.call(’setup_widget_accel’,’.pane.top.tabs.fmanual.mist’,’Downdraft’)

# b’’пb’’b’’еb’’b’’рb’’b’’еb’’b’’кb’’b’’оb’’b’’нb’’b’’аb’’b’’йb’’b’’тb’’b’’еb’’b’’сb’’b’ ↩
    ’яb’’, b’’щb’’b’’оb’’ b’’вb’’b’’оb’’b’’нb’’b’’аb’’ b’’вb’’b’’іb’’b’’дb’’b’’оb’’b’’бb’’b’ ↩
    ’рb’’b’’аb’’b’’жb’’b’’аb’’b’’єb’’b’’тb’’b’’ьb’’b’’сb’’b’’яb’’ (b’’пb’’b’’оb’’b’’тb’’b’ ↩
    ’рb’’b’’іb’’b’’бb’’b’’нb’’b’’оb’’ b’’лb’’b’’иb’’b’’шb’’b’’еb’’ b’’вb’’ b’’цb’’b’’ьb’’b’ ↩
    ’оb’’b’’мb’’b’’уb’’ b’’вb’’b’’иb’’b’’пb’’b’’аb’’b’’дb’’b’’кb’’b’’уb’’, b’’яb’’b’’кb’’b’ ↩
    ’щb’’b’’оb’’ b’’кb’’b’’нb’’b’’оb’’b’’пb’’b’’кb’’b’’аb’’ b’’тb’’b’’уb’’b’’мb’’b’’аb’’b’ ↩
    ’нb’’b’’уb’’ b’’бb’’b’’уb’’b’’лb’’b’’аb’’ b’’пb’’b’’рb’’b’’иb’’b’’хb’’b’’оb’’b’’вb’’b’ ↩
    ’аb’’b’’нb’’b’’аb’’)
root_window.tk.call(’grid’,’.pane.top.tabs.fmanual.mist’,’-column’,’1’,’-row’,’5’,’- ↩
    columnspan’,’2’,’-padx’,’4’,’-sticky’,’w’)
```

10.1.14.16 Перенаправити існуючу команду

```
# b’’зb’’b’’аb’’b’’хb’’b’’оb’’b’’пb’’b’’иb’’b’’тb’’b’’иb’’ b’’іb’’b’’сb’’b’’нb’’b’’уb’’b’ ↩
    ’юb’’b’’чb’’b’’уb’’ b’’кb’’b’’оb’’b’’мb’’b’’аb’’b’’нb’’b’’дb’’b’’уb’’
# b’’сb’’b’’пb’’b’’оb’’b’’чb’’b’’аb’’b’’тb’’b’’кb’’b’’уb’’ b’’кb’’b’’нb’’b’’оb’’b’’пb’’b’ ↩
    ’кb’’b’’аb’’ mist b’’вb’’b’’иb’’b’’кb’’b’’лb’’b’’иb’’b’’кb’’b’’аb’’b’’єb’’ b’’фb’’b’ ↩
    ’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’’юb’’ mist
root_window.tk.call(’.pane.top.tabs.fmanual.mist’,’configure’,’-command’,’hijacked_command ↩
    ’)

# b’’Нb’’b’’оb’’b’’вb’’b’’аb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’’яb’’
def hijacked_command():
    print(’hijacked mist command’)

# b’’дb’’b’’оb’’b’’дb’’b’’аb’’b’’тb’’b’’иb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’ ↩
    ’юb’’ b’’дb’’b’’оb’’ TclCommands
TclCommands.hijacked_command = hijacked_command
commands = TclCommands(root_window)
```

10.1.14.17 Зміна кольору DRO

```
# b’’зb’’b’’мb’’b’’іb’’b’’нb’’b’’иb’’b’’тb’’b’’иb’’ b’’еb’’b’’кb’’b’’рb’’b’’аb’’b’’нb’’ b’ ↩
    ’пb’’b’’аb’’b’’дb’’b’’іb’’b’’нb’’b’’нb’’b’’яb’’
root_window.tk.call(’.pane.top.right.fnumbers.text’,’configure’,’-foreground’,’green’,’- ↩
    background’,’black’)
```

10.1.14.18 Зміна кнопок панелі інструментів

```
# b’’зb’’b’’мb’’b’’іb’’b’’нb’’b’’иb’’b’’тb’’b’’иb’’ b’’кb’’b’’нb’’b’’оb’’b’’пb’’b’’кb’’b’ ↩
    ’иb’’ b’’пb’’b’’аb’’b’’нb’’b’’еb’’b’’лb’’b’’іb’’ b’’іb’’b’’нb’’b’’сb’’b’’тb’’b’’рb’’b’ ↩
    ’уb’’b’’мb’’b’’еb’’b’’нb’’b’’тb’’b’’іb’’b’’вb’’
```

```
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 Зміна кольорів плоттера

У форматі RGBA, у такому порядку: штовхання, швидка, подача, дуга, зміна інструменту, зонд

```
# b''зb''b''мb''b''іb''b''нb''b''иb''b''тb''b''иb'' b''кb''b''оb''b''лb''b''ьb''b''оb''b' ↩
    'рb''b''иb'' b''пb''b''лb''b''оb''b''тb''b''еb''b''рb''b''аb''
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 Вступ

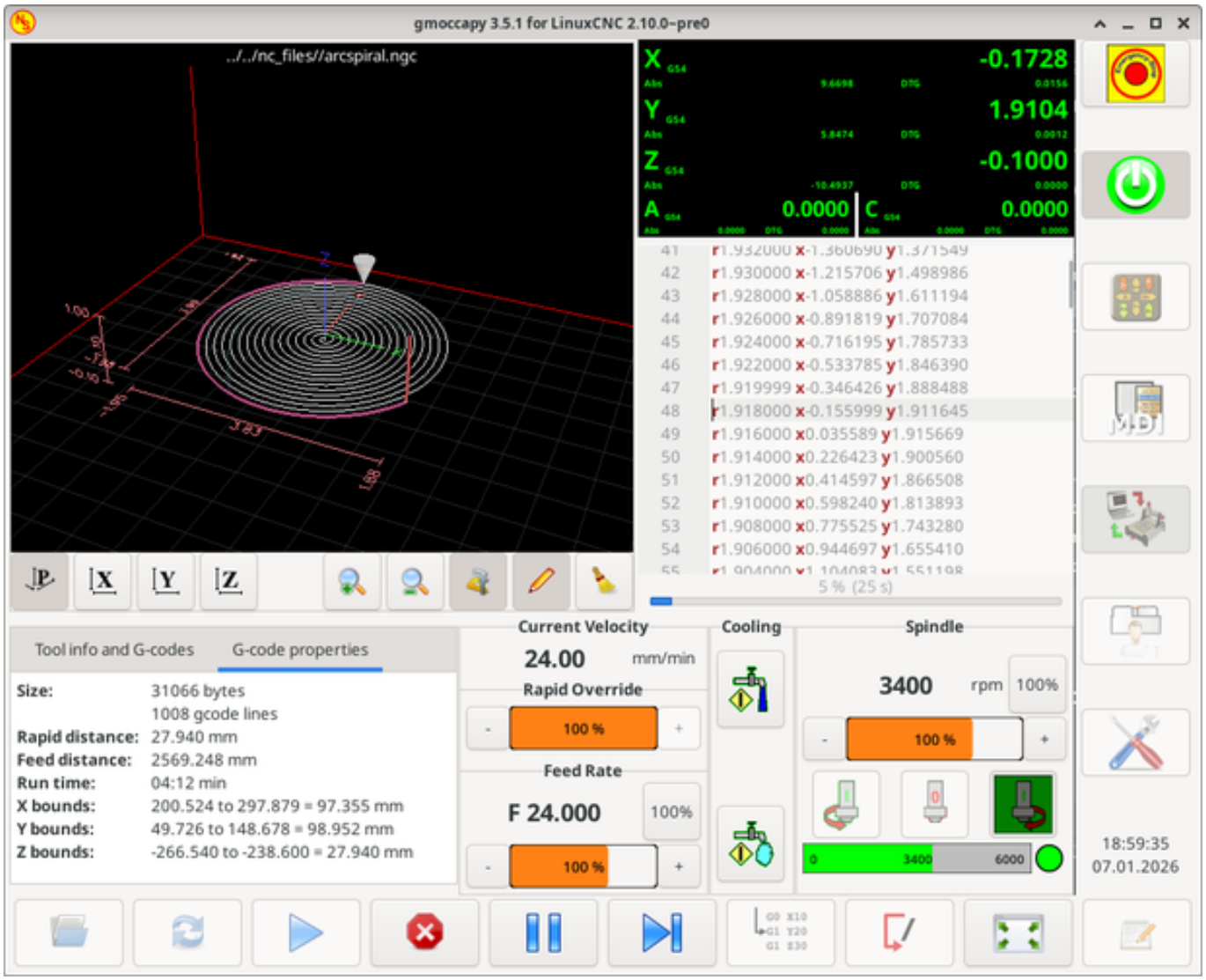
GMOCCAPY — це графічний інтерфейс для LinuxCNC, призначений для використання з сенсорним екраном, але його також можна використовувати на звичайних екранах за допомогою миші або апаратних кнопок і коліщаток MPG, оскільки він підтримує HAL Pins для найпоширеніших потреб. Більш детальну інформацію можна знайти нижче.

Він пропонує можливість відображення до 9 осей, підтримує режим токарного верстата для звичайного та зворотного інструменту і може бути адаптований практично до будь-яких потреб, оскільки GMOCCAPY підтримує вбудовані вкладки та бічні панелі. Хорошим прикладом цього є gmoccapy_plasma.

GMOCCAPY 3 підтримує до 9 осей та 9 суглобів. Оскільки код GMOCCAPY 3 був змінений для підтримки змін суглобів/осі в LinuxCNC, він не працює на гілках 2.7 або 2.6!

Він підтримує вбудовану віртуальну клавіатуру (вбудовану або клавіатуру matchbox), тому немає необхідності в апаратній клавіатурі або миші, але його також можна використовувати з цим обладнанням. GMOCCAPY пропонує окрему сторінку налаштувань для конфігурації більшості налаштувань графічного інтерфейсу без редагування файлів.

GMOCCAPY можна дуже легко локалізувати, оскільки відповідні файли відокремлені від файлів linuxcnc.po, тому немає потреби перекладати непотрібні елементи. Якщо ви хочете взяти участь у перекладі, скористайтеся посиланням:веб-редактор перекладів Weblate. Для отримання додаткової інформації див. розділ Translations



10.2.2 Вимоги

GMOCCAPY 3 було протестовано на Debian Jessie, Debian Stretch та MINT 18 з LinuxCNC master та версією 2.8. Воно повністю підтримує зміни з'єднань/осей LinuxCNC, що робить його придатним як графічний інтерфейс для Scara, роботів або будь-яких інших конфігурацій з більшою кількістю з'єднань, ніж осей. Тому воно також підтримує конфігурації портальних систем. Якщо ви використовуєте інші версії, повідомте про проблеми та/або рішення на форум LinuxCNC або http://www.cncecke.de/-forum/showthread.php?t=78549 [Німецький форум CNC Ecke] або Список розсилки користувачів LinuxCNC.

Мінімальна роздільна здатність екрану для GMOCCAPY для нормального макета (без бічних панелей) становить **980 x 750 пікселів**, тому вона повинна підходити для кожного стандартного екрану. Рекомендується використовувати екрани з мінімальною роздільною здатністю 1024x768. Існує також конфігурація, яка підходить для екранів 800x600 (введена в GMOCCAPY 3.4.8).

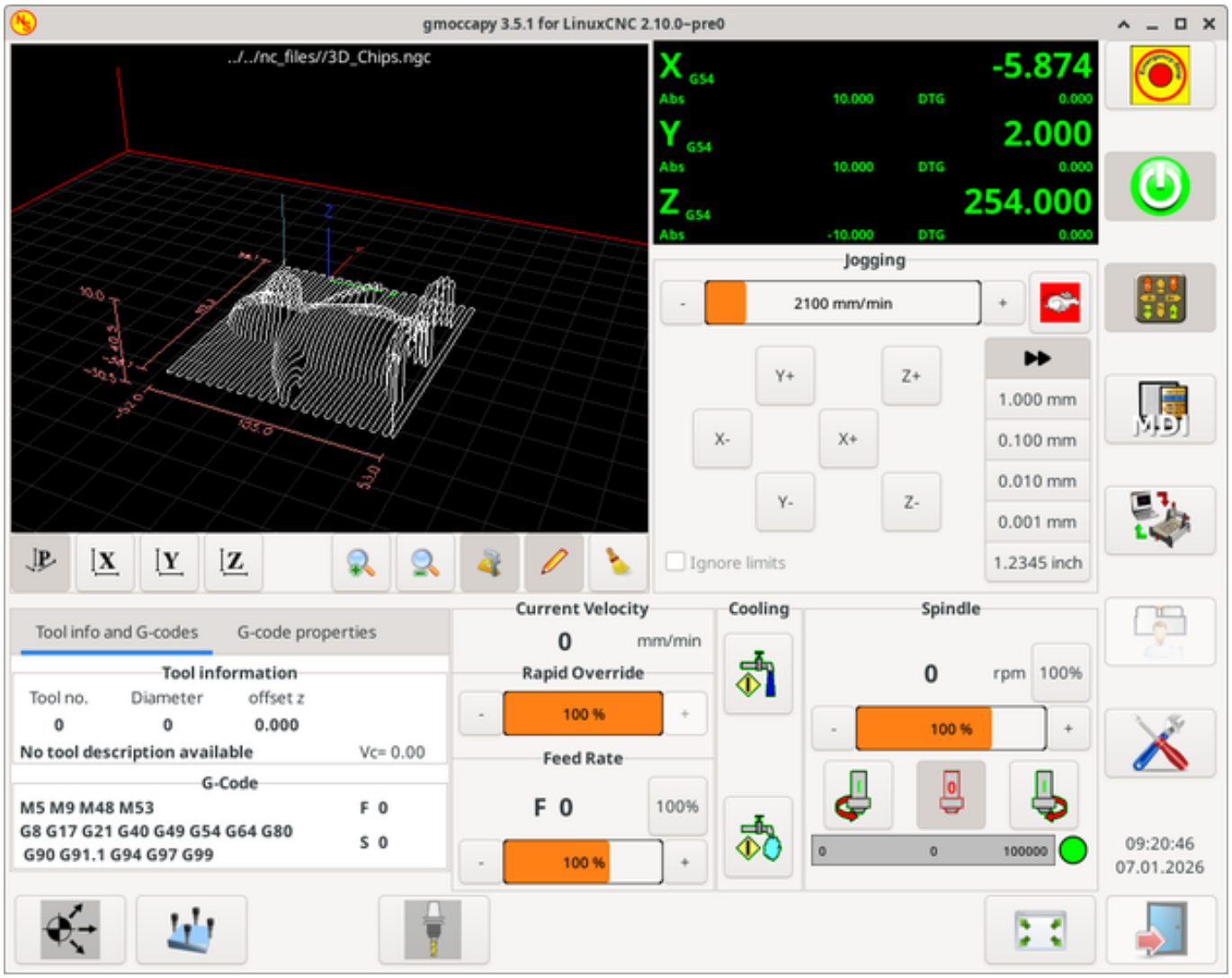
10.2.3 Як отримати GMOCCAPY

GMOCCAPY 3 входить до стандартного дистрибутива LinuxCNC починаючи з версії 2.7. Тому найпростіший спосіб отримати GMOCCAPY на ваш контрольний ПК — це просто завантажити

ISO і встановити його з CD/DVD/USB-накопичувача. Це дозволить вам отримувати оновлення разом із звичайними пакетами Debian.

За посиланням: gmoccapy_release_notes.txt[**нотатки до випуску**], також відомим як список змін, ви можете відстежувати останні виправлення помилок та нові функції.

Ви отримаєте екран, подібний до наступного (дизайн може відрізнятися залежно від вашої конфігураці



10.2.4 Базова конфігурація

GMOCCAPY 3 підтримує такі параметри командного рядка:

- *-user_mode*: Якщо встановлено, кнопка налаштування буде вимкнена, тому звичайні оператори машини не зможуть редагувати її налаштування.
- *-logo <path to logo file>*: Якщо вказано, логотип приховає вкладку кнопки поетапного перемикання в ручному режимі, це корисно лише для машин з апаратними кнопками для поетапного перемикання та вибору приросту.

Насправді для запуску GMOCCAPY не потрібно багато налаштовувати, але є деякі моменти, на які слід звернути увагу, якщо ви хочете використовувати всі можливості графічного інтерфейсу.

Ви знайдете багато конфігурацій симуляції (INI-файлів), просто щоб показати основи:

- 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
- і багато іншого...

Назви повинні пояснювати основне призначення різних INI-файлів.

Якщо ви використовуєте існуючу конфігурацію вашої машини, просто відредагуйте свій INI відповідно до цього документа.

Отже, давайте детальніше розглянемо INI-файл і те, що потрібно включити, щоб використовувати GMOCCAPY на вашому комп'ютері:

10.2.4.1 Розділ «ДИСПЛЕЙ»

```
[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* - Це вказує LinuxCNC використовувати GMOCCAPY.
- *PREFERENCE_FILE_PATH* - Вказує розташування та ім'я файлу налаштувань, який буде використову
  У більшості випадків цей рядок не буде потрібний, він використовується GMOCCAPY для зберігання ваших налаштувань графічного інтерфейсу, таких як теми, одиниці виміру DRO, кольори, налаштуван клавіатури тощо. Детальніше див. settings page.

Note
Якщо шлях або файл не вказані, GMOCCAPY буде використовувати за замовчуванням <your_machinename>.pref, якщо ім'я машини не вказано у вашому INI-файлі, він буде використовувати gmoccapy.pref. Файл буде збережений у вашому каталозі конфігурації, тому налаштування не будуть змішані, якщо ви використовуєте кілька конфігурацій. Якщо ви хочете використовувати один файл для декількох машин, вам потрібно включити PREFERENCE_FILE_PATH у ваш INI.

- *MAX_FEED_OVERRIDE = 1.5* - Встановлює максимальне значення зміни подачі, у наведеному прикладі вам буде дозволено змінити подачу на 150%.

Note
Якщо значення не вказано, буде встановлено значення 1.0.

- *MIN_SPINDLE_OVERRIDE = 0.5* та *MAX_SPINDLE_OVERRIDE = 1.2* - Дозволяє змінювати корекцію шпинделя в межах від 50% до 120%.

Note
Якщо значення не вказано, MIN буде встановлено на 0,1, а MAX на 1,0.

- *LATHE = 1* - Встановлює макет екрана для керування токарним верстатом.
- *BACK_TOOL_LATHE = 1* - є опціональним і перемикає вісь X так, як це потрібно для токарного верстата з задньою обробкою. Також клавіатурні скорочення будуть реагувати по-іншому. GMOCCAPY дозволяє налаштувати токарний верстат також з додатковими осями, тому ви можете використовувати також конфігурацію XZCW для токарного верстата.

Tip
Див. також Розділ.

- *PROGRAM_PREFIX = ../../nc_files/* – це запис, який вказує LinuxCNC/GMOCCAPY, де шукати файли NGC.

Note
Якщо не вказано, GMOCCAPY шукатиме файли NGC у такому порядку: спочатку linuxcnc/nc_files, а потім домашній каталог користувача.

- *DEFAULT_SPINDLE_SPEED* – Початкове значення для "Starting RPM", якщо значення відсутнє у файлі налаштувань або файл відсутній. Не матиме жодного ефекту з дійсним файлом налаштувань.
- *MIN_ANGULAR_VELOCITY* – Встановлює мінімальну швидкість поштовхового переміщення верстата для обертових осей.
- *MAX_ANGULAR_VELOCITY* – Встановлює максимальну швидкість поштовхового переміщення верстата для обертових осей.
- *DEFAULT_ANGULAR_VELOCITY* – Встановлює швидкість поштовхового переміщення верстата за замовчуванням для обертових осей.

10.2.4.2 Розділ TRAJ

- *DEFAULT_LINEAR_VELOCITY = 85.0* - Встановлює швидкість поштовхового переміщення верстата за замовчуванням.

  Note
  Якщо не встановлено, буде використано половину значення «MAX_LINEAR_VELOCITY». Якщо це значення також не вказано, значення за замовчуванням дорівнюватиме 180.

- *MAX_LINEAR_VELOCITY = 230.0* - Встановлює максимальну швидкість верстата. Це значення також буде максимальною лінійною швидкістю поштовхового переміщення.

  Note
  Якщо не встановлено, значення за замовчуванням становить 600.

10.2.4.3 Кнопки макросів

Ви можете додавати макроси до GMOCCAPY, подібно до способу Touchy. Макрос — це не що інше, як файл NGC. Ви можете виконувати повні програми CNC в режимі MDI, просто натиснувши одну кнопку. Для цього спочатку потрібно вказати шлях пошуку макросів:

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

Це встановлює шлях для пошуку макросів та інших підпрограм. Кілька шляхів до підпрограм можна розділяти символом ":".

Тоді вам просто потрібно додати такий розділ:

Налаштування п'яти макросів, які будуть відображатися у списку 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
```

Потім вам потрібно надати відповідні файли NGC, які повинні відповідати таким правилам:

- Ім'я файлу має точно збігатися з ім'ям, зазначеним у рядку макросу, тільки з розширенням ".ngc" (враховуючи регістр).
- Файл має містити підпрограму типу ***O<i_am_lost> sub***, назва підпрограми має точно збігатися (з урахуванням регістру) з назвою макросу.
- Файл має закінчуватися endsub ***O<i_am_lost> endsub***, а потім командою ***M2***.
- Файли потрібно розмістити в папці, зазначеній у вашому INI-файлі за допомогою параметра «SUBROUTINE_PATH» у розділі RS274NGC

Код між sub та endsub буде виконано натисканням відповідної кнопки макросу.

Note
У графічному інтерфейсі буде показано максимум 16 макросів. Через обмеження місця може знадобитися натиснути на стрілку, щоб перейти на іншу сторінку і відобразити приховані кнопки макросів. Кнопки макросів будуть відображатися в порядку записів у файлі INI. Немає помилки в тому, щоб розмістити більше 16 макросів у файлі INI, просто вони не будуть показані.

Note
Зразки макросів можна знайти в папці «macros», розташованій у папці GMOCCAPY sim. Якщо ви вказали кілька шляхів до підпрограм, вони будуть шукатися в порядку вказаних шляхів. Буде використано перший знайдений файл.

GMOCCAPY також прийматиме макроси, що запитують параметри, такі як:

```
[MACROS]
MACRO = go_to_position X-pos Y-pos Z-pos
```

Параметри мають бути розділені пробілами. У цьому прикладі викликається файл *go_to_position.ngc* з таким вмістом:

```
; b''Tb''b''eb''b''cb''b''тb''b''ob''b''вb''b''иb''b''йb'' b''фb''b''ab''b''йb''b''лb'' "b' ↩
    'пb''b''eb''b''pb''b''eb''b''йb''b''тb''b''иb'' b''дb''b''ob'' b''пb''b''ob''b''зb''b' ↩
    'иb''b''цb''b''ib''b''їb''"
; b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''cb''b''тb''b''иb''b''тb''b''ьb'' b''мb''b' ↩
    'ab''b''шb''b''иb''b''нb''b''yb'' b''вb'' b''зb''b''ab''b''дb''b''ab''b''нb''b''yb'' b' ↩
    'пb''b''ob''b''зb''b''иb''b''цb''b''ib''b''юb''

O<go_to_position> sub

G17
G21
G54
G61
G40
G49
G80
G90

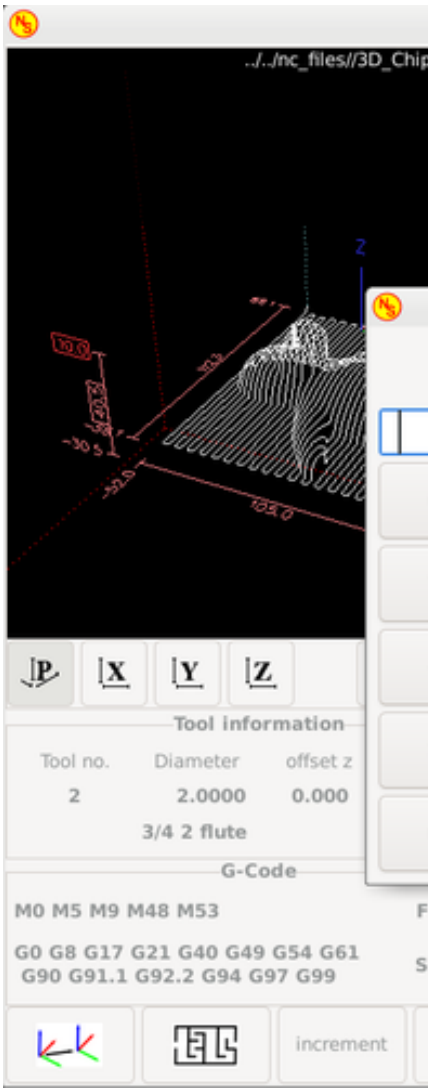
;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>

(b''Hb''b''Ab''b''Лb''b''Ab''b''Гb''b''Ob''b''Дb''b''Жb''b''Eb''b''Hb''b''Hb''b''Яb'', b' ↩
    'тb''b''eb''b''пb''b''eb''b''pb'' b''бb''b''yb''b''дb''b''eb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''мb''b''ib''b''щb''b''eb''b''нb''b''ob'' b''мb''b''ab''b''шb''b''иb''b''нb''b' ↩
    'yb'' b''дb''b''ob'' X = #1 , Y = #2 , Z = #3)
G0 X #1 Y #2 Z #3

O<go_to_position> endsub
M2
```

Після натискання кнопки ***виконати макрос*** вам буде запропоновано ввести значення для ***позиція X, позиція Y, позиція Z***, і макрос буде виконано, лише якщо всі значення були введені.

Note
Якщо ви хочете використовувати макрос без будь-якого руху, див. також примітки в known problems.



Приклад макросу з використанням макросу ”перейти до позиції”

10.2.4.4 Вбудовані вкладки та панелі

Ви можете додавати вбудовані програми до GMOCCAPY, як це можна робити в AXIS, Touchy та Gscreen. GMOCCAPY робить все автоматично, якщо ви додасте кілька рядків до вашого INI-файлу в розділі DISPLAY.

Якщо ви ніколи не користувалися панеллю Glade, рекомендую ознайомитися з чудовою документацією за адресою Glade VCP.

Приклад вбудованої 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, про що вам потрібно подбати, це про те, щоб ви включили для кожної вкладки або бічної панелі згадані три рядки:

- EMBED_TAB_NAME = Представляє назву вкладки або бічної панелі, вам вирішувати, яку назву використовувати, але вона має бути присутня!

- EMBED_TAB_LOCATION = Місце, де буде розміщено вашу програму в графічному інтерфейсі, див. рисунок Розташування вбудованих вкладок. Дійсні значення:
  - **ntb_user_tabs** (як головна вкладка, що охоплює весь екран **(7)**)
  - **ntb_preview** (як вкладка на стороні попереднього перегляду **(1)**)
  - **hbox_jog** (приховає кнопки пробіжок та розмістить ваш файл Glade тут **(2)**)
  - **box_left** (ліворуч, повна висота екрана **(10)**)
  - **box_right** (праворуч, між звичайним екраном та списком кнопок **(11)**)
  - **box_tool_and_code_info** (приховає інформацію про інструмент та кадри G-коду, а ваш файл glade буде розміщено тут **(3)**)
  - **box_tool_info** (приховає рамку інформації про інструмент і розмістить тут ваш файл glade **(3a)**)
  - **box_code_info** (приховає інформаційний кадр G-коду та розмістить тут ваш файл glade **(3b)**)
  - **box_vel_info** (приховає кадри швидкості та введе ваш файл Glade **(4)**)
  - **box_coolant_and_spindle** (приховає рамки охолоджувальної системи та шпинделя, а ваш файл Glade буде розміщено тут **(5)+(6)**)
  - **box_cooling** (приховає рамку охолодження та введе ваш файл Glade **(5)**)
  - **box_spindle** (приховає раму шпинделя та введе ваш напилок Glade **(6)**)
  - **box_custom_1** (представить ваш файл glade ліворуч від vel_frame)
  - **box_custom_2** (представить ваш файл glade ліворуч від cooling_frame)
  - **box_custom_3** (представить ваш файл glade ліворуч від spindle_frame)
  - **box_custom_4** (опублікує ваш файл glade праворуч від spindle_frame)
  - **box_dro_side** (опублікую ваш файл Glade одразу від DRO **(8)**)
  - **ntb_setup** (як вкладка на сторінці налаштувань **(9)**)

Note
Дивіться також додані зразки INI-файлів, щоб побачити відмінності.

- EMBED_TAB_COMMAND = Команда для виконання, тобто.

```
gladevcp -x {XID} dro.glade
```

містить користувацький файл glade під назвою dro.glade у зазначеному місці. Файл має бути поміщений у папку конфігурації вашого комп’ютера.

```
gladevcp h_buttonlist.glade
```

просто відкриється нове вікно користувача під назвою h_buttonlist.glade, зверніть увагу на різницю. Це окреме вікно, яке можна переміщувати незалежно від вікна GMOCCAPY.

```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

додасть панель manual-example.ui, включить власний обробник Python hitcounter.py та встановить усі з’єднання після реалізації панелі згідно з manual-example.hal.

```
hide
```

приховає вибране поле.

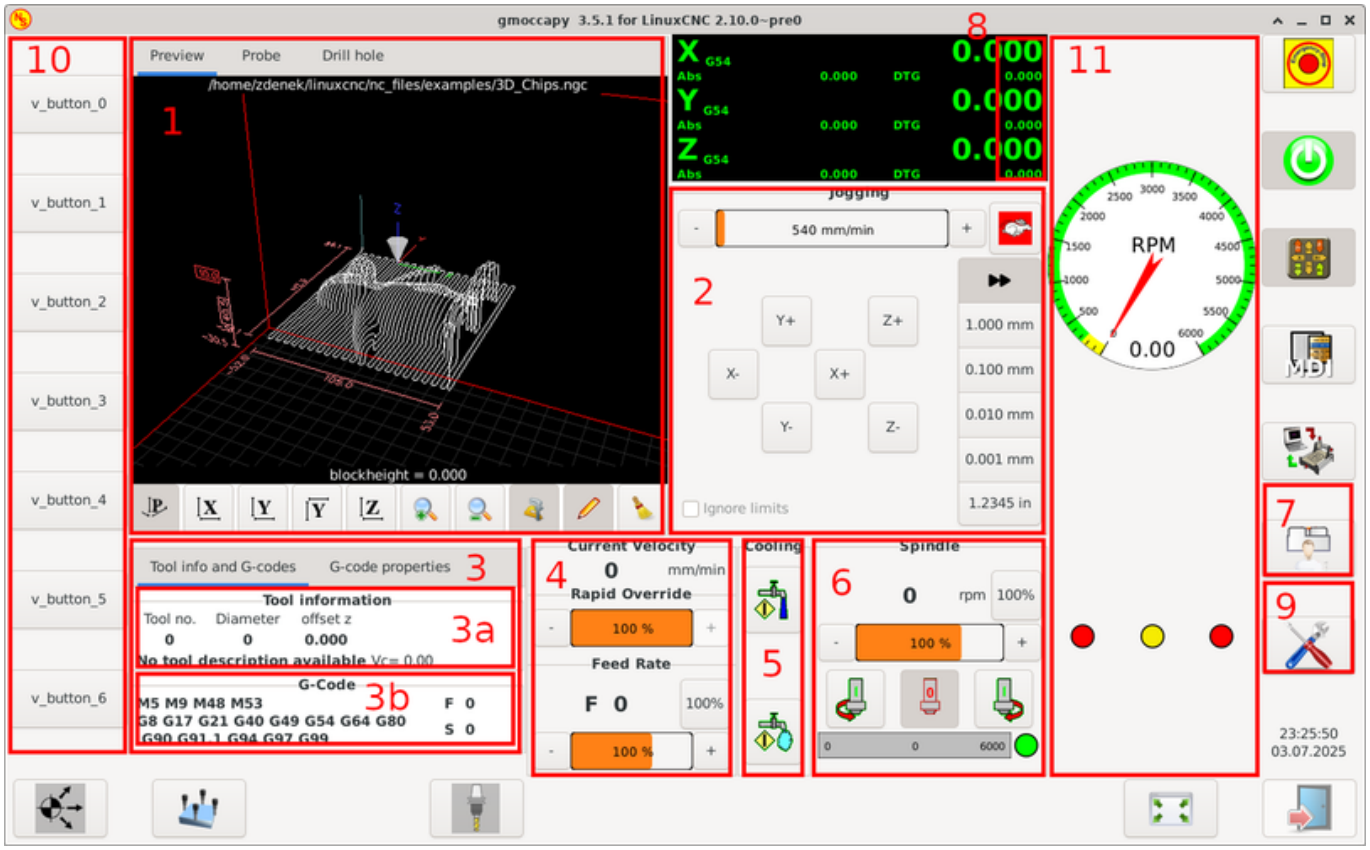


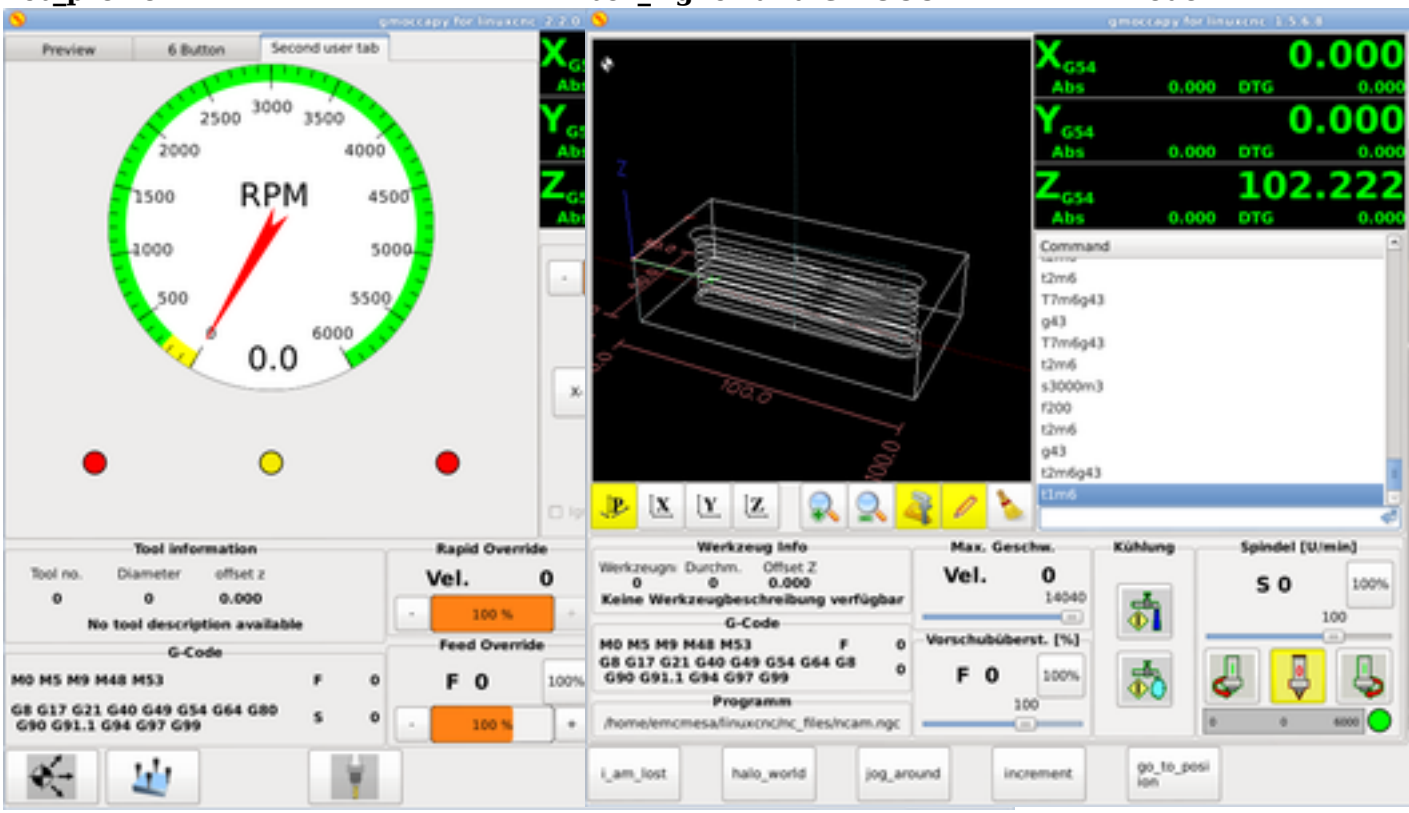
Figure 10.14: Розташування вбудованих вкладок

Note

Якщо ви створюєте будь-які HAL-з’єднання з вашою власною панеллю Glade, вам потрібно зробити це у файлі HAL, вказаному в рядку EMBED_TAB_COMMAND, інакше ви можете отримати помилку про те, що HAL-контакт не існує — це відбувається через умови гонки при завантаженні файлів HAL. З’єднання з контактами GMOCCAPY HAL потрібно встановлювати у файлі postgui HAL, вказаному у вашому файлі INI, оскільки ці контакти не існують до реалізації графічного інтерфейсу користувача.

Ось кілька прикладів:

ntb_preview **box_right - and GMOCCAPY in MDI mode**



10.2.4.5 Повідомлення, створені користувачем

GMOCCAPY має можливість створювати повідомлення користувача, керовані HAL. Щоб використовува їх, потрібно додати кілька рядків у розділ [DISPLAY] INI-файлу.

Ці три рядки потрібні для визначення спливаючого діалогового вікна повідомлення користувача:

```
MESSAGE_TEXT    = b''Tb''b''eb''b''кb''b''cb''b''тb'', b''яb''b''кb''b''иb''b''йb'' b''бb'' ↩
    b''уb''b''дb''b''eb'' b''вb''b''ib''b''дb''b''ob''b''бb''b''pb''b''ab''b''жb''b''eb''b' ↩
    'нb''b''ob'', b''мb''b''ob''b''жb''b''eb'' b''бb''b''уb''b''тb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''фb''b''ob''b''pb''b''мb''b''ab''b''тb''b''ob''b''вb''b''ab''b''нb''b''иb''b' ↩
    'йb'' b''зb''b''ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' b' ↩
    'pb''b''ob''b''зb''b''мb''b''ib''b''тb''b''кb''b''иb'' pango
MESSAGE_TYPE    = "status" , "okdialog" , "yesnodialog"
MESSAGE_PINNAME = b''нb''b''ab''b''зb''b''вb''b''ab'' b''гb''b''pb''b''уb''b''пb''b''иb'' b ↩
    ''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''ib''b''вb'' HAL, b''яb''b''кb''b' ↩
    'уb'' b''пb''b''ob''b''тb''b''pb''b''ib''b''бb''b''нb''b''ob'' b''cb''b''тb''b''вb''b' ↩
    'ob''b''pb''b''иb''b''тb''b''иb''
```

Повідомлення підтримують мову розмітки pango. Детальну інформацію про мову розмітки можна знайти за адресою Розмітка Pango.

Доступні такі три типи діалогових вікон:

- **status** - Просто відобразить повідомлення у спливаючому вікні, використовуючи систему обміну повідомленнями GMOCCAPY.
- **okdialog** - Фокус буде утримуватися на діалоговому вікні повідомлення та активуватиметься HAL-пін `-waiting`.
- **yesnodialog** - Утримуватиме фокус на діалоговому вікні повідомлення та активуватиме HAL-пін `-waiting` і надасть HAL-пін `-response`.

Для отримання детальнішої інформації про піни див. Піни HAL для повідомлень.

Приклад конфігурації повідомлення користувача

```
MESSAGE_TEXT = This is a <span background="#ff0000" foreground="#ffffff">info-message</span ↩
    > test
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

MESSAGE_TEXT = b''Цb''b''eb'' b''тb''b''eb''b''cb''b''тb'' b''дb''b''ib''b''ab''b''лb''b' ↩
    'ob''b''гb''b''yb'' «b''тb''b''ab''b''кb''/b''нb''b''ib''»
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog

MESSAGE_TEXT = b''Tb''b''eb''b''кb''b''cb''b''тb'' b''мb''b''ob''b''жb''b''eb'' b''бb''b' ↩
    'yb''b''тb''b''иb'' <small>b''мb''b''ab''b''лb''b''иb''b''мb''</small>, <big>b''вb''b' ↩
    'eb''b''лb''b''иb''b''кb''b''иb''b''мb''</big>, <b>b''жb''b''иb''b''pb''b''нb''b''иb''b' ↩
    'мb''</b <i>b''кb''b''yb''b''pb''b''cb''b''иb''b''вb''b''ob''b''мb''</i> b''ib'' b''нb'' ↩
    b''ab''b''вb''b''ib''b''тb''b''ьb'' <span color="red">b''чb''b''eb''b''pb''b''вb''b' ↩
    'ob''b''нb''b''иb''b''мb''</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog
```

Note
Наразі форматування не працює.

10.2.4.6 Контроль попереднього перегляду

Магічні коментарі можуть використовуватися для управління попереднім переглядом G-коду. У дуже великих програмах попередній перегляд може завантажуватися досить довго. Ви можете контролювати, що відображається, а що приховується на графічному екрані, додаючи відповідні коментарі з цього списку до вашого G-коду:

```
(PREVIEW,hide)
<G-b''кb''b''ob''b''дb'', b''яb''b''кb''b''иb''b''йb'' b''бb''b''yb''b''дb''b''eb'' b''пb'' ↩
    b''pb''b''иb''b''xb''b''ob''b''вb''b''ab''b''нb''b''ob''>
(PREVIEW,show)
```

10.2.4.7 Файл команд користувача

Якщо файл `~/.gmoccapyrc` існує, його вміст виконується як вихідний код Python одразу після відображення графічного інтерфейсу користувача. Деталі того, що може бути записано у файлі `~/.gmoccapyrc`, можуть змінюватися протягом циклу розробки.

Специфічний для конфігурації файл Python може бути вказаний за допомогою параметра INI-файлу

```
[DISPLAY]
USER_COMMAND_FILE=filename.py
```

Якщо цей файл вказано, його джерело буде отримано одразу після відображення графічного інтерфейсу GMOCCAPY **замість** `~/.gmoccapyrc`.

У наступному прикладі змінюється розмір вертикальних кнопок: .Приклад файлу .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)
```

Назви віджетів можна знайти у файлі /usr/share/gmoccapy.glade

10.2.4.8 CSS-файл користувача

Подібно до файлу команд користувача, можна впливати на зовнішній вигляд за допомогою каскадних таблиць стилів (CSS). Якщо існує файл `~/.gmoccapy_css`, його вміст завантажується в постачальник таблиць стилів і таким чином застосовується до графічного інтерфейсу користувача.

CSS-файл, що відповідає конфігурації, може бути вказаний за допомогою параметра INI-файлу

```
[DISPLAY]
USER_CSS_FILE=filename.css
```

Якщо цей файл вказано, він використовується **замість** `~/.gmoccapy_css`.

Інформацію про те, що можна контролювати за допомогою CSS, можна знайти тут: посилання: https://docs.gtk.org/gtk3/css-overview.html [Огляд CSS у GTK]

Ось приклад того, як можна встановити жовтий колір для позначених кнопок: .Приклад жовтого кольору для позначених кнопок

```
button:checked {
    background: rgba(250,230,0,0.8);
}
```

10.2.4.9 Лісозаготівля

GMOCCAPY підтримує визначення рівня інформації (рівня журналу), який буде виведено на консоль та до файлу журналу.

Порядок такий: *VERBOSE*, *DEBUG*, *INFO*, *WARNING*, *ERROR*, *CRITICAL*. За замовчуванням — *WARNING*, це означає, що друкуються *WARNING*, *ERROR* та *CRITICAL*.

Ви можете вказати рівень журналу в INI-файлі таким чином:

```
[DISPLAY]
DISPLAY = gmoccapy <log_level_param>
```

використовуючи ці параметри:

```
b''Pb''b''ib''b''вb''b''eb''b''нb''b''ьb'' b''жb''b''yb''b''pb''b''нb''b''ab''b''лb''b' ↩
    'yb''   <log_level_param>
DEBUG       -d
INFO        -i
VERBOSE     -v
ERROR       -q
```

Приклад: Налаштування журналювання для виведення лише помилок

```
[DISPLAY]
DISPLAY = gmoccapy -q
```

Ви можете вказати, де зберігати файл журналу:

```
[DISPLAY]
LOG_FILE = gmoccapy.log
```

Якщо LOG_FILE не встановлено, логування відбувається у $HOME/<base_log_name>.log.

10.2.5 Піни HAL

GMOCCAPY експортує кілька виводів HAL, щоб мати змогу реагувати на апаратні пристрої. Мета полягає в тому, щоб отримати графічний інтерфейс, яким можна користуватися в інструментальній майстерні, повністю/переважно без миші чи клавіатури.

Note
Ви повинні виконати всі підключення до контактів GMOCCAPY у вашому файлі postgui.hal. Коли GMOCCAPY запускається, він створює контакти HAL для графічного інтерфейсу користувача, а потім виконує файл HAL після графічного інтерфейсу користувача, названий у файлі INI:

```
[HAL]
POSTGUI_HALFILE=<filename>
```

Зазвичай <filename> буде базовим ім’ям конфігурації + _postgui.hal, наприклад lathe_postgui.hal, але може бути будь-яким допустимим іменем файлу.
Ці команди виконуються після побудови екрану, гарантуючи доступність контактів HAL віджета.
У файлі INI можна мати кілька рядків POSTGUI_HALFILE=<filename>. Кожен з них буде виконуватися по черзі в порядку їх появи.

10.2.5.1 Списки правої та нижньої кнопок

Екран має два основних списки кнопок: один праворуч, інший знизу. Праві кнопки не змінюються під час роботи, але список кнопок знизу змінюється дуже часто. Кнопки нумеруються зверху вниз і зліва направо, починаючи з 0.

Note
Назви контактів у GMOCCAPY 2 змінилися для кращого їхнього розташування.

Контакти для правих (вертикальних) 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 нижніх (горизонтальних) кнопок це:

- **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)*

Оскільки кнопки в нижньому списку змінюються залежно від режиму та інших факторів, апаратні кнопки активують відображені функції. Тому вам не потрібно турбуватися про перемикання функцій в HAL, оскільки це повністю виконує GMOCCAPY!

Для тривісного верстата XYZ контакти HAL реагуватимуть, як показано в наступних трьох таблицях:

Table 10.3: Функціональне призначення горизонтальних кнопок (1)

Закріпити	**Ручний режим**	**Режим MDI**	**Автоматичний режим**
gmoccapy.h-button.button-0	кнопка відкритого хомінгу	macro 1 (якщо визначено)	відкрити файл
gmoccapy.h-button.button-1	відкритий контактний матеріал	macro 2 (якщо визначено)	програма перезавантаження
gmoccapy.h-button.button-2		macro 3 (якщо визначено)	запуск
gmoccapy.h-button.button-3	відкрити діалогові вікна інструментів	macro 4 (якщо визначено)	стій
gmoccapy.h-button.button-4		macro 5 (якщо визначено)	пауза
gmoccapy.h-button.button-5		macro 6 (якщо визначено)	крок за кроком
gmoccapy.h-button.button-6		macro 7 (якщо визначено)	запускати з рядка, якщо це ввімкнено в налаштуваннях, інакше нічого
gmoccapy.h-button.button-7		macro 8 (якщо визначено)	додаткові блоки
gmoccapy.h-button.button-8	повнорозмірний попередній перегляд	макрос 9 або кнопка для відображення додаткових макросів	повнорозмірний попередній перегляд
gmoccapy.h-button.button-9	вихід якщо машина вимкнена, інакше нічого	відкрити клавіатуру або перервати, якщо макрос запущено	редагувати код

Table 10.4: Функціональне призначення горизонтальних кнопок (2)

Закріпити	**Режим налаштувань**	**Режим самонаведення**	**Режим вимкнення сенсорного керування**
gmoccapy.h-button.button-0	відкрити історію MDI		редагувати зміщення
gmoccapy.h-button.button-1	всі будинки	всі будинки	дотик X
gmoccapy.h-button.button-2			дотик Y
gmoccapy.h-button.button-3		головна сторінка x	дотик Z
gmoccapy.h-button.button-4	відкриті класичні сходи	головна сторінка y	
gmoccapy.h-button.button-5	відкрита область HAL	головна сторінка z	
gmoccapy.h-button.button-6	відкритий стан HAL		нуль G92
gmoccapy.h-button.button-7	відкритий HAL-лічильник		
gmoccapy.h-button.button-8	відкрите калібрування HAL	зняти з дому все	встановити вибране
gmoccapy.h-button.button-9	відкрите шоу HAL	назад	назад

Table 10.5: Функціональне призначення горизонтальних кнопок (3)

Закріпити	**Режим інструменту**	**Режим редагування**	**Виберіть файл**
gmoccapy.h-button.button-0	інструмент(и) видалення		перейти до домашнього каталогу
gmoccapy.h-button.button-1	новий інструмент	перезавантажити файл	на один рівень каталогу вище
gmoccapy.h-button.button-2	перезавантажити таблицю інструментів	зберегти	
gmoccapy.h-button.button-3	застосувати зміни	зберегти як	перемістити виділення ліворуч
gmoccapy.h-button.button-4	змінити інструменту за номером T? M6		перемістити виділення праворуч
gmoccapy.h-button.button-5	встановити інструмент за номером без змін M61 Q?		перейти до каталогу, як встановлено в налаштуваннях
gmoccapy.h-button.button-6	змінити інструмент на вибраний	новий файл	
gmoccapy.h-button.button-7			вибрати / ENTER
gmoccapy.h-button.button-8	доторкніться інструменту по Z	показати клавіатуру	

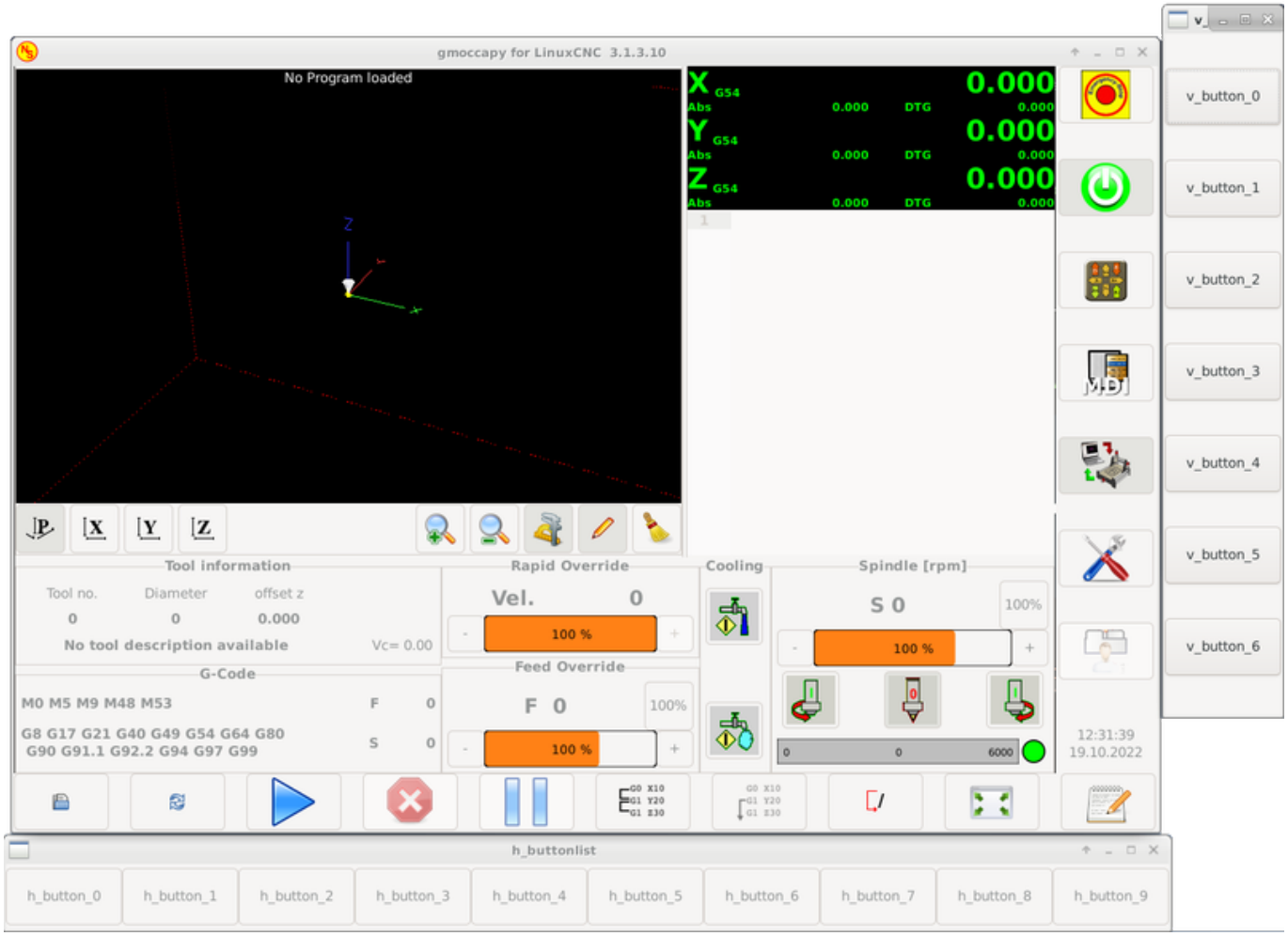
Table 10.5: (continued)

Закріпити	Режим інструменту	Режим редагування	Виберіть файл
gmoccapy.h-button.button-9	назад	назад	назад

Отже, у нас є 67 реакцій лише з 10 HAL-пінами!

Ці контакти доступні для використання екрану без сенсорної панелі або для захисту його від надмірного використання шляхом розміщення апаратних кнопок навколо панелі. Вони доступні в зразковій конфігурації, як показано на зображенні image below.

Приклад конфігурації "gmoccapy_sim_hardware_button" із зображенням бічних кнопок



10.2.5.2 Швидкості та корекції

Всі повзунки від GMOCCAPY можна підключити до апаратних енкодерів або апаратних потенціометрів.

Note
У GMOCCAPY 3 деякі назви контактів HAL змінилися після впровадження нових елементів керування. Максимальної швидкості більше не існує, її замінили швидким перевизначенням через вимоги багатьох користувачів.

Для підключення енкодерів експортуються такі контакти:

- **gmoccapy.jog.jog-velocity.counts** *(s32 IN)* - Швидкість штовхання
- **gmoccapy.jog.jog-velocity.count-enable** *(bit IN)* - Має бути значення True, щоб увімкнути підрахунок
- **gmoccapy.feed.feed-override.counts** *(s32 IN)* - перевизначення подачі
- **gmoccapy.feed.feed-override.count-enable** *(bit IN)* - Має бути значення True, щоб увімкнути підрахунок
- **gmoccapy.feed.reset-feed-override** *(bit IN)* - скинути перевизначення подачі до 0%
- **gmoccapy.spindle.spindle-override.counts** *(s32 IN)* - корекція шпинделя
- **gmoccapy.spindle.spindle-override.count-enable** *(bit IN)* - Має бути значення True, щоб увімкнути підрахунок
- **gmoccapy.spindle.reset-spindle-override** *(bit IN)* - скинути корекцію шпинделя до 0%
- **gmoccapy.rapid.rapid-override.counts** *(s32 IN)* - Максимальна швидкість машини
- **gmoccapy.rapid.rapid-override.count-enable** *(bit IN)* - Має бути значення True, щоб увімкнути підрахунок

Для підключення потенціометрів використовуйте такі контакти:

- **gmoccapy.jog.jog-velocity.direct-value** *(float IN)* - Щоб налаштувати повзунок швидкості джогу
- **gmoccapy.jog.jog-velocity.analog-enable** *(bit IN)* - Має бути значення «True», щоб дозволити аналогові входи
- **gmoccapy.feed.feed-override.direct-value** *(float IN)* - Щоб налаштувати повзунок корекції подачі
- **gmoccapy.feed.feed-override.analog-enable** *(bit IN)* - Має бути значенням True, щоб дозволити аналогові входи
- **gmoccapy.spindle.spindle-override.direct-value** *(float IN)* - Щоб налаштувати повзунок корекції шпинделя
- **gmoccapy.spindle.spindle-override.analog-enable** *(bit IN)* - Має бути значенням True, щоб дозволити аналогові входи
- **gmoccapy.rapid.rapid-override.direct-value** *(float)* - Щоб налаштувати повзунок максимальної швидкості
- **gmoccapy.rapid.rapid-override.analog-enable** *(bit IN)* - Має бути значення «True», щоб дозволити аналогові входи

Крім того, GMOCCAPY 3 пропонує додаткові контакти HAL для керування новими слайдер-віджетами за допомогою миттєвих перемикачів. Значення швидкості збільшення або зменшення необхідно встановити у файлі glade. У майбутній версії це буде інтегровано на сторінці налаштувань.

ШВИДКІСТЬ

- **gmoccapy.spc_jog_vel.increase** *(bit IN)* - Доки значення True, значення повзунка збільшуватиметься
- **gmoccapy.spc_jog_vel.decrease** *(bit IN)* - Доки значення True, значення повзунка зменшуватиметься
- **gmoccapy.spc_jog_vel.scale** *(float IN)* - Значення для масштабування вихідного значення (зручно для зміни одиниць/хв на одиниці/с)
- **gmoccapy.spc_jog_vel.value** *(float OUT)* - Значення віджета

- **gmoccapy.spc_jog_vel.scaled-value** *(float OUT)* - Масштабоване значення віджета

КОРМИ

- **gmoccapy.spc_feed.increase** *(bit IN)* - Доки значення True, значення повзунка збільшуватиметься
- **gmoccapy.spc_feed.decrease** *(bit IN)* - Доки значення True, значення повзунка зменшуватиметься
- **gmoccapy.spc_feed.scale** *(float IN)* - Значення для масштабування вихідного значення (зручно для зміни одиниць/хв на одиниці/с)
- **gmoccapy.spc_feed.value** *(float OUT)* - Значення віджета
- **gmoccapy.spc_feed.scaled-value** *(float OUT)* - Масштабоване значення віджета

SPINDLE

- **gmoccapy.spc_spindle.increase** *(bit IN)* - Доки значення True, значення повзунка збільшуватиметьс
- **gmoccapy.spc_spindle.decrease** *(bit IN)* - Доки значення True, значення повзунка зменшуватиметьс
- **gmoccapy.spc_spindle.scale** *(float IN)* - Значення для масштабування вихідного значення (зручно для зміни одиниць/хв на одиниці/с)
- **gmoccapy.spc_spindle.value** *(float OUT)* - Значення віджета
- **gmoccapy.spc_spindle.scaled-value** *(float OUT)* - Масштабоване значення віджета

ПОРОГИ

- **gmoccapy.spc_rapid.increase** *(bit IN)* - Доки значення True, значення повзунка збільшуватиметься
- **gmoccapy.spc_rapid.decrease** *(bit IN)* - Доки значення True, значення повзунка зменшуватиметься
- **gmoccapy.spc_rapid.scale** *(float IN)* - Значення для масштабування вихідного значення (зручно для зміни одиниць/хв на одиниці/с)
- **gmoccapy.spc_rapid.value** *(float OUT)* - Значення віджета
- **gmoccapy.spc_rapid.scaled-value** *(float OUT)* - Масштабоване значення віджета

Піни з плаваючою точкою приймають значення від 0,0 до 1,0, що є відсотковим значенням, яке потрібно встановити для значення повзунка.



Warning
Якщо ви використовуєте обидва типи з'єднання, не підключайте один і той же повзунок до обох контактів, оскільки взаємодія між ними не була перевірена! Різні повзунки можуть бути підключені до одного або іншого типу з'єднання HAL.



Important
Зверніть увагу, що швидкість переміщення залежить від стану кнопки черепахи. Це призведе до різних шкал повзунка залежно від режиму (черепаха або кролик). Для отримання додаткової інформації див. також jog velocities and turtle-jog HAL pin.

Example 10.1 Встановлення значення повзунка

Мінімальне значення перевищення шпинделя = 20 %
Максимальне значення перевищення шпинделя = 120 %
gmoccapy.analog-enable = 1
gmoccapy.spindle-override-value = 0,25

значення, яке потрібно встановити = мінімальне значення + (максимальне значення - мінімальне значення) * gmoccapy.spindle-override-value
значення, яке потрібно встановити = 20 + (120 - 20) * 0,25
значення, яке потрібно встановити = 45 %

10.2.5.3 Jog HAL Pins

Усі осі, зазначені в INI-файлі, мають контакти jog-plus та jog-minus, тому для штовхання осей можна використовувати апаратні миттєві перемикачі.

Note
Назви цих HAL-пінів змінилися в GMOCCAPY 2.

Для стандартної конфігурації XYZ будуть доступні такі HAL-піни:

- **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)*

Якщо ви використовуєте конфігурацію з 4 осями, будуть два додаткові контакти:

- **gmoccapy.jog.jog-<your fourth axis letter >-plus** *(bit IN)*
- **gmoccapy.jog.jog-<your fourth axis letter >-minus** *(bit IN)*

Для осі C ви побачите:

- **gmoccapy.jog.axis.jog-c-plus** *(bit IN)*
- **gmoccapy.jog.axis.jog-c-minus** *(bit IN)*

10.2.5.4 Швидкості поштовху та штифт HAL Turtle-Jog

Швидкість переміщення можна вибрати за допомогою відповідного повзунка. Шкала повзунка буде змінена, якщо кнопка черепахи (та, що показує кролика або черепаху) була переключена. Якщо кнопка не видна, вона, можливо, була вимкнена на сторінці settings. Якщо на кнопці зображено кролика, шкала відображає швидкість від мінімальної до максимальної. Якщо на ній зображено черепаху, шкала за замовчуванням досягає лише 1/20 від максимальної швидкості. Використовуваний дільник можна налаштувати на сторінці settings.

Тож за допомогою сенсорного екрана набагато легше вибирати менші швидкості.

GMOCCAPY пропонує цей HAL-штифт для перемикання між бігом у формі черепахи та кролика:

- **gmoccapy.jog.turtle-jog** *(bit IN)*

10.2.5.5 Штифти HAL для приросту поштовху

Приріст поштовху, вказаний у INI-файлі, наприклад

```
[DISPLAY]
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm
```

можна вибрати за допомогою контактів HAL, тому для вибору кроку можна використовувати апаратний перемикач. Для кроків, зазначених у файлі INI, буде доступно максимум 10 контактів HAL. Якщо у файлі INI вказано більше кроків, вони не будуть доступні з графічного інтерфейсу, оскільки не відображатимуться.

Якщо у вашому INI-файлі є 6 приростів, як у наведеному вище прикладі, ви отримаєте **7** контактів:

- **gmoccapy.jog.jog-inc-0** *(bit IN)* - Цей фіксований і представлятиме безперервний біг підтюпцем.
- **gmoccapy.jog.jog-inc-1** *(bit IN)* - Перший приріст, заданий у INI-файлі.
- **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 також пропонує вивід HAL для виведення вибраного приросту поштовхової передачі:

- **gmoccapy.jog.jog-increment** *(float OUT)*

10.2.5.6 PIN-код для розблокування апаратного забезпечення

Щоб мати змогу використовувати ключовий перемикач для розблокування сторінки налаштувань, експортується наступний PIN-код:

- **gmoccapy.unlock-settings** *(bit IN)* - Сторінка налаштувань розблоковується, якщо PIN-код має високий рівень. Щоб використовувати цей PIN-код, потрібно активувати його на сторінці налаштувань.

10.2.5.7 Піни помилок/попереджень

- **gmoccapy.error** *(bit OUT)* - Вказує на помилку, тому може загорітися індикатор, або навіть машина може бути зупинена. Скидання налаштувань буде здійснено за допомогою PIN-коду `gmoccapy.delete-message`.
- **gmoccapy.delete-message** *(bit IN)* - Видалить першу помилку та скине значення `gmoccapy.error` у значення false після усунення останньої помилки.
- **gmoccapy.warning-confirm** *(bit IN)* - Підтверджує діалогове вікно попередження, наприклад, натискання кнопки «OK»

Note
Повідомлення або інформація про користувача не вплинуть на пін-код `gmoccapy.error`, але пін-код `gmoccapy.delete-message` видалить останнє повідомлення, якщо помилка не відображається!

10.2.5.8 Піни HAL, створені користувачем, для повідомлень

GMOCCAPY можна налаштувати для реагування на зовнішні помилки, використовуючи 3 різні повідомлення користувача:

статус

- **gmoccapy.messages.status** *(bit IN)* - Запускає діалог.

окдіалог

- **gmoccapy.messages.okdialog** *(bit IN)* - Запускає діалог.
- **gmoccapy.messages.okdialog-waiting** *(bit OUT)* - Буде «1», поки діалогове вікно відкрите. Закриття повідомлення скине цей контакт.

yesnodialog

- **gmoccapy.messages.yesnodialog** *(bit IN)* - Запускає діалог.
- **gmoccapy.messages.yesnodialog-waiting** *(bit OUT)* - Буде «1», поки діалогове вікно відкрите. Закриття повідомлення скине цей контакт.
- **gmoccapy.messages.yesnodialog-response** *(bit OUT)* - Цей пін-код зміниться на «1», якщо користувач натисне «OK», а в усіх інших випадках він буде «0». Цей пін-код залишатиметься «1», доки діалогове вікно не буде викликано знову.

Щоб додати повідомлення, створене користувачем, потрібно додати його до INI-файлу в розділі DISPLAY. Див. Налаштування повідомлень.

Приклад повідомлення користувача (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
```

Щоб підключити ці нові контакти, потрібно зробити це у HAL-файлі postgui. Ось кілька прикладів підключень, які з’єднують сигнали повідомлень з деяким іншим місцем у HAL-файлі.

Приклад підключення користувацьких повідомлень (файл HAL)

```
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 отримання додаткової інформації про файли HAL та команду net див. Основи HAL.

10.2.5.9 Штифти зворотного зв’язку шпинделя

Є два контакти для зворотного зв’язку шпинделя:

- **gmoccapy.spindle_feedback_bar** *(float IN)* - Закріпіть, щоб показати швидкість шпинделя на шпиндельній планці.
- **gmoccapy.spindle_at_speed_led** *(bit IN)* - Підключіть, щоб увімкнути світлодіод, що показує швидкість.

10.2.5.10 Шпильки для відображення інформації про хід виконання програми

Є три піктограми, які надають інформацію про хід виконання програми:

- **gmoccapy.program.length** *(s32 OUT)* - Показує загальну кількість рядків програми.
- **gmoccapy.program.current-line** *(s32 OUT)* - Вказує поточний робочий рядок програми.
- **gmoccapy.program.progress** *(float OUT)* - Показує прогрес програми у відсотках.

Значення можуть бути не дуже точними, якщо ви працюєте з підпрограмами або великими процедурами перепризначення. Також цикли призводитимуть до різних значень.

10.2.5.11 Штифти, пов’язані з інструментами

Штифти для зміни інструменту Ці контакти призначені для використання внутрішнього діалогового вікна зміни інструменту GMOCCAPY, схожого на те, що відоме з AXIS, але з декількома модифікаціями. Таким чином, ви отримаєте не тільки повідомлення про зміну на «інструмент № 3», але й опис цього інструменту, наприклад «7,5 мм 3-різцевий фрезер». Інформація береться з таблиці інструментів тому ви самі вирішуєте, що відображати.

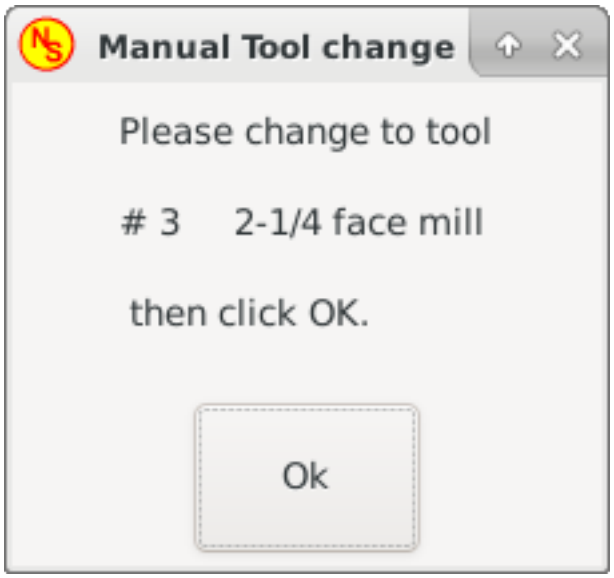


Figure 10.15: Діалогове вікно зміни інструменту GMOCCAPY

- **gmoccapy.toolchange-number** *(s32 IN)* - Номер інструменту, який потрібно замінити
- **gmoccapy.toolchange-change** *(bit IN)* - Означає, що інструмент потрібно замінити
- **gmoccapy.toolchange-changed** *(bit OUT)* - Вказує на зміну інструменту
- **gmoccapy.toolchange-confirm** *(bit IN)* - Підтверджує зміну інструменту

Зазвичай для ручної зміни інструменту їх з’єднують 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
Будь ласка, зверніть увагу, що ці з'єднання мають бути виконані у HAL-файлі postgui.

Штифти зміщення інструменту Ці контакти дозволяють відображати значення активного зміщення інструменту для X та Z у фреймі інформації про інструмент. Слід знати, що вони активні лише після надсилання G43.



Figure 10.16: Область інформації про інструмент

- **gmoccapy.tooloffset-x** *(float IN)*
- **gmoccapy.tooloffset-z** *(float IN)*

Note
Рядок tooloffset-x не потрібен на фрезерному верстаті та не відображатиметься на фрезерному верстаті з тривіальною кінематикою.

Щоб відобразити поточні зміщення, піни повинні бути з'єднані ось так у HAL-файлі postgui:

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

Important
Зверніть увагу, що GMOCCAPY самостійно оновлює компенсації, надсилаючи G43 після кожної зміни інструменту, **але не в автоматичному режимі!**
Тому під час написання програми ви повинні включити G43 після кожної зміни інструменту!

10.2.6 Автоматичне вимірювання інструментів

GMOCCAPY пропонує інтегрований інструмент для вимірювання автомобілів. Щоб скористатися цією функцією, вам потрібно буде виконати деякі додаткові налаштування, і, можливо, ви захочете використовувати запропонований контакт HAL, щоб отримати значення у власній процедурі перерозподілу NGC.

Important
Перед початком першого тесту не забудьте ввести висоту зонда та швидкості зонда на сторінці налаштувань! Див. Сторінка налаштувань Вимірювання інструмента.

Також було б гарною ідеєю переглянути відео про вимірювання інструментів, див. tool measurement related videos.

Вимірювання інструментів у GMOCCAPY виконується дещо інакше, ніж у багатьох інших графічних інтерфейсах. Вам слід виконати такі кроки:

1. Прикладіть до заготовки контакти по осях X та Y.
2. Виміряйте висоту вашого блоку від основи, де розташований перемикач інструментів, до верхньої поверхні блоку (включаючи патрон тощо).
3. Натисніть кнопку висоти блоку та введіть виміряне значення.
4. Перейдіть в автоматичний режим і запустіть програму.

Ось невеликий ескіз:

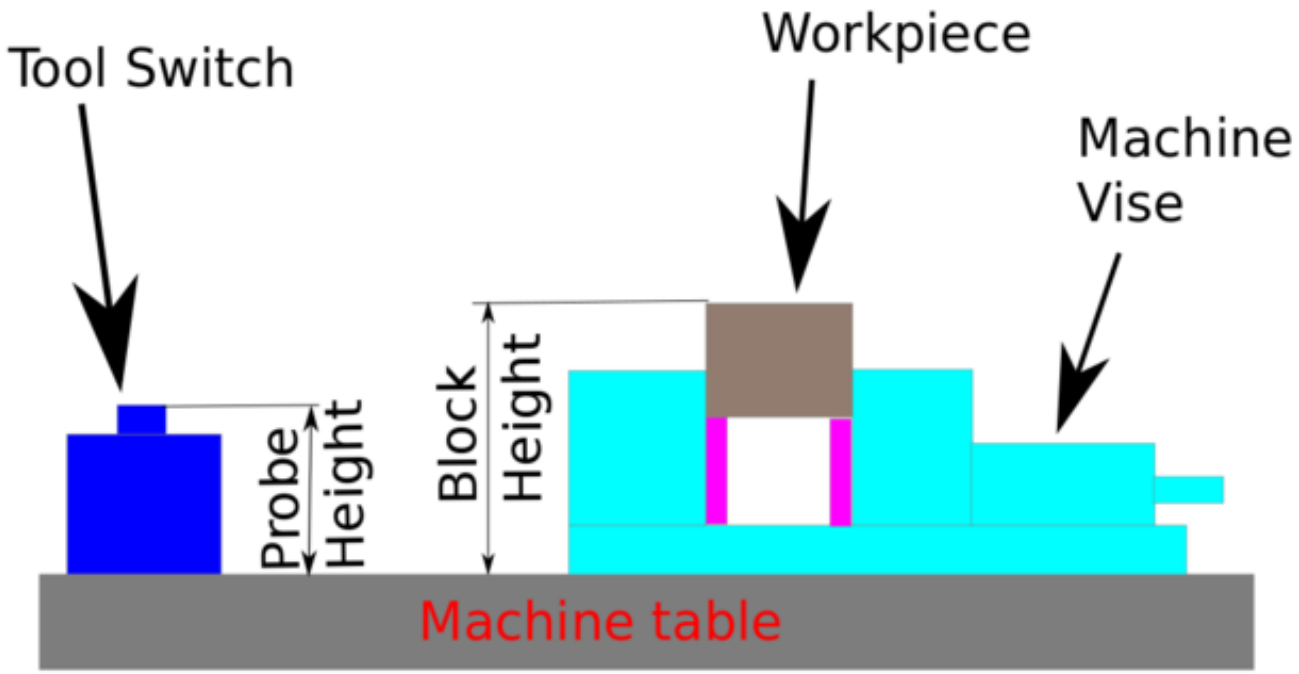


Figure 10.17: Дані вимірювання інструменту

При першій заміні інструменту інструмент буде виміряний, а зміщення буде встановлено автоматично відповідно до висоти блоку. Перевага методу GMOCCAPY полягає в тому, що вам не потрібен еталонний інструмент.

Note
Ваша програма повинна містити зміну інструменту на початку! Інструмент буде виміряний, навіть якщо він вже використовувався раніше, тому немає небезпеки, якщо висота блоку змінилася. На YouTube є кілька відео, що показують, як це зробити.

10.2.6.1 Надані піни

GMOCCAPY пропонує п’ять виводів для вимірювання інструментів. Ці виводи здебільшого використову для зчитування з підпрограми G-коду, тому код може реагувати на різні значення.

- **gmoccapy.toolmeasurement** *(bit OUT)* - Увімкнути або вимкнути вимірювання інструменту
- **gmoccapy.blockheight** *(float OUT)* - Виміряне значення верхньої грані заготовки
- **gmoccapy.probeheight** *(float OUT)* - Висота перемикача зонда
- **gmoccapy.searchvel** *(float OUT)* - Швидкість пошуку перемикача зонда інструменту
- **gmoccapy.probevel** *(float OUT)* - Швидкість до довжини зонду інструменту

10.2.6.2 Модифікації INI-файлу

Змініть свій INI-файл, включивши до нього наступні розділи.

Розділ RS274NGC

```
[RS274NGC]
# b''цb''b''eb'' b''пb''b''ib''b''дb''b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''яb'', b' ↩
    'яb''b''кb''b''ab'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''тb''b' ↩
    'ьb''b''сb''b''яb'', b''кb''b''оb''b''лb''b''иb'' b''вb''b''иb''b''нb''b''иb''b''кb''b' ↩
    'ab''b''єb'' b''пb''b''оb''b''мb''b''иb''b''лb''b''кb''b''ab'' b''пb''b''ib''b''дb'' b' ↩
    'чb''b''ab''b''сb'' b''зb''b''мb''b''ib''b''нb''b''иb'' b''ib''b''нb''b''сb''b''тb''b' ↩
    'pb''b''уb''b''мb''b''eb''b''нb''b''тb''b''уb'', b''нb''b''eb'' b''пb''b''оb''b''тb''b' ↩
    'pb''b''ib''b''бb''b''нb''b''ab'' b''дb''b''лb''b''яb'' b''кb''b''оb''b''жb''b''нb''b' ↩
    'оb''b''їb'' b''кb''b''оb''b''нb''b''фb''b''ib''b''гb''b''уb''b''pb''b''ab''b''цb''b' ↩
    'ib''b''їb'' b''вb''b''eb''b''pb''b''сb''b''тb''b''ab''b''тb''b''ab''
ON_ABORT_COMMAND=O <on_abort> call

# b''Kb''b''оb''b''дb'' b''пb''b''eb''b''pb''b''eb''b''пb''b''pb''b''иb''b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb''
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

Note
Переконайтеся, що INI_VARS та HAL_PIN_VARS не встановлені на 0. За замовчуванням вони встановлені на 1.

Розділ датчика інструменту Положення датчика інструменту та початкове положення зондувальної руху, всі значення є абсолютними координатами, окрім MAXPROBE, яке має бути задане у відносному русі.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

Розділ «Зміна позиції» Це навмисно не називається TOOL_CHANGE_POSITION — **Canon використо цю назву, і в іншому випадку це спричинить конфлікт.** Позиція, в яку потрібно перемістити верстат перед тим, як дати команду на зміну інструменту. Усі значення вказані в абсолютних координатах.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

Розділ Python Плагіни Python обслуговують інтерпретаторів та задач.

```
[PYTHON]
# b''Шb''b''лb''b''яb''b''хb'' b''дb''b''лb''b''яb'' b''пb''b''оb''b''чb''b''аb''b''тb''b' ↩
    'кb''b''уb'' b''пb''b''оb''b''шb''b''уb''b''кb''b''уb'' b''кb''b''оb''b''рb''b''иb''b' ↩
    'сb''b''тb''b''уb''b''вb''b''аb''b''цb''b''ьb''b''кb''b''иb''b''хb'' b''мb''b''оb''b' ↩
    'дb''b''уb''b''лb''b''ib''b''вb''
PATH_PREPEND = python
# b''Bb''b''ib''b''дb''b''пb''b''pb''b''аb''b''вb''b''нb''b''аb'' b''тb''b''оb''b''чb''b' ↩
    'кb''b''аb'' b''дb''b''лb''b''яb'' b''вb''b''cb''b''ib''b''xb''.
TOPLEVEL = python/toplevel.py
```

10.2.6.3 Необхідні файли

Спочатку створіть каталог ”python” у вашій папці config. Зі `<каталог_ваш_лінуксцнц-розробника>/con` скопіюйте наступні файли у щойно створену папку `config_dir/python`:

- `toplevel.py`
- `remap.py`
- `stdglue.py`

З копії `<your_linuxcnc-dev_directory>/configs/sim/gmoccapy/macros`

- `on_abort.ngc`
- `change.ngc`

до каталогу, вказаного як `SUBROUTINE_PATH`, див. RS274NGC.

Відкрийте `change.ngc` за допомогою редактора та розкоментуйте наступні рядки (49 та 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

Можливо, ви захочете змінити цей файл, щоб він більше відповідав вашим потребам.

10.2.6.4 Необхідні HAL-з’єднання

Підключіть зонд інструменту у вашому HAL-файлі таким чином:

```
net probe  motion.probe-input <= <your_input_pin>
```

Рядок може виглядати так:

```
net probe  motion.probe-input <= parport.0.pin-15-in
```

У вашому файлі postgui.hal додайте такі рядки:

```
# b''Hb''b''ab''b''cb''b''тb''b''yb''b''пb''b''нb''b''ib'' b''pb''b''яb''b''дb''b''кb''b' ↩
    'иb'' b''пb''b''ob''b''тb''b''pb''b''ib''b''бb''b''нb''b''ib'' b''лb''b''иb''b''шb''b' ↩
    'eb'' b''вb'' b''тb''b''ob''b''мb''b''yb'' b''вb''b''иb''b''пb''b''ab''b''дb''b''кb''b' ↩
    'yb'', b''яb''b''кb''b''щb''b''ob'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b' ↩
    'иb'' b''бb''b''yb''b''лb''b''иb'' b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b' ↩
    'eb''b''нb''b''ib'' b''pb''b''ab''b''нb''b''ib''b''шb''b''eb''.
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.tool-changed
unlinkp iocontrol.0.tool-prep-number
unlinkp iocontrol.0.tool-prepared

# b''пb''b''ob''b''cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b''зb''b' ↩
    'мb''b''ib''b''нb''b''yb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b' ↩
    'нb''b''тb''b''yb'' GMOCCAPY, b''щb''b''ob'' b''дb''b''ob''b''зb''b''вb''b''ob''b''лb''b ↩
    ''яb''b''єb'' b''пb''b''eb''b''pb''b''eb''b''гb''b''лb''b''яb''b''дb''b''ab''b''тb''b' ↩
    'иb'' b''ob''b''пb''b''иb''b''cb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b' ↩
    'eb''b''нb''b''тb''b''yb'' b''вb'' b''дb''b''ib''b''ab''b''лb''b''ob''b''гb''b''ob''b' ↩
    'вb''b''ob''b''мb''b''yb'' b''вb''b''ib''b''кb''b''нb''b''ib'' b''зb''b''мb''b''ib''b' ↩
    'нb''b''иb''
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 Сторінка налаштувань

Щоб увійти на сторінку, вам потрібно натиснути на зображення:images/gmoccapy_settings_button.png[a
і ввести код розблокування, який за замовчуванням є **123**. Якщо ви хочете змінити його в цей момент, вам потрібно буде відредагувати прихований файл налаштувань, детальніше див. розділ дисплея.

Сторінка розділена на три основні вкладки:

10.2.7.1 Зовнішній вигляд

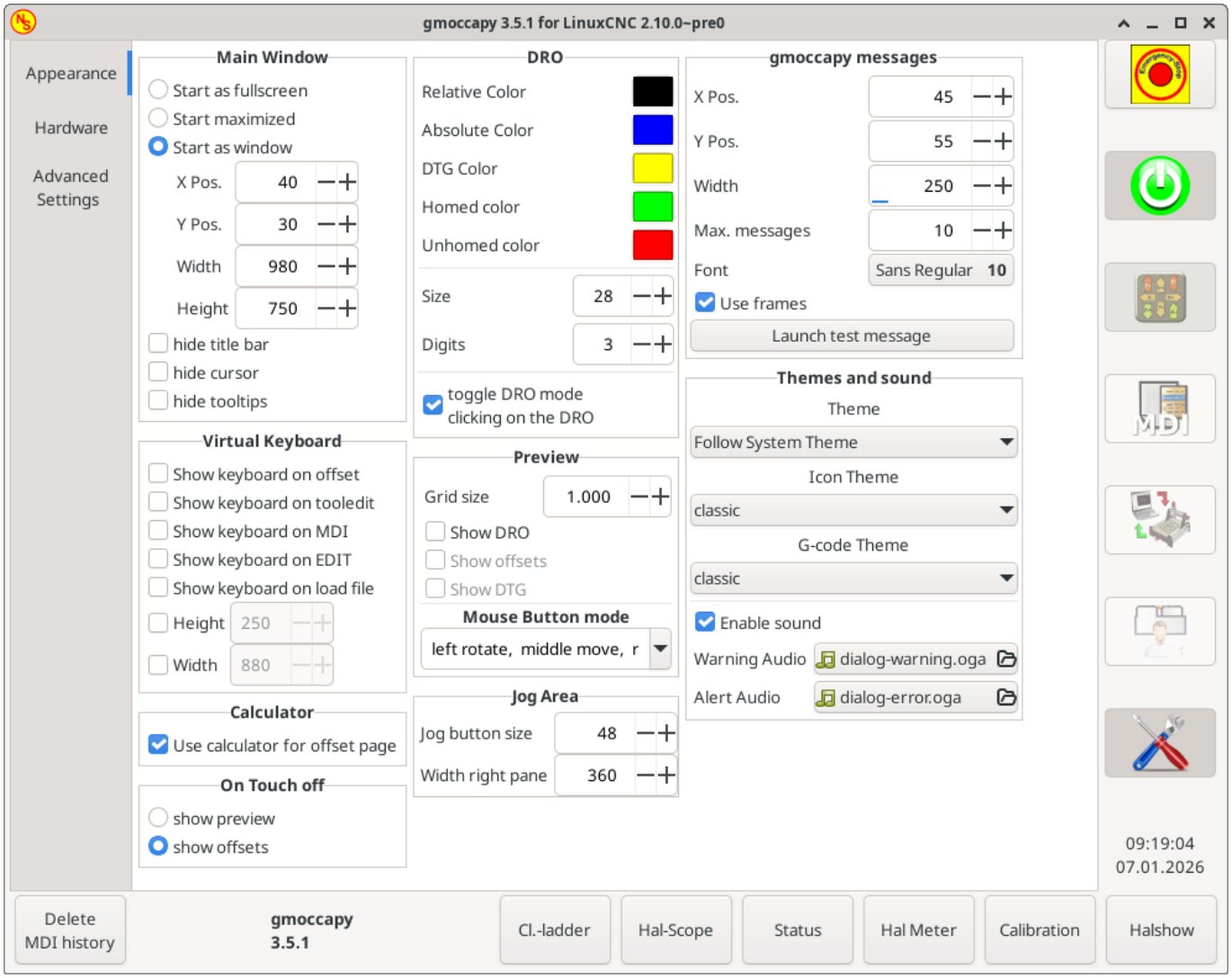


Figure 10.18: Зовнішній вигляд сторінки налаштувань GMOCCAPY

На цій вкладці ви знайдете такі опції:

Головне вікно Тут ви можете вибрати, як ви хочете запускати графічний інтерфейс. Головною причиною цього було бажання надати користувачеві простий спосіб налаштування параметрів запуску без необхідності втручання в код. У вас є три варіанти:

- *Почати в повноекранному режимі*
- *Почати розгорнутий*
- *Запустити як вікно* - Якщо ви виберете «Запустити як вікно», поля для введення позиції та розміру стануть активними. Після одноразового налаштування графічний інтерфейс буде запускатися щоразу у вибраному місці та з вибраним розміром. Проте користувач може змінювати розмір та позицію за допомогою миші, але це не вплине на налаштування.
- *hide title bar* - Дозволяє приховати рядок заголовка. (за замовчуванням: рядок заголовка видимий).

- *hide cursor* - Дозволяє приховати курсор, що дуже корисно, якщо ви використовуєте сенсорний екран.
- *hide tooltips* - Приховує підказки.

Віртуальна клавіатура Прапорці дозволяють користувачеві вибрати, чи хоче він, щоб вбудована клавіатура відображалася відразу при вході в режим MDI, на сторінці зміщення, у віджеті редагування інструментів або при відкритті програми в режимі EDIT. Ці налаштування не впливають на кнопку клавіатури в нижньому списку кнопок, тому ви можете показувати або приховувати клавіатуру, натискаючи цю кнопку.

Налаштування за замовчуванням:

- *Показати клавіатуру зі зміщенням* = False
- *Показати клавіатуру в tooledit* = False
- *Показати клавіатуру на MDI* = True
- *Показати клавіатуру під час РЕДАГУВАННЯ* = True
- *Показувати клавіатуру під час завантаження файлу* = False

Note
Якщо цей розділ не є чутливим, у вас не встановлено віртуальну клавіатуру, підтримувані *onboard* та *matchbox-keyboard*.

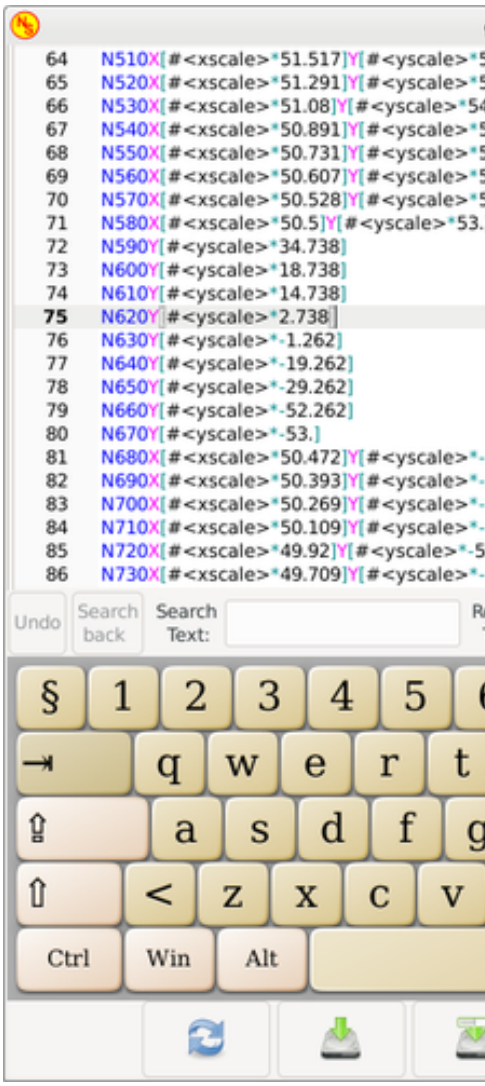
Note
Якщо розкладка клавіатури неправильна, тобто натискання клавіші Y дає Z, то розкладка не налаштована належним чином відповідно до ваших локальних налаштувань. Для вбудованої клавіатури це можна вирішити за допомогою невеликого командного файлу з таким вмістом:

```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

Літери «de» відповідають німецькій мові, їх потрібно налаштувати відповідно до ваших локальних налаштувань. Просто запустіть цей файл перед запуском LinuxCNC, це також можна зробити, додавши стартовий файл до вашої локальної папки.

```
./config/autostart
```

Щоб макет встановлювався автоматично під час запуску.
Для клавіатури Matchbox вам доведеться створити власну розкладку, для німецької розкладки запитайте на форумі.



GMOCCAPY з вбудованою клавіатурою в режимі редагування

Вимкнено на дотик Це дає можливість відображати вкладку попереднього перегляду чи вкладку зміщеної сторінки, коли ви переходите в режим дотику натисканням відповідної нижньої кнопки.

- *показати попередній перегляд*
- *показати зміщення*

Варіанти DRO Ви маєте можливість вибрати кольори фону для різних станів DRO. Таким чином, користувачі, які страждають на протанопію (слабкість червоного/зеленого), можуть вибрати відповідні кольори.

За замовчуванням кольори фону:

- *Відносний колір* = чорний
- *Абсолютний колір* = синій
- *DTG Color* = жовтий

Колір переднього плану DRO можна вибрати за допомогою:

- *Колір _Homed* = зелений
- *Неприхований колір* = червоний

Note
Ви можете змінювати режими DRO (абсолютний, відносний, відстань до кінця) натискаючи на цифру на DRO! Якщо ви натиснете на літеру зліва від DRO, з’явиться спливаюче вікно, в якому ви зможете встановити значення осей, що спростить процес налаштування, оскільки вам не доведеться натискати на кнопку «touch off bottom».

- *size* - Дозволяє встановити розмір шрифту DRO, за замовчуванням - 28, якщо ви використовуєте більший екран, ви можете збільшити розмір до 56. Якщо ви використовуєте 4 осі, розмір шрифту DRO буде 3/4 від значення, через обмеження простору.
- *digits* - Встановлює кількість цифр указівника від 1 до 5.

Note
Імперські одиниці вимірювання показуватимуть на одну цифру більше, ніж метричні. Тож, якщо ви використовуєте імперські одиниці вимірювання та встановите значення цифри на 1, ви взагалі не отримаєте жодної цифри в метричних одиницях.

- *toggle DRO mode* - Якщо неактивний, клацання мишею на DRO не призведе до жодних дій.
  За замовчуванням цей прапорець активний, тому кожне клацання на будь-якому DRO перемикає показання DRO з фактичного на відносне до DTG (відстань до кінця).
  Проте клацання на літері осі відкриє спливаюче діалогове вікно для встановлення значення осі.

Попередній перегляд

- *Grid Size* - Встановлює розмір сітки вікна попереднього перегляду. На жаль, розмір **потрібно встановлювати в дюймах**, навіть якщо ваші одиниці вимірювання метричні. Ми сподіваємося виправити це в майбутньому випуску.

Note
Сітка не буде відображатися в перспективі.

- *Show DRO* - DRO також відображатиметься на панелі попереднього перегляду, він завжди відображатиметься в повнорозмірному режимі.
- *Show DTG* - Відображатиме DTG (пряму відстань до кінцевої точки) на панелі попереднього перегляду, якщо активовано параметр «Показати DRO». В іншому випадку – лише в режимі попереднього перегляду в повному розмірі.
- *Show Offsets* - Зміщення відображатимуться на панелі попереднього перегляду, коли активовано параметр «Показати DRO». В іншому випадку – лише в режимі попереднього перегляду в повному розмірі.
- *Mouse Button Mode* - Це випадаюче поле дозволяє вибрати поведінку кнопок миші для обертання, переміщення або масштабування в попередньому перегляді:
  - поворот ліворуч, рух посередині, масштабування праворуч
  - масштабування ліворуч, рух посередині, поворот праворуч
  - ліворуч, поворот посередині, масштабування праворуч
  - масштабування ліворуч, поворот посередині, рух праворуч
  - ліворуч, масштабування посередині, поворот праворуч
  - поворот ліворуч, масштабування посередині, рух праворуч

За замовчуванням: рух ліворуч, масштабування посередині, поворот праворуч.
Коліщатко миші все одно масштабуватиме попередній перегляд у будь-якому режимі.

Tip
Якщо ви оберете елемент у попередньому перегляді, вибраний елемент буде прийнято як центр обертання, а в автоматичному режимі відповідний рядок коду буде виділено.

Повідомлення Gmoccapy

Це призведе до появи невеликих спливаючих вікон із повідомленням або текстом про помилку, подібних до тих, що відомі з AXIS. Ви можете видалити конкретне повідомлення, натиснувши кнопку закриття. Якщо ви хочете видалити останнє повідомлення, просто натисніть клавішу `WINDOWS` на клавіатурі або видаліть усі повідомлення одночасно за допомогою комбінації клавіш `Control + Space`.

Ви можете встановити деякі опції:

- *X Pos* - Положення верхнього лівого кута повідомлення в X, відлічене в пікселях від верхнього лівого кута екрана.
- *Y Pos* - Положення верхнього лівого кута повідомлення по осі Y, відліковане в пікселях від верхнього лівого кута екрана.
- *Width* - Ширина вікна повідомлення.
- *Max. messages* - Максимальна кількість повідомлень, які ви хочете бачити одночасно. Якщо встановити значення 10, 11-те повідомлення видалить перше, тож ви побачите лише останні 10.
- *Font* - Шрифт і розмір, які ви хочете використовувати для відображення повідомлень.
- *Use frames* - Якщо ви активуєте прапорець, кожне повідомлення відображатиметься в рамці, тому розрізняти повідомлення буде набагато легше. Але вам знадобиться трохи більше місця.
- Кнопка «*Запустити тестове повідомлення*» – вона відобразить повідомлення, щоб ви могли побачити зміни ваших налаштувань без необхідності генерувати помилку.

Теми та звуки Це дозволяє користувачеві вибрати тему робочого столу та звуки помилок і повідомлень, які слід відтворювати.

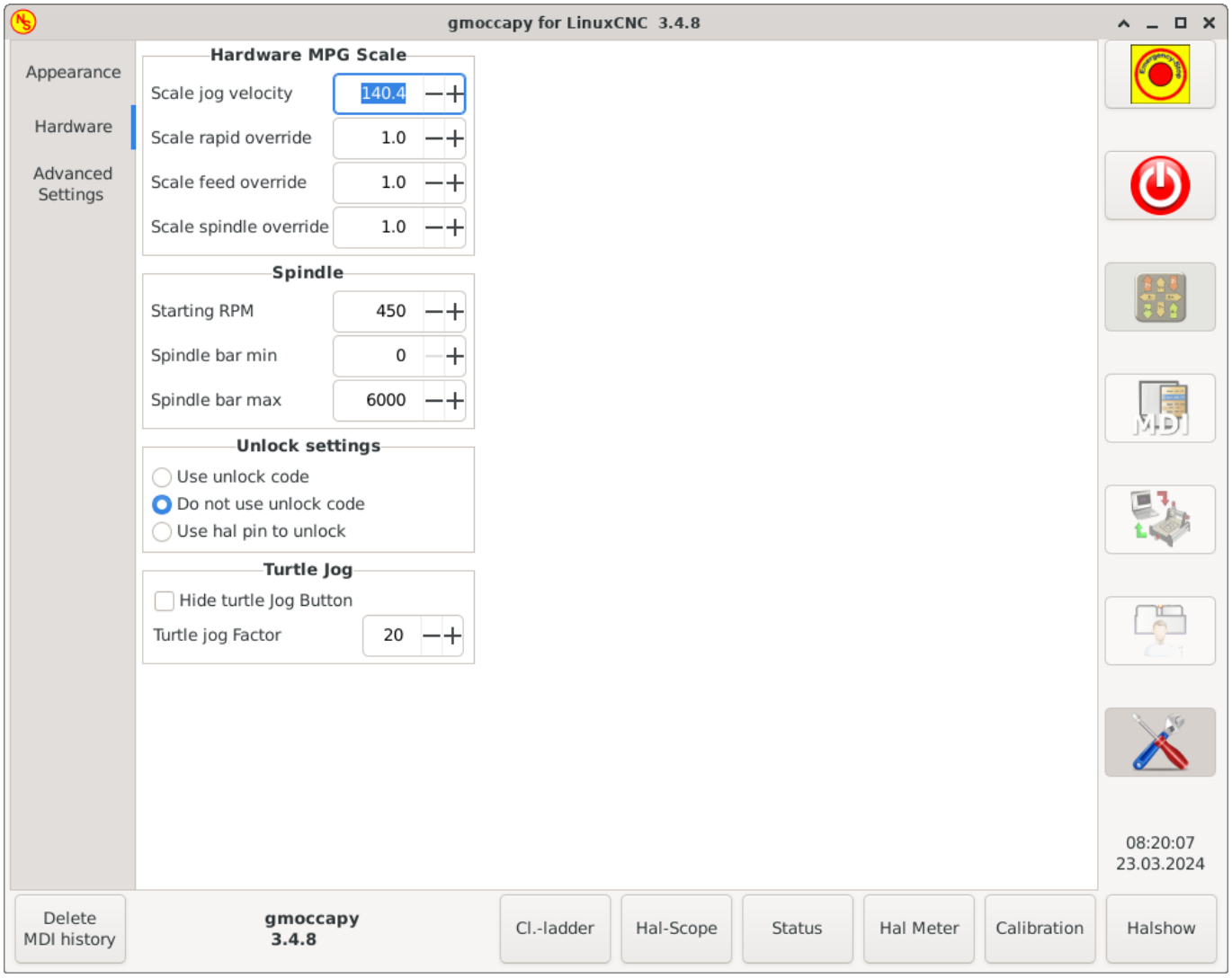
За замовчуванням встановлено опцію «Дотримуватися системної теми».

Також дозволяє змінювати тему іконок. Наразі доступні три теми:

- класичний
- матеріал
- матеріал світло

Щоб створити власні теми значків, див. розділ Тема значків для отримання детальної інформації.

10.2.7.2 Апаратне забезпечення



Апаратна шкала миль на галон Для різних контактів HAL, до яких підключаються колеса MPG, можна вибрати окремі шкали, які будуть застосовуватися. Головною причиною цього було моє власне тестування для вирішення цієї проблеми за допомогою підключень HAL, що призвело до створення дуже складного файлу HAL. Уявіть, що користувач має колесо MPG з 100 ppr і хоче уповільнити максимальну швидкість з 14000 до 2000 мм/хв, для чого потрібно 12000 імпульсів, що призведе до 120 обертів колеса! Або інший користувач, який має колесо MPG з 500 ipr і хоче встановити перевищення шпинделя, яке має обмеження від 50 до 120 %, тому він переходить від мінімального до максимального значення за 70 імпульсів, що означає навіть не 1/4 оберту.

За замовчуванням усі шкали встановлюються за допомогою розрахунку:

```
(MAX - MIN)/100
```

Шпиндель

- *Starting RPM* - Встановлює кількість обертів за хвилину, яка використовуватиметься, якщо шпиндель запущено, але значення S не встановлено.

Note
Це значення буде попередньо встановлено відповідно до ваших налаштувань у [DISPLAY] DEFAULT_SPINDLE_SPEED вашого INI-файлу. Якщо ви зміните налаштування на сторінці налаштувань, це значення буде стандартним з цього моменту, ваш INI-файл не буде змінено.

- *Spindle bar min* і *Spindle bar max* - Встановлює межі смуги шпинделя, що відображається в рамці INFO на головному екрані.

  Значення за замовчуванням:
  MIN = 0
  MAX = 6000

Note
Не помилка вказати неправильні значення. Якщо ви вкажете максимум 2000 об/хв, а ваш шпиндель обертається 4000 об/хв, то на швидкостях, вищих за 2000 об/хв, неправильним буде лише рівень прутка.

Параметри розблокування Є три варіанти розблокування сторінки налаштувань:

- *Use unlock code* - Користувач повинен ввести код, щоб увійти.
- *Do not use unlock code* - Перевірки безпеки не буде.
- *Use HAL pin to unlock* - Для розблокування налаштувань апаратний PIN-код має бути високим, див. hardware unlock pin.

За замовчуванням *використовувати код розблокування* (код за замовчуванням **123**).

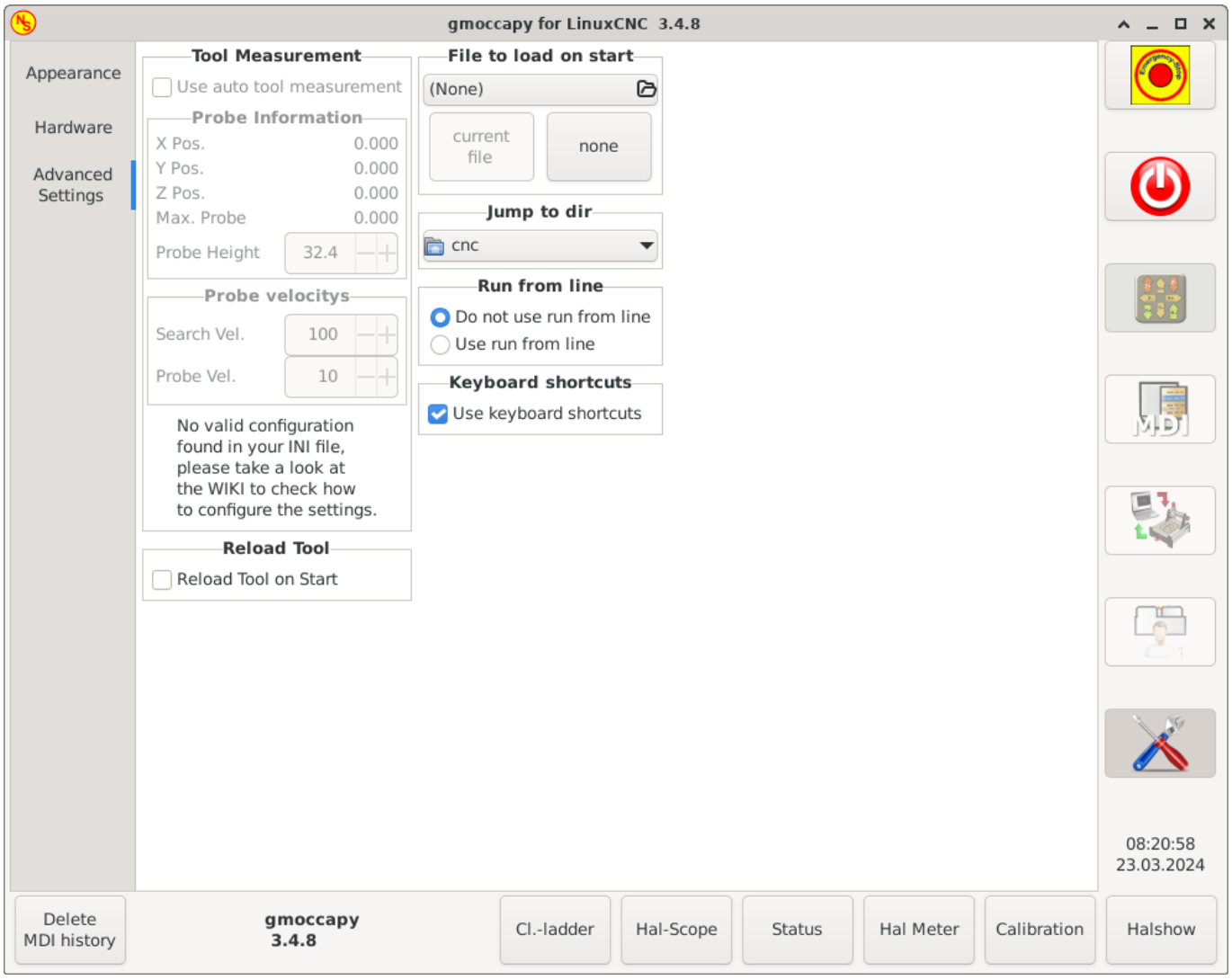
Черепашача пробіжка

Ці налаштування впливатимуть на швидкість штовхання.

- *Приховати кнопку черепашачого ходу* - Приховає кнопку праворуч від повзунка швидкості ходу. Якщо ви приховаєте цю кнопку, переконайтеся, що активовано «режим кролика», інакше ви не зможете рухатися швидше, ніж черепашачий хід, який розраховується за допомогою коефіцієнта черепашачого ходу.
- *Turtle jog factor* - Встановлює масштаб для режиму поштовхового переміщення черепахи (кнопка натиснута, відображається черепаха). Якщо встановити коефіцієнт 20, максимальна швидкість поштовхового переміщення черепахи становитиме 1/20 від максимальної швидкості машини.

Note
Цією кнопкою можна керувати за допомогою Turtle-Jog HAL Pin.

10.2.7.3 Розширені налаштування



Вимірювання інструменту

Будь ласка, перевірте Автоматичне вимірювання інструменту

Note
Якщо ця частина не є чутливою, у вас немає дійсної конфігурації INI-файлу для використання вимірювання інструменту.

- *Use auto tool measurement* - Якщо позначено, після кожної зміни інструменту буде виконуватися вимірювання інструменту, результат буде збережено в таблиці інструментів, а після зміни буде виконано команду G43.

Інформація про зонд

Наступна інформація береться з вашого INI-файлу та має бути надана в абсолютних координатах

- *X Pos.* - Положення X перемикача інструменту.
- *Y Pos.* - Положення перемикача інструменту по осі Y.
- *Z Pos.* - Положення перемикача інструменту по осі Z, ми будемо швидко рухатися до цієї координати.

- *Max. Probe* - Відстань для пошуку контакту, буде запущена помилка, якщо в цьому діапазоні немає контакту. Відстань повинна бути вказана у відносних координатах, починаючи рух від Z Pos., тому для руху вниз потрібно вказати від'ємне значення!
- *Probe Height* - Висоту перемикача зонда можна виміряти. Просто торкніться основи, на якій розташований перемикач зонда, і встановіть значення нуль. Потім змініть інструмент і подивіться значення tool_offset_z, яке і потрібно ввести в цьому полі.

Швидкості зонда

- *Search Vel.* - Швидкість пошуку перемикача інструменту. Після контакту інструмент знову підніметься, а потім знову рухатиметься до зонда зі швидкістю зонда, що дозволить отримати кращі результати.
- *Probe Vel.* - Швидкість другого руху до перемикача. Вона повинна бути повільнішою, щоб отримати кращі результати дотику. У режимі симуляції це закомментовано в macros/change.ng інакше користувач повинен був би двічі натиснути на кнопку зонда.

Інструмент перезарядки::

- *Reload Tool on Start* - Завантажує останній інструмент на початку після повернення до початкового положення.

Якщо встановлено прапорець, інструмент у шпинделі буде зберігатися при кожній зміні у файлі налаштувань, що дозволить завантажити останній встановлений інструмент при запуску. Інструмент буде завантажено після повернення всіх осей у вихідне положення, оскільки до цього не дозволяється виконувати команди MDI. Якщо ви використовуєте NO_FORCE_HOMING, ви не можете використовувати цю функцію, оскільки необхідний сигнал all_homed_signal ніколи не буде подано.

Файл для завантаження під час запуску Виберіть файл, який потрібно завантажити під час запуску. Якщо файл завантажено, його можна встановити, натиснувши кнопку «Поточний». Щоб уникнути завантаження будь-якої програми під час запуску, просто натисніть кнопку «Немає».

Екран вибору файлів буде використовувати фільтри, які ви встановили в файлі INI. Якщо фільтри не вказані, ви побачите тільки файли **NGC**. Шлях буде встановлено відповідно до налаштувань INI в `[DISPLAY] PROGRAM_PREFIX`.

Перейти до каталогу Тут ви можете встановити каталог, до якого потрібно перейти, якщо натиснути відповідну кнопку у діалоговому вікні вибору файлу.

Бігти від лінії Ви можете дозволити або заборонити запуск з рядка. Це зробить відповідну кнопку нечутливою (сірою), тому користувач не зможе використовувати цю опцію. За замовчуванням запуск з рядка вимкнено.



Warning
Не рекомендується використовувати запуск з рядка, оскільки LinuxCNC не враховуватиме попередні рядки коду перед початковим рядком. Тому помилки або збої досить ймовірні.

Сполучення клавіш Деякі користувачі хочуть керувати своїм пристроєм за допомогою клавіш клавіатури, а інші ніколи цього не дозволять. Тому кожен може вибрати, чи використовувати їх, чи ні.
Клавіатурні скорочення за замовчуванням вимкнені. Їх можна активувати за допомогою прапорця

- *Використання комбінацій клавіш*



Warning
Не рекомендується використовувати джоггинг за допомогою клавіатури, оскільки це становить серйозну небезпеку для оператора та машини.

Будь ласка, будьте обережні, якщо ви використовуєте токарний верстат, тоді скорочення будуть іншими, див. Розділ.

Загальне

- *F1* - Активувати Estop (працюватиме, навіть якщо комбінації клавіш вимкнено)
- *F2* - Увімкнення/вимкнення машини
- *F3* - Ручний режим
- *F5* - Режим MDI
- *ESC* - Перервати

У ручному режимі

- *Arrow_Left* або *NumPad_Left* - Jog X мінус
- *Arrow_Right* або *NumPad_Right* - Jog X плюс
- *Arrow_up* або *NumPad_Up* - Jog Y плюс
- *Arrow_Down* або *NumPad_Down* - Правий Y мінус
- *Page_Up* або *NumPad_Page_Up* - Jog Z плюс
- *Page_Down* або *NumPad_Page_Down* - Jog Z мінус

В автоматичному режимі

- *R* or *r* - Запустити програму
- *P* or *p* - Призупинити програму
- *S* or *s* - Продовжити програми
- *Control* + *R* або *Control* + *r* - Перезавантажити завантажений файл

Обробка повідомлень (див. Поведінка та зовнішній вигляд повідомлень)

- *WINDOWS* - Видалити останнє повідомлення
- *Control* + *Space* - Видалити всі повідомлення

10.2.8 Тема значків

Теми значків використовуються для налаштування зовнішнього вигляду та функціональності значків GMOCCAPY.

GMOCCAPY постачається з трьома різними темами іконок:

- *classic* - Класичні іконки GMOCCAPY.
- *material* – сучасна тема значків, натхненна іконками Material Icons від Google, яка автоматично приймає колір з вибраної теми робочого столу.
- *material-light* – Похідний від material, але оптимізований для світлих тем робочого столу.

Тема іконок, яка використовується в GMOCCAPY, є звичайною темою іконок GTK, що відповідає специфікації теми іконок freedestktop. Таким чином, будь-яка дійсна тема іконок GTK може бути використана як тема іконок GMOCCAPY, якщо вона містить необхідні іконки.

GMOCCAPY сканує такі каталоги на наявність тем значків:

- linuxcnc/share/gmoccapy/icons
- ~/.icons

10.2.8.1 Тема власних значків

Створити власну тему іконок досить просто. Все, що вам потрібно, це текстовий редактор і, звичайно, бажані іконки у вигляді піксельної або векторної графіки. Детальну інформацію про те, як саме створюється тема іконок, можна знайти на сайті Freedesktop Icon Theme Specification.

Почніть із створення порожнього каталогу з назвою теми іконок. Помістіть каталог в один із каталогів тем іконок GMOCCAPY. Потім нам знадобиться файл index.theme у кореневій папці нашої теми іконок, який містить необхідні метадані для теми. Це простий текстовий файл, що містить принаймні такі розділи:

- [Тема значків]

```
[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
```

  - Назва: Назва вашої теми значків.
  - Коментар: Опис вашої теми значків.
  - Успадковується: Тема значків може походити від іншої теми значків, за замовчуванням використовується hicolor.
  - Каталоги: список усіх каталогів вашої теми іконок, розділених комами.
    Кожен каталог зазвичай містить усі іконки теми певного розміру, наприклад 16x16/actions повинен містити всі іконки категорії «actions» розміром 16x16 пікселів у вигляді піксельної графіки (наприклад, файли png). Окремим випадком є каталог «scalable/actions», який містить масштабовані іконки, не прив’язані до певного розміру (наприклад, файли svg).
    Надаючи версії іконок різних розмірів, ми можемо гарантувати гарний вигляд іконок різних розмірів, а також маємо можливість змінювати іконку відповідно до її розміру, наприклад, іконка розміром 64x64 пікселів може містити більше деталей, ніж її версія розміром 16x16 пікселів.

- Для кожного каталогу нам також потрібно написати розділ у файлі index.theme:

```
[16x16/actions]
Size=16
Type=Fixed
Context=Actions

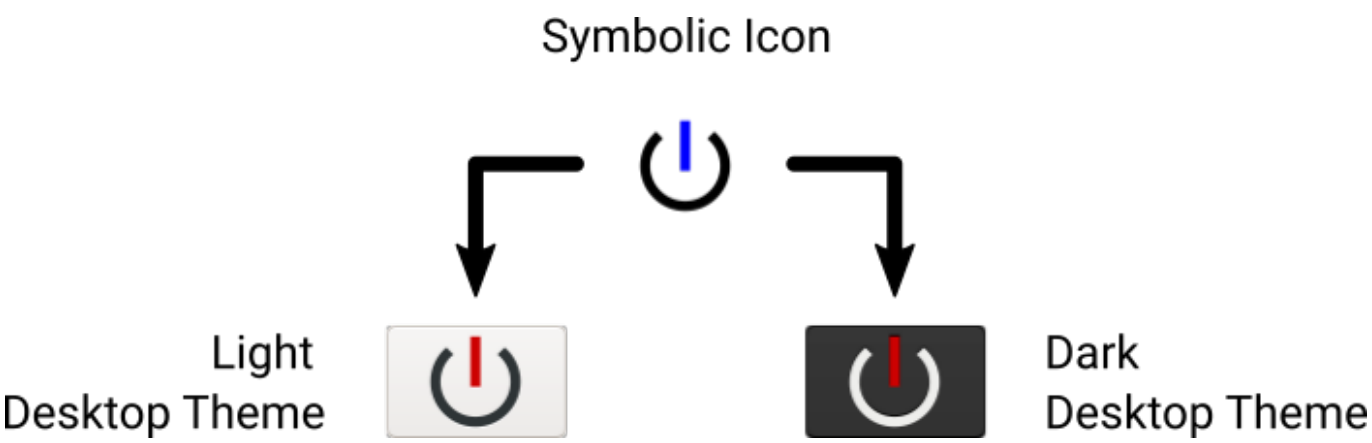
[scalable/actions]
Size=48
Type=Scalable
Context=Actions
```

  - Розмір: Номінальний розмір значка в цьому каталозі
  - Тип: Фіксований, Пороговий або Масштабований
  - Контекст: Передбачувана «категорія» іконок

В основному це все, що потрібно для створення власної теми значків.

10.2.8.2 Символічні ікони

Символічні іконки — це особливий тип іконок, зазвичай монохромні зображення. Особливість символічних іконок полягає в тому, що вони автоматично забарвлюються під час виконання відповідно до теми робочого столу. Таким чином можна створювати теми іконок, які добре поєднуються як з темними, так і зі світлими темами робочого столу (насправді це не завжди найкращий варіант, тому існує спеціальна тема «material-light»).



Щоб скористатися функцією символіки, файл значка повинен мати суфікс .symbolic.$ext (де $ext — це звичайне розширення файлу, таке як png), наприклад, ”power_on.symbolic.png”.

З такою назвою GTK розглядає це зображення як символічний значок і застосовує деякі зміни кольорів під час завантаження значка. Дозволено використовувати лише чотири кольори:

Колір	**Шістнадцятковий код**	**Опис**
чорний	`#000000`	Основний колір змінюється відповідно до основного кольору теми робочого столу.
red	`#ff0000`	Успіх: цей колір вказує на «успіх» (зазвичай щось зелене).
green	`#00ff00`	Попередження: цей колір означає «попередження» (зазвичай щось жовте/помаранчеве).
blue	`#0000ff`	Помилка: цей колір вказує на «помилку» (зазвичай щось червонувате).

Tip
Приклади символічних значків можна знайти за адресою `linuxcnc/share/gmoccapy/icons/material`.

10.2.9 Розділ, специфічний для токарного верстата

Якщо в файлі INI вказано `LATHE = 1`, графічний інтерфейс зміниться відповідно до особливих потреб токарного верстата. В основному, вісь Y буде прихована, а кнопки ручного переміщення будуть розташовані в іншому порядку.

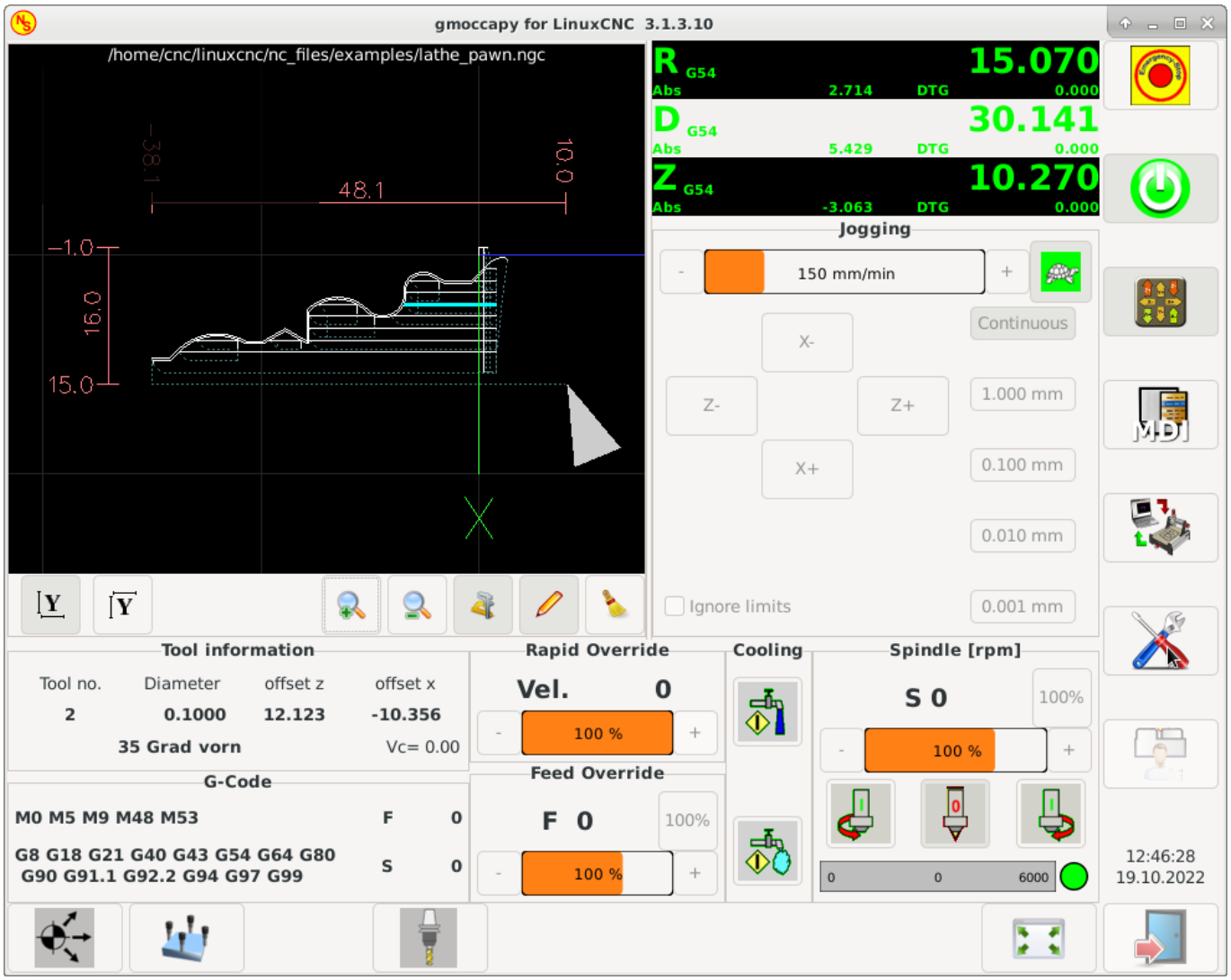


Figure 10.19: Звичайний токарний верстат

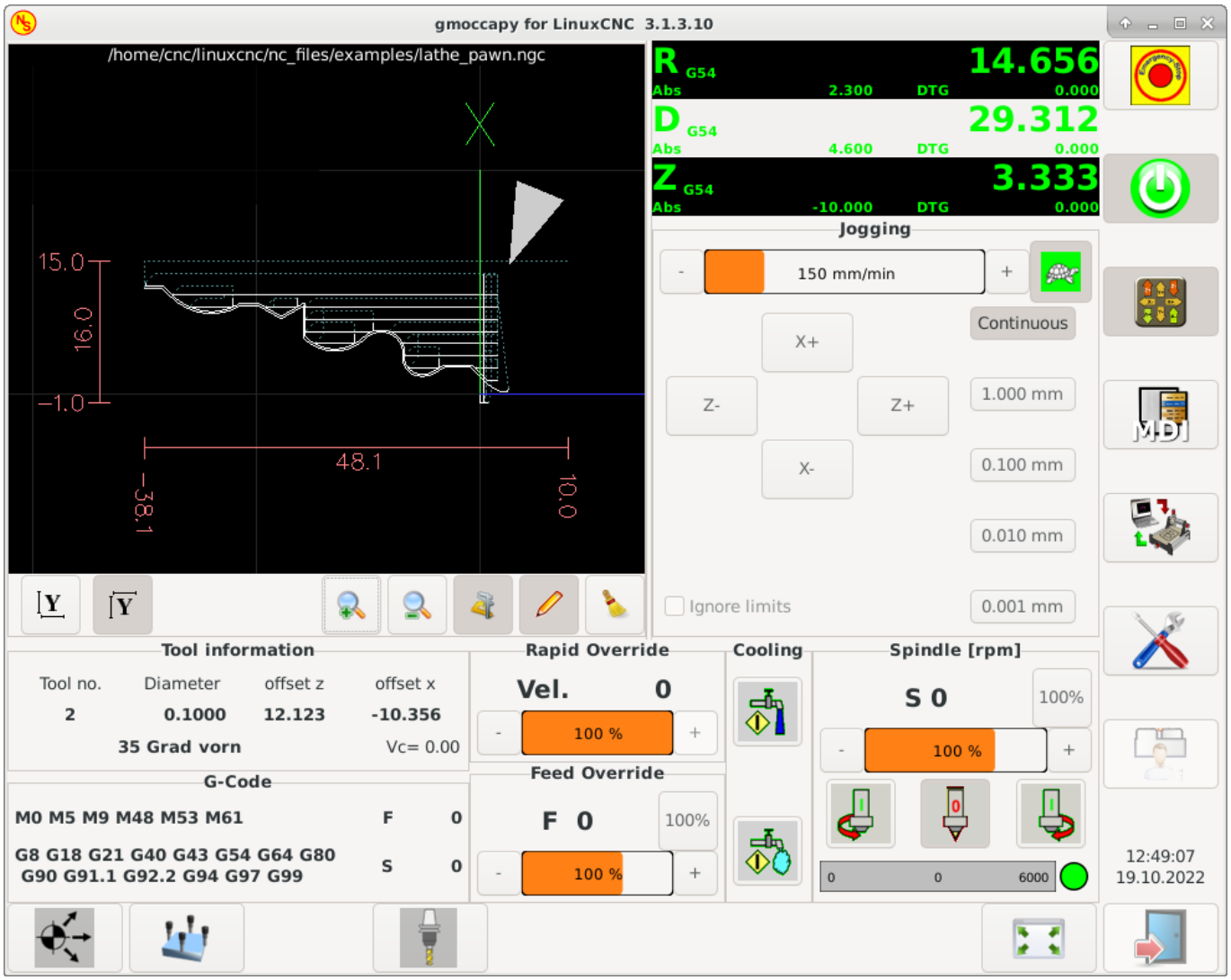


Figure 10.20: Токарний верстат для заднього інструменту

Як бачите, R DRO має чорний фон, а D DRO — сірий. Це змінюється відповідно до активного G-коду G7 або G8. Активний режим видно за чорним фоном, тобто на показаних зображеннях активний G8.

Наступною відмінністю від стандартного екрану є розташування кнопок перемикання. X і Z поміняліся місцями, а Y зникла. Ви помітите, що кнопки X+ і X- змінюють своє розташування відповідно до нормального або зворотного інструменту токарного верстата.

Також зміниться поведінка клавіатури:

Звичайний токарний верстат:

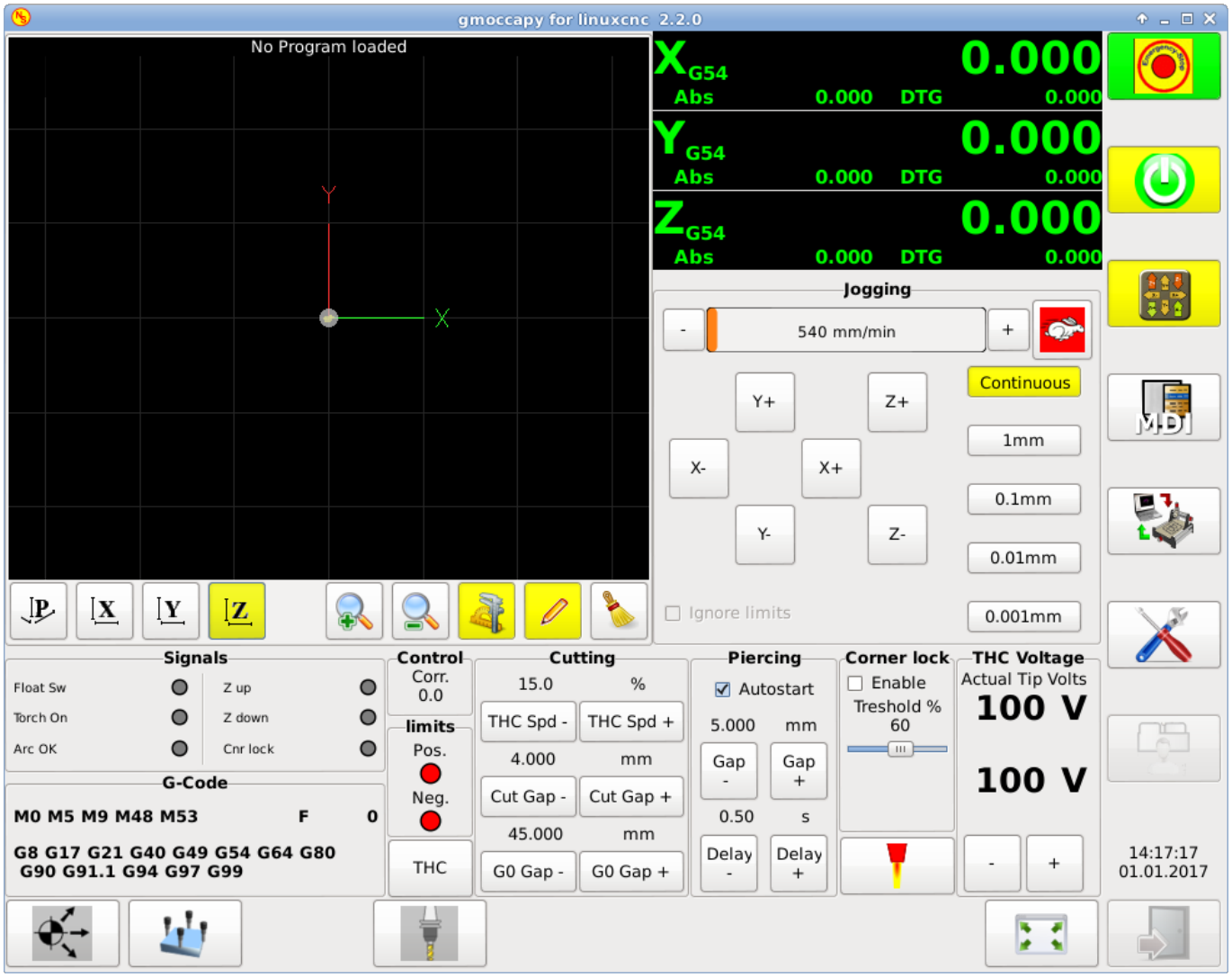
- *Arrow_Left* або *NumPad_Left* - Jog Z мінус
- *Arrow_Right* або *NumPad_Right* - Jog Z плюс
- *Arrow_up* або *NumPad_Up* - Jog X мінус
- *Arrow_Down* або *NumPad_Down* - Jog X плюс

Токарний верстат із заднім інструментом:

- *Arrow_Left* або *NumPad_Left* - Jog Z мінус
- *Arrow_Right* або *NumPad_Right* - Jog Z плюс
- *Arrow_up* або *NumPad_Up* - Jog X плюс
- *Arrow_Down* або *NumPad_Down* - Jog X мінус

У інформаційному полі інструменту відображатиметься не лише зміщення Z, але й зміщення X, а таблиця інструментів відображатиме всю інформацію, що стосується токарного верстата.

10.2.10 Розділ, специфічний для плазми



Існує дуже гарна BIKI, яка насправді розвивається, підтримується Маріусом, див. сторінка вікі про Плазму.

10.2.11 Відео на YouTube

Нижче наведено серію відеороликів, які демонструють роботу GMOCCAPY. На жаль, ці відеоролики не показують останню версію GMOCCAPY, але спосіб її використання залишиться таким самим, як і в поточній версії. Я оновлю відеоролики якомога швидше.

10.2.11.1 Базове використання

https://youtu.be/O5B-s3uiI6g

10.2.11.2 Імітація джог-колес

https://youtu.be/ag34SGxt97o

10.2.11.3 Сторінка налаштувань

https://youtu.be/AuwhSHRJoiI

10.2.11.4 Імітація апаратної кнопки

Німецька: https://youtu.be/DTqhY-MfzDE
Англійська: https://youtu.be/ItVWJBK9WFA

10.2.11.5 Вкладки користувача

https://youtu.be/rG1zmeqXyZI

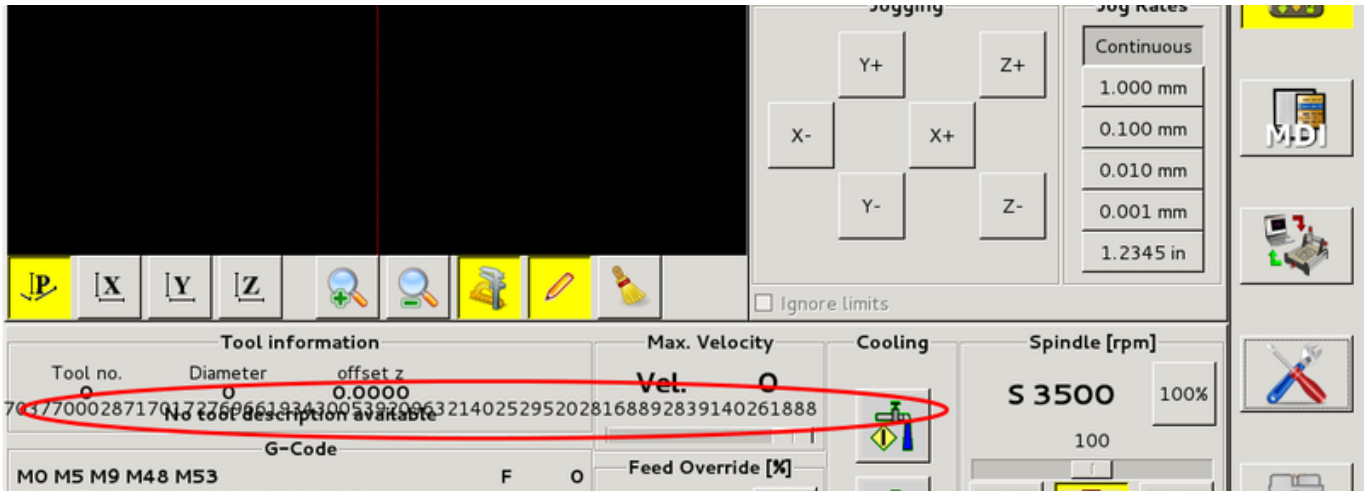
10.2.11.6 Відео про вимірювання інструментів

Моделювання вимірювання автоматичного інструменту: https://youtu.be/rrkMw6rUFdk
Екран автоматичного вимірювання інструменту: https://youtu.be/Z2ULDj9dzvk
Автоматична машина для вимірювання інструментів: https://youtu.be/1arucCaDdX4

10.2.12 Відомі проблеми

10.2.12.1 Дивні цифри в інформаційній області

Якщо ви отримуєте дивні числа в інформаційній області GMOCCAPY, наприклад:



Ви створили файл конфігурації за допомогою старішої версії StepConfWizard. Він створив неправильний запис у файлі INI під назвою [TRAJ] з назвою MAX_LINEAR_VELOCITY = xxx. Змініть цей запис на MAX_VELOCITY = xxx.

10.2.12.2 Не завершується макрос

Якщо ви використовуєте макрос без руху, як-от цей:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

o<zeroxy> endsub
m2
```

GMOCCAPY не побачить кінець макросу, оскільки інтерпретатор повинен змінити свій стан на IDLE, але макрос навіть не встановлює інтерпретатор у новий стан. Щоб уникнути цього, просто додайте рядок G4 P0.1, щоб отримати необхідний сигнал. Правильний макрос буде таким:

```
 o<zeroxy> sub

G92.1
G92.2
G40

G10 L20 P0 X0 Y0

G4 P0.1

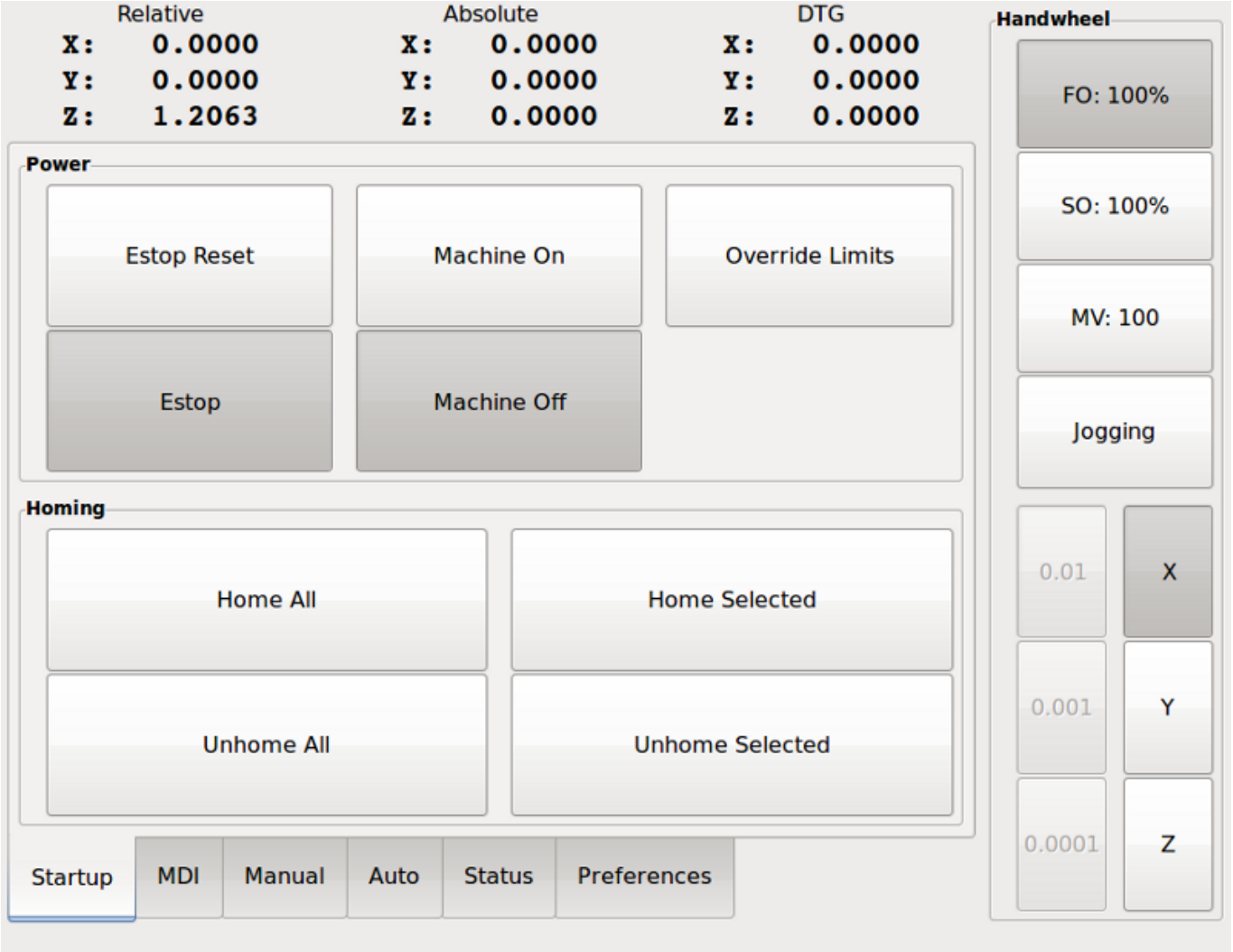
o<zeroxy> endsub
m2
```

10.3 Сенсорний графічний інтерфейс користувача

Touchy — це інтерфейс користувача для LinuxCNC, призначений для використання на панелях керування верстатами, тому не потребує клавіатури чи миші.

Він призначений для використання із сенсорним екраном і працює в поєднанні з кермом/MPG та кількома кнопками та перемикачами.

Вкладка «Ручне колесо» має перемикачі для вибору між функціями «Перевищення подачі», «Перевищення швидкості шпинделя», «Максимальна швидкість» та «Поступальний рух» для введення колеса/MPG. Також передбачені перемикачі для вибору осі та кроку поступального руху.



10.3.1 Конфігурація панелі

10.3.1.1 HAL-з’єднання

Touchy шукає в файлі INI під заголовком «[HAL]» записи «POSTGUI_HALFILE=<ім’я файлу>». Зазвичай «<ім’я файлу>» буде «touchy_postgui.hal», але це може бути будь-яке допустиме ім’я файлу. Ці команди виконуються після побудови екрана, що гарантує доступність контактів HAL віджета. У файлі INI може бути кілька рядків «POSTGUI_HALFILE=<ім’я файлу>». Кожен з них буде виконуватися по черзі в тому порядку, в якому вони з’являються у файлі INI.

Note
Раніше Touchy вимагав створення файлу з назвою «touchy.hal» у вашому каталозі конфігурації (каталозі, в якому знаходиться ваш INI-файл). З міркувань сумісності це буде продовжувати працювати, але INI-файли postgui є кращим варіантом.

Для отримання додаткової інформації про файли HAL та команду net див. Основи HAL.

Touchy має кілька вихідних контактів, призначених для підключення до контролера руху для керування біговим переміщенням колеса:

- *touchy.jog.wheel.increment*, який має бути підключений до контакту *axis*.N.*jog-scale* кожної осі *N*.
- *touchy.jog.wheel*.N, який має бути підключений до *axis*.N.*jog-enable* для кожної осі *N*.

Note
N представляє номер осі 0-8.

- Окрім підключення до «touchy.wheel-counts», кількість обертів колеса також повинна бути підключена до «axis.*N*.jog-counts» для кожної осі *N*. Якщо ви використовуєте компонент HAL «ilowpass» для згладжування коливань колеса, переконайтеся, що згладжується тільки «axis.*N*.jog-counts», а не «touchy.wheel-counts».

Необхідні елементи керування

- Кнопка переривання (мимохідний контакт), підключена до контакту HAL «touchy.abort».
- Кнопка запуску циклу (мимохідний контакт) підключена до *touchy.cycle-start*.
- Wheel/MPG, підключено до *touchy.wheel-counts* та контактів руху, як описано вище.
- Одинарний блок (тумблер), підключений до *touchy.single-block*.

Додаткові елементи керування

- Для безперервного поштовхового переміщення, один двонаправлений перемикач миттєвого перемикання (або дві кнопки миттєвого перемикання) з центруванням поза центром для кожної осі, підключені до *touchy.jog.continuous.x.negative*, *touchy.jog.continuous.x.positive* тощо.
- Якщо потрібна кнопка підйому пінолі (для поштовхового переміщення Z до верхньої точки руху з максимальною швидкістю), кнопка тимчасового натискання, підключена до «touchy.quill-up».

Додаткові панельні лампи

- *touchy.jog.active* показує, коли елементи керування поштовхом панелі активні.
- «touchy.status-indicator» світиться, коли верстат виконує G-код, і блимає, коли верстат виконує код, але знаходиться в режимі паузи/зупинки подачі.

10.3.1.2 Рекомендовано для будь-якої конфігурації

- Кнопка Estop, вбудована в ланцюг Estop

10.3.2 Налаштування

10.3.2.1 Увімкнення Touchy

Щоб використовувати Touchy, у розділі *[DISPLAY]* вашого INI-файлу змініть рядок вибору дисплея на *DISPLAY = touchy*.

10.3.2.2 Параметри

Під час першого запуску Touchy перевірте вкладку «Налаштування». Якщо ви використовуєте сенсорний екран, для найкращих результатів виберіть опцію приховування курсора.

Вікно стану має фіксовану висоту, яка визначається розміром фіксованого шрифту. На це може впливати DPI Gnome, яке налаштовується в Система / Налаштування / Зовнішній вигляд / Шрифти / Деталі. Якщо нижня частина екрана обрізана, зменште значення DPI.

Усі інші розміри шрифту можна змінити на вкладці «Налаштування».

10.3.2.3 Макроси

Touchy може викликати макроси O-word за допомогою інтерфейсу MDI. Щоб налаштувати це, у розділі *[MACROS]* INI-файлу додайте один або кілька рядків *MACRO*. Кожен з них має бути такого формату:

```
MACRO=increment xinc yinc
```

У цьому прикладі increment – це назва макросу, і він приймає два параметри з назвами xinc та yinc.

Тепер помістіть макрос у файл з назвою *increment.ngc*, у директорії *PROGRAM_PREFIX* або будь-якій директорії в *SUBROUTINE_PATH*.

Це має виглядати так:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Зверніть увагу, що назва підпрограми точно відповідає назві файлу та назві макросу, включаючи регістр літер.

Коли ви викликаєте макрос, натиснувши кнопку «Макрос» на вкладці MDI в Touchy, ви можете ввести значення для xinc та yinc. Вони передаються до макросу як «#1» та «#2» відповідно. Параметри, які ви залишили порожніми, передаються як значення 0.

Якщо є кілька різних макросів, натискайте кнопку Макрос кілька разів, щоб перемикатися між ними.

У цьому простому прикладі, якщо ввести -1 для xinc та натиснути кнопку початку циклу, буде запущено швидкий рух *G0*, який перемістить одиницю ліворуч.

Ця макрофункція корисна для зондування країв/отворів та інших завдань налаштування, а також, можливо, для фрезерування отворів або інших простих операцій, які можна виконувати з панелі без необхідності використання спеціально написаних програм G-коду.

10.4 G-екран

10.4.1 Вступ

Gscreen — це інфраструктура для відображення спеціального екрана для керування LinuxCNC. Gscreen багато запозичує з GladeVCP. GladeVCP використовує редактор віджетів GTK GLADE для створення віртуальних панелей керування (VCP) за допомогою миші. Gscreen поєднує це з програмуванням на Python для створення графічного інтерфейсу користувача для керування верстатом з CNC.

Gscreen можна налаштувати, якщо ви хочете мати інші кнопки та світлодіоди стану. Gscreen підтримує GladeVCP, який використовується для додавання елементів керування та індикаторів. Для налаштування Gscreen використовується редактор Glade. Gscreen не обмежується додаванням настроюваної панелі праворуч або настроюваної вкладки, він є повністю редагованим.

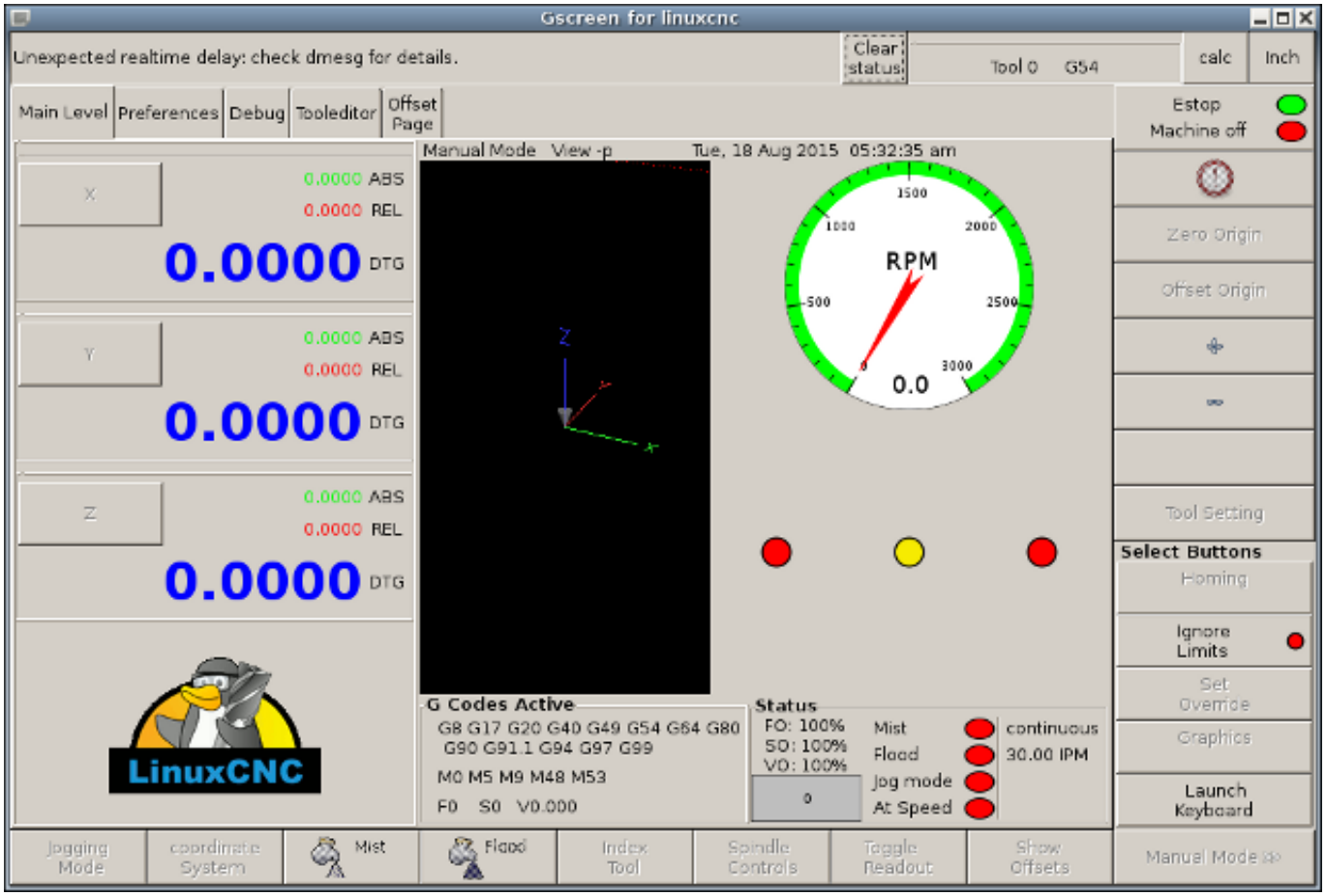


Figure 10.21: Стандартний екран Gscreen

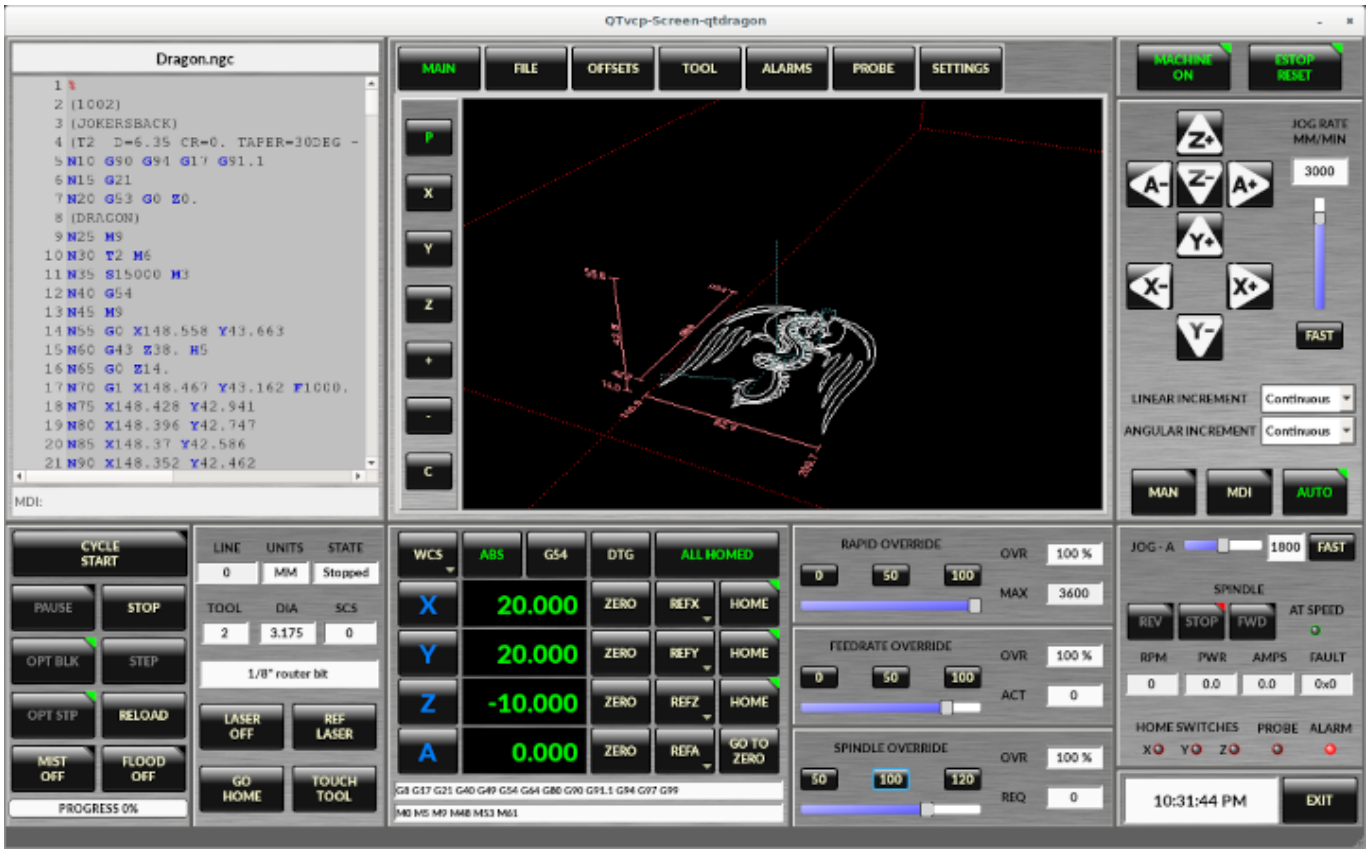


Figure 10.22: Екран Silverdragon Gscreen

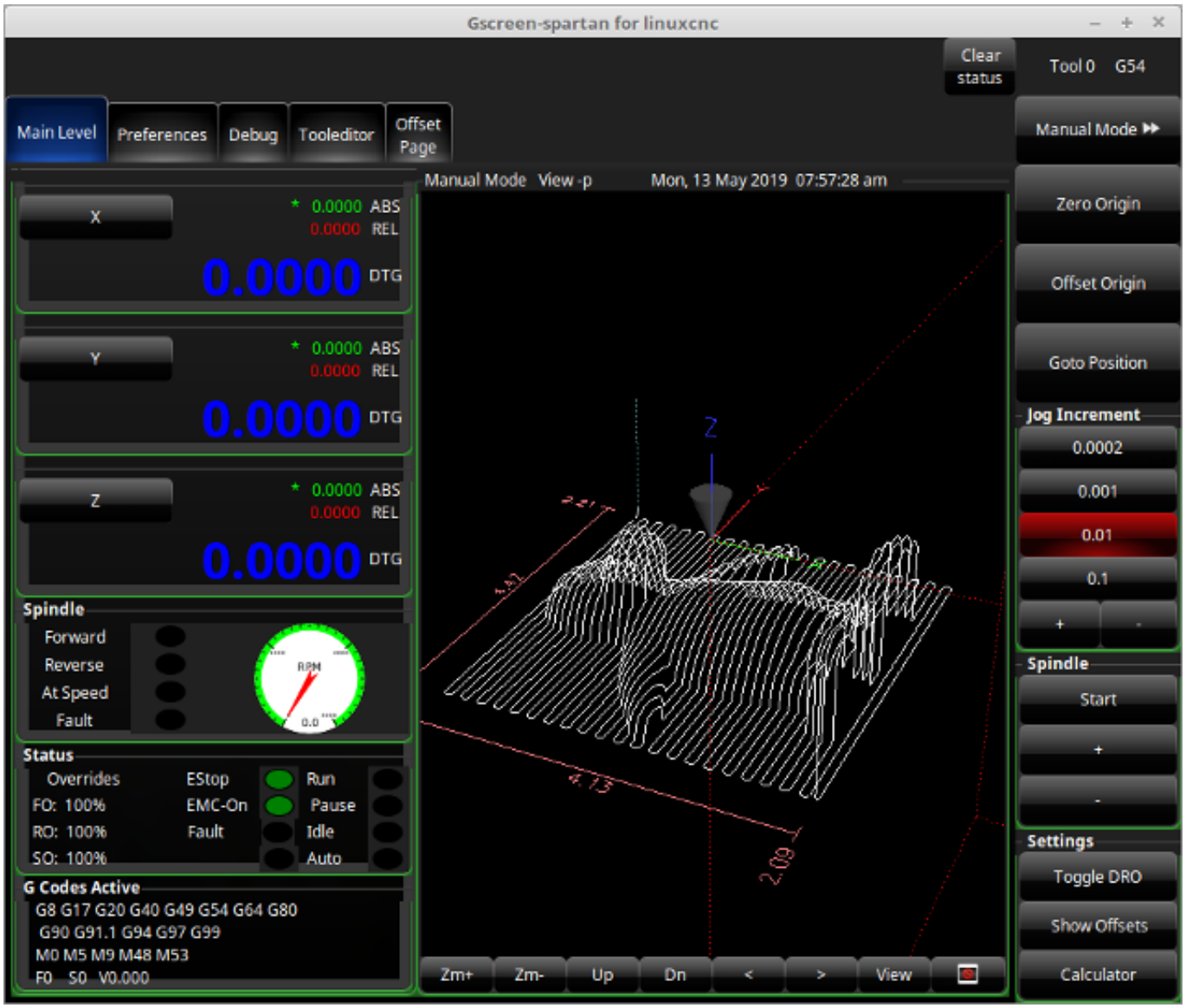


Figure 10.23: Gscreen Spartan Screen

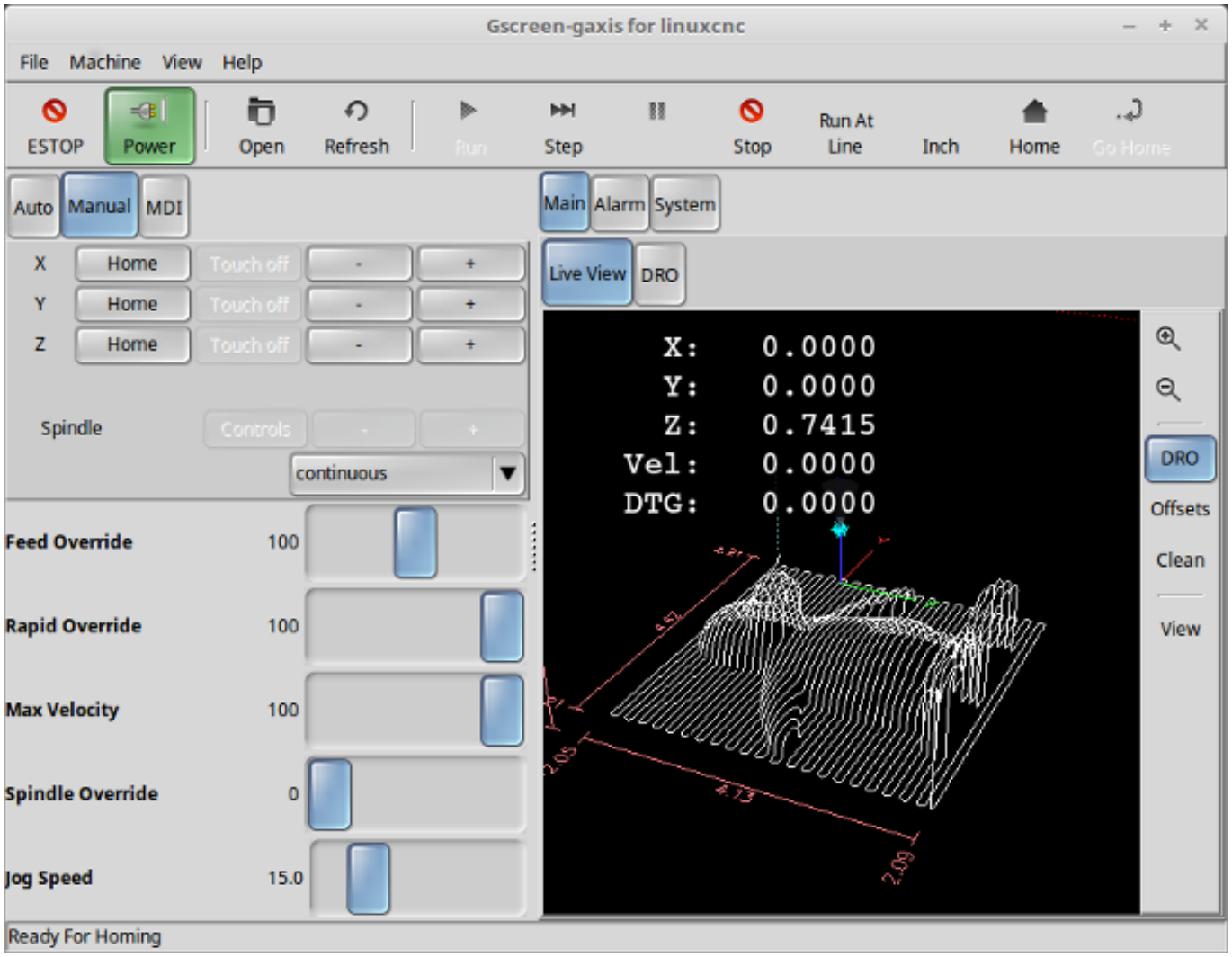


Figure 10.24: Екран Gscreen Gaxis



Figure 10.25: Промисловий екран Gscreen

Gscreen базується на «Glade» (редакторі), «PyGTK» (наборі інструментів для створення віджетів) та «GladeVCP» (з'єднанні LinuxCNC з Glade та PyGTK). GladeVCP має кілька спеціальних віджетів та дій, доданих спеціально для LinuxCNC. Віджет — це загальна назва, яка використовується для кнопок, повзунків, міток тощо набору інструментів PyGTK.

10.4.1.1 Щасливий файл

Файл Glade — це текстовий файл, організований за стандартом XML, який описує макет і віджети екрана. PyGTK використовує цей файл для фактичного відображення та реагування на ці віджети. Редактор Glade дозволяє відносно легко створювати та редагувати цей файл. Ви повинні використовува редактор Glade 3.38.2, який використовує віджети GTK3.

10.4.1.2 PyGTK

PyGTK — це зв'язування Python з GTK. GTK — це «інструментарій» візуальних віджетів, він запрограмов на C. PyGTK використовує Python для «зв'язування» з GTK.

10.4.2 GladeVCP

GladeVCP об’єднує LinuxCNC, HAL, PyGTK і Glade. LinuxCNC потребує деяких спеціальних віджетів, тому GladeVCP їх надає. Багато з них є просто розширеннями HAL для існуючих віджетів PyGTK. GladeVCP створює контакти HAL для спеціальних віджетів, описаних у файлі Glade. GladeVCP також дозволяє додавати команди Python для взаємодії з віджетами, щоб вони виконували дії, недоступні в їх стандартній формі. Якщо ви можете створити панель GladeVCP, ви можете налаштувати Gscreen!

10.4.2.1 Огляд

Існує два файли, які можна використовувати окремо або в поєднанні для додавання налаштувань. Локальні файли Glade та файли обробників. Зазвичай Gscreen використовує стандартний файл Glade та, можливо, файл обробника (якщо використовується зразок «шкіри»). Ви можете вказати Gscreen використовувати «локальні» файли Glade та обробників. Gscreen шукає в папці, яка містить усі файли конфігурації для вибраної вами конфігурації.

Локальні файли Glade Якщо вони присутні, замість стандартних файлів Glade будуть завантажені локальні файли Glade з папки конфігурації. Локальні файли Glade дозволяють використовувати власні дизайни замість стандартних екранів. В INI-файлі є перемикач для встановлення базового імені: -c name, щоб Gscreen шукав MYNAME.glade та MYNAME_handler.py.

Ви можете вказати Gscreen просто завантажити файл Glade і не підключати до нього внутрішні сигнали. Це дозволяє gscreen завантажувати будь-який файл Glade, збережений GTK builder. Це означає, що ви можете відображати повністю налаштований екран, але для цього потрібно використовувати файл обробника. Gscreen використовує файл Glade для визначення віджетів, щоб відображати їх і взаємодіяти з ними. Багато з них мають конкретні імена, інші мають загальні імена, надані Glade. Якщо віджет буде відображатися, але ніколи не змінюватиметься, то загальне ім’я підійде. Якщо потрібно керувати віджетом або взаємодіяти з ним, то йому надається, сподіваємося, цілеспрямоване ім’я (усі імена повинні бути унікальними). Віджети також можуть мати сигнали, визначені для них у редакторі GLADE. Він визначає, який сигнал подається і який метод викликати.

Зміна стандартних скінів Якщо ви зміните ім’я віджета, Gscreen може не знайти його. Якщо на цей віджет є посилання в коді Python, в кращому випадку віджет перестане працювати, в гіршому — виникне помилка під час завантаження. Стандартні екрани Gscreen не використовують багато сигналів, визначених в редакторі, вони визначаються в коді Python. Якщо ви перемістите (виріжете і вставите) віджет із сигналами, сигнали не будуть скопійовані. Ви повинні додати їх знову вручну.

Файли обробників Файл обробника — це файл, що містить код Python, який Gscreen додає до своїх стандартних процедур. Файл обробника дозволяє змінювати стандартні налаштування або додавати логіку до оболонки Gscreen без необхідності змінювати сам Gscreen. Ви можете комбінувати нові функції з функціями Gscreen, щоб змінювати поведінку на свій розсуд. Ви можете повністю обійти всі функції Gscreen і змусити його працювати зовсім по-іншому. Якщо присутній файл обробника з назвою gscreen_handler.py (або MYNAME_handler.py, якщо використовується перемикач INI), він буде завантажений і зареєстрований. Допускається лише один файл. Gscreen шукає файл обробника, і якщо його знаходить, то шукає конкретні імена функцій і викликає їх замість стандартних. Якщо ви додаєте віджети, ви можете налаштувати виклики сигналів з редактора Glade для виклику процедур, які ви написали у файлі обробника. Таким чином ви можете отримати налаштовану поведінку. Процедури обробника можуть викликати процедури Gscreen за замовчуванням, як до, так і після виконання власного коду. Таким чином ви можете додати додаткову поведінку, наприклад додавання звуку. Дивіться розділ GladeVCP для ознайомлення з основами файлів обробника GladeVCP. Gscreen використовує дуже схожу техніку.

Теми Gscreen використовує інструментарій PyGTK для відображення екрану. PyGTK — це зв’язок мови Python із GTK. GTK підтримує «теми». Теми — це спосіб змінити зовнішній вигляд і відчуття від віджетів на екрані. Наприклад, за допомогою тем можна змінити колір або розмір кнопок і

повзунків. В Інтернеті доступно багато тем GTK. Теми також можна налаштовувати для зміни зовнішнього вигляду певних віджетів. Це тісніше пов'язує файл теми з файлом Glade. Деякі зразки оболонок екрану дозволяють користувачеві вибирати будь-яку з тем у системі. Зразок «gscreen» є прикладом. Деякі завантажують тему з такою ж назвою у файлі конфігурації. Зразок «gscreen-gaxis» є прикладом. Це робиться шляхом розміщення папки теми в папці конфігурації, яка містить файли INI та HAL, і присвоєння їй імені: SCREENNAME_theme (SCREENNAME — це базова назва файлів, наприклад gaxis_theme). Усередині цієї папки є інша папка під назвою gtk-2.0, всередині якої знаходяться файли теми. Якщо ви додасте цей файл, Gscreen за замовчуванням буде використовувати цю тему при запуску. gscreen-gaxis має зразок настроюваної теми, яка шукає певні віджети з іменами та змінює візуальну поведінку цих конкретних віджетів. Кнопки Estop та machine-on використовують інші кольори, ніж решта кнопок, щоб вони виділялися. Це робиться у файлі обробника, надаючи їм конкретні імена та додаючи конкретні команди у файл gtkrc теми. Детальнішу інформацію про теми GTK (зразкова тема використовує механізм тем pixmap) див.: GTK Themes, Pixmap Theme Engine.

10.4.2.2 Створіть панель GladeVCP

Gscreen — це просто велика складна панель GladeVCP з кодом на Python для керування нею. Щоб налаштувати її, нам потрібно завантажити файл Glade в редактор Glade.

Встановлено LinuxCNC Якщо у вас встановлено LinuxCNC 2.6+ на Ubuntu 10.04, просто запустіть редактор Glade з меню програм або з терміналу. Новіші версії Linux вимагатимуть встановлення Glade 3.8.0 - 3.8.6 (можливо, вам доведеться самостійно скомпілювати його).

Команди, скомпільовані RIP Використовуючи скомпільовану з вихідного коду версію LinuxCNC, відкрийте термінал і cd у верхній частині папки LinuxCNC. Налаштуйте середовище, ввівши *. ./scripts/rip-environment*, а тепер введіть *glade*. У терміналі з'явиться низка попереджень, які можна проігнорувати, і редактор має відкритися. Стандартний файл Gscreen Glade знаходиться в: src/emc/usr_intf/gscreen/, зразки скінів знаходяться в /share/gscreen/skins/. Його слід скопіювати в папку конфігурації. Або ви можете створити чистий файл Glade, зберігши його в папці конфігурації.

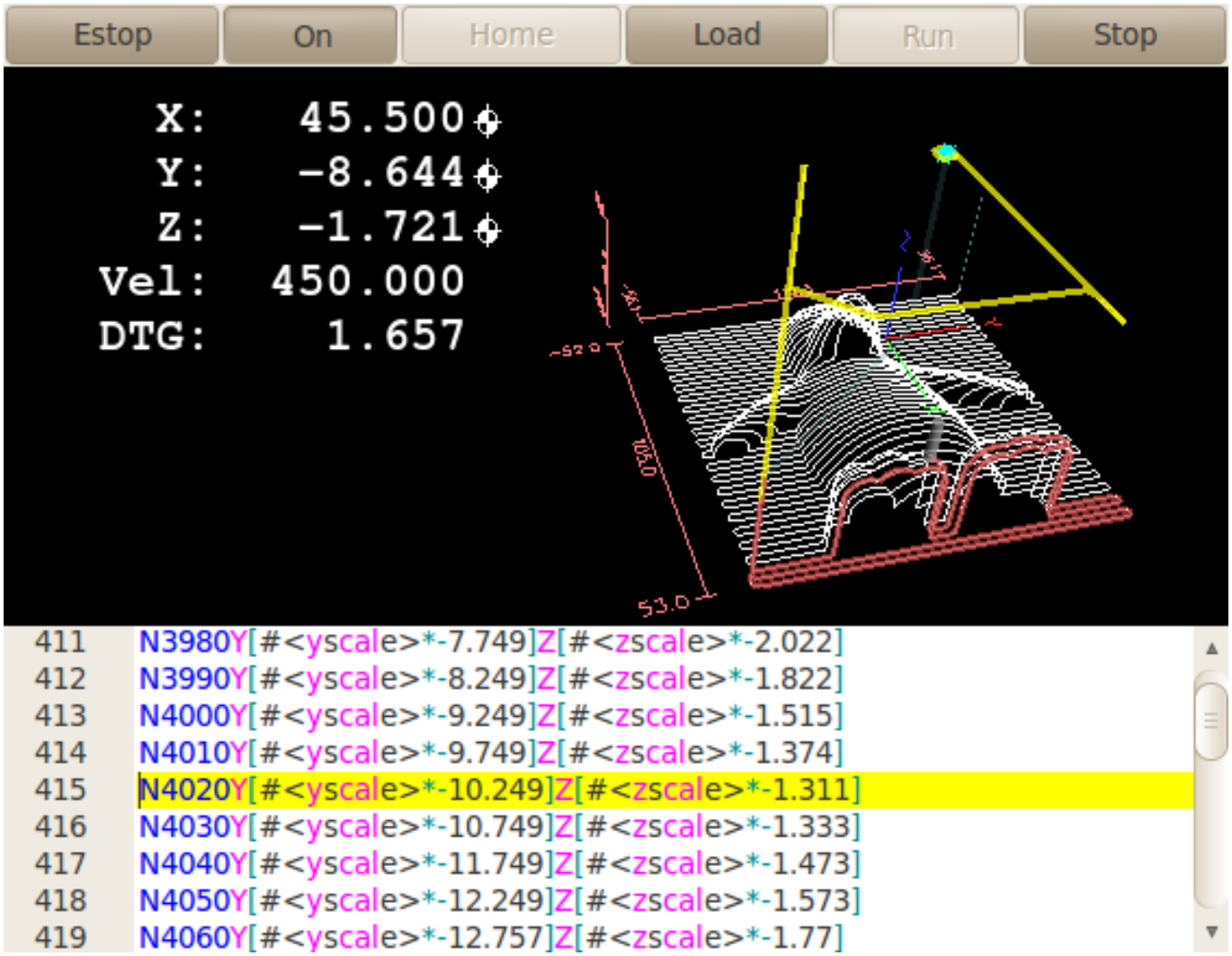
Ви завантажили стандартний файл Glade і тепер можете його редагувати. Перше, що ви помітите, це те, що в редакторі він виглядає не так, як відображається в Gscreen, який використовує деякі хитрощі, такі як приховування всіх полів кнопок, крім одного, і зміна цього поля залежно від режиму. Те саме стосується ноутбуків, деякі екрани використовують ноутбуки з невидимими вкладками. Щоб змінити сторінки в редакторі, вам потрібно тимчасово показати ці вкладки.

При внесенні змін набагато простіше додавати віджети, ніж видаляти їх, і при цьому екран продовжує працювати належним чином. Одним із способів змінити відображення без помилок є зробити об'єкти «невидимими». Це не завжди працює, оскільки деякі віджети знову стануть видимими. Зміна імен звичайних віджетів Gscreen, ймовірно, не буде працювати без зміни коду Python, але переміщення віджета з збереженням імені зазвичай є можливим.

Gscreen максимально використовує віджети GladeVCP, щоб уникнути додавання коду Python. Необхідно ознайомитися з віджетами GladeVCP. Якщо існуючі віджети надають вам потрібні функції, то додавати код Python не потрібно, просто збережіть файл Glade у вашій папці конфігурації. Якщо вам потрібно щось більш індивідуальне, то вам доведеться написати код Python. Ім'я батьківського вікна має бути window1. Gscreen використовує це ім'я.

Пам'ятайте, що якщо ви використовуєте опцію користувацького екрана, ВИ несете відповідальність за її виправлення (за потреби) під час оновлення LinuxCNC.

10.4.3 Створення простого користувацького екрану з чистого аркуша

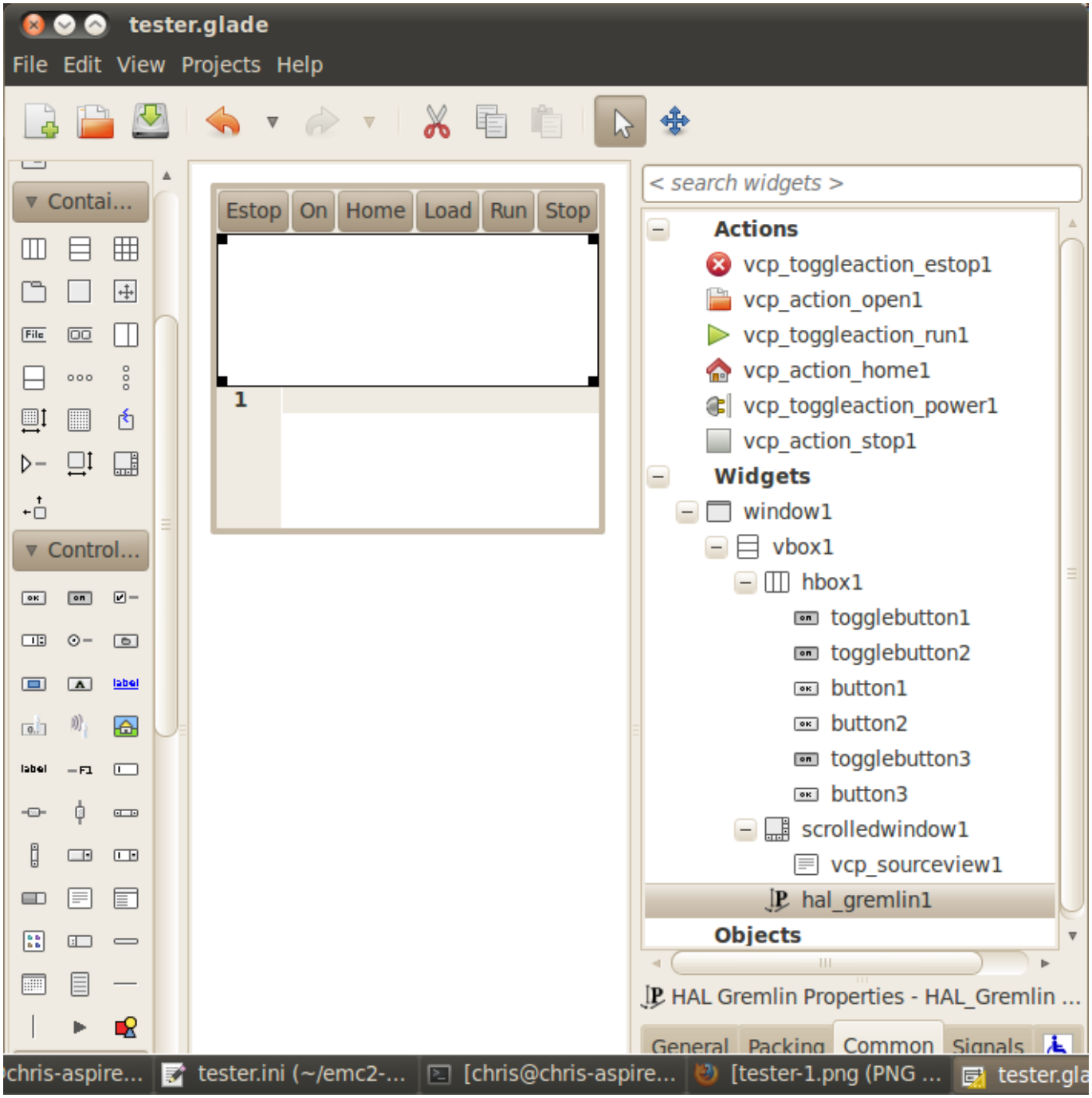


Давайте створимо простий зручний екран. Зробіть це в редакторі Glade (якщо використовуєте пакет RIP, запустіть його з терміналу після використання .scripts/rip-environment).

Що слід зазначити:

- Вікно верхнього рівня має мати ім’я за замовчуванням, «window1» — Gscreen покладається на це.
- Додайте дії, клацнувши правою кнопкою миші та вибравши «додати як віджет верхнього рівня». Вони не додають нічого візуального до вікна, але додаються до списку дій, розташованого праворуч. Додайте всі, які ви бачите вгорі праворуч.
- Після додавання дій нам потрібно пов’язати кнопки з діями, щоб вони працювали (див. нижче).
- Віджет gremlin не має розміру за замовчуванням, тому встановлення запитуваного розміру корисне (див. нижче).
- Віджет sourceview намагатиметься використати все вікно, тому додавання його до прокручуваного вікна охопить це. (Це вже зроблено в прикладі.)
- Кнопки розширюватимуться зі збільшенням вікна, що негарно, тому ми встановимо поле, в якому вони знаходяться, так, щоб воно не розширювалося (див. нижче).

- Типи кнопок, які слід використовувати, залежать від використовуваної VCP_action – наприклад, vcp_toggle_action зазвичай вимагає кнопок-перемикачів (поки що дотримуйтесь прикладу).
- Кнопки в цьому прикладі звичайні, а не кнопки HAL. Нам не потрібні контакти HAL.



На цьому екрані ми використовуємо VCP_actions для передачі LinuxCNC необхідних нам дій. Це дозволяє нам використовувати стандартні функції без додавання коду Python у файл обробника. Давайте пов’яжемо кнопку перемикання estop з дією estop Виберіть кнопку перемикання estop і на вкладці «Загальні» знайдіть «Пов’язана дія» та натисніть кнопку поруч із нею. Тепер виберіть дію перемикання estop. Тепер кнопка буде вмикати та вимикати estop при натисканні. На вкладці «Загальні» ви можете змінити текст мітки кнопки, щоб описати її функцію. Зробіть це для всіх кнопок.

Виберіть віджет gremlin, перейдіть на вкладку «Загальні» та встановіть потрібну висоту на 100, а потім поставте прапорець поруч із ним.

Клацніть горизонтальний блок, який містить кнопки. Перейдіть на вкладку упаковки та натисніть «розгорнути» на «Ні».

Збережіть його як tester.glade і збережіть у папці sim/gscreen/gscreen_custom/. Тепер запустіть LinuxCNC, натисніть sim/gscreen/gscreen_custom/tester і запустіть його. Якщо все йде добре, з'явиться наш екран і кнопки будуть працювати. Це працює, тому що tester.ini вказує gscreen шукати і завантажувати tester.glade і tester_handler.py. Файл tester_handler.py знаходиться в цій папці і кодований так, щоб просто показувати екран і не більше того. Оскільки спеціальні віджети безпосередньо спілкуються з LinuxCNC, ви все ще можете робити корисні речі. Якщо ваші потреби в екрані покриваються доступними спеціальними віджетами, то це все, що вам потрібно для створення екрану. Якщо ви хочете чогось більше, є ще багато хитрощів, починаючи від простого додавання «викликів функцій» для отримання стандартної поведінки. До написання власного коду Python для налаштування саме того, що ви хочете. Але це означає вивчення файлів обробників.

10.4.4 Приклад файлу обробника

Існують спеціальні функції, на наявність яких Gscreen перевіряє файл обробника. Якщо ви додасте їх до свого файлу обробника, Gscreen викликатиме їх замість внутрішніх однойменних функцій gscreen.

- initialize_preferences(self): Ви можете встановити нові процедури налаштувань.
- initialize_keybindings(self) Ви можете встановити нові процедури прив'язки клавіш. У більшості випадків вам не потрібно це робити, ви захочете замінити окремі виклики прив'язки клавіш. Ви також можете додати більше прив'язок клавіш, які будуть викликати довільну функцію.
- initialize_pins(self): створює / ініціалізує піни HAL
- connect_signals(self,handlers): Якщо ви використовуєте екран, який повністю відрізняється від стандартного Gscreen, ви повинні додати це, інакше gscreen буде намагатися підключити сигнали до віджетів, яких немає. Стандартна функція Gscreen викликається за допомогою self.gscreen.connect_signals(handlers). Якщо ви хочете просто додати додаткові сигнали до свого екрану, але все одно хочете використовувати стандартні, спочатку викличте цю функцію, а потім додайте більше сигналів. Якщо ваші сигнали прості (не передаються дані користувача), ви також можете використовувати вибір сигналів Glade в редакторі Glade.
- initialize_widgets(self): Ви можете використовувати це для налаштування будь-яких віджетів. Gscreen зазвичай викликає *self.gscreen.initialize_widgets()*, що фактично викликає багато окремих функцій. Якщо ви бажаєте включити деякі з цих віджетів, просто викличте ці функції безпосередньо. Або додайте self.gscreen.init_show_windows(), щоб віджети просто відображалися. Потім, якщо бажаєте, ініціалізуйте/налаштуйте свої нові віджети.
- initialize_manual_toolchange(self): дозволяє повністю оновити систему ручної зміни інструменту.
- set_restart_line(self.line):
- timer_interrupt(self): дозволяє повністю перевизначити процедуру переривання. Це використовуєть для виклику periodic() та перевірки на наявність помилок з linuxcnc.status.
- check_mode(self): використовується для перевірки режиму роботи екрана. Повертає список[] 0 - ручний режим 1 - mdi 2 - автоматичний режим 3 - повільний режим.
- on_tool_change(self,widget): Ви можете використовувати це для перевизначення діалогового вікна ручної зміни інструменту — воно викликається, коли *gscreen.tool-change* змінює стан.

- dialog_return(self,dialog_widget,displaytype,pinname): Використовуйте це для перевизначення будь-якого повідомлення користувача або діалогового вікна ручної зміни інструменту. Викликається коли діалогове вікно закрито.
- periodic(self): Викликається кожні (за замовчуванням 100) мілісекунд. Використовуйте його для оновлення ваших віджетів/контактів HAL. Ви також можете викликати Gscreen regular periodic після цього, self.gscreen.update_position() або просто додати pass, щоб нічого не оновлювати. Функція update_position() Gscreen насправді викликає багато окремих функцій. Якщо ви хочете включити деякі з цих віджетів, просто викличте ці функції безпосередньо.

Ви також можете додати власні функції для виклику в цьому файлі. Зазвичай ви додаєте сигнал до віджета для виклику вашої функції.

10.4.4.1 Додавання функцій комбінацій клавіш

Наш приклад тестувальника був би кориснішим, якби він реагував на команди клавіатури. Існує функція keybindings(), яка намагається це налаштувати. Хоча ви можете повністю її замінити, ми цього не зробили, але вона передбачає деякі речі:

- Припускається, що кнопка перемикання estop називається *button_estop*, і що нею керує клавіша F1.
- Припускається, що кнопка живлення називається «button_machine_on», і нею керує клавіша F2.

Це легко виправити, перейменувавши кнопки в редакторі Glade відповідно. Але замість цього ми перевизначимо стандартні виклики та додамо свої власні.

Додайте ці команди до файлу обробника:

```
# b''пb''b''eb''b''pb''b''eb''b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''йb'' Gscreen
# b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''иb'' b''кb''b''оb''b''мb''b''бb''b''ib''b' ↩
    'нb''b''ab''b''цb''b''ib''b''йb'' b''кb''b''лb''b''ab''b''вb''b''ib''b''шb''
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
```

Тепер ми перезаписали виклики функцій Gscreen з однаковою назвою і обробляємо їх у нашому файлі обробника. Тепер ми посилаємося на віджети за назвою, яку використовували в редакторі Glade. Ми також додали вбудовану функцію gscreen, щоб видавати звук при зміні Estop. Зверніть увагу, що для виклику вбудованих функцій Gscreen ми повинні використовувати self.gscreen.[FUNCTION NAME](). Якщо ми використали self.[FUNCTION NAME](), це викликало б функцію в нашому файлі обробника.

Додамо ще одне призначення клавіші, яке завантажує Halmeter при натисканні F4.

У файлі обробника в розділі *def initialize_widgets(self):* змініть на:

```
  def initialize_widgets(self):
    self.gscreen.init_show_windows()
    self.gscreen.keylookup.add_conversion('F4','TEST','on_keycall_HALMETER')
```

Потім додайте ці функції до класу *HandlerClass*:

```
  def on_keycall_HALMETER(self,state,SHIFT,CNTRL,ALT):
    if state:
      self.gscreen.on_halmeter()
      return True
```

Це додає перетворення комбінації клавіш, яке наказує gscreen викликати on_keycall_HALMETER при натисканні F4. Потім ми додаємо функцію до файлу дескрипторів, щоб викликати вбудовану функцію Gscreen для запуску halmeter.

10.4.4.2 Стан Linuxcnc

Модуль «Gstat» опитує стан LinuxCNC кожні 100 мс і надсилає повідомлення зворотного виклику до функцій користувача, коли стан змінюється. Ви можете зареєструвати повідомлення, щоб діяти при певних змінах стану. Як приклад, ми зареєструємося, щоб отримувати повідомлення «file-loaded», коли LinuxCNC завантажує новий файл. Спочатку ми повинні імпортувати модуль і створити його екземпляр: У розділі імпорту файлу обробника додайте:

```
from hal_glib import GStat
GSTAT = GStat()
```

У файлі обробника в розділі *def __init__(self):* додайте:

```
GSTAT.connect('file-loaded', self.update_filepath)
```

Потім у *HandlerClass* додайте функцію:

```
self.update_filepath(self, obj, path):
    self.widgets.my_path_label.set_text(path)
```

Коли LinuxCNC завантажує новий файл, Gstat надсилає повідомлення зворотного виклику до функції «update_filepath». У цьому прикладі ми оновлюємо мітку з цим іменем шляху (припускаючи, що у файлі GLADE є мітка з іменем «my_path_label»).

10.4.4.3 Клавіші для бігу

Немає спеціальних віджетів для керування кнопками на екрані, тому нам доведеться робити це за допомогою коду Python. У функції connect_signals додайте цей код:

```
        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)
```

Додайте ці функції до класу HandlerClass:

```
    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)
```

Нарешті, додайте до файлу GLADE дві кнопки для кожної осі — одну для позитивного, одну для негативного напрямку переміщення. Назвіть ці кнопки відповідно xneg, xpos, yneg, ypos zneg, zpos. Додайте до файлу GLADE віджет SpeedControl і назвіть його jog_speed.

10.4.5 Запуск Gscreen

Gscreen — це насправді просто інфраструктура для завантаження власного файлу GladeVCP та взаємодії з ним.

1. Gscreen зчитує параметри, з якими його було запущено.
2. Gscreen встановлює режим налагодження та встановлює додаткову назву скіна.
3. Gscreen перевіряє, чи є в папці конфігурації «локальні» XML-файли, файли обробників та/або файли локалізації. Він буде використовувати їх замість файлів за замовчуванням (у папці share/gscreen/skins/) (можуть відображатися два окремі екрани).
4. Головний екран завантажується, і переклади налаштовуються. Якщо присутній, буде завантажен другий екран і переклади налаштовано.
5. Додатковий звук ініціалізується, якщо він доступний.
6. Він зчитує частину INI-файлу для ініціалізації одиниць вимірювання та кількості/типу осей.
7. Ініціалізує прив’язку Python до HAL для створення компонента, що не працює в реальному часі, з назвою Gscreen.
8. Викликається функція makepins GladeVCP для розбору XML-файлу з метою створення HAL-пінів для віджетів HAL та реєстрації підключених віджетів LinuxCNC.
9. Перевіряє наявність «локального» файлу обробника в папці конфігурації або використовує стандартний файл з папки скіна.
10. Якщо є файл-обробник, gscreen аналізує його та реєструє виклики функцій у просторі імен Gscreen.
11. Glade зіставляє/реєструє всі виклики сигналів з функціями в gscreen та файлі обробника.
12. Gscreen перевіряє INI-файл на наявність назви файлу налаштувань опцій, інакше використовуєтьс *.gscreen_preferences* =.
13. Gscreen перевіряє, чи є виклик функції налаштувань (*initialize_preferences(self)*) у файлі обробника, інакше він використовує стандартну функцію Gscreen.
14. Gscreen перевіряє наявність компонента ClassicLadder у реальному часі.
15. Gscreen перевіряє наявність загальносистемної теми GTK.
16. Gscreen збирає прирости бігової швидкості з INI-файлу.
17. Gscreen збирає значення кутового приросту штовхання з INI-файлу.
18. Gscreen збирає значення за замовчуванням та максимальну швидкість поштовху з INI.
19. Gscreen збирає дані про максимальну швидкість будь-яких осей з розділу TRAJ INI.
20. Gscreen перевіряє наявність кутових осей, а потім збирає значення швидкості за замовчуванням та максимальної швидкості з INI-файлу.

21. Gscreen збирає всі налаштування перевизначення з INI.
22. Gscreen перевіряє, чи це конфігурація токарного верстата з INI-файлу.
23. Gscreen знаходить назву tool_table, tool editor та param файлу з INI.
24. Gscreen перевіряє файл обробника на наявність функції прив'язок клавіш (*initialize_keybindings(se*
або ж використовує стандартну функцію Gscreen.
25. Gscreen перевіряє файл обробника на наявність функції пінів (*initialize_pins(self)*) або використову
стандартну Gscreen.
26. Gscreen перевіряє файл обробника на наявність функції manual_toolchange (*initialize_manual_toolc*
або використовує стандартну версію Gscreen.
27. Gscreen перевіряє файл обробника на наявність функції connect_signals (*initialize_connect_signals(*
або ж використовує стандартну версію Gscreen.
28. Gscreen перевіряє файл обробника для функції віджетів (*initialize_widgets(self)*) або ж використову
стандартну версію Gscreen.
29. Gscreen налаштовує повідомлення, зазначені у файлі INI.
30. Gscreen повідомляє HAL, що компонент HAL Gscreen завершив створення пінів і готовий до роботи. Якщо на екрані є віджет терміналу, він виведе на нього всі піни Gscreen.
31. Gscreen встановлює час циклу відображення на основі INI-файлу.
32. Gscreen перевіряє файл обробника на наявність виклику функції *timer_interupt(self)*, інакше використовується виклик функції Gscreen за замовчуванням.

10.4.6 Налаштування INI

Під заголовком [DISPLAY]:

```
DISPLAY = gscreen -c tester
  options:
   -d debugging on
   -v verbose debugging on
```

Перемикач -c дозволяє вибрати «шкіру». Gscreen припускає, що файл Glade і файл обробника мають однакову назву. Опціональний другий екран матиме таку саму назву з додаванням цифри 2 (наприклад, tester2.glade). Другий файл обробника не допускається. Він буде завантажений тільки в тому випадку, якщо він присутній. Gscreen шукатиме файли спочатку у файлі конфігурації LinuxCNC, який був запущений першим, а потім у системній папці скінів.

10.4.7 Повідомлення діалогового вікна користувача

Ця функція використовується для відображення спливаючих діалогових повідомлень на екрані. Вони визначені у файлі INI та контролюються контактами HAL:

MESSAGE_BOLDTEXT
: зазвичай це титул.

MESSAGE_TEXT
: нижче цього і зазвичай довше.

MESSAGE_DETAILS
: приховано, якщо на нього не натиснути.

MESSAGE_PINNAME
– це базова назва контактів HAL.

MESSAGE_TYPE
вказує, чи це повідомлення «так/ні», «добре» чи статусне повідомлення

- Повідомлення про стан
  - буде відображатися в рядку стану та діалоговому вікні сповіщень,
  - не потребують втручання користувача.
- повідомлення
  - вимагати від користувача натискання кнопки «OK», щоб закрити діалогове вікно.
  - мати один контакт HAL для запуску діалогу та один для позначення очікування відповіді.
- повідомлення «так/ні»
  - вимагати від користувача натискання кнопок «так» або «ні» для закриття діалогового вікна.
  - мають три виводи HAL:
    1. один для відображення діалогу,
    2. один для очікування, і
    3. один за відповідь.

Ось приклад INI-коду. Він буде під заголовком [DISPLAY].

```
# b''Цb''b''eb'' b''пb''b''рb''b''оb''b''сb''b''тb''b''оb'' b''вb''b''іb''b''дb''b''оb''b' ↩
    'бb''b''рb''b''аb''b''жb''b''аb''b''єb''b''тb''b''ьb''b''сb''b''яb'' b''вb'' b''рb''b' ↩
    'яb''b''дb''b''кb''b''уb'' b''сb''b''тb''b''аb''b''нb''b''уb'' b''тb''b''аb'' b''сb''b' ↩
    'пb''b''лb''b''иb''b''вb''b''аb''b''юb''b''чb''b''оb''b''мb''b''уb'' b''вb''b''іb''b' ↩
    'кb''b''нb''b''іb'' b''сb''b''пb''b''оb''b''вb''b''іb''b''щb''b''eb''b''нb''b''ьb'' b' ↩
    'нb''b''аb'' b''рb''b''оb''b''бb''b''оb''b''чb''b''оb''b''мb''b''уb'' b''сb''b''тb''b' ↩
    'оb''b''лb''b''іb''.
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

# b''Зb'''b''яb''b''вb''b''иb''b''тb''b''ьb''b''сb''b''яb'' b''дb''b''іb''b''аb''b''лb''b' ↩
    'оb''b''гb''b''оb''b''вb''b''eb'' b''вb''b''іb''b''кb''b''нb''b''оb'' b''зb'' b''пb''b' ↩
    'иb''b''тb''b''аb''b''нb''b''нb''b''яb''b''мb'' «b''тb''b''аb''b''кb'' b''іb'' b''нb''b' ↩
    'іb''»
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest

# b''Цb''b''eb'' b''зb''b'''b''b''яb''b''вb''b''иb''b''тb''b''ьb'' b''дb''b''іb''b''аb''b' ↩
    'лb''b''оb''b''гb''b''оb''b''вb''b''eb'' b''вb''b''іb''b''кb''b''нb''b''оb'', b''яb''b' ↩
    'кb''b''eb'' b''вb''b''иb''b''мb''b''аb''b''гb''b''аb''b''єb'' b''вb''b''іb''b''дb''b' ↩
    'пb''b''оb''b''вb''b''іb''b''дb''b''іb'' «b''Оb''b''Кb''», b''іb'' b''вb''b''оb''b''нb'' ↩
    b''оb'' b''вb''b''іb''b''дb''b''оb''b''бb''b''рb''b''аb''b''зb''b''иb''b''тb''b''ьb''b' ↩
    'сb''b''яb'' b''вb'' b''рb''b''яb''b''дb''b''кb''b''уb'' b''сb''b''тb''b''аb''b''нb''b' ↩
    'уb'' b''тb''b''аb''
# b''уb'' b''сb''b''пb''b''лb''b''иb''b''вb''b''аb''b''юb''b''чb''b''оb''b''мb''b''уb'' b' ↩
    'вb''b''іb''b''кb''b''нb''b''іb'' b''сb''b''пb''b''оb''b''вb''b''іb''b''щb''b''eb''b' ↩
    'нb''b''ьb'' b''нb''b''аb'' b''рb''b''оb''b''бb''b''оb''b''чb''b''оb''b''мb''b''уb'' b' ↩
    'сb''b''тb''b''оb''b''лb''b''іb''.
MESSAGE_BOLDTEXT = b''Цb''b''eb'' b''кb''b''оb''b''рb''b''оb''b''тb''b''кb''b''иb''b''йb''  ↩
    b''тb''b''eb''b''кb''b''сb''b''тb''
```

```
TEXT_MESSAGE = b''Цb''b''eb'' b''дb''b''ob''b''вb''b''шb''b''иb''b''йb'' b''тb''b''eb''b' ↩
    'кb''b''cb''b''тb'' b''тb''b''eb''b''cb''b''тb''b''уb'' b''ob''b''бb''b''ob''b''хb'' b' ↩
    'тb''b''иb''b''пb''b''ib''b''вb''. b''Bb''b''ib''b''нb'' b''мb''b''ob''b''жb''b''eb'' b' ↩
    'бb''b''уb''b''тb''b''иb'' b''дb''b''ob''b''вb''b''шb''b''иb''b''мb'' b''зb''b''аb'' b' ↩
    'тb''b''eb''b''кb''b''cb''b''тb'' b''уb'' b''pb''b''яb''b''дb''b''кb''b''уb'' b''cb''b' ↩
    'тb''b''аb''b''нb''b''уb''
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
```

10.4.7.1 Скопіюйте файл обробника Stock/Glade для модифікації

Якщо ви хочете використовувати стандартний екран, але змінити його файл обробки, вам потрібно скопіювати стандартний файл у папку конфігураційних файлів. Gscreen побачить це і буде використову скопійований файл. Але де знаходиться оригінальний файл? Якщо ви використовуєте RIP LinuxCNC, зразки скінів знаходяться в /share/gscreen/skins/*SCREENNAME*. Встановлені версії LinuxCNC мають їх у дещо інших місцях, залежно від використовуваної дистрибуції. Простий спосіб знайти розташування — відкрити термінал і запустити екран sim, який ви хочете використовувати. У терміналі буде надруковано розташування файлів. Може бути корисно додати перемикач -d до рядка завантаження gscreen в INI.

Ось зразок:

```
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']
```

Лінія:

```
**** GSCREEN INFO: b''шb''b''лb''b''яb''b''хb'' b''дb''b''ob'' b''фb''b''аb''b''йb''b''лb'' ↩
    b''уb'' b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''аb'': ['/home/chris/ ↩
    emc-dev/share/gscreen/skins/industrial/industrial_handler.py']
```

показує, де знаходиться файл зі стандартними файлами. Скопіюйте цей файл у папку конфігурації. Це працює так само для файлу Glade.

10.5 QtDragon GUI

10.5.1 Вступ

QtDragon і QtDragon_hd побудовані на базі фреймворку QtVCP. Це творча ідея учасника форуму Persei8. Значна частина базується на чудовій роботі інших учасників спільноти LinuxCNC. Версія LinuxCNC адаптована з версій Persei8 на Github. Вона в першу чергу призначена для 3-5-осьових верстатів, таких як фрезерні верстати або маршрутизатори. Вона добре працює з сенсорним екраном та/або мишею. QtDragon підтримує кілька способів дотику до інструментів та зондування деталей. Ви можете використовувати функцію зовнішніх зміщень LinuxCNC для автоматичного підйому шпинделя під час паузи. Якщо ви маєте опцію VersaProbe та код перемапування, ви можете додати автоматичне зондування довжини інструменту під час зміни інструменту.

Note
QtDragon і QtVCP — це відносно нові програми, додані до LinuxCNC. Можливі помилки та несподіванки. Будь ласка, ретельно тестуйте їх під час використання небезпечного обладнання. Будь ласка, надсилайте звіти на форум або в список розсилки.

10.5.1.1 QtDragon

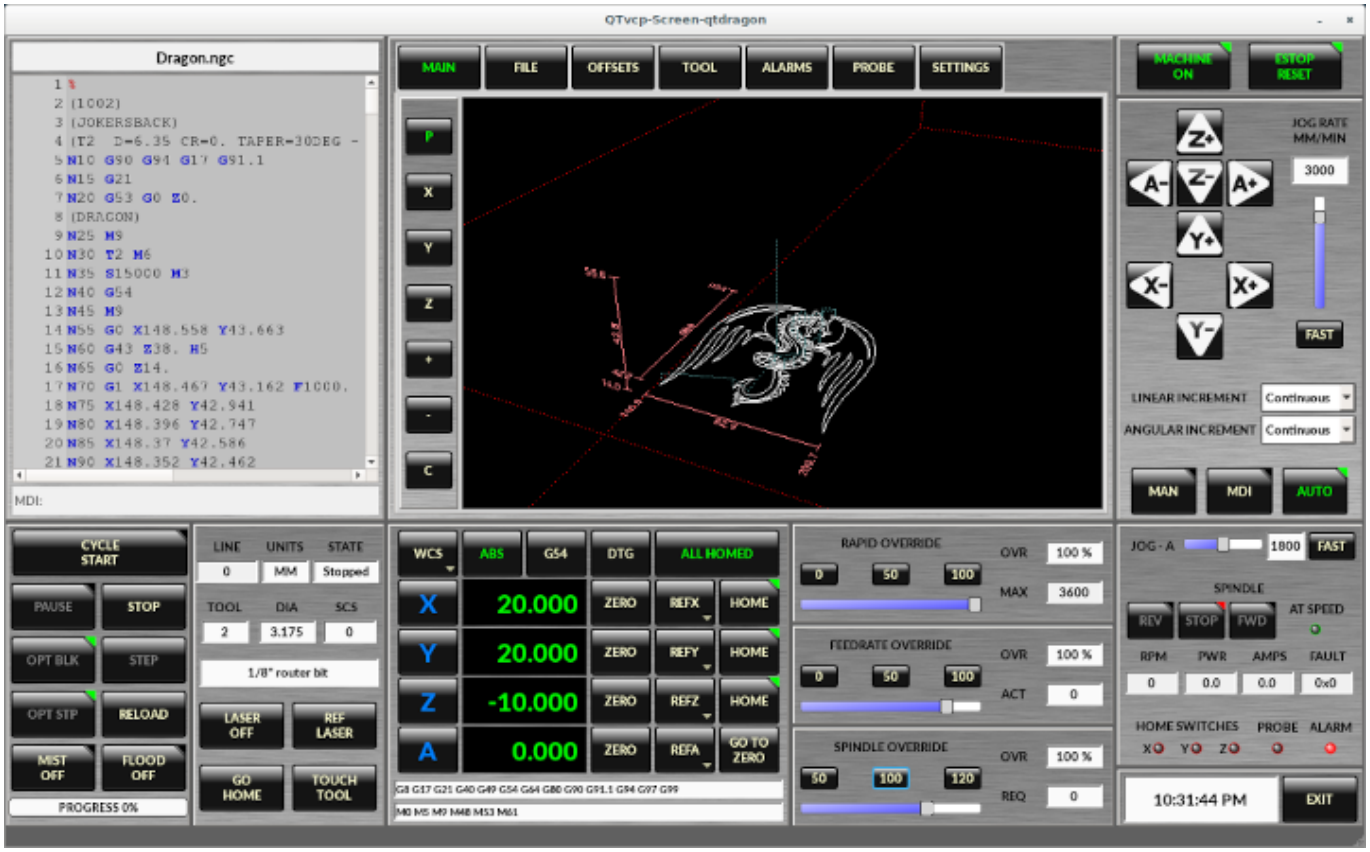
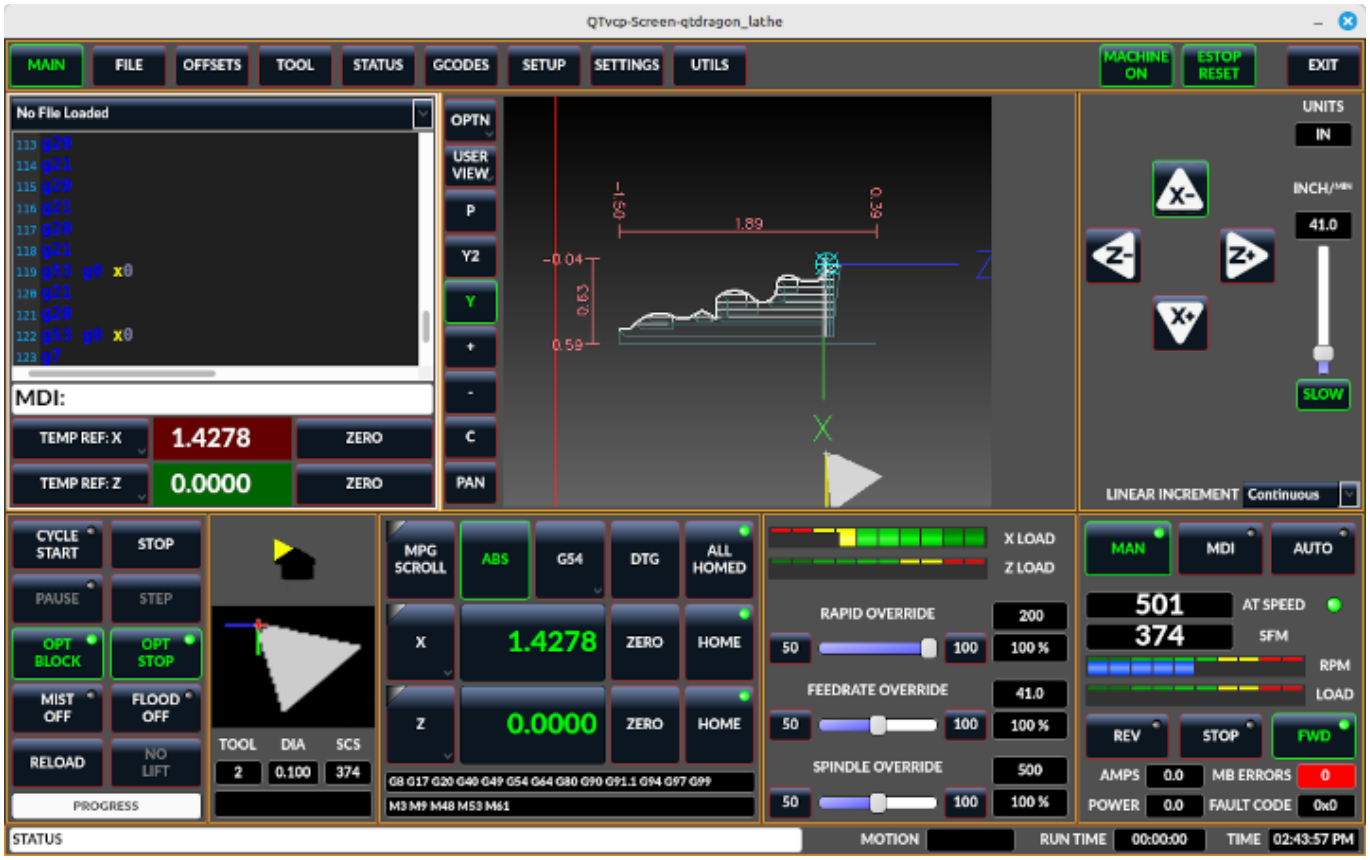


Figure 10.26: QtDragon - Зразок для 3–5 осей (1440x860) у срібній темі

Роздільна здатність QtDragon можна змінювати від 1280x768 до 1680x1200. Він працюватиме у віконному режимі на будь-якому моніторі з вищою роздільною здатністю, але не на моніторах з нижчою роздільною здатністю.

10.5.1.2 QtDragon_lathe



QtDragon_lathe — це модифікована версія QtDragon, яка більше підходить для токарних верстатів.
Розмір вікна можна змінювати від роздільної здатності 1280x768 до 1680x1200.
Програма працюватиме у віконному режимі на будь-якому моніторі з вищою роздільною здатністю, але не на моніторах з нижчою роздільною здатністю.

10.5.1.3 QtDragon_hd

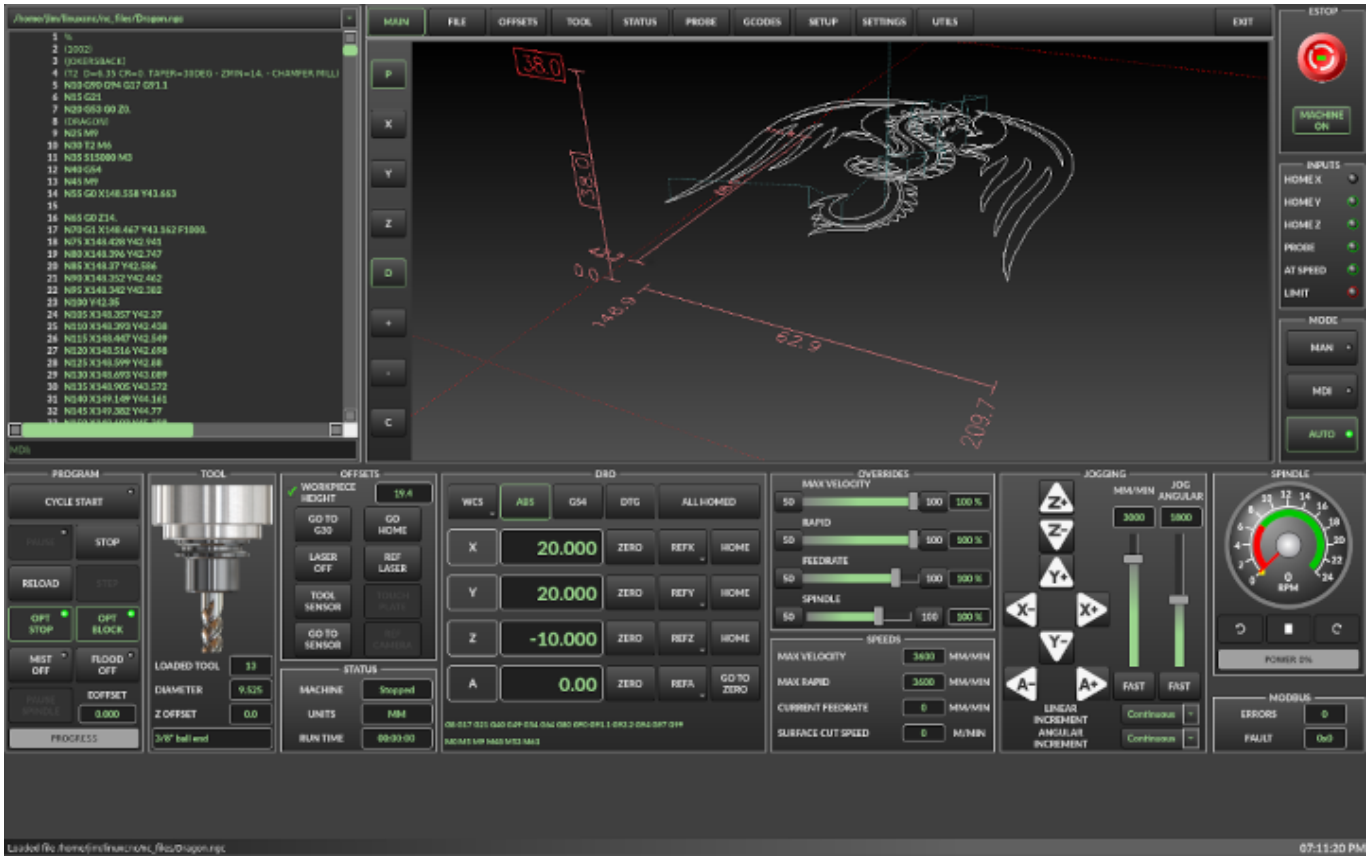


Figure 10.27: QtDragon_hd - зразок для 3-5 осей на більших моніторах (1920x1056) у темній темі

QtDragon_hd має схожий дизайн до QtDragon, але модифікований для використання додаткового простору сучасних більших моніторів. Існують деякі невеликі відмінності в компонуванні та функціональності.

QtDragon_hd має роздільну здатність 1920x1056 і не може бути змінений у розмірі. Він працюватиме у віконному режимі на будь-якому моніторі з вищою роздільною здатністю, але не на моніторах з нижчою роздільною здатністю.

10.5.1.4 QtDragon_hd_vertical

QtDragon_hd_vertical — це вертикально орієнтована версія. Її розмір не можна змінювати.

10.5.2 Початок роботи - INI-файл

Якщо ваша конфігурація наразі не налаштована на використання QtDragon, ви можете змінити її, редагуючи розділи файлу INI. Вичерпний перелік опцій див. у розділі display section документації до файлу INI.

Note
У файлі INI може бути лише один розділ кожного розділу (наприклад, [HAL]). Якщо в цій документації ви бачите кілька варіантів розділів, розмістіть їх усі під однією відповідною назвою розділу.

10.5.2.1 Дисплей

У розділі `[DISPLAY]` змініть присвоєння `DISPLAY =` на таке:

- `qtdragon` для маленької версії
- `qtdradon_hd` для великої версії.

Ви можете додати `-v`, `-d`, `-i` або `-q` для (відповідно) детального, налагоджувального, інформаційного або тихого виводу в термінал.

```
[DISPLAY]
DISPLAY = qtvcp qtdragon
```

10.5.2.2 Параметри

Щоб відстежувати налаштування, QtDragon шукає текстовий файл налаштувань. Додайте наступний запис під заголовком `[DISPLAY]`.
Він може використовувати ~ для домашнього каталогу або `WORKINGFOLDER` чи `CONFIGFOLDER`, щоб представити уявлення QtVCP про ці каталоги:
У цьому прикладі файл буде збережено в папці конфігурації екрана запуску. (Можливі й інші варіанти, див. документацію до віджету screenoption QtVCP.)

```
[DISPLAY]
PREFERENCE_FILE_PATH = WORKINGFOLDER/qtdragon.pref
```

10.5.2.3 Лісозаготівля

Ви можете вказати, де зберігати історію/журнали.
Ці імена файлів може вибрати користувач.
У розділі `[DISPLAY]` додайте:

```
[DISPLAY]
MDI_HISTORY_FILE = mdi_history.dat
MACHINE_LOG_PATH = machine_log.dat
LOG_FILE = qtdragon.log
```

10.5.2.4 Перевизначення елементів керування

Ці параметри встановлюють елементи керування перевизначенням qtdragon (1.0 = 100 відсотків):

```
[DISPLAY]
MAX_SPINDLE_0_OVERRIDE = 1.5
MIN_SPINDLE_0_OVERRIDE = .5
MAX_FEED_OVERRIDE       = 1.2
```

10.5.2.5 Шпиндельні елементи керування

Налаштування керування шпинделем (в об/хв та ватах):

```
[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 Інкременти бігу підтюпцем

Встановіть вибрані кроки штовхання.
Ці кроки може змінити користувач.

```
[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 Приріст сітки

Встановіть доступні опціональні розміри сітки для графічного дисплея.
Це замінить розміри за замовчуванням.
Для вказання одиниць виміру використовуються «мм» та «дюйми».
Поле вибору сітки відображається після натискання кнопки «OPTN» на графічному дисплеї.

```
[DISPLAY]
GRIDS = 0, .1 mm, 1 mm, 2 mm, 5 mm, 10 mm, .25 in, .5 in
```

10.5.2.8 Швидкість штовхання

Встановлення швидкості штовхання (в одиницях за секунду)

```
[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 Система діалогів повідомлень користувача

Додаткові спливаючі діалогові вікна з налаштованими повідомленнями, що керуються контактами HAL.
MESSAGE_TYPE може бути «okdialog» або «yesnodialog». Дивіться `qtvcp/library/messages` для отримання додаткової інформації.
Цей приклад показує, як створити діалогове вікно, яке вимагає від користувача вибрати «ok» для підтвердження та приховування.
Ці діалогові вікна можуть використовуватися для таких речей, як попередження про низький рівень мастила тощо.

```
[DISPLAY]
MESSAGE_BOLDTEXT = b''Цb''b''eb'' b''кb''b''ob''b''pb''b''ob''b''тb''b''кb''b''иb''b''йb''  ↩
    b''тb''b''eb''b''кb''b''cb''b''тb''
MESSAGE_TEXT = b''Цb''b''eb'' b''дb''b''ob''b''вb''b''шb''b''иb''b''йb'' b''тb''b''eb''b' ↩
    'кb''b''cb''b''тb'' b''тb''b''eb''b''cb''b''тb''b''уb'' b''ob''b''бb''b''ob''b''хb'' b' ↩
    'тb''b''иb''b''пb''b''ib''b''вb''. b''Bb''b''ib''b''нb'' b''мb''b''ob''b''жb''b''eb'' b' ↩
    'бb''b''уb''b''тb''b''иb'' b''дb''b''ob''b''вb''b''шb''b''иb''b''мb'' b''зb''b''аb'' b' ↩
    'тb''b''eb''b''кb''b''cb''b''тb'' b''pb''b''яb''b''дb''b''кb''b''аb'' b''cb''b''тb''b' ↩
    'аb''b''нb''b''уb''
MESSAGE_DETAILS = BOTH DETAILS
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = oktest
```

Мультиповідомлення використовують контакт HAL s32 для відображення кількох визначених повідомлень.

```
[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 Вбудувати власні панелі VCP

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

Приклад додавання вбудованої панелі до вкладки утиліт, тобто графічної анімованої машини з використанням бібліотеки vismach.

```
[DISPLAY]
EMBED_TAB_NAME = Vismach demo
EMBED_TAB_COMMAND = qtvcp vismach_mill_xyz
EMBED_TAB_LOCATION = tabWidget_utilities
```

Ця панель-приклад призначена для відображення додаткових даних частотного перетворювача RS485, а також для налаштування шпиндельного приводу з 4 шківами та 2 ременями за допомогою серії кнопок.




```
[DISPLAY]
EMBED_TAB_NAME = Spindle Belts
EMBED_TAB_COMMAND = qtvcp spindle_belts
EMBED_TAB_LOCATION = tabWidget_utilities
```

10.5.2.11 Шляхи підпрограм

Якщо ви використовуєте NGCGUI, перепризначені або власні процедури M-кодів, LinuxCNC повинен знати, де шукати файли.
Цей приклад є типовим для того, що потрібно для NgcGui, Basic Probe та Versa Probe перепризначеного коду.
Ці шляхи потрібно буде налаштувати, щоб вони вказували на фактичні файли у вашій системі.
RS274NZGC Деталі розділу

```
[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
```

Програма NGCGUI від QtVCP також повинна знати, де відкривати підпрограми для вибору та попереднього вибору.
NGCGUI_SUBFILE_PATH повинен вказувати на фактичний шлях у вашій системі, а також на шлях, описаний у SUBROUTINE_PATHS.

```
[DISPLAY]
# b''Шb''b''лb''b''яb''b''хb'' b''дb''b''оb'' b''пb''b''іb''b''дb''b''пb''b''pb''b''оb''b' ↩
    'гb''b''pb''b''аb''b''мb''b''иb'' NGCGUI.
```

```
# b''Шb''b''лb''b''яb''b''xb'' Thr b''тb''b''ab''b''кb''b''ob''b''жb'' b''мb''b''ab''b' ↩
    'єb'' b''бb''b''yb''b''тb''b''иb'' b''вb'' [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# b''вb''b''кb''b''лb''b''ab''b''дb''b''кb''b''иb'' b''пb''b''ob''b''пb''b''eb''b''pb''b' ↩
    'eb''b''дb''b''нb''b''ьb''b''ob'' b''вb''b''иb''b''бb''b''pb''b''ab''b''нb''b''иb''b' ↩
    'xb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''
# b''вb''b''кb''b''ab''b''зb''b''yb''b''йb''b''тb''b''eb'' b''лb''b''иb''b''шb''b''eb'' b' ↩
    'ib''b''мb''b''eb''b''нb''b''ab'' b''фb''b''ab''b''йb''b''лb''b''ib''b''вb'', b''фb''b' ↩
    'ab''b''йb''b''лb''b''иb'' b''пb''b''ob''b''вb''b''иb''b''нb''b''нb''b''ib'' b''бb''b' ↩
    'yb''b''тb''b''иb'' b''вb'' NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

10.5.2.12 Контроль попереднього перегляду

Для керування попереднім переглядом G-коду можна використовувати магічні коментарі.

У дуже великих програмах завантаження попереднього перегляду може зайняти багато часу. Ви можете контролювати, що відображається, а що приховується на графічному екрані, додавши відповідні коментарі з цього списку до вашого G-коду:

```
(PREVIEW,stop)
(PREVIEW,hide)
(PREVIEW,show)
```

10.5.2.13 Розширення/фільтри програми

Ви можете контролювати, які програми відображаються у вікні файлового менеджера, за допомогою розширень програм.
Створіть рядок із закінченнями «.», які ви хочете використовувати, розділивши їх комами, а потім пробілом і описом.
Ви можете додати кілька рядків для різних варіантів вибору в комбінованому полі.

```
[FILTER]
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code file (*.ngc,*.nc,*.tap)
```

QtDragon має можливість надсилати завантажені файли через «програму фільтрації». Цей фільтр може виконувати будь-яке бажане завдання: від такого простого, як перевірка, чи файл закінчується на «M2», до такого складного, як генерація G-коду з зображення. Дивіться Filter Programs для отримання додаткової інформації.

Розділ «[FILTER]» файлу INI контролює роботу фільтрів. Спочатку для кожного типу файлу напишіть рядок «PROGRAM_EXTENSION». Потім вкажіть програму, яку потрібно виконати для кожного типу файлу. Ця програма отримує ім'я вхідного файлу як перший аргумент і повинна записувати код rs274ngc у стандартний вивід. Цей вивід буде відображатися в текстовій області, попередньо переглядатися в області відображення та виконуватися LinuxCNC при натисканні «Run».

Наступні рядки додають підтримку конвертера `image-to-gcode`, що входить до комплекту LinuxCNC та запускає програми фільтрації на основі Python:

```
[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 Налаштування зонда/сенсорної панелі/лазера

QtDragon має INI-файли для двох додаткових вкладок зондування. Коментуйте/розкоментуйте, як вам зручніше.

- «Versa probe» — це портована на QtVCP версія популярної панелі зондування GladeVCP.
- «Базовий зонд» — це портована на QtVCP версія, заснована на екрані базового зонда стороннього розробника.

Обидва виконують подібні процедури зондування, хоча зонд Versa опціонально виконує автоматичне вимірювання інструменту.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

10.5.2.15 Виявлення переривання

При використанні процедур зондування qtdragon важливо виявляти запити користувача на перериван
За замовчуванням LinuxCNC не повідомляє про переривання у спосіб, корисний для процедур зондування.
Необхідно додати файл ngc, щоб вивести на екран помилку, яку можна виявити. Перепризначення деталей переривання

```
[RS274NGC]
# b''Пb''b''pb''b''иb'' b''пb''b''eb''b''pb''b''eb''b''pb''b''иb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''ib'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''тb''b''ьb''b' ↩
    'cb''b''яb'' b''цb''b''eb''b''йb'' ngc-b''фb''b''ab''b''йb''b''лb''. b''Пb''b''ob''b' ↩
    'тb''b''pb''b''ib''b''бb''b''нb''b''ob'' b''дb''b''лb''b''яb'' b''бb''b''ab''b''зb''b' ↩
    'ob''b''вb''b''иb''b''xb''/Versa-b''пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b' ↩
    'pb''b''ab''b''мb'' b''зb''b''ob''b''нb''b''дb''b''уb''b''вb''b''ab''b''нb''b''нb''b' ↩
    'яb''. +
ON_ABORT_COMMAND=O <on_abort> call
```

Цей приклад коду надсилає повідомлення про переривання. Програми-зонди можуть виявити цей зразок.
Відповідно до налаштувань, наведених вище, його потрібно зберегти як «on_abort.ngc» у шляхах пошуку LinuxCNC [RS274NGC] SUBROUTINE_PATHS та [DISPLAY] PROGRAM_PREFIX.

```
o<on_abort> sub

o100 b''яb''b''кb''b''щb''b''ob'' [#1 b''дb''b''ob''b''pb''b''ib''b''вb''b''нb''b''юb''b' ↩
    'єb'' 5]
    (b''мb''b''ab''b''шb''b''иb''b''нb''b''ab'' b''уb''b''вb''b''ib''b''мb''b''кb''b''нb''b ↩
        ''eb''b''нb''b''ab'')
o100 b''ib''b''нb''b''ab''b''кb''b''шb''b''eb'' b''яb''b''кb''b''щb''b''ob'' [#1 b''дb''b' ↩
    'ob''b''pb''b''ib''b''вb''b''нb''b''юb''b''єb'' 6]
    (b''мb''b''ab''b''шb''b''иb''b''нb''b''ab'' b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b ↩
        ''нb''b''ab'')
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 Коди запуску

Вам слід встановити код M/G за замовчуванням для запуску. Він буде замінений запуском файлу NGC.
Це лише зразки кодів, інтегратор повинен вибрати відповідні коди.

```
[RS274NGC]
# b''зb''b''ab''b''пb''b''уb''b''cb''b''кb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''дb''b' ↩
    'иb'' G/M b''пb''b''іb''b''дb'' b''чb''b''ab''b''cb'' b''пb''b''eb''b''рb''b''шb''b' ↩
    'ob''b''гb''b''ob'' b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b' ↩
    'нb''b''нb''b''яb''
RS274NGC_STARTUP_CODE = G17 G20 G40 G43H0 G54 G64P0.0005 G80 G90 G94 G97 M5 M9
```

10.5.2.17 Кнопки макросів

QtDragon має до десяти зручних кнопок для виклику «макро-дій».
Вони знаходяться під заголовком «[MDI_COMMAND_LIST]» як «MDI_COMMAND_MACRO0 =» до «MDI_COMMAND_MACRO9 =».
За бажанням вони також можуть викликати процедури OWord.
У зразках конфігурацій вони позначені для переміщення між поточною системою користувача (нульовою точкою) і системою верстата.
Система користувача є першою командою MDI в списку INI, система верстата — другою.
Цей приклад показує, як спочатку перемістити вісь Z вгору. Команди, розділені символом «;», виконуються одна за одною.
Текст мітки кнопки можна встановити будь-яким текстом після коми, символ «\n» додає розрив рядка.

```
[MDI_COMMAND_LIST]
# b''дb''b''лb''b''яb'' b''мb''b''ab''b''кb''b''pb''b''ob''b''кb''b''нb''b''ob''b''пb''b' ↩
    'ob''b''кb''
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 Файл HAL для публікації графічного інтерфейсу

Ці опціональні HAL-файли будуть викликані після того, як QtDragon завантажить все інше.
Ви можете додати кілька рядків для декількох файлів. Кожен з них буде викликаний у порядку їх появи.
Виклик HAL-файлів після завантаження QtDragon гарантує, що HAL-контакти QtDragon будуть доступні.

Приклад із типовими записами для специфікації HAL-файлів, які потрібно зчитати після запуску QtDragon. Відредагуйте ці рядки відповідно до фактичних вимог.

```
[HAL]
POSTGUI_HALFILE = qtdragon_hd_postgui.hal
POSTGUI_HALFILE = qtdragon_hd_debugging.hal
```

10.5.2.19 Команда HAL після графічного інтерфейсу

Ці опціональні команди HAL будуть виконані після того, як QtDragon завантажить все інше.
Ви можете додати кілька рядків. Кожен з них буде викликаний у порядку їх появи.
Можна використовувати будь-яку команду HAL.

Зробіть зразок типових файлів у INI-файлі для завантаження модулів після того, як графічний інтерфейс стане доступним. Налаштуйте їх відповідно до ваших фактичних вимог.

```
[HAL]
POSTGUI_HALCMD = loadusr qtvcp test_probe
POSTGUI_HALCMD = loadusr qtvcp test_led
POSTGUI_HALCMD = loadusr halmeter
```

10.5.2.20 Міст HAL

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.

- Він може повідомляти/змінювати поточну вибрану кнопку Осі.
- Буде повідомлено про швидкість/приріст поштовхового переміщення.
- Є контакт запуску та паузи циклу — вони викликають код у QtDragon, а не контролер руху.
- Якщо в INI визначено макроси, будуть доступні піни для їх ініціювання.

Приклад запису. Вилучити -d, щоб заглушити повідомлення налагодження.

```
[HAL]
HALBRIDGE= hal_bridge -d
```

Типові доступні HAL-піни:

```
b''Bb''b''ив''b''вв''b''об''b''дв''b''ив'' b''кв''b''об''b''мв''b''пв''b''об''b''нв''b' ↩
    'еb''b''нв''b''тв''b''ib''b''вв'':
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 Вбудовані зразки конфігурацій

Зразки конфігурацій `sim/qtdragon/` або `sim/qtdragon_hd` вже налаштовані для використання QtDragon як екрана. Існує кілька прикладів, що демонструють різні конфігурації машин.

10.5.3 Прив’язки клавіш

QtDragon не призначений для використання клавіатури в якості основного засобу управління машиною.
У ньому відсутні багато клавіатурних скорочень, які, наприклад, є в AXIS, але ви можете використовува мишу або сенсорний екран.
Для зручності управління машиною існує кілька комбінацій клавіш.

```
F1 - b’’Уb’’b’’вb’’b’’іb’’b’’мb’’b’’кb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’/b’’вb’’b’’иb’’b ↩
    ’’мb’’b’’кb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ Estop
F2 - b’’Уb’’b’’вb’’b’’іb’’b’’мb’’b’’кb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’/b’’вb’’b’’иb’’b ↩
    ’’мb’’b’’кb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’мb’’b’’аb’’b’’шb’’b’’иb’’b’’нb’’b’ ↩
    ’иb’’
F12 - b’’Рb’’b’’еb’’b’’дb’’b’’аb’’b’’кb’’b’’тb’’b’’оb’’b’’рb’’ b’’сb’’b’’тb’’b’’иb’’b’’лb’’ ↩
    b’’іb’’b’’вb’’
Home - b’’Гb’’b’’оb’’b’’лb’’b’’оb’’b’’вb’’b’’нb’’b’’аb’’ b’’сb’’b’’тb’’b’’оb’’b’’рb’’b’ ↩
    ’іb’’b’’нb’’b’’кb’’b’’аb’’ b’’вb’’b’’сb’’b’’іb’’b’’хb’’ b’’зb’’’b’’єb’’b’’дb’’b’’нb’’b’ ↩
    ’аb’’b’’нb’’b’’ьb’’ b’’мb’’b’’аb’’b’’шb’’b’’иb’’b’’нb’’b’’иb’’
Escape - b’’Пb’’b’’еb’’b’’рb’’b’’еb’’b’’рb’’b’’иb’’b’’вb’’b’’аb’’b’’нb’’b’’нb’’b’’яb’’ b’ ↩
    ’рb’’b’’уb’’b’’хb’’b’’уb’’
Pause - b’’Пb’’b’’рb’’b’’иb’’b’’зb’’b’’уb’’b’’пb’’b’’иb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’ ↩
    ’яb’’ b’’рb’’b’’уb’’b’’хb’’b’’уb’’ b’’мb’’b’’аb’’b’’шb’’b’’иb’’b’’нb’’b’’иb’’
```

10.5.4 Кпонки

Кнопки, які можна позначити, змінять колір тексту після встановлення позначки. Це контролюється таблицею стилів/темою

10.5.5 Віртуальна клавіатура

QtDragon включає віртуальну клавіатуру для використання з сенсорними екранами. Щоб увімкнути клавіатуру, встановіть прапорець «Використовувати віртуальну клавіатуру» на сторінці «Налаштуванн Клавіатура з’явиться після натискання на будь-яке поле введення, наприклад параметри зонда або записи таблиці інструментів. Щоб приховати клавіатуру, виконайте одну з таких дій:

- press the *HIDE* button on the virtual keyboard.
- натисніть кнопку ГОЛОВНА сторінка
- перейти в режим АВТО

Слід зазначити, що під час використання віртуальної клавіатури керування клавіатурою вимкнено.

10.5.6 Піни HAL

Ці контакти є специфічними для екрана QtDragon.
Звичайно, для функціонування LinuxCNC необхідно підключити ще багато контактів HAL.

Якщо вам потрібна підказка для ручної зміни інструменту, додайте ці рядки до вашого файлу postgui.
QtDragon емулює виводи HAL hal_manualtoolchange — не завантажуйте окремий компонент HAL «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
```

Також, якщо у вас немає автоматичного змінника інструментів, переконайтеся, що ці контакти підключені в одному з файлів HAL:

```
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

Цей вхідний контакт слід підключити для індикації стану зонда.

```
qtdragon.led-probe
```

Ці контакти є входами, пов’язаними з індикацією частотного перетворювача шпинделя.
Контакти вольт та ампер використовуються для розрахунку потужності шпинделя. Ви також повинні встановити MAX_SPINDLE_POWER в INI.

```
qtdragon.spindle-modbus-connection
qtdragon.spindle-modbus-errors
qtdragon.spindle-amps
qtdragon.spindle-fault
qtdragon.spindle-volts
```

Цей бітовий вивід є виходом для керування шпинделем для його призупинення.
Ви б підключили його до `spindle.0.inhibit`.

```
qtdragon.spindle-inhibit
```

Для відображення швидкості шпинделя QtDragon і світлодіода швидкості шпинделя необхідно, щоб `spindle.0.speed-in` був підключений до зворотного зв’язку швидкості шпинделя.
Можна використовувати зворотний зв’язок енкодера або VFD, якщо зворотний зв’язок вимірюється в обертах за секунду (RPS).
Якщо зворотний зв’язок недоступний, ви можете відобразити необхідну швидкість на дисплеї, підключивши контакти таким чином:

```
net spindle-speed-feedback spindle.0.speed-out-rps => spindle.0.speed-in
```

Цей вихідний контакт біта можна підключити для увімкнення лазера:

```
qtdragon.btn-laser-on
```

Цей висновок з плаваючою точкою показує обертання камери в градусах:

```
qtdragon.cam-rotation
```

Ці бітові/s32/float-виводи пов’язані із зовнішніми зміщеннями, якщо вони використовуються:

```
qtdragon.eoffset-clear
qtdragon.eoffset-enable
qtdragon.eoffset-value
qtdragon.eoffset-spindle-count
qtdragon.eoffset-zlevel-count
qtdragon.eoffset-is-active
```

Ці виводи з плаваючою точкою відображають поточний хід повзунка (у машинних одиницях):

```
qtdragon.slider-jogspeed-linear
qtdragon.slider-jogspeed-angular
```

Ці виводи з плаваючою точкою відображають поточні коефіцієнти перевизначення повзунка:

```
qtdragon.slider-override-feed
qtdragon.slider-override-maxv
qtdragon.slider-override-rapid
qtdragon.slider-override-spindle
```

Ці вихідні контакти доступні під час налаштування опції Versa Probe INI. Їх можна використовувати для автоматичного вимірювання довжини інструменту за допомогою датчика під час зміни інструмент – з додаванням коду перепризначення.

```
qtversaprobe.enable
qtversaprobe.blockheight
qtversaprobe.probeheight
qtversaprobe.probevel
qtversaprobe.searchvel
qtversaprobe.backoffdist
```

Цей вивід буде активним, коли завантажений інструмент дорівнюватиме номеру, встановленому в номері інструменту Versa Probe у файлі налаштувань.
Його можна використовувати (наприклад) для блокування шпинделя, коли зонд завантажений, підключивши його до `spindle.0.inhibit`.

```
qtversaprobe.probe-loaded
```

Цей вихідний контакт доступний при встановленні опції Basic Probe INI.
Цей контакт буде активним, коли завантажений інструмент дорівнює номеру, встановленому в полі редагування номера інструменту Basic Probe.
Його можна використовувати (наприклад) для блокування шпинделя при завантаженні зонда, підключивши його до `spindle.0.inhibit`.

```
qtbasicprobe.probe-loaded
```

Цей вхідний контакт доступний для перемикання між паузою та відновленням запущеної програми.

```
qtdragon.external-pause
```

Ви можете зовнішньо керувати відповідями діалогів у більшості діалогових вікон qtdragon.

```
qtdragon.dialog-ok
qtdragon.dialog-no
qtdragon.dialog-cancel
```

10.5.7 HAL-файли

Надані файли HAL призначені тільки для моделювання. Реальна машина потребує власних файлів HAL. Екран QtDragon працює з 3 або 4 осями з одним з’єднанням на вісь або 3 або 4 осями в конфігурації порталу (2 з’єднання на 1 вісь).

10.5.8 Ручна зміна інструментів

Якщо ваша машина вимагає ручної заміни інструментів, QtDragon може відобразити вікно з повідомленням, яке допоможе вам. QtDragon емулює контакти HAL hal_manualtoolchange — не завантажуйте окремий компонент HAL «hal_manualtoolchange». Для цього ви повинні підключити відповідний контакт HAL у файлі HAL postgui, наприклад:

```
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 Шпиндель

Екран призначений для підключення до VFD, але буде працювати і без нього.
У дистрибутив LinuxCNC входить ряд драйверів VFD.
Користувач самостійно підбирає відповідний драйвер і файли HAL відповідно до конфігурації свого комп’ютера.

10.5.10 Автоматичне підняття осі Z під час паузи програми

QtDragon можна налаштувати так, щоб він автоматично піднімав і опускав вісь Z та зупиняв шпиндель, коли програма призупинена.
Ви можете перемикати кнопки «SPINDLE LIFT» (Підняти шпиндель) або «NO LIFT» (Не піднімати), щоб вибрати опцію підняття шпинделя по осі Z під час призупинення.
Тоді, коли ви натиснете кнопку «PAUSE» (Призупинити), шпиндель підніметься на величину, встановлену на вкладці «Settings» (Налаштування), і зупиниться.
Натискання кнопки «RESUME» (Продовжити) запустить шпиндель і опустить його.
Якщо ви підключили контакт HAL «spindle.0.at-speed» до контакту приводу, шпиндель не опуститься, поки контакт не стане істинним.
Зазвичай його підключають до таймера або логічного пристрою, що визначає швидкість шпинделя.
Якщо цей контакт не підключений до контакту приводу, з’явиться діалогове вікно з попередженням про необхідність очікування швидкості шпинделя.
Шпиндель опуститься, коли ви закриєте це діалогове вікно.
Величина підйому встановлюється на вкладці «Налаштування» під заголовком «ПІДЙОМ ШПИНДЕЛЯ
Це поле редагування можна налаштувати безпосередньо тільки в режимі, відмінному від автоматичног
Кнопки вгору/вниз можна використовувати для регулювання величини підйому в будь-який час, навіть коли шпиндель вже піднятий.
Крок кнопки становить 1 дюйм або 5 мм (залежно від одиниць виміру, на яких базується машина).

Note
Якщо використовується опція підйому шпинделя, HALUI не може бути використаний для призупинення/відновлення програми. Для призупинення/відновлення з зовнішнього джерела доступний контакт «QtDragon.external-pause». Також необхідно ввімкнути зовнішні зміщення. На вкладці налаштувань встановіть прапорець «використовувати зовнішні зміщення». Якщо ви хочете заблокувати шпиндель при завантаженні зонда, вам потрібно буде використовувати логічний компонент «або» для об’єднання двох сигналів блокування шпинделя, щоб підключити їх до «spindle.0.inhibit».

Ця необов’язкова поведінка вимагає доповнень до INI та HAL-файлу QtDragon_postgui.

У INI-файлі, під заголовком AXIS_Z.

```
[AXIS_Z]
OFFSET_AV_RATIO  = 0.2
```

Це резервує 20% максимальної швидкості та максимального прискорення для зовнішніх зміщень.
Це обмежить максимальну швидкість машини на 20%

У файлі qtdragon_postgui.hal додайте:

```
# b’’Hb’’b’’ab’’b’’лb’’b’’ab’’b’’шb’’b’’тb’’b’’yb’’b’’вb’’b’’ab’’b’’нb’’b’’нb’’b’’яb’’ b’ ↩
    ’зb’’b’’ob’’b’’вb’’b’’нb’’b’’ib’’b’’шb’’b’’нb’’b’’ib’’b’’xb’’ b’’зb’’b’’мb’’b’’ib’’b’ ↩
    ’щb’’b’’eb’’b’’нb’’b’’ьb’’ b’’ob’’b’’cb’’b’’ib’’ Z
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

# b''Бb''b''лb''b''оb''b''кb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''еb''b''лb''b''яb'' b''пb''b''іb''b''дb'' b''чb''b''аb''b''сb'' b' ↩
    'пb''b''аb''b''уb''b''зb''b''иb''
net spindle-pause    qtdragon.spindle-inhibit        =>  spindle.0.inhibit

# b''pb''b''оb''b''зb''b''кb''b''оb''b''мb''b''еb''b''нb''b''тb''b''уb''b''вb''b''аb''b' ↩
    'тb''b''иb'' 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 Компенсація рівня Z

QtDragon_hd можна налаштувати для зондування та компенсації змін висоти рівня Z за допомогою зовнішньої програми «G-code Ripper».

Note
Це доступно лише у версії QtDragon_hd.

Компенсація рівня Z — це функція вирівнювання/корекції спотворення, яка зазвичай використовується в 3D-друку або гравіруванні. Вона використовує компонент HAL, що не працює в режимі реального часу, який використовує функцію зовнішніх зміщень LinuxCNC. Компонент має контакт HAL, який визначає тип інтерполяції, який повинен бути кубічним, лінійним або найближчим (відповідно 0, 1, 2). Якщо тип не вказано або вказано недійсне число, за замовчуванням вважається кубічний.

Коли Z LEVEL COMP увімкнено, компонент компенсації зчитує файл даних зонда, який повинен називатися «probe_points.txt». Файл можна змінювати або оновлювати в будь-який час, коли компенсація вимкнена. При наступному увімкненні файл буде перечитано, а карта компенсації перерахована. Цей файл повинен знаходитися в каталозі конфігурації.

Файл даних зонда генерується програмою зондування, яка, в свою чергу, генерується зовнішньою програмою Python під назвою «gcode_ripper», яку можна запустити з вкладки файлового менеджера за допомогою кнопки «G-code Ripper».

10.5.11.1 Використання G-коду Ripper для компенсації рівня Z

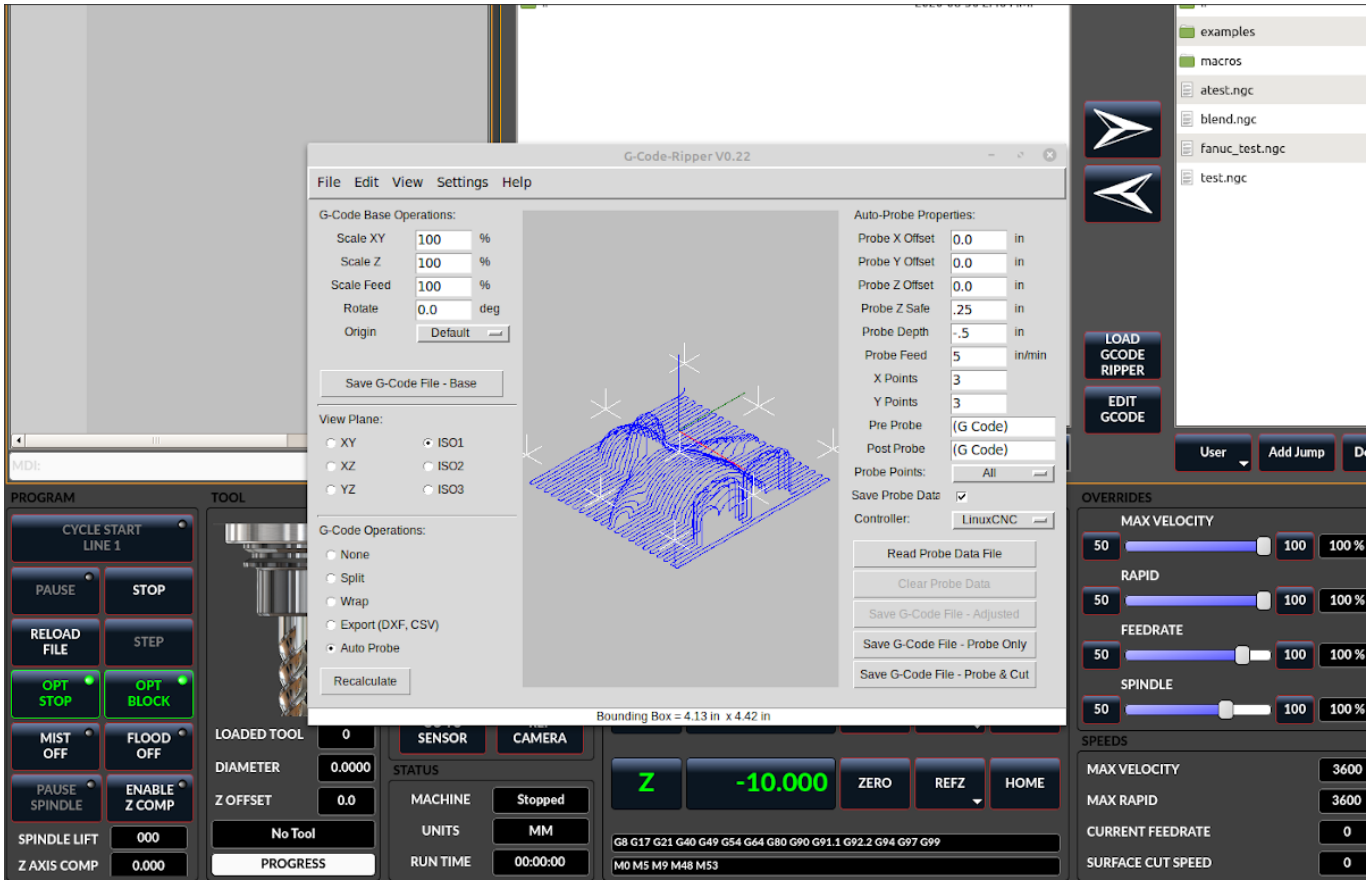


Figure 10.28: QtDragon_hd показує G-код Ripper

Note
G-code Ripper пропонує багато функцій, які ми тут не розглядатимемо. Ця функція доступна лише у версії QtDragon_hd.

- У qtdragon_hd перейдіть на вкладку файлів і натисніть кнопку завантаження G-коду Ripper.
- Встановіть початок координат відповідно до початку координат файлу G-коду, який потрібно зондувати.
- У розділі «Операції G-коду» позначте пункт «Автоматичне зондування».
- Файл -> Відкрити файл G-коду (файл, який ви запустите після компенсації)
- За потреби внесіть корективи та натисніть «Перерахувати».
- Натисніть «Зберегти файл G-коду – тільки зонд».
- Збережіть згенерований файл у папці nc_files.
- Вихід з gcode_ripper.
-

- Не змінюючи зміщення, запустіть цю програму. Переконайтеся, що інструмент зондування встановлено. Після завершення в каталозі конфігурації з'явиться файл під назвою *probe_points.txt*.
- У qtdragon_hd натисніть кнопку «Enable Z Comp» (Увімкнути компенсацію Z), щоб увімкнути компенсацію. Подивіться на рядок стану, щоб перевірити, чи операція була успішною. Активна компенсація буде відображатися поруч із написом: «Z Level Comp» (Компенсація рівня Z). Під час роботи дисплей повинен змінюватися відповідно до компонента компенсації.

Note
Якщо ви використовуєте автоматичне підняття Z для підйому шпинделя під час паузи, ви повинні поєднати ці два параметри з компонентом HAL та подати його до компонента руху LinuxCNC.

Зразок HAL-файлу postgui для комбінованого підйому шпинделя та компенсації рівня Z

```
# b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''иb'' b''нb''b''ab''b' ↩
    'вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb''
########################################################################

loadrt logic names=logic-and personality=0x102
addf logic-and servo-thread

# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b''кb''b' ↩
    'ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'' b''cb''b''yb''b''мb''b''yb''b' ↩
    'вb''b''ab''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''дb''b''ob''b''дb''b''ab''b' ↩
    'вb''b''ab''b''нb''b''нb''b''яb'' b''пb''b''ib''b''дb''b''йb''b''ob''b''мb''b''yb'' b' ↩
    'шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''тb''b''ab'' b''кb''b''ob''b' ↩
    'мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''їb'' Z
loadrt scaled_s32_sums
addf scaled-s32-sums.0 servo-thread

loadusr -Wn z_level_compensation z_level_compensation
# b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' b''мb''b''eb''b''тb''b''ob''b' ↩
    'дb''b''yb'' b''мb''b''ab''b''єb'' b''бb''b''yb''b''тb''b''иb'' b''ob''b''дb''b''нb''b' ↩
    'иb''b''мb'' b''ib''b''зb'' b''вb''b''ab''b''pb''b''ib''b''ab''b''нb''b''тb''b''ib''b' ↩
    'вb'': b''нb''b''ab''b''йb''b''бb''b''лb''b''иb''b''жb''b''чb''b''иb''b''йb''(2), b' ↩
    'лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''иb''b''йb''(1), b''кb''b''yb''b''бb''b''ib''b ↩
    ''чb''b''нb''b''иb''b''йb'' (0)
setp z_level_compensation.fade-height 0.0
setp z_level_compensation.method 1

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''иb''b''тb''b''иb'' b''cb''b''иb''b' ↩
    'гb''b''нb''b''ab''b''лb''b''иb'' b''дb''b''ob'' b''кb''b''ob''b''мb''b''пb''b''ob''b' ↩
    'нb''b''eb''b''нb''b''тb''b''ab'' b''pb''b''yb''b''xb''b''yb'' LinuxCNC
########################################################################

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

# b''зb''b''ob''b''вb''b''нb''b''ib''b''шb''b''нb''b''ib'' b''зb''b''мb''b''ib''b''щb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''фb''b''yb''b''нb''b''кb''b''цb''b' ↩
    'ib''b''їb'' b''пb''b''ab''b''yb''b''зb''b''иb'' b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'eb''b''лb''b''яb''
########################################################################
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

## b''Kb''b''ob''b''мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''яb'' b''pb''b' ↩
    'ib''b''вb''b''нb''b''яb'' Z
#####################################################
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

# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''pb''b''ib''b''вb''b''нb''b''яb'' Z b''тb''b''ab'' b''pb''b''ib''b''вb''b ↩
    ''нb''b''яb'' b''пb''b''ib''b''дb''b''йb''b''ob''b''мb''b''уb'' b''мb''b''ab''b''cb''b' ↩
    'шb''b''тb''b''ab''b''бb''b''ob''b''вb''b''ab''b''нb''b''ob''b''гb''b''ob'' b''шb''b' ↩
    'пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'' b''pb''b''ab''b''зb''b''ob''b''мb''
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 Зондування

Екран зонда пройшов базове тестування, але все ще можуть бути деякі незначні помилки. Під час виконання процедур зондування будьте дуже обережні, поки не ознайомитеся з принципом роботи. Процедури зондування виконуються без блокування основного графічного інтерфейсу користувача. Це дає оператору можливість спостерігати за цифровими індикаторами та зупинити процедуру в будь-який момент.

Note
Зондування дуже невблаганне до помилок; обов'язково перевірте налаштування перед використанням.

QtDragon має 2 методи для встановлення Z0. Перший — це сенсорна пластина, де металева пластина відомої товщини розміщується зверху заготовки, потім інструмент опускається, доки не торкнеться пластини, викликаючи сигнал зонда. Поточна система користувача (G5x) Z0 встановлюється на висоту зонда — введену товщину пластини.

Другий метод використовує налаштовувач інструменту у фіксованому положенні на відомій висоті над столом, де буде спрацьовувати сигнал зонда. Щоб встановити Z0 на верхню точку заготовки, він повинен знати

1. наскільки високо над столом знаходиться точка спрацьовування зонда (висота налаштування інструменту) та
2. наскільки високо над столом знаходиться верхня частина заготовки.

Цю операцію потрібно виконувати щоразу, коли змінюється інструмент, оскільки довжина інструменту не зберігається.

При зондуванні зондом, незалежно від того, чи використовуєте ви режим роботи з товщиною, встановленою на 0, чи використовуєте процедуру зондування, параметр висоти від столу до верхньої частини заготовки не враховується і може бути проігнорований. Він призначений тільки для налаштування інструменту.

10.5.12.1 Зонд Versa

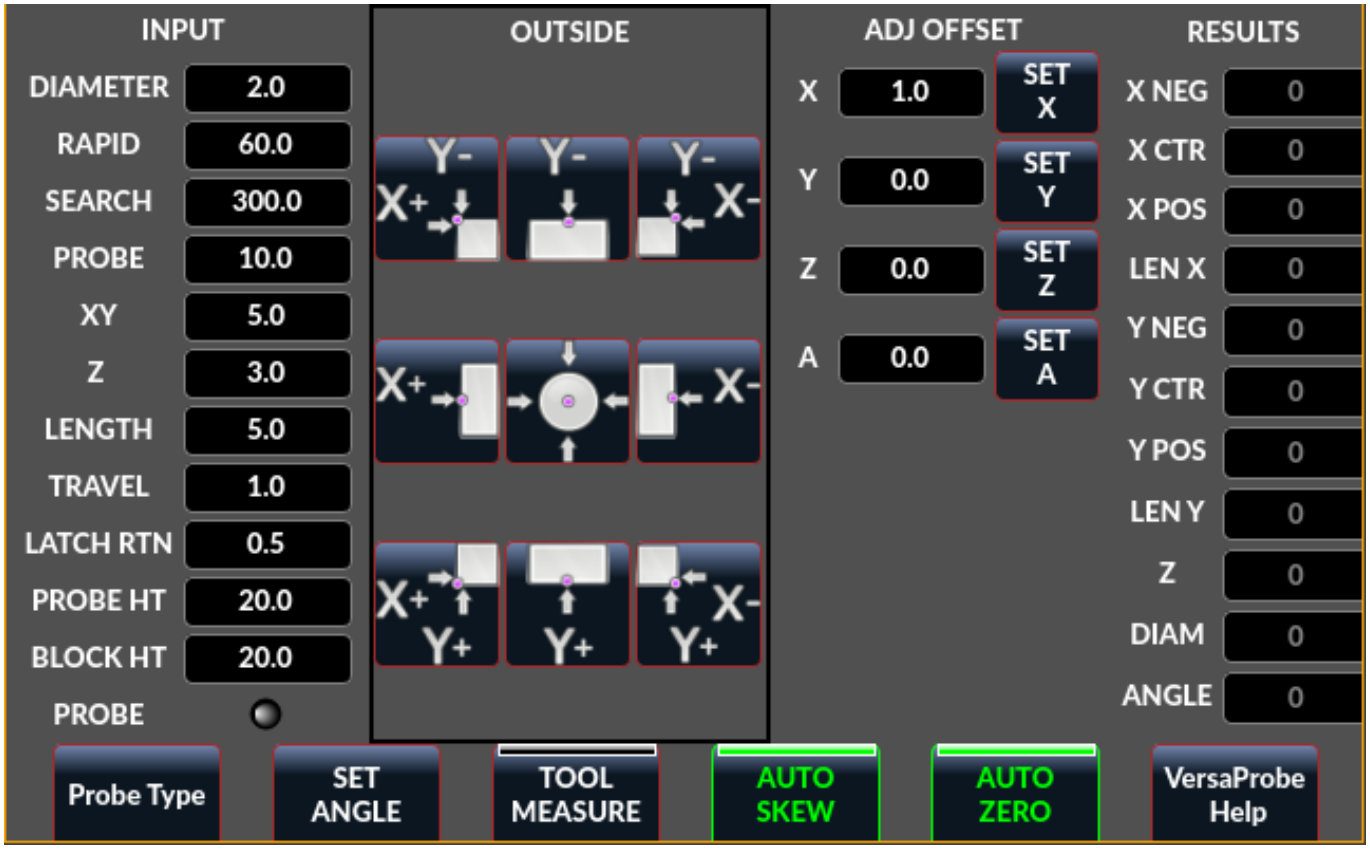


Figure 10.29: QtDragon - Варіант зонда Versa

Зонд Versa використовується для напівавтоматичного зондування заготовок для визначення країв, центрів та кутів.
Його також можна використовувати для автоматичного зондування довжини інструменту під час його зміни за допомогою доданого коду перепризначення.

Ви повинні ретельно встановити «Параметри зондування»:

ДІАМЕТР
: Це діаметр наконечника зонда. Точність вимірювань зонда безпосередньо залежить від точності діаметра наконечника зонда.

ПОДОРОЖІ
: Відстань, яку зонд пройде під час початкового пошуку. Якщо відстань пошуку занадто коротка, ви отримаєте повідомлення типу «G38 завершено без встановлення контакту». З міркувань безпеки рекомендується встановити цей параметр на 3-4 мм більше, ніж діаметр стилуса зонда.

ЗАСУВКА RTN
: Відстань, на яку зонд відсувається після першого контакту з деталлю. Ця відстань повинна бути невеликою, оскільки другий підхід буде здійснюватися на низькій швидкості, але достатньою для того, щоб зонд розірвав контакт і перейшов у стан готовності до пошуку. Якщо відстань Latch Rtn занадто велика, ви витратите багато часу на очікування завершення пошуку. Рекомендація: 1-2 мм

ПОШУК
: Це швидкість подачі, з якою зонд шукає цільову заготовку в одиницях машини за хвилину. Швидкість пошуку повинна бути достатньо повільною, щоб забезпечити прийнятну початкову точність, але достатньо швидкою, щоб не витрачати час на очікування руху. Рекомендація: 200-500 мм/хв.

ЗОНД
: Після встановлення першого контакту і відведення зонда він буде чекати 0,5 секунди, перш ніж знову виконати пошук на меншій швидкості, швидкості зонда. Ця менша швидкість гарантує, що машина зможе якомога швидше зупинити рух при контакті з заготовкою.

ШВИДКИЙ
: Рухи осей, не пов’язані з пошуком, виконуються зі швидкістю, визначеною RAPID у машинних одиницях за хвилину.

ДОВЖИНА БІКУ/КРАЮ
: Це відстань, на яку зонд переміститься з високою швидкістю до положення, в якому він почне пошук. При вимірюванні кута він переміститься на відстань, що дорівнює EDGE LENGTH, від кута, потім віддалиться від заготовки на відстань, що дорівнює XY CLEARANCE, опуститься на відстань, що дорівнює Z CLEARANCE, і почне початковий пошук. При вимірюванні внутрішнього кола EDGE LENGTH слід встановити на приблизний радіус кола. Примітка: НЕ на діаметр.

ЗОНД НТ
: Висота датчика інструменту від поверхні столу верстата. Це значення використовується для розрахунку нульової висоти Z для поточної системи координат заготовки під час використання зонда з датчиком налаштування інструменту.

БЛОК НТ
: Висота верхньої частини заготовки від поверхні столу верстата. Це значення використовується для обчислення нульової висоти Z для поточної системи координат обробки при використанні датчика з датчиком налаштування інструменту.

XY КЛІРАНС
: Відстань, на яку зонд відсунеться від краю або кута перед виконанням пошуку. Вона повинна бути достатньо великою, щоб зонд не торкався заготовки або будь-яких інших кріплень перед опусканням. Вона повинна бути достатньо малою, щоб уникнути надмірного очікування руху під час пошуку.

Z РОЗПРОДАЖ
: Відстань, на яку зонд переміститься перед виконанням пошуку. Якщо вимірюється внутрішній отвір, зонд можна вручну перемістити на початкову висоту Z, а потім встановити Z CLEARANCE на 0.

Є три кнопки-перемикачі:

Автоматичне обнулення
: Це визначає, чи після зондування відповідна вісь встановлена на нуль у поточній системі користувача.

Автоматичний нахил
: Це вибирає, чи після зондування система буде обертатися, чи просто відображатиметься розраховане обертання.

Вимірювання інструменту
: Це (якщо інтегровано) вмикає та вимикає автоматичне зондування інструменту.

Versaprobe пропонує 5 вихідних контактів для вимірювання інструменту та один, який можна використовувати для блокування шпинделя під час завантаження зонда.

5 контактів використовуються для зчитування з підпрограми G-коду перерозподілу, щоб код міг реагувати на різні значення.
Наразі номер зонда можна редагувати лише у файлі налаштувань:

```
[VERSA_PROBE_OPTIONS]
ps_probe_tool = 1
```

qtversaprobe.enable (HAL_BIT)
Інструмент вимірювання ввімкнено чи ні. Відображає стан кнопок на екрані.

qtversaprobe.blockheight (HAL_FLOAT)
Виміряна висота верхньої грані заготовки. Відображає вхід на екрані.

qtversaprobe.probeheight (HAL_FLOAT)
Висота перемикача зонда налаштування інструменту. Відображає введений на екрані.

qtversaprobe.searchvel (HAL_FLOAT)
Швидкість пошуку перемикача зонда інструменту

qtversaprobe.probevel (HAL_FLOAT)
Швидкість вимірювання довжини інструменту. Відображає введений на екрані запис.

qtversaprobe.backoffdist (HAL_FLOAT)
Відстань, на яку зонд відходить після спрацьовування. Відображає вхід на екрані.

qtversaprobe.probe-loaded (HAL_BIT)
Відображає, чи поточний інструмент дорівнює номеру зонда файлу налаштувань.

10.5.12.2 Базовий зонд

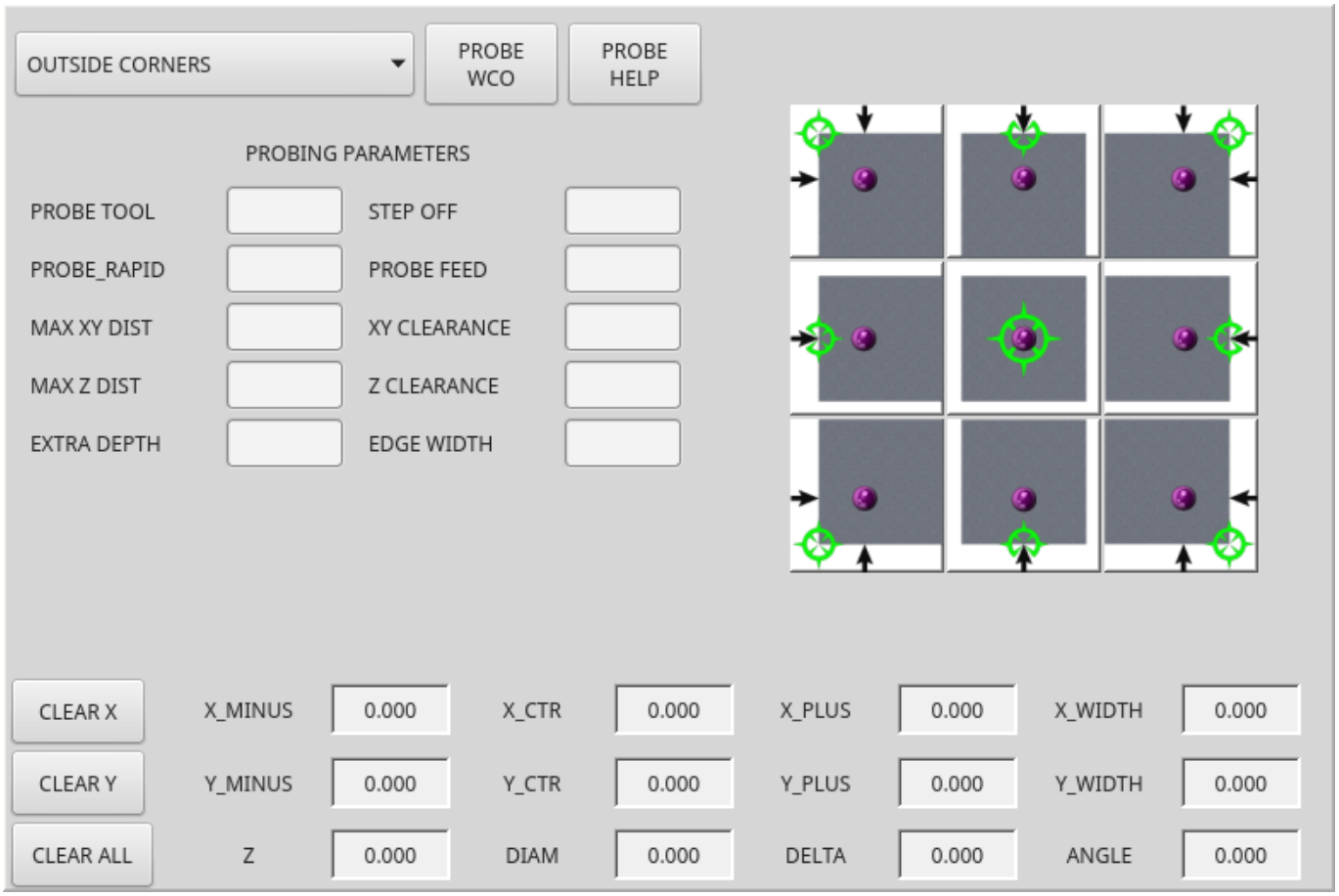


Figure 10.30: QtDragon - Базовий варіант зонда

Базовий зонд використовується для напівавтоматичного зондування заготовок для визначення країв, центрів та кутів. Комбінований список дозволяє вибрати основні типи зондування, що відображаються:

- Зовнішні кути
- Внутрішні кути
- Кути краю
- Бос і кишені
- Хребет і долини
- Калібрування

Ви повинні ретельно встановити «Параметри зондування»:

Зонд-інструмент
: Дозволятиме вимірювання лише за наявності інструменту з цим номером у шпинделі

Діаметр зонда
: Розмір кінчика зонда

Зонд Rapid
Швидкість швидких переміщень у верстатних агрегатах

Пошук зонда
Швидкість першого «чорного» пошуку в машинних одиницях

Подача зонда
Швидкість другого «тонкого» пошуку в машинних одиницях

Зійдіть
Відійдіть назад і знову виміряйте відстань

Макс. відстань XY
Максимальна відстань, яку зонд шукатиме по осях X та Y, перш ніж він видасть помилку

Макс. відстань Z
Максимальна відстань, яку зонд шукатиме по осі Z, перш ніж він видасть помилку

XY Розпродаж
Запасна відстань від зонда до краю стіни перед швидким переміщенням вниз по Z та «чорновим» зондуванням

Z Розпродаж
Зазор від зонду до верху матеріалу

Додаткова глибина
Відстань від верху матеріалу до бажаної глибини зонда

Також є параметри підказок залежно від вибраного типу зондування:

Ширина краю
Бажана відстань від початкової позиції зонда, вздовж краю стіни перед початком зондування

Підказка щодо діаметра
Використовується для зондування круглої бобишки або круглої кишені (початковий рух: 1/2 діаметра плюс зазор XY)

X Підказка
Використовується для зондування прямокутної бобишки/кишені (початковий рух: 1/2 довжини X плюс зазор XY)

Y Підказка
Використовується для зондування прямокутної бобишки/кишені (початковий рух: 1/2 довжини Y плюс зазор XY)

Після налаштування параметрів та підказок:

- Вручну перемістіть зонд у приблизне положення, позначене зеленою мішенню на кнопці.
- Переконайтеся, що параметри є прийнятними.
- Натисніть потрібну кнопку зондування.

Процедура зондування розпочнеться негайно.

Note
Натискання кнопки зупинки або клавіші Escape на клавіатурі перерве зондування.

Це можна використовувати для блокування шпинделя, коли зонд завантажено.
Ви повинні підключити його до spindle.0.inhibit

```
qtbasicprobe.probe-loaded
```

Давайте обговоримо зондування внутрішнього кута за допомогою верхньої правої кнопки в Basic Probe (back_right_inside). Хоча всі параметри зондування повинні бути правильними, найважливішими налаштуваннями, які потрібно змінити для кожного зонда, є:

XY РОЗПРОДАЖ
: Відстань від краю перед грубим зондуванням,

Z РОЗПРОДАЖ
: Відстань від зонда до верху матеріалу,

ДОДАТКОВА ГЛИБИНА
: Відстань до нижнього зонда від верху матеріалу,

ШИРИНА КРАЮ
: Відстань вздовж крайньої стіни (від кута), щоб почати зондування.

Note
Ці відстані завжди слід встановлювати в «машинних одиницях» (мм для метричних машин, дюйми для імперських машин).

Попередньо налаштовано:

- Вручну встановіть зонд на перетині країв (тобто кута) матеріалу, як позначено зеленим яблучком на кнопці. Встановіть його на Z-клімат вище верху матеріалу. Це можна зробити на око.
- Встановіть для параметра EXTRA CLEARANCE значення, при якому зонд має опускатися нижче *верхньої* межі матеріалу. (Таким чином, зонд переміститься з початкового положення вниз по Z Clearance + відстань Extra Clearance.)
- Встановіть для параметра XY CLEARANCE значення, яке точно забезпечить відстань від стіни, щоб зонд, опускаючись, ні об що не торкнувся.
- Встановіть для параметра «ШИРИНА КРАЙ» значення, яке описує відстань, виміряну від кута вздовж стіни до місця, де ви хочете виміряти точність. Ця відстань від краю буде використана вздовж стіни X, а потім вздовж стіни Y.

Послідовність після натискання кнопки зонда:

1. Швидке віддалення від кута на відстань ШИРИНИ КРАЙНОГО РОЗМІРУ з одночасним віддаленням від краю по осі XY. Отже, це злегка діагональний рух.
2. Перемістіть зонд вниз на Z CLEARANCE + EXTRA DEPTH,
3. стінка зонда двічі (шорстка та тонка),
4. рухатися по діагоналі до іншої стіни, як встановлено параметрами ШИРИНА КРАЙНОГО РЕЖИМУ та ЗАСІБ XY,
5. зонд стінки двічі,
6. підняти зонд на Z CLEARANCE + EXTRA DEPTH (повернення на початкову висоту),
7. швидке повернення до початкового кута (тепер розраховане з використанням зондованих стін),
8. Якщо кнопка автоматичного обнулення увімкнена, встановіть X та Y поточної користувацької системи на нуль.

10.5.12.3 Налаштування віджета екрана зонда

Можна завантажити налаштовану версію віджета зонда.

У папці config повинна бути папка для екранів: <CONFIG FOLDER>/qtvcp/.
Може бути (або може бути додана) папка «lib/» і «widgets/».
У папку widgets можна скопіювати «basic_probe.py» (або «versa_probe.py») і «probe_subprog.py».
У папці lib скопіюйте файл «touchoff_subprogram.py».
Якщо ці файли будуть знайдені, вони будуть використовуватися замість оригіналів.
Ви можете модифікувати файли, щоб змінити поведінку.

10.5.13 Сенсорна панель



Figure 10.31: QtDragon - Сенсорна панель

Ви можете використовувати провідну сенсорну пластину або її еквівалент для автоматичного зняття (обнулення координат користувача) для положення Z інструменту. Перед зондуванням необхідно завантажити інструмент. На вкладці «Інструмент» або «Налаштування» встановіть висоту сенсорної пластини, швидкість пошуку та зондування, а також максимальну відстань зондування.

Note
При використанні провідної пластини швидкість пошуку і зонда повинна бути однаковою і повільною. Якщо ви використовуєте інструмент для налаштування з пружинним ходом, ви можете встановити більш високу швидкість пошуку. LinuxCNC знижує швидкість при максимальному прискоренні, тому при високій швидкості після спрацьовування зонда може відбутися переміщення.

Покладіть пластину на поверхню, на якій ви хочете обнулити Z. Підключіть вхідний провід датчика до інструменту (якщо використовуєте провідну пластину). Перед початком вимірювання світлодіод підтвердить надійність підключення датчика. Вручну перемістіть інструмент у межах максимальної відстані датчика. Натисніть кнопку «Touch Plate» (Доторкнутися до пластини). Машина двічі виміряє відстань, і поточне зміщення користувача (G5X) буде обнулено в нижній частині пластини за розрахунком на основі налаштування висоти пластини.

10.5.14 Автоматичне вимірювання інструментів

10.5.14.1 Огляд

QtDragon можна налаштувати для виконання інтегрованого автоматичного вимірювання інструментів за допомогою віджета Versa Probe та коду перепризначення.

Ця функція передбачає використання двох зондів одночасно:

1. Датчик перемикання інструменту, закріплений десь на верстаті (іноді його називають налаштовув інструменту), та
2. шпиндельний зонд, який тимчасово встановлюється на початку роботи (іноді його називають зондом xyz або зондом Renishaw).

Ці методи корисні для верстатів, які не мають повторюваних тримачів інструментів і не мають пристроїв автоматичної заміни інструментів. (Для верстатів з повторюваними тримачами інструментів див. розділ measuring tool length. Для верстатів з пристроями автоматичної заміни інструментів див. роботу, виконану в репозиторії LinuxCNC за адресою `configs/sim/axis/remap/rack-toolchange`.)

Щоб скористатися цією функцією, вам потрібно буде виконати деякі додаткові налаштування, і, можливо, ви захочете використовувати запропоновані контакти HAL для отримання значень у власній процедурі перемапування ngc. Ці налаштування розглядаються далі в цьому розділі.

По-перше, у цьому документі описано, як використовувати цю техніку. По-друге, у цьому документі описано, як налаштувати цю техніку на початку виробничого циклу.

10.5.14.2 Огляд робочого процесу

Після цього огляду наведено детальний покроковий огляд робочого процесу.

Кроки налаштування включають:

- Введення швидкостей зонда на сторінці налаштувань зонда Versa.
- Увімкнення функції «Використовувати вимірювання інструменту» на вкладці Versa Probe.
- Увімкнення опції «Використовувати датчик інструмента» в налаштуваннях.

Important
Під час першої настройки автоматичного вимірювання інструменту будьте обережні, поки не підтвердите зміну інструменту та розташування датчика — інструмент/датчик легко зламати. Команда «Перервати» буде виконана, поки датчик знаходиться в русі.

Вимірювання інструменту в QtDragon організовано за такими кроками, які будуть детальніше пояснені в наступному розділі:

1. Встановіть нульовий інструмент зонда, вимірявши інструментальний налаштовувач зі встановлен шпиндельним зондом
2. Прикладіть до заготовки контакти по осях X та Y.
3. Виміряйте висоту вашого блоку від основи, де розташований перемикач інструментів, до верхньої поверхні блоку (включаючи патрон тощо).
4. У вкладці зонда Versa введіть виміряне значення висоти блоку.
5. Перейдіть в автоматичний режим і запустіть програму.

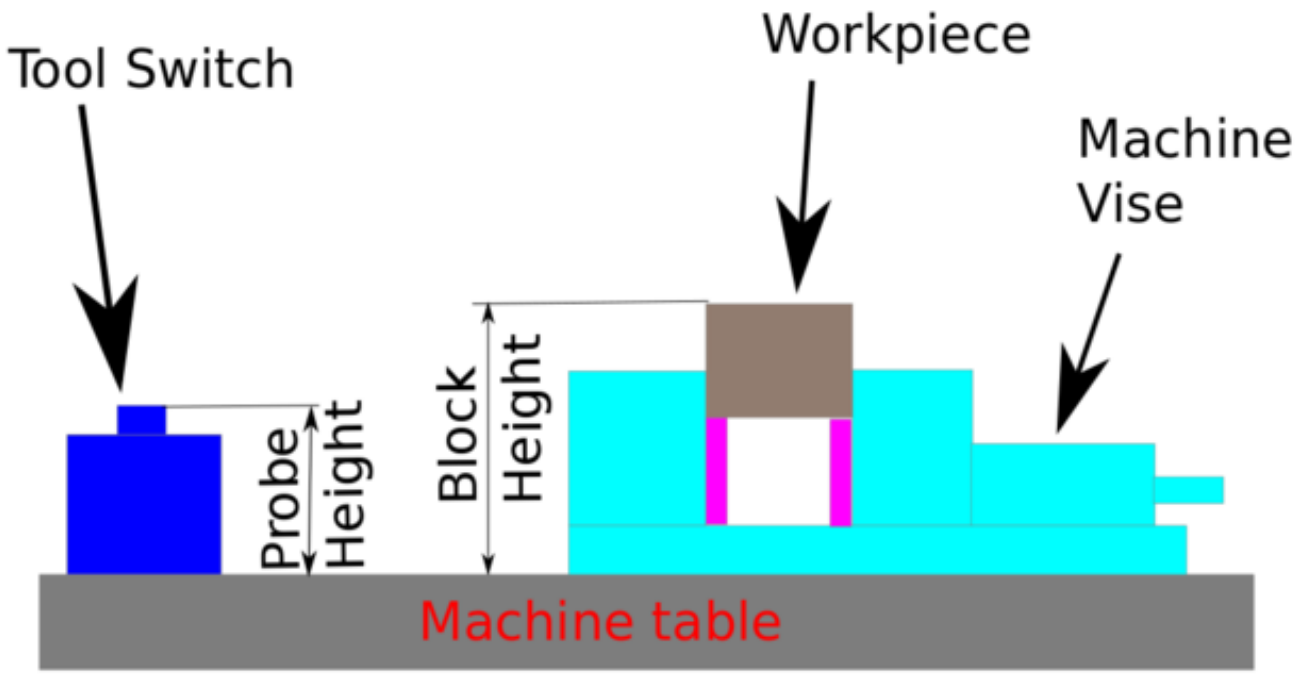


Figure 10.32: Автоматичне вимірювання інструменту

З першою зміною інструменту інструмент буде виміряно, а зміщення буде встановлено автоматично відповідно до висоти блоку. Перевагою цього способу є те, що вам не потрібен опорний інструмент.

Note
Ваша програма повинна містити зміну інструменту на початку. Інструмент буде виміряний, навіть якщо він вже використовувався раніше, тому немає небезпеки, якщо висота блоку змінилася. На YouTube є кілька відео, які демонструють техніку використання GMOCCAPY. Екран GMOCCAPY був піонером цієї техніки.

Послідовність подій (з використанням файлів за замовчуванням у налаштуваннях за замовчуванням):

1. Швидкий рух по осі Z до позиції, визначеної в INI-файлі [TOOL_CHANGE] Z.
2. Швидкий рух по X та Y до числа, визначеного в INI-файлі [TOOL_CHANGE] X та Y.
3. Виконайте звичайну заміну інструменту M6, тобто зупиніть шпиндель, надішліть повідомлення користувачеві для заміни інструменту.
4. Швидкий рух по X та Y до позиції, визначеної в INI-файлі [VERSA_TOOLSETTER] X та Y.
5. Швидкий рух вниз по осі Z до позиції, визначеної в INI-файлі [VERSA_TOOLSETTER] Z.
6. Здійснити зондування вниз по Z до максимуму, визначеного в MAXPROBE INI-файлу [VERSA_TOOL
7. Змінює зміщення поточної системи координат заготовки відповідно до різниці між попереднім інструментом та поточним виміряним інструментом.
8. Швидке переміщення вгору по осі Z до позиції, визначеної в INI-файлі [VERSA_TOOLSETTER] Z_MAX_CLEAR.
9. Швидкий перехід до позиції X Y під час виклику зміни інструменту.
10. Швидкий рух вниз до позиції Z під час виклику зміни інструменту.

Note
Позиція Z [TOOL_CHANGE] повинна бути достатньо високою, щоб інструмент не зачепив зонд інструменту під час переміщення до позиції X та Y [VERSA_TOOLSETTER].

Відстань MAXPROBE повинна бути достатньо великою, щоб інструмент міг торкнутися зонда.

10.5.14.3 Детальний приклад робочого процесу

Наведені нижче налаштування потрібно виконати лише один раз, якщо вони залишаються чинними:

1. У розділі «Екрани інструментів зондування»: переконайтеся, що значення для «Швидке» та «Пошук» прийнятні. Це швидкості, з якими буде виконуватися зондування, вказані в машинних одиницях за хвилину.

2. У розділі «Екрани інструментів зонда»: переконайтеся, що опція «Вимірювання інструмента» увімкнена (цю кнопку потрібно виділити.)
3. У розділі «Налаштування»: переконайтеся, що опція «Використовувати датчик інструмента» увімкнена (це прапорець, який потрібно встановити)
4. У таблиці інструментів: налаштуйте інструмент для шпиндельного зонда та переконайтеся, що його зміщення Z встановлено на нуль.

Note
Можна використовувати ненульову довжину інструмента для зонда інструмента, але це вимагає додаткових кроків і легко може призвести до помилок. Наведена нижче процедура передбачає нульову довжину зонда інструмента.

Наступне налаштування виконується перед запуском програми, яка містить команди зміни інструмент
M6.

1. Фізично вставте шпиндельний зонд у шпиндель.
2. Логічно завантажте шпиндельний зонд у шпиндель за допомогою команди M61 Qx, де x — це номер у таблиці інструментів для шпиндельного зонда (у вкладці таблиці інструментів є кнопка, яку також можна використовувати.)
3. Позиціонування до інструментального верстата: Використовуйте кнопку під екранами зондів для «Перейти до інструментального верстата», щоб розташувати шпиндель над інструментальним верстата.
4. Вимірювання налаштовувачем інструменту: використовуйте кнопку під екранами датчика для ”Висота Z налаштовувача інструменту датчика”. Зверніть увагу, що це встановить та відобразить на екрані налаштувань датчика значення ”Висота датчика” в координатах ABS
5. Перейдіть до заготовки.
6. Вимірювання заготовки: Використовуйте кнопку під екранами зонда для «Вимірювання висоти матеріалу зондом:» це встановить і відобразить на екрані налаштувань зонда значення «Висота блоку» в координатах ABS. (Зазвичай це також буде нульовою точкою Z для вашої робочої системи координат)
7. Встановлення системи координат заготовки (наприклад, G54 або інша): Використовуйте інструмент «Зонд» та будь-який екран зонда чи інший відповідний метод для встановлення системи координат X, Y та Z, необхідної для вашого завдання.
8. Якщо ваша програма починається з команди TnM6 перед обертанням шпинделя, ви можете залишити датчик шпинделя встановленим. Ви також можете видати команду TnM6 для заміни датчика шпинделя, і якщо програма видасть ту саму команду, вона пропустить заміну інструменту.



Caution
Будьте обережні, щоб не залишити зонд шпинделя в шпинделі, якщо програма може запустити шпиндель.

Після завершення цих кроків можна запустити програму з кількома змінами інструментів TnM6, яка працюватиме з автоматичними паузами для ручної зміни інструменту та подальшим автоматичним вимірюванням інструменту.

Note
Після вимірювання довжини нового інструменту за допомогою пристрою для налаштування інструменту цей код перемапування використовує G43, який застосовує зміщення до системи робочих координат, яка діяла на момент видачі команди M6. Оскільки перемапування скоригувало систему робочих координат за допомогою зміщення між попереднім і поточним інструментом, кінчик інструменту опиниться в тій самій точці простору, що й кінчик попереднього інструменту на момент виклику зміни інструменту.

10.5.14.4 Вимірювання висоти заготовки зондом у QtDragon_hd

Позиція Z [TOOL_CHANGE] повинна бути достатньо високою, щоб інструмент не зачепив зонд інструменту під час переміщення до позиції X та Y [VERSA_TOOLSETTER]. Відстань MAXPROBE повинна бути достатньо великою, щоб інструмент міг торкнутися зонда.

10.5.14.5 Вимірювання висоти заготовки зондом

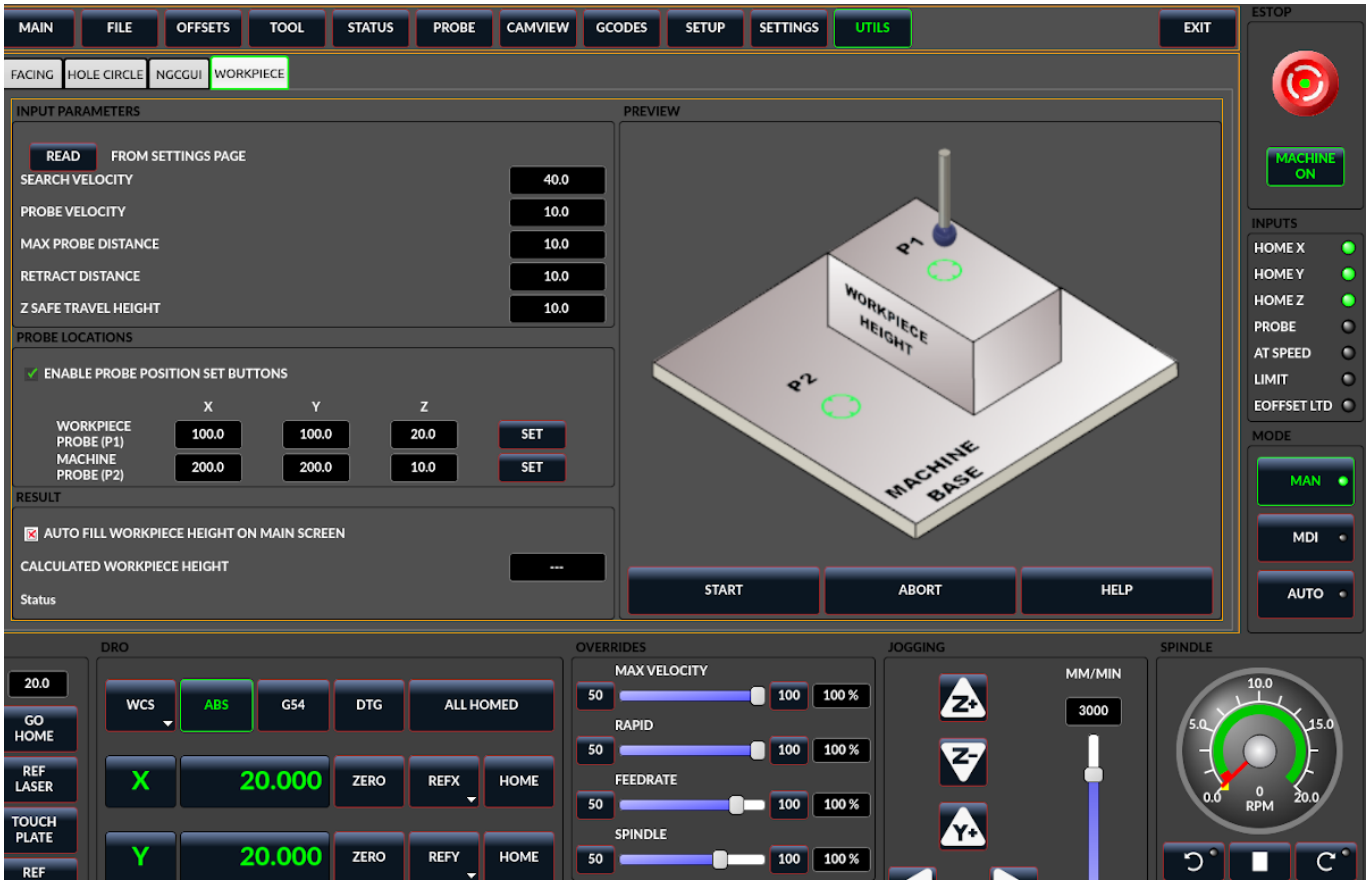


Figure 10.33: QtDragon_hd - Вимірювання висоти заготовки зондом

Ця програма вимірює 2 заданих користувачем місця на осі Z та обчислює різницю висот.

Note
Це доступно лише у версії QtDragon_hd.

Увімкнути кнопки встановлення положення зонда

- Якщо позначено, кнопки SET активовані.
- Це дозволяє користувачеві автоматично заповнювати параметри X, Y та Z поточним положенням, яке відображається на цифрових індикаторах (DRO).

Автоматичне заповнення висоти заготовки на головному екрані

- Після встановлення прапорця розрахована висота автоматично переноситься в поле «Висота заготовки» на головному екрані.
- В іншому випадку головний екран не змінюється.

Зонд заготовки в

- Координати X, Y та Z вказують, де має розпочатися перша процедура зондування в поточній системі WCS

Машинний зонд у

- Координати X, Y та Z вказують, де має розпочатися друга процедура зондування в поточній системі WCS

Безпечна висота руху Z

- Машину піднімають на безпечну висоту переміщення Z перед поштовхом до координат X та Y.
- Потім шпиндель опускається до заданої координати Z.
- Його слід вибрати таким чином, щоб інструмент усував усі перешкоди під час бігу.

Кнопка ПУСК

- Машина переміститься до першого місця, а потім проведе зондування вниз.
- Потім машина переміщується до другого місця та знову проводить зондування.
- Різниця між вимірюваними значеннями повідомляється як Розрахована висота заготовки.
- Параметри швидкості пошуку, швидкості зонда, максимальної відстані зонда та відстані повернення зчитуються з головної сторінки налаштувань графічного інтерфейсу.

Кнопка ПЕРЕРВАТИ

- призводить до зупинки всіх виконуваних на даний момент процедур штовхання та зондування.

Кнопка ДОПОМОГА

- відображає цей файл довідки.
- Можна вказати будь-які 2 точки в межах робочого об’єму машини.
- Якщо перша точка вища за другу, розрахована висота буде додатним числом.
- Якщо перша точка нижча за другу, розрахована висота буде від’ємним числом.
- Одиниці вимірювання не мають значення в цій програмі. Досліджені значення не зберігаються, а відображається лише різниця.



Caution
Встановлення неправильних значень може призвести до зіткнень з пристосуваннями на робочій поверхні верстата. Рекомендується проводити початкове тестування без інструменту та на безпечній висоті.

10.5.14.6 Штифти для вимірювання інструментів

Versaprobe пропонує 5 вихідних контактів для вимірювання інструменту. Ці контакти використовуютьс
для зчитування з підпрограми G-коду перепризначення, тому код може реагувати на різні значення.

qtversaprobe.enable (HAL_BIT)
Інструмент вимірювання ввімкнено чи ні. Відображає стан кнопок на екрані.

qtversaprobe.blockheight (HAL_FLOAT)
Виміряна висота верхньої грані заготовки. Відображає вхід на екрані.

qtversaprobe.probeheight (HAL_FLOAT)
Висота перемикача зонда налаштування інструменту. Відображає введений на екрані.

qtversaprobe.searchvel (HAL_FLOAT)
Швидкість пошуку перемикача зонда інструменту

qtversaprobe.probevel (HAL_FLOAT)
Швидкість вимірювання довжини інструменту. Відображає введений на екрані запис.

qtversaprobe.backoffdist (HAL_FLOAT)
Відстань, на яку зонд відходить після спрацьовування. Відображає вхід на екрані.

10.5.14.7 Зміни INI-файлу вимірювання інструменту

Змініть свій INI-файл, включивши наступне:

QtDragon дозволяє вибрати один із двох стилів процедур зондування. Зонд Versa працює з перепризнач
M6 для додавання автоматичного зондування інструменту.

```
[PROBE]
#USE_PROBE = versaprobe
USE_PROBE = basicprobe
```

Відомості про розділ RS274NGC
Відомості про оператор перепризначення
Відомості про переривання перепризначення

Note
Ці стандартні записи повинні працювати нормально в більшості випадків. Деякі системи можуть потребувати використання *linuxcnc/nc_files/examples/* замість *linuxcnc/nc_files/*. Перевірте, чи шляхи є дійсними. Можливі власні записи, що вказують на модифікований файл.

```
[RS274NGC]

# b''3b''b''мb''b''ib''b''нb''b''ib''b''тb''b''ьb'' b''цb''b''ib'' b''шb''b''лb''b''яb''b' ↩
    'xb''b''иb'', b''щb''b''ob''b''6b'' b''вb''b''ob''b''нb''b''иb'' b''вb''b''кb''b''ab''b' ↩
    'зb''b''уb''b''вb''b''ab''b''лb''b''иb'' b''нb''b''ab'' b''пb''b''ab''b''пb''b''кb''b' ↩
    'иb'' b''зb''b''ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb''  ↩
    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
```

```
# b''єb'' b''пb''b''іb''b''дb''b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''єb''b''юb'', b' ↩
    'яb''b''кb''b''аb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''єb''b''тb''b' ↩
    'ьb''b''сb''b''яb'', b''кb''b''оb''b''лb''b''иb'' b''вb''b''иb''b''нb''b''иb''b''кb''b' ↩
    'аb''b''єb'' b''пb''b''оb''b''мb''b''иb''b''лb''b''кb''b''аb'' b''пb''b''іb''b''дb'' b' ↩
    'чb''b''аb''b''сb'' b''зb''b''мb''b''іb''b''нb''b''иb'' b''іb''b''нb''b''сb''b''тb''b' ↩
    'рb''b''уb''b''мb''b''еb''b''нb''b''тb''b''уb''.
ON_ABORT_COMMAND=O <on_abort> call

# b''Kb''b''оb''b''дb'' b''пb''b''еb''b''рb''b''еb''b''пb''b''рb''b''иb''b''зb''b''нb''b' ↩
    'аb''b''чb''b''еb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''аb''b''вb''b''тb''b' ↩
    'оb''b''мb''b''аb''b''тb''b''иb''b''чb''b''нb''b''оb''b''гb''b''оb'' b''зb''b''оb''b' ↩
    'нb''b''дb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb'' Z b''іb''b''нb''b''сb''b''тb''b' ↩
    'рb''b''уb''b''мb''b''еb''b''нb''b''тb''b''оb''b''мb'' versaprobe b''уb'' QtVCP
REMAP=M6  modalgroup=6 prolog=change_prolog ngc=qt_auto_probe_tool epilog=change_epilog
```

Положення датчика інструменту та початкове положення руху зондування.
Усі значення є абсолютними (G53) координатами, за винятком MAXPROBE, яке виражається як абсолютна довжина руху.
Усі значення вказані в одиницях виміру, властивих для даної машини.
X, Y та Z встановлюють місце розташування зонда інструменту.

Послідовність дій автоматичного зонда в прикладі qt_auto_probe_tool за замовчуванням, перепризначе вище (цю поведінку можна змінити, модифікувавши оператор remap у розділі RS274NGC або модифікувавши код qt_auto_probe_tool.ngc.):

1. швидкий перехід до позиції Z [CHANGE_POSITION] INI (це відносний перехід, він додає це значення Z до поточної координати Z)
2. швидкий перехід до позиції [CHANGE_POSITION] X та Y INI.
3. зачекайте підтвердження ручної зміни інструменту
4. швидкий перехід до позицій X та Y INI [VERSA_TOOLSETTER]
5. швидкий перехід до позиції Z_MAX_CLEAR INI [VERSA_TOOLSETTER]
6. швидкий зонд
7. повільний зонд
8. швидкий перехід до позиції Z_MAX_CLEAR INI [VERSA_TOOLSETTER]

Z_MAX_CLEAR – це позиція Z, до якої потрібно перейти перед переходом до налаштовувача інструменту під час використання кнопки «Переміщення до налаштовувача інструменту».
Послідовність дій «Переміщення до налаштовувача інструменту»:

1. швидкий перехід до позиції Z [VERSA_TOOLSETTER] Z_MAX_CLEAR
2. швидкий перехід до позиції XY [VERSA_TOOLSETTER]
3. швидкий перехід до позиції Z [VERSA_TOOLSETTER].

Приклади налаштувань:

```
[VERSA_TOOLSETTER]
X = 10
Y = 10
Z = -20
Z_MAX_CLEAR = -2
MAXPROBE =  -20
```

Це навмисно не називається TOOL_CHANGE_POSITION — **Canon використовує цю назву, і в іншому випадку це спричинить конфлікт**. Позиція, в яку потрібно перемістити верстат перед тим, як дати команду на зміну інструменту. Усі значення вказані в абсолютних координатах. Усі значення вказані в одиницях виміру, властивих для верстата.

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

Розділ Python визначає, які файли шукає інтерпретатор Python LinuxCNC, наприклад, файл `toplevel.p` у папці `python` у каталозі конфігурації: Ці стандартні записи повинні працювати нормально в більшості випадків. Деякі системи можуть потребувати використання *linuxcnc/nc_files/examples/* замість *linuxcnc/nc_files/*. Можливі власні записи, що вказують на модифікований файл.

```
# b''Пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ab'' b''тb''b''ob''b''чb''b' ↩
    'кb''b''ab'' b''шb''b''лb''b''яb''b''хb''b''yb'' b''дb''b''лb''b''яb'' b''вb''b''cb''b' ↩
    'ib''b''xb'' b''пb''b''ob''b''шb''b''yb''b''кb''b''ib''b''вb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'тb''b''ob''b''бb''b''тb''b''ob'' sys.path.append() b''yb'' Python
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 Необхідні HAL-з'єднання

Переконайтеся, що ви підключили вхід зонда інструмента до вашого HAL-файлу: якщо підключено правильно, ви зможете перемикати світлодіод зонда в QtDragon, натискаючи стилус зонда.

```
net probe  motion.probe-input <= <your_input_pin>
```

10.5.15 Бігти від лінії

Програму G-коду можна запустити з будь-якого рядка, натиснувши на потрібний рядок у вікні відображення G-коду в режимі AUTO. Оператор несе відповідальність за те, щоб машина перебувала в потрібному режимі роботи. З'явиться діалогове вікно, в якому можна заздалегідь встановити напрямок і швидкість шпинделя. Початкова рядок вказана в полі з написом LINE (РЯДОК) поруч із кнопкою CYCLE START (СТАРТ ЦИКЛУ). Функцію запуску з рядка можна вимкнути на сторінці налаштувань.

Note
Команда run-from-line в LinuxCNC не дуже зручна для користувача. Наприклад, вона не запускає шпиндель і не підтверджує правильність вибору інструменту. Крім того, вона не дуже добре обробляє підпрограми. Якщо її використовувати, то найкраще починати з швидкого переміщення.

10.5.16 Лазерні кнопки

Кнопка LASER ON/OFF призначена для ввімкнення або вимкнення виходу, який підключений до невеликого лазерного проектора з перехрестям. Коли перехрестя розташоване над бажаною контрольною точкою на заготовці, можна натиснути кнопку REF LASER, яка встановлює зміщення по осях X і Y на значення, вказані в полях LASER OFFSET на сторінці налаштувань.

10.5.17 Опис вкладок

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 Головна вкладка

Ця вкладка відображає графічне представлення поточної програми. Бічні кнопки керуватимуть дисплеєм.

- *User View*: Вибрати/відновити встановлений користувачем вигляд поточної програми.
- *P,X,Y,Z*: Встановити стандартні подання.
- *D*: Увімкнути/вимкнути відображення розмірів.
- +, -: Елементи керування масштабуванням.
- *C*: Чітка графіка ліній руху інструменту.

У `qtdragon_hd` також є макро-кнопки, доступні з правого боку. В INI можна визначити до десятків кнопок.

10.5.17.2 Вкладка «Файл»

Ви можете використовувати цю вкладку для завантаження або перенесення програм. Редагування програм G-коду можна вибрати з цієї вкладки. За допомогою `qtdragon_hd` саме тут можна завантажити «G-code Ripper».

10.5.17.3 Вкладка «Зміщення»

Ви можете контролювати/змінювати системні зміщення на цій вкладці. Є зручні кнопки для обнулення обертання, G92 та поточного користувацького зміщення G5x.

10.5.17.4 Вкладка інструментів

На цій вкладці можна контролювати/змінювати зміщення інструментів. На цій вкладці також можна додавати та видаляти інструменти з файлу інструментів. Коли вибрано цю вкладку, окремі кнопки повернення в початкове положення в області DRO змінюються на кнопки налаштування зміщення інструментів. Вони повертаються до кнопок повернення в початкове положення, коли ви вибираєте іншу вкладку. Натискання цієї кнопки інструменту відкриває меню опцій:

- Встановити поточне положення інструменту
- Налаштування поточного положення інструменту
- Нульове поточне положення інструменту
- Безпосереднє встановлення зміщення інструменту
- Скинути до останнього

10.5.17.5 Вкладка стану

Тут буде відображатися журнал важливих подій машини або системи з позначкою часу. Події машини більше підійдуть оператору, якому системні події можуть допомогти у вирішенні проблем.

10.5.17.6 Вкладка зонда

На цій вкладці відображаються параметри процедур зондування. Залежно від параметрів INI, це може бути стиль VersaProbe або BasicProbe. Вони функціонально схожі. QtDragon_hd також покаже менше вікно графічного дисплея.

10.5.17.7 Вкладка «Вигляд камери»

Якщо розпізнана веб-камера підключена, на цій вкладці буде відображатися відеозображення з накладеним перехрестям, колом і показником градусів. Це можна налаштувати відповідно до особливостей деталі, наприклад для дотику. Базова бібліотека використовує модуль openCV Python для підключення до веб-камери.
Щоб налаштувати співвідношення сторін X або Y (у відсотках), номер порту камери, API-бекенд або необхідну роздільну здатність, знайдіть у файлі налаштувань:

```
[CUSTOM_FORM_ENTRIES]
Camview xscale = 100
Camview yscale = 100
Camview cam number = 0
Camview cam api = V4L2
Camview cam resolution = 1280,720
```

Шкали вказані у відсотках, зазвичай діапазон становить 100–200 на одній осі.
Змінюючи ці шкали, можна перевернути зображення по осі X, Y або по обох осях.
API походить від openCV, доступні бекенди будуть перелічені, якщо використовувати -V debugging. Встановіть значення «ANY», щоб opencv вибрав сам.
Встановіть роздільну здатність на DEFAULT, щоб opencv вибрав її. Доступні роздільні здатності будуть перелічені, якщо використовується -V debugging.

Note
Файл налаштувань можна редагувати лише тоді, коли QtDragon не запущено.

10.5.17.8 Вкладка G-кодів

На цій вкладці буде відображено список G-коду LinuxCNC. Якщо ви натиснете на рядок, відобразиться опис коду.

10.5.17.9 Вкладка налаштувань

Можна завантажити файл HTML або PDF (з розширенням .html / .pdf) з примітками щодо налаштування і він буде відображатися на вкладці налаштування.
Якщо ви завантажуєте програму G-коду і є файл HTML/PDF з такою ж назвою, він завантажиться автоматично.
Деякі програми, такі як Fusion 360 і Aspire, створюють ці файли за вас. Ви також можете написати власні HTML-документи за допомогою кнопки SetUp Writer, що входить до комплекту.
Є три підвкладки:

- *HTML* - Тут відображаються будь-які завантажені HTML-сторінки. Кнопки навігації працюють на цій сторінці.
- *PDF* - тут відображаються будь-які завантажені сторінки налаштувань PDF-файлу.
- *PROPERTIES* - Коли програма завантажується, тут відображаються властивості її G-коду.

Є кнопки навігації для HTML-сторінки:

- Стрілка вгору повертає вас на стандартну HTML-сторінку.
- Стрілка вліво переміщує вас назад на одну HTML-сторінку.
- Стрілка вправо переміщує на одну HTML-сторінку вперед.

Якщо ви бажаєте додати власну HTML-сторінку за замовчуванням, назвіть її «default_setup.html» і розмістіть у папці конфігурації.
Власні панелі QtVCP можна відобразити на цій вкладці, встановивши для параметра EMBED_TAB_LOCA значення «tabWidget_setup».



Figure 10.34: QtDragon - Зразок вкладки налаштування

10.5.17.10 Вкладка налаштувань

Вкладка налаштувань використовується для налаштування параметрів роботи, зміщень зонду/сенсорн панелі/лазера/камери та завантаження зовнішніх програм налагодження.

10.5.17.11 Вкладка «Утиліти»

This tab will display another tab selection of G-code utility programs (and any embedded panels):

- *Facing*: дозволяє швидке фрезерування торця визначеної області під кутами 0,45 та 90 градусів.
- *Hole Circle*: дозволяє швидко налаштувати програму для свердління окружності болтів визначеного діаметра та кількості отворів.
- *NGCGUI*: — це версія QtVCP популярного конструктора/селектора підпрограм G-коду, див. 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 Вкладка користувача

Ця вкладка буде відображатися тільки в тому випадку, якщо вбудована панель була призначена для розташування `stackedWidget_mainTab`. Якщо було призначено більше однієї вбудованої вкладки, то натискання вкладки користувача буде перемикати їх по черзі.

10.5.18 Стилі

Майже всі аспекти зовнішнього вигляду графічного інтерфейсу користувача можна налаштувати за допомогою файлу стилів QtDragon.qss. Файл можна редагувати вручну або за допомогою діалогового вікна стилів у графічному інтерфейсі користувача. Щоб викликати діалогове вікно, натисніть клавішу F12 у головному вікні. Нові стилі можна тимчасово застосувати, а потім зберегти в новому файлі qss або замінити поточний файл qss.

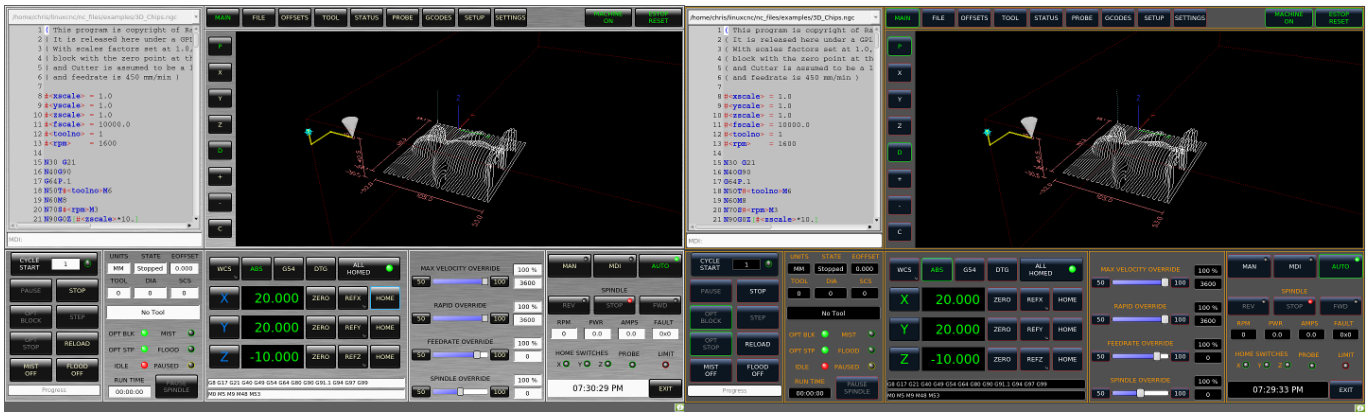


Figure 10.35: QtDragon - Два приклади стилю

10.5.19 Інтернаціоналізація

Можна створювати файли перекладу для QtDragon для відображення мовою поточної локалізації.

Для створення та/або редагування файлу перекладу необхідно, щоб LinuxCNC був встановлений як запущений на місці.

Наступне припускає, що каталог LinuxCNC git — ~/linuxcnc-dev.

Note
Якщо використовується QtDragon_hd, замініть *qtdragon_hd* на *qtdragon*

Усі мовні файли зберігаються у ~/linuxcnc-dev/share/screens/qtdragon/languages.

Файл qtdragon.py — це версія файлу графічного інтерфейсу на Python, яка використовується для перекладу.

Файли .ts – це вихідні файли перекладів. Це файли, які потрібно створити/відредагувати для кожної мови.

Файли .qm — це скомпільовані файли перекладу, що використовуються pyqt.

Мова визначається підкресленням та першими двома літерами локалі, наприклад, якщо виконується італійський переклад, то це буде `_it`. У цьому документі вона буде позначатися як `_xx`, тому `qtdragon_xx.ts` у цьому документі насправді буде `qtdragon_it.ts` для італійського перекладу.

Локаль QtDragon за замовчуванням — `_en`, що означає, що будь-які файли перекладу, створені як `qtdragon_en.*`, не будуть використані для перекладів.

Якщо будь-яка з необхідних утиліт (pyuic5, pylupdate5, linguist) не встановлена, користувачеві потрібно буде встановити необхідні інструменти розробки:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Перехід до каталогу мов:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtdragon/languages
```

Якщо до графічного інтерфейсу було внесено будь-які зміни тексту, виконайте наступну команду, щоб оновити файл графічного інтерфейсу Python:

```
pyuic5 ../qtdragon.ui > qtdragon.py
```

Користувач може створити новий файл-джерело перекладу для неіснуючої мови перекладу або змінити існуючий файл-джерело перекладу у зв’язку зі змінами, внесеними до деякого тексту у файлі-джерелі QtDragon. Якщо ви змінюєте існуючий переклад, у якому не було змін у файлі-джерелі, цей крок не є обов’язковим.

Створення або редагування файлу .ts:

```
./langfile xx
```

Note
Ця команда є скриптом, який виконує наступне: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtdragon/*.py -ts qtdragon_xx.ts

---

Редагування перекладу виконується за допомогою лінгвістичного застосунку:

```
linguist
```

1. Відкрийте файл TS та перекладіть рядки

Не обов’язково надавати переклад для кожного текстового рядка. Якщо переклад для рядка не вказано, у програмі буде використовуватися оригінальний рядок. Користувач повинен бути обережним із довжиною рядків, що з’являються на віджетах, оскільки простір обмежений. Якщо можливо, намагайтеся, щоб переклад не був довшим за оригінал.

Після завершення редагування збережіть файл:
`Файл -> Зберегти`

Потім створіть файл .qm:
`Файл -> Реліз`

QtDragon буде перекладено мовою поточної локалізації під час наступного запуску, якщо існує файл .qm цією мовою.

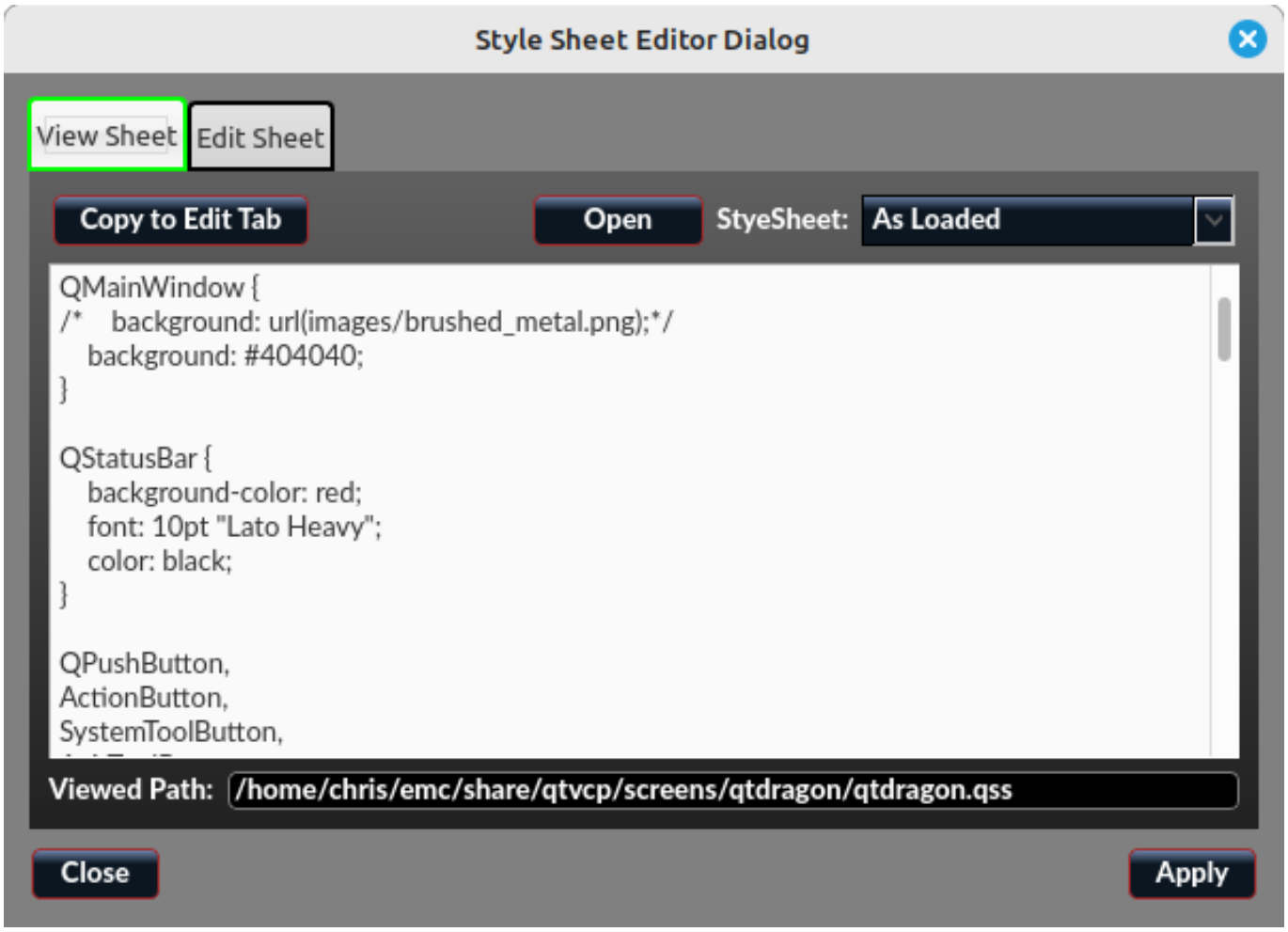
Користувачі можуть надсилати файли перекладів для включення в QtDragon. Найкращий спосіб — надіслати запит на витягнення з облікового запису користувача GitHub, як описано в документації contributing to LinuxCNC. Єдині файли, які необхідно підтвердити, — це qtdragon_xx.ts та qtdragon_xx.qm.

10.5.20 Налаштування

Загальний огляд Налаштування стокових екранів.

10.5.20.1 Таблиці стилів

Стильові таблиці можна використовувати для значної міри налаштування, але зазвичай потрібно трохи знати про назви віджетів. Натискання клавіші F12 відкриє діалогове вікно редактора стильових таблиць для завантаження/тестування/збереження змін.
Вкладка «Перегляд таблиці» дозволяє вибрати та застосувати таблицю стилів, яку QtDragon буде використовувати при першому завантаженні.
Натисніть кнопку «Копіювати у вкладку редагування», щоб скопіювати поточну таблицю стилів у вкладку редагування.
Вкладка «Редагування таблиці» дозволяє редагувати, застосовувати та зберігати зміни у відображеній таблиці стилів.



Іноді ці рядки будуть присутні, і ви можете їх змінити, інакше вам потрібно буде їх додати.

Наприклад, щоб змінити шрифт DRO (знайдіть цей запис і змініть назву шрифту):

```
DROLabel,
StatusLabel#status_rpm {
    border: 1px solid black;
    border-radius: 4px;
    font: 20pt ”Noto Mono”;
}
```

Щоб змінити шрифт та формат відображення DRO:

```
DROLabel {
    font: 25pt ”Lato Heavy”;
    qproperty-imperial_template: ’%9.5f’;
    qproperty-metric_template: ’%10.4f’;
    qproperty-angular_template: ’%11.2f’;

    /*b’’Hb’’b’’ab’’b’’лb’’b’’ab’’b’’шb’’b’’тb’’b’’уb’’b’’вb’’b’’ab’’b’’нb’’b’’нb’’b’’яb’’ ↩
        b’’пb’’b’’ab’’b’’pb’’b’’ab’’b’’мb’’b’’eb’’b’’тb’’b’’pb’’b’’ib’’b’’вb’’ b’’мb’’b’ ↩
        ’eb’’b’’нb’’b’’юb’’ */
    qproperty-showLast: true;
    qproperty-showDivide : false;
    qproperty-showGotoOrigin: false;
    qproperty-showZeroOrigin: false;
    qproperty-showSetOrigin: true;
```

```
    qproperty-dialogName: CALCULATOR;
}
```

Змініть пункти меню натискання кнопки вибору осі:

```
AxisToolButton {
    /* b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''йb''b''тb''b''eb'' b''вb''b' ↩
        'cb''b''ib'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''мb'' ↩
        b''eb''b''нb''b''юb'' */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

Щоб змінити текст кнопки «туман» на «повітря» (додайте ці рядки)

```
#action_mist{
    qproperty-true_state_string: ”Air\\nOn”;
    qproperty-false_state_string: ”Air\\nOff”;
}
```

Щоб змінити шрифт і формат відображення «Зсуви»:

```
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’;
}
```

Щоб зупинити ефект розмиття за допомогою діалогових вікон:

```
#screen_options {
    qproperty-focusBlur_option: false;
}
```

Щоб змінити кольори виділення/підсвічування статусу:

```
#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 */
}
```

Зміна кольорів/шрифтів відображення тексту G-коду:

```
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”;
}
```

Щоб кнопки ручного керування шпинделем також поступово збільшували/зменшували швидкість:

```
#action_spindle_fwd{
    qproperty-spindle_up_action: true;
}
#action_spindle_rev{
    qproperty-spindle_down_action: true;
}
```

10.5.20.2 Код Qt Designer та Python

Всі аспекти графічного інтерфейсу користувача можна повністю налаштувати за допомогою Qt Designer та/або коду Python.
Ця можливість включена в середовище розробки QtVCP.
Широке використання віджетів QtVCP дозволяє мінімізувати обсяг необхідного коду Python, що спрощує внесення змін.
На веб-сайті LinuxCNC є докладна документація щодо встановлення та використання бібліотек QtVCP.
Дивіться QtVCP для отримання додаткової інформації про QtVCP в цілому.
Користувацькі модифікації можна додати шляхом «підкласування» файлу обробника. Це додає код поверх оригіналу.
QtDragon також може використовувати файл rc QtVCP для внесення незначних модифікацій коду Python без використання користувацького файлу обробника.

Див. Зміна екранів для отримання додаткової інформації про налаштування.

Деякі налаштування віджетів доступні для базового зонда та навпаки для зонда.

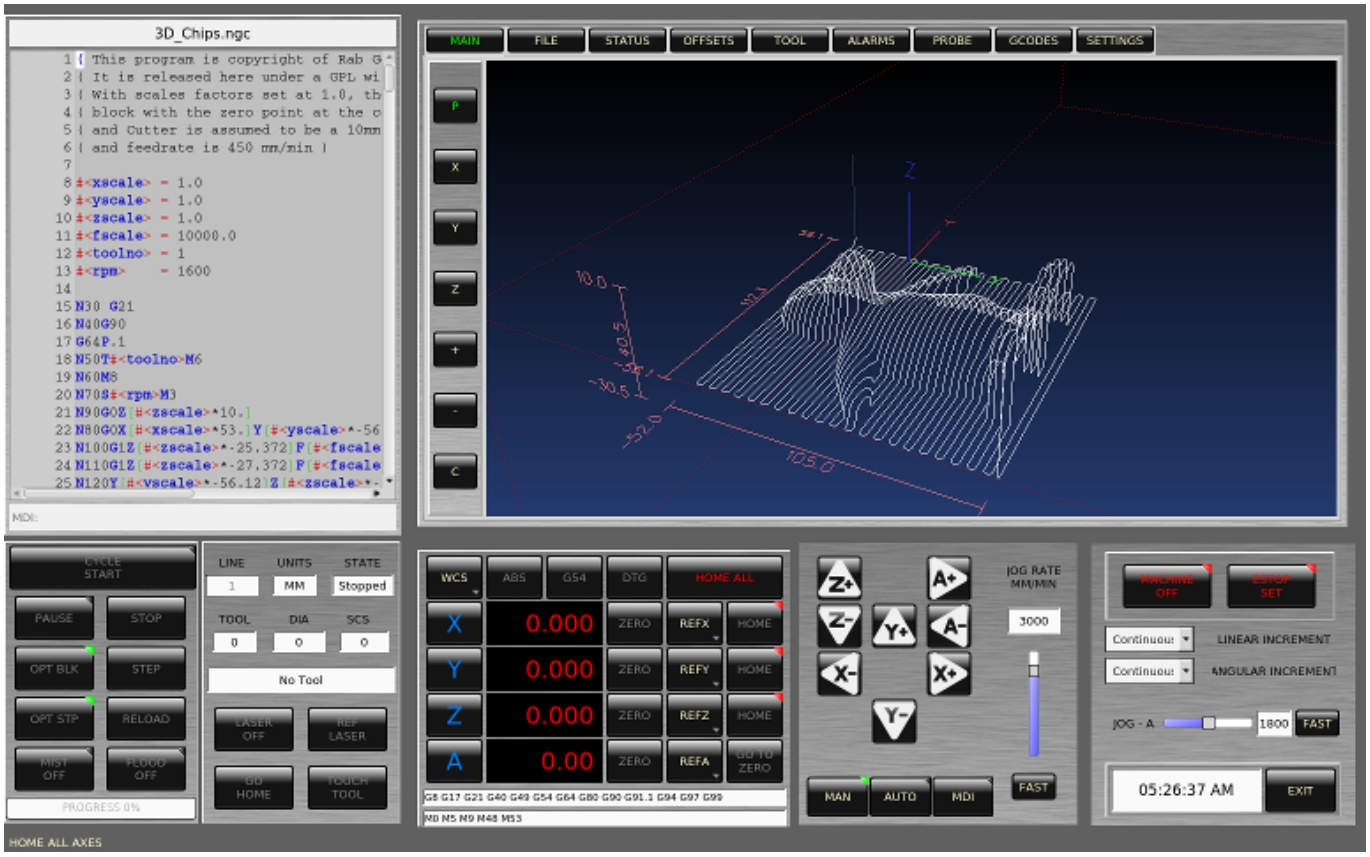


Figure 10.36: QtDragon - Налаштований QtDragon

10.6 NGCGUI

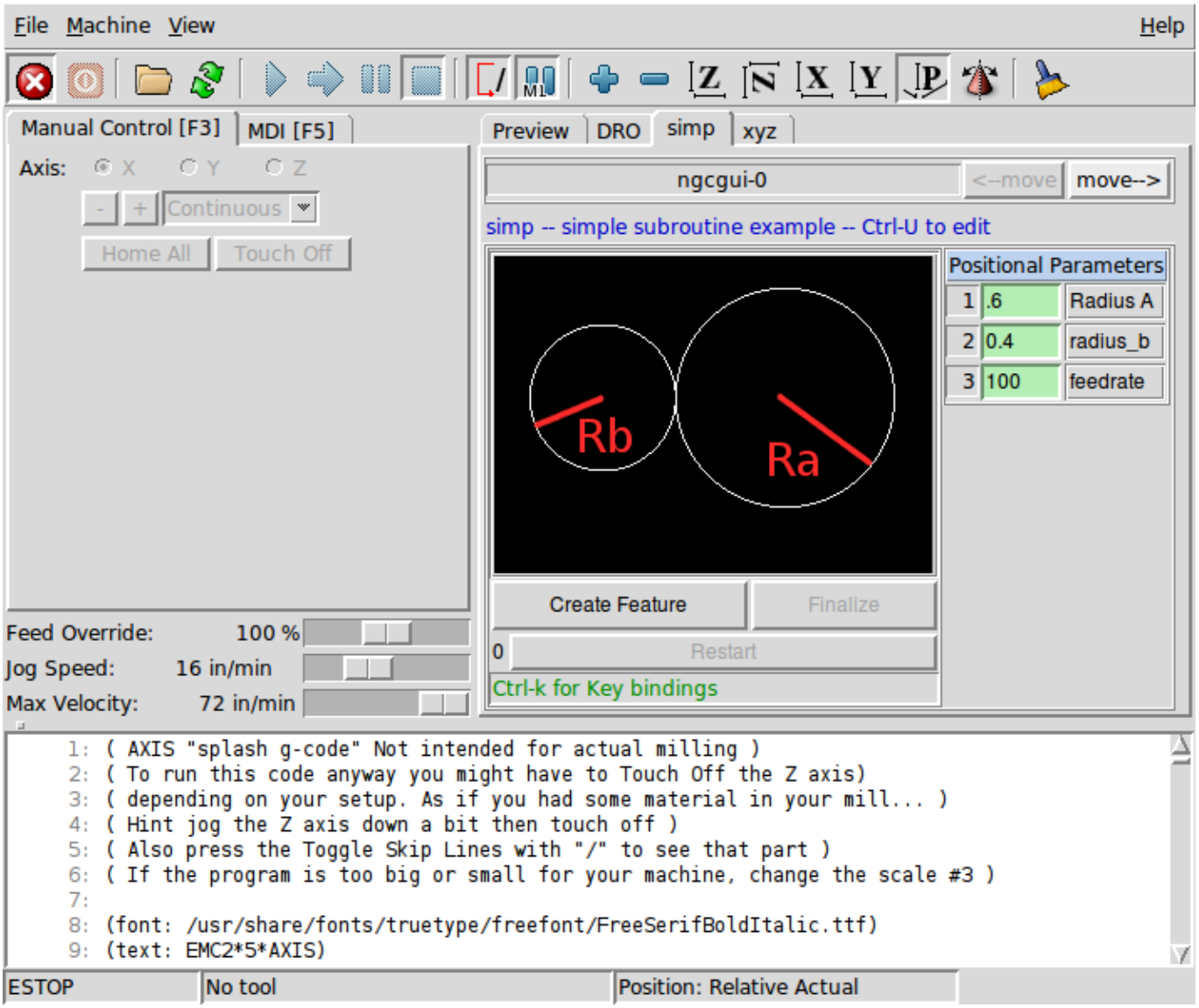


Figure 10.37: NGCGUI вбудований в AXIS

10.6.1 Огляд

* «NGCGUI» — це програма Tcl для роботи з підпрограмами. Вона дозволяє вам мати діалоговий інтерфейс з LinuxCNC. Ви можете організувати підпрограми в порядку, в якому вони повинні виконуватися, і об’єднати підпрограми в один файл для повної програми обробки деталі. * «NGCGUI» може працювати як окремий додаток або бути вбудованим у кілька вкладок графічного інтерфейсу AXIS. * «PyNGCGUI» — це альтернативна реалізація NGCGUI на Python. * «PyNGCGUI» може працювати як самостійна програма або бути вбудованою як вкладка (з власним набором підвкладок) в будь-який графічний інтерфейс, що підтримує вбудовування програм Glade-VCP AXIS, Touchy, Gscreen і GMOCCAPY.

Використання NGCGUI або PyNGCGUI:

- Для кожної підпрограми, зазначеної у файлі INI, надаються сторінки вкладок.

- Нові вкладки підпрограм можна додавати на льоту за допомогою custom tab.
- Кожна вкладка підпрограми містить поля введення для всіх параметрів підпрограми.
- Поля введення можуть мати значення за замовчуванням та мітку, які позначаються спеціальними коментарями у файлі підпрограми.
- Виклики підпрограм можна об'єднати, утворюючи програму з кількох кроків.
- Можна використовувати будь-яку однофайлову підпрограму G-коду, яка відповідає домовленостям NGCGUI.
- Можна використовувати будь-яку програму gcmc (G-code-meta-compiler), яка відповідає конвенціям NGCGUI щодо тегування змінних. (Виконуваний файл gcmc необхідно встановити окремо, див.: https://www.vagrearg.org/content/gcmc)

Note
NGCGUI та PyNGCGUI реалізують однакові функції та обробляють файли .ngc та .gcmc, які відповідають декільком специфічним для NGCGUI конвенціям. У цьому документі термін «NGCGUI» загалом стосується обох програм.

10.6.2 Демонстраційні конфігурації

Декілька демонстраційних конфігурацій знаходяться в каталозі sim зразків конфігурацій, що пропонуються конфігуратором LinuxCNC. Конфігуратор знаходиться в головному меню системи: Програми > CNC > LinuxCNC

Наведено приклади для AXIS, Touchy, gscreen та GMOCCAPY. Ці приклади демонструють як 3-осьові (XYZ) декартові конфігурації (наприклад, фрезерні верстати), так і токарні (XZ) установки. Деякі приклади показують використання спливаючої клавіатури для систем з сенсорним екраном, а інші приклади демонструють використання файлів, створених для програми gcmc (G-code Meta Compiler). Приклади Touchy також демонструють інтеграцію програми перегляду графіків Glade-VCP (gremlin_view).

Найпростіше застосування можна знайти так:

```
b''3b''b''pb''b''ab''b''зb''b''ob''b''кb'' Configurations/sim/axis/ngcgui/ngcgui_simple
```

Повний приклад, що демонструє сумісність з gcmc, знаходиться за посиланням:

```
b''3b''b''pb''b''ab''b''зb''b''ob''b''кb'' Configurations/sim/axis/ngcgui/ngcgui_gcmc
```

Повний приклад вбудованого застосунку GladeVCP з використанням gcmc знаходиться за адресою:

```
b''3b''b''pb''b''ab''b''зb''b''ob''b''кb'' Configurations/sim/gscreen/ngcgui/pyngcgui_gcmc
```

У прикладах конфігурацій симулятора використовуються бібліотечні файли, що містять файли підпрограм G-коду (.ngc) та файли метакомпілятора G-коду (.gcmc):

- *nc_files/ngcgui_lib*
  - *ngcgui.ngc* - Легкий для розуміння приклад використання підпрограм
  - *arc1.ngc* - базова дуга з використанням компенсації радіуса різця
  - *arc2.ngc* - дуга, що визначається центром, зміщенням, шириною, кутом (викликає arc1)
  - *backlash.ngc* - процедура вимірювання люфту осі за допомогою індикатора годинникового типу
  - *db25.ngc* - створює виріз для штекера DB25

  - *gosper.ngc* – демонстрація рекурсії (flowsnake)
  - *helix.ngc* - різання спіралі або D-подібного отвору
  - *helix_rtheta.ngc* - спіраль або D-подібний отвір, розташовані за радіусом та кутом
  - *hole_circle.ngc* - рівновіддалені отвори по колу
  - *ihex.ngc* - внутрішній шестигранник
  - *iquad.ngc* - внутрішній чотирикутник
  - *ohex.ngc* - зовнішній шестикутник
  - *oquad.ngc* - зовнішній чотирикутник
  - *qpex_mm.ngc* - Демонстрація qpockets (на основі мм)
  - *qpex.ngc* - Демонстрація qpockets (дюймова)
  - *qpocket.ngc* - чотиристоронння кишеня
  - *rectangle_probe.ngc* - зондувати прямокутну область
  - *simp.ngc* - простий приклад підпрограми, яка створює два кола
  - *slot.ngc* - слот від з'єднання двох кінцевих точок
  - *xyz.ngc* - тренажер обмежений формою коробки
  - *Custom* - Створює власні вкладки
  - *ttt* - True Type Tracer, для створення текстів для гравіювання
- *nc_files/ngcgui_lib/lathe*
  - *ngcgui-lathe* - Приклад підпрограми токарного верстата
  - *g76base.ngc* - Графічний інтерфейс користувача для нарізання різьби G76
  - *g76diam.ngc* - Специфікація різьблення за основним та основним діаметрами
  - *id.ngc* - розточує внутрішній діаметр
  - *od.ngc* - повертає зовнішній діаметр
  - *taper-od.ngc* - конусність на зовнішньому діаметрі
  - *Custom* - Створює власні вкладки
- *nc_files/gcmc_lib*
  - *drill.gcmc* - свердліть отвори у формі прямокутника
  - *square.gcmc* - проста демонстрація змінних тегів для файлів gcmc
  - *star.gcmc* - Демонстрація gcmc, що ілюструє функції та масиви
  - *wheels.gcmc* - Демонстрація складних шаблонів у GCMC

Щоб спробувати демонстрацію, виберіть конфігурацію симулятора та запустіть програму LinuxCNC.

Якщо ви використовуєте графічний інтерфейс AXIS, натисніть кнопку *E-Stop*  потім «Потужність машини»  потім «Головна сторінка Все». Виберіть вкладку NGCGUI, заповніть усі порожні поля розумними значеннями та натисніть «Створити функцію», а потім «Завершити». Нарешті натисніть кнопку «Виконати»  кнопку, щоб спостерігати за її роботою. Експериментуйте, створюючи кілька функцій та функцій з різних вкладок.

Щоб створити кілька підпрограм, об'єднаних в один файл, перейдіть на кожну вкладку, заповніть поля, натисніть «Створити функцію», а потім за допомогою клавіш зі стрілками перемістіть необхідні вкладки, щоб розташувати їх у потрібному порядку. Тепер натисніть «Завершити» і дайте відповідь на запит, щоб створити

Інші графічні інтерфейси матимуть подібну функціональність, але кнопки та назви можуть відрізнятися

Note
Демонстраційні конфігурації створюють вкладки лише для декількох наданих прикладів. Будь-який графічний інтерфейс з custom tab може відкрити будь-яку з підпрограм бібліотеки або будь-який файл користувача, якщо він знаходиться в шляху підпрограм LinuxCNC.
Щоб переглянути спеціальні комбінації клавіш, клацніть всередині вкладки NGCGUI, щоб отримати фокус, а потім натисніть Control-k.
Демонстраційні підпрограми повинні працювати на модельованих конфігураціях машин, що входять до дистрибутива. Користувач завжди повинен розуміти поведінку та призначення програми перед її запуском на реальній машині.

10.6.3 Розташування бібліотек

У інсталяціях LinuxCNC, встановлених з deb-пакетів, конфігурації симуляції для NGCGUI використовую символічні посилання на бібліотеки LinuxCNC, які не можуть бути записані користувачем, для:

- *nc_files/ngcgui_lib* Підфайли, сумісні з NGCGUI
- *nc_files/ngcgui_lib/lathe*, підфайли токарного верстата, сумісні з NGCGUI
- *nc_files/gcmc_lib* Програми, сумісні з NGCGUI-gcmc
- *nc_files/ngcgui_lib/utilitysubs* Допоміжні підпрограми
- *nc_files/ngcgui_lib/mfiles* Користувацькі M-файли

Ці бібліотеки розташовані за допомогою елементів INI-файлу, які вказують шляхи пошуку, що використовуються LinuxCNC (та 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
Це довгі рядки (не продовжуються на декількох рядках), які вказують каталоги, що використовуються в патчі пошуку. Імена каталогів розділяються двокрапками (:). Між іменами каталогів не повинно бути пробілів.

Користувач може створювати нові каталоги для власних підпрограм та M-файлів і додавати їх до шляху(ів) пошуку.

Наприклад, користувач може створювати каталоги з терміналу за допомогою команд:

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

А потім створіть або скопіюйте файли, надані системою, до цих каталогів, доступних для запису користувачем. Наприклад, користувач може створити сумісний з NGCGUI підфайл з назвою:

```
/home/myusername/mysubs/example.ngc
```

Щоб використовувати файли в нових каталогах, файл INI необхідно відредагувати, включивши нові підфайли та розширивши шлях(и) пошуку. Для цього прикладу:

```
[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 (та NGCGUI) використовують перший файл, знайдений під час пошуку в каталогах у шляху пошуку. Завдяки такій поведінці ви можете замінити підфайл ngcgui_lib, розмістивши підфайл з ідентичною назвою в каталозі, який знаходиться раніше в шляху пошуку. Більш детальну інформацію можна знайти в розділі INI посібника для інтеграторів.

10.6.4 Автономне використання

10.6.4.1 Автономний NGCGUI

Для використання введіть у терміналі:

```
ngcgui --help
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
  ngcgui --help | -?
  ngcgui [Options] -D <nc files directory name>
  ngcgui [Options] -i <LinuxCNC INI file name>
  ngcgui [Options]

b''Ob''b''пb''b''цb''b''ib''b''їb'':
  [-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
Як окремий застосунок, NGCGUI обробляє один файл підпрограми, який можна викликати кілька разів. Кілька окремих застосунків NGCGUI можна запускати незалежно один від одного.

10.6.4.2 Автономний PyNGCGUI

Для використання введіть у терміналі:

```
pyngcgui --help
b''Bb''b''ив''b''кb''b''ob''b''pb''b''ив''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
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
Як окремий застосунок, PyNGCGUI може зчитувати INI-файл (або запущений застосунок LinuxCNC) для створення вкладок для кількох підфайлів.

10.6.5 Вбудовування NGCGUI

10.6.5.1 Вбудовування NGCGUI в AXIS

Наступні елементи INI-файлу розміщуються в розділі [DISPLAY]. (Див. додаткові розділи нижче щодо необхідних додаткових елементів)

- *TKPKG = Ngcgui 1.0* - пакет NGCGUI
- *TKPKG = Ngcguittt 1.0* - пакет True Type Tracer для генерації тексту для гравіювання (необов’язково, має відповідати TKPKG = Ngcgui).
- *NGCGUI_FONT = Helvetica -12 normal* - Встановлює шрифт, що використовується
- *NGCGUI_PREAMBLE = in_std.ngc* - Файл преамбули, який буде додано на початок підпрограми. Коли об’єднуються кілька підпрограм, він додається лише один раз.
- *NGCGUI_SUBFILE = simp.ngc* - Створює вкладку з іменованої підпрограми.
- *NGCGUI_SUBFILE = ””* - Створює власну вкладку
- *#NGCGUI_OPTIONS = opt1 opt2 …* - Варіанти NGCGUI:
  - *nonew* — Забороняє створення нової користувацької вкладки

  - *noremove* — Забороняє видалення сторінки вкладки
  - *noauto* — Не запускати автоматично (makeFile, потім ручний запуск)
  - *noiframe* — Немає внутрішнього зображення, зображення на окремому верхньому рівні
- *TTT = truetype-tracer* - назва програми трасування TrueType (вона має бути в користувацькому PATH)
- *TTT_PREAMBLE = in_std.ngc* - Необов'язково, вказує ім'я файлу для преамбули, що використовується для підфайлів, створених ttt. (альтернатива: mm_std.ngc)

Note
Додаткові елементи truetype tracer використовуються для визначення сумісної з NGCGUI вкладки, яка використовує програму truetype-tracer. Програма truetype-tracer повинна бути встановлена окремо і знаходитися в PATH користувача.

10.6.5.2 Вбудовування PyNGCGUI як вкладки GladeVCP у графічний інтерфейс

Наступні елементи INI-файлу розміщуються в розділі [DISPLAY] для використання з графічними інтерфейсами AXIS, Gscreen або Touchy. (Додаткові необхідні елементи дивіться в розділах нижче)

EMBED_ Items

- `EMBED_TAB_NAME = PyNGCGUI` - ім'я, яке відображатиметься на вбудованій вкладці
- `EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui` - викликає GladeVCP
- `EMBED_TAB_LOCATION = name_of_location` - де розташована вбудована сторінка

Note
Специфікатор EMBED_TAB_LOCATION не використовується для графічного інтерфейсу AXIS. Хоча PyNGCGUI може бути вбудований в AXIS, інтеграція є більш повною при використанні NGCGUI (з використанням TKPKG = Ngcgui 1.0). Щоб вказати EMBED_TAB_LOCATION для інших графічних інтерфейсів, див. розділ DISPLAY Section глави «Конфігурація INI».

Note
Графічний інтерфейс трасувальника Truetype наразі недоступний для програм GladeVCP.

10.6.5.3 Додаткові елементи INI-файлу, необхідні для NCGUI або PyNGCGUI

Наступні елементи INI-файлу розміщуються в розділі [DISPLAY] для будь-якого графічного інтерфейсу, який вбудовує NGCGUI або PyNGCGUI.

- *NGCGUI_FONT = Helvetica -12 normal* - визначає назву шрифту, розмір, звичайний|жирний
- *NGCGUI_PREAMBLE = in_std.ngc* - файл преамбули, який буде додано перед підпрограмами. Під час об'єднання кількох спільних викликів підпрограм ця преамбула додається лише один раз. Для машин на базі mm використовуйте mm_std.ngc
- *NGCGUI_SUBFILE = filename1.ngc* - створює вкладку з підпрограми filename1
- *NGCGUI_SUBFILE = filename2.ngc* - створює вкладку з підпрограми filename2

- *... etc.*
- *NGCGUI_SUBFILE = gcmcname1.gcmc* - створює вкладку з файлу gcmcname1
- *NGCGUI_SUBFILE = gcmcname2.gcmc* - створює вкладку з файлу gcmcname2
- *... etc.*
- *NGCGUI_SUBFILE = ””* - створює власну вкладку, яка може відкривати будь-яку підпрограму в шляху пошуку
- *NGCGUI_OPTIONS = opt1 opt2 ...* - Параметри NGCGUI
  - *nonew* - заборонити створення нової власної вкладки
  - *noremove* - заборонити вилучення будь-якої вкладки
  - *noauto* - без автоматичного надсилання (використовуйте makeFile, а потім збережіть або надішліть вручну)
  - *noiframe* - немає внутрішнього зображення, відображати зображення на окремому віджеті верхнього рівня
  - *nom2* - Не завершуйте на m2, використовуйте термінатор %. Цей параметр усуває всі побічні ефекти завершення на m2
- *GCMC_INCLUDE_PATH = dirname1:dirname2* - пошук каталогів для файлів включення gcmc

Це приклад вбудовування NGCGUI в AXIS. Підпрограми повинні знаходитися в каталозі, вказаному в [RS274NGC]SUBROUTINE_PATH. Деякі приклади підпрограм використовують інші підпрограми, тому переконайтеся, що у каталозі SUBROUTINE_PATH є всі необхідні залежності, якщо такі є. Деякі підпрограми можуть використовувати власні M-файли, які повинні знаходитися в каталозі, вказаному в [RS274NGC]USER_M_PATH.

Метакомпілятор G-коду (gcmc) може містити такі оператори, як:

```
include(”filename.inc.gcmc”);
```

За замовчуванням gcmc включає поточний каталог, який для LinuxCNC буде каталогом, що містить файл INI LinuxCNC. Додаткові каталоги можна додати до порядку пошуку gcmc за допомогою елемента GCMC_INCLUDE_PATH.

Зразок INI-файлу на основі графічного інтерфейсу AXIS

```
[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 b’’мb’’b’’ab’’b’’єb’’ b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’дb’’b’’yb’’b’’вb’’b’’ab’’b’ ↩
    ’тb’’b’’иb’’ Ngcguittt

NGCGUI_FONT       = Helvetica -12 normal
# b’’вb’’b’’кb’’b’’ab’’b’’зb’’b’’yb’’b’’йb’’b’’тb’’b’’eb’’ b’’лb’’b’’иb’’b’’шb’’b’’eb’’ b’ ↩
    ’ib’’b’’мb’’b’’eb’’b’’нb’’b’’ab’’ b’’фb’’b’’ab’’b’’йb’’b’’лb’’b’’ib’’b’’вb’’, b’’фb’’b’ ↩
    ’ab’’b’’йb’’b’’лb’’b’’иb’’ b’’пb’’b’’ob’’b’’вb’’b’’иb’’b’’нb’’b’’нb’’b’’ib’’ b’’бb’’b’ ↩
    ’yb’’b’’тb’’b’’иb’’ b’’вb’’ [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
Вищезазначене не є повним INI-файлом графічного інтерфейсу AXIS – показані елементи використовуються NGCGUI. Для повного INI-файлу LinuxCNC потрібно багато додаткових елементів.

10.6.5.4 Трасер Truetype

Ngcgui_ttt забезпечує підтримку truetype-tracer (v4). Він створює вкладку AXIS, яка дозволяє користувачеві створити нову вкладку NGCGUI після введення тексту та вибору шрифту та інших параметрів. (Truetype-tracer повинен бути встановлений окремо).

Щоб вбудувати ngcgui_ttt в AXIS, вкажіть наступні елементи на додаток до елементів NGCGUI:

```
Item:    [DISPLAY]TKPKG = Ngcgui_ttt version_number
Example: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
b''Пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'':    b''Ob''b''бb''b''ob''b''вb''b ↩
    ''яb''b''зb''b''кb''b''ob''b''вb''b''иb''b''йb'', b''вb''b''кb''b''ab''b''зb''b''уb''b' ↩
    'єb'' b''нb''b''ab'' b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b' ↩
    'нb''b''нb''b''яb'' ngcgui_ttt b''нb''b''ab'' b''вb''b''кb''b''лb''b''ab''b''дb''b''цb'' ↩
    b''ib'' AXIS b''зb'' b''нb''b''ab''b''зb''b''вb''b''ob''b''юb'' ttt.
         Must follow the TKPKG = Ngcgui item.

b''Eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb'':    [DISPLAY]TTT = b''шb''b''лb''b''яb''b ↩
    ''хb''_b''дb''b''ob''_truetype-tracer
b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'': [DISPLAY]TTT = truetype-tracer
b''Пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'':    b''Hb''b''eb''b''ob''b''бb''b' ↩
    'ob''b''вb''b''яb''b''зb''b''кb''b''ob''b''вb''b''ob'', b''яb''b''кb''b''щb''b''ob'' b' ↩
    'нb''b''eb'' b''вb''b''кb''b''ab''b''зb''b''ab''b''нb''b''ob'', b''cb''b''пb''b''pb''b' ↩
    'ob''b''бb''b''уb''b''йb''b''тb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b' ↩
    'cb''b''тb''b''ob''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb'' /usr/local/bin/truetype- ↩
    tracer.
         b''Bb''b''кb''b''ab''b''жb''b''ib''b''тb''b''ьb'' b''ab''b''бb''b''cb''b''ob''b' ↩
             'лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''шb''b''лb''b''яb''b''хb'' b''ab''b' ↩
             'бb''b''ob'' b''пb''b''pb''b''ob''b''cb''b''тb''b''eb'' b''ib''b''мb''b''яb''  ↩
             b''вb''b''иb''b''кb''b''ob''b''нb''b''уb''b''вb''b''ab''b''нb''b''ob''b''гb''b' ↩
             'ob'' b''фb''b''ab''b''йb''b''лb''b''уb'',
```

```
        b''yb'' b''цb''b''ьb''b''ob''b''мb''b''yb'' b''вb''b''иb''b''пb''b''ab''b''дb''b' ↩
            'кb''b''yb'' b''дb''b''лb''b''яb'' b''пb''b''ob''b''шb''b''yb''b''кb''b''yb'' b ↩
            ''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb'' b''бb''b''yb''b''дb''b' ↩
            'eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b' ↩
            'ob'' b''cb''b''eb''b''pb''b''eb''b''дb''b''ob''b''вb''b''иb''b''щb''b''eb''  ↩
            PATH b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''yb''b''вb''b''ab''b''чb''b' ↩
            'ab''.

b''Eb''b''лb''b''eb''b''мb''b''eb''b''нb''b''тb'': [DISPLAY]TTT_PREAMBLE =  ↩
    preambule_filename
b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'': [DISPLAY]TTT_PREAMBLE = in_std.ngc
b''Пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'': b''Hb''b''eb''b''ob''b''бb''b' ↩
    'ob''b''вb''b''яb''b''зb''b''кb''b''ob''b''вb''b''ob'', b''вb''b''кb''b''ab''b''зb''b' ↩
    'yb''b''єb'' b''ib''b''мb''b''яb'' b''фb''b''ab''b''йb''b''лb''b''yb'' b''дb''b''лb''b' ↩
    'яb'' b''пb''b''pb''b''eb''b''ab''b''мb''b''бb''b''yb''b''лb''b''иb'', b''щb''b''ob'' b' ↩
    'вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb''b' ↩
    'тb''b''ьb''b''cb''b''яb'' b''дb''b''лb''b''яb'' b''cb''b''тb''b''вb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''иb''b''xb'' ttt b''пb''b''ib''b''дb''b''фb''b''ab''b''йb''b''лb''b''ib''b ↩
    ''вb''.
```

10.6.5.5 Специфікації шляху до INI-файлу

NGCGUI використовує шлях пошуку LinuxCNC для пошуку файлів. Шлях пошуку починається зі стандартного каталогу, визначеного за допомогою:

```
[DISPLAY]PROGRAM_PREFIX = directory_name
```

а потім кілька каталогів, визначених за допомогою:

```
[RS274NGC]SUBROUTINE_PATH = directory1_name:directory1_name:directory3_name ...
```

Довідники Каталоги можуть бути вказані як абсолютні або відносні шляхи.

- Приклад: `[DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files`
- Приклад: `[DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/nc_files`
- Приклад: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`

Абсолютні шляхи Абсолютний шлях, що починається з ”/”, вказує повне розташування файлової системи. Шлях, що починається з ”~/”, вказує шлях, що починається з домашнього каталогу користувача. Шлях, що починається з ”~username/”, вказує шлях, що починається з домашнього каталогу користувача username.

Відносні шляхи Відносні шляхи базуються на стартовому каталозі, який є каталогом, що містить файл INI. Використання відносних шляхів може полегшити переміщення конфігурацій, але вимагає хорошого розуміння специфікаторів шляхів Linux.

- `./d0` те саме, що й d0, наприклад, каталог з іменем d0 у каталозі автозавантаження
- `../d1` посилається на каталог d1 у батьківському каталозі
- `../../d2` посилається на каталог d2 у батьківському елементі батьківського каталогу
- `../../../d3` etc.

За допомогою [RS274NGC]SUBROUTINE_PATH можна вказати кілька каталогів, розділивши їх двокрапкою. Наступний приклад ілюструє формат для декількох каталогів і показує використання відносних та абсолютних шляхів.

Приклад кількох каталогів:

```
[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs ↩
    :/tmp/tmpngc
```

Це один довгий рядок, не продовжуйте на кількох рядках. Коли LinuxCNC та/або NGCGUI шукають файли, використовується перший знайдений файл.

LinuxCNC (та NGCGUI) повинні мати можливість знаходити всі підпрограми, включаючи допоміжні підпрограми, які викликаються з підфайлів NGCGUI. Зручно розміщувати підпрограми утиліт в окремому каталозі, як показано у прикладі вище.

Дистрибутив містить каталог ngcgui_lib та демонстраційні файли для преамбул, підфайлів, пост-амбул та допоміжних файлів. Щоб змінити поведінку файлів, ви можете скопіювати будь-який файл і розмістити його в попередній частині шляху пошуку. Першим каталогом, який шукається, є [DISPLAY]PROGRAM_PREFIX. Ви можете використовувати цей каталог, але краще створити спеціальні каталоги та розмістити їх на початку [RS274NGC]SUBROUTINE_PATH.

У наступному прикладі файли з /home/myname/linuxcnc/mysubs будуть знайдені раніше за файли з ../../nc_files/ngcgui_lib.

Приклад додавання каталогу користувачів:

```
[RS274NGC]SUBROUTINE_PATH = /home/myname/linuxcnc/mysubs:../../nc_files/ngcgui_lib:../../ ↩
    nc_files/ngcgui_lib/utilitysubs‘
```

Нові користувачі можуть ненавмисно спробувати використовувати файли, які не відповідають вимогам NGCGUI. NGCGUI, ймовірно, повідомить про численні помилки, якщо файли не кодовані відповідно до його конвенцій. Згідно з передовою практикою, підфайли, сумісні з NGCGUI, слід розміщувати в спеціальному каталозі, а преамбули, пост-амбули та допоміжні файли — в окремому каталозі (каталогах), щоб уникнути спроб використовувати їх як підфайли. Файли, не призначені для використання як підфайли, можуть містити спеціальний коментар: «(not_a_subfile)», щоб NGCGUI автоматично відхиляв їх із відповідним повідомленням.

10.6.5.6 Зведена інформація про елементи INI-файлу для використання NGCGUI

[RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...
Приклад: `[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_l`
Примітка: Необов’язково, але дуже корисно для впорядкування підфайлів та файлів утиліт.

[RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...
Приклад: `[RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles`
Примітка: Необов’язково, потрібно для пошуку власних M-файлів користувача.

[DISPLAY]EMBED_TAB_NAME = ім’я для відображення на вбудованій вкладці
Приклад: `[DISPLAY]EMBED_TAB_NAME = Pyngcgui`
Примітка: Записи: `EMBED_TAB_NAME`, `EMBED_TAB_COMMAND`, `EMBED_TAB_LOCATION` визначають вбудовану програму для кількох графічних інтерфейсів LinuxCNC.

[DISPLAY]EMBED_TAB_COMMAND = назва програми, а потім аргументи
Приклад: `[DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui`
Примітка: Щодо програм GladeVCP див. Розділ GladeVCP.

[DISPLAY]EMBED_TAB_LOCATION = name_of_location
Приклад: `[DISPLAY]EMBED_TAB_LOCATION = notebook_main`
Примітка: Див. приклади INI-файлів для можливих розташувань.
Не потрібно для графічного інтерфейсу AXIS.

[DISPLAY]PROGRAM_PREFIX = dirname
Приклад: `[DISPLAY]PROGRAM_PREFIX = ../../nc_files`
Примітка: Обов’язковий та необхідний для численних функцій LinuxCNC.
Це перший каталог, який використовується під час пошуку файлів.

[DISPLAY]TKPKG = NGCGUI version_number
Приклад: `[DISPLAY]TKPKG = Ngcgui 1.0`
Примітка: Потрібно лише для вбудовування графічного інтерфейсу AXIS.
Визначає завантаження вкладок NGCGUI AXIS.

[DISPLAY]NGCGUI_FONT = font_descriptor
Приклад: [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
Примітка: Опціонально, font_descriptor є tcl-сумісним специфікатором шрифту з елементами для fonttype -fontsize fontweight.
За замовчуванням: Helvetica -10 normal.
Менші розміри шрифту можуть бути корисними для невеликих екранів.
Більші розміри шрифту можуть бути корисними для додатків з сенсорним екраном.

[DISPLAY]NGCGUI_SUBFILE = subfile_filename
Приклад: `[DISPLAY]NGCGUI_SUBFILE = simp.ngc`
Приклад: `[DISPLAY]NGCGUI_SUBFILE = square.gcmc`
Приклад: `[DISPLAY]NGCGUI_SUBFILE = ""`
Примітка: Використовуйте один або кілька елементів, щоб вказати сумісні з NGCGUI підфайли або програми gcmc, які вимагають вкладки під час запуску.
Вкладка «Custom» буде створена, якщо ім'я файлу «».
Користувач може використовувати вкладку «Custom», щоб переглянути файлову систему та визначити файли преамбули, підфайли та пост-амбули.

[DISPLAY]NGCGUI_PREAMBLE = preamble_filename
Приклад: `[DISPLAY]NGCGUI_PREAMBLE = in_std.ngc`
Примітка: Необов'язково, якщо вказано, файл додається до підфайлу.
Файли, створені за допомогою вкладок «Custom», використовують преамбулу, вказану на сторінці.

[DISPLAY]NGCGUI_POSTAMBLE = postamble_filename
Приклад: `[DISPLAY]NGCGUI_POSTAMBLE = bye.ngc`
Примітка: Необов'язково, якщо вказано, файл додається до підфайлів.
Файли, створені за допомогою вкладок «Custom», використовують пост-амбле, вказаний на сторінці.

[DISPLAY]NGCGUI_OPTIONS = opt1 opt2 ...
Приклад: [DISPLAY]NGCGUI_OPTIONS = nonew noremove
Примітка: Кілька опцій розділяються пробілами.
За замовчуванням NGCGUI налаштовує вкладки таким чином, щоб:
1) користувач може створювати нові вкладки;
2) користувач може вилучати вкладки (окрім останньої, що залишилася);
3) фіналізовані файли автоматично надсилаються до LinuxCNC;
4) Для відображення зображення для підфайлу (якщо зображення надано) надається фрейм зображення (iframe);
5) Файл результатів NGCGUI, надісланий до LinuxCNC, завершується кодом M2 (і спричиняє побічні ефекти M2).

Опції `nonew`, `noremove`, `noauto`, `noiframe`, `nom2` відповідно вимикають ці типові поведінки.

За замовчуванням, якщо файл зображення (.png,.gif,jpg,pgm) знаходиться в тому ж каталозі, що і підфайл, зображення відображається в iframe. Вказавши опцію `noiframe`, ви отримаєте доступ до додаткових кнопок для вибору преамбули, підфайлу та пост-амбули, а також до додаткових прапорців. Вибір прапорців завжди доступний за допомогою спеціальних клавіш:
`Ctrl-R` Перемикання «Зберегти значення при читанні підфайлу»,
`Ctrl-E` Перемикання «Розгорнути підпрограму»,
`Ctrl-a` Перемикання «Автовідправлення»,
`Ctrl-k` Перелік усіх клавіш і функцій.

Якщо вказано `noiframe` і знайдено файл зображення, зображення відображається в окремому вікні, а всі функції доступні на вкладці. Параметри `NGCGUI_OPTIONS` застосовуються до всіх вкладок NGCGUI, за винятком того, що параметри `nonew`, `noremove` та `noiframe` не застосовуються до вкладок «Custom». Не використовуйте вкладки «Custom», якщо ви хочете обмежити можливість користувача вибирати підфайли або створювати додаткові вкладки.

[DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...
Приклад: `[DISPLAY]GCMC_INCLUDE_PATH = /home/myname/gcmc_includes:/home/myname/gcmc_inc`
Примітка: Необов'язково, кожен каталог буде включено при виклику gcmc з використанням опції: `--include dirname`.

10.6.6 Вимоги до файлів для сумісності з NGCGUI

10.6.6.1 Вимоги до підпрограми однофайлового G-коду (.ngc)

Субфайл, сумісний з NGCGUI, містить одне визначення підпрограми. Ім'я підпрограми повинно збігатися з іменем файлу (без суфікса .ngc). LinuxCNC підтримує іменовані або пронумеровані підпрограми, але тільки іменовані підпрограми сумісні з NGCGUI. Для отримання додаткової інформації див. розділ O-Codes.

Перший рядок без коментарів має бути оператором `sub`.
Останній рядок без коментарів має бути оператором `endsub`.

examp.ngc:

```
(info: info_text_to_appear_at_top_of_tab_page)
; b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb'', b''щb''b''ob'' b''пb''b''ob''b' ↩
    'чb''b''иb''b''нb''b''ab''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' b''кb''b''pb''b' ↩
    'ab''b''пb''b''кb''b''иb'' b''зb'' b''кb''b''ob''b''мb''b''ob''b''юb''
( b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb'', b''щb''b''ob'' b''вb''b''иb''b' ↩
    'кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb'' b''дb''b''yb''b' ↩
    'жb''b''кb''b''иb'')
o<examp> sub
  BODY_OF_SUBROUTINE
o<examp> endsub
; b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb'', b''щb''b''ob'' b''пb''b''ob''b' ↩
    'чb''b''иb''b''нb''b''ab''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' b''кb''b''pb''b' ↩
    'ab''b''пb''b''кb''b''иb'' b''зb'' b''кb''b''ob''b''мb''b''ob''b''юb''
( b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb'', b''щb''b''ob'' b''вb''b''иb''b' ↩
    'кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb'' b''дb''b''yb''b' ↩
    'жb''b''кb''b''иb'')
```

Тіло підпрограми повинно починатися з набору операторів, що визначають локальні іменовані параметри для кожного позиційного параметра, очікуваного для виклику підпрограми. Ці визначення повинні бути послідовними, починаючи з № 1 і закінчуючи останнім використаним номером параметра. Визначення повинні бути надані для кожного з цих параметрів (без пропусків).

Нумерація параметрів

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC вважає всі пронумеровані параметри в діапазоні від #1 до #30 параметрами виклику, тому NGCGUI надає поля введення для будь-якого входження параметрів у цьому діапазоні. Рекомендується уникати використання пронумерованих параметрів від #1 до #30 в будь-якому іншому місці підпрограми. Для всіх внутрішніх змінних рекомендується використовувати локальні іменовані параметри.

Кожен визначальний оператор може за бажанням містити спеціальний коментар та значення за замовчуванням для параметра.

Прототип заяви

```
#<vname> = #n (=default_value)
or
#<vname> = #n (comment_text)
or
#<vname> = #n (=default_value comment_text)
```

Приклади параметрів

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z start setting)
```

Якщо вказано значення_за_замовчуванням, воно буде введено в поле введення параметра під час запуску. Якщо включено текст_коментара, він буде використано для ідентифікації вхідних даних замість назви параметра.

Глобальні іменовані параметри Примітки щодо глобальних іменованих параметрів та NGCGUI:

(Глобальні іменовані параметри мають початкове підкреслення в назві, наприклад #<_someglobalname>)

Як і в багатьох мовах програмування, використання глобальних змінних є потужним інструментом, але часто може призводити до несподіваних наслідків. У LinuxCNC існуючі глобальні іменовані параметри будуть дійсними під час виконання підпрограм, а підпрограми можуть змінювати або створювати глобальні іменовані параметри.

Не рекомендується передавати інформацію до підпрограм за допомогою глобальних іменованих параметрів, оскільки таке використання вимагає створення та підтримки чітко визначеного глобального контексту, який важко підтримувати. Використання пронумерованих параметрів від #1 до #30 як вхідних даних підпрограм повинно бути достатнім для задоволення широкого спектру вимог до проектування. NGCGUI підтримує деякі глобальні іменовані параметри введення, але їх використання застаріло та тут не задокументовано.

Хоча введення глобальних іменованих параметрів не рекомендується, підпрограми LinuxCNC повинні використовувати глобальні іменовані параметри для повернення результатів. Оскільки підфайли, сумісні з NGCGUI, призначені для використання в графічному інтерфейсі, значення повернення не є загальною вимогою. Однак NGCGUI корисний як інструмент тестування підпрограм, які повертають глобальні іменовані параметри, і підфайли, сумісні з NGCGUI, зазвичай викликають файли підпрограм утиліт, які повертають результати з глобальними іменованими параметрами.

Для підтримки цих випадків використання NGCGUI ігнорує глобальні іменовані параметри, що містять у своїй назві символ двокрапки (:). Використання двокрапки (:) у назві заважає NGCGUI створювати поля введення для цих параметрів.

Приклад глобальних іменованих параметрів

```
o<examp> sub
...
#<_examp:result> = #5410       (b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b' ↩
    'иb'' b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''иb''b''йb'' b''дb''b''іb''b''аb''b' ↩
    'мb''b''eb''b''тb''b''pb'' b''іb''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b' ↩
    'нb''b''тb''b''yb'')
...
o<helper> call [#<x1>] [#<x2>] (call a subroutine)
#<xresult> = #<_helper:answer> (b''нb''b''eb''b''гb''b''аb''b''йb''b''нb''b''ob'' b''лb''b' ↩
    'ob''b''кb''b''аb''b''лb''b''іb''b''зb''b''yb''b''вb''b''аb''b''тb''b''иb'' b''гb''b' ↩
    'лb''b''ob''b''бb''b''аb''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''зb''b' ↩
    'yb''b''лb''b''ьb''b''тb''b''аb''b''тb'' b''дb''b''ob''b''пb''b''ob''b''мb''b''іb''b' ↩
    'жb''b''нb''b''ob''b''їb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''іb''b''їb'')
```

```
#<_helper:answer> = 0.0        (b''ob''b''6b''b''нb''b''yb''b''лb''b''иb''b''тb''b''иb'' b' ↩
    'гb''b''лb''b''ob''b''6b''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''ib''b''мb''b' ↩
    'eb''b''нb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'мb''b''eb''b''тb''b''pb'', b''яb''b''кb''b''иb''b''йb'' b''вb''b''иb''b''кb''b''ob''b' ↩
    'pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''юb'')
...
o<examp> endsub
```

У наведеному вище прикладі підпрограма утиліти буде знаходитися в окремому файлі з назвою helper.ngc. Допоміжна процедура повертає результат у глобальному іменованому параметрі з назвою #<_helper:answer.

Для дотримання належної практики підфайл, що викликає, негайно локалізує результат для використання в інших місцях підфайлу, а глобальний іменований параметр, що використовується для повернення результату, анулюється з метою зменшення ймовірності його ненавмисного використання в інших місцях глобального контексту. Значення анулювання 0,0 не завжди може бути хорошим вибором.

NGCGUI підтримує створення та об'єднання декількох функцій для підфайлу та для декількох підфайлів. Іноді для підфайлів корисно визначати їх порядок під час виконання, тому NGCGUI вставляє спеціальний глобальний параметр, який можна перевірити в підпрограмах. Параметр має назву #<_feature:>. Його значення починається з 0 і збільшується для кожної доданої функції.

Додаткові функції Спеціальний коментар «info» можна включити будь-де в підфайл, сумісний з NGCGUI. Формат такий:

```
(info: info_text)
```

Інформаційний текст (info_text) відображається у верхній частині вкладки NGCGUI в AXIS.

Файли, не призначені для використання як підфайли, можуть містити спеціальний коментар, щоб NGCGUI автоматично відхиляв їх відповідним повідомленням.

```
(not_a_subfile)
```

До підфайлу можна додати додатковий файл зображення (.png, .gif, .jpg, .pgm). Файл зображення може допомогти уточнити параметри, що використовуються підфайлом. Файл зображення повинен знаходитися в тому ж каталозі, що і підфайл, і мати те ж ім'я з відповідним розширенням для зображень, наприклад, підфайл example.ngc може супроводжуватися файлом зображення example.png. NGCGUI намагається змінити розмір великих зображень шляхом під вибірки до розміру з максимальною шириною 320 і максимальною висотою 240 пікселів.

Жодна з умовних домовленостей, необхідних для створення підфайлу, сумісного з NGCGUI, не виключає його використання як файлу підпрограм загального призначення для LinuxCNC.

Дистрибутив LinuxCNC містить бібліотеку (каталог ngcgui_lib), яка включає як приклади підфайлів, сумісних з NGCGUI, так і файли утиліт для ілюстрації можливостей підпрограм LinuxCNC та використання NGCGUI. Інша бібліотека (gcmc_lib) містить приклади файлів підпрограм для метакомпілятора G-коду (gcmc).

Додаткові підпрограми, створені користувачем, можна знайти на форумі в розділі «Підпрограми».

10.6.6.2 Вимоги до файлу G-code-meta-compiler (.gcmc)

Файли для Gcode-meta-compiler (gcmc) зчитуються NGCGUI, який створює поля введення для змінних, позначених у файлі. Коли функція для файлу завершена, NGCGUI передає файл як вхідні дані до компілятора gcmc, і, якщо компіляція проходить успішно, отриманий файл G-коду надсилається до LinuxCNC для виконання. Результуючий файл форматується як підпрограма з одним файлом; файли .gcmc і .ngc можуть бути змішані NGCGUI.

Змінні, визначені для включення до NGCGUI, позначені рядками, які відображатимуться як коментарі для компілятора gcmc.

Формати змінних тегів

```
//ngcgui: varname1 =
//ngcgui: varname2 = value2
//ngcgui: varname3 = value3, label3;
```

Приклади змінних тегів

```
//ngcgui: zsafe =
//ngcgui: feedrate = 10
//ngcgui: xl = 0, x limit
```

У цих прикладах поле введення для varname1 не матиме значення за замовчуванням, поле введення для varname2 матиме значення за замовчуванням value2, а поле введення для varname 3 матиме значення за замовчуванням value 3 і мітку label3 (замість varname3). Значення за замовчуванням повинні бути числами.

Щоб полегшити модифікацію дійсних рядків у файлі gcmc, допускаються альтернативні формати тегів. Альтернативні формати ігнорують кінцеві крапки з комою (;) та кінцеві маркери коментарів (//). Завдяки цьому часто можна просто додати тег //ngcgui: до існуючих рядків у файлі .gcmc.

Альтернативні формати змінних тегів

```
//ngcgui: varname2 = value2;
//ngcgui: varname3 = value3; //, label3;
```

Приклади альтернативних змінних тегів

```
//ngcgui: feedrate = 10;
//ngcgui: xl = 0; //, x limit
```

Інформаційний рядок, який відображатиметься у верхній частині сторінки вкладки, може бути додатково позначений рядком з таким тегом:

Інформаційний день

```
//ngcgui: info: text_to_appear_at_top_of_tab_page
```

За потреби, опції можна передати компілятору gcmc за допомогою рядка з тегом:

Формат тегу рядка опції

```
//ngcgui: -option_name [ [=] option_value]
```

Приклади тегів рядка опцій

```
//ngcgui: -I
//ngcgui: --imperial
//ngcgui: --precision 5
//ngcgui: --precision=6
```

Параметри для gcmc доступні за допомогою команди терміналу:

```
gcmc --help
```

Програма gcmc за замовчуванням використовує метричний режим. Режим можна встановити в дюймах за допомогою налаштування опції:

```
//ngcgui: --imperial
```

Файл преамбули, якщо він використовується, може встановити режим (g20 або g21), який конфліктує з режимом, що використовується файлом gcmc. Щоб переконатися, що режим програми gcmc діє, включіть наступну інструкцію у файл .gcmc:

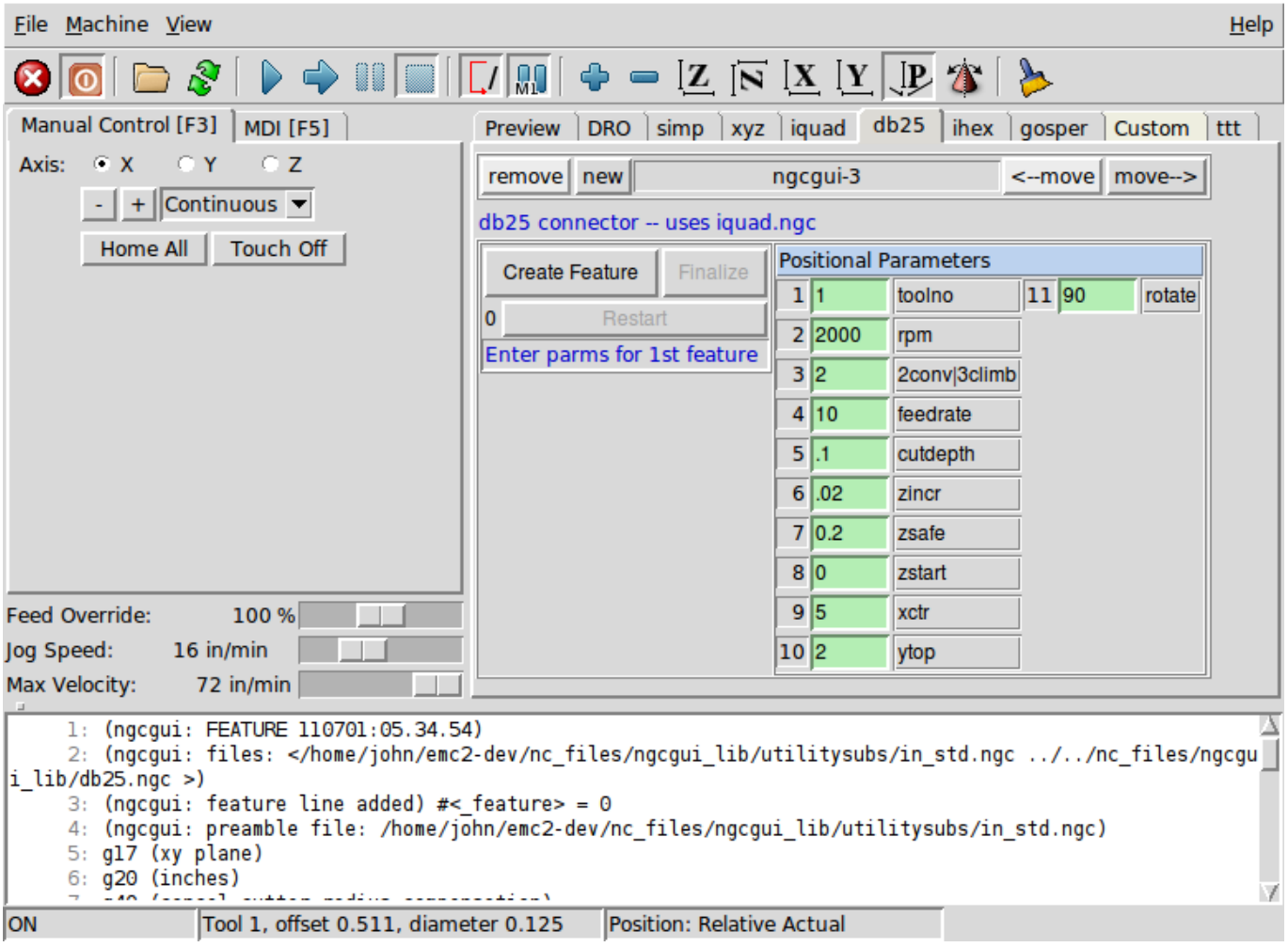
```
include("ensure_mode.gcmc")
```

та вкажіть правильний шлях для gcmc include_files у INI-файлі, наприклад:

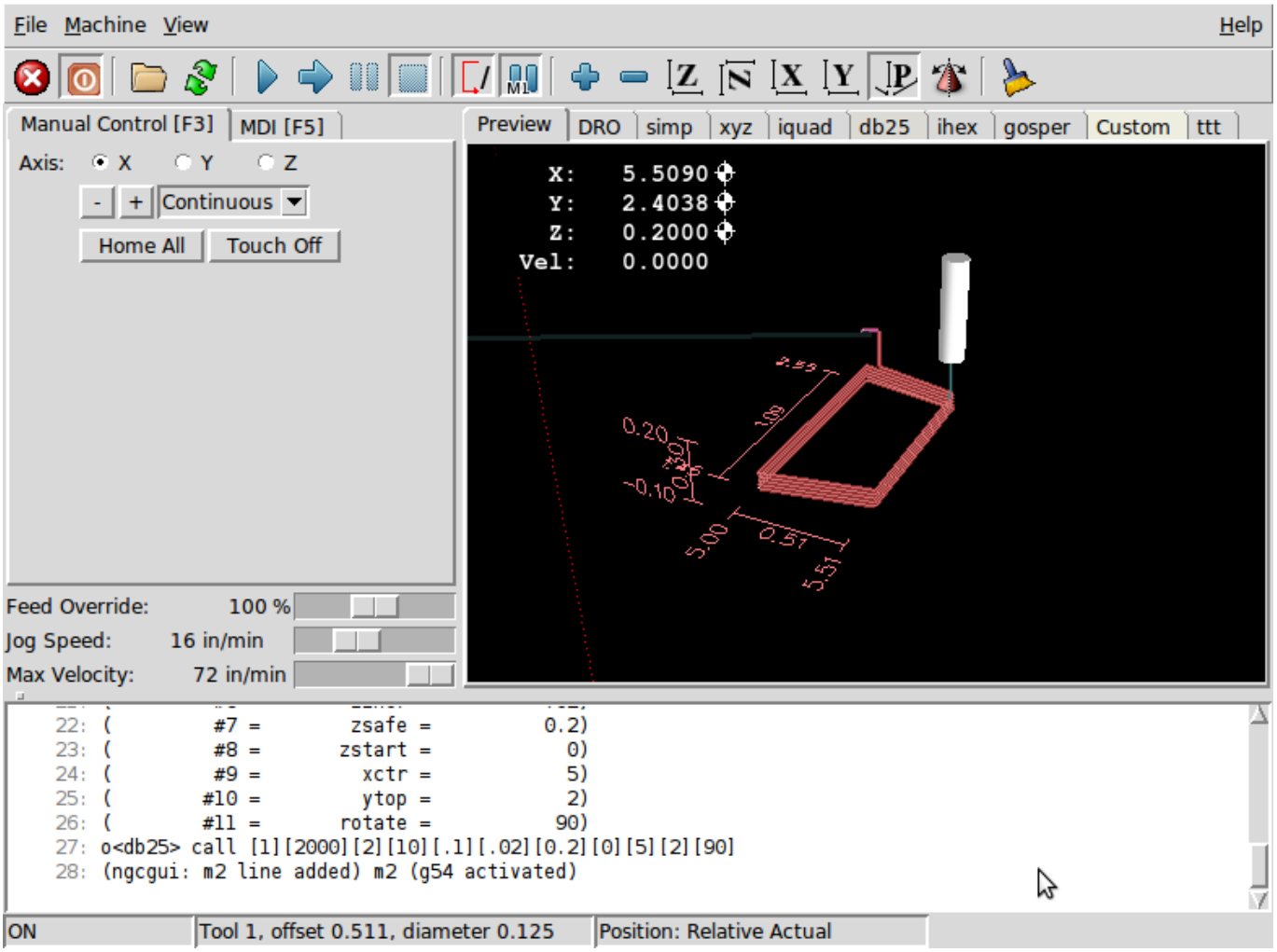
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

10.6.7 Приклад DB25

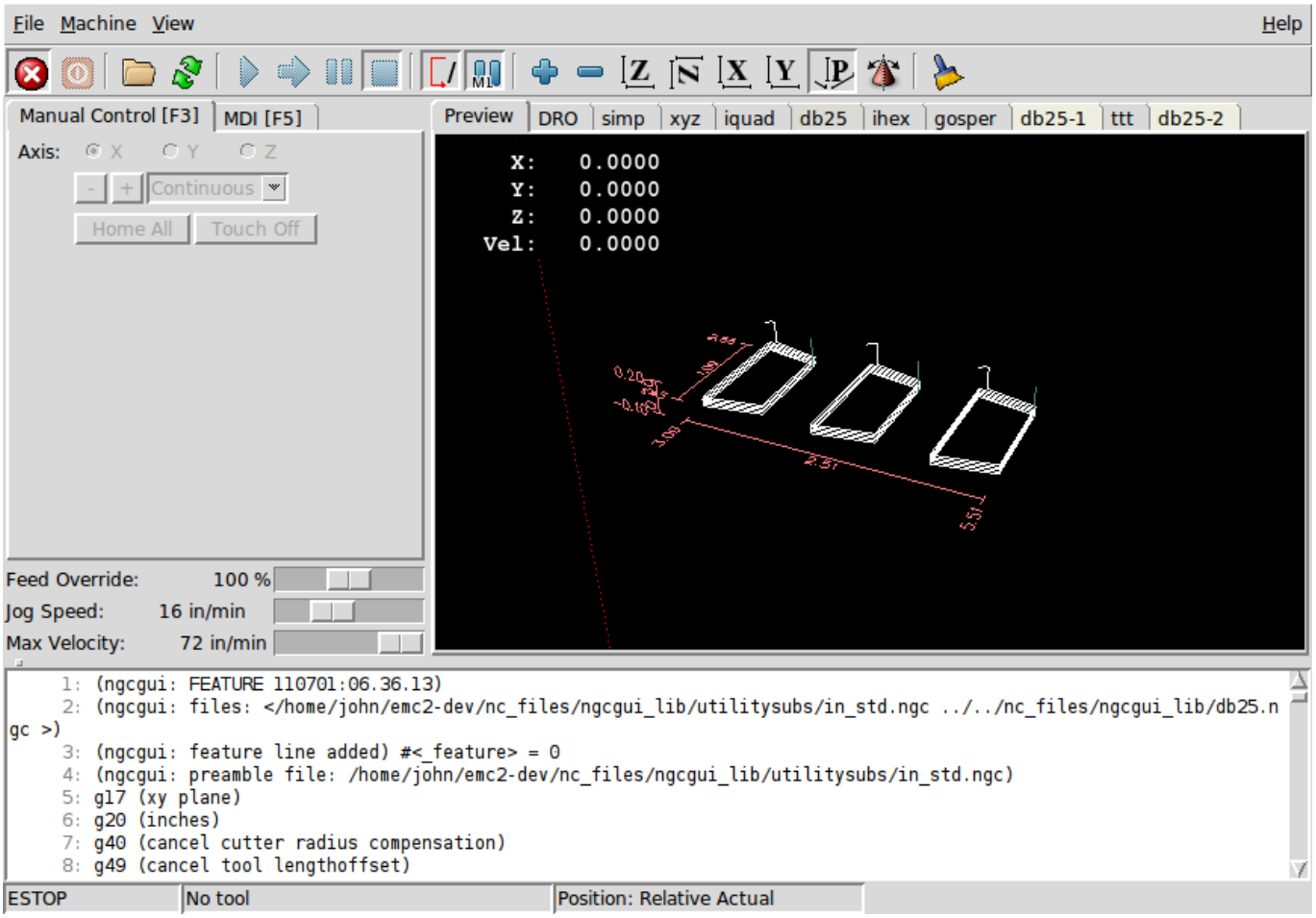
Далі показано підпрограму DB25. На першому фото ви бачите, де потрібно заповнити пробіли для кожної змінної.



На цьому фото показано задню схему підпрограми DB25.



На цьому фото показано використання нової кнопки та вкладки ”Налаштування” для створення трьох вирізів DB25 в одній програмі.



10.6.8 Створення підпрограми

- Для створення підпрограми для використання з NGCGUI ім’я файлу та ім’я підпрограми мають збігатися.
- Файл має бути поміщений у підкаталог, на який вказує INI-файл.
- У першому рядку може бути коментар типу `info:`
- Підпрограма має бути оточена тегами `sub` та `endsub`.
- Використані змінні повинні бути пронумерованими змінними та не повинні пропускати номер.
- Можуть бути включені коментарі та пресети.

Приклад скелета підпрограми

```
(info: simp -- simple exemple de sous-programme -- Ctrl-U pour éditer)
o<simp> sub
  #<ra>       = #1 (=.6 Rayon A) ;b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b''пb'' ↩
      b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ab'' b''зb'' b''кb''b''ob''b''мb'' ↩
      b''eb''b''нb''b''тb''b''ab''b''pb''b''eb''b''мb''
  #<radius_b> = #2 (=0.4)         ;b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b' ↩
      'пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ab'' b''бb''b''eb''b''зb'' b' ↩
      'кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''ab''b''pb''b''ib''b''вb''
  #<feedrate> = #3 (Feedrate)     ;b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' b' ↩
      'пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''ab'' b''бb''b''eb''b''зb'' b' ↩
      'пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob''b''гb''b''ob'' b' ↩
      'нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''
```

```
  g0x0y0z1
  g3 i#<ra> f#<feedrate>
  g3 i[0-#<radius_b>]
o<simp> endsub
```

10.7 TkLinuxCNC GUI

10.7.1 Вступ

TkLinuxCNC — один із перших графічних інтерфейсів для LinuxCNC. Він написаний на мові Tcl і використовує набір інструментів Tk для відображення. Написання на мові Tcl робить його дуже портативним (він працює на багатьох платформах). Можна відобразити окреме вікно backplot, як показано на малюнку.

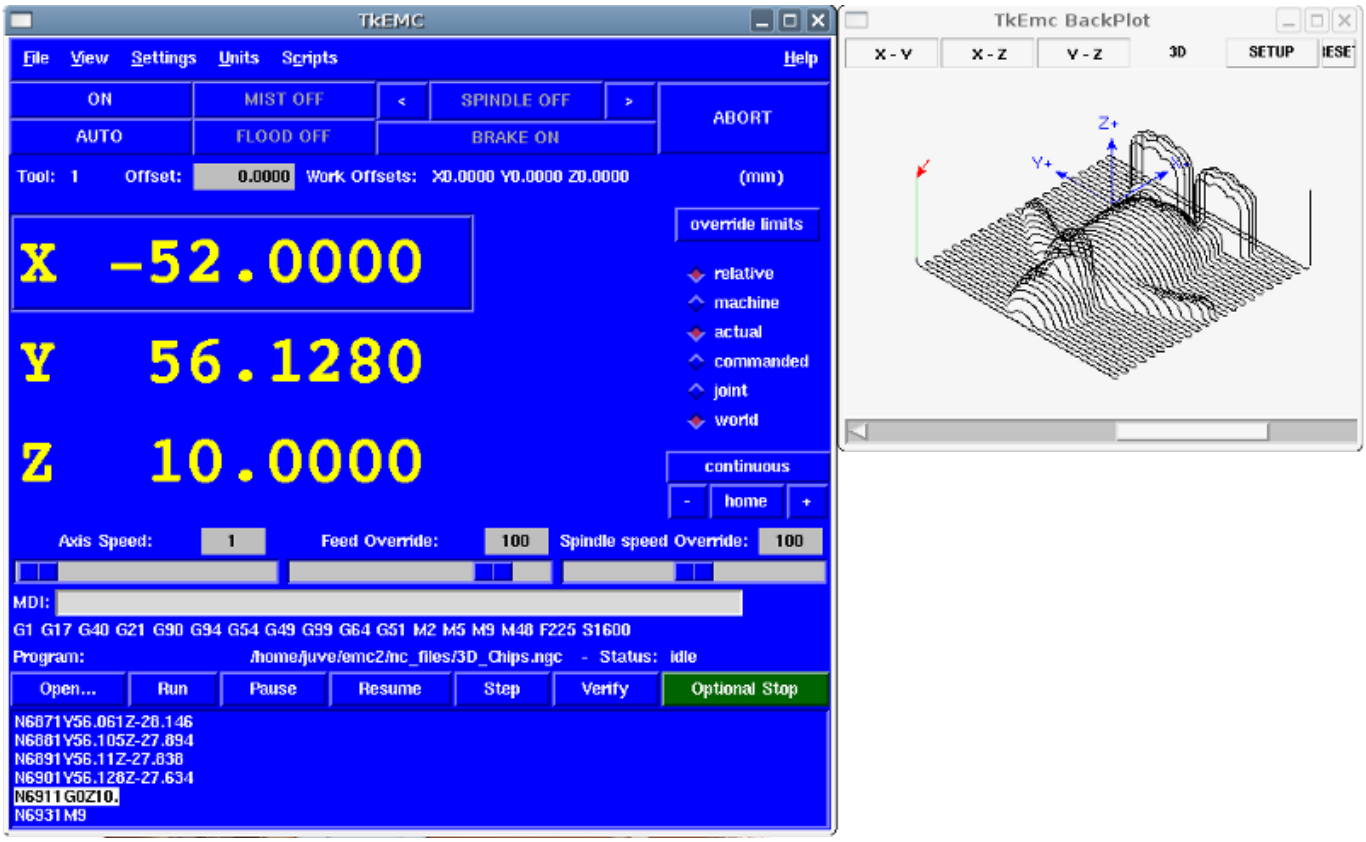


Figure 10.38: TkLinuxCNC Window

10.7.2 Початок роботи

Щоб вибрати TkLinuxCNC як інтерфейс для LinuxCNC, відредагуйте INI-файл. У розділі *[DISPLAY]* змініть рядок *DISPLAY* на

```
DISPLAY = tklinuxcnc
```

Потім запустіть LinuxCNC та виберіть цей INI-файл. Зразок конфігурації *sim/tklinuxcnc/tklinuxcnc.ini* вже налаштований на використання TkLinuxCNC як інтерфейсу.

Після запуску LinuxCNC відкривається вікно TKLinuxCNC.

10.7.2.1 Типовий сеанс роботи з TkLinuxCNC

1. Запустіть LinuxCNC та виберіть файл конфігурації.
2. Скасуйте умову «АВАРІЙНА СТОП» та увімкніть машину (натиснувши F1, а потім F2).
3. «Додому» для кожної осі.
4. Завантажте файл для фрезерування.
5. Покладіть заготовку, яку потрібно подрібнити, на стіл.
6. Встановіть відповідні зміщення для кожної осі, використовуючи режим ручного переміщення, і знову виконайте повернення в початкове положення або клацніть правою кнопкою миші на назві осі та введіть значення зміщення. Примітка: [Для деяких з цих дій може знадобитися зміна режиму, в якому наразі працює LinuxCNC.]
7. Запустіть програму.
8. Щоб знову обробити той самий файл фрезером, поверніться до кроку 6. Щоб обробити інший файл фрезером, поверніться до кроку 4. Після завершення вийдіть з LinuxCNC.

10.7.3 Елементи вікна TkLinuxCNC

Вікно TkLinuxCNC містить такі елементи:

- Панель меню, яка дозволяє виконувати різні дії
- Набір кнопок, що дозволяють змінювати поточний режим роботи, запускати/зупиняти шпиндель та виконувати інші відповідні операції введення/виведення
- Рядок стану для різних дисплеїв, пов’язаних зі зміщенням
- Область відображення координат
- Набір повзунків, що керують параметрами «Швидкість поштовхового переміщення», «Перевірка подачі» та «Перевірка швидкості шпинделя», що дозволяє збільшувати або зменшувати ці параметри
- Текстове поле для ручного введення даних «MDI»
- Відображення рядка стану з активними G-кодами, M-кодами, F- та S-словами
- Кнопки, пов’язані з перекладачем
- Область відображення тексту, яка показує вихідний G-код завантаженого файлу

10.7.3.1 Основні кнопки

Зліва направо кнопки такі:

- Увімкнення машини: *ESTOP > ESTOP RESET > ON*
- Перемикач охолоджувальної рідини туманом
- Зменшення швидкості шпинделя
- Встановити напрямок шпинделя *SPINDLE OFF > SPINDLE FORWARD . SPINDLE REVERSE*
- Збільшення швидкості шпинделя

• Перервати

тоді на другому рядку:

• Режим роботи: *MANUAL* > *MDI* > *AUTO*

• Перемикач затоплення охолоджувальної рідини

• Перемикач керування гальмом шпинделя

10.7.3.2 Рядок стану зсуву дисплея

У рядку стану відображення зміщення відображається поточний вибраний інструмент (вибраний за допомогою Txx M6), зміщення довжини інструмента (якщо активне) та робочі зміщення (встановлен клацанням правою кнопкою миші на координатах).

10.7.3.3 Область відображення координат

Основна частина дисплея показує поточне положення інструменту. Колір індикації положення залежить від стану осі. Якщо вісь не встановлена, вона буде відображатися жовтими літерами. Після встановлення вона буде відображатися зеленими літерами. Якщо з поточною віссю сталася помилка, TkLinuxCNC використає червоні літери, щоб показати це (наприклад, якщо спрацював апаратний кінцевий вимикач).

Щоб правильно інтерпретувати ці цифри, зверніться до радіо-боксів праворуч. Якщо положення «Машина», то відображуване число знаходиться в системі координат машини. Якщо «Відносне», то відображуване число знаходиться в системі координат зміщення. Далі вниз варіанти можуть бути «фактичні» або «задані». Фактичне значення відноситься до зворотного зв'язку, що надходить від енкодерів (якщо у вас є сервомашина), а «задане» значення відноситься до команди положення, що надсилається до двигунів. Ці значення можуть відрізнятися з кількох причин: помилка слідування, мертва зона, роздільна здатність енкодера або розмір кроку. Наприклад, якщо ви задаєте рух до X 0,0033 на вашому фрезерному верстаті, але один крок вашого крокового двигуна становить 0,00125, то «задане» положення буде 0,0033, але «фактичне» положення буде 0,0025 (2 кроки) або 0,00375 (3 кроки).

Інший набір перемикачів дозволяє вибрати між «спільним» і «світовим» видом. Вони не мають особливого сенсу на звичайних машинах (наприклад, з тривіальною кінематикою), але допомагають на машинах з нетривіальною кінематикою, таких як роботи або платформи Стюарта. (Більше про кінематику можна прочитати в Посібнику з інтегратора).

Задній план Коли машина рухається, вона залишає слід, який називається зворотною діаграмою. Ви можете відкрити вікно зворотної діаграми, вибравши Вигляд → зворотна діаграма.

10.7.3.4 Інтерпретатор TkLinuxCNC / Автоматичне керування програмою

Інтерпретатор / керування програмою TkLinuxCNC

Кнопки керування Кнопки в нижній частині TkLinuxCNC використовуються для керування виконанням програми:

+ * «Відкрити», щоб завантажити програму, * «Перевірити», щоб перевірити на наявність помилок, * «Виконати», щоб розпочати фактичне різання, * «Пауза», щоб зупинити його під час роботи, * «Відновити», щоб відновити вже призупинену програму, * «Крок», щоб перейти на один рядок у програмі та * «Додаткова зупинка» для перемикання додаткового перемикача зупинки (якщо кнопка зелена, виконання програми буде зупинено на будь-якій зустрічній точці M1).

Область відображення текстової програми Під час роботи програми рядок, що виконується в даний момент, виділяється білим кольором. Текстовий дисплей автоматично прокручується, щоб відобразити поточний рядок.

10.7.3.5 Ручне керування

Неявні ключі TkLinuxCNC дозволяє рухати верстат вручну. Ця дія називається «поступальним рухом». Спочатку виберіть вісь, яку потрібно перемістити, клацнувши на ній. Потім натисніть і утримуйте кнопку «+» або «-» залежно від бажаного напрямку руху. Перші чотири осі також можна переміщати за допомогою клавіш зі стрілками на клавіатурі (X і Y), клавіш PAGE UP і PAGE DOWN (Z) та клавіш «[» і «]» (A/4-та).

+ Якщо вибрано «Безперервно», рух триватиме доти, доки натиснуто кнопку або клавішу. Якщо вибрано інше значення, машина буде рухатися на відстань, що відображається, кожного разу, коли натискається кнопка або клавіша. Доступні значення:

+

```
1.0000, 0.1000, 0.0100, 0.0010, 0.0001
```

+ Натискання кнопки «Home» або клавіші HOME призведе до повернення вибраної осі в початкове положення. Залежно від конфігурації, це може просто встановити значення осі в абсолютне положення 0,0 або змусити верстат переміститися в певне початкове положення за допомогою «перемикачів початкового положення». Докладнішу інформацію див. у розділі Homing Chapter.

+ Натиснувши «Перевищити обмеження», машина тимчасово отримає дозвіл на роботу поза межами обмежень, визначених у файлі INI. (Примітка: якщо функція «Перевищити обмеження» активна, кнопка буде відображатися червоним кольором).

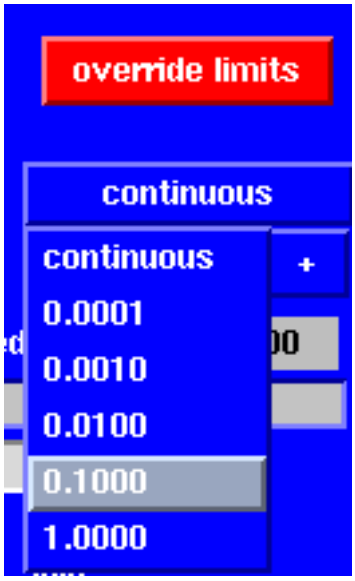


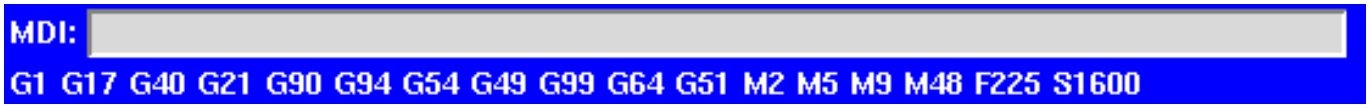
Figure 10.39: Приклад обмежень перевизначення TkLinuxCNC та кроків поштовхового переміщення

Група «Шпиндель» Кнопка в першому ряду вибирає напрямок обертання шпинделя: проти годинникової стрілки, зупинка, за годинниковою стрілкою. Кнопки поруч дозволяють користувачеві збільшувати або зменшувати швидкість обертання. Кнопка в другому ряду дозволяє вмикати або вимикати гальмо шпинделя. Залежно від конфігурації вашої машини, не всі елементи в цій групі можуть мати ефект.

Група охолоджувальної рідини Дві кнопки дозволяють вмикати та вимикати охолоджувальні рідини «Розпилення» та «Потік». Залежно від конфігурації вашої машини, можуть відображатися не всі елементи цієї групи.

10.7.3.6 Введення коду

Ручне введення даних (також називається MDI) дозволяє вводити програми G-коду вручну, по одному рядку за раз. Коли верстат не ввімкнений і не встановлений у режим MDI, елементи керування введенням коду недоступні.



Це дозволяє ввести команду G-коду для виконання. Виконайте команду, натиснувши Enter.

Активні G-коди Це показує «модальні коди», активні в інтерпретаторі. Наприклад, «G54» вказує на те, що «зміщення G54» застосовується до всіх введених координат.

10.7.3.7 Швидкість штовхання

Переміщуючи цей повзунок, можна змінювати швидкість переміщення. Числа вище вказані в одиницях осі / секунда. На текстове поле з числом можна натиснути. Після натискання з’явиться спливаюче вікно, в якому можна ввести число.

10.7.3.8 Перевизначення каналу

Переміщуючи цей повзунок, можна змінити запрограмовану швидкість подачі. Наприклад, якщо програма вимагає «F60», а повзунок встановлений на 120 %, то кінцева швидкість подачі буде 72. Текстове поле з цифрою можна натиснути. Після натискання з’явиться спливаюче вікно, в якому можна ввести цифру.

10.7.3.9 Коригування швидкості шпинделя

Повзунок перевищення швидкості шпинделя працює точно так само, як повзунок перевищення подачі, але він контролює швидкість шпинделя. Якщо програма вимагає S500 (швидкість шпинделя 500 об/хв), а повзунок встановлений на 80%, то кінцева швидкість шпинделя буде 400 об/хв. Цей повзунок має мінімальне і максимальне значення, визначені в файлі INI. Якщо вони відсутні, повзунок застрягає на 100%. Текстове поле з цифрою можна натиснути. Після натискання з’явиться спливаюче вікно, в якому можна ввести цифру.

10.7.4 Елементи керування клавіатурою

Майже всі дії в TkLinuxCNC можна виконати за допомогою клавіатури. Багато комбінацій клавіш недоступні в режимі MDI.

Найчастіше використовувані комбінації клавіш наведено в наступній таблиці.

Table 10.7: Найпоширеніші комбінації клавіш

Натискання клавіші	**Вжиті заходи**
F1	Увімкнути/вимкнути аварійну зупинку
F2	Увімкнення/вимкнення машини
`, 1 .. 9, 0	Встановити перевизначення подачі від 0% до 100%

Table 10.7: (continued)

Натискання клавіші	**Вжиті заходи**
X, `	Активувати першу вісь
Y, 1	Активувати другу вісь
Z, 2	Активувати третю вісь
A, 3	Активувати четверту вісь
Головна сторінка	Надіслати активну вісь додому
Ліворуч, праворуч	Поштовховий рух першої осі
Вгору, вниз	Поворот другої осі
Сторінка вгору, сторінка вниз	Поворот третьої осі
[,]	Поворот четвертої осі
ESC	Зупинити виконання

10.8 QtPlasmaC

10.8.1 Преамбула

Якщо не вказано інше, у цьому посібнику передбачається, що користувач використовує останню версію QtPlasmaC. Історію версій можна переглянути за посиланням link, де буде показано останню доступну версію. Встановлена версія QtPlasmaC відображається в рядку заголовка. Інформацію про оновлення QtPlasmaC див. у розділі Update QtPlasmaC.

10.8.2 Ліцензія

QtPlasmaC та все пов’язане з ним програмне забезпечення випускаються під ліцензією GPLv2.

10.8.3 Вступ

Версія QtPlasmaC для розробки є графічним інтерфейсом для плазмового різання, який використовує компонент plasmac для керування плазмовим столом за допомогою версії LinuxCNC (v2.10) для розробки, що використовує дистрибутив Debian Bullseye або пізніший.

Графічний інтерфейс QtPlasmaC підтримує до п’яти осей та використовує інфраструктуру QtVCP, що постачається з LinuxCNC.

Стандартна тема базується на дизайні користувача ”pinder” на форумі LinuxCNC, а кольори може змінювати користувач.

Версія QtPlasmaC GUI з гілки розробки буде працювати на будь-якому обладнанні, яке підтримується версією LinuxCNC (v2.10) з головної гілки, за умови, що є достатньо апаратних вхідних/вихідних контактів для виконання вимог конфігурації плазми.

Є три доступні формати:

- 16:9 з мінімальною роздільною здатністю 1366 x 768
- 9:16 з мінімальною роздільною здатністю 768 x 1366

- 4:3 з мінімальною роздільною здатністю 1024 x 768

Приклади знімків екрана QtPlasmaC наведено нижче:

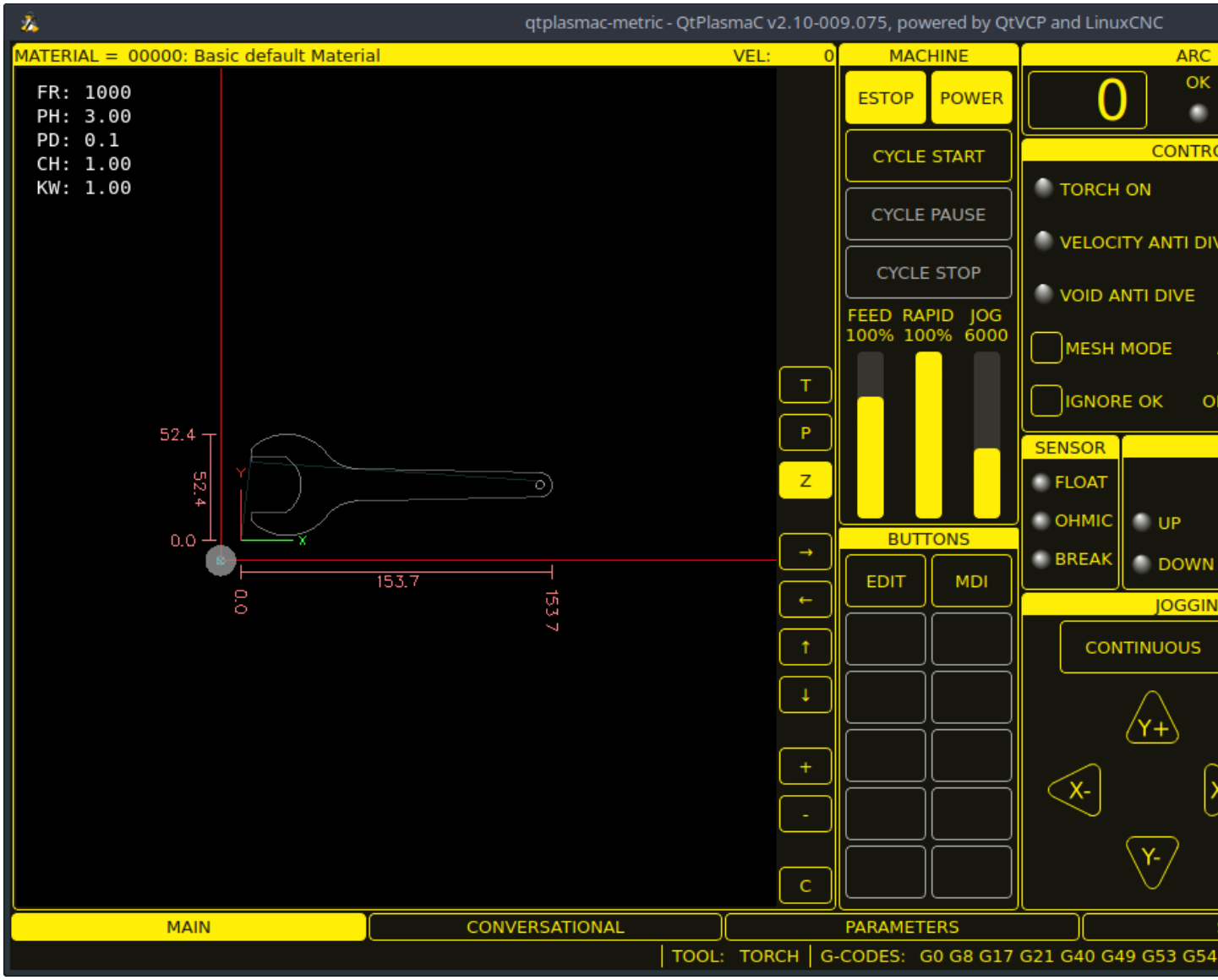
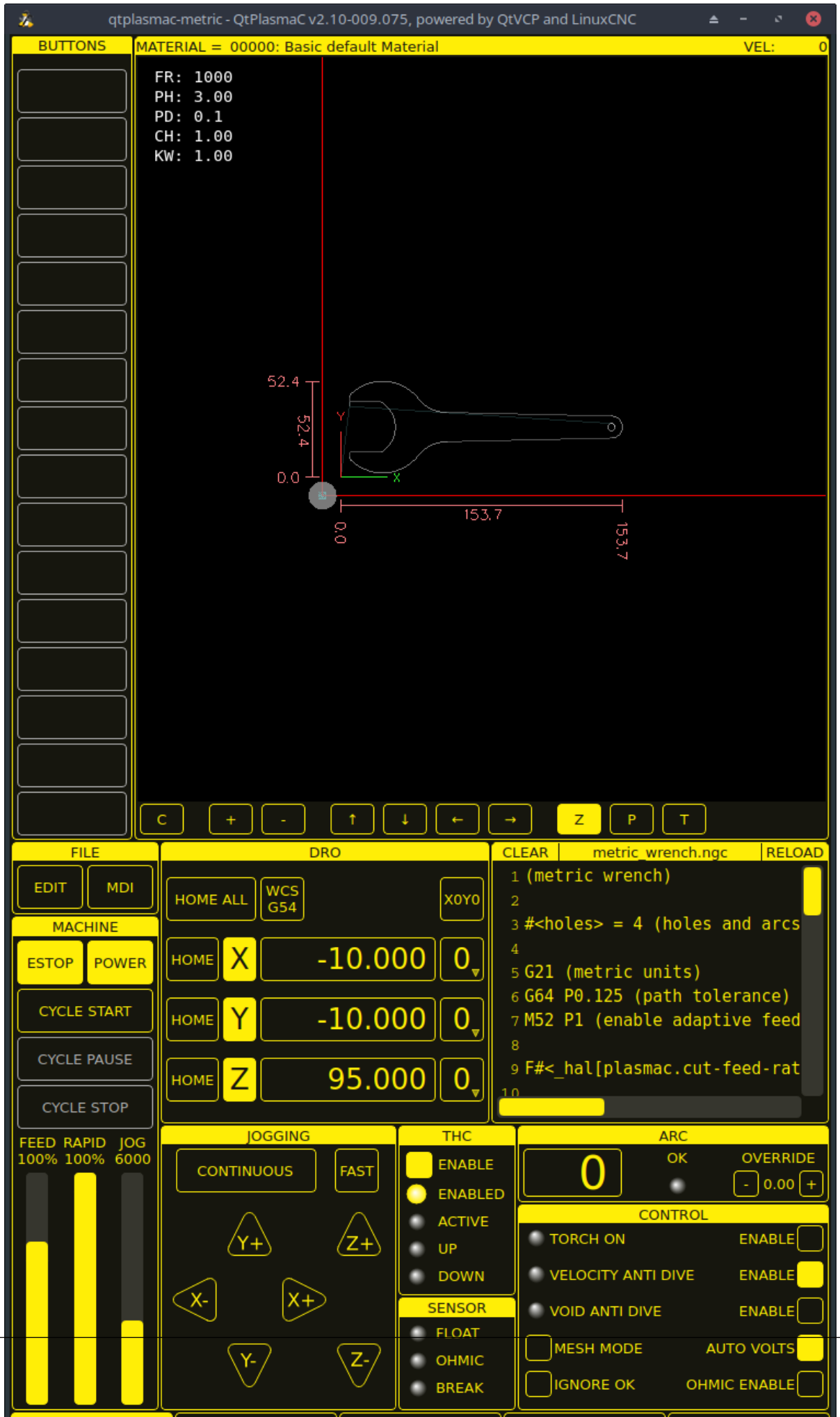


Figure 10.40: **16:9**


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
FEED 100%
RAPID 100%
JOG 6000
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
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

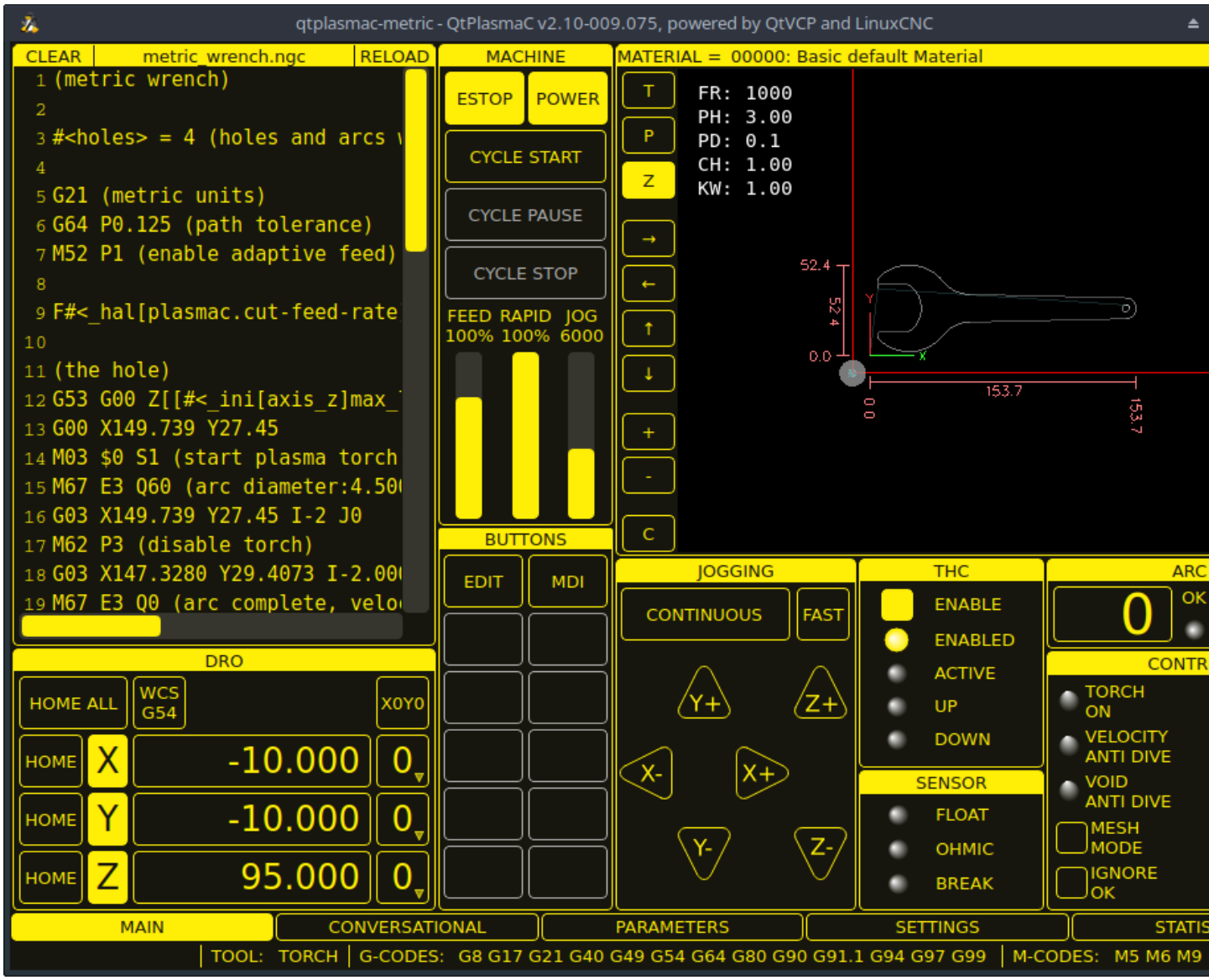


Figure 10.42: **4:3**

10.8.4 Встановлення LinuxCNC

Бажаний спосіб встановлення LinuxCNC – через ISO-образ, як описано нижче.

Note
Можна встановити та запустити LinuxCNC на різних дистрибутивах Linux, однак це виходить за межі цього посібника користувача. Якщо користувач бажає встановити дистрибутив Linux, відмінний від рекомендованих, йому спочатку потрібно встановити бажаний дистрибутив Linux, а потім встановити версію LinuxCNC (v2.10) з головної гілки разом з усіма необхідними залежностями. Слід також зазначити, що Bullseye є найранішим дистрибутивом Debian, який підтримується головною гілкою LinuxCNC (v2.10). Buster більше не підтримується.

10.8.4.1 Якщо у користувача не встановлено Linux

Інструкції з встановлення доступні за тут.

Дотримуючись цих інструкцій, ви отримаєте машину з поточною стабільною гілкою LinuxCNC (v2.9) на Debian 12 (Bookworm). Потім користувач повинен буде дотримуватися відповідних інструкцій для оновлення до версії LinuxCNC (v2.10) з головної гілки.

10.8.4.2 Встановлення пакета (Buildbot), якщо користувач має Linux на Debian 12 (Bookworm)

Дотримуйтесь інструкцій з розділу «Оновлення LinuxCNC на Debian Bookworm» за посиланням:../gettin started/getting-linuxcnc.html[тут].

10.8.4.3 Встановлення пакета (Buildbot), якщо користувач має Linux на Debian 12 (Bookworm) або Debian 11 (Bullseye)

Встановлення пакета (Buildbot) використовує попередньо зібрані пакети з LinuxCNC Buildbot.

Додайте ключі GPG та додайте репозиторій до списку джерел, щоб він відповідав версії Debian.

У наведеному нижче розділі буде додано репозиторій Bookworm з основної гілки (v2.10).

```
deb     http://buildbot2.highlab.com/ bookworm master-uspace
```

10.8.4.4 Запуск інсталяції на місці, якщо у користувача встановлено Linux

Інсталяція на місці запускає LinuxCNC з локально скомпільованої версії, яка зазвичай знаходиться в ~/linuxcnc-dev. Інструкції щодо створення інсталяції на місці доступні за тут.

10.8.5 Створення конфігурації QtPlasmaC

Перед створенням конфігурації QtPlasmaC важливо, щоб користувач добре розумів доступні режими роботи, а також входи/виходи, необхідні для успішної роботи плазми.

10.8.5.1 Режими

QtPlasmaC вимагає вибору одного з наступних трьох режимів роботи:

Режим	Опис
0	Використовує зовнішній вхід напруги дуги для розрахунку як напруги дуги (для керування висотою пальника), так і стану дуги в нормі.
1	Використовує зовнішній вхід напруги дуги для розрахунку напруги дуги (для керування висотою пальника). Використовує зовнішній вхід Arc OK для Arc OK.
2	Використовує зовнішній вхід Arc OK для Arc OK. Використовуйте зовнішні сигнали вгору/вниз для керування висотою пальника.



Important
Якщо джерело живлення плазми має вихід Arc OK (Transfer), то рекомендується використовувати його для Arc OK, а не м'який (розрахований) Arc OK, що надається режимом 0. Також можна використовувати reed relay як альтернативний метод для встановлення сигналу Arc OK, коли джерело живлення його не надає.

Note
Для точного налаштування режиму 0 Ark OK див. Tuning Mode 0 Arc OK у розділі ”Додаткові теми” цього посібника.

10.8.5.2 Доступно I/Os

Note
У цьому розділі розглядаються лише апаратні засоби вводу/виводу, необхідні для QtPlasmaC. Базові вимоги до машини, такі як кінцеві вимикачі, перемикачі додому тощо, наведено додатково до них.

Ім’я	Режими	Опис
Напруга дуги	0, 1	Аналоговий вхід; **опціонально.** Назва виводу HAL `plasmac.arc-voltage-in` Підключений до виходу швидкості кодерної плати роз’єму. Цей сигнал використовується для зчитування напруги дуги з метою визначення необхідних корекцій для підтримки відстані пальника від заготовки під час різання.
Дуга в порядку	1, 2	Цифровий вхід; **опціонально.** Назва виводу HAL `plasmac.arc-ok-in` Підключається від виводу Arc OK джерела живлення плазми до входу на роз’ємній платі. Цей сигнал використовується для визначення, чи встановлено дугу різання і чи можна переміщати верстат (іноді це називають передачею дуги).
Поплавковий вимикач	0, 1, 2	Цифровий вхід; **опціонально, див. інформацію під таблицею:** Назва виводу HAL `plasmac.float-switch` Підключається від входу плати розширення до перемикача на плаваючій голівці. Цей сигнал використовується для механічного зондування заготовки паяльною лампою та встановлення нуля Z у верхній частині заготовки. Якщо використовується і омічний зонд не налаштований, це є методом зондування. Якщо використовується і омічний зонд налаштований, це є резервним методом зондування.
Омічний зонд	0, 1, 2	Цифровий вхід; **опціонально, див. інформацію під таблицею:** Назва виводу HAL `plasmac.ohmic-probe` Підключається від виходу омічного зонда до входу роз’ємної плати. Цей сигнал використовується для електронного зондування шляхом замикання ланцюга за допомогою заготовки та витратних матеріалів пальника і встановлення нуля Z у верхній частині заготовки. Якщо використовується, це основний метод зондування. Якщо омічний зонд не може визначити місцезнаходження заготовки, а плаваючий вимикач відсутній, зондування продовжуватиметься доти, доки пальник не відірветься або не буде досягнуто мінімального обмеження Z.

Ім’я	Режими	Опис
Увімкнення омічного зонда	0, 1, 2	Цифровий вихід; **необов’язково, див. інформацію в таблиці нижче:** Назва виводу HAL `plasmac.ohmic-enable` Підключено від виходу плати розподільчої плати до входу для керування живленням омічного зонда.
Розривний перемикач	0, 1, 2	Цифровий вхід; **опціонально, див. інформацію під таблицею:** Назва виводу HAL `plasmac.breakaway` Підключається від входу плати роз’єднання до вимикача виявлення від’єднання пальника. Цей сигнал визначає, чи від’єднався пальник від свого кріплення.
Ліхтарик увімкнено	0, 1, 2	Цифровий вихід; **обов’язково.** Назва виводу HAL `plasmac.torch-on` Підключено від виводу плати розширення до входу torch-on джерела живлення плазми. Цей сигнал використовується для керування джерелом живлення плазми та запуску дуги.
Перемістити вгору	2	Цифровий вхід; **опціонально.** Назва виводу HAL `plasmac.move-up` Підключається від виходу «вгору» зовнішнього регулятора THC до входу роз’ємної плати. Цей сигнал використовується для управління віссю Z у напрямку вгору та внесення необхідних корегувань для підтримки відстані пальника від заготовки під час різання.
Перемістити вниз	2	Цифровий вхід; **опціонально.** Назва виводу HAL `plasmac.move-down` Підключається від виходу вниз зовнішнього регулятора THC до входу роз’ємної плати. Цей сигнал використовується для управління віссю Z у напрямку вниз і внесення необхідних коректив для підтримки відстані пальника від заготовки під час різання.
Озброєння писаря	0, 1, 2	Цифровий вихід; **опціонально.** Назва виводу HAL `plasmac.scribe-arm` Підключено від виводу плати розширення до схеми озброєння різака. Цей сигнал використовується для розміщення різака в потрібному положенні на заготовці .
Писати далі	0, 1, 2	Цифровий вихід; **необов’язково.** Назва виводу HAL `plasmac.scribe-on` Підключено від виходу плати розгалуження до схеми нанесення скарбів. Цей сигнал використовується для ввімкнення пристрою нанесення скарбів.
Лазер увімкнено	0, 1, 2	Цифровий вихід; **необов’язково.** Назва виводу HAL `qtplasmac.laser_on` Цей сигнал використовується для ввімкнення лазера вирівнювання.

Потрібен лише один з цих параметрів: **Поплавковий вимикач** або **Омічний зонд**. Якщо використовуються обидва, то **Поплавковий вимикач** буде резервним варіантом, якщо **Омічний зонд** не спрацьовує.

Якщо використовується **Омічний зонд**, тоді потрібно позначити опцію **Увімкнути Омічний зонд** у графічному інтерфейсі QtPlasmaC.

Роз’єднувальний вимикач не є обов’язковим, оскільки **плаваючий вимикач** розглядається так само, як роз’єднувальний, коли не проводиться зондування. Якщо це два окремі вимикачі, а на роз’єднувальному блоці недостатньо входів, їх можна об’єднати і підключити як **плаваючий вимикач**.

Note
Мінімальні вимоги до вводу-виводу для функціонування конфігурації QtPlasmaC: вхід **Arc Voltage** АБО вхід **Arc OK**, вхід **Float Switch** і вихід **Torch On**. Повторюємо, що в цьому випадку QtPlasmaC буде розглядати плаваючий перемикач як розривний перемикач, коли він не проводить зондування.

10.8.5.3 Рекомендовані налаштування:

Refer to the Heights Diagrams for a visual representation of the terms below.

- **[AXIS_Z] MIN_LIMIT** повинен бути трохи нижче верхньої частини планок з урахуванням похибки float_switch_travel і похибки перебігу. Наприклад, якщо поплавковий вимикач користувача потребує 4 мм (0,157”) для активації, встановіть мінімальне значення Z на 5 мм (0,2”) плюс допуск для перевищення (розрахований за допомогою рівняння нижче або допуск 5 мм (0,2”) нижче найнижчої планки).
- **[AXIS_Z] MAX_LIMIT** має бути найвищим значенням, на яке користувач хоче, щоб перемістилася вісь Z (воно не повинно бути нижчим за Z HOME_OFFSET).
- **[AXIS_Z] Значення HOME** слід встановити приблизно на 5–10 мм (0,2–0,4 дюйма) нижче максимального значення.
- **Плаваюча головка** – рекомендується використовувати плаваючу головку, яка має достатній рух для забезпечення перебігу під час зондування. Перебіг можна розрахувати за такою формулою:

```
o = 0.5 * a * (v / a)^2
```

де: o = перевищення, a = прискорення в одиницях/с^2 та v = швидкість в одиницях/с.

Приклад метричної системи: для MAX_ACCELERATION осі Z, що дорівнює 600 мм/с², та MAX_VELOCITY що дорівнює 60 мм/с, перевищення становитиме 3 мм.

Приклад для імперських систем: враховуючи MAX_ACCELERATION осі Z, що дорівнює 24 дюймам/с², та MAX_VELOCITY, що дорівнює 2,4 дюймам/с, перевищення становитиме 0,12 дюйма.

На верстатах, які використовують омічний зонд як основний метод зондування, настійно рекомендується встановити перемикач на плаваючій головці як резервний засіб зупинки руху по осі Z у разі виходу з ладу омічного зонда через забруднення поверхні.

10.8.5.4 Налаштування

LinuxCNC надає два майстри налаштування, які можна використовувати для створення конфігурації верстата. Вибір цих майстрів залежить від обладнання, яке використовується для керування верстатом.

Якщо користувач бажає використовувати інсталяцію Run In Place, то перед запуском однієї з наступних команд йому потрібно буде виконати таку команду з терміналу:

```
source ~/linuxcnc-dev/scripts/rip-environment
```

Якщо використовується інсталяція пакета, то жодних додаткових дій не потрібно.

Якщо використовуєте паралельний порт, скористайтеся майстром StepConf, виконавши команду `stepconf` у вікні терміналу або запустивши її за допомогою пункту меню робочого столу **Application -> CNC -> StepConf Wizard**.

Якщо ви використовуєте плату Mesa Electronics, скористайтеся PnCconf wizard, запустивши команду `pncconf` у вікні терміналу або запустивши її за допомогою пункту меню робочого столу **Application -> CNC -> PnCConf Wizard**.

Якщо ви використовуєте плату Pico Systems, ця тема форуму LinuxCNC може бути корисною.

Специфічні для машини налаштування тут не описані, зверніться до документації до конкретного майстра налаштування, який використовується.

Для цих майстрів доступні розділи форуму LinuxCNC:

StepConf Wizard

PnCconf Wizard

Заповніть необхідні поля відповідно до конфігурації підключення машини/плати розподілу.

QtPlasmaC додає дві сторінки до майстрів налаштування LinuxCNC для параметрів, специфічних для QtPlasmaC. Ці дві сторінки — це параметри QtPlasmaC та User Buttons. Заповніть кожну сторінку майстра QtPlasmaC відповідно до машини, що налаштовується, та вимог до кнопок користувача.

Зверніть увагу, що параметри PnCConf дозволяють користувачеві вибирати такі параметри, як корекція подачі, лінійна швидкість та крок поштовхового переміщення, тоді як у StepConf вони автоматично розраховуються та встановлюються.

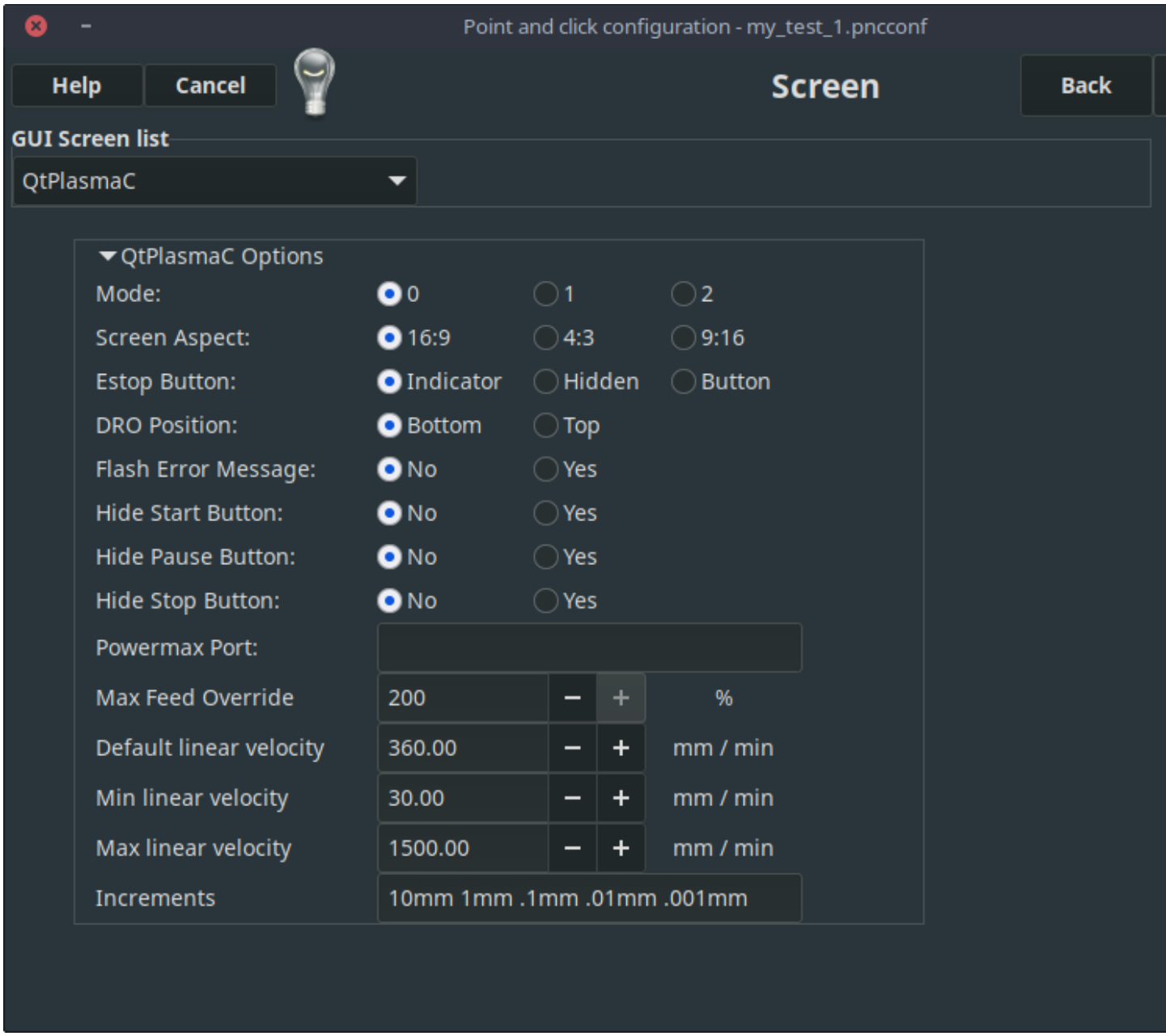


Figure 10.43: Параметри PnCConf QtPlasmaC

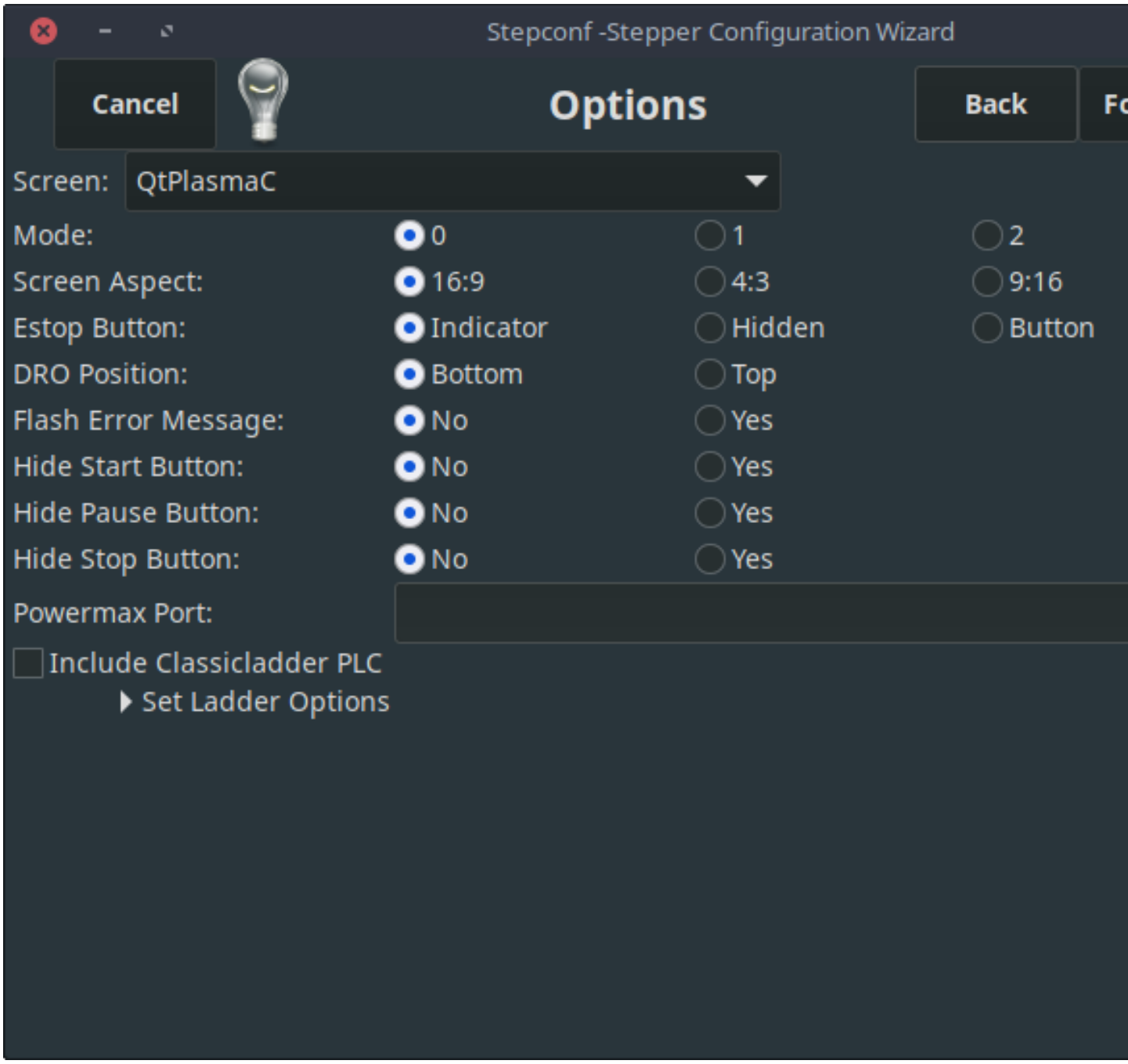


Figure 10.44: Параметри StepConf QtPlasmaC

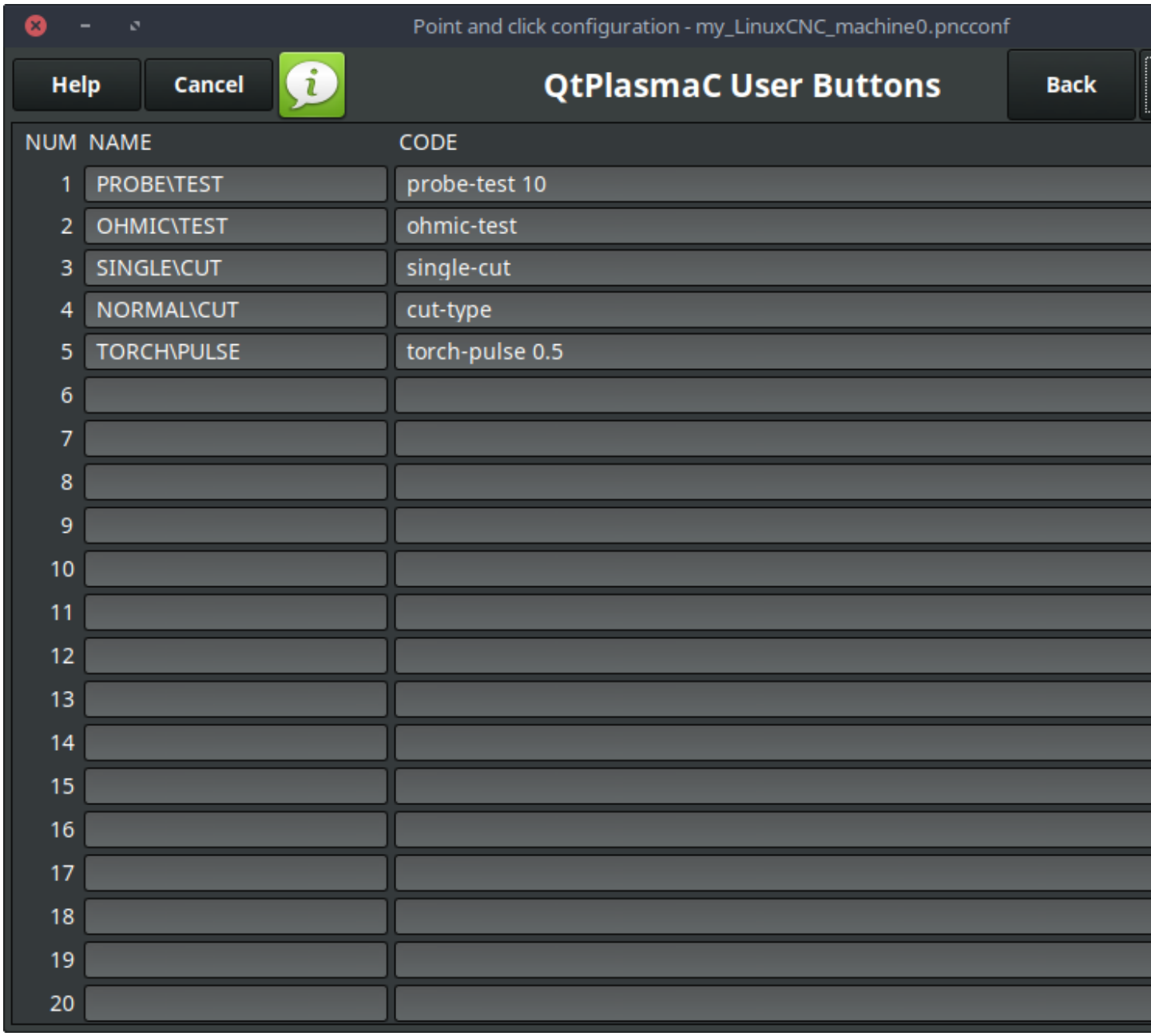


Figure 10.45: Кнопки користувача QtPlasmaC

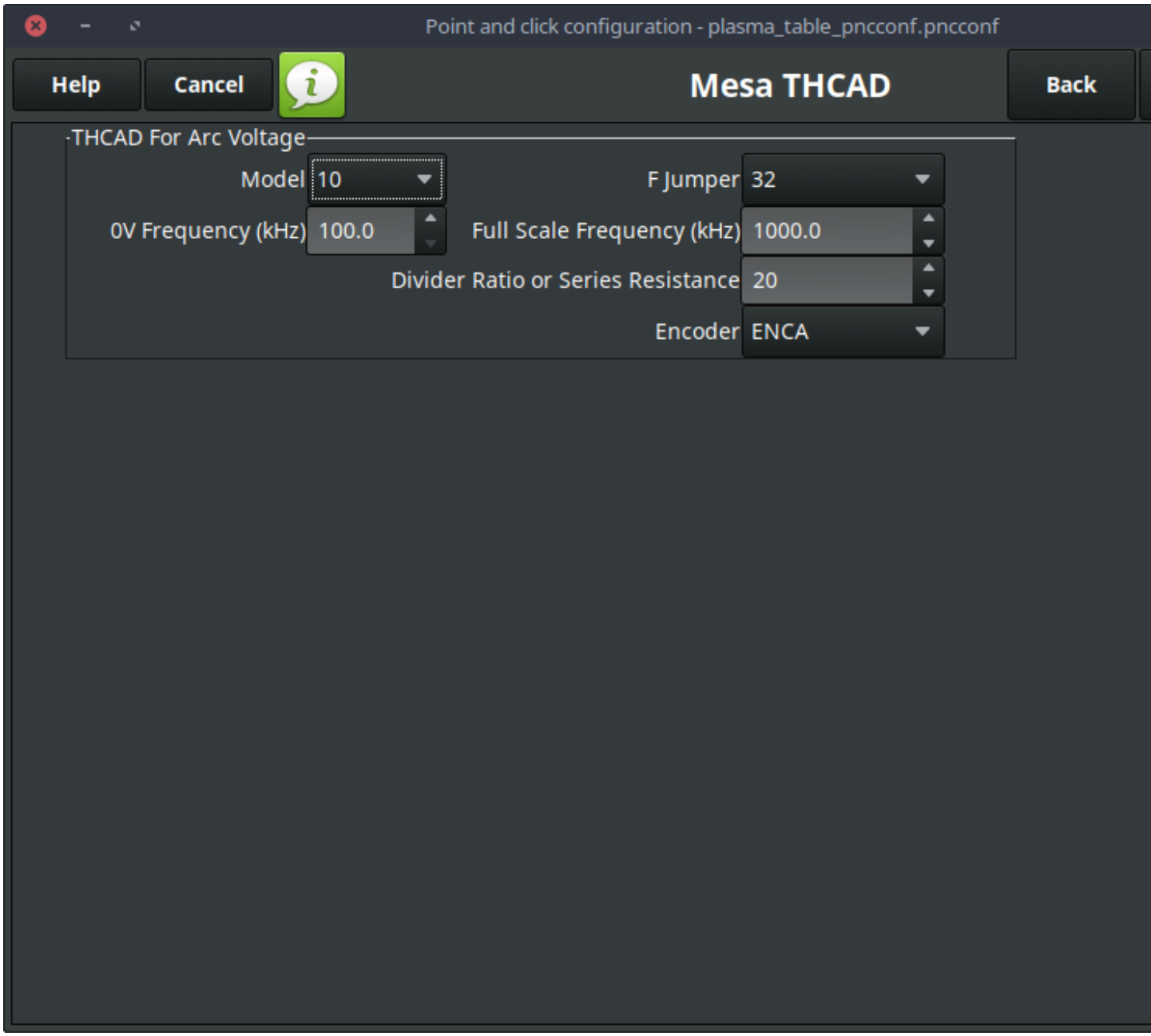


Figure 10.46: QtPlasmaC THCAD

Екран THCAD з’явиться, лише якщо на екрані карти вибрано плазмовий кодер. Для отримання додаткової інформації див. розділ присвячений Mesa THCAD.

Після завершення налаштування майстер збереже копію конфігурації, яку можна буде завантажити та редагувати пізніше. Робоча конфігурація QtPlasmaC буде створена в наступному каталозі: ~/linuxcnc/configs/<*machine_name*>.

Спосіб запуску щойно створеної конфігурації QtPlasmaC з командного рядка терміналу дещо відрізняється залежно від способу встановлення LinuxCNC:

Для встановлення пакета (Buildbot):

```
linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

Для встановлення на місці:

```
~/linuxcnc-dev/scripts/linuxcnc ~/linuxcnc/configs/_<machine_name>_/_<machine_name>_.ini
```

Після виконання наведеної вище команди LinuxCNC має працювати з видимим графічним інтерфейсом QtPlasmaC.



Important
ПЕРЕД ТИМ, ЯК ПРОДОВЖИТИ, КОРИСТУВАЧ ПОВИНЕН БУТИ МОЖЛИВИЙ ВИСТАВИТИ ВЕРСТАТ У ХОДУЮЧЕ ПОЛОЖЕННЯ, ВСТАНОВИТИ КОЖНУ ВІСЬ У НУЛЬ, ПЕРЕМІЩИТИ ВСІ ОСІ ДО М’ЯКИХ МЕЖ БЕЗ ЗБІЙ ТА ЗАПУСКУ ТА ЗАПУСТИТИ ТЕСТОВІ ПРОГРАМИ G-КОДУ БЕЗ БУДЬ-ЯКИХ ПОМИЛОК.

ТІЛЬКИ КОЛИ цей критерій виконано, користувач повинен продовжувати початкове налаштування QtPlasmaC.

Note
Можна створити конфігурацію симулятора за допомогою StepConf, але неможливо мати тандемні з’єднання в конфігурації симулятора.

10.8.5.5 Помилки залежностей Qt

Якщо під час спроби запуску конфігурації QtPlasmaC виникнуть будь-які помилки залежностей Qt, користувачеві може знадобитися запустити скрипт встановлення QtVCP, щоб вирішити ці проблеми.

Для встановлення пакета (Buildbot) введіть таку команду у вікні терміналу:

```
/usr/lib/python3/dist-packages/qtvcp/designer/install_script
```

Для встановлення на місці введіть таку команду у вікні терміналу:

```
~/linuxcnc-dev/lib/python/qtvcp/designer/install_script
```

10.8.5.6 Початкове налаштування

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**.

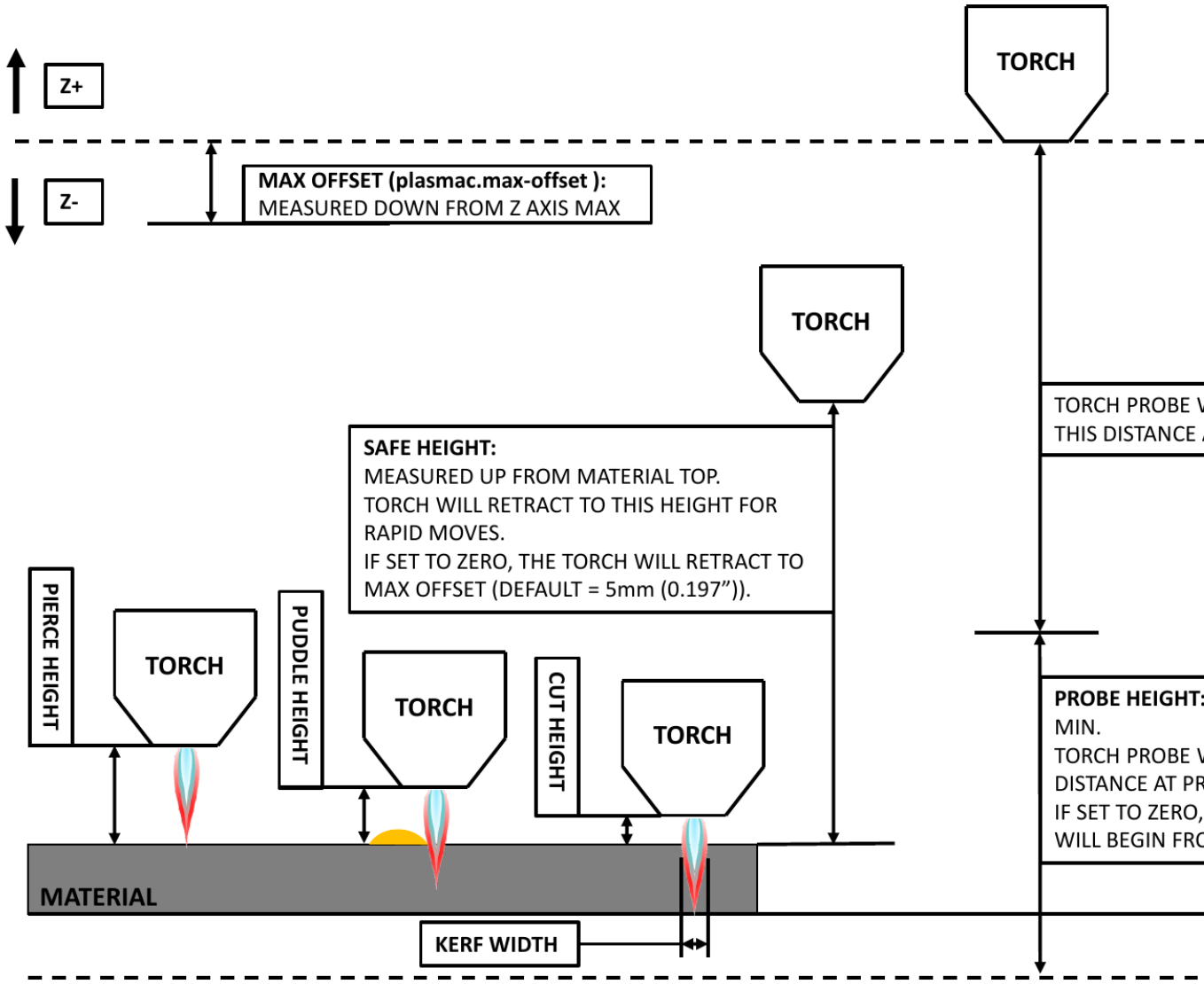


Figure 10.47: **Probe Height Only**

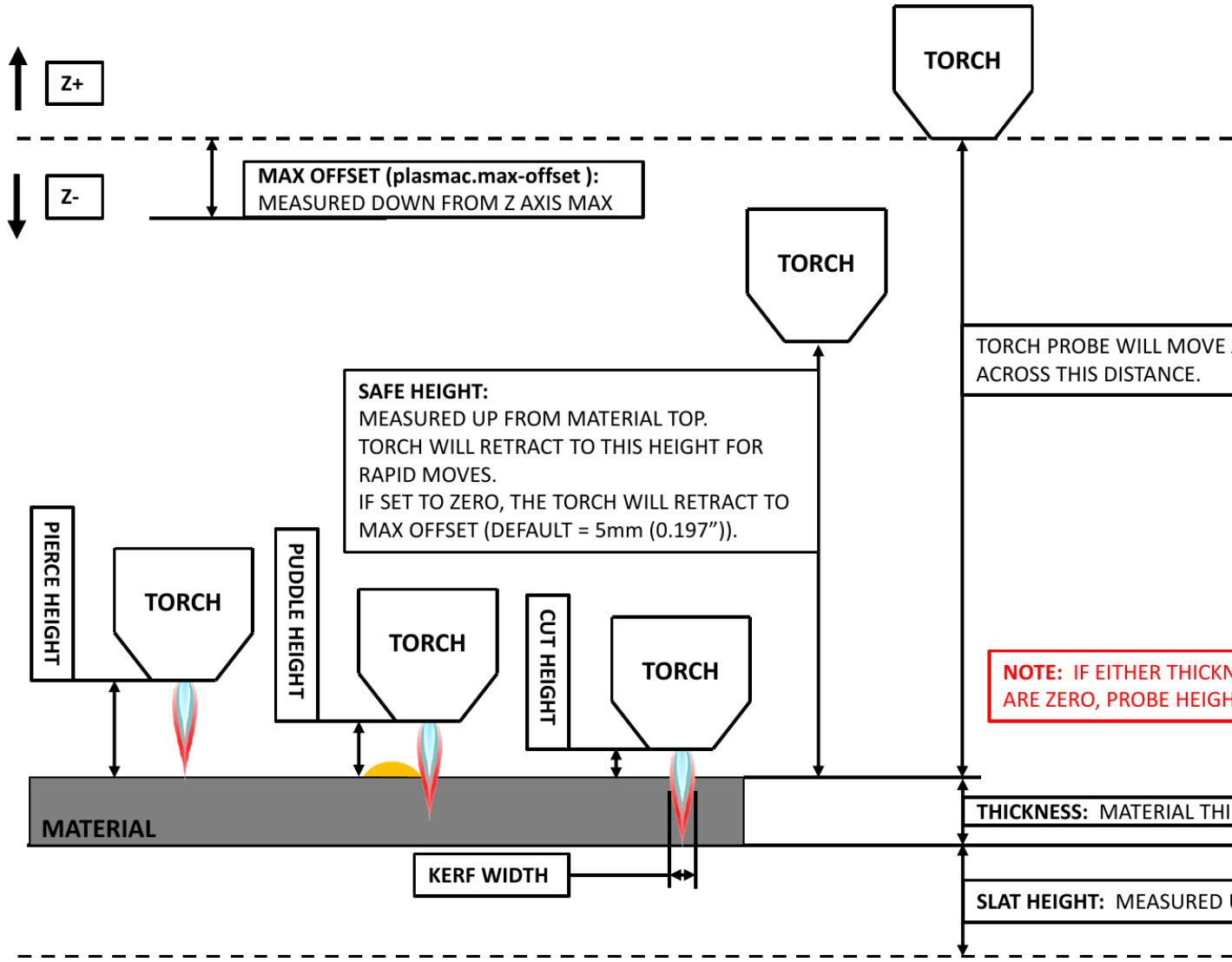


Figure 10.48: **Slat Height and Material Thickness**

Натисніть на вкладку Parameters Tab, щоб переглянути розділ **CONFIGURATION**, в якому відображаю параметри, що можуть бути налаштовані користувачем. Необхідно переконатися, що кожне з цих налаштувань відповідає вимогам обладнання.

Щоб встановити DRO осі Z відносно MINIMUM_LIMIT осі Z, користувач повинен виконати наступні кроки. Важливо розуміти, що в QtPlasmaC відключення DRO осі Z не впливає на положення осі Z під час виконання програми G-коду. Ці кроки просто дозволяють користувачеві легше встановити висоту зонда, оскільки після виконання цих кроків відображуване значення DRO осі Z буде відносно MINIMUM_LIMIT осі Z.

Note
Користувач повинен бути ознайомлений з рекомендованими Z Axis Settings.

1. Перевести вісь Z додому.

2. Переконайтеся, що під пальником нічого немає, потім опустіть вісь Z до зупинки на мінімальному обмеженні вісі Z, а потім натисніть 0 поруч із віссю Z DRO, щоб виконати **Touch Off** (відключення датчика) з вибраною віссю Z, щоб встановити вісь Z на нульове зміщення. Цей крок служить лише для того, щоб користувач міг легше візуалізувати та налаштувати **Probe Height** (висоту датчика). Це значення вимірюється від мінімального обмеження вісі Z вгору.
3. Знову перемістіть вісь Z у вихідне положення.

Тест зонда

Якщо машина оснащена поплавковим вимикачем, користувачеві потрібно буде встановити зміщення в розділі **КОНФІГУРАЦІЯ** на вкладці **ПАРАМЕТРИ**. Це можна зробити, запустивши цикл «Перевірка зонда».

1. Перевірте, чи правильні значення швидкості зонда та висоти зонда в розділі **CONFIGURATION** (Конфігурація) на вкладці **PARAMETERS** (Параметри). QtPlasmaC може проводити зондування з повною швидкістю по осі Z, якщо машина має достатній рух у поплавковому вимикачі, щоб поглинути будь-яке перевищення. Якщо машина підходить, користувач може встановити висоту зонда на значення, близьке до мінімального значення осі Z, і виконувати все зондування на повній швидкості.
2. Якщо машина ще не переведена у вихідне положення та не знаходиться у вихідному положенні, переведіть її у вихідне положення.
3. Покладіть трохи матеріалу на планки під пальником.
4. Натисніть кнопку **ПЕРЕВІРКА ЗОНДА**.
5. Вісь Z опуститься вниз, знайде матеріал, а потім підніметься до заданої **висоти пробивання**, встановленої для поточного вибраного матеріалу. Пальник буде чекати в цьому положенні протягом часу, заданого у файлі <*machine_name*>.prefs. За замовчуванням час утримання зонда становить 10 секунд, це значення можна редагувати у файлі <*machine_name*>.prefs. Після цього пальник повернеться до початкової висоти.
6. Виміряйте відстань між матеріалом і кінчиком пальника, поки пальник чекає на **Висоті проколу**.
7. Якщо вимірювання перевищує **Висоту пробивання** поточного вибраного матеріалу, зменште «Хід поплавка» в розділі **КОНФІГУРАЦІЯ** на вкладці **ПАРАМЕТРИ** на різницю між виміряним значенням і заданим значенням. Якщо вимірювання менше, ніж **Висота пробивання** поточного вибраного матеріалу, збільште «Хід поплавка» в розділі **КОНФІГУРАЦІЯ** на вкладці **ПАРАМЕТРИ** на різницю між заданим значенням і виміряним значенням.
8. Після внесення змін до параметра «Float Travel» (Хід поплавка) повторіть процес, описаний у пункті 4 вище, доки виміряна відстань між матеріалом і кінчиком пальника не збіжиться з **Pierce Height** (Висота проколу) для поточного вибраного матеріалу.
9. Якщо стіл має лазер або камеру для вирівнювання листа, рисувальник або використовує зсувне зондування, то необхідні зсуви потрібно застосувати, дотримуючись процедури, описаної в розділі Peripheral Offsets. Кнопки LASER та/або CAMERA не будуть видимими, доки користувач не встановить відповідні зсуви та не збереже їх у файлі <*machine_name*>.prefs.
10. ВІТАЄМО! Тепер користувач має мати робочу конфігурацію QtPlasmaC.

Note
Якщо час між контактом пальника з матеріалом і моментом, коли пальник піднімається і зупиняється на висоті проколювання, здається надмірним, див. розділ «Плазма: зондування, зондування» для можливого вирішення.



Important
ПРИ ВИКОРИСТАННІ **Mesa Electronics THCAD** ТОГДА ЗНАЧЕННЯ **Шкали напруги** БУЛО ОТРИМАНО МАТЕМАТИЧНИМ ШЛЯХОМ. ЯКЩО КОРИСТУВАЧ МАЄ НАМІР ВИКОРИСТОВУВАТИ НАПРУГИ З ТАБЛИЦІ ВИРОБНИКА, ТОДІ РЕКОМЕНДУЄТЬСЯ ВИКОНАТИ ВИМІРЮВАННЯ ФАКТИЧНИХ НАПРУГ І ТОЧНО НАЛАШТУВАТИ **ШКАЛУ НАПРУГИ** І **ЗСУВ НАПРУГИ**.



Warning
НАПРУГА ПЛАЗМОВОГО РІЗАННЯ МОЖЕ БУТИ СМЕРЕЛЬНОЮ. ЯКЩО КОРИСТУВАЧ НЕ МАЄ ДОСВІДУ У ВИКОНАННІ ЦИХ ВИМІРЮВАНЬ, ЗВЕРНІТЬСЯ ЗА КВАЛІФІКОВАНОЮ ДОПОМОГОЮ.

10.8.6 Міграція на QtPlasmaC з PlasmaC (AXIS або GMOCCAPY)

Автоматична міграція з PlasmaC до QtPlasmaC більше не підтримується. Користувач повинен буде або вручну перетворити конфігурацію PlasmaC, або створити нову конфігурацію за допомогою конфігурації.

10.8.7 Інші міркування щодо налаштування QtPlasmaC

10.8.7.1 Низькочастотний фільтр

Компонент plasmac HAL має вбудований фільтр нижніх частот, який, якщо використовується, застосовується до вхідного контакту **plasmac.arc-voltage-in** для фільтрації будь-яких шумів, які можуть спричинити помилкові показання напруги. Фільтр нижніх частот слід використовувати лише після використання Halscope для визначення необхідної частоти та того, чи амплітуда шуму є достатньо великою, щоб спричинити будь-які проблеми. Для більшості плазмових машин фільтр нижніх частот не є необхідним і не повинен використовуватися, якщо в цьому немає потреби.

Контакт HAL, призначений для цього фільтра, має назву **plasmac.lowpass-frequency** і за замовчуванн встановлений на 0 (відключений). Щоб застосувати фільтр нижніх частот до напруги дуги, користувач повинен відредагувати наступний запис у файлі custom.hal у каталозі конфігурації машини, щоб додати відповідну частоту відсічення, виміряну в герцах (Гц).

Наприклад:

```
setp plasmac.lowpass-frequency 100
```

У наведеному вище прикладі гранична частота становила б 100 Гц.

10.8.7.2 Відмова від контакту

Бризки контактів від механічних реле, перемикачів або зовнішніх перешкод можуть спричинити нестабільну роботу таких перемикачів:

- Поплавковий вимикач
- Омічний зонд
- Розривний перемикач
- Дуга в порядку (для режимів 1 та 2)

Оскільки програмне забезпечення здатне здійснювати вибірку з частотою, що перевищує період відскоку контакту, воно може сприймати відскок контакту як кілька змін стану входу, що відбуваються за дуже короткий проміжок часу, і неправильно інтерпретувати це як дуже швидке вмикання-вимикання входу. Одним із способів зменшення відскоку контакту є «дебаунс» входу. Якщо коротко описати дебаунс, то для цього потрібно, щоб стан входу був стабільним у протилежному стані до стану виходу протягом декількох послідовних періодів затримки, перш ніж змінювати стан виходу.

Періоди затримки усунення дребезгу можна змінити, відредагувавши відповідне значення усунення дребезгу у файлі custom.hal у каталозі <*назва_комп'ютера*> config.

Кожне збільшення затримки додає один цикл сервопотоку до часу дебаунсу. Наприклад: при періоді сервопотоку 1000000 (вимірюється в наносекундах) затримка дебаунсу 5 дорівнюватиме 5000000 нс або 5 мс.

Для поплавкових та омічних перемикачів це дорівнює збільшенню результату вимірювання висоти на 0,001 мм (0,00004 дюйма).

Рекомендується встановлювати значення дебаунсу якомога нижче, але при цьому досягати стабільних результатів. Використання Halscope для побудови графіку вхідних даних є хорошим способом встановити правильне значення.

Для установок QtPlasmaC дебаунс досягається за допомогою посилання HAL dbounce component, яке є пізнішою альтернативою оригінальному компоненту дебаунс. Ця нова версія дозволяє завантажувати та називати окремі екземпляри дебаунс і сумісна з обробкою файлів Twopass HAL.

Всі чотири сигнали вище мають індивідуальний компонент дебаунсу, тому періоди дебаунсу можуть бути індивідуально налаштовані для кожного входу. Будь-які зміни, внесені в ці значення у файлі custom.hal, не будуть перезаписані пізнішими оновленнями QtPlasmaC.

За замовчуванням затримка для всіх чотирьох входів становить п'ять періодів сервоприводу. У більшості випадків це значення працює досить добре. Якщо будь-який із входів не використовує механічні перемикачі, можна зменшити або скасувати затримку для цих входів.

Якщо для іншого обладнання, такого як домашні або кінцеві вимикачі тощо, потрібне усунення дребезгу, тоді в будь-який з HAL-файлів можна додати більше компонентів усунення дребезгу, незалежно від сигналів, перелічених тут.

10.8.7.3 Контактне навантаження

Механічні реле та перемикачі зазвичай вимагають мінімального струму, що проходить через контакти, для надійної роботи. Цей струм залежить від матеріалу, з якого виготовлені контакти в пристрої.

Залежно від заданого мінімального струму контактів та струму, що споживається вхідним пристроєм, може виникнути потреба забезпечити метод збільшення струму через контакти.

Більшість реле з використанням золотих контактів не потребують додаткового струму для надійної роботи.

Існує два різних методи забезпечення цього мінімального струму, якщо це необхідно:

1. Плівковий конденсатор ємністю 0,1 мкФ, розміщений між контактами.
2. Резистор 1200 Ом і 1 Вт, підключений до навантаження (див. розрахунки нижче).

Схеми показано за адресою схеми контактного-навантаження.

Більше інформації про навантаження на комутацію контактів можна знайти на сторінці VI документа finder https://cdn.findernet.com/app/uploads/TecEN.pdf [Загальна технічна інформація].

Розрахунки:

При використанні карти Mesa вхідний опір 7I96 становить 4700 Ом (символ R) (завжди звертайтеся до інструкції до продукту, що відповідає використовуваній версії, оскільки ці значення іноді відрізняються в різних версіях), що дає струм контакту 5,1 мА (символ I) при напрузі живлення (символ U) 24 В (I = U/R) примітка: [У США літера V зазвичай використовується як символ (напруга) і як одиниця виміру (вольт).].

Наприклад, типове реле, що використовується в плазмовому різаку Hypertherm Powermax 65 (link:https://www.te.com/commerce/DocumentDelivery/DDEController? Action=showdoc&DocId=Data+Sh 1393194-0 [TE T77S1D10-24]) вимагає мінімального навантаження контакту 100 мА при 5 В постійного струму, що розсіює 0,5 Вт (P = I * V). При використанні джерела живлення 24 В постійного струму це буде відповідати мінімальному струму 20,8 мА. Оскільки вхід Mesa споживає менше струму, ніж потрібно реле, необхідно збільшити струм.

Опір можна розрахувати за формулою $R = U_s / (I_m - I_i)$, де:

- R = розрахунковий опір
- U_s = напруга живлення
- I_m = мінімальний необхідний струм
- I_i = вхідний струм

Використання 7I96 з вхідним струмом 5,1 мА дає розрахункове значення 1529 Ом (= 24 В / (.0208 - .0051) А). Це значення можна округлити до загальнодоступного резистора 1500 Ом, що дасть невеликий запас міцності.

Розсіювання потужності можна розрахувати за формулою $P = U_s^2 / R_s$, де:

- P = потужність
- U_s = напруга живлення
- R_s = вибраний опір

Це дає значення 0,38 Вт. Потім його можна округлити до 1 Вт, що забезпечить хороший запас міцності. Остаточний вибір – резистор 1500 Ом і 1 Вт.

10.8.7.4 Запуск робочого столу

Якщо під час створення конфігурації не було створено посилання для її запуску, користувач може створити ярлик на робочому столі, клацнувши правою кнопкою миші на робочому столі та вибравши «Створити ярлик» або подібну опцію. Відкриється діалогове вікно для створення ярлика. Дайте іконці коротке ім’я, введіть будь-яку команду та натисніть «OK».

Після того, як панель запуску з’явиться на робочому столі, клацніть на ній правою кнопкою миші та відредагуйте її у вибраному користувачем редакторі. Відредагуйте файл, щоб він виглядав приблизно так:

```
[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
```

Якщо користувач хоче, щоб вікно терміналу відкривалося позаду вікна графічного інтерфейсу, змініть рядок терміналу на:

```
Terminal=true
```

Відображення терміналу може бути зручним для повідомлень про помилки та інформації.

10.8.7.5 Файли QtPlasmaC

Після успішної інсталяції QtPlasmaC у каталозі конфігурації створюються такі файли:

Ім’я файлу	**Функція**
<*machine_name*>.ini	Файл конфігурації для машини.
<*machine_name*>.hal	HAL для машини.
<*machine_name*>.prefs	Файл конфігурації для специфічних параметрів та налаштувань QtPlasmaC.
custom.hal	HAL-файл для налаштування користувачем.
custom_postgui.hal	HAL-файл для налаштування користувача, який запускається після ініціалізації графічного інтерфейсу.
shutdown.hal	HAL-файл, який запускається під час послідовності завершення роботи.
tool.tbl	Таблиця інструментів, що використовується для зберігання інформації про зміщення додаткових інструментів (писець тощо), що використовуються конфігурацією QtPlasmaC.
qtplasmac	Посилання на каталог, що містить поширені файли підтримки QtPlasmaC.
резервна копія	Директорія для резервних копій конфігураційних файлів.

Note
<*machine_name*> – це будь-яке ім’я, яке користувач ввів у поле «Ім’я машини» програми майстра налаштування.

Note
Користувацькі команди дозволені у файлах custom.hal та custom_postgui.hal, оскільки вони не перезаписуються під час оновлень.

Після першого запуску нової конфігурації в каталозі конфігурації будуть створені такі файли:

Ім’я файлу	**Функція**
<machine_name>_material.cfg	Файл для зберігання налаштувань матеріалу з розділу MATERIAL вкладки PARAMETERS.
update_log.txt	Файл для зберігання журналу основних оновлень. Основні оновлення – це ті, які вносять будь-які зміни до конфігурації користувача.
qtvcp.prefs	Файл, що містить налаштування QtVCP.
qtplasmac.qss	Файл, що зберігає таблицю стилів для поточного завантаженого сеансу QtPlasmaC.

Note
Конфігураційні файли (<*machine_name*>.ini та <*machine_name*>.hal), які створюються майстром конфігурації, містять пояснення вимог, що полегшують ручне редагування цих конфігурацій. Їх можна редагувати за допомогою будь-якого текстового редактора.

Note
Файл <*назва_машини*>.prefs є звичайним текстом і може бути редагуваний у будь-якому текстовому редакторі.

10.8.7.6 INI-файл

QtPlasmaC має деякі специфічні змінні файлу <*назва_машини*>.ini, а саме:

[FILTER] Розділ

Ці змінні є обов’язковими.

```
PROGRAM_EXTENSION = .ngc,.nc,.tap G-code File (*.ngc, *.nc, *.tap)
ngc               = qtplasmac_gcode
nc                = qtplasmac_gcode
tap               = qtplasmac_gcode
```

[RS274NGC] Розділ

Ці змінні є обов’язковими.

```
RS274NGC_STARTUP_CODE = G21 G40 G49 G80 G90 G92.1 G94 G97 M52P1
SUBROUTINE_PATH       = ./:../../nc_files
USER_M_PATH           = ./:../../nc_files
```

Note
для імперської конфігурації замініть G21 вище на G20.

Note
Обидва вищезазначені шляхи показують мінімальні вимоги.

Important
ДИВ. PATH TOLERANCE ДЛЯ ІНФОРМАЦІЇ ПРО RS274NGC_STARTUP_CODE, ПОВ’ЯЗАНУ З G64.

[HAL] Розділ

Ці змінні є обов’язковими.

```
HALUI           = halui (b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb''b''ob''b''вb''b' ↩
    'ob'')
HALFILE         = _<machine_name>_.hal (b''фb''b''ab''b''йb''b''лb'' HAL b''мb''b''ab''b' ↩
    'шb''b''иb''b''нb''b''иb'')
HALFILE         = plasmac.tcl (b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b' ↩
    'нb''b''иb''b''йb'' b''фb''b''ab''b''йb''b''лb'' HAL QtPlasmaC)
HALFILE         = custom.hal (b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b' ↩
    'ab''b''цb''b''ьb''b''кb''b''ib'' b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''иb'' HAL)
POSTGUI_HALFILE = postgui_call_list.hal (b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb'' ↩
    b''ob''b''вb''b''ob'')
SHUTDOWN        = shutdown.hal (b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''иb'' HAL b' ↩
    'дb''b''лb''b''яb'' b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''нb''b''яb'')
```

Note
Користувач може розміщувати власні команди HAL у файлі custom.hal, оскільки цей файл не перезаписується оновленнями QtPlasmaC.

[DISPLAY] Розділ

Ця змінна є обов’язковою.

```
DISPLAY = qtvcp qtplasmac      (b’’вb’’b’’иb’’b’’кb’’b’’оb’’b’’рb’’b’’иb’’b’’сb’’b’’тb’’b’ ↩
    ’оb’’b’’вb’’b’’уb’’b’’вb’’b’’аb’’b’’тb’’b’’иb’’ b’’рb’’b’’оb’’b’’зb’’b’’дb’’b’’іb’’b’ ↩
    ’лb’’b’’ьb’’b’’нb’’b’’уb’’ b’’зb’’b’’дb’’b’’аb’’b’’тb’’b’’нb’’b’’іb’’b’’сb’’b’’тb’’b’ ↩
    ’ьb’’ 16:9)
        = qtvcp qtplasmac_9x16 (b’’вb’’b’’иb’’b’’кb’’b’’оb’’b’’рb’’b’’иb’’b’’сb’’b’’тb’’b’ ↩
            ’оb’’b’’вb’’b’’уb’’b’’вb’’b’’аb’’b’’тb’’b’’иb’’ b’’рb’’b’’оb’’b’’зb’’b’’дb’’b’ ↩
            ’іb’’b’’лb’’b’’ьb’’b’’нb’’b’’уb’’ b’’зb’’b’’дb’’b’’аb’’b’’тb’’b’’нb’’b’’іb’’b’ ↩
            ’сb’’b’’тb’’b’’ьb’’ 9:16)
        = qtvcp qtplasmac_4x3  (b’’вb’’b’’иb’’b’’кb’’b’’оb’’b’’рb’’b’’иb’’b’’сb’’b’’тb’’b’ ↩
            ’оb’’b’’вb’’b’’уb’’b’’вb’’b’’аb’’b’’тb’’b’’иb’’ b’’рb’’b’’оb’’b’’зb’’b’’дb’’b’ ↩
            ’іb’’b’’лb’’b’’ьb’’b’’нb’’b’’уb’’ b’’зb’’b’’дb’’b’’аb’’b’’тb’’b’’нb’’b’’іb’’b’ ↩
            ’сb’’b’’тb’’b’’ьb’’ 4:3)
```

Тут описано кілька варіантів QtVCP: Налаштування INI-файлу QtVCP

Наприклад, наступний код запустить екран QtPlasmaC з роздільною здатністю 16:9 у повноекранному режимі:

```
DISPLAY = qtvcp -f qtplasmac
```

[TRAJ] Розділ

Ця змінна є обов’язковою.

```
SPINDLES = 3
```

[AXIS_X] Розділ

Ці змінні є обов’язковими.

```
MAX_VELOCITY     = b’’пb’’b’’оb’’b’’дb’’b’’вb’’b’’оb’’b’’їb’’b’’тb’’b’’иb’’ b’’зb’’b’’нb’’b ↩
    ’’аb’’b’’чb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’уb’’ b’’вb’’b’’іb’’b’’дb’’b’’пb’’b’’оb’’b’ ↩
    ’вb’’b’’іb’’b’’дb’’b’’нb’’b’’оb’’b’’мb’’b’’уb’’ b’’сb’’b’’уb’’b’’гb’’b’’лb’’b’’оb’’b’ ↩
    ’бb’’b’’іb’’
MAX_ACCELERATION = b’’пb’’b’’оb’’b’’дb’’b’’вb’’b’’оb’’b’’їb’’b’’тb’’b’’иb’’ b’’зb’’b’’нb’’b ↩
    ’’аb’’b’’чb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’уb’’ b’’вb’’b’’іb’’b’’дb’’b’’пb’’b’’оb’’b’ ↩
    ’вb’’b’’іb’’b’’дb’’b’’нb’’b’’оb’’b’’мb’’b’’уb’’ b’’сb’’b’’уb’’b’’гb’’b’’лb’’b’’оb’’b’ ↩
    ’бb’’b’’іb’’
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Y] Розділ

Ці змінні є обов’язковими.

```
MAX_VELOCITY     = b’’пb’’b’’оb’’b’’дb’’b’’вb’’b’’оb’’b’’їb’’b’’тb’’b’’иb’’ b’’зb’’b’’нb’’b ↩
    ’’аb’’b’’чb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’уb’’ b’’вb’’b’’іb’’b’’дb’’b’’пb’’b’’оb’’b’ ↩
    ’вb’’b’’іb’’b’’дb’’b’’нb’’b’’оb’’b’’мb’’b’’уb’’ b’’сb’’b’’уb’’b’’гb’’b’’лb’’b’’оb’’b’ ↩
    ’бb’’b’’іb’’
MAX_ACCELERATION = b’’пb’’b’’оb’’b’’дb’’b’’вb’’b’’оb’’b’’їb’’b’’тb’’b’’иb’’ b’’зb’’b’’нb’’b ↩
    ’’аb’’b’’чb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’уb’’ b’’вb’’b’’іb’’b’’дb’’b’’пb’’b’’оb’’b’ ↩
    ’вb’’b’’іb’’b’’дb’’b’’нb’’b’’оb’’b’’мb’’b’’уb’’ b’’сb’’b’’уb’’b’’гb’’b’’лb’’b’’оb’’b’ ↩
    ’бb’’b’’іb’’
OFFSET_AV_RATIO  = 0.5
```

[AXIS_Z] Розділ

Ці змінні є обов’язковими.

```
MIN_LIMIT        = b''тb''b''pb''b''ob''b''xb''b''иb'' b''нb''b''иb''b''жb''b''чb''b''eb''  ↩
    b''вb''b''eb''b''pb''b''xb''b''нb''b''ьb''b''ob''b''їb'' b''чb''b''ab''b''cb''b''тb''b' ↩
    'иb''b''нb''b''иb'' b''пb''b''лb''b''ab''b''нb''b''ob''b''кb'' b''cb''b''тb''b''ob''b' ↩
    'лb''b''yb''
MAX_VELOCITY     = b''пb''b''ob''b''дb''b''вb''b''ob''b''їb''b''тb''b''иb'' b''зb''b''нb''b ↩
    ''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''yb'' b''вb''b''іb''b''дb''b''пb''b''ob''b' ↩
    'вb''b''іb''b''дb''b''нb''b''ob''b''мb''b''yb'' b''cb''b''yb''b''гb''b''лb''b''ob''b' ↩
    'бb''b''ib''
MAX_ACCELERATION = b''пb''b''ob''b''дb''b''вb''b''ob''b''їb''b''тb''b''иb'' b''зb''b''нb''b ↩
    ''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''yb'' b''вb''b''іb''b''дb''b''пb''b''ob''b' ↩
    'вb''b''іb''b''дb''b''нb''b''ob''b''мb''b''yb'' b''cb''b''yb''b''гb''b''лb''b''ob''b' ↩
    'бb''b''ib''
OFFSET_AV_RATIO  = 0.5
```

Note
За винятком різання труб з кутовою віссю A, B або C, QtPlasmaC використовує функцію зовнішніх зміщень LinuxCNC для всіх рухів осі Z, а також для переміщення осі X та/або Y для заміни витратних матеріалів або відновлення різання під час паузи. Для отримання додаткової інформації про цю функцію, будь ласка, прочитайте External Axis Offsets в документації LinuxCNC.

10.8.8 Огляд графічного інтерфейсу QtPlasmaC

У наступних розділах буде надано загальний огляд макета QtPlasmaC.

10.8.8.1 Вихід з QtPlasmaC

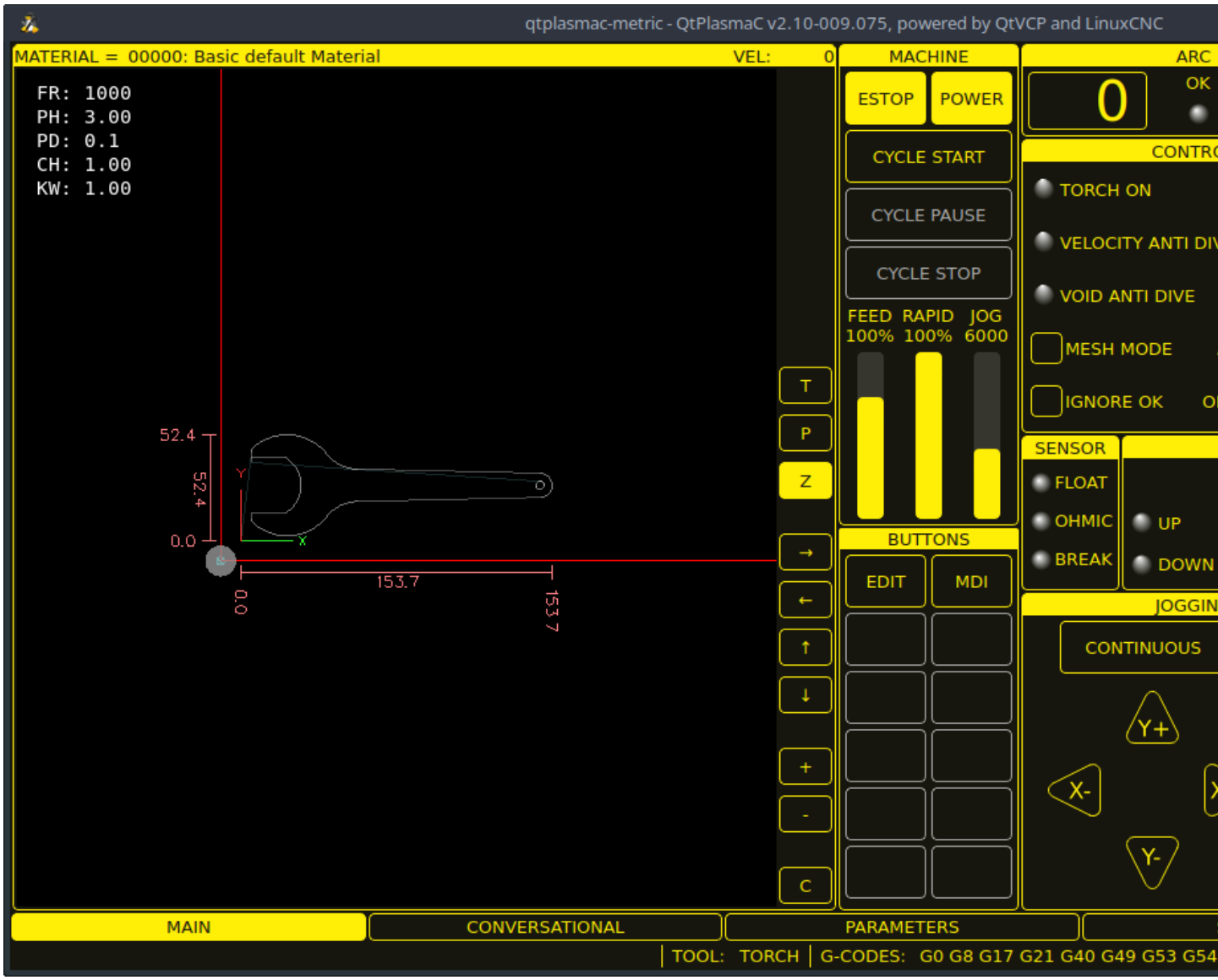
Вихід з QtPlasmaC або його завершення здійснюється одним із таких способів:

1. Натисніть кнопку закриття вікна в рядку заголовка вікна
2. Тривало натисніть кнопку **ЖИВЛЕННЯ** на вкладці MAIN (ОСНОВНЕ).

Попередження про завершення роботи можна відображати під час кожного завершення роботи, якщо встановити прапорець **Попередження про вихід** на вкладці вкладка НАЛАШТУВАННЯ.

10.8.8.2 ГОЛОВНА вкладка

Приклад знімка екрана вкладки QtPlasmaC MAIN у співвідношенні сторін **16:9**:



Деякі функції/можливості використовуються лише для певних режимів і не відображаються, якщо вони не потрібні для вибраного режиму QtPlasmaC.

Table 10.12: Особливості **PREVIEW WINDOW**

Ім'я	**Опис**
Матеріал	Верхній заголовок у цій області є клікабельним і відкриває випадаюче меню. Воно використовується для ручного вибору поточних параметрів різання матеріалу. Якщо у файлі матеріалів немає матеріалів, то буде відображатися тільки матеріал за замовчуванням.
НУ:	Відображає поточну швидкість подачі столу, включаючи як швидкі, так і ріжучі рухи. Під час різання, якщо активна функція Зниження швидкості, ця мітка оновлюється, щоб відобразити відсоток від початкової швидкості подачі, що використовується. Наприклад, «VEL@20%:» означає, що стіл ріже зі швидкістю 20% від запрограмованої швидкості подачі — зниження на 80%.
FR:	Якщо на вкладці вкладці НАЛАШТУВАННЯ вибрано опцію «Переглянути матеріал», це відображає швидкість подачі для поточного вибраного матеріалу.

Table 10.12: (continued)

Ім’я	**Опис**
PH:	Якщо на вкладці SETTINGS вибрано опцію «Переглянути матеріал», буде відображено висоту проколу для поточного вибраного матеріалу.
PD:	Якщо на вкладці SETTINGS вибрано опцію «Переглянути матеріал», це відображає затримку проколювання для поточного вибраного матеріалу.
CH:	Якщо на вкладці вкладці НАЛАШТУВАННЯ вибрано опцію «Переглянути матеріал», буде відображено висоту різу для поточного вибраного матеріалу.
CA:	Якщо на вкладці вкладці НАЛАШТУВАННЯ вибрано опцію «Переглянути матеріал» і зв’язок RS485 увімкнено, відображається сила струму різання для поточного вибраного матеріалу.
T	Ця кнопка змінює preview на повний табличний вигляд зверху вниз.
P	Ця кнопка змінює preview на ізометричний вигляд.
Z	Ця кнопка змінює preview на вигляд зверху вниз.
→	Ця кнопка переміщує preview праворуч.
←	Ця кнопка переміщує preview ліворуч.
↑	Ця кнопка переміщує preview вгору.
↓	Ця кнопка перегортає preview вниз.
+	Ця кнопка масштабує preview.
-	Ця кнопка масштабує preview.
C	Ця кнопка очищує графік у реальному часі.

Table 10.13: **МАШИННЕ** представлення

Ім’я	**Опис**
ЕСТІЙ	Встановлення типу Estop = 0 у розділі **[GUI_OPTIONS]** файлу <*machine_name*>.prefs змінить цю кнопку на індикатор стану апаратного аварійного вимкнення. Це стандартна поведінка. Встановлення параметра Estop type = 1 у розділі **[GUI_OPTIONS]** файлу <*machine_name*>.prefs приховає цю кнопку. Встановлення параметра Estop type = 2 у розділі **[GUI_OPTIONS]** файлу <*machine_name*>.prefs дозволить цій кнопці діяти як аварійна зупинка GUI.
ПОТУЖНІСТЬ	Ця кнопка вмикає графічний інтерфейс користувача і дозволяє QtPlasmaC/LinuxCNC керувати апаратним забезпеченням. Натискання і утримання кнопки **POWER** довше двох секунд відкриє діалогове вікно для виходу з програми QtPlasmaC.
ПОЧАТОК ЦИКЛУ	Ця кнопка запускає цикл для будь-якого завантаженого файлу G-коду.
ЦИКЛ ПАУЗА	Ця кнопка призупиняє цикл для будь-якого завантаженого файлу G-коду. Якщо цикл призупинено, ця кнопка відображатиме напис **ВІДНОВИТИ ЦИКЛ** та блиматиме. Натискання кнопки **ВІДНОВИТИ ЦИКЛ** відновить цикл.

Table 10.13: (continued)

Ім'я	**Опис**
ЗУПИНКА ВЕЛОСИПЕДУ	Ця кнопка зупиняє будь-який цикл, що активно виконується або призупинений. Це включає: - Програми G-коду - Імпульс пальника, якщо імпульс був запущений під час **ПРИЗУПИНЕННЯ ЦИКЛУ** (це також скасує виконання призупиненої програми G-коду) - Тест зонда - Фреймінг - Ручне різання
КОРМИ	Цей повзунок замінює швидкість подачі для всіх рухів подачі. Будь-яке значення, відмінне від 100%, призведе до блимання позначки. Натискання на позначку поверне повзунок до значення 100%.
ШВИДКИЙ	Цей повзунок замінює швидкість для всіх швидких переміщень. Будь-яке значення, відмінне від 100%, призведе до блимання підпису. Натискання на підпис поверне повзунок до 100%.
JOG	Цей повзунок встановлює швидкість поштовхового переміщення. Натискання на мітку поверне повзунок до лінійної швидкості за замовчуванням, встановленої у файлі <*назва_машини*>.ini.

КПОНКА Панель кнопок містить кнопки, корисні для роботи пристрою.

Кнопки **EDIT** та **MDI** є постійними, всі інші кнопки програмуються користувачем у файлі <*machine_name*>.prefs.

Див. custom user buttons для отримання детальної інформації про користувацькі кнопки.

Ім'я	**Опис**
РЕДАГУВАТИ	Ця кнопка відкриває редактор G-коду для поточної завантаженої програми.
MDI	Ця кнопка переводить QtPlasmaC у режим ручного введення даних (MDI), в якому над вікном G-коду відображається ІСТОРІЯ MDI та поле введення. Після натискання цієї кнопки з'явиться напис «MDI CLOSE». Натискання **MDI CLOSE** закриє MDI. Додаткову інформацію про MDI див. у розділі MDI.
ОМІЧНИЙ ТЕСТ	Ця кнопка вмикає вихідний сигнал «Ohmic Probe Enable» (Увімкнути омічний зонд), і якщо вхід омічного зонда спрацьовує, на панелі SENSOR (СЕНСОР) загоряється світлодіодний індикатор. Основна мета цього — забезпечити можливість швидкого тестування на наявність короткого замикання в наконечнику пальника.
ТЕСТ ЗОНДА	Ця кнопка ініціює Probe Test.
ОДИНАРНИЙ РІЗ	Ця кнопка відкриє діалогове вікно для запуску автоматичного Одинарний розріз.
ЗВИЧАЙНИЙ РІЗ	Ця кнопка перемикатиметься між Cut-types (NORMAL CUT та PIERCE ONLY).
ІМПУЛЬС ФАКЕЛУ	Ця кнопка ініціює Імпульс пальника.

Table 10.15: **ARC**

Ім’я	**Режими**	**Опис**
Напруга дуги	0, 1	Відображає фактичну напругу дуги.
OK	0, 1, 2	Показує стан сигналу «Дуга в порядку».
+	0, 1	Кожне натискання цієї кнопки збільшуватиме цільову напругу на величину порогової напруги THC (змінена відстань буде дорівнювати Висоті на вольт * Порогова напруга THC).
-	0, 1	Кожне натискання цієї кнопки знижуватиме цільову напругу на величину порогової напруги THC (змінена відстань буде дорівнювати Висоті на вольт * Порогова напруга THC).
ЗАМІНИТИ	0, 1	Клацання на цю мітку поверне будь-яке значення напруги до 0,00.

Table 10.16: **КОНТРОЛЬ**

Ім’я	**Режими**	**Опис**
ФАКЕЛ УВІМКНЕНО	0, 1, 2	Вказує стан вихідного сигналу «Пальник увімкнено».
УВІМК. ФАКЕЛ	0, 1, 2	Це поле перемикає між увімкненням і вимкненням пальника. Це поле за замовчуванням не заповнене (вимкнене) при першому запуску QtPlasmaC. Це поле необхідно заповнити, щоб змінити його на «Пальник увімкнено», перш ніж можна буде розпочати різання матеріалу. Якщо це поле не заповнене, запуск завантаженої програми призведе до того, що машина виконає цикл без запалювання пальника. Це іноді називають «сухим запуском». Якщо у користувача встановлений лазер, то також можна виконати сухий запуск з лазером. Детальні інструкції див. у розділі розділ LASER.
ШВИДКІСТЬ ANTI DIVE	0, 1, 2	Вказує на те, що THC заблоковано на поточній висоті через те, що швидкість різання впала нижче відсотка порогу швидкості проти занурення (VAD), встановленого на PARAMETERS Tab.
УВІМКНЕННЯ ЗАХИСТУ ВІД ЗАНЯТТЯ ПРОТИ ЗАНУРЮВАННЯ	0, 1, 2	Це поле перемикається між увімкненням та вимкненням функції захисту від занурення (VELOCITY ANTI DIVE).
VOID ANTI DIVE	0, 1	Вказує на те, що THC заблоковано через виявлення порожнечі.
УВІМКНЕННЯ ЗАХИСТУ ВІД ЗАНУРЮВАННЯ VOID	0, 1	Це поле перемикається між увімкненням та вимкненням функції захисту від занурення VOID.

Table 10.16: (continued)

Ім’я	**Режими**	**Опис**
РЕЖИМ СІТКИ	0, 1, 2	Цей прапорець увімкне або вимкне Mesh Mode для різання розширеного металу. Цей прапорець можна увімкнути або вимкнути в будь-який момент під час звичайного різання. Режим сітки: - Для запуску руху машини необхідний сигнал Arc OK. - Вимкне THC. - Не зупинить рух машини у разі втрати сигналу Arc OK. - Автоматично вибере режим CPA, якщо використовується зв’язок PowerMax. Докладнішу інформацію див. у розділі Mesh Mode (expanded metal).
АВТОВОЛЬТИ	0, 1	Це поле вмикає або вимикає Auto Volts.
ІГНОРУВАТИ OK	0, 1, 2	Цей прапорець визначає, чи ігноруватиме QtPlasmaC сигнал Arc OK. Цей прапорець можна ввімкнути або вимкнути в будь-який момент під час звичайного різання. Крім того, цей режим можна ввімкнути або вимкнути за допомогою відповідних кодів M у запущеній програмі. Режим ігнорування сигналу Arc OK: - Не вимагатиме отримання сигналу Arc OK перед початком руху машини після подачі сигналу «Torch On» (Увімкнення пальника). - Вимкне THC. - Не зупинить рух машини у разі втрати сигналу Arc OK. Докладнішу інформацію див. у розділі Ignore Arc Ok.
Омічний зонд	0, 1, 2	Це поле вмикає або вимикає вхід омічного зонда. Якщо вхід омічного зонда вимкнено, світлодіод омічного зонда все одно показуватиме стан входу зонда, але результати омічного зонда ігноруватимуться.
RS485	0, 1, 2	Це поле вмикає або вимикає зв’язок з PowerMax. Ця кнопка видима лише тоді, коли в розділі `[POWERMAX]` файлу *<machine_name>*.prefs налаштовано опцію PM_PORT.
Статус	0, 1, 2	Коли комунікації PowerMax увімкнені, на екрані відображатиметься одне з наступних повідомлень: **CONNECTING** (Підключення), **CONNECTED** (Підключено), **COMMS ERROR** (Помилка комунікації) або **Fault Code** (Код помилки). Докладнішу інформацію див. у розділі PowerMax Communications.

Table 10.17: **ДАТЧИК**

Ім’я	**Опис**
З ПОПЛАТОЮ	Показує, що поплавковий вимикач активовано.
ОМІК	Означає, що зонд виявився на матеріалі.

Table 10.17: (continued)

Ім’я	**Опис**
ПЕРЕРВА	Вказує на те, що датчик відриву пальника активовано.

Table 10.18: **THC**

Ім’я	**Опис**
ENABLE	Це поле визначає, чи буде THC увімкнено чи вимкнено під час різання.
УВІМКНЕНО	Цей світлодіодний індикатор показує, чи THC увімкнено чи вимкнено.
АКТИВНИЙ	Цей світлодіод вказує на те, що THC активно керує віссю Z.
ВГОРУ	Цей світлодіод показує, що контролер кульшового супорта (THC) дає команду осі Z на підйом.
ВНИЗ	Цей світлодіод показує, що контролер кульшового суглоба (THC) дає команду осі Z опуститися.

Note
Під час паузи в русі цей розділ стане CUT RECOVERY

Ім’я	**Опис**
БЕЗПЕРЕРВНИЙ	Ця кнопка з випадаючого списку змінюватиме крок поштовху. Параметри визначаються значеннями в розділі **[DISPLAY]** файлу *<назва_машини>*.ini та починаються з мітки ”INCREMENTS =”.
ШВИДКО	Ця кнопка перемикатиметься між ШВИДКОЮ, що є лінійною швидкістю за замовчуванням у файлі *<назва_машини>*.ini, та ПОВІЛЬНОЮ, що становить 10% від значення за замовчуванням.
Y+	Ця кнопка переміщує вісь Y у позитивному напрямку.
Y-	Ця кнопка переміщує вісь Y у негативному напрямку.
X+	Ця кнопка переміщує вісь X у позитивному напрямку.
X-	Ця кнопка переміщує вісь X у негативному напрямку.
Z+	Ця кнопка переміщує вісь Z у позитивному напрямку.
Z-	Ця кнопка переміщує вісь Z у негативному напрямку.

Note
Під час паузи руху ця секція буде відображатися у верхній частині панелі JOGGING. У наступній секції буде розглянуто кожну кнопку, що міститься на цій панелі. Детальний опис функції відновлення після розриву див. у розділі CUT RECOVERY.

Ім’я	**Опис**
ПОВЗУНЕЦЬ ПОДАЧІ РУХУ НА ПАУЗІ	У разі призупинення програми цей інтерфейс дозволяє руху по осях X/Y слідувати запрограмованому шляху в зворотному або прямому напрямку. Діапазон цього повзунка становить від 1% до 100% швидкості подачі різання для поточного вибраного матеріалу.
КОРМИ	Це відображає швидкість подачі призупиненого руху.

Ім’я	Опис
REV	У разі призупинення програми ця кнопка перемістить верстат у зворотному напрямку по запрограмованому шляху, доки він не досягне останньої команди M3, яка була виконана або яку QtPlasmaC намагався виконати перед призупиненням програми.
FWD	У разі призупинення програми ця кнопка переміщуватиме машину вперед по запрограмованому шляху на невизначений термін до кінця програми, пропускаючи команди M3.
СКАСУВАТИ ПЕРЕМІЩЕННЯ	Ця кнопка скасує будь-який рух відновлення різання, що був виконаний, і поверне пальник у положення, з якого було розпочато рух відновлення різання. Зверніть увагу, що якщо для переміщення пальника було використано FWD або REV, CANCEL не поверне пальник у положення, в якому він перебував на момент паузи.
РУХ x.xxx	Це показує величину переміщення, яке відбуватиметься при кожному натисканні клавіші зі стрілкою, у напрямку, в якому була натиснута клавіша зі стрілкою. Це значення, що відображається під MOVE, представляє ширину пропилу поточного вибраного матеріалу.
СТРІЛКИ НАПРЯМКУ	Ці кнопки переміщуватимуть пальник у напрямку, вказаному на відстань, що дорівнює одній ширині різу (поточного вибраного матеріалу) за одне натискання.

Table 10.21: **ВІКНО G-КОДУ**

Ім’я	Опис
ЧИСТО	Ця кнопка очистить поточну відкриту програму. Якщо файл відкритий, буде вибрано матеріал за замовчуванням. Якщо файл не відкритий, preview буде скинуто до повного перегляду таблиці зверху вниз. Якщо пальник (T0) не був активним інструментом, він буде вибраний. Попередні повідомлення про помилки та стан помилки будуть очищені. Тип різання буде встановлений на NORMAL CUT.
ВІДКРИТИ	Ця кнопка відкриє панель ВІДКРИТТЯ ФАЙЛУ поверх ВІКНА ПОПЕРЕДНЬОГО ПЕРЕГЛЯДУ.
ПЕРЕЗАВАНТАЖИТИ	Ця кнопка перезавантажить поточний завантажений файл G-коду.

Table 10.22: **DRO**

Ім’я	Опис
ДОМАШНЯ СТОРІНКА ВСІ	Ця кнопка перемістить усі осі у вихідне положення в порядку, встановленому параметром HOME_SEQUENCE у файлі <*machine_name*>.ini.
WCS G54	Ця кнопка випадаючого списку змінить поточне зміщення робочої точки.
КАМЕРА	Ця кнопка відображає панель CAMVIEW у верхній частині вікна PREVIEW WINDOW і дозволяє користувачеві встановити початок координат з обертанням або без нього. Детальні інструкції див. у розділі CAMERA section. Ця кнопка не буде видима, доки в файлі <*machine_name*>.prefs не буде встановлено зміщення CAMERA.

Table 10.22: (continued)

Ім’я	**Опис**
ЛАЗЕР	Ця кнопка дозволяє користувачеві використовувати лазер для встановлення початку координат з обертанням або без нього. Детальні інструкції див. у розділі LASER. Ця кнопка не буде видима, доки в файлі <*machine_name*>.prefs не буде встановлено зміщення LASER.
X0 Y0	Ця кнопка встановить поточну позицію на X0 Y0.
ГОЛОВНА [AXIS]	Ця кнопка перемістить відповідну вісь до вихідного положення.
0 [AXIS]	Ця кнопка випадаючого меню відображає наступні опції: **Нуль** - обнуляє вісь. **Встановити** - відкриває діалогове вікно для ручного введення координати осі. **Поділити на 2** - ділить поточну координату, що відображається в DRO, на два. **Встановити останнє значення** - встановлює вісь на попередньо встановлену координату.

10.8.8.3 Попередній перегляд

Екран попереднього перегляду QtPlasmaC має можливість перемикатися між різними режимами перегляду та відображеннями, а також збільшувати та зменшувати масштаб, а також панорамувати по горизонталі та вертикалі.

Під час першого запуску QtPlasmaC, вигляд Z (зверху вниз) буде вибрано як вигляд за замовчуванням для завантаженого файлу G-коду, але відображатиметься повний табличний вигляд.

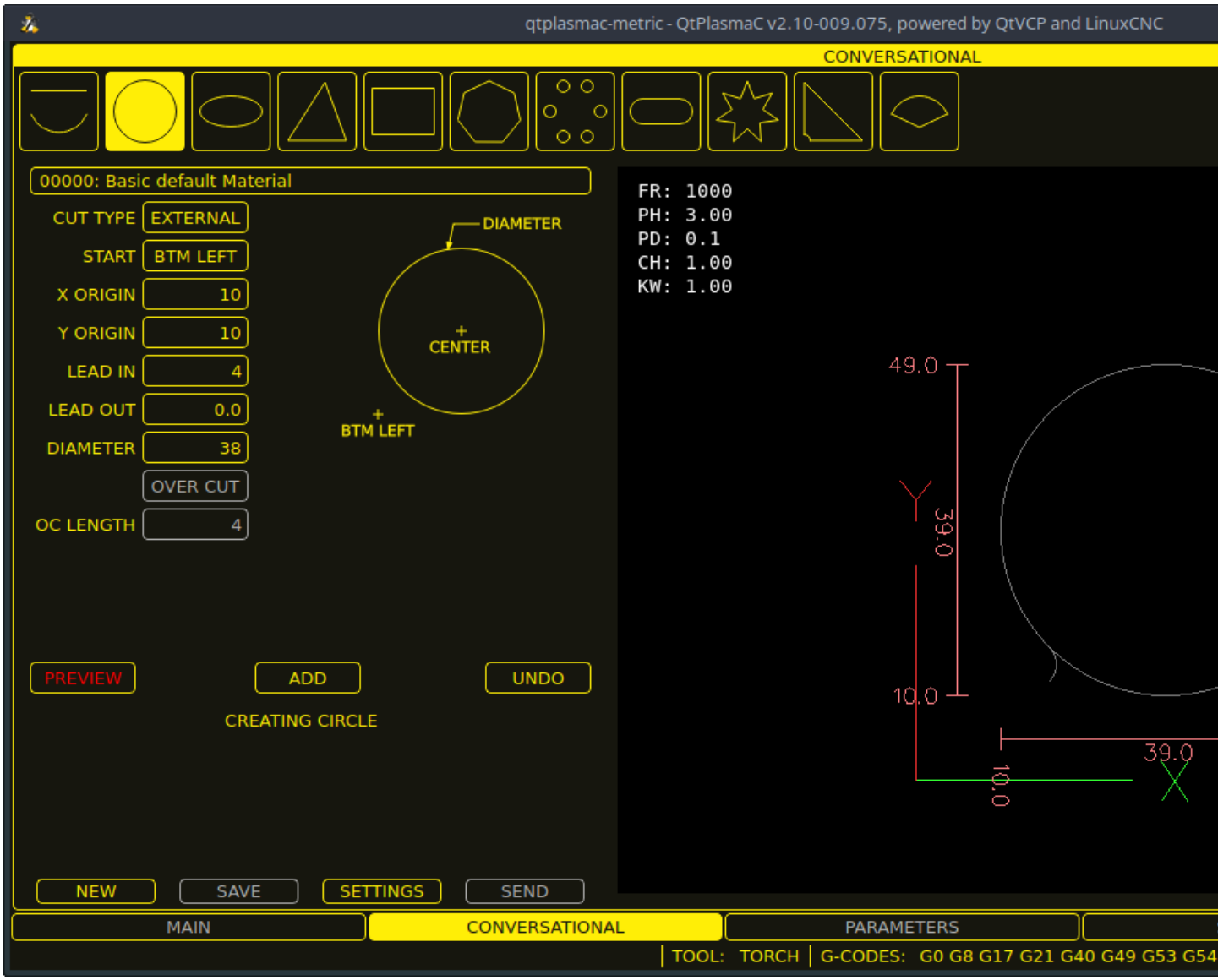
Після завантаження файлу G-коду дисплей зміниться на вибраний вигляд.

Якщо файл G-коду не завантажено, автоматично відображатиметься повна таблиця незалежно від того, який вигляд наразі вибрано (підсвічена кнопка, що позначає наразі вибраний вигляд, не змінюватиметься).

Якщо відображається повна таблиця через те, що файл G-коду не завантажено, і користувач бажає змінити орієнтацію перегляду, натискання клавіші Z або P змінить відображення на новообраний перегляд. Якщо користувач бажає відобразити повну таблицю, зберігаючи поточний обраний перегляд як перегляд за замовчуванням для завантаженого файлу G-коду, натискання клавіші CLEAR дозволить це зробити і зберегти обрану орієнтацію перегляду для наступного завантаження файлу G-коду.

10.8.8.4 РОЗМОВНА вкладка

Приклад знімка екрана вкладки QtPlasmaC CONVERSATIONAL у співвідношенні сторін **16:9**:



Вкладка CONVERSATIONAL дозволяє користувачеві швидко програмувати різні прості форми для швидкого різання без необхідності використання програмного забезпечення CAM.

Див. Розмовна бібліотека фігур для отримання детальної інформації про функцію «Розмовний режим».

Цю вкладку можна приховати, щоб оператор не міг користуватися функцією розмови. Це можна зробити, підключивши контакт до фізичного ключового перемикача або подібного пристрою, або ж налаштувати в файлі HAL за допомогою такої команди:

```
setp qtplasmac.conv_disable 1
```

10.8.8.5 Вкладка ПАРАМЕТРИ

Приклад знімка екрана вкладки QtPlasmaC PARAMETERS у співвідношенні сторін **16:9**:



Деякі функції/можливості використовуються лише для певних режимів і не відображаються, якщо вони не потрібні для вибраного режиму QtPlasmaC.

Ця вкладка використовується для відображення параметрів конфігурації, які рідко змінюються.

Цю вкладку можна приховати, щоб налаштування машини не могли бути змінені сторонніми особами. Це можна зробити, підключивши контакт до фізичного ключового вимикача або подібного пристрою, або ж налаштувати в файлі HAL за допомогою такої команди:

```
setp qtplasmac.param_disable 1
```

Table 10.23: **КОНФІГУРАЦІЯ - ARC**

Ім’я	Режими	Опис
Таймер запуску при збої	0, 1, 2	Це встановлює час (у секундах), який QtPlasmaC чекатиме між командою ”Увімкнення пальника” та отриманням сигналу ”Дуга в порядку”, перш ніж спрацює тайм-аут та відобразиться повідомлення про помилку.

Table 10.23: (continued)

Ім’я	Режими	Опис
Макс. кількість запусків	0, 1, 2	Це встановлює кількість спроб QtPlasmaC запустити дугу.
Затримка повторної спроби	0, 1, 2	Це встановлює час (у секундах) між збоєм дуги та наступною спробою запалювання дуги.
Шкала напруги	0, 1	Це встановлює шкалу вхідної напруги дуги та використовується для відображення правильної напруги дуги. Для початкового налаштування див. Calibration Values.
Зсув напруги	0, 1	Це встановлює зміщення напруги дуги та використовується для відображення нульового значення напруги, коли на вході немає напруги дуги. Для початкового налаштування див. Calibration Values.
Висота на вольт	0, 1, 2	Це встановлює відстань, на яку потрібно перемістити пальник, щоб змінити напругу дуги на один вольт. Використовується лише для ручного керування висотою.
Високі вольти	0	Це встановлює поріг напруги, нижче якого сигнал «Дуга в порядку» є дійсним.
Низький вольт	0	Це встановлює поріг напруги, вище якого сигнал «Дуга в порядку» є дійсним.

Note
При встановленні значень OK Low Volts і OK High Volts в режимі 0, напруга відсічення стабільної дуги повинна бути більшою за значення OK Low Volts, але меншою за значення OK High Volts, щоб QtPlasmaC отримав дійсний сигнал Arc OK. Для більшої зрозумілості: щоб отримати дійсний сигнал Arc OK, напруга дуги повинна знаходитися між цими двома межами.

Table 10.24: **КОНФІГУРАЦІЯ - ЗОНДУВАННЯ**

Ім’я	Опис
Подорожі на плаву	Це встановлює величину переміщення поплавкового вимикача перед завершенням роботи ланцюга поплавкового вимикача. Цю відстань можна виміряти за допомогою кнопки Probe Test (Тест зонда) та методу, описаного в розділі Початкова настройка.
Швидкість зонда	Це встановлює швидкість, з якою пальник буде зондувати матеріал після досягнення висоти зонда.
Висота зонда	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.
Омічне зміщення	Це встановлює відстань над матеріалом, на яку має пройти пальник після успішного омічного зондування. Це в основному використовується для компенсації високих швидкостей зондування.

Table 10.24: (continued)

Ім’я	Опис
Омічні повторні спроби	Це встановлює кількість разів, коли QtPlasmaC повторно спробує спрацювати через несправний омічний зонд, перш ніж повернутися до поплавкового вимикача для виявлення матеріалу.
Пропустити IHS	Це встановлює поріг відстані, який використовується для визначення того, чи можна пропустити початкове вимірювання висоти (зонд) для поточного розрізу, див. IHS Skip.
Швидкість зміщення	Це встановлює швидкість, з якою зонд рухатиметься до положення зміщення по осях X та Y.

Note
Якщо час між контактом пальника з матеріалом і моментом, коли пальник піднімається і зупиняється на висоті проколювання, здається надмірним, див. розділ «Плазма: зондування, зондування» для можливого вирішення.

Table 10.25: **КОНФІГУРАЦІЯ - БЕЗПЕКА**

Ім’я	Опис
Безпечна висота	This sets the height above the material that the torch will retract to before executing rapid moves. If set to zero, then Max Offset (plasmac.max-offset) will be used for the safe height. This defaults to 5mm (0.197”). Refer to the Heights Diagrams for a visual representation.

Table 10.26: **КОНФІГУРАЦІЯ - РОЗПИС**

Ім’я	Опис
Затримка активації	Це встановлює затримку (у секундах) від моменту отримання команди маркера до його активації. Це дозволяє маркеру досягти поверхні матеріалу перед його активацією.
Затримка	Це встановлює затримку (у секундах), щоб механізм розмітки міг запуститися перед початком руху.

Table 10.27: **КОНФІГУРАЦІЯ - ВИЗНАЧЕННЯ**

Ім’я	Опис
Поріг	Це встановлює напругу дуги, за якої почнеться таймер затримки. 0 В запускає затримку, коли активується сигнал увімкнення пальника.
Час увімкнення	Це встановлює час (у мілісекундах), протягом якого пальник увімкнений після досягнення порогового значення напруги.

Table 10.28: **КОНФІГУРАЦІЯ - ТІЛЬКИ ПРОКОЛ**

Ім’я	**Опис**
Зсув по осі X	Переміщує точку проколювання на цю відстань вздовж осі X під час проколювання в режимі «Тільки проколювання».
Зсув по осі Y	Переміщує точку проколювання на цю відстань вздовж осі Y під час проколювання в режимі «Тільки проколювання».

Table 10.29: **КОНФІГУРАЦІЯ - РУХ**

Ім’я	**Опис**
Швидкість налаштування	Швидкість осі Z для переміщень налаштування (переміщення до висоти зонда, висоти проколу, висоти різання тощо).

Note
Швидкість налаштування не впливає на швидкість THC, яка здатна досягти швидкості, що відображається в полі Макс. швидкість.

Table 10.30: **КОНФІГУРАЦІЯ - THC**

Ім’я	**Режими**	**Опис**
Затримка	0, 1, 2	Це встановлює затримку (у секундах), що вимірюється від моменту отримання сигналу «Дуга в порядку» до активації контролера висоти пальника (THC). Ця функція доступна лише тоді, коли автоматичний THC не ввімкнено.
Кількість зразків	0, 1	Це встановлює кількість послідовних показників напруги дуги в межах порогового значення вибірки THC, необхідних для активації контролера висоти пальника (THC). Ця функція доступна лише тоді, коли ввімкнено автоматичний THC.
Поріг вибірки	0, 1	Це встановлює максимальне відхилення напруги, дозволене для підрахунку зразків THC. Це доступно лише тоді, коли ввімкнено автоматичний THC.
Поріг	0, 1	Це встановлює допустиме відхилення напруги від цільової напруги, перш ніж THC здійснить рухи для корекції висоти пальника.
Швидкість (PID-P)	0, 1, 2	Це встановлює пропорційне підсилення для контуру PID-регулятора THC. Це приблизно дорівнює тому, як швидко THC намагається виправити зміни висоти.
ЯКИЙ поріг	0, 1, 2	(Запобігання зануренню швидкості) Встановлює відсоток поточної швидкості подачі різання, до якого верстат може сповільнитися, перш ніж заблокувати різак різання, щоб запобігти зануренню пальника.
Порожній схил	0, 1	(Захист від несправності) Встановлює розмір зміни напруги різання за секунду, необхідної для блокування THC, щоб запобігти зануренню пальника (вищі значення потребують більшої зміни напруги для блокування THC).

Table 10.30: (continued)

Ім’я	Режими	Опис
PID-I	0, 1	Це встановлює інтегральне підсилення для контуру PID-регулятора THC. Інтегральне підсилення пов’язане із сумою помилок у системі з плином часу та не завжди потрібне.
PID-D	0, 1	Це встановлює коефіцієнт похідної підсилення для контуру PID-регулятора THC. Коефіцієнт похідної працює для гасіння системних коливань та зменшення коливань надмірної корекції і не завжди потрібен.

Доступні два методи активації THC, які вибираються за допомогою кнопки **Автоматична активація**. Обидва методи починають обчислення, коли поточна швидкість пальника відповідає швидкості подачі різання, заданій для вибраного матеріалу:

1. Затримка активації (за замовчуванням) вибрано, коли не позначено параметр **Автоматична активація**. Цей метод використовує часову затримку, встановлену за допомогою параметра **Затримка**.
2. Автоматична активація вибирається, коли позначено опцію **Автоматична активація**. Цей метод визначає стабільність напруги дуги за допомогою параметрів **Кількість зразків** та **Поріг зразка**.

Note
Налаштування контуру PID є складним процесом і виходить за межі цього посібника користувача. Існує багато джерел інформації, які допоможуть зрозуміти та налаштувати контури PID. Якщо THC не вносить корективи достатньо швидко, рекомендується збільшувати коефіцієнт підсилення P невеликими кроками, доки система не почне працювати належним чином. Значне регулювання коефіцієнта підсилення P може призвести до надмірної корекції та коливань.

Кнопки ЗБЕРЕЖЕННЯ ТА ПЕРЕЗАВАНТАЖЕННЯ Кнопка **ЗБЕРЕГТИ** збереже поточні відображе параметри у файлі <*назва_машини*>.prefs.

Кнопка **RELOAD** перезавантажить усі параметри з файлу <*machine_name*>.prefs.

Table 10.31: **MATERIAL** - Параметри, активні для поточного розрізу.

Ім’я	Опис
Матеріал	Верхнє випадаюче меню використовується для ручного вибору поточних параметрів різання матеріалу. Якщо у файлі матеріалів немає матеріалів, то буде відображатися лише матеріал за замовчуванням.
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.
Ширина виїмки	This sets the kerf width for the currently selected material. Refer to the Heights Diagrams for a visual representation.

Table 10.31: (continued)

Ім’я	**Опис**
Пірс Хайт	This sets the pierce height for the currently selected material. Refer to the Heights Diagrams for a visual representation.
Пірс Затримка	Це встановлює затримку проколювання (у секундах) для вибраного матеріалу.
Висота зрізу	This sets the cut height for the currently selected material. Refer to the Heights Diagrams for a visual representation.
Швидкість подачі різання	Це встановлює швидкість подачі різання для вибраного матеріалу.
Зменшення струму	Це встановлює силу струму різання для вибраного матеріалу. Це візуальний індикатор лише для оператора, якщо не використовується зв’язок PowerMax.
Зрізані вольти	Це встановлює напругу різання для вибраного матеріалу.
Висота калюжі	Expressed as a percentage of Pierce Height, this sets the Puddle Jump height for the currently selected material. Typically used for thicker materials, Puddle Jump allows the torch to have an intermediate step between Pierce Height and Cut Height. If set, the torch will proceed from Pierce Height to P-Jump Height for a period of time (P-Jump Delay) before proceeding to Cut Height to effectively ”jump” over the molten puddle. Refer to the Heights Diagrams for a visual representation.
Затримка через калюжу	Це встановлює час (у секундах), протягом якого пальник залишатиметься на висоті P-Jump, перш ніж перейти до висоти різання.
Пауза в кінці	Це встановлює час (у секундах), протягом якого пальник залишатиметься увімкненим після завершення різання, перш ніж виконати команду M5 для вимкнення та підняття пальника. Докладнішу інформацію див. у розділі Пауза після завершення різання.
Тиск газу	Це налаштування встановлює тиск газу для вибраного матеріалу. Цей параметр дійсний лише за умови використання зв’язку PowerMax. 0 = Використовувати автоматичний режим тиску PowerMax.
Режим вирізання	Це налаштування встановлює режим різання для вибраного матеріалу. Цей параметр дійсний лише за умови використання зв’язку PowerMax. 1 = Нормальний 2 = CPA (Постійна пілотна дуга) 3 = Стрижка/Маркування

Note
Дивіться розділ товсті матеріали для отримання додаткової інформації про стрибок калюжі.

Кнопки ЗБЕРЕГТИ, ПЕРЕЗАВАНТАЖИТИ, СТВОРИТИ ТА ВИДАЛИТИ Кнопка **ЗБЕРЕГТИ** збереже поточний набір матеріалів у файлі <*назва_машини*>_material.cfg.

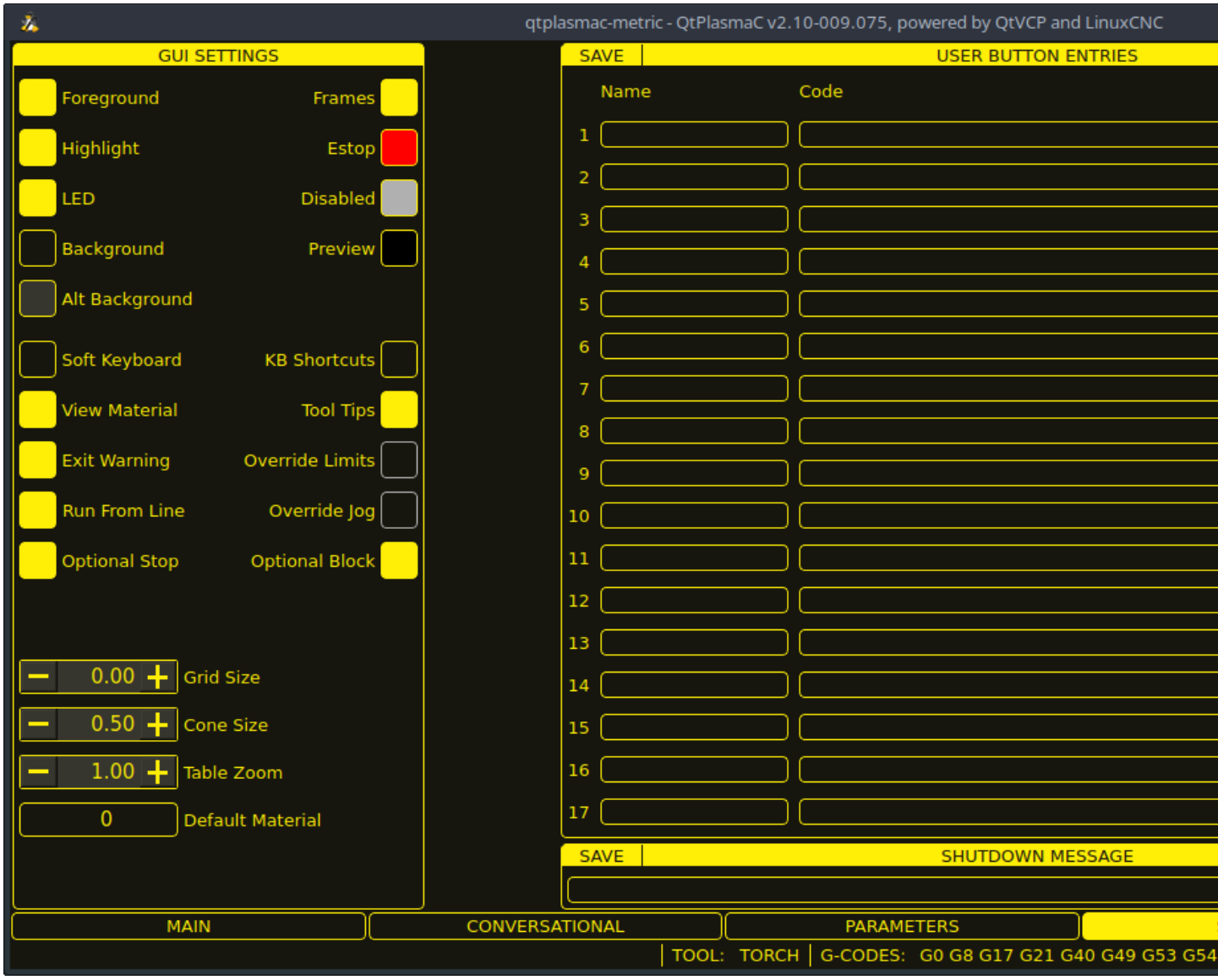
Кнопка **RELOAD** перезавантажить набір матеріалів з файлу <*machine_name*>_material.cfg.

Кнопка **NEW** (НОВИЙ) дозволяє додати новий матеріал до файлу матеріалів. Користувачеві буде запропоновано ввести номер матеріалу та його назву, всі інші параметри будуть зчитані з поточного вибраного матеріалу. Після введення даних QtPlasmaC перезавантажить файл матеріалів і відобразить новий матеріал. Потім необхідно буде налаштувати та зберегти параметри різання для нового матеріалу.

Кнопка **DELETE** використовується для видалення матеріалу. Після натискання на неї користувачеві буде запропоновано ввести номер матеріалу, який потрібно видалити, а потім ще раз підтвердити своє рішення. Після видалення файл матеріалу буде перезавантажено, а в списку, що розкривається, буде відображено матеріал за замовчуванням.

10.8.8.6 Вкладка НАЛАШТУВАННЯ

Приклад знімка екрана QtPlasmaC вкладка НАЛАШТУВАННЯ у співвідношенні сторін **16:9**:



Ця вкладка використовується для відображення параметрів конфігурації графічного інтерфейсу, тексту кнопок та тексту вимкнення, які змінюються рідко, а також деяких кнопок утиліт.

Цю вкладку можна приховати, щоб налаштування машини не могли бути змінені сторонніми особами. Це можна зробити, підключивши контакт до фізичного ключового вимикача або подібного пристрою, або ж налаштувати в файлі HAL за допомогою такої команди:

```
setp qtplasmac.settings_disable 1
```

НАЛАШТУВАННЯ ГРАФІЧНОГО ІНФОРМАЦІЙ ... У цьому розділі показано параметри, що впливають на зовнішній вигляд та поведінку графічного інтерфейсу.

Щоб повернути будь-які зміни кольору до значень за замовчуванням, див. розділ Повернення до стилю за замовчуванням.

Table 10.32: **GUI SETTINGS** Параметри, що впливають на зовнішній вигляд та поведінку графічного інтерфейсу.

Ім'я	**Опис**
Передній план	Ця кнопка дозволяє користувачеві змінювати колір переднього плану графічного інтерфейсу.
Виділити	Ця кнопка дозволяє користувачеві змінити колір підсвічування графічного інтерфейсу.
LED	Ця кнопка дозволяє користувачеві змінювати колір світлодіода графічного інтерфейсу.
Фон	Ця кнопка дозволяє користувачеві змінити колір фону графічного інтерфейсу.
Альтернативний фон	Ця кнопка дозволяє користувачеві змінити колір альтернативного фону графічного інтерфейсу.
Рамки	Ця кнопка дозволяє користувачеві змінювати колір рамок графічного інтерфейсу.
Estop	Ця кнопка дозволяє користувачеві змінити колір графічного інтерфейсу Estop.
Відключено	Ця кнопка дозволяє користувачеві змінити колір вимкнених функцій графічного інтерфейсу.
Попередній перегляд	Ця кнопка дозволяє користувачеві змінити колір фону вікна попереднього перегляду графічного інтерфейсу.
М'яка клавіатура	Цей перемикач дозволяє користувачеві вмикати або вимикати клавіатуру з м'яким сенсорним екраном. Якщо встановлено вбудовану віртуальну клавіатуру, тоді буде ввімкнено custom layouts.
Скорочення бази знань	Ця кнопка дозволяє користувачеві ввімкнути або вимкнути Keyboard Shortcuts у графічному інтерфейсі (наприклад, клавіші перемикання). Окрім стандартних клавіш перемикання, у розділі keyboard shortcuts доступний перелік додаткових клавіш швидкого доступу.
Переглянути матеріал	Ця кнопка дозволяє користувачеві вмикати або вимикати додавання візуального посилання, що показує основні налаштування різання матеріалу, до вікон попереднього перегляду вкладки MAIN та CONVERSATIONAL. Приклади: швидкість подачі, висота проколу, затримка проколу та висота різання. Якщо ввімкнено зв'язок PowerMax, буде показано силу струму різання.
Попередження про вихід	Ця кнопка дозволяє користувачеві ввімкнути або вимкнути постійне відображення попередження під час вимкнення системи. До попередження можна додати власне повідомлення, відредагувавши опцію EXIT WARNING MESSAGE у розділі **[GUI_OPTIONS]** файлу *<machine_name>*.prefs. Повідомлення можна зробити багаторядковим, додавши між рядками символ «\».
Додаткова зупинка	Цей перемикач дозволяє користувачеві вмикати або вимикати, чи буде запущена програма призупинятися після команди **M1**.
Бігти від лінії	Цей перемикач дозволяє користувачеві вмикати або вимикати Запуск з рядка. Якщо ввімкнено, користувач може клацнути на рядку G-коду та розпочати програму з цього рядка.

Table 10.32: (continued)

Ім’я	Опис
Замінити обмеження	Ця кнопка дозволяє користувачеві тимчасово замінити вхідні дані від кінцевого вимикача у випадку, якщо кінцевий вимикач спрацьовує під час роботи. Цю кнопку можна натиснути тільки тоді, коли спрацьовує кінцевий вимикач.
Перевизначити поштовх	Ця радіокнопка також дозволить поштовховий хід, коли поштовховий хід заборонено поплавковим вимикачем, вимикачем розриву або активацією омічного зонда. Цю кнопку можна натиснути, лише коли поштовховий хід заборонено.
Додатковий блок	Цей перемикач дозволяє користувачеві вмикати або вимикати пропускання рядків, що починаються з ”/”, якщо вони присутні у запущеній програмі.
Розмір сітки	Це дозволяє користувачеві змінювати розмір сітки у вікні попереднього перегляду на вкладці MAIN. Розмір сітки 0.0 вимкне сітку.
Розмір конуса	Це дозволяє користувачеві змінювати розмір конуса (який представляє поточний інструмент) у вікні попереднього перегляду на вкладці MAIN.
Масштаб таблиці	Це дозволяє користувачеві змінити рівень масштабування за замовчуванням для перегляду всієї таблиці зверху вниз у вікні попереднього перегляду на вкладці MAIN.

ЗАПИСИ КНОПОК КОРИСТУВАЧА КНОПКА КОРИСТУВАЧА

У цьому розділі показано текст, який з’являється на Custom User Buttons, а також код, пов’язаний з кнопкою користувача. Кнопки користувача можна змінювати, а нові налаштування використовувати без перезапуску LinuxCNC.

Текст та/або код можна редагувати будь-коли та будуть завантажені готовими до використання, якщо натиснути кнопку **ЗБЕРЕГТИ**.

Видалення тексту **Ім’я** та **Код** призведе до приховування цієї кнопки користувача, якщо натиснути кнопку **ЗБЕРЕГТИ**.

Щоб повернути всі тексти **Ім’я** та **Код** до їхніх останніх збережених значень, натисніть кнопку **ПЕРЕЗАГРУЗИТИ**.

Ім’я	Код
Текст, який відображається на кнопці	Код, який виконується при натисканні кнопки.

Note
Доступно 20 кнопок користувача, але не всі з них можуть відображатися залежно від вікна.

ПОВІДОМЛЕННЯ ПРО ВИХІД

У цьому розділі показано текст, який відображається у діалоговому вікні завершення роботи, якщо увімкнено **Попередження про вихід**.

Текст можна редагувати будь-коли та буде завантажено готовим до використання, якщо натиснути кнопку **ЗБЕРЕГТИ**.

Щоб повернути текст **ПОВІДОМЛЕННЯ ПРО ВИХІД** до останнього збереженого значення, натисніть кнопку **ПЕРЕЗАВАНТАЖИТИ**.

КОМУНАЛЬНІ ПОСЛУГИ Деякі стандартні утиліти LinuxCNC надаються як допомога в діагностиці проблем, які можуть виникнути:

- Halshow
- Halscope
- Halmeter
- Calibration
- Status

Крім того, надаються такі дві утиліти, специфічні для QtPlasmaC:

Кнопка **SET OFFSETS** використовується, якщо стіл має лазер або камеру для вирівнювання листа, рисувальник або використовує зондування з зміщенням. Необхідні зміщення для цих периферійних пристроїв потрібно застосовувати, дотримуючись процедури, описаної в розділі Peripheral Offsets.

Кнопка **BACKUP CONFIG** (Резервне копіювання конфігурації) створить повну резервну копію конфігурації машини для архівування або для допомоги в діагностиці несправностей. Стисла резервна копія конфігурації машини буде збережена в домашньому каталозі користувача Linux. Ім'я файлу буде <*machine_name*><*version*><date>_<time>.tar.gz, де <*machine_name*> — це ім'я машини, введене в майстрі конфігурації, <*version*> — це поточна версія QtPlasmaC, яку використовує користувач, <*date*> — це поточна дата (РР-ММ-ДД), а <*time*> — поточний час (ГГ-ММ-СС).

Перед створенням резервної копії журнал машини буде збережено у файлі в каталозі конфігурації під назвою machine_log_<date>_<time>.txt, де <*date*> та <*time*> мають формат, описаний вище. Цей файл разом із п'ятьма попередніми журналами машини також буде включено до резервної копії.

Ці файли не потрібні для QtPlasmaC і їх можна безпечно видалити в будь-який час.

10.8.8.7 Вкладка СТАТИСТИКА

Вкладка STATISTICS Tab містить статистичні дані, що дозволяють відстежувати знос витратних матеріалів та час виконання завдань. Ці статистичні дані відображаються як для поточного завдання, так і для загального підсумку. Статистичні дані попередніх завдань скидаються після запуску наступної програми. Загальні значення можна скинути окремо, натиснувши відповідну кнопку «RESET», або всі разом, натиснувши «RESET ALL».

Панель **RS485 PMX STATISTICS** (СТАТИСТИКА RS485 PMX) відображається тільки в тому випадку, якщо користувач має комунікацію Hypertherm PowerMax і встановлено дійсне з'єднання RS485 з PowerMax. На цій панелі відображається час **ARC ON TIME** (ЧАС ДІЇ ДУГИ) для PowerMax у форматі hh:mm:ss.

ЖУРНАЛ МАШИНИ також відображається на вкладці STATISTICS Tab. Цей журнал відображає будь-які помилки та/або важливу інформацію, що виникають під час поточної сесії LinuxCNC. Якщо користувач створює резервну копію конфігурації з вкладки SETTINGS Tab, журнал машини також включається до резервної копії.



10.8.9 Використання QtPlasmaC

Після успішної інсталяції QtPlasmaC рух по осі Z не є обов'язковою частиною програми різання G-коду. Насправді, якщо в програмі різання присутні будь-які посилання на вісь Z, стандартна конфігурація QtPlasmaC видалить їх під час завантаження програми.

Для надійного використання QtPlasmaC користувачеві **НЕ** слід використовувати будь-які зміщення осі Z, окрім зміщень системи координат (G54-G59.3). З цієї причини зміщення G92 було вимкнено в усьому графічному інтерфейсі.

QtPlasmaC автоматично додасть рядок G-коду для переміщення осі Z на правильну висоту на початку кожної програми G-коду.

Note
Можна зберегти рух по осі Z для використання з різними інструментами, додавши магічний коментар #<keep-z-motion>=1. Якщо для tube cutting використовується кутова вісь A, B або C, то в файлі G-коду необхідний рух по осі Z.

Інформація про версію - QtPlasmaC відображатиме інформацію про версію в заголовку головного вікна. Інформація буде відображатися наступним чином: «QtPlasmaC v*N.XXX.YYY* - powered by QtVCP on LinuxCNC vZ.Z.Z», де *N* - версія QtPlasmaC, *XXX* - версія компонента HAL (PlasmaC.comp), *YYY* - версія графічного інтерфейсу, а *Z.Z.Z* - версія LinuxCNC.

10.8.9.1 Системи одиниць

Усі налаштування та параметри в QtPlasmaC повинні бути в тих самих одиницях вимірювання, що й у файлі <*machine_name*>.ini, тобто метричних або імперських.

Якщо користувач намагається запустити файл G-коду, який знаходиться в системі «іншого» пристрою, то всі параметри, включаючи параметри файлу матеріалу, все одно повинні бути в одиницях виміру рідної машини. Будь-які подальші перетворення, необхідні для запуску файлу G-коду, будуть оброблятися автоматично програмою фільтрації G-коду.

Наприклад: якщо користувач мав метричну машину і хотів запустити файл G-коду, який був налаштований для різання матеріалу товщиною 1/4 дюйма з використанням імперських одиниць виміру (дюйм - G20), то користувач з метричною машиною повинен був переконатися, що номер матеріалу в файлі G-коду був встановлений на відповідний метричний матеріал, який потрібно різати, або що був створений новий матеріал з правильними метричними параметрами для метричного матеріалу, який потрібно різати. Якщо користувач метричної системи хоче вирізати файл G-коду, використовуючи імперський матеріал, то нові параметри матеріалу потрібно буде перетворити з імперських одиниць виміру в метричні під час їх введення.

10.8.9.2 Коди преамбули та постамбули

Наведені нижче строфи є мінімальними рекомендованими кодами для включення до преамбули та постамбули будь-якого файлу G-коду, який запускатиметься QtPlasmaC:

Метрика:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Імперський:

```
G20 G40 G49 G64p0.004 G80 G90 G92.1 G94 G97
```

Детальний опис кожного G-коду можна знайти в документації за посиланням:../gcode/g-code.html[тут].

Зверніть увагу, що в цьому посібнику користувача міститься кілька додаткових рекомендацій щодо кодів, які доцільно додавати як до преамбули, так і до пост-амбули, залежно від функцій, які користувач бажає використовувати.

10.8.9.3 Обов’язкові коди

Окрім коду преамбули, коду пост-амбули та коду руху X/Y, єдиним обов’язковим синтаксисом G-коду для QtPlasmaC для запуску програми G-коду з використанням пальника для різання є `M3 $0 S1` для початку різання та `M5 $0` для закінчення різання.

Для забезпечення зворотної сумісності допускається використання `M3 S1` замість `M3 $0 S1` для початку операції різання та `M5` замість `M5 $0` для завершення операції різання. Зверніть увагу, що це стосується лише операцій різання, для операцій розмітки та нанесення точок обов’язковим є використання ідентифікатора інструменту `$n`.

10.8.9.4 Координати

Див. налаштування рекомендовані налаштування осі Z.

Кожного разу, коли запускається LinuxCNC (QtPlasmaC), необхідне спільне повернення в початкове положення. Це дозволяє LinuxCNC (QtPlasmaC) встановити відомі координати кожної осі та встановити м'які обмеження на значення, вказані у файлі *<machine_name>*.ini, щоб запобігти аварійному зупинці машини під час нормального використання.

Якщо машина не має перемикачів додому, то користувачеві потрібно переконатися, що всі осі знаходяться в домашніх координатах, зазначених у файлі *<machine_name>*.ini, перш ніж переміщувати додому.

Якщо машина має перемикачі додому, то вона переміститься до вказаних координат додому, коли з'єднання будуть переведені додому.

Залежно від конфігурації верстата, буде або кнопка **Home All** (Повернути все додому), або кожну вісь потрібно буде перевести в початкове положення окремо. Використовуйте відповідну кнопку/кнопк для переведення верстата в початкове положення.

Як зазначено в розділі «plasma:initial-setup,Initial Setup» (плазма: початкова настройка, початкова настройка), при першому використанні QtPlasmaC рекомендується переконатися, що під паяльною лампою нічого немає, а потім перемістити вісь Z вниз, доки вона не зупиниться на мінімальному обмеженні вісі Z (MINIMUM_LIMIT), після чого натиснути 0 поруч із DRO вісі Z, щоб виконати «Touch Off» (дотик до нуля) з вибраною віссю Z, щоб встановити вісь Z на нульове зміщення. Це не потрібно повторювати.

Якщо користувач має намір розміщувати матеріал щоразу в одному і тому ж місці на столі, він може перемістити осі X і Y верстата у відповідне положення X0 Y0, встановлене програмним забезпеченням CAM, а потім виконати **Touch Off** (відключення) обох осей із нульовим зміщенням.

Якщо користувач має намір розмістити матеріал на столі випадковим чином, то перед запуском програми він повинен **встановити** осі X та Y у відповідній позиції.

10.8.9.5 Швидкість подачі різання

QtPlasmaC може зчитувати файл матеріалу для завантаження всіх необхідних параметрів різання. Щоб файл G-коду міг використовувати налаштування швидкості подачі різання з параметрів різання, використовуйте наступний код у файлі G-коду:

```
F#<_hal[plasmac.cut-feed-rate]>
```

Можна використовувати стандартне слово G-коду **F** для встановлення швидкості подачі різання наступним чином:

```
F 1000
```

Якщо використовується слово **F**, а значення слова **F** не відповідає швидкості подачі різання вибраного матеріалу, то під час завантаження файлу G-коду з'явиться діалогове вікно попередження.

10.8.9.6 Файл матеріалу

Для обробки матеріалів використовується файл матеріалів, який був створений для конфігурації верстата під час запуску майстра конфігурації і дозволяє користувачеві зручно зберігати відомі налаштування матеріалів для легкого виклику вручну або автоматично за допомогою G-коду. Результуючий файл material file має назву ***<machine_name>*_material.cfg**.

QtPlasmaC не вимагає використання файлу матеріалу. Натомість користувач може вручну змінити параметри різання в розділі MATERIAL на вкладці PARAMETERS Tab. Також не обов'язково

використовувати автоматичну зміну матеріалів. Якщо користувач не бажає використовувати цю функцію, він може просто опустити коди зміни матеріалів у файлі G-коду.

Також можливо не використовувати файл матеріалів та automatically load materials з файлу G-коду.

Номери матеріалів у файлі матеріалів не обов’язково мають бути послідовними, а також не обов’язково в числовому порядку.

Наведені нижче змінні є обов’язковими, і якщо під час завантаження файлу матеріалів будь-яка з них не знайдеться, з’явиться повідомлення про помилку.

- PIERCE_HEIGHT
- PIERCE_DELAY
- CUT_HEIGHT
- CUT_SPEED

Note
Якщо виконувати команду tube cutting з використанням магічного коментаря #<tube_cut>=1, то єдиною обов’язковою змінною є PIERCE_DELAY, усі інші змінні є необов’язковими.

Наступні змінні є необов’язковими. Якщо вони не виявлені або їм не призначено значення, їм буде призначено значення 0, і повідомлення про помилку не з’явиться.

- IM’Я
- KERF_WIDTH
- THC
- PUDDLE_JUMP_HEIGHT
- PUDDLE_JUMP_DELAY
- CUT_AMPS
- CUT_VOLTS
- PAUSE_AT_END
- GAS_PRESSURE
- CUT_MODE

Note
Номери матеріалів 1000000 і вище зарезервовані для тимчасових матеріалів.

Warning
Оператор несе відповідальність за забезпечення включення змінних, якщо вони є вимогою для виконання G-коду.

Файл матеріалу використовує такий формат:

```
[MATERIAL_NUMBER_1]
NAME                = b''ib''b''мb'''b''яb''
KERF_WIDTH          = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
THC                 = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' (0 = b''вb'' ↩
    b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob'', 1 = b''уb''b''вb''b''ib''b''мb''b' ↩
    'кb''b''нb''b''eb''b''нb''b''ob'')
b''Bb''b''Иb''b''Cb''b''Ob''b''Tb''b''Ab'' b''Пb''b''Pb''b''Ob''b''Kb''b''Ob''b''Лb''b' ↩
    'Уb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
b''3b''b''Ab''b''Пb''b''Ib''b''3b''b''Hb''b''Eb''b''Hb''b''Hb''b''Яb'' b''Пb''b''Pb''b' ↩
    'Ob''b''Kb''b''Ob''b''Лb''b''Уb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b' ↩
    'яb''
b''Bb''b''Иb''b''Cb''b''Ob''b''Tb''b''Ab'' b''Пb''b''Eb''b''Pb''b''Eb''b''Xb''b''Ob''b' ↩
    'Дb''b''Уb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
b''3b''b''Ab''b''Пb''b''Ib''b''3b''b''Hb''b''Eb''b''Hb''b''Hb''b''Яb'' b''Пb''b''Eb''b' ↩
    'Pb''b''Eb''b''Xb''b''Ob''b''Дb''b''Уb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b' ↩
    'нb''b''яb''
b''Bb''b''Иb''b''Cb''b''Ob''b''Tb''b''Ab'' b''Pb''b''Ib''b''3b''b''Ab''b''Hb''b''Hb''b' ↩
    'Яb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
b''Шb''b''Bb''b''Иb''b''Дb''b''Kb''b''Ib''b''Cb''b''Tb''b''Ьb'' b''Pb''b''Ib''b''3b''b' ↩
    'Ab''b''Hb''b''Hb''b''Яb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
b''Cb''b''Иb''b''Лb''b''Ab'' b''Cb''b''Tb''b''Pb''b''Уb''b''Mb''b''Уb'' b''Pb''b''Ib''b' ↩
    '3b''b''Ab''b''Hb''b''Hb''b''Яb'' = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b' ↩
    'яb'' (b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''дb''b''лb''b''яb'' b''ib''b''нb''b' ↩
    'фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''їb'', b''яb''b''кb''b''щb''b''ob'' b' ↩
    'нb''b''eb'' b''вb''b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob'' b''зb''b' ↩
    'вb'''b''яb''b''зb''b''ob''b''кb'' PowerMax)
CUT_VOLTS           = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' (b''тb''b' ↩
    'ib''b''лb''b''ьb''b''кb''b''иb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''иb'' 0 b''ib''  ↩
    1, b''яb''b''кb''b''щb''b''ob'' b''нb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''pb''b' ↩
    'иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''ab''b' ↩
    'вb''b''тb''b''ob''b''мb''b''ab''b''тb''b''иb''b''чb''b''нb''b''eb'' b''зb''b''чb''b' ↩
    'иb''b''тb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab''b''пb''b''pb''b' ↩
    'уb''b''гb''b''иb'')
PAUSE_AT_END        = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb''
GAS_PRESSURE        = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' (b''вb''b' ↩
    'иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''єb''b''тb''b' ↩
    'ьb''b''cb''b''яb'' b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''дb''b''лb''b''яb'' b' ↩
    'зb''b''вb'''b''яb''b''зb''b''кb''b''уb'' PowerMax)
CUT_MODE            = b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' (b''вb''b' ↩
    'иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''єb''b''тb''b' ↩
    'ьb''b''cb''b''яb'' b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''дb''b''лb''b''яb'' b' ↩
    'зb''b''вb'''b''яb''b''зb''b''кb''b''уb'' PowerMax)
```

Можна додавати новий матеріал, видаляти матеріал або редагувати існуючий матеріал у вкладці PARAMETERS. Це також можна зробити за допомогою magic comments у файлі G-коду.

Файл матеріалу можна редагувати за допомогою текстового редактора під час роботи LinuxCNC. Після збереження змін натисніть **Reload** (Перезавантажити) у розділі MATERIAL (МАТЕРІАЛ) вкладки PARAMETERS Tab (плазма:вкладка параметрів,вкладка ПАРАМЕТРИ), щоб перезавантажити файл матеріалу.

10.8.9.7 Ручне оброблення матеріалів

Для ручного управління матеріалами користувач повинен вручну вибрати матеріал зі списку матеріалів у розділі MATERIAL (МАТЕРІАЛ) на вкладці PARAMETERS Tab (плазма:вкладка параметрів,в ПАРАМЕТРИ) перед запуском програми G-коду. Крім вибору матеріалів зі списку матеріалів у розділі MATERIAL (МАТЕРІАЛ) на вкладці PARAMETERS Tab (плазма:вкладка параметрів,вкладка ПАРАМЕТРИ), користувач може використовувати MDI для зміни матеріалів за допомогою наступної команди:

```
M190 Pn
```

Наведений нижче код є мінімальним кодом, необхідним для успішного різання за допомогою методу ручного вибору матеріалу:

```
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

Note
Ручне оброблення матеріалів обмежить користувача лише одним матеріалом для всієї роботи.

10.8.9.8 Автоматичне оброблення матеріалів

Для автоматичної обробки матеріалів користувач повинен додати команди до свого G-коду, які дозволять QtPlasmaC автоматично змінювати матеріал.

Наступні коди можуть бути використані, щоб дозволити QtPlasmaC автоматично змінювати матеріали:

- **M190 P*n*** – Змінює поточний відображений матеріал на матеріал з номером *n*.
- **M66 P3 L3 Q1** – Додає невелику затримку (1 секунду в цьому прикладі), щоб очікувати, поки QtPlasmaC підтвердить успішну зміну матеріалів.
- **F#<_hal[plasmac.cut-feed-rate]>** – Встановлює швидкість подачі для різання на швидкість подачі, що відображається в розділі МАТЕРІАЛ вкладки PARAMETERS.

Для автоматичної обробки матеріалів коди ПОВИННІ застосовуватися в зазначеному порядку. Якщо завантажено програму G-коду, яка містить одну або кілька команд зміни матеріалу, то перший матеріал буде відображатися у верхньому заголовку ВІКНА ПЕРЕДПЕРЕГЛЯДУ на вкладці MAIN Tab під час завантаження програми.

Мінімальний код, необхідний для успішного різання за допомогою методу автоматичного вибору матеріалу:

```
M190 Pn
M66 P3 L3 Q1
F#<_hal[plasmac.cut-feed-rate]>
M3 $0 S1
.
.
M5 $0
```

Note
Повернення до матеріалу за замовчуванням до завершення програми можливе за допомогою коду **M190 P-1**.

10.8.9.9 Додавання матеріалів за допомогою магічних коментарів у G-коді

Використовуючи ”магічні коментарі” у файлі G-коду, можна зробити наступне:

- Додайте нові матеріали до файлу <*назва_машини*>_material.cfg.
- Відредагуйте наявні матеріали у файлі <*назва_машини*>_material.cfg.
- Використовуйте один або декілька тимчасових матеріалів.

Тимчасові матеріали нумеруються автоматично за допомогою QtPlasmaC, а зміна матеріалу також здійснюється за допомогою QtPlasmaC і не повинна додаватися до файлу G-коду за допомогою програмного забезпечення CAM або іншим способом. Номери матеріалів починаються з 1000000 і збільшуються для кожного тимчасового матеріалу. Зберегти тимчасовий матеріал неможливо, однак користувач може створити новий матеріал під час відображення тимчасового матеріалу, і він буде використовувати налаштування тимчасового матеріалу як стандартні.

Tip
Можна використовувати лише тимчасові матеріали та мати порожній файл <*назва_машини*>_material.cfg. Це усуває необхідність оновлювати файл матеріалів QtPlasmaC файлом інструменту CAM.

- Весь коментар має бути в дужках.
- Початок магічного коментаря має бути: **(o=**
- Знак рівності повинен стояти одразу після кожного параметра без пробілу.
- Обов’язкові параметри мають бути в магічному коментарі (для опції 0 **na** є необов’язковим, а **nu** не використовується).
- У файлі G-коду може бути будь-яка кількість і тип магічних коментарів.
- Якщо опцію 0 потрібно використовувати на додаток до опції 1 та/або опції 2, то всі опції 0 повинні з’являтися після всіх опцій 1 або всіх опцій 2 у файлі G-коду.

Варіанти такі:

Варіант	**Опис**
0	Створює тимчасовий матеріал за замовчуванням. Інформація про матеріал, додана за допомогою цієї опції, буде видалена при перезапуску LinuxCNC або перезавантаженні матеріалів. Вона також може бути перезаписана новим файлом G-коду, що містить тимчасові матеріали.
1	Додає новий матеріал, якщо вказаний номер не існує.
2	Перезаписує існуючий матеріал, якщо вказаний номер існує. Додає новий матеріал, якщо вказаний номер не існує.

Обов’язкові параметри:

Ім’я	**Опис**
o	Вибирає опцію, яку потрібно використовувати.
nu	Встановлює номер матеріалу (не використовується для опції 0).
na	Встановлює назву матеріалу (необов’язково для опції 0).
ph	Встановлює висоту проколювання.
pd	Встановлює затримку проколювання.

Ім’я	Опис
ch	Встановлює висоту зрізу.
fr	Встановлює швидкість подачі.

Додаткові параметри:

Ім’я	Опис
mt	Sets the material thickness.
kw	Встановлює ширину пропила.
th	Встановлює стан THC (0=вимкнено, 1=увімкнено).
ca	Встановлює значення струму зрізу.
cv	Встановлює напругу відрізу.
pe	Встановлює паузу в кінці затримки.
gp	Встановлює тиск газу (PowerMax).
cm	Встановлює режим різання (PowerMax).
jh	Встановлює висоту стрибка по калюжі.
jd	Встановлює затримку стрибка по калюжі.

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)
```

Якщо в файлі G-коду вказано тимчасовий матеріал, то рядок зміни матеріалу (M190...) і рядок очікування зміни (M66...) будуть додані фільтром G-коду і не потрібні в файлі G-коду.

10.8.9.10 Конвертер матеріалів

Ця програма використовується для конвертації існуючих таблиць інструментів у файли матеріалів QtPlasmaC. Вона також може створювати файл матеріалів з ручного введення користувачем у поля введення.

На цьому етапі доступні лише конвертації для таблиць інструментів, експортованих із SheetCam або Fusion 360.

Таблиці інструментів SheetCam готові, а конвертація повністю автоматична. Файл інструмента SheetCam має бути у форматі SheetCam .tools.

Таблиці інструментів Fusion 360 не містять усіх обов’язкових полів, тому користувачеві буде запропоновано вказати відсутні параметри. Файл інструмента Fusion 360 має бути у форматі JSON для Fusion 360.

Якщо користувач має формат з іншого програмного забезпечення CAM, який він хотів би конвертувати створіть **Нову тему** в розділі PlasmaC forum на форумі LinuxCNC forum, щоб запросити це доповнення.

Конвертер матеріалів можна запустити з терміналу одним із двох наступних методів.

Для встановлення пакета (Buildbot) введіть таку команду у вікні терміналу:

```
qtplasmac-materials
```

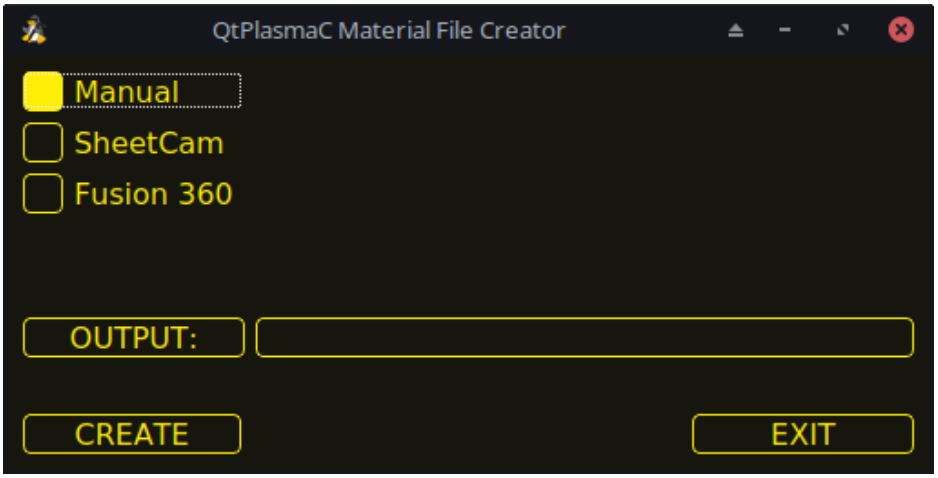
Для встановлення на місці введіть такі дві команди у вікні терміналу:

```
source ~/linuxcnc-dev/scripts/rip-environment
qtplasmac-materials
```

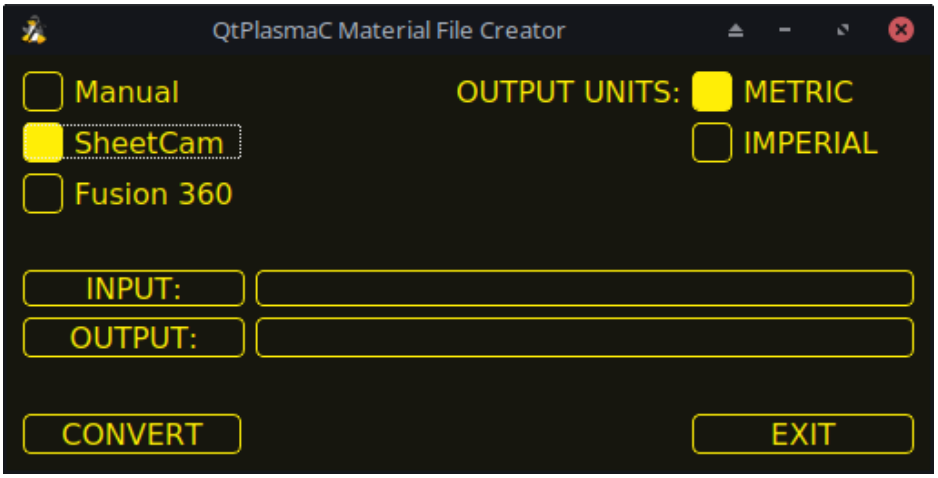
Це відкриє головне діалогове вікно конвертера матеріалів, у якому за замовчуванням вибрано параметр «Вручну».

Виберіть одне з:

- **Вручну** – для ручного створення нового файлу матеріалу.

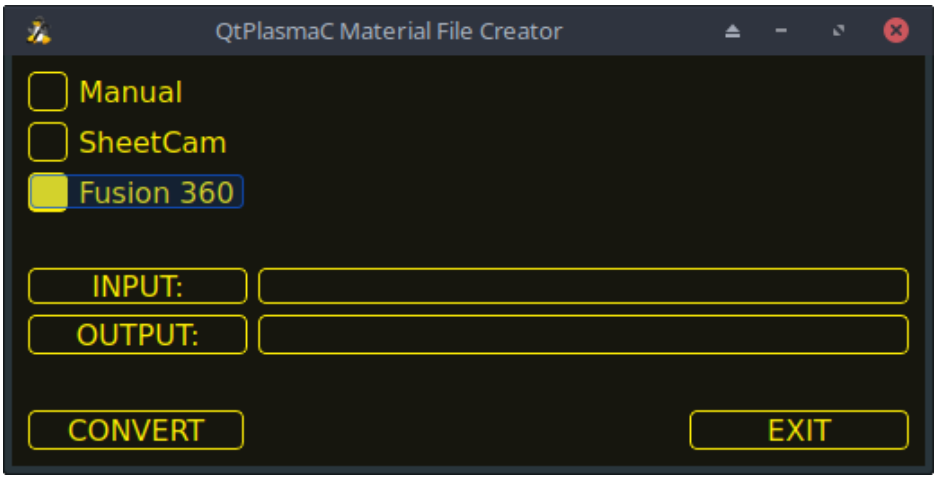


- **SheetCam** – для конвертації файлу інструмента SheetCam.



Тільки для SheetCam виберіть, чи потрібен користувачеві метричний чи імперський вихідний файл.

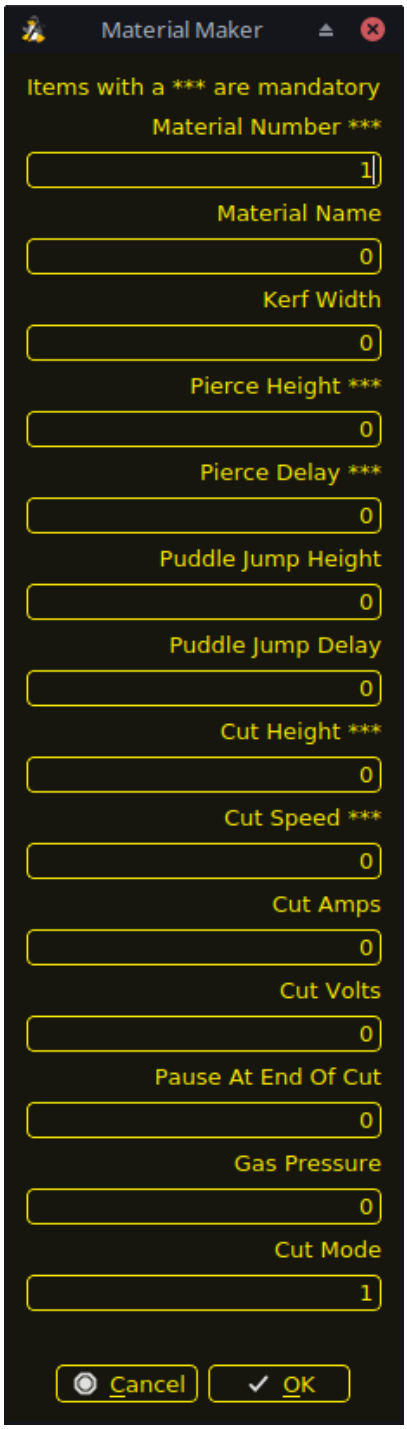
- **Fusion 360** – для конвертації файлу інструменту Fusion 360.



Щоб конвертувати:

1. Виберіть вхідний файл для конвертації, натисніть **INPUT**, щоб відкрити селектор файлів, або безпосередньо введіть файл у поле введення.
2. Виберіть вихідний файл для запису, натисніть **OUTPUT**, щоб відкрити вікно вибору файлу, або введіть файл безпосередньо в поле введення. Зазвичай це буде ~/linuxcnc/configs/<*machine_na*
   При необхідності користувач може вибрати інший файл і вручну відредагувати файл <*machine_name*>_material.cfg.
3. Натисніть кнопку **СТВОРИТИ/КОНВЕРТУВАТИ**, і новий файл матеріалу буде створено.

Як для ручного створення, так і для конвертації Fusion 360, з’явиться діалогове вікно з усіма доступними параметрами для введення. Будь-який запис, позначений як ***, є обов’язковим, а всі інші записи є необов’язковими, залежно від потреб користувача щодо конфігурації.



Note
Якщо користувач вибере ~/linuxcnc/configs/<*назва_комп'ютера*>_material.cfg, а файл вже існує, його буде перезаписано.

10.8.9.11 ЛАЗЕР

QtPlasmaC має можливість використовувати лазер для встановлення початку координат з компенсацією обертання або без неї. Компенсація обертання може використовуватися для вирівнювання робочого зміщення до листа матеріалу з краями, які не паралельні осям X/Y верстата. Кнопка LASER буде

активована після повернення верстата в початкове положення. Ця кнопка не буде видима, поки зміщення LASER не буде встановлено у файлі <*machine_name*>.prefs.

Щоб скористатися цією функцією, користувач повинен встановити зміщення лазера відносно центру пальника, виконавши процедуру, описану в Peripheral Offsets.

Щоб змінити зміщення вручну, користувач може відредагувати один або обидва наступні параметри в розділі **[LASER_OFFSET]** файлу <*machine_name*>.prefs:

```
X axis = n.n
Y axis = n.n
```

де *n.n* – відстань від центральної лінії пальника до перехрестя лазера.

Крім того, лазер можна підключити до будь-якого доступного виходу для вмикання та вимикання лазера через контакт HAL з такою назвою:

```
qtplasmac.laser_on
```

Щоб встановити початок координат з нульовим обертанням:

1. Натисніть кнопку **ЛАЗЕР**.
2. **LASER** Мітка кнопки зміниться на **MARK EDGE**, а контакт HAL з назвою qtplasmac.laser_on буде ввімкнено.
3. Рухайтеся стрілками, доки лазерне перехрестя не опиниться на вершині потрібної початкової точки.
4. Натисніть **MARK EDGE**. Напис кнопки **MARK EDGE** зміниться на **SET ORIGINAL**.
5. Натисніть **SET ORIGIN**. Напис кнопки **SET ORIGIN** зміниться на **MARK EDGE**, а контакт HAL з назвою qtplasmac.laser_on буде вимкнено.
6. Тепер пальник переміститься в положення X0 Y0.
7. Зміщення тепер успішно встановлено.

Щоб встановити початок координат з обертанням:

1. Натисніть кнопку **ЛАЗЕР**.
2. **LASER** Мітка кнопки зміниться на **MARK EDGE**, а контакт HAL з назвою qtplasmac.laser_on буде ввімкнено.
3. Рухайтеся стрілками, доки лазерне перехрестя не опиниться на краю матеріалу на відповідній відстані від потрібної точки початку.
4. Натисніть **MARK EDGE**. Напис кнопки **MARK EDGE** зміниться на **SET ORIGINAL**.
5. Рухайтеся стрілками, доки лазерне перехрестя не опиниться у початковій точці матеріалу.
6. Натисніть **SET ORIGIN**. Напис кнопки **SET ORIGIN** зміниться на **MARK EDGE**, а контакт HAL з назвою qtplasmac.laser_on буде вимкнено.
7. Тепер пальник переміститься в положення X0 Y0.
8. Зміщення тепер успішно встановлено.

Щоб вимкнути лазер і скасувати вирівнювання:

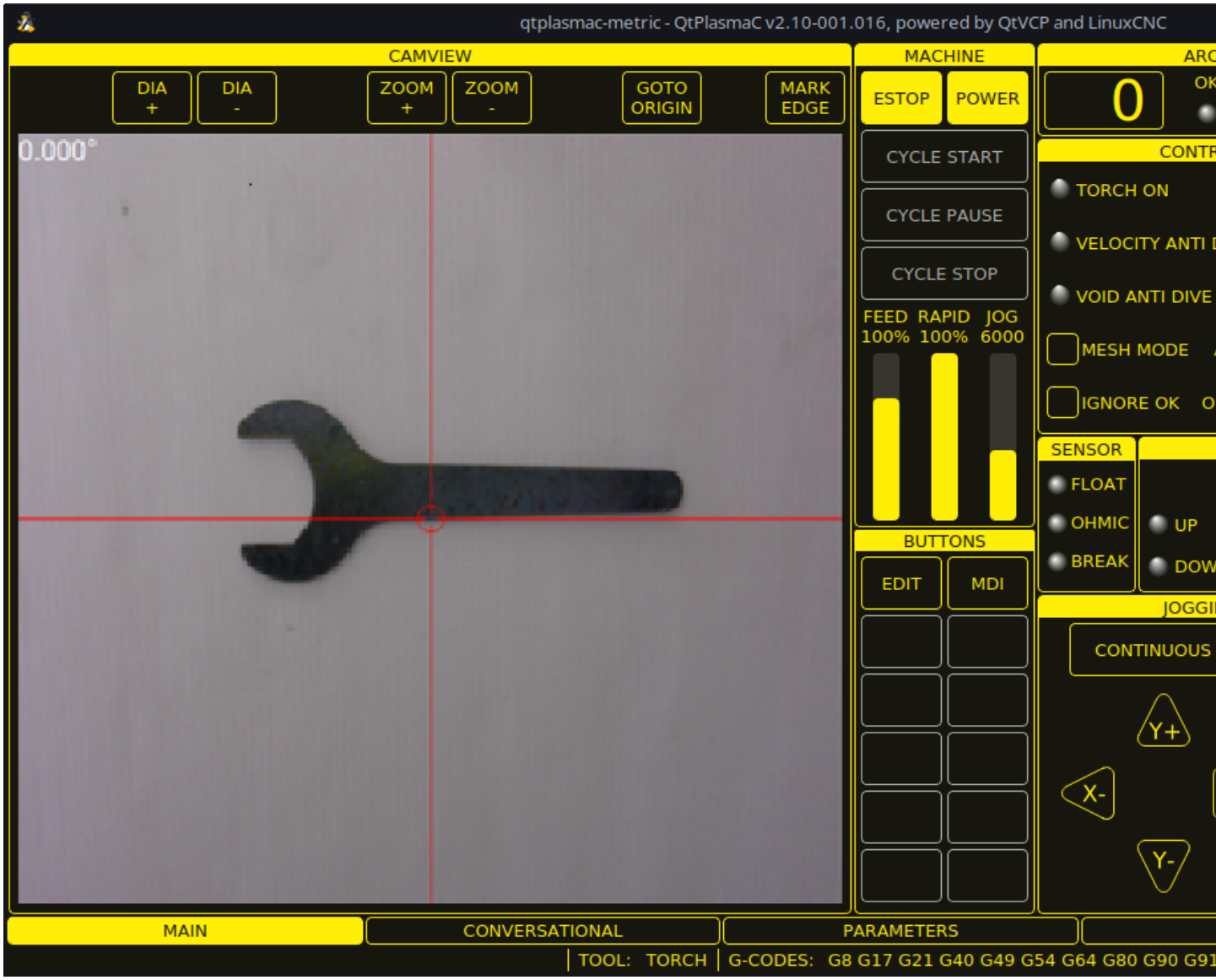
1. Натисніть кнопку **LASER** та утримуйте її довше 750 мс.

2. Напис кнопки **LASER** зміниться на **LASER**, а контакт HAL з назвою qtplasmac.laser_on буде вимкнено.
3. Відпустіть кнопку **LASER**.

Якщо налаштовано вирівнювальний лазер, то його можна використовувати під час ВІДНОВЛЕННЯ ПІСЛЯ РІЗАННЯ для точного позиціонування нових початкових координат.

Щоб виконати пробний запуск G-коду за допомогою лазера: . Переконайтеся, що немає помилок меж, і що увімкнено функцію CYCLE START. . Натисніть кнопку **LASER** та утримуйте її довше 750 мс, лазер увімкнеться та розпочнеться пробний запуск. . Відпустіть кнопку **LASER**.

10.8.9.12 КАМЕРА



QtPlasmaC має можливість використовувати USB-камеру для встановлення початку координат з компенсацією повороту або без неї. Компенсація повороту може використовуватися для вирівнювання робочого зміщення до листа матеріалу з краями, які не паралельні осям X/Y верстата. Кнопка CAMERA буде активована після повернення верстата в початкове положення. Ця кнопка не буде видима, поки зміщення CAMERA не буде встановлено у файлі <*machine_name*>.prefs.

Щоб скористатися цією функцією, користувач повинен встановити зміщення камери відносно центру факела, виконавши процедуру, описану в Peripheral Offsets.

Щоб змінити зміщення вручну, користувач може відредагувати один або обидва наступні параметри осей у розділі **[CAMERA_OFFSET]** файлу <*machine_name*>.prefs:

```
X axis = n.n
Y axis = n.n
Camera port = 0
```

де *n.n* – відстань від центральної лінії ліхтарика до перехрестя прицілу камери.

Щоб встановити початок координат з нульовим обертанням:

1. Рухайтеся по стрілках, доки перехрестя не опиниться на вершині потрібної початкової точки.
2. Натисніть **MARK EDGE**. Напис кнопки **MARK EDGE** зміниться на **SET ORIGINAL**, а кнопка **GOTO ORIGINAL** буде деактивована.
3. Натисніть **ВСТАНОВИТИ ПОЧАТОК**. Напис кнопки **ВСТАНОВИТИ ПОЧАТОК** зміниться на **ПОМІТКА КРАЙ**, а кнопка **ПЕРЕЙТИ ДО ПОЧАТКУ** стане активною.
4. Тепер пальник переміститься в положення X0 Y0.
5. Зміщення тепер успішно встановлено.

Щоб встановити початок координат з обертанням:

1. Переміщуйтесь, доки перехрестя не опиняться на краю матеріалу на відповідній відстані від потрібної початкової точки.
2. Натисніть **MARK EDGE**. Напис кнопки **MARK EDGE** зміниться на **SET ORIGINAL**, а кнопка **GOTO ORIGINAL** буде деактивована.
3. Рухайтеся стрілками, доки перехрестя не опиниться у початковій точці матеріалу.
4. Натисніть **ВСТАНОВИТИ ПОЧАТОК**. Напис кнопки **ВСТАНОВИТИ ПОЧАТОК** зміниться на **ПОМІТКА КРАЙ**, а кнопка **ПЕРЕЙТИ ДО ПОЧАТКУ** стане активною.
5. Тепер пальник переміститься в положення X0 Y0.
6. Зміщення тепер успішно встановлено.

На панелі CAMVIEW миша може впливати на перехрестя прицілу та рівень масштабування наступним чином:

- Прокручування коліщатка миші - Зміна діаметра перехрестя.
- Подвійне клацання кнопкою коліщатка миші – відновлює діаметр перехрестя до значення за замовчуванням.
- Ліва кнопка миші + прокручування коліщатком - змінює рівень масштабування камери.
- Mouse Left Button Clicked + Wheel Button Double Click - Відновлює рівень масштабування камери за замовчуванням.

10.8.9.13 Допуск шляху

Допуск траєкторії встановлюється за допомогою команди G64 та наступного значення P. Значення P відповідає величині, на яку фактична траєкторія різання, якою рухається верстат, може відхилятися від запрограмованої траєкторії різання.

Допуск траєкторії LinuxCNC за замовчуванням встановлений на максимальну швидкість, що призведе до сильного закруглення кутів при використанні зі звичайними швидкостями плазмового різання.

Рекомендується встановлювати допуск траєкторії, розміщуючи відповідну команду G64 та значення P у заголовку кожного файлу G-коду.

Надана програма фільтрації G-коду перевірить наявність команди `G64 P__n__` перед першою командою руху. Якщо команда G64 не знайдена, вона вставиться командою `G64 P0.1`, яка встановлює допуск траєкторії на 0,1 мм. Для імперської конфігурації команда буде `G64 P0.004`.

Для метрики:

```
G64 P0.1
```

Для Імперіал:

```
G64 P0.004
```

10.8.9.14 Призупинений рух

QtPlasmaC має можливість дозволяти змінювати положення осей X та Y вздовж поточного шляху різання, поки програма G-коду призупинена, що сприяє Cut Recovery.

Щоб скористатися цією функцією, необхідно ввімкнути адаптивне керування подачею (M52) LinuxCNC (P1). Це також є вимогою для Hole Cutting Velocity Reduction.

Щоб увімкнути **Пауза руху**, преамбула G-коду повинна містити наступний рядок:

```
M52 P1
```

Щоб вимкнути **Пауза руху** у будь-який момент, скористайтеся такою командою:

```
M52 P0
```

10.8.9.15 Пауза в кінці монтажу

Ця функція може бути використана для того, щоб дуга «наздогнала» положення пальника і повністю завершила різання. Зазвичай це потрібно для більш товстих матеріалів і особливо корисно при різанні нержавіючої сталі.

Використання цієї функції призведе до зупинки всіх рухів в кінці різання, поки пальник все ще увімкнений. Після закінчення часу витримки (в секундах), встановленого параметром **Pause At End** (Пауза в кінці) в розділі MATERIAL (МАТЕРІАЛ) вкладки PARAMETERS Tab (плазма:вкладка параметрів,вкладка ПАРАМЕТРИ), QtPlasmaC продовжить виконання команди M5, щоб вимкнути і підняти пальник.

10.8.9.16 Кілька інструментів

QtPlasmaC має можливість використовувати більше одного типу плазмового інструменту, використовуй
шпинделі LinuxCNC як плазмовий інструмент під час запуску програми G-коду.

Допустимі плазмові інструменти для використання:

Ім'я	ІНСТРУМЕНТ #	Опис
Плазмовий пальник	0	Використовується для звичайного плазмового різання.
Писар	1	Використовується для гравіювання матеріалів.
Плазмовий пальник	2	Використовується для плямистості (створення заглиблень для полегшення свердління).

Номер шпинделя LinuxCNC (позначений як $*n*) повинен бути присутнім у початковій команді, а також у команді завершення, щоб мати змогу запускати та зупиняти правильний плазмовий інструмент. Приклади:

- `M3 $0 S1` вибере та запустить інструмент плазмового різання.
- `M3 $1 S1` вибере та запустить писаря.
- `M3 $2 S1` вибере та запустить інструмент для визначення плазми.
- `M5 $0` зупинить інструмент плазмового різання.
- `M5 $1` зупинить писаря.
- `M5 $2` зупинить інструмент для виявлення плазми.

Дозволяється використовувати **M5 $-1** замість кодів M5 $*n*, наведених вище, для зупинки всіх інструментів.

Щоб використовувати рильце, користувач повинен додати зміщення осей X і Y до таблиці інструментів LinuxCNC. Інструмент 0 призначається для плазмового пальника, а інструмент 1 — для рильця. Інструменти вибираються за допомогою команди **T*n* M6**, а потім для застосування зміщень для вибраного інструменту потрібна команда **G43 H0**. Важливо зазначити, що таблиця інструментів LinuxCNC і команди інструментів застосовуються тільки в тому випадку, якщо користувач використову
scribe на додаток до плазмового пальника. Для отримання додаткової інформації див. scribe.

10.8.9.17 Зменшення швидкості

Існує контакт HAL з назвою **motion.analog-out-03**, який можна змінити в G-коді за допомогою команд **M67 (синхронізовано з рухом)/M68 (негайно)**. Цей контакт зменшить швидкість початкової подачі до відсотка, зазначеного в команді.

Мітка «VEL:» у верхньому правому куті вікна попереднього перегляду оновлюється, відображаючи відсоток від початкової швидкості подачі, що використовується. Наприклад, «VEL@20%:» означає, що стіл ріже зі швидкістю, яка становить 20% від запрограмованої швидкості подачі, тобто зі скороченням на 80%.

Note
Через відмінності між інтервалами опитування графічного інтерфейсу користувача та компонента PlasmaC, оновлення міток швидкості можуть затримуватися (зазвичай до 100 мс).

Важливо ретельно розуміти різницю між **Синхронізовано з рухом** та **Негайно**:

- `M67` (Синхронізовано з рухом) — Фактична зміна зазначеного виходу (наприклад, P2 (THC)) відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходу не відбудуться. Найкраще програмувати код руху (наприклад, G0 або G1) одразу після M67.
- `M68` (Негайне) — ці команди виконуються негайно після їх отримання контролером руху. Оскільки вони не синхронізовані з рухом, вони порушують злиття. Це означає, що якщо ці коди використовуют в середині активних кодів руху, рух призупиняється для активації цих команд.

Приклади:

- `M67 E3 Q0` встановило б швидкість на 100% від **CutFeedRate**.
- `M67 E3 Q40` встановив би швидкість на 40% від **CutFeedRate**.
- `M67 E3 Q60` встановив би швидкість на 60% від **CutFeedRate**.
- `M67 E3 Q100` встановило б швидкість на 100% від **CutFeedRate**.

Значення Q, які менші або дорівнюють 0 або більші або дорівнюють 100, будуть встановлені на 100.

Якщо користувач має намір використовувати цю функцію, було б доцільно додати `M68 E3 Q0` як до преамбули, так і до постамбули програми G-коду, щоб машина запускалася та завершувалася у відомому стані.



Important
G-КОД THC ТА **VELOCITY BASED THC** НЕ МОЖУТЬ ВИКОРИСТОВУВАТИСЯ, ЯКЩО ДІЄ **КОМПЕНСАЦІЯ РІЖКА**; ВІДОБРАЖАЄТЬСЯ ПОВІДОМЛЕННЯ ПРО ПОМИЛКУ.



Warning
Якщо параметр Cut Feed Rate (Швидкість подачі різання) у розділі MATERIAL (МАТЕРІАЛ) вкладки PARAMETERS Tab (Плазма: вкладка параметрів) встановлено на нуль, то QtPlasmaC використовуватиме **motion.requested-velocity** (швидкість, задану стандартним викликом швидкості подачі в G-коді) для розрахунків THC. Це не рекомендується, оскільки це не є надійним способом реалізації THC на основі швидкості.

Note
Усі посилання на CutFeedRate стосуються значення **Cut Feed Rate**, яке відображається в розділі MATERIAL вкладки PARAMETERS.

10.8.9.18 THC (Контролер висоти пальника)

THC можна ввімкнути або вимкнути з фрейму THC вкладки MAIN.

THC також можна вмикати або вимикати безпосередньо з програми G-коду.

THC не активується, поки швидкість не досягне 99,9% від **CutFeedRate**, а потім не закінчиться час **Delay** THC, якщо він вказаний у розділі THC на вкладці PARAMETERS Tab. Це необхідно для стабілізації напруги дуги.

QtPlasmaC використовує керуючу напругу, яка залежить від стану прапорця **AUTO VOLTS** на вкладці MAIN Tab:

1. Якщо позначено опцію **Використовувати автоматичне налаштування напруги**, фактична напруга різання вимірюється в кінці часу **Затримки** THC, і це значення використовується як цільова напруга для регулювання висоти пальника.
2. Якщо опція «Використовувати автоматичне напруження» не відмічена, то напруження, яке відображається як «Напруження відсічення» у розділі «МАТЕРІАЛ» на вкладці «Вкладка ПАРАМЕТРИ», використовується як цільове напруження для регулювання висоти пальника.

G-код THC THC можна вимкнути та увімкнути безпосередньо з G-коду, за умови, що THC не вимкнено в розділі THC вкладки MAIN Tab, шляхом встановлення або скидання контакту **motion.digita out-02** за допомогою M-кодів M62-M65:

- `M62 P2` вимкне THC (синхронізацію з рухом)
- `M63 P2` увімкне THC (синхронізацію з рухом)
- `M64 P2` вимкне THC (негайно)
- `M65 P2` увімкне THC (негайно)

Важливо ретельно розуміти різницю між **Синхронізовано з рухом** та **Негайно**:

- `M62` та `M63` (синхронізовано з рухом) — фактична зміна зазначеного виходу (наприклад, P2 (THC)) відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходу не відбудуться. Найкраще програмувати код руху (наприклад, G0 або G1) одразу після M62 або M63.
- `M64` та `M65` (Негайні) — ці команди виконуються негайно після їх отримання контролером руху. Оскільки вони не синхронізовані з рухом, вони порушують злиття. Це означає, що якщо ці коди використовуються посеред активних кодів руху, рух призупиняється для активації цих команд.

На основі швидкості THC

Якщо швидкість різання падає нижче відсотка **CutFeedRate** (як визначено значенням VAD Threshold % у кадрі THC розділу CONFIGURATION на вкладці PARAMETERS Tab), THC буде заблоковано, доки швидкість різання не повернеться до рівня щонайменше 99,9% від **CutFeedRate**. Це буде відображено індикатором **VELOCITY ANTI DIVE**, що світиться в панель управління на головна вкладка.

Система гармонійного склеювання (THC) на основі швидкості запобігає зміні висоти пальника при зменшенні швидкості для гострого кута або невеликого отвору.

Важливо зазначити, що Зменшення-швидкості впливає на ТГК на основі швидкості таким чином:

1. Якщо зменшення швидкості (Velocity Reduction) буде задіяно посеред різу (cut-to-case), THC буде заблоковано.
2. Регулятор різання THC залишатиметься заблокованим, доки зменшення швидкості не буде скасовано шляхом повернення її до значення, що перевищує **порогове значення VAD**, і пальник фактично не досягне 99,9% від **швидкості подачі різання**.

10.8.9.19 Компенсація різця

LinuxCNC (QtPlasmaC) має можливість автоматично коригувати траєкторію різання поточної програми на величину, вказану в параметрі «Ширина пропилу» (Kerf Width) параметрів різання вибраного матеріалу. Це корисно, якщо G-код запрограмований на номінальну траєкторію різання, а користувач буде запускати програму на матеріалах різної товщини, щоб забезпечити однаковий розмір деталей.

Щоб використовувати компенсацію різця, користувачеві потрібно буде використовувати G41.1, G42.1 та G40 зі штифтом HAL ширини пропила:

- `G41.1 D#<_hal[plasmac.kerf-width]>` : зміщує пальник ліворуч від запрограмованого шляху
- `G42.1 D#<_hal[plasmac.kerf-width]>` : зміщує пальник праворуч від запрограмованого шляху
- `G40` вимикає компенсацію різця



Important
ЯКЩО ДІЄ **КОМПЕНСАЦІЯ РІЖКА G-КОД THC**, **VELOCITY BASED THC** ТА **OVER CUT** НЕ МОЖУТЬ ВИКОРИСТОВУВАТИСЯ; ВІДОБРАЖАЄТЬСЯ ПОВІДОМЛЕННЯ ПРО ПОМИЛКУ.

10.8.9.20 Початкове відчуття висоти (IHS) Пропустити

Початкове визначення висоти можна пропустити одним із двох різних способів:

1. Якщо THC вимкнено або THC увімкнено, але не активно, то пропуск IHS відбудеться, якщо початок різу знаходиться менш ніж на відстань **Пропустити IHS** від останнього успішного зондування.
2. Якщо THC увімкнено та активно, то пропуск IHS відбудеться, якщо початок різу знаходиться на відстані менше ніж **Skip IHS** від кінця останнього різу.

Нульове значення для **Пропустити IHS** вимкне всі пропуски IHS.

Будь-які помилки, що виникли під час розрізу, вимкнуть пропуск IHS для наступного розрізу, якщо ввімкнено опцію **Пропустити IHS**.

10.8.9.21 Зондування

Зондування може здійснюватися за допомогою омічного датчика або поплавкового вимикача. Також можна поєднати ці два методи, і в цьому випадку поплавковий вимикач буде забезпечувати резервне омічне зондування. Альтернативою омічному зондуванню є Зондування з зміщенням

Якщо пальник верстата не підтримує омічне зондування, користувач може мати окремий зонд поруч із пальником. У цьому випадку користувач повинен висунути зонд нижче пальника. Зонд НЕ повинен виходити за межі мінімальної висоти різання нижче пальника, а відстань зміщення осі Z потрібно ввести як **Омічне зміщення** у полі PROBING (Зондування) розділу CONFIGURATION (Конфігурація) вкладки PARAMETERS Tab.

Налаштування зондування виконується у фреймі PROBING розділу CONFIGURATION вкладки PARAMETERS.

QtPlasmaC може проводити зондування з повною швидкістю по осі Z, якщо машина має достатній хід поплавкового вимикача для поглинання будь-якого перевищення. Якщо хід поплавкового вимикача машини є відповідним, користувач може встановити висоту зондування близько до MINIMUM_LIMIT осі Z і проводити все зондування на повній швидкості.

Деякі поплавкові вимикачі можуть демонструвати великий гістерезис перемикання, який проявляється в послідовності зондування як надмірний час для завершення останнього зондування.

- Цей час можна зменшити, збільшивши швидкість останнього зонда.
- Ця швидкість за замовчуванням становить 0,001 мм (0,000039 дюйма) за цикл серводвигуна.
- Цю швидкість можна збільшити до 10 разів, додавши наступний рядок до файлу custom.hal:

```
setp plasmac.probe-final-speed n
```

де *n* — це значення від 1 до 10. Рекомендується підтримувати це значення якомога нижчим.

Використання цієї функції дещо змінить кінцеву висоту та вимагатиме ретельного тестування зонда для її підтвердження.

Це значення швидкості впливає на BCI вимірювання, тому якщо користувач використовує омічне вимірювання і змінює це значення швидкості, йому потрібно буде провести тест, щоб встановити необхідне зміщення для компенсації цієї зміни швидкості, а також переміщення поплавка.

Надійність цієї функції буде настільки ж хорошою, як і повторюваність роботи поплавкового вимикача.

Note
Висота зонда стосується висоти над віссю Z MINIMUM_LIMIT.

10.8.9.22 Зі зміщенням зондування

Зондування з зміщенням — це використання зонда, який зміщений від пальника. Цей метод є альтернативою омічному зондуванню і використовує вихідний контакт `plasmac.ohmic-enable` для управління соленоїдом, що висуває та втягує зонд. Вхідний контакт `plasmac.ohmic-probe` використовується для виявлення матеріалу, а **Ohmic Offset** у кадрі PROBING розділу CONFIGURATION вкладки PARAMETERS Tab використовується для встановлення правильної виміряної висоти.

Зонд може бути механічним, постійно встановленим датчиком наближення або навіть просто жорстким шматком дроту, що виступає приблизно на 0,5 мм (0,2 дюйма) нижче кінчика пальника. Якщо зонд є механічним, він повинен швидко висуватися/втягуватися, щоб уникнути надмірного часу зондування, і зазвичай працює на пневматичній основі.

Щоб скористатися цією функцією, користувач повинен встановити зміщення зонда відносно центру пальника, виконавши процедуру, описану в Peripheral Offsets.

Щоб змінити зміщення вручну, користувач може відредагувати один або обидва наступні параметри в розділі **[OFFSET_PROBING]** файлу <*machine_name*>.prefs:

```
X axis = n.n
Y axis = n.n
Delay = t.t
```

де *n.n* – це зміщення зонда від центру пальника в одиницях вимірювання для осей X та Y, а *t.t* – це час у секундах, необхідний для механічного розгортання зонда, якщо це необхідно.

Кожен із цих параметрів є необов'язковим і також може відображатися в будь-якому порядку. Якщо параметр не виявлено, значення за замовчуванням дорівнює 0.0. Після X або Z не може бути пробілу, допускаються малі літери.

Коли ця змінна з'являється у файлі <*machine_name*>.prefs з X або Y, що не дорівнюють нулю, QtPlasmaC виконуватиме **всі** омічні зондування як зондування з зміщенням. Якщо Offset Probing є дійсним, швидкість подачі, з якою осі X і Y переміщуються до позиції зміщення, можна регулювати за допомогою параметра **Offset Speed** у рамці PROBING на вкладці PARAMETERS Tab.

Коли послідовність зондування розпочалася, контакт `plasmac.ohmic-enable` буде встановлений у стан True, що призведе до висунення зонда. Коли матеріал буде виявлено, контакт `plasmac.ohmic-en` буде скинуто у стан false, що призведе до втягування зонда.

Зонд почне рухатися до положення зміщення одночасно з осі Z, що рухається вниз до висоти зонда, зондування не почнеться, поки не закінчиться таймер розгортання. Необхідно, щоб **Висота зонда** в рамці PROBING розділу CONFIGURATION вкладки PARAMETERS Tab була вище верхньої частини матеріалу, щоб забезпечити повне зміщення зонда в правильне положення X/Y перед остаточним вертикальним рухом зонда вниз.



Important
Для зміщеного зондування висоту зонда потрібно встановити вище за верхню частину матеріалу.

10.8.9.23 Типи розрізів

QtPlasmaC дозволяє використовувати два різні режими різання:

1. **NORMAL CUT** - запускає завантажену програму G-коду для проколювання, а потім різання.
2. ТІЛЬКИ ПРОКОЛЮВАТИ - проколює матеріал лише в кожній початковій позиції різання, корисно перед **ЗВИЧАЙНИМ РІЗАННЯМ** на thick materials

Існує два способи активації цієї функції:

1. Використовуйте кнопку за замовчуванням custom user button для перемикання між типами різання.
2. Додавання наступного рядка до G-коду програми перед першим різанням, щоб увімкнути режим **Тільки прорізання** для поточного файлу:

```
#<pierce-only> = 1
```

Якщо використовується користувацька кнопка, QtPlasmaC автоматично перезавантажить файл після перемикання типу вирізання.

10.8.9.24 Різання отворів - вступ

Рекомендується, щоб діаметр отворів, що вирізаються, був не менше ніж у півтора рази більший за товщину матеріалу, що ріжеться.

Також рекомендується, щоб отвори діаметром менше 32 мм (1,26 дюйма) вирізалися зі швидкістю подачі 60% від швидкості, що використовується для профільного різання. Це також повинно блокувати викривлення контуру різання через обмеження швидкості.

QtPlasmaC може використовувати команди G-коду, які зазвичай встановлюються CAM-постпроцесором (PP), для допомоги у вирізанні отворів. Якщо користувач не має PP або його PP не підтримує ці методи, QtPlasmaC може автоматично адаптувати G-код відповідно до потреб. Цей автоматичний режим за замовчуванням вимкнений.

Існує три методи покращення якості малих отворів:

1. **Зменшення швидкості** - Зменшення швидкості приблизно до 60% від **Швидкості подачі**.
2. **Затримка дуги (Pause At End)** – Утримування пальника увімкненим на короткий час в кінці отвору під час зупинки руху, щоб дуга встигла наздогнати згасання.
3. **Over cut** - Вимкніть ліхтарик в кінці отвору, а потім продовжуйте рухатися стежкою.

Note
Якщо одночасно активні обидва режими: **Затримка дуги** та **Оверструм**, то пріоритет матиме **Оверструм**.



Important
ФУНКЦІЮ **ПЕРЕРІЗАННЯ** НЕ МОЖНА ВИКОРИСТОВУВАТИ, ЯКЩО ДІЄ КОМПЕНСАЦІЯ РІЗАКА; ВІДОБРАЖАЄТЬСЯ ПОВІДОМЛЕННЯ ПРО ПОМИЛКУ.

10.8.9.25 Різання отворів

Команди G-коду можна налаштувати або за допомогою постпроцесора CAM (PP), або вручну.

Зменшення швидкості різання отворів

Якщо для різання отвору потрібна знижена швидкість, користувач може встановити швидкість за допомогою такої команди: `M67 E3 Qnn`, де `nn` — це відсоток бажаної швидкості. Наприклад, команда `M67 E3 Q60` встановить швидкість на рівні 60 % від поточної швидкості різання матеріалу **CutFeedRate**.

Щоб скористатися цією функцією, необхідно ввімкнути адаптивне керування подачею (M52) LinuxCNC (P1). Це також є вимогою для Paused Motion під час Cut Recovery.

Щоб увімкнути **Зменшення швидкості різання отворів**, преамбула G-коду повинна містити наступний рядок:

```
M52 P1
```

Щоб вимкнути **Зменшення швидкості різання отворів** у будь-який момент, скористайтеся такою командою:

```
M52 P0
```

Див. розділ ТГК на основі швидкості.

Приклад коду для різання отвору зі зменшеною швидкістю.

```
G21 (b''мb''b''eb''b''тb''b''pb''b''иb''b''чb''b''нb''b''ab'')
G64 P0.005
M52 P1 (b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''ab''b''дb''b' ↩
    'ab''b''пb''b''тb''b''иb''b''вb''b''нb''b''yb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'yb'')
F#<_hal[plasmac.cut-feed-rate]> (b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b' ↩
    'ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b''зb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'мb''b''eb''b''тb''b''pb''b''ib''b''вb'' b''pb''b''ib''b''зb''b''ab''b''нb''b''нb''b' ↩
    'яb'')
G0 X10 Y10
M3 $0 S1 (b''пb''b''ob''b''чb''b''ab''b''тb''b''иb'' b''pb''b''ib''b''зb''b''ab''b''нb''b' ↩
    'нb''b''яb'')
G1 X0
M67 E3 Q60 (b''зb''b''мb''b''eb''b''нb''b''шb''b''иb''b''тb''b''иb'' b''шb''b''вb''b''иb''b ↩
    ''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b' ↩
    'дb''b''ob'' 60%)
G3 I10 (b''ob''b''тb''b''вb''b''ib''b''pb'')
M67 E3 Q0 (b''вb''b''ib''b''дb''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'ib'' b''дb''b''ob'' 100%)
M5 $0 (b''зb''b''ab''b''кb''b''ib''b''нb''b''чb''b''иb''b''тb''b''иb'' b''pb''b''ib''b' ↩
    'зb''b''ab''b''нb''b''нb''b''яb'')
G0 X0 Y0
M2 (b''зb''b''ab''b''кb''b''ib''b''нb''b''чb''b''иb''b''тb''b''иb'' b''pb''b''ob''b''бb''b' ↩
    'ob''b''тb''b''yb'')
```

Затримка дуги (пауза в кінці) Цей метод можна викликати, встановивши параметр Pause At End у фреймі MATERIAL вкладки PARAMETERS.

Зріз

Пальник можна вимкнути в кінці отвору, встановивши контакт `motion.digital-out-03` за допомогою M-кодів `M62` (синхронізовано з рухом)* або `M64` (негайно). Після вимкнення пальника необхідно дозволити його повторне ввімкнення перед початком наступного різання, скинувши контакт `motion. digital-out-03` за допомогою M-кодів `M63` або `M65`. Це буде зроблено автоматично парсером G-коду QtPlasmaC, якщо він досягне команди M5, не побачивши `M63 P3` або `M65 P3`.

Після вимкнення пальника траєкторія отвору буде проходити на довжині за замовчуванням 4 мм (0,157 дюйма). Цю відстань можна вказати, додавши `#<oclength> = n` до файлу G-коду.

- `M62 P3` вимкне ліхтарик (синхронізовано з рухом)
- `M63 P3` дозволить увімкнути ліхтарик (синхронізовано з рухом)
- `M64 P3` вимкне ліхтарик (негайно)
- `M65 P3` дозволить увімкнути ліхтарик (негайно)

Важливо ретельно розуміти різницю між **Синхронізовано з рухом** та **Негайно**:

- `M62 та M63` (синхронізовано з рухом) — фактична зміна зазначеного виходу (наприклад, P2 (THC)) відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходу не відбудуться. Найкраще програмувати код руху (наприклад, G0 або G1) одразу після M62 або M63.
- `M64 та M65` (Негайні) — ці команди виконуються негайно після їх отримання контролером руху. Оскільки вони не синхронізовані з рухом, вони порушують злиття. Це означає, що якщо ці коди використовуються посеред активних кодів руху, рух призупиняється для активації цих команд.

Зразок коду:

```
G21 (b''мb''b''eb''b''тb''b''pb''b''иb''b''чb''b''нb''b''ab'')
G64 P0.005
M52 P1 (b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''ab''b''дb''b' ↩
    'ab''b''пb''b''тb''b''иb''b''вb''b''нb''b''yb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'yb'')
F#<_hal[plasmac.cut-feed-rate]> (b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b' ↩
    'ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b''зb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'мb''b''eb''b''тb''b''pb''b''ib''b''вb'' b''pb''b''ib''b''зb''b''ab''b''нb''b''нb''b' ↩
    'яb'')
G0 X10 Y10
M3 $0 S1 (b''пb''b''ob''b''чb''b''ab''b''тb''b''иb'' b''pb''b''ib''b''зb''b''ab''b''нb''b' ↩
    'нb''b''яb'')
G1 X0
M67 E3 Q60 (b''зb''b''мb''b''eb''b''нb''b''шb''b''иb''b''тb''b''иb'' b''шb''b''вb''b''иb''b ↩
    ''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b' ↩
    'дb''b''ob'' 60%)
G3 I10 (b''ob''b''тb''b''вb''b''ib''b''pb'')
M62 P3 (b''вb''b''иb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''пb''b''ab''b''лb''b' ↩
    'ьb''b''нb''b''иb''b''кb'')
G3 X0.8 Y6.081 I10 (b''пb''b''pb''b''ob''b''дb''b''ob''b''вb''b''жb''b''иb''b''тb''b''иb''  ↩
    b''pb''b''yb''b''xb'' b''нb''b''ab'' 4 b''мb''b''мb'')
M63 P3 (b''дb''b''ob''b''зb''b''вb''b''ob''b''лb''b''иb''b''тb''b''иb'' b''вb''b''вb''b' ↩
    'ib''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''пb''b''ab''b''лb''b''ьb''b''нb''b' ↩
    'иb''b''кb'')
M67 E3 Q0 (b''вb''b''ib''b''дb''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'ib'' b''дb''b''ob'' 100%)
M5 $0 (b''зb''b''ab''b''кb''b''ib''b''нb''b''чb''b''иb''b''тb''b''иb'' b''pb''b''ib''b' ↩
    'зb''b''ab''b''нb''b''нb''b''яb'')
G0 X0 Y0
M2 (b''зb''b''ab''b''кb''b''ib''b''нb''b''чb''b''иb''b''тb''b''иb'' b''pb''b''ob''b''бb''b' ↩
    'ob''b''тb''b''yb'')
```

10.8.9.26 Різання отворів - автоматичне

QtPlasmaC має можливість автоматично змінювати G-код для зменшення швидкості та/або застосуванн **Over cut**, що може бути корисним під час різання отворів.

Для коректного визначення отвору необхідно, щоб усі значення в рядку G-коду G2 або G3 були явними. Якщо будь-які значення будуть обчислені математично, відобразиться діалогове повідомленн про помилку.

Визначення отворів у QtPlasmaC за замовчуванням вимкнено. Його можна ввімкнути/вимкнути за допомогою наступних параметрів G-коду для вибору потрібного режиму визначення отворів:

- `#<holes> = 0` - Змушує QtPlasmaC вимикати розпізнавання отворів, якщо воно було раніше ввімкнено.
- `#<holes> = 1` - Змушує QtPlasmaC зменшувати швидкість обробки отворів діаметром менше 32 мм (1,26 дюйма) до 60% від **CutFeedRate**.
- `#<holes> = 2` - Змушує QtPlasmaC Overcut обробляти отвір на додаток до змін швидкості в налаштуванні 1.
- `#<holes> = 3` - Змушує QtPlasmaC зменшувати швидкість обробки отворів діаметром менше 32 мм (1,26 дюйма) та дуг діаметром менше 16 мм (0,63 дюйма) до 60% від **CutFeedRate**.
- `#<holes> = 4` - Змушує QtPlasmaC Overcut отвір на додаток до зміни швидкості в налаштуванні 3.

Розмір отвору за замовчуванням для вимірювання отворів QtPlasmaC становить 32 мм (1,26 дюйма). Це значення можна змінити за допомогою наступної команди у файлі G-коду:

- `#<h_diameter>` = *nn* - Щоб встановити діаметр (*nn*) у тій самій системі одиниць, що й решта файлу G-коду.

Швидкість за замовчуванням для малих отворів QtPlasmaC становить 60% від поточної швидкості подачі. Це значення можна змінити за допомогою наступної команди у файлі G-коду:

- `#<h_velocity>` = *nn* - щоб встановити відсоток (*nn*) від необхідної поточної швидкості подачі.

Зріз Якщо активні режими зондування отвору 2 або 4, QtPlasmaC додатково до змін швидкості, пов’язаних з режимами 1 та 3, перерозріже отвір.

Стандартна довжина різання для вимірювання отвору QtPlasmaC становить 4 мм (0,157 дюйма). Це значення можна змінити за допомогою наступної команди у файлі G-коду:

- `#<oclength>` = *nn* для визначення довжини перерізу (*nn*) у тій самій системі одиниць, що й решта файлу G-коду.

Затримка дуги (пауза в кінці) Цю функцію можна використовувати на додаток до налаштування бажаного режиму виявлення отворів за допомогою відповідного параметра G-коду, встановивши параметр Pause At End у рамці MATERIAL на вкладці PARAMETERS Tab.

Зразок коду:

```
G21 (b''мb''b''eb''b''тb''b''pb''b''иb''b''чb''b''нb''b''иb''b''йb'')
G64 P0.005
M52 P1 (b''yb''b''вb''b''іb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''ab''b''дb''b' ↩
    'ab''b''пb''b''тb''b''иb''b''вb''b''нb''b''yb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'yb'')
F#<_hal[plasmac.cut-feed-rate]> (b''шb''b''вb''b''иb''b''дb''b''кb''b''іb''b''cb''b''тb''b' ↩
    'ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''іb'' b''зb'' b''пb''b''ab''b''pb''b''ab''b' ↩
    'мb''b''eb''b''тb''b''pb''b''іb''b''вb'' b''pb''b''іb''b''зb''b''ab''b''нb''b''нb''b' ↩
    'яb'')
```

```
#<holes> = 2 (b''нв''b''ab''b''дв''b''pb''b''ib''b''зb'' b''дв''b''лb''b''яв'' b''ob''b' ↩
    'тb''b''вв''b''ob''b''pb''b''ib''b''вв'')
#<oclength> = 6,5 (b''ob''b''пb''b''цb''b''ib''b''ob''b''нв''b''ab''b''лb''b''ьb''b''нв''b' ↩
    'ob'', b''дв''b''ob''b''вв''b''жb''b''ив''b''нв''b''ab'' b''нв''b''ab''b''дв''b''pb''b' ↩
    'ib''b''зb''b''уb'' 6,5 b''мb''b''мb'')
G0 X10 Y10
M3 $0 S1 (b''пb''b''ob''b''чb''b''ab''b''тb''b''ob''b''кb'' b''pb''b''ib''b''зb''b''ab''b' ↩
    'нв''b''нв''b''яв'')
G1 X0
G3 I10 (b''ob''b''тb''b''вв''b''ib''b''pb'')
M5 $0 (b''зb''b''ab''b''кb''b''ib''b''нв''b''чb''b''eb''b''нв''b''нв''b''яв'' b''pb''b' ↩
    'ib''b''зb''b''ab''b''нв''b''нв''b''яв'')
G0 X0 Y0
M2 (b''зb''b''ab''b''кb''b''ib''b''нв''b''чb''b''eb''b''нв''b''нв''b''яв'' b''pb''b''ob''b' ↩
    'бb''b''ob''b''тb''b''ив'')
```

Note
Допустимо мати кілька змішаних команд для отворів в одному файлі G-коду.

10.8.9.27 Одинарний розріз

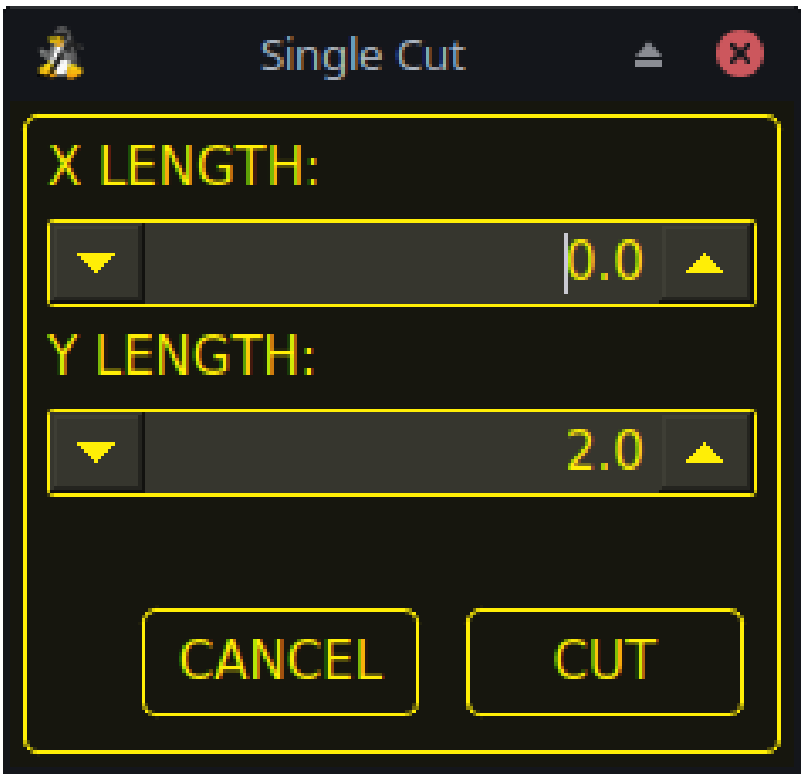
Одинарний розріз — це один односпрямований рух різання, який часто використовується для розрізання листа на менші шматки перед запуском програми G-коду.

Перед початком окремого різання необхідно встановити машину в початкове положення.

Одинарний різ розпочнеться з поточного положення верстата по осях X/Y.

Автоматичний одинарний розріз

Це найкращий метод. Параметри для цього методу вводяться у наступному діалоговому вікні, яке з'являється після натискання кнопки user button, яка була запрограмована для виконання одинарного різання:



1. Перемістіться стрілками до потрібної початкової позиції X/Y.
2. Встановіть потрібний відповідний матеріал або відредагуйте швидкість подачі для матеріалу за замовчуванням у PARAMETERS Tab.
3. Натисніть призначену кнопку користувача для одиночного обрізання.
4. Введіть довжину різу вздовж осей X та/або Y.
5. Натисніть кнопку **CUT**, і розпочнеться різання.

Підвіска одинарного огранювання Якщо верстат оснащений підвісним пультом, який може запускати та зупиняти шпиндель, а також переміщувати осі X та Y, користувач може вручну виконувати один різ.

1. Перемістіться стрілками до потрібної початкової позиції X/Y.
2. Встановіть потрібну швидкість подачі за допомогою повзунка «Швидкість подачі».
3. Розпочніть процес різання, запустивши шпиндель.
4. Після зондування пальник спалахне.
5. Після отримання сигналу «Дуга в порядку» верстат можна переміщувати вздовж лінії різу за допомогою кнопок поштовху.
6. Після завершення різання зупиніть шпиндель.
7. Пальник вимкнеться, і вісь Z повернеться у вихідне положення.

Ручний одинарний розріз

Для ручного одноразового вирізання необхідно, щоб у розділі «Налаштування» вкладки «Налаштуванн було ввімкнено «плазмові:клавіатурні скорочення,клавіатурні скорочення» або щоб була вказана спеціальна кнопка користувача як кнопка «плазмові:кнопка-ручне вирізання,ручне вирізання».

Якщо користувач використовує власну кнопку користувача, тоді замініть **F9** на **Кнопка користувача** у наступному описі.

1. Перемістіться стрілками до потрібної початкової позиції X/Y.
2. Почніть процедуру, натиснувши **F9**. Швидкість поштовху буде автоматично встановлена на швидкість подачі поточного вибраного матеріалу. Мітка поштовху буде блимати, вказуючи, що швидкість поштовху тимчасово змінена (маніпуляція швидкістю поштовху буде вимкнена, поки активне ручне різання). **CYCLE START** (СТАРТ ЦИКЛУ) зміниться на **MANUAL CUT** (РУЧНЕ РІЗАННЯ) і буде блимати.
3. Після зондування пальник спалахне.
4. Після отримання сигналу «Дуга в порядку» верстат можна переміщувати вздовж лінії різу за допомогою клавіш поштовхового переміщення.
5. Висота Z залишатиметься заблокованою на висоті різання протягом усього ручного різання, незалежно від стану контролера висоти пальника **УВІМК.**.
6. Після завершення різання натисніть **F9** або **Esc** або кнопку **CYCLE STOP**.
7. Пальник вимкнеться, і вісь Z повернеться у вихідне положення.
8. Швидкість ручного переміщення автоматично повернеться до значення, яке було до початку процесу ручного різання, етикетка перестане блимати, і буде увімкнено регулювання швидкості ручного переміщення. **MANUAL CUT** (Ручне різання) перестане блимати і повернеться до **CYCLE START** (Початок циклу).

Note
Якщо пальник згасне під час різання, користувач все одно повинен натиснути **F9** або **Esc** або кнопку **CYCLE STOP**, щоб завершити різання. Це очистить зміщення Z і поверне пальник у початкове положення..

10.8.9.28 Товсті матеріали

Різання товстих матеріалів може бути проблематичним, оскільки велика кількість розплавленого металу, що утворюється під час пробивання, може скоротити термін служби витратних матеріалів, а також може спричинити утворення калюжі, висота якої може бути такою, що пальник може зачепити її під час переміщення на висоту різання.

У QtPlasmaC вбудовано кілька функцій, які допомагають полегшити ці проблеми: Pierce Only і Puddle Jump, описані в цьому розділі, а також Wiggle Pierce і Ramp Pierce, описані в розділі Moving Pierce.

Тільки Пірс

Режим «Тільки пробивання» перетворює завантажену програму G-коду, а потім запускає програму для пробивання матеріалу у вихідній позиції кожного різу. Команди «Нанесення контуру» та «Позначення» будуть проігноровані, і в цих місцях пробивання не відбуватиметься.

Цей режим корисний для товстих матеріалів, на поверхні яких під час проколювання може утворюватися достатня кількість шлаку, що заважає роботі пальника під час різання. Весь лист можна проколоти, а потім очистити перед різанням.

Можна використовувати витратні матеріали, термін служби яких майже добігає кінця, для проколюван
а потім їх можна замінити на якісні витратні матеріали, які можна використовувати під час різання.

Місце проколювання в режимі **Тільки проколювання** може бути зміщене по осях X та/або Y, щоб забезпечити правильне переміщення дуги під час проколювання після повернення до режиму **Звичайне різання**. Параметри зміщення по осях X та Y знаходяться в рамці ТІЛЬКИ ПРОКОЛЮВАННЯ розділу КОНФІГУРАЦІЯ вкладки Вкладка ПАРАМЕТРИ

Тільки прокол – це один із двох різних типів різання

Стрибок по калюжі Стрибок над калюжею — це висота, на яку переміщується пальник після проколювання і перед переміщенням на **висоту різання**, яка виражається у відсотках від **висоти проколювання**. Це дозволяє пальнику очистити будь-яку калюжу розплавленого матеріалу, яка може утворитися в результаті проколювання. Максимально допустима висота становить 200% від **висоти проколювання**

Налаштування для **Puddle Jump** описано в параметри різання

Рекомендований варіант — використовувати **Тільки проколювання**, оскільки він дозволяє використо
витратні матеріали, термін служби яких майже добігає кінця.



Important
ФУНКЦІЯ «СТРИБОК ПО КАЛЮЖІ» ВИМКНЕНА ПІД ЧАС ВІДНОВЛЕННЯ ПІСЛЯ РІЗАННЯ

10.8.9.29 Режим сітки (різання розширеного металу)

QtPlasmaC здатний різати розширений (сітчастий) метал за умови, що машина має пальник із запальною дугою та здатна працювати в режимі постійної запальної дуги (CPA).

Сітчастий режим вимикає THC, а також ігнорує втрату сигналу Arc OK під час різання. Його можна вибрати, встановивши прапорець **Сітчастий режим** у розділі КЕРУВАННЯ на вкладці MAIN.

Якщо на верстаті ввімкнено зв'язок RS485 із плазмовим різаком Hypertherm PowerMax, вибір **Mesh Mode** (Режим сітки) автоматично замінить **Cut Mode** (Режим різання) для поточного вибраного матеріалу та встановить режим різання 2 (CPA). Коли **Mesh Mode** (Режим сітки) вимкнено, **Cut Mode** (Режим різання) повернеться до режиму різання за замовчуванням для поточного вибраного матеріалу.

Також можна розпочати різання в **режимі сітки** без отримання сигналу «Дуга в порядку», встановивши прапорець **Ігнорувати дугу в порядку** у розділі КЕРУВАННЯ вкладки ОСНОВНА.

Як **Режим сітки**, так і **Ігнорувати дугу OK** можна вмикати/вимикати будь-коли під час виконання завдання.

10.8.9.30 Ігнорувати дугу OK

Ignore Arc OK Режим вимикає THC, розпочинає різання без необхідності сигналу Arc OK та ігнорує втрату сигналу Arc OK під час різання.

Цей режим можна вибрати:

1. Позначення кнопки **Ігнорувати дугу OK** у розділі КЕРУВАННЯ вкладки MAIN.
2. Встановлення виводу HAL **motion.digital-out-01** на 1 за допомогою G-коду.
   - `M62 P1` увімкне **Ігнорувати дугу OK** (синхронізовано з рухом)
   - `M63 P1` вимкне **Ігнорувати дугу OK** (синхронізовано з рухом)
   - `M64 P1` увімкне **Ігнорувати дугу OK** (негайно)
   - `M65 P1` вимкне **Ігнорувати дугу OK** (негайно)

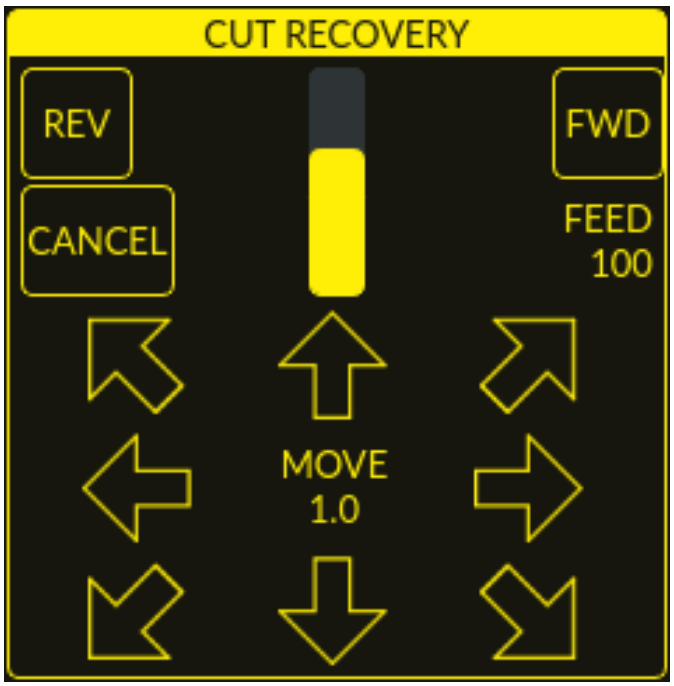
Важливо ретельно розуміти різницю між **Синхронізовано з рухом** та **Негайно**:

- `M62` та `M63` (синхронізовано з рухом) — фактична зміна зазначеного виходу (наприклад, P2 (THC)) відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходу не відбудуться. Найкраще програмувати код руху (наприклад, G0 або G1) одразу після M62 або M63.
- `M64` та `M65` (Негайні) — ці команди виконуються негайно після їх отримання контролером руху. Оскільки вони не синхронізовані з рухом, вони порушують злиття. Це означає, що якщо ці коди використовуються посеред активних кодів руху, рух призупиняється для активації цих команд.

Цей режим також можна використовувати разом із **Режимом сітки**, якщо користувачеві не потрібен сигнал «Дуга в порядку» для початку різання.

Як **Режим сітки**, так і **Ігнорувати дугу OK** можна вмикати/вимикати будь-коли під час виконання завдання.

10.8.9.31 Відновлення від порізів



Ця функція створить панель CUT RECOVERY (Відновлення різання), яка дозволить перемістити пальник подалі від траєкторії різання під час події paused motion (плазма:призупинений рух,призупине рух), щоб розташувати пальник над відходами матеріалу, що ріжеться, так щоб різання поновилося з мінімальним виривом дуги. Панель CUT RECOVERY (Відновлення різання) автоматично відобразитьс над панеллю JOGGING (Поступовий рух), коли рух буде призупинено.

Бажано регулювати положення пальника з точки, в якій відбулася пауза руху, однак якщо перед встановленням нової початкової точки необхідно перемістится вздовж траєкторії різання, користувач може скористатися елементами керування паузою руху (**REV**, **FWD** та повзунком **JOG-SPEED**) у верхній частині панелі CUT RECOVERY. Коли користувач задоволений положенням пальника вздовж траєкторії різання, переміщення з траєкторії різання здійснюється натисканням кнопок **DIRECTION**. Кожне натискання кнопки **DIRECTION** переміщує пальник на відстань, еквівалентну параметру **Kerf Width** (Ширина пропилу) для поточного вибраного матеріалу.

Щойно пальник буде зміщено з траєкторії різання, елементи керування рухом на паузі (**REV**, **FWD** та повзунок **JOG-SPEED**) у верхній частині панелі ВІДНОВЛЕННЯ РІЗАННЯ стануть деактивован

Коли положення пальника буде задовільним, натисніть **CYCLE RESUME** (ПРОДОВЖИТИ ЦИКЛ), і різання відновиться з нової позиції та пройде найкоротшу відстань до початкового місця зупинки руху. Панель CUT RECOVERY (ВІДНОВЛЕННЯ РІЗАННЯ) закриється, а панель JOGGING (ПЕРЕМІЩЕН з’явиться, коли пальник повернеться до початкового місця зупинки руху.

Натискання кнопки **СКАСУВАТИ ПЕРЕМІЩЕННЯ** призведе до повернення пальника до того місця, де він був розташований до використання клавіш напрямку для зміщення пальника. Це не скине жодного руху **НАЗАД** або **ВПЕРЕД**.

Натискання кнопки **CYCLE STOP** призведе до повернення пальника в положення, в якому він знаходився до використання клавіш напрямку для зміщення пальника, а накладка панелі CUT RECOVERY повернеться до панелі JOGGING. Це не призведе до скидання будь-яких рухів **REV** або **FWD**.

Якщо встановлено лазерний вирівнювач, його можна використовувати під час відновлення різання для дуже точного позиціонування нових координат початку. Якщо зміщення осі X або осі Y для лазера призведе до виходу верстата за межі, на екрані з’явиться повідомлення про помилку.

Щоб використовувати лазер для відновлення після порізу під час паузи:

1. Натисніть кнопку **ЛАЗЕР**.
2. Кнопка **LASER** стане неактивною, контакт HAL з назвою qtplasmac.laser_on буде увімкнено, а осі X і Y зміститься так, що лазерні перехрестя вкажуть початкові координати різання після його відновлення.
3. Продовжуйте відновлення після розрізу, як описано вище.

Якщо під час натискання кнопки **CANCEL MOVE** діє лазерне зміщення, це зміщення також буде очищено.

Note
Рухи відновлення після різання будуть обмежені радіусом 10 мм (0,4 дюйма) або від точки, де програма була призупинена, або від останньої точки на шляху різання, якщо використовувався призупинений рух.



Important
СТРИБОК ЧЕРЕЗ КАЛЮЖУ ВИМКНЕНО ПІД ЧАС ВІДНОВЛЕННЯ ПІСЛЯ РІЗАННЯ

10.8.9.32 Бігти від лінії

Якщо користувач увімкнув опцію «Run From Line» (Виконати з рядка) у розділі «GUI SETTINGS» (Налаштування графічного інтерфейсу) вкладки «SETTINGS Tab», він матиме можливість запускати програму G-коду з будь-якого рядка за допомогою таких методів:

1. Клацання будь-якої лінії у вікні попереднього перегляду
2. Клацання будь-якого рядка у вікні G-коду

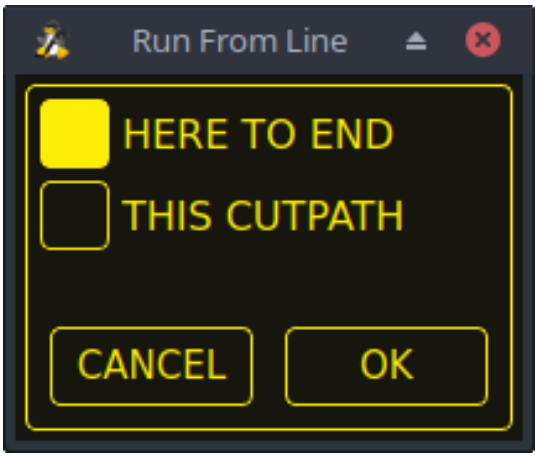
Важливо зазначити, що програми G-коду можна запускати з будь-якої вибраної лінії за допомогою цього методу, однак залежно від вибраної лінії введення може бути неможливим. У цьому випадку буде відображено повідомлення про помилку, щоб повідомити користувача про те, що розрахунок введення був неможливим.

Після того, як користувач вибрав місце початку, кнопка **CYCLE START** (Початок циклу) буде блимати **”SELECTED *nn*”** (Вибрано *nn*), де *nn* — це відповідний номер вибраного рядка. Натискання цієї кнопки відкриє наступне діалогове вікно Run From Line (Виконати з рядка):

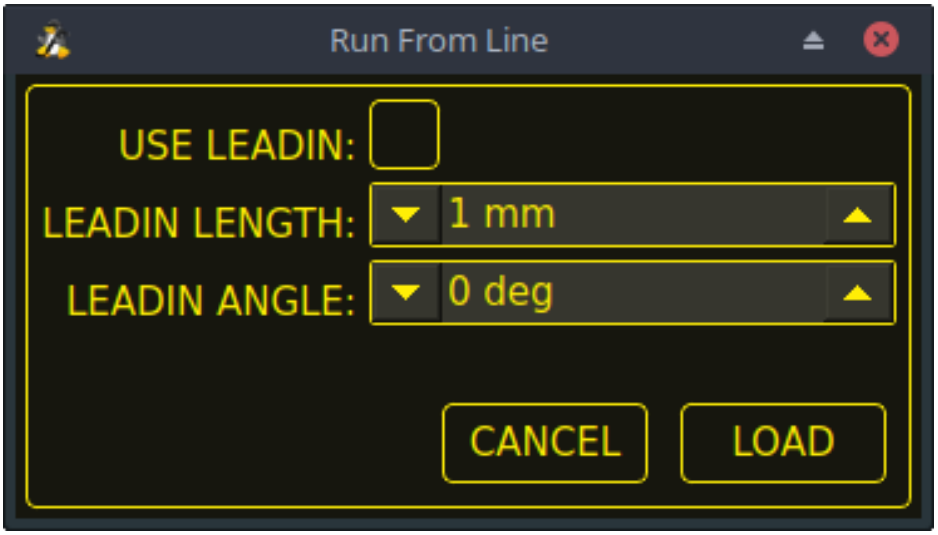
Неможливо використовувати команду «Виконати з рядка» з підпрограми. Якщо користувач вибере рядок у підпрограмі та натисне **«SELECTED *nn*»**, з’явиться повідомлення про помилку, що містить ім’я O-коду підпрограми.

Неможливо використовувати функцію «Виконати з лінії», якщо попередній G-код активував компенсац різака. Якщо користувач вибере лінію, коли компенсація різака активна, і натисне **«SELECTED *nn*»**, з’явиться повідомлення про помилку.

Ви можете вибрати нову лінію, поки активна опція «Виконати від лінії».



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).



Ім’я	**Опис**
ВИКОРИСТОВУВАТИ LEADIN	Цей перемикач дозволить користувачеві розпочати вибраний рядок з вступу.
ДОВЖИНА ВІДВОДУ	Якщо вибрано «ВИКОРИСТАТИ ВІДСТАНОВЛЕННЯ», це встановить довжину відступу в одиницях вимірювання машини.
КУТ НАХОДУ	Якщо вибрано USE LEADIN, це встановить кут наближення для введення. Кут вимірюється таким чином, що позитивне збільшення значення переміщує введення проти годинникової стрілки: 0 градусів = положення 3 години 90 градусів = положення 12 годин 180 градусів = положення 9 годин 270 градусів = положення 6 годин
СКАСУВАТИ	Ця кнопка скасує діалогове вікно «Виконати від рядка» та будь-який вибір.

Ім’я	Опис
НАВАНТАЖЕННЯ	Ця кнопка завантажить тимчасову програму «rfl.ngc» із застосуванням будь-яких вибраних параметрів введення. Якщо введення не може бути обчислено для вибраної лінії, з’явиться таке повідомлення про помилку: «Неможливо обчислити введення для цього різу Програма буде працювати від вибраної лінії без застосування введення»

Після натискання кнопки **ЗАВАНТАЖИТИ** миготлива кнопка ”ВИБРАНА *nn*” зміниться на кнопку **ЗАПУСТИТИ З РЯДКА ЦИКЛУ**. Натисніть цю кнопку, щоб запустити програму з початку вибраного рядка.

THIS CUTPATH буде виконуватися лише той шлях розрізу, частиною якого є вибраний сегмент.

Миготлива кнопка ”SELECTED *nn*” зміниться на кнопку **RUN FROM LINE CYCLE START**. Натисніть цю кнопку, щоб запустити вибрану траєкторію різання.

Вибір «Біжи з лінії» можна скасувати такими способами:

1. Клацніть на фоні вікна попереднього перегляду — цей метод скасує вибір лінії розрізу у вікні попереднього перегляду або лінії G-коду у вікні G-коду.
2. Клацніть текст першого рядка програми G-коду на екрані G-коду — цей метод скасує вибір лінії розрізу у вікні попереднього перегляду або лінії G-коду у вікні G-коду.
3. Натискання кнопки **RELOAD** у заголовку вікна G-коду — цей метод скасує процес Run From Line, якщо в діалоговому вікні Run From Line було натиснуто кнопку LOAD і в заголовку вікна G-коду як ім’я завантаженого файлу відображається «rfl.ngc». Це поверне користувача до спочатку завантаженого файлу.

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 Писар

Окрім плазмового пальника, QtPlasmaC може керувати писачем.

Використання рискача вимагає використання таблиці інструментів LinuxCNC. Інструмент 0 призначає для плазмового пальника, а інструмент 1 — для рискача. Зсуви осей X і Y рисувального інструменту від плазмового пальника потрібно ввести в таблицю інструментів LinuxCNC. Це робиться шляхом редагування таблиці інструментів через головний графічний інтерфейс або шляхом редагування файлу **tool.tbl** у конфігураційному каталозі ***<machine_name>***. Це буде зроблено після того, як рисувальний інструмент зможе переміститися до заготовки, щоб допомогти визначити відповідний зсув.

Зсуви плазмового пальника для X і Y завжди будуть дорівнювати нулю. Інструменти вибираються за допомогою команди **T*n* M6**, за якою слідує команда **G43 H0**, необхідна для застосування зсувів. Потім інструмент запускається командою **M3 $*n* S1**. Для *n* використовуйте 0 для різання пальником або 1 для розмітки.

Щоб зупинити писач, скористайтеся командою G-коду **M5 $1**.

Якщо користувач ще не призначив контакти HAL для скриба в майстрі конфігурації, він може зробити це за допомогою відповідного configuration wizard або вручну, редагуючи файл HAL, див. modifying QtPlasmaC.

Для управління ритчем використовуються два виходи HAL: перший вихід використовується для озброєння ритча, який переміщує ритч на поверхню матеріалу. Після закінчення часу Arm Delay другий вихід використовується для запуску ритча. Після закінчення часу On Delay починається рух.

Використання QtPlasmaC після ввімкнення скарбника вимагає вибору або пальника, або скарбника в кожному файлі G-коду як інструмента LinuxCNC.

Першим кроком є встановлення зміщень для скарбника, виконавши процедуру, описану в Peripheral Offsets.

Останній крок – встановити необхідний параметр scribe delays:

1. **Arm Delay** - дає час писареві спуститися на поверхню матеріалу.
2. **On Delay** - дає час писареві почати роботу, перш ніж почнеться рух.

Збережіть параметри на вкладці «Конфігурація».

Після виконання вищевказаних інструкцій, можна перевірити ручне нанесення ліній, ввівши команду **M3 $1 S1** у вхід MDI. Користувачеві може бути корисно скористатися цим методом, щоб нанести невелику лінію, а потім спробувати подати імпульс пальника в тому ж місці, щоб вирівняти зміщення між лінією та пальником.

Щоб скористатися інструментом різання з G-коду:

```
...
M52 P1 (b''уb''b''вb''b''іb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''аb''b''дb''b' ↩
    'аb''b''пb''b''тb''b''иb''b''вb''b''нb''b''уb'' b''пb''b''оb''b''дb''b''аb''b''чb''b' ↩
    'уb'')
F#<_hal[plasmac.cut-feed-rate]>
T1 M6 (b''вb''b''иb''b''6b''b''рb''b''аb''b''тb''b''иb'' b''рb''b''іb''b''зb''b''еb''b' ↩
    'цb''b''ьb'')
G43 H0 (b''зb''b''аb''b''сb''b''тb''b''оb''b''сb''b''уb''b''вb''b''аb''b''тb''b''иb'' b' ↩
    'зb''b''мb''b''іb''b''щb''b''еb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b' ↩
    'оb''b''тb''b''оb''b''чb''b''нb''b''оb''b''гb''b''оb'' b''іb''b''нb''b''сb''b''тb''b' ↩
    'рb''b''уb''b''мb''b''еb''b''нb''b''тb''b''уb'')
M3 $1 S1 (b''зb''b''аb''b''пb''b''уb''b''сb''b''тb''b''иb''b''тb''b''иb'' b''рb''b''іb''b' ↩
    'зb''b''еb''b''цb''b''ьb'').

M5 $1 (b''зb''b''уb''b''пb''b''иb''b''нb''b''иb''b''тb''b''иb'' b''рb''b''іb''b''зb''b' ↩
    'еb''b''цb''b''ьb'')
.
T0 M6 (b''вb''b''иb''b''6b''b''рb''b''аb''b''тb''b''иb'' b''пb''b''аb''b''лb''b''ьb''b' ↩
    'нb''b''иb''b''кb'')
G43 H0 (b''зb''b''аb''b''сb''b''тb''b''оb''b''сb''b''уb''b''вb''b''аb''b''тb''b''иb'' b' ↩
    'зb''b''мb''b''іb''b''щb''b''еb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b' ↩
    'оb''b''тb''b''оb''b''чb''b''нb''b''оb''b''гb''b''оb'' b''іb''b''нb''b''сb''b''тb''b' ↩
    'рb''b''уb''b''мb''b''еb''b''нb''b''тb''b''уb'')
G0 X0 Y0 (b''пb''b''оb''b''зb''b''иb''b''цb''b''іb''b''яb'' b''пb''b''аb''b''рb''b''кb''b' ↩
    'уb''b''вb''b''аb''b''нb''b''нb''b''яb'')
M5 $-1 (b''зb''b''аb''b''вb''b''еb''b''рb''b''шb''b''иb''b''тb''b''иb'' b''вb''b''сb''b' ↩
    'еb'')
```

Рекомендується перемикатися назад на пальник в кінці програми перед останнім швидким паркуванням, щоб машина завжди перебувала в одному й тому ж стані на холостому ходу.

Користувач може перемикатися між пальником та розмічувачем будь-яку кількість разів під час програми, використовуючи відповідні G-коди.

Видача команди **M3 S1** (без $*n*) призведе до того, що машина поводитиметься так, ніби була видана команда **M3 $0 S1**, а видача команди **M5** (без $*n*) призведе до того, що машина поводитиметься так, ніби була видана команда **M5 $0**. Це контролюватиме запалювання пальника за замовчуванням, щоб забезпечити зворотну сумісність із попередніми файлами G-коду.



Warning
Якщо у файлі <*machine_name*>.hal встановлено параметр ручної зміни інструменту, то QtPlasmaC перетворить його на параметр автоматичної зміни інструменту.

10.8.9.34 Плямистість

Для точкового маркування матеріалу перед свердлінням тощо, QtPlasmaC може імпульсно працювати з пальником протягом короткого часу, щоб позначити місце для свердління.

Споттінг можна налаштувати, виконавши такі кроки:

1. Встановіть **поріг** напруги дуги в розділі «Споттинг» на вкладці «PARAMETERS Tab». Встановлення порогу напруги на нуль призведе до того, що таймер затримки почне відлік відразу після запуску пальника. Встановлення порогу напруги вище нуля призведе до того, що таймер затримки почне відлік, коли напруга дуги досягне порогу напруги.
2. Встановіть **Час увімкнення** у розділі «Виявлення» на вкладці Вкладка ПАРАМЕТРИ. Після закінчення часу, встановленого в **Час увімкнення**, пальник вимкнеться. Час можна регулювати в діапазоні від 0 до 9999 мілісекунд.

Потім пальник вмикається в G-коді за допомогою команди **M3 $2 S1**, яка вибирає плазмовий пальник як інструмент для виявлення плям.

Щоб вимкнути пальник, скористайтеся командою G-коду **M5 $2**.

Для отримання додаткової інформації про кілька інструментів див. кілька інструментів.

LinuxCNC (QtPlasmaC) вимагає певного руху між будь-якими командами **M3** та **M5**. З цієї причини потрібно запрограмувати мінімальний рух на високій швидкості.

Приклад G-коду:

```
G21 (b''мb''b''eb''b''тb''b''pb''b''иb''b''чb''b''нb''b''ab'' b''cb''b''иb''b''cb''b''тb''b ↩
    ''eb''b''мb''b''ab'')
F99999 (b''вb''b''иb''b''cb''b''ob''b''кb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'')

.
G0 X10 Y10
M3 $2 S1 (b''тb''b''ob''b''чb''b''кb''b''ob''b''вb''b''eb'' b''вb''b''ib''b''дb''b''лb''b' ↩
    'ib''b''кb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb'')
G91 (b''pb''b''eb''b''жb''b''иb''b''мb'' b''вb''b''ib''b''дb''b''нb''b''ob''b''cb''b''нb''b ↩
    ''ob''b''їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G1 X0.000001
G90 (b''pb''b''eb''b''жb''b''иb''b''мb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b ↩
    ''нb''b''ob''b''їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
M5 $2 (b''тb''b''ob''b''чb''b''кb''b''ob''b''вb''b''eb'' b''вb''b''ib''b''дb''b''лb''b' ↩
    'ib''b''кb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb'')

.
G0 X0 Y0
G90
M2
```

Note
Висока швидкість подачі 99999 забезпечує найвищу швидкість подачі верстата.



Important
ДЕЯКІ ПЛАЗМОВІ РІЗАКИ НЕ ПІДХОДЯТЬ ДЛЯ ЦІЄЇ ФУНКЦІЇ.
РЕКОМЕНДУЄТЬСЯ КОРИСТУВАЧУ ПРОВЕТИ ДЕЯКІ ТЕСТОВІ ВИПРОБУВАННЯ, ЩОБ ПЕРЕКОНАТИСЯ, ЩО ПЛАЗМОВИЙ РІЗАК ЗДАТНИЙ ВИКОРИСТОВУВАТИ ЦЮ ФУНКЦІЮ.

10.8.9.35 Різання труб

Різання труб за кутовою віссю A, B або C досягається за допомогою наступного у файлі G-коду:

- #<tube_cut>=1 магічний коментар перед будь-якою командою руху.
- Усі вимірювання матеріалу необхідно виконувати за допомогою кодів прямого зондування за посиланням:../gcode/g-code.html#gcode:g38[G38].
- Потрібний весь рух по осі Z, PlasmaC не виконує внутрішнього руху по осі Z під час різання труби.
- PIERCE_DELAY — єдиний обов’язковий параметр material
- Почніть розріз з M3 $0 S1.
- Завершити розріз за допомогою M5 $0

10.8.9.36 Користувацькі розкладки віртуальної клавіатури

Підтримка віртуальної клавіатури доступна лише для ”вбудованої” екранної клавіатури. Якщо її ще немає в системі, її можна встановити, ввівши наступну команду в терміналі:

```
sudo apt install onboard
```

Для підтримки програмних клавіш використовуються такі два налаштовані макети:



Figure 10.49: Цифрова клавіатура – використовується для вкладок РОЗМОВНА та ПАРАМЕТРИ

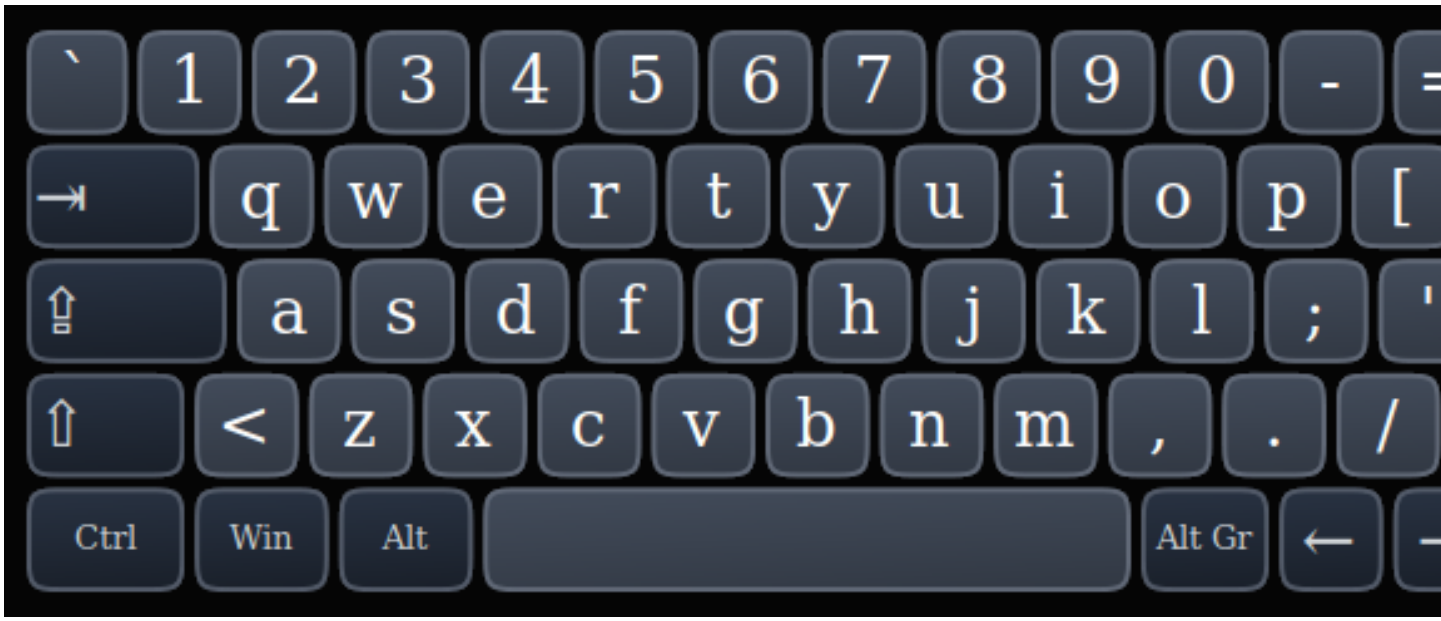


Figure 10.50: Алфавітно-цифрова клавіатура — використовується для редагування G-коду та керування файлами.

Якщо віртуальна клавіатура була переміщена і при наступному відкритті віртуальної клавіатури вона не відображається, то подвійне клацання на піктограмі в системному треї перемістить віртуальну клавіатуру так, щоб ручка переміщення була видима.

10.8.9.37 Комбінації клавіш

Нижче наведено список усіх доступних комбінацій клавіш у QtPlasmaC.

Note
Усі комбінації клавіш вимкнено за замовчуванням.

Щоб скористатися ними, потрібно ввімкнути **Комбінації баз знань** у розділі **ПАРАМЕТРІ графічног**
інтерфейсу на вкладці ПАРАМЕТРІ.

Сполучення клавіш	**Дія**
ESC	Перериває поточний автоматизований рух (наприклад: запущена програма, тест зонда тощо), а також активний імпульс пальника (поводиться так само, як натискання кнопки CYCLE STOP).
F1	Вмикає/вмикає кнопку аварійної зупинки графічного інтерфейсу (якщо кнопка аварійної зупинки графічного інтерфейсу ввімкнена).
F2	Вмикає/вимикає кнопку живлення графічного інтерфейсу.
F9	Вмикає/вимикає команду «Різання», яка використовується для початку або завершення ручного різання.
F12	Показати редактор таблиць стилів.
ALT+RETURN	Переводить QtPlasmaC у режим ручного введення даних (MDI). Зверніть увагу, що ALT + ENTER дасть той самий результат. Крім того, натискання RETURN (або ENTER) без введення даних у MDI призведе до закриття вікна MDI.
`, 1-9, 0	Змінює швидкість поштовхового переміщення на 0%, 10%-90%, 100% від значення, зазначеного у змінній DEFAULT_LINEAR_VELOCITY у розділі **[DISPLAY]** файлу <*назва_машини*>.ini.
SHIFT+`, 1-9, 0	Змінює швидкість швидкого пересування на 0%, 10%-90%, 100%.
CTRL+1-9, 0	Змінює швидкість подачі на 10%-90%, 100%.
CTRL+HOME	Переводить у вихідне положення всі осі, якщо вони ще не переведені у вихідне положення та мають послідовність переведення у вихідне положення, встановлену у файлі <*назва_машини*>.ini. Якщо вони вже переведені у вихідне положення, вони більше не будуть переведені у вихідне положення.
CTRL+R	Запуск циклу, якщо програма ще не запущена. Продовження циклу, якщо програма призупинена.
END	Торкається X та Y до 0.
DEL	Дозволяє користувачеві використовувати лазер для встановлення початку координат з обертанням або без нього. Див. розділ ЛАЗЕР для отримання детальних інструкцій.
ПРОБІЛ	Призупиняє рух.
CTRL+SPACE BAR	Очищає сповіщення.
O	Відкриває нову програму.
L	Завантажує попередньо відкриту програму, якщо жодної програми не завантажено. Перезавантажує поточну програму, якщо програма завантажена.
→	Здійснює додатне струсне переміщення осі X.
←	Здійснює струсне переміщення осі X у негативне положення.
↑	Здійснює додатне струсне переміщення осі Y.
↓	Здійснює струсне переміщення осі Y у негативне положення.
СТОРІНКА ВГОРУ	Здійснює додатне штовхання осі Z.
СТОРІНКА ВНИЗ	Здійснює штовхання осі Z у негативному напрямку.
[	Здійснює додатне штовхання осі A.
]	Здійснює штовхання осі A у негативному напрямку.
.	Здійснює додатне штовхання осі B.
,	Здійснює штовхання осі B у негативному напрямку.

Сполучення клавіш	**Дія**
SHIFT (+ Клавіша Jog)	Клавіша Shift використовується з будь-якою клавішею поштовхового переміщення для виклику швидкого поштовхового переміщення.
+ (+Клавіша Jog)	Клавішу зі знаком «плюс» можна використовувати з будь-якою клавішею поштовхового переміщення для виклику швидкого поштовхового переміщення (поводиться так само, як і SHIFT).
- (+Клавіша Jog)	Клавішу «мінус» можна використовувати з будь-якою клавішею поштовхового переміщення для запуску повільного поштовхового переміщення (10% від відображеної швидкості поштовхового переміщення). Якщо ПОВІЛЬНЕ поштовхове переміщення вже активне, вісь буде рухатися поштовховим переміщенням із відображеною швидкістю поштовхового переміщення.

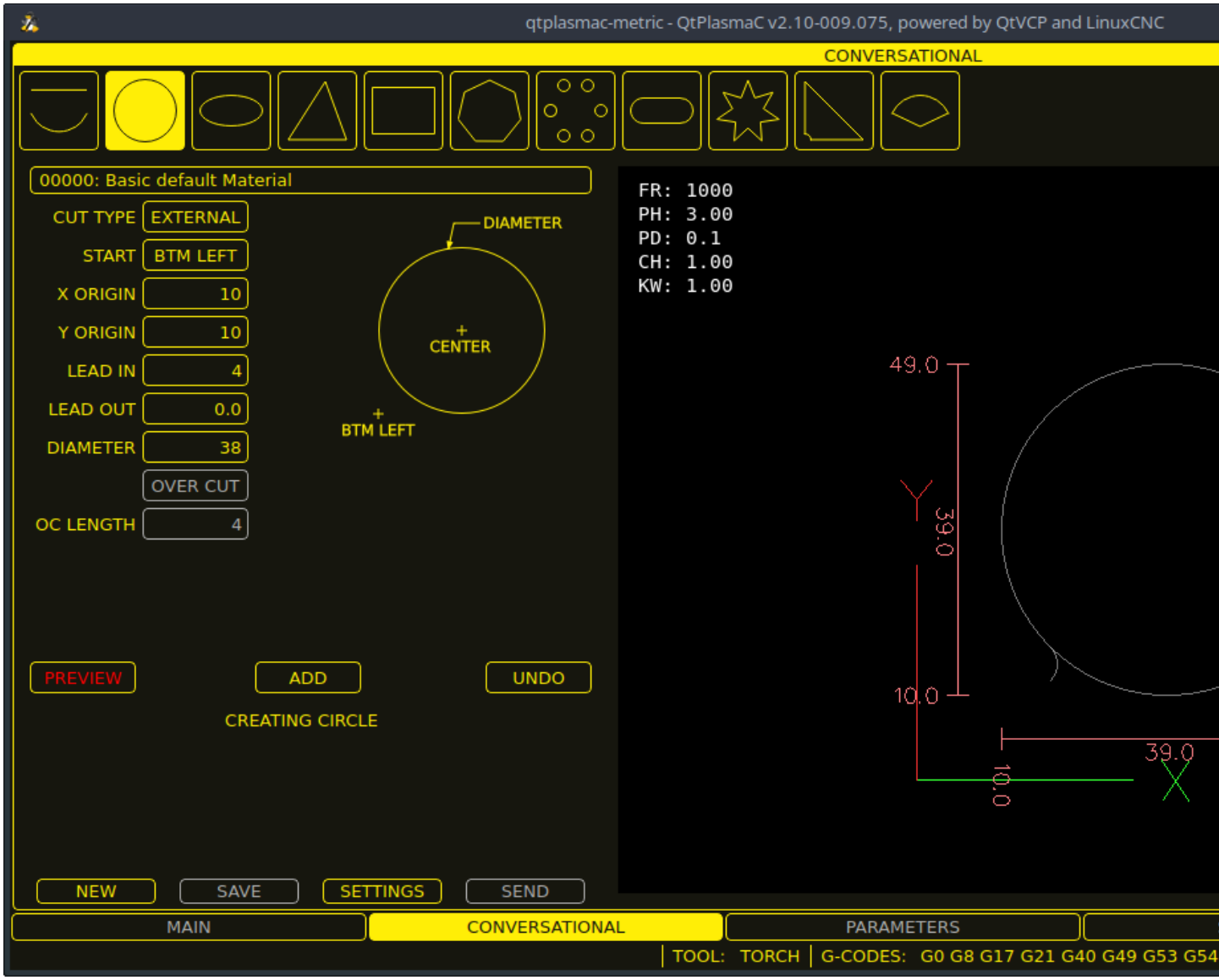
10.8.9.38 MDI

На додаток до типових кодів G і M, які допускаються LinuxCNC у режимі MDI, MDI в QtPlasmaC можна використовувати для доступу до кількох інших корисних функцій. Наступне посилання містить опис функцій та їх використання: Section 12.7.2.15[MDILine Widget]

Note
M3, M4 та M5 не дозволені в MDI QtPlasmaC.

Крім того, натискання RETURN (або ENTER) без введення даних у MDI призведе до закриття вікна MDI.

10.8.10 Бібліотека розмовних фігур



Бібліотека **Conversational Shape Library** складається з декількох основних форм і функцій, які допомагають користувачеві швидко генерувати G-код на верстаті для швидкого вирізання простих форм. Ця функція знаходиться на вкладці CONVERSATIONAL Tab.

Note
Розмовна бібліотека не призначена для заміни CAD/CAM, оскільки її можливості обмежені.

Пусті записи в полях введення форми використовуватимуть поточні налаштування на момент створення G-коду. Наприклад, якщо поле **X start** залишити порожнім, то використовуватиметься поточне положення осі X.

Усі вступні та вихідні лінії є дугами, окрім **Кіл** та **Зірок**:

Кола:

- Якщо коло зовнішнє, то будь-який вхід або вихід буде дугою.

- Якщо коло внутрішнє і являє собою **маленький отвір**, то будь-який вхід буде перпендикулярним, а виходу не буде.
- Якщо коло є внутрішнім і не є **маленьким отвором**, то будь-яке введення та виведення буде дугою. Якщо довжина введення перевищує половину радіуса, то введення повернеться до перпендикулярного положення і виведення не буде. Якщо довжина виведення перевищує половину радіуса, то виведення не буде.

Зірки:

- Вхід знаходиться під тим самим кутом, що й перший розріз, а вихідний — під тим самим кутом, що й останній розріз.

Note
Маленький отвір — це коло, діаметр якого менший за ДІАМЕТР МАЛОГО ОТВОРУ, зазначений на сторінці НАЛАШТУВАННЯ РОЗМОВИ.

Note
Отвори у формі BOLT CIRCLE також відповідатимуть вищезазначеним правилам.

Порядок вирізання відбуватиметься в тому ж порядку, в якому була побудована фігура.

Натискання клавіші **Return** на клавіатурі під час редагування параметрів автоматично відобразить попередній перегляд фігури, якщо введено достатньо параметрів для її створення. Те саме можна зробити, натиснувши будь-який із доступних прапорців.

Загальні функції такі:

Ім’я	Опис
Випадаючий список матеріалів	Дозволяє користувачеві вибрати бажаний матеріал для різання. Якщо на вкладці Вкладка НАЛАШТУВАННЯ вибрано «ПЕРЕГЛЯНУТИ МАТЕРІАЛ», у вікні попереднього перегляду діалогового вікна буде відображено візуальне посилання з основними налаштуваннями різання матеріалу. Приклади: швидкість подачі, висота проколу, затримка проколу, висота різання та ширина прорізу (тільки для діалогового вікна). Якщо увімкнено зв’язок PowerMax, буде показано силу струму різання.
НОВИЙ	Видаляє поточний файл G-коду та завантажує порожній файл G-коду.
ЗБЕРЕГТИ	Відкриває діалогове вікно, яке дозволяє зберегти поточну фігуру як файл G-коду.
НАЛАШТУВАННЯ	Дозволяє змінювати глобальні налаштування.
НАДІСЛАТИ	Завантажує поточну фігуру в LinuxCNC (QtPlasmaC). Якщо останнє редагування не було додано, воно буде відхилено.
ПОПЕРЕДНІЙ ПЕРЕГЛЯД	Відображає попередній перегляд поточної фігури за умови наявності необхідної інформації.
ПРОДОВЖИТИ	Ця кнопка використовується лише для ліній та дуг. Дозволяє додати ще один сегмент до поточного сегмента/сегментів.
ДОДАТИ	Зберігає поточну фігуру в поточному завданні.
ВІДМІНИТИ	Повертає до попередньо збереженого стану.
ПЕРЕЗАВАНТАЖИТИ	Перезавантажує оригінальний файл G-коду або порожній файл, якщо жодного не було завантажено.

Якщо в LinuxCNC (QtPlasmaC) завантажено файл G-коду, коли вибрано CONVERSATIONAL Tab, цей код буде імпортований у діалогове вікно як перша форма завдання. Якщо цей код не потрібен, його можна видалити, натиснувши кнопку **NEW**.

Якщо є додана фігура, яка не збережена або не надіслана, то неможливо перемикатися між вкладками в графічному інтерфейсі користувача. Щоб знову увімкнути перемикання між вкладками, необхідно або **ЗБЕРЕГТИ** фігуру, **НАДІСЛАТИ** фігуру, або натиснути **НОВЕ**, щоб видалити фігуру.

Якщо натиснути **НОВЕ** для видалення доданої фігури, яка не збережена або не надіслана, з’явиться діалогове вікно з попередженням.

Note
Усі відстані вказані в машинних одиницях відносно поточної системи координат користувача, а всі кути — у градусах.

10.8.10.1 Налаштування розмовного режиму

Глобальні налаштування бібліотеки фігур можна встановити, натиснувши кнопку **SETTINGS** (Налаштування) у вкладці CONVERSATIONAL Tab (Плазма: вкладка діалогу). Це відобразить усі доступні параметри налаштувань, які використовуються для створення програми G-коду. До них належать:

- **Преамбула**
- **Постамбула**
- **Origin (По центру або внизу ліворуч)**
- **Довжина введення**
- **Довжина виводу**
- **Малий діаметр отвору**
- **Швидкість малого отвору**
- **Розмір сітки вікна попереднього перегляду**

Будь-яке внутрішнє коло, діаметр якого менше **Діаметра малого отвору**, класифікується як малий отвір і матиме прямий вхід з довжиною, яка є меншою за радіус отвору або задану довжину входу. Швидкість подачі також буде встановлена на **Швидкість малого отвору**.

Преамбула та пост-амбула можуть бути введені як послідовність G-кодів та M-кодів, розділених пробілами. Якщо користувач бажає, щоб у згенерованому G-коді кожен код був у окремому рядку, це можна зробити, розділивши коди за допомогою **\n**.

Це розмістить усі коди в одному рядку:

```
G21 G40 G49 G64p0.1 G80 G90 G92.1 G94 G97
```

Це розмістить кожен код на окремому рядку:

```
G21\nG40\nG49\nM52P1\nG64p0.1\nG80\nG90\nG92.1\nG94\nG97
```

Зверніть увагу, що LinuxCNC не дозволяє використовувати кілька слів **P** в одному рядку.

Натискання кнопки **PRELOAD** скасує всі змінені, але незбережені налаштування.

Натискання кнопки **ЗБЕРЕГТИ** збереже всі налаштування, як вони відображаються.

Натискання кнопки **ВИХІД** закриє панель налаштувань і поверне попередню форму.

10.8.10.2 Розмовні лінії та дуги



Лінії та дуги мають додаткову можливість поєднувати їх для створення складної форми.

Доступні два типи ліній та три типи дуг:

1. **Лінія** з початковою та кінцевою точками.
2. **Лінія**, якій задано початкову точку, довжину та кут.
3. **Дуга**, якій задано початкову точку, точку маршруту та кінцеву точку.
4. **Дуга** за заданими початковою точкою, кінцевою точкою та радіусом.
5. **Дуга** за заданими початковою точкою, довжиною, кутом та радіусом.

Щоб використовувати лінії та дуги:

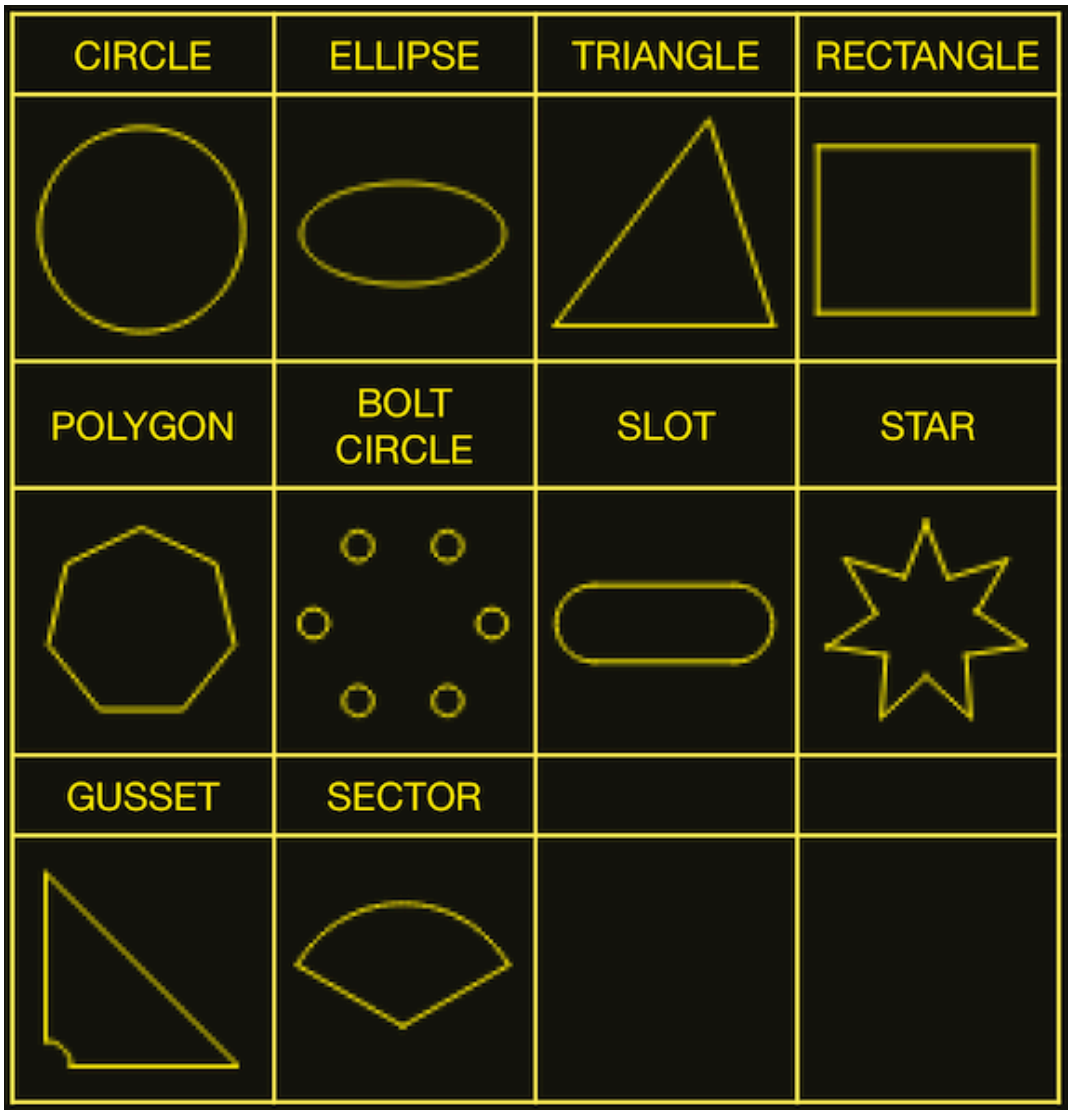
1. Виберіть значок **Лінії та дуги**.
2. Виберіть тип лінії або дуги для створення.
3. Виберіть матеріал з випадаючого списку МАТЕРІАЛ. Якщо матеріал не вибрано, буде використано матеріал за замовчуванням (00000).
4. Введіть потрібні параметри.
5. Натисніть **ПОПЕРЕДНІЙ ПЕРЕГЛЯД**, щоб побачити форму.
6. Якщо вас влаштовує форма, натисніть **ПРОДОВЖИТИ**.
7. За потреби змініть тип лінії або дуги та продовжуйте цю процедуру, доки фігура не буде завершена.
8. Натисніть **НАДІСЛАТИ**, щоб надіслати файл G-коду до LinuxCNC (QtPlasmaC) для різання.

Якщо користувач бажає створити замкнену фігуру, йому потрібно буде створити будь-який необхідний вступний елемент як перший сегмент фігури. Якщо потрібен вступний елемент, він має бути останнім сегментом фігури.

Note
На цьому етапі немає автоматичної опції для створення входу/виходу, якщо форма замкнена.

10.8.10.3 Розмовна одинарна форма

Для створення доступні такі форми:



Щоб створити фігуру:

1. Виберіть відповідний значок для фігури, яку потрібно створити. Будуть відображені доступні параметри.
2. Виберіть матеріал з випадаючого списку МАТЕРІАЛ. Якщо матеріал не вибрано, буде використано матеріал за замовчуванням (00000).
3. Введіть відповідні значення та натисніть **ПОПЕРЕДНІЙ ПЕРЕГЛЯД**, щоб відобразити фігуру.
4. Якщо форма неправильна, відредагуйте значення та натисніть **ПОПЕРЕДНІЙ ПЕРЕДПРОДАЖ**, після чого відобразиться нова форма. Повторюйте, доки не будете задоволені результатом.
5. Натисніть **ДОДАТИ**, щоб додати фігуру до файлу G-коду.
6. Натисніть **НАДІСЛАТИ**, щоб надіслати файл G-коду до LinuxCNC (QtPlasmaC) для різання.

Для **CIRCLE** кнопка **OVER CUT** стане активною, коли буде вибрано CUT TYPE INTERNAL, а значення, введене в поле DIAMETER, буде меншим за параметр Small Hole Diameter у розділі Conversational SETTINGS.

Для **КОЛА БОЛТІВ** кнопка **ОБІРІЗ** буде дійсною, якщо значення, введене в поле ДІАМЕТР ОТВОРУ, менше за параметр ДІАМЕТР МАЛЕНЬКИХ ОТВОРІВ у розділі НАЛАШТУВАННЯ діалогового режиму.

Для наступних форм параметр ЗМІЩЕННЯ РЕЗУ стане активним після визначення ВХОДУ:

1. ТРИКУТНИК
2. ПРЯМОКУТНИК
3. ПОЛІГОН
4. СЛОТ
5. ЗІРКА
6. КЛАВИЦЯ

10.8.10.4 Розмовна група фігур

Кілька фігур можна об’єднати, щоб створити складну групу.

Порядок вирізання групи визначається порядком, у якому окремі фігури додаються до групи.

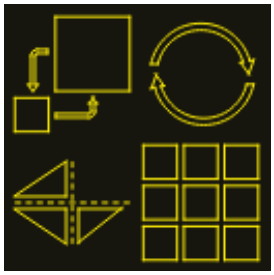
Після додавання фігури до групи її не можна редагувати або вилучати.

Групи не можуть мати фігури, їх можна лише додавати.

Щоб створити групу фігур:

1. Створіть першу фігуру, як у **Одинарна фігура**.
2. Натисніть **ДОДАТИ**, і фігура буде додана до групи.
3. Якщо користувач бажає додати ще одну версію тієї ж форми, відредагуйте необхідні параметри та натисніть **ДОДАТИ**, коли вас задовольнить форма.
4. Якщо користувач бажає додати іншу фігуру, виберіть цю фігуру та створіть її, як у **Одна фігура**.
5. Повторюйте, доки не будуть додані всі необхідні фігури для завершення групи.
6. Натисніть **НАДІСЛАТИ**, щоб надіслати файл G-коду до LinuxCNC (QtPlasmaC) для різання.

10.8.10.5 Розмовний блок



Функція «Розмовний блок» дозволяє виконувати операції з блоками над поточною фігурою або групою фігур, що відображаються у вкладці CONVERSATIONAL Tab. Це може включати файл G-коду, який не був створений за допомогою бібліотеки розмовних фігур, що була раніше завантажена з вкладки MAIN Tab.

Раніше збережений файл G-коду блоку також можна завантажити з MAIN Tab, а потім відредагувати будь-які його операції за допомогою функції діалогового блоку.

Всі операції з блоками виконуються в одиницях виміру, властивих для даної машини. Якщо було відкрито файл з іншою системою одиниць виміру, то копія, завантажена у вкладку CONVERSATIONAL, буде перетворена в одиниці виміру, властиві для даної машини, коли вкладка CONVERSATIONAL стане активною.

Блокові операції:

- Повернути
- Масштаб
- Масив
- Дзеркало
- Фліп

Щоб створити блок:

1. Створіть фігуру, групу або використовуйте попередньо завантажений файл G-коду.
2. Натисніть значок **Блокувати**, щоб відкрити вкладку «Блокувати».
3. Введіть відповідні значення на вкладці «Блок» і натисніть **ПОПЕРЕДНІЙ ПЕРЕГЛЯД**, щоб відобразити отримані зміни.
4. Якщо результат неправильний, відредагуйте значення та натисніть **PREVIEW**, після чого буде показано новий результат. Повторюйте, доки не будете задоволені результатом.
5. Натисніть **ДОДАТИ**, щоб завершити процедуру.
6. Натисніть **НАДІСЛАТИ**, щоб надіслати файл G-коду до LinuxCNC (QtPlasmaC) для різання, або **ЗБЕРЕГТИ**, щоб зберегти файл G-коду.

СТОЛОПЦІ ТА РЯДКИ
: визначає кількість дублікатів оригінальної фігури, розташованих у стовпцях і рядках, а також відстань між початком кожної фігури.

ORIGIN
: змістити результат відносно початку координат.

ANGLE
: повернути результат.

SCALE
: масштабувати результат.

РОТАЦІЯ
: повернути фігуру в межах результату.

ДЗЕРКАЛО
: віддзеркалити фігуру відносно її координат X у результаті.

ФЛІП
: перевернути фігуру відносно її координат Y в межах результату.

Якщо результатом є масив фігур, то порядок вирізання результату – від лівого стовпця до правого, починаючи з нижнього рядка та закінчуючи верхнім рядком.

10.8.10.6 Розмовне збереження роботи

Поточне завдання, що відображається на панелі попереднього перегляду, можна зберегти в будь-який момент за допомогою кнопки **SAVE** (Зберегти) внизу. Якщо G-код був надісланий до LinuxCNC (QtPlasmaC) і користувач залишив вкладку CONVERSATIONAL Tab, він все одно може зберегти файл G-коду з графічного інтерфейсу користувача. Крім того, користувач може натиснути CONVERSATIONAL Tab, що перезавантажить завдання, після чого можна натиснути кнопку **SAVE**.

10.8.11 Повідомлення про помилки

10.8.11.1 Реєстрація помилок

Всі помилки записуються в журнал машини, який можна переглянути в STATISTICS Tab. Файл журналу зберігається в каталозі конфігурації після завершення роботи QtPlasmaC. Зберігаються п’ять останніх файлів журналу, після чого найстаріший файл журналу видаляється кожного разу, коли створюється новий файл журналу. Ці збережені файли журналу можна переглянути за допомогою будь-якого текстового редактора.

10.8.11.2 Відображення повідомлення про помилку

За замовчуванням QtPlasmaC відображатиме повідомлення про помилки у спливаючому вікні Operator Error (Помилка оператора). Крім того, QtPlasmaC сповіщатиме користувача про те, що помилка була відправлена до журналу системи, відображаючи повідомлення **«ERROR SENT TO MACHINE LOG»** (Помилка відправлена до журналу системи) у нижній лівій частині рядка стану.

Користувач може вимкнути спливаюче вікно «Помилка оператора» і переглядати повідомлення про помилки, перейшовши на вкладку STATISTICS Tab, змінивши наступну опцію на **False** у **[SCREEN_OPTIONS]** файлу <*machine_name*>.prefs у каталозі <*machine_name*>:

```
desktop_notify
```

Note
<*machine_name*>.Налаштування необхідно редагувати із закритим QtPlasmaC, інакше будь-які зміни будуть перезаписані після виходу.

Крім того, можна зробити так, щоб повідомлення **ERROR SENT TO MACHINE LOG** (Помилка відправлена до журналу машини) мигало, щоб привернути увагу користувача, додавши або відредагува наступну опцію в розділі **[GUI_OPTIONS]** файлу <*machine_name*>.prefs:

```
b''Пb''b''ob''b''мb''b''иb''b''лb''b''кb''b''ab'' b''cb''b''пb''b''ab''b''лb''b''ab''b' ↩
    'xb''b''yb'' = True
```

10.8.11.3 Критичні помилки

QtPlasmaC виводить низку повідомлень про помилки, щоб інформувати користувача про несправності в міру їх виникнення. Повідомлення можна розділити на дві групи: **Критичні** та **Попередження**.

Критичні помилки призведуть до призупинення виконання програми, і оператору потрібно буде усунути причину помилки, перш ніж продовжити.

Якщо помилка виникла під час різання, то рух вперед або назад дозволено, поки машина призупинена, щоб користувач міг змінити положення машини перед відновленням різання.

Після усунення помилки програму можна відновити.

Ці помилки вказують на те, що відповідний датчик був активований під час різання:

- **активовано вимикач розриву, програма призупинена**
- **поплавковий вимикач активовано, програма призупинена**
- **омічний зонд активовано, програма призупинена**

Ці помилки вказують на те, що відповідний датчик був активований до початку зондування:

- **омічний зонд виявлено до призупинення програми зондування**
- **поплавковий вимикач виявлено до призупинення програми зондування**
- **виявлено розривний вимикач до призупинення програми зондування**

Сигнал Arc OK було втрачено під час різання, до досягнення команди **M5**:

- **дійсна програма втрати дуги призупинена**

Вісь Z досягла нижньої межі до того, як було виявлено заготовку:

- **досягнуто нижньої межі під час паузи програми зондування**

Заготовка занадто висока для будь-якого безпечного швидкого вилучення:

- **матеріал занадто високий для безпечного переміщення, програма призупинена**

Одне з цих значень у розділі MATERIAL вкладки PARAMETERS є недійсним (наприклад: якщо воно встановлено на нуль):

- **недійсна висота проколу або недійсна висота різання або недійсне значення напруги різання, програму призупинено**

Дугу не виявлено після спроб запуску заданої кількості разів, зазначеної у **Max Starts** (Максимальна кількість запусків) у кадрі ARC розділу CONFIGURATION на вкладці PARAMETERS:

- **дуга не виявлена після <*n*>d спроб запуску, програма призупинена**
- **дуга не виявлена після <*n*>d спроб запуску, ручне різання зупинено**

THC призвів до досягнення нижньої межі під час різання:

- **досягнуто нижньої межі, поки програма зниження рівня THC призупинена**

THC призвело до досягнення верхньої межі під час різання:

- **досягнуто верхньої межі, поки програма підвищення рівня THC призупинена**

Ці помилки вказують на те, що висота переміщення для проколу перевищить MAX_LIMIT осі Z для відповідного методу зондування:

- **висота проколу перевищить максимальну граничну умову осі Z, виявлену під час переміщення до висоти зонда під час зондування поплавковим вимикачем**
- **висота проколу перевищить максимальну граничну умову осі Z, виявлену під час руху до висоти зонда під час омічного зондування**

Ці помилки вказують на те, що висота проколу перевищить максимальну безпечну висоту осі Z для відповідного методу зондування:

- **висота проколу перевищить максимальну безпечну висоту осі Z, виявлену під час зондування поплавковим вимикачем**
- **висота проколу перевищить максимальну безпечну висоту осі Z, виявлену під час омічного зондування**

10.8.11.4 Попереджувальні повідомлення

Попереджувальні повідомлення не призупиняють роботу програми та є лише інформаційними.

Ці повідомлення вказують на те, що відповідний датчик був активований до початку перевірки зонда:

- **омічний зонд виявлено до переривання тесту зондувального зонда**
- **поплавковий вимикач виявлено перед перериванням тесту зонда**
- **виявлено розривний вимикач перед перериванням тесту зонду**

Це вказує на те, що відповідний датчик активувався під час заміни витратного матеріалу:

- **під час заміни витратних матеріалів активовано обрив, плаваючий сигнал або омічний сигнал, рух призупиняється**
  ПОПЕРЕДЖЕННЯ: РУХ ВІДНОВИТЬСЯ НЕГАЙНО ПІСЛЯ ВИРІШЕННЯ ЦІЄЇ СТАНОВИ!



Warning
РУХ ЗМІНИ ВИТРАТНИХ МАТЕРІАЛІВ ВІДНОВИТЬСЯ НЕГАЙНО ПІСЛЯ ВИРІШЕННЯ ПРОБЛЕМИ З АКТИВАЦІЄЮ ВІДПОВІДНОГО ДАТЧИКА.

Це вказує на те, що відповідний датчик був активований під час тестування зонда:

- **під час перевірки зонда виявлено розривний вимикач**

Це вказує на те, що контакт із зондом було втрачено перед тим, як зондування почалося вгору для знаходження нульової точки:

- **помилка спрацьовування зонда під час зондування**

Це вказує на те, що під час випробування зонда було досягнуто нижньої межі:

- **під час тестування зонда досягнуто нижньої межі**

Це вказує на те, що переміщення для висоти проколу перевищить MAX_LIMIT осі Z під час відповідного методу зондування:

- **висота проколу перевищить максимальну граничну умову осі Z, виявлену під час переміщення до висоти зонда під час тестування зонда поплавкового вимикача**
- **висота проколу перевищить максимальну граничну умову осі Z, виявлену під час руху до висоти зонда під час омічного тестування зонда**

Це вказує на те, що безпечна висота була зменшена через те, що THC піднімає вісь Z під час різання:

- **безпечну висоту траверсу зменшено**

Це вказує на те, що значення напруги дуги було недійсним (NAN або INF) під час запуску Qt-PlasmaC.

- **недійсна напруга дуги на вході**

10.8.12 Оновлення QtPlasmaC

10.8.12.1 Стандартне оновлення

Повідомлення про оновлення QtPlasmaC розміщені за адресою https://forum.linuxcnc.org/plasmac/-37233-plasmac-updates.

Користувачам наполегливо рекомендується створити ім’я користувача та підписатися на вищезазначену тему, щоб отримувати сповіщення про оновлення.

Для стандартної інсталяції ISO LinuxCNC буде оновлюватися тільки після виходу нової другорядної версії. QtPlasmaC автоматично оновить свою конфігурацію під час першого запуску після оновлення LinuxCNC.

LinuxCNC зазвичай оновлюється шляхом введення наступних команд у вікно терміналу (по одній):

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

10.8.12.2 Постійне оновлення

Покращення та виправлення помилок не будуть доступні в стандартній інсталяції до випуску нової незначної версії LinuxCNC. Якщо користувач бажає оновлювати систему щоразу, коли виходить нова версія QtPlasmaC, він може скористатися репозиторієм LinuxCNC Buildbot замість стандартного репозиторію LinuxCNC, дотримуючись інструкцій на сторінці http://buildbot.linuxcnc.org/ .

10.8.13 Зміна існуючої конфігурації QtPlasmaC

Існує два способи змінити існуючу конфігурацію QtPlasmaC:

1. Запуск відповідного configuration wizard та завантаження файлу .conf, збереженого майстром.
2. Вручну відредагуйте INI- та/або HAL-файл конфігурації.



Important
Будь-які ручні зміни до файлів <*machine_name*>.ini та <*machine_name*>.hal не будуть зареєстровані в PnCConf або StepConf.

Note
Якщо користувач не впевнений у повній назві виводу HAL, він може запустити LinuxCNC та виконати **HalShow** для отримання повного списку всіх виводів HAL.

10.8.14 Налаштування графічного інтерфейсу QtPlasmaC

Стилізація графічного інтерфейсу QtPlasmaC здійснюється за допомогою таблиць стилів Qt, а деякі налаштування можна виконати за допомогою власних таблиць стилів. Це дозволяє користувачев змінювати деякі елементи графічного інтерфейсу, такі як колір, межі, розмір тощо. Розміщення елементів графічного інтерфейсу змінити неможливо.

Інформація про таблиці стилів Qt доступна за посиланням тут.

Існує два способи застосування власних стилів:

1. Додати власний стиль: використовуйте це для незначних змін стилю.
2. Створити новий стиль, використовуйте це для повної зміни стилю.

10.8.14.1 Додати власний стиль

Додавання змін стилю до стандартного стильового аркуша здійснюється шляхом створення файлу в каталозі конфігурації <*machine_name*>. Цей файл ПОВИНЕН мати назву qtplasmac_custom.qss. Усі необхідні зміни стилю додаються до цього файлу.

Наприклад, користувач може захотіти, щоб напруга дуги відображалася червоним кольором, зелений світлодіод Torch On був більшого розміру, а кнопка Torch Enable була більшою. Це можна зробити за допомогою наступного коду в 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 Створіть новий стиль

Налаштування власних таблиць стилів можна ввімкнути, встановивши відповідний параметр у розділі **[GUI_OPTIONS]** файлу <*machine_name*>.prefs. Цей параметр потрібно встановити на ім'я файлу таблиці стилів, як показано нижче.

```
b''Bb''b''лb''b''ab''b''cb''b''нb''b''иb''b''йb'' b''cb''b''тb''b''иb''b''лb''b''ьb'' =  ↩
    the_cool_style.qss
```

Ім'я файлу може бути будь-яким дійсним ім'ям файлу. Стандартне розширення — .qss, але це не обов'язково.

Існують деякі обмеження щодо налаштування стилю для QtPlasmaC, наприклад, кнопки перемикання, кнопки відновлення вирізаного тексту та кнопки діалогової форми є файлами зображень і не можуть бути налаштовані.

Файл користувацького стилю потребує заголовка в такому форматі:

```
/*****************************
b''3b''b''ab''b''rb''b''ob''b''лb''b''ob''b''вb''b''ob''b''кb'' b''вb''b''лb''b''ab''b' ↩
    'cb''b''нb''b''ob''b''rb''b''ob'' b''ab''b''pb''b''кb''b''yb''b''шb''b''ab'' b''cb''b' ↩
    'тb''b''иb''b''лb''b''ib''b''вb''

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

*****************************/
```

Кольори можуть бути виражені в будь-якому допустимому форматі таблиці стилів.

Вищезазначені кольори використовуються для наступних віджетів. Тому будь-яке налаштування стилю повинно враховувати ці кольори. Кольори, показані нижче, є стандартними кольорами, що використовуються в QtPlasmaC, разом з назвами кольорів з SETTINGS Tab.

Колір	**Параметр**	**Аффект**
color1 (#ffee06)	Передній план	передній план кнопок посуву передній план кнопок користувача, що фіксуються передній план кнопок камери/лазера передній план кнопок діалогової форми фон активних кнопок діалогової форми
color2 (#16160e)	Фон	фон кнопок користувача, що фіксуються фон кнопок камери/лазера фон активної лінії редактора G-коду фон кнопок діалогової форми
color3 (#ffee06)	Виділити	фон активних кнопок користувача, що фіксуються фон активних кнопок камери/лазера курсор редактора G-коду на передньому плані
color4 (#36362e)	Альтернативний фон	фон активного рядка відображення G-коду

10.8.14.3 Повернення до стилю за замовчуванням

Користувач може повернутися до стилю за замовчуванням у будь-який час, виконавши такі дії:

1. Закрийте QtPlasmaC, якщо він відкрито.
2. Видаліть qtplasmac.qss з каталогу конфігурації машини.
3. Видаліть qtplasmac_custom.qss з каталогу конфігурації машини (якщо він існує).
4. Відкрийте файл <*ім'я_машини*>.prefs.

5. Видаліть розділ **[COLOR_OPTIONS]**.
6. Видаліть опцію «Налаштування стилю» з розділу **[GUI_OPTIONS]**.
7. Збережіть файл.

Під час наступного завантаження QtPlasmaC усі користувацькі стилі будуть видалені, і буде повернуто стиль за замовчуванням.

Нижче наведено приклад розділу та параметрів, які потрібно видалити з <*назва_машини*>.prefs:

```
[b''Пb''b''Ab''b''Pb''b''Ab''b''Mb''b''Eb''b''Tb''b''Pb''b''Ыb''_b''Kb''b''Ob''b''Лb''b' ↩
    'Ьb''b''Ob''b''Pb''b''Уb'']
b''Пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ib''b''йb'' b''пb''b''лb''b''ab''b''нb'' = # ↩
    ffee06
b''Пb''b''ib''b''дb''b''cb''b''вb''b''ib''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' ↩
     = #ffee06
b''Cb''b''вb''b''ib''b''тb''b''лb''b''ob''b''дb''b''ib''b''ob''b''дb'' = #ffee06
b''Фb''b''ob''b''нb'' = #16160e
b''Bb''b''иb''b''cb''b''ob''b''тb''b''ab'' b''фb''b''ob''b''нb''b''уb'' = #36362e
b''Kb''b''ab''b''дb''b''pb''b''иb'' = #ffee06
b''3b''b''уb''b''пb''b''иb''b''нb''b''кb''b''ab'' = #ff0000
b''Bb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob'' = #b0b0b0
b''Пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ib''b''йb'' b''пb''b''eb''b' ↩
    'pb''b''eb''b''гb''b''лb''b''яb''b''дb'' = #000000
```

10.8.14.4 Користувацький код Python

Можна додавати власний код Python, щоб змінити деякі існуючі функції або додати нові. Власний код можна додавати двома різними способами: файл команд користувача або періодичний файл користувача.

Файл команд користувача вказується в розділі DISPLAY файлу <*machine_name*>.ini та містить код Python, який обробляється лише один раз під час запуску.

```
USER_COMMAND_FILE = my_custom_code.py
```

Періодичний файл користувача повинен мати назву user_periodic.py і бути завантаженим у каталог конфігурації машини. Цей файл обробляється кожний цикл (зазвичай 100 мс) і використовується для функцій, які потребують регулярного оновлення.

10.8.14.5 Користувацький фільтр G-коду

Весь вхідний G-код аналізується фільтром G-коду, щоб переконатися, що він підходить для QtPlasmaC. Цей фільтр можна розширити за допомогою власного коду Python, що виконується з файлу в каталозі конфігурації, щоб полегшити перетворення різних варіантів G-коду у формат, придатний для QtPlasmaC.

Назва цього файлу — custom_filter.py, і він буде використаний автоматично, якщо існує.

Існує три попередньо встановлені методи, доступні для використання:

Ім’я	**Функція**
custom_pre_process	Це виконує базову обробку кожного рядка перед будь-якою обробкою у фільтрі.
custom_pre_parse	Це аналізує будь-який G-код з рядка **до** будь-якого розбору, виконаного у фільтрі.

Ім’я	Функція
custom_post_parse	Це аналізує будь-який G-код з рядка **після** будь-якого розбору, виконаного у фільтрі.

Ці методи застосовуються за такою процедурою:

- Визначте метод з аргументом для вхідних даних.
- Додайте будь-який необхідний код для маніпулювання даними.
- Поверніть результуючі дані.
- Приєднайте новий метод.

Наприклад, щоб видалити будь-який код, що починається з «G71», та змінити «M2» на «M5 $0» та «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
```

Крім того, таким же чином можна перевизначити будь-який існуючий метод у фільтрі. Для цього потрібно визначити таку саму кількість аргументів, як і в існуючому методі, враховуючи, що «self» в оригіналі не є аргументом.

```
def new_method_name(data):
    if data[:3] == 'G71':
        return(None)
    return(data)
self.old_method_name = new_method_name
```

Note
Існуючий код фільтра можна знайти у файлі /bin/qtplasmac_gcode.
Файл sim/qtplasmac/custom_filter.py містить приклад скелетного коду для налаштування фільтрації.

10.8.15 Розширені теми QtPlasmaC

10.8.15.1 Налаштовані кнопки користувача

Графічний інтерфейс QtPlasmaC пропонує користувацькі кнопки, які можна налаштувати, додавши команди в розділ USER BUTTON ENTRIES вкладки SETTINGS Tab у файлі <*machine_name*>.prefs.

Кількість кнопок користувача залежить від типу дисплея та роздільної здатності, як зазначено нижче:

- 16:9 та 4:3 - мінімум 8, максимум 20
- 9:16 - Мінімум 15, максимум 20

Користувачеві потрібно буде запустити QtPlasmaC з потрібним розміром екрана, щоб визначити, скільки кнопок користувача доступно для використання.

Усі налаштування файлу <*machine_name*>.prefs для кнопок знаходяться в розділі **[BUTTONS]**.

Назви кнопок Текст, який відображається на кнопці, встановлюється таким чином:

```
n Name = HAL Show
```

Де *n* — це номер кнопки, а **HAL Show** — це текст.

Для тексту в кілька рядків розділіть текст за допомогою \ (зворотної скісної риски):

```
n b''Ib''b''мb'''b''яb'' = HAL\Show
```

Якщо амперсанд має відображатися як текст, тоді потрібні два послідовні амперсанди:

```
n Name = PIERCE&&CUT
```

Код кнопки Кнопки можуть виконувати наступне:

1. External commands
2. External python scripts
3. G-code commands
4. Dual code
5. Toggle a HAL pin
6. Увімкнути/вимкнути вирівнювальний лазерний HAL-штифт
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. Завантажує останній змінений файл NGC
19. Показати/приховати онлайн-інструкцію користувача у форматі HTML
20. Перемикання між режимами суглоба та телеопу

Зовнішні команди

Щоб виконати зовнішню команду, перед нею ставиться символ %.

```
n b''Kb''b''ob''b''дb'' = %halshow
```

Зовнішні скрипти Python

Щоб запустити зовнішній скрипт Python, перед назвою скрипта ставиться символ %, а також потрібне розширення .py. Символ ~ можна використовувати як скорочення для домашнього каталогу користувача.

```
n Code = %~/user_script.py
```

G-код

Щоб запустити G-код, просто введіть код для запуску.

```
n Code = G0 X100
```

Щоб запустити існуючу підпрограму.

```
n Code = o<the_subroutine> call
```

<machine_name>.Змінні ini-файлу можна вводити за допомогою стандартного формату G-коду LinuxCNC. Якщо включені вирази, їх потрібно взяти в квадратні дужки.

```
n Code = G0 X#<_ini[joint_0]home> Y1
n Code = G53 G0 Z[#<_ini[axis_z]max_limit> - 1.001]
```

Змінні файлу *<machine_name>*.prefs, а також змінні *<machine_name>*.ini можна ввести, уклавши кожну опцію в **{}** . Після кожного **}** необхідно ставити пробіл, якщо за ним йдуть інші символи. Якщо вирази містять вирази, їх необхідно укласти в дужки.

```
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]
```

Кілька кодів можна виконати, розділяючи їх символом ”\” (зворотна скісну риску). Винятком є спеціальні команди, які мають бути однією командою на кнопку.

```
n Code = G0 X0 Y0 \ G1 X5 \ G1 Y5
```

Зовнішні команди та G-код можна використовувати одночасно на одній кнопці.

```
n Code = %halshow \ g0x.5y.5 \ %halmeter
```

Подвійний код

Подвійний код дозволяє виконувати два фрагменти коду по черзі з кожним натисканням кнопки. Текст кнопки чергуватиметься з кожним натисканням кнопки, а індикатор можна за бажанням увімкнути.

Обов’язково вказуйте код кнопки в такому порядку: «подвійний код», перший код, альтернативний текст кнопки та другий код, розділені подвійною крапкою з комою. Якщо потрібен індикатор, то за бажанням додайте ”;; правда” в кінці.

```
n Code = dual-code ;; code1 ;; name1 ;; code2 ;; true
```

При першому натисканні кнопки буде виконано код1, текст кнопки зміниться на ім’я1, а якщо вказано ”true”, індикатор засвітиться.

При другому натисканні кнопки буде виконано код2, текст кнопки зміниться на n Name, а індикатор згасне, якщо він світиться.

code1 і code2 відповідають правилам, описаним у попередніх поясненнях щодо коду: Зовнішні команди, Код Python та G-код. Допускається використання декількох кодів, а також їх поєднання.

Наведений нижче код дозволить користувачеві використовувати одну кнопку для почергового виконання двох фрагментів коду при кожному натисканні кнопки:

```
n Name = X+10
n Code = dual-code ;; G91\G0X10\G90 ;; X-10 ;; G91\G0X-10\G90
```

Початкова мітка буде X+10, при натисканні пальник переміститься на 10 в плюс по осі X, а мітка зміниться на X-10. При повторному натисканні пальник переміститься на 10 в мінус по осі X, а мітка зміниться на X+10.

Спеціальні команди Наступні команди повинні бути єдиною командою для кожної кнопки користувача, а код кнопки повинен починатися зі спеціальної команди. Винятком є toggle-laser, яка може з'являтися в будь-якому місці коду, як показано нижче.

Перемикання HAL Pin

Наведений нижче код дозволить користувачеві використовувати кнопку для інвертування поточного стану виводу біта HAL:

```
n Code = toggle-halpin the-hal-pin-name
```

Цей код потрібно використовувати як окрему команду, і він може керувати лише одним бітом HAL на кожну кнопку.

Кольори кнопок будуть відповідати стану виводу HAL.

Після встановлення коду при натисканні кнопка змінить кольори, а контакт HAL змінить стан контакту. Кнопка залишиться «зафіксованою» доти, доки її не натиснути знову, після чого кнопка повернеться до початкових кольорів, а контакт HAL — до початкового стану.

Користувач також може вказати альтернативний текст, який буде відображатися на кнопці, поки вона перебуває в зафіксованому стані. Щоб вказати альтернативний текст, використовуйте подвійну крапку з комою, за якою йде необхідний текст. Це повинен бути останній елемент у коді кнопки.

```
n Code = toggle-halpin the-hal-pin-name ;; PIN\TOGGLED
```

Є три External HAL Pins, які можна перемикати як вихід, імена контактів — `qtplasmac.ext_out_0`, `qtplasmac.ext_out_1` та `qtplasmac.ext_out_2`. Підключення HAL до цих контактів HAL потрібно вказати у файлі postgui HAL, оскільки контакти HAL недоступні до завантаження графічного інтерфейсу QtPlasmaC.

Для перемикаючих кнопок HAL користувач може позначити відповідний контакт HAL як такий, що повинен бути увімкнений перед початком послідовності різання, додавши «cutcritical» після контакту HAL у коді кнопки. Якщо встановлено прапорець **TORCH ENABLE** (Увімкнути пальник) і запущено **CYCLE START** (Почати цикл), **MANUAL CUT** (Ручне різання) або **SINGLE CUT** (Одноразове різання), а кнопка «cutcritical» не ввімкнена, користувач отримає діалогове вікно з попередженням і запитом про продовження (CONTINUE) або скасування (CANCEL). У діалоговому вікні буде перелічено всі неперемикані кнопки з відповідними прапорцями, і користувач зможе вибрати, які з кнопок слід перемикати автоматично після натискання CONTINUE (ПРОДОВЖИТИ).

```
n Code = b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''ab''b''чb''-halpin b''нb''b' ↩
    'ab''b''зb''b''вb''b''ab''-hal-pin cutcritical
```

Перемикач вирівнювального лазера HAL Pin

Наведений нижче код дозволить користувачеві використовувати кнопку для інвертування поточного стану виводу біта HAL лазера для вирівнювання:

```
n b''Kb''b''ob''b''дb'' = toggle-laser
```

Цей код також можна використовувати як багатокомандну команду з G-кодом або зовнішніми командами, але він може керувати лише виводом біта HAL лазера вирівнювання.

Кольори кнопок будуть відповідати стану штифта HAL вирівнювального лазера.

Після встановлення коду при натисканні кнопка змінить кольори, а вивід HAL лазера вирівнювання змінить стан виводу. Кнопка залишиться «зафіксованою» до наступного натискання, після чого вона повернеться до початкових кольорів, а вивід HAL лазера вирівнювання — до початкового стану.

Наступний код дозволить користувачеві за допомогою кнопки змінити поточний стан виводу HAL лазера вирівнювання, а потім перемістити осі X і Y на відстань зміщення лазера вирівнювання, як зазначено у файлі <*machine_name*>.prefs:

```
n Code = G0 X{LASER_OFFSET X axis} Y{LASER_OFFSET Y axis} \ toggle-laser
```

Позиція команди ”toggle-laser” не важлива, оскільки вона завжди виконується першою командою, незалежно від її позиції.

Імпульсний HAL Pin

Наведений нижче код дозволить користувачеві за допомогою кнопки імпульсно передавати біт HAL протягом 0,5 секунд:

```
n Code = pulse-halpin the-hal-pin-name 0.5
```

Цей код потрібно використовувати як окрему команду, і він може керувати лише одним бітом HAL на кожну кнопку.

Тривалість імпульсу вказується в секундах, якщо тривалість імпульсу не вказана, то за замовчуванням встановлюється одна секунда.

Кольори кнопок будуть відповідати стану виводу HAL.

Після встановлення коду, при натисканні кнопки, кнопка змінить кольори, контакт HAL змінить стан контакту, а на кнопці буде відображено час, що залишився. Колір кнопки та стан контакту залишатимуться зміненими до закінчення таймера тривалості імпульсу, після чого кнопка повернеться до початкових кольорів, контакт HAL — до початкового стану, а кнопка — до початкової назви.

Активний імпульс можна скасувати повторним натисканням кнопки.

Є три External HAL Pins, які доступні для імпульсів як вихід, імена контактів: qtplasmac.ext_out_0, qtplasmac.ext_out_1 та qtplasmac.ext_out_2. Підключення HAL до цих контактів HAL потрібно вказати у файлі postgui HAL, оскільки контакти HAL недоступні до завантаження графічного інтерфейсу QtPlasmaC.

Тест зонда

QtPlasmaC почне зондування, і коли матеріал буде виявлений, вісь Z підніметься до висоти проколу, яка в даний момент відображається в розділі MATERIAL (МАТЕРІАЛ) на вкладці PARAMETERS Tab (плазма:вкладка параметрів,вкладка PARAMETERS). Якщо користувач вибрав «Переглянути матеріал» у розділі GUI SETTINGS (Налаштування графічного інтерфейсу) вкладки SETTINGS Tab, це значення буде відображатися у верхньому лівому куті вікна PREVIEW (Попередній перегляд) поруч із **PH:**.

QtPlasmaC буде чекати в цьому стані протягом зазначеного часу (округленого до цілих чисел), перш ніж повернути вісь Z у вихідне положення. Приклад 6-секундної затримки наведено нижче. Якщо час не вказано, час зондування за замовчуванням становитиме 10 секунд.

```
n Code = probe-test 6
```

Note
Увімкнення кнопки користувача як кнопки Probe Test додасть external HAL pin, який може бути підключений від підвісного пристрою тощо. Підключення HAL до цього контакту HAL потрібно вказати у файлі postgui HAL, оскільки контакт HAL недоступний до завантаження графічного інтерфейсу QtPlasmaC.

Омічний тест

QtPlasmaC увімкне вихідний сигнал Ohmic Probe Enable, і якщо буде виявлено вхідний сигнал Ohmic Probe, на панелі SENSOR загориться світлодіодний індикатор. Основна мета цього — забезпечити можливість швидкого тестування на наявність короткого замикання в наконечнику пальника.

```
n b''Kb''b''ob''b''дb'' = ohmic-test
```

Note
Увімкнення кнопки користувача як кнопки омічного тесту додасть external HAL pin, який може бути підключений від підвісного пристрою тощо. Підключення HAL до цього контакту HAL потрібно вказати у файлі postgui HAL, оскільки контакт HAL недоступний до завантаження графічного інтерфейсу QtPlasmaC.

Тип різу

Якщо вибрано цю кнопку, вона перемикатиметься між двома режимами тип-різання, Пробиття та Різання (режим різання за замовчуванням) або Тільки пробиття.

```
n b''Kb''b''ob''b''дb'' = cut-type
```

Зміна витратних матеріалів

Натискання цієї кнопки переміщує пальник до заданих координат, коли машина перебуває в стані паузи, що дозволяє користувачеві легко змінювати витратні матеріали пальника.

Дійсні записи: X*nnn* Y*nnn* F*nnn*. Принаймні одна з координат X або Y обов'язкова, швидкість подачі (F) необов'язкова.

Координати X та Y вказані в абсолютних машинних координатах. Якщо X або Y відсутні, буде використано поточну координату для цієї осі.

Швидкість подачі (F) необов'язкова. Якщо вона відсутня або недійсна, буде використано швидкість подачі поточного матеріалу.

Існує три способи повернення до попередніх координат:

1. Знову натисніть кнопку **Змінити витратні матеріали** — пальник повернеться до початкових координат, і машина чекатиме в цьому положенні, поки користувач відновить програму.
2. Натисніть **ВІДНОВИТИ ЦИКЛ** — пальник повернеться до початкових координат, і програма продовжить роботу.
3. Натисніть **CYCLE STOP** – пальник повернеться до початкових координат, і програма перерветься.

```
n Code = change-consumables X10 Y10 F1000
```

Note
Увімкнення кнопки користувача як кнопки «Змінити витратні матеріали» додасть external HAL pin, який може бути підключений від підвісного пристрою тощо. Підключення HAL до цього контакту HAL потрібно вказати у файлі postgui HAL, оскільки контакт HAL недоступний до завантаження графічного інтерфейсу QtPlasmaC.

Навантаження

Завантаження G-кодової програми з каталогу, зазначеного змінною **PROGRAM_PREFIX** у файлі <*machine_name*>.ini (зазвичай ~/linuxcnc/nc_files), можливе за допомогою такого формату:

```
n Code = load G-code.ngc
```

Якщо файл G-коду користувача знаходиться в підкаталозі каталогу **PROGRAM_PREFIX**, доступ до нього можна отримати, додавши ім'я підкаталогу на початку імені файлу G-коду. Приклад для підкаталогу з ім'ям **plasma**:

```
n Code = load plasma/G-code.ngc
```

Зверніть увагу, що перший символ "/" не є обов'язковим, оскільки він буде додано автоматично.

Імпульс факела

Увімкніть пальник на заданий час. Час повинен бути вказаний в секундах з точністю до одного десяткового знака. Максимально допустимий час становить 3 секунди, будь-яке значення, вказане вище цього значення, буде обмежено 3 секундами. Приклад імпульсу тривалістю 0,5 секунди наведено нижче. Якщо час не вказано, за замовчуванням буде встановлено 1 секунду. Час імпульсу з більш ніж одним десятковим знаком буде округлено до одного десяткового знака.

Повторне натискання кнопки під час зворотного відліку призведе до вимкнення ліхтарика, як і натискання клавіші «Esc», якщо комбінації клавіш увімкнено у вкладці НАЛАШТУВАННЯ.

Якщо кнопка буде відпущена до завершення відліку, ліхтарик вимкнеться після завершення відліку. Якщо кнопка буде утримуватися до завершення відліку, ліхтарик залишиться увімкненим доти, доки кнопка не буде відпущена.

```
n Code = torch-pulse 0.5
```

Note
Увімкнення кнопки користувача як кнопки Torch Pulse додасть external HAL pin, який може бути підключений від підвісного пристрою тощо. Підключення HAL до цього контакту HAL потрібно вказати у файлі postgui HAL, оскільки контакт HAL недоступний до завантаження графічного інтерфейсу QtPlasmaC.

Одинарний розріз

Виконайте одинарний різ. При цьому використовується автоматична функція одинарний різ.

```
n b''Kb''b''ob''b''дb'' = single-cut
```

Обрамлення

Обрамлення — це можливість переміщувати пальник по периметру прямокутника, який охоплює межі поточного завдання.

Контакт HAL для увімкнення лазера (qtplasmac.laser_on) буде увімкнено під час переміщення кадру, а будь-які зміщення X/Y для лазерної указки у файлі <*machine_name*>.prefs також будуть застосовані до руху X/Y. Після завершення руху кадрування пальник переміститься в положення X0 Y0, щоб очистити всі застосовані зміщення лазера, а qtplasmac.laser_on буде вимкнено.

Під час запуску циклу кадрування важливо зазначити, що за замовчуванням вісь Z буде переміщена на висоту [AXIS_Z]MAX_LIMIT - 5 мм (0,2 дюйма) перед початком руху X/Y.

Швидкість рухів XY фреймінгового руху можна задати таким чином, щоб фреймінговий рух завжди відбувався із заданою швидкістю. Це можна зробити, додавши швидкість подачі (F) як останню частину коду кнопки. Якщо швидкість подачі не вказана в коді кнопки, швидкість фреймінгового руху за замовчуванням буде дорівнювати швидкості подачі для поточного вибраного матеріалу.

Наступні кнопки графічного інтерфейсу та комбінації клавіш (якщо їх увімкнено у вкладці НАЛАШТУВ
дійсні під час кадрування руху:

1. Натискання **CYCLE STOP** або ESC keyboard shortcut - Зупиняє рух кадрування.
2. Натискання **ЦИКЛ ПАУЗА** або ПРОБІЛ keyboard shortcut - Призупиняє рух кадрування.
3. Натискання **CYCLE RESUME** або CTRL+r keyboard shortcut - відновлює призупинений рух кадрування.
4. Зміна **ПОВЗУНЦЯ ПОДАЧІ** або будь-якого з CTRL+0-9 keyboard shortcuts - Уповільнює швидкість подачі.

Note
ЯКЩО ШВИДКІСТЬ ПОДАЧІ ЗМІНЮЄТЬСЯ ДЛЯ РУХУ КАДРУВАННЯ, ПЕРЕД НАТИСКАННЯМ КНОПКИ «ПУСК ЦИКЛУ» ТА РІЗАННЯМ ЗАВАНТАЖЕНОГО ЗАВДАННЯ ПОТРІБНО ПОВЕРНУТИ ПОВЗУНЕЦЬ ПОДАЧІ У ПОЛОЖЕННЯ 100%.

```
n b’’Kb’’b’’ob’’b’’дb’’ = framing
```

Користувач може пропустити початковий рух Z за замовчуванням та запустити послідовність кадрування на поточній висоті Z, додавши ”usecurrentzheight” після ”framing”.

```
n Code = framing usecurrentzheight
```

Щоб вказати швидкість подачі:

```
n Code = framing F100
```

або:

```
n Code = framing usecurrentzheight F100
```

Увімкнення кнопки користувача як кнопки кадрування додасть external HAL pin, який може бути підключений від підвісного пристрою тощо. Підключення HAL до цього контакту HAL потрібно вказати у файлі postgui HAL, оскільки контакт HAL недоступний до завантаження графічного інтерфейсу QtPlasmaC.

Ручне різання

Ручне різання функціонує ідентично кнопці **F9** для початку або завершення ручного різання.

```
n b’’Kb’’b’’ob’’b’’дb’’ = manual-cut
```

Переглядач зміщень

Це дозволяє показувати/приховувати екран перегляду зміщень, на якому відображаються всі зміщення верстата. Всі відносні зміщення можна редагувати, а координатам робочої системи G54 ~ G59.3 можна присвоювати власні імена.

```
n b’’Kb’’b’’ob’’b’’дb’’ = offsets-view
```

Завантажити останній файл

Це дозволяє завантажувати останній змінений файл у каталозі. Ім’я каталогу необов’язкове, і якщо його пропустити, за замовчуванням буде використано останній каталог, з якого було завантажено файл.

```
n Code = latest-file /home/me/linuxcnc/nc_files/qtplasmac-test
```

Посібник користувача

Це дозволяє показувати/приховувати онлайн-інструкцію користувача у форматі HTML для поточної версії LinuxCNC. Зверніть увагу, що для цієї функції потрібен доступ до Інтернету.

```
n Code = user-manual
```

Перемикання спільного режиму

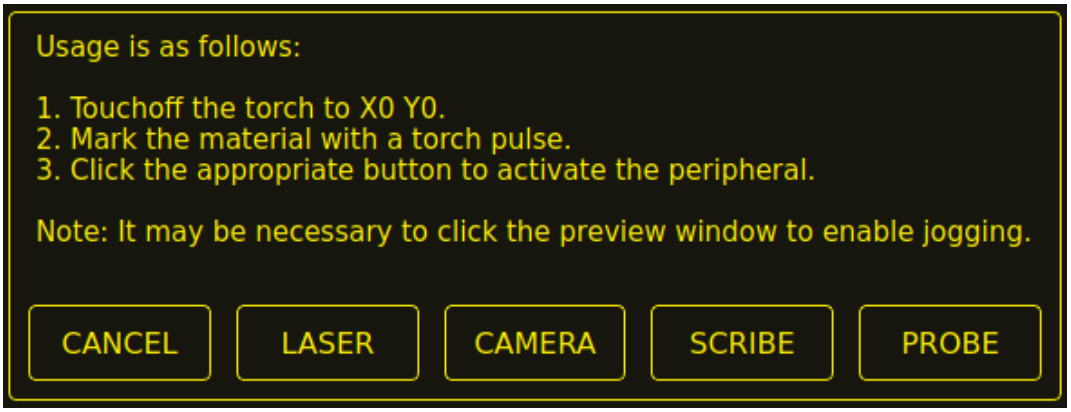
Це дозволяє перемикатися між режимом суглоба та режимом телеоп. Щоб ця кнопка була активною, машина має бути увімкнена та переведена в початкове положення.

```
n Code = toggle-joint
```

10.8.15.2 Периферійні зміщення (лазер, камера, розмітка, зонд зміщення)

Використовуйте наступну послідовність для встановлення зміщень для лазера, камери, розмітки або зонда зміщення:

1. Покладіть шматок обрізків матеріалу під пальник.
2. Перш ніж продовжувати, машина повинна бути в головному стані та в режимі очікування.
3. Відкрийте вкладку НАЛАШТУВАННЯ.
4. Натисніть кнопку «ВСТАНОВИТИ ЗМІЩЕННЯ», щоб відкрити діалогове вікно «Встановити периферійні зміщення».



5. Натисніть кнопку X0Y0, щоб встановити положення пальника на нуль.
6. Зробіть позначку на матеріалі одним із таких способів:
   a. Зменште налаштування пальника до потрібної висоти проколу, а потім увімкніть імпульсний струм пальника, щоб створити заглиблення в матеріалі.
   b. Нанесіть маркувальний барвник на екран пальника, потім поштовхом опустіть пальник, щоб позначити матеріал.
7. Натисніть відповідну кнопку, щоб активувати периферійний пристрій.
8. Тепер з’явиться діалогове вікно «Отримати периферійні зміщення».



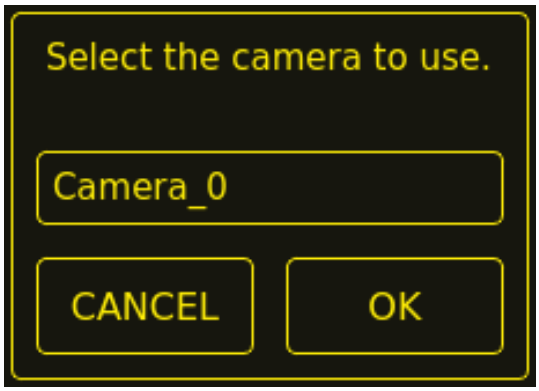
9. Підніміть вісь Z так, щоб пальник та периферійне обладнання не торкалися матеріалу.
10. Пересуньте осі X/Y так, щоб периферійний пристрій був по центру позначки пальника.
11. Натисніть кнопку «ОТРИМАТИ ЗМІЩЕННЯ», щоб отримати зміщення, і відкриється діалогове вікно підтвердження.



12. Натисніть кнопку «ВСТАНОВИТИ ЗМІЩЕННЯ», і зміщення тепер будуть збережені.

Скасування можна здійснити на будь-якому етапі, натиснувши кнопку СКАСУВАТИ, яка закриє діалогове вікно, і жодні зміни не будуть збережені.

Якщо в пункті 7 вище було вибрано CAMERA і існує більше однієї камери, з’явиться діалогове вікно вибору камери. Необхідно вибрати відповідну камеру, перш ніж з’явиться діалогове вікно Get Peripheral Offsets.



Якщо у пункті 7 вище було вибрано PROBE (ЗОНД), то перед діалоговим вікном підтвердження у пункті 11 з’явиться діалогове вікно затримки. Це вікно встановлюється для затримки, необхідної для розгортання зонда в робоче положення.



Note
Можливо, знадобиться клацнути вікно попереднього перегляду, щоб увімкнути поштовх. Дотримуючись вищезазначеної процедури, зміщення будуть доступні для використання негайно, і перезапуск LinuxCNC не потрібен.

10.8.15.3 Зберігайте рух Z

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 Зовнішні контакти HAL

QtPlasmaC створює деякі HAL-піни, які можна використовувати для підключення зовнішньої кнопки або брелока тощо.

З’єднання HAL з цими контактами HAL потрібно вказати у файлі HAL postgui, оскільки контакти HAL недоступні, доки не завантажиться графічний інтерфейс QtPlasmaC.

Наступні піни HAL-бітів створюються завжди. Пін HAL має ідентичну поведінку, як і відповідна кнопка графічного інтерфейсу QtPlasmaC.

Функція кнопки користувача	**HAL Pin**	**Функція графічного інтерфейсу**
Перемикач живлення машини	qtplasmac.ext_power	ПОТУЖНІСТЬ
Запустіть завантажену програму G-коду	qtplasmac.ext_run	ПОЧАТОК ЦИКЛУ
Призупинити/відновити завантажену програму G-коду	qtplasmac.ext_pause	ПАУЗА/ВІДНОВЛЕННЯ ЦИКЛУ
Призупинити завантажену програму G-коду	qtplasmac.ext_pause_only	ЦИКЛ ПАУЗА
Відновіть завантажену програму G-коду	qtplasmac.ext_resume	РЕЗЮМЕ ЦИКЛУ
Перервати завантажену програму G-коду	qtplasmac.ext_abort	ЗУПИНКА ВЕЛОСИПЕДУ
Доторкання до осей X та Y до нуля	qtplasmac.ext_touchoff	X0Y0
Використовуйте лазер для встановлення початку координат з обертанням або без нього	qtplasmac.ext_laser_touchoff	ЛАЗЕР
Увімкнути/вимкнути закріплення qtplasmac.laser_on	qtplasmac.ext_laser_toggle	N/A
Запуск/Пауза/Відновлення завантаженої програми G-коду	qtplasmac.ext_run_pause	ПУСК ЦИКЛУ, ПАУЗА ЦИКЛУ, ВІДНОВЛЕННЯ ЦИКЛУ послідовно
Коригування висоти пальника плюс	qtplasmac.ext_height_ad_plus	ЗАМІНИТИ
Мінус коригування висоти пальника	qtplasmac.ext_height_ad_minus	ЗАМІНИТИ -
Скидання корекції висоти пальника	qtplasmac.ext_height_ad_reset	ЗАМІНА СКИДАННЯ ДО 0.00
Шкала корекції висоти пальника	qtplasmac.ext_height_ad_scale	N/A
Перемикання швидкості бігу між швидкою та повільною	qtplasmac.ext_jog_slow	БІГ ТРУБОМ ШВИДКО/ПОВІЛЬНО
Увімкнути/вимкнути THC	qtplasmac.ext_thc_enable	УВІМКНУТИ THC
Увімкнути/вимкнути ліхтарик	qtplasmac.ext_torch_enable	УВІМК. ФАКЕЛ
Увімкнути/вимкнути блокування кута	qtplasmac.ext_cornerlock_enable	УВІМКНЕННЯ ЗАХИСТУ ВІД ЗАНЯТТЯ ПРОТИ ЗАНУРЮВАННЯ
Увімкнути/вимкнути блокування порожнечі	qtplasmac.ext_voidlock_enable	УВІМКНЕННЯ ЗАХИСТУ ВІД ЗАНУРЮВАННЯ VOID
Перемикач використання автоматичного налаштування напруги	qtplasmac.ext_auto_volts_enable	АВТОВОЛЬТИ
Увімкнення/вимкнення омічного зонда	qtplasmac.ext_ohmic_probe_enable	ОМІЧНЕ УВІМКНЕННЯ
Перемикання режиму сітки	qtplasmac.ext_mesh_mode	РЕЖИМ СІТКИ
Ігнорування дуги/ОК	qtplasmac.ext_ignore_arc_ok	ІГНОРУВАТИ ОК
Вперед за запрограмованим шляхом	qtplasmac.ext_cutrec_fwd	ВІДНОВЛЕННЯ РІЗАННЯ ВПЕРЕД
Зворотний рух вздовж запрограмованого шляху	qtplasmac.ext_cutrec_rev	ВІДНОВЛЕННЯ РІЗАННЯ REV
Скасувати будь-який рух відновлення від розрізів	qtplasmac.ext_cutrec_cancel	РІЗАТИ ВІДНОВЛЕННЯ СКАСУВАТИ ПЕРЕМІЩЕННЯ
Перемістити вгору	qtplasmac.ext_cutrec_up	Стрілка вгору для відновлення після розрізання

Функція кнопки користувача	**HAL Pin**	**Функція графічного інтерфейсу**
Перемістити вниз	qtplasmac.ext_cutrec_s	Стрілка вниз для відновлення після розрізання
Рухатися праворуч	qtplasmac.ext_cutrec_e	Стрілка праворуч для відновлення після розрізу
Рухатися ліворуч	qtplasmac.ext_cutrec_w	Стрілка ліворуч для відновлення після розрізу
Перемістити вгору-вправо	qtplasmac.ext_cutrec_ne	Стрілка вгору праворуч для відновлення після розрізу
Перемістити вгору-ліворуч	qtplasmac.ext_cutrec_nw	Стрілка ВІДНОВЛЕННЯ РІЗІВ ліворуч угору
Перемістити вниз праворуч	qtplasmac.ext_cutrec_se	Стрілка вниз праворуч для відновлення після розрізу
Перемістити вниз ліворуч	qtplasmac.ext_cutrec_sw	Стрілка вниз ліворуч для відновлення після розрізу

Наступні виводи HAL, які дозволяють використовувати MPG для керування коригуванням висоти, завжди створюються.

Функція	**HAL Pin**
Увімкнути керування висотою MPG	qtplasmac.ext_height_ovr_count_enable
Зміна висоти MPG	qtplasmac.ext_height_ovr_counts

Наступні бітові контакти HAL створюються тільки в тому випадку, якщо функція вказана в custom user button. Контакт HAL має таку саму поведінку, як і відповідна кнопка користувача.

Функція кнопки користувача	**HAL Pin**
Тест зонда	qtplasmac.ext_probe
Імпульс факела	qtplasmac.ext_pulse
Омічний тест	qtplasmac.ext_ohmic
Зміна витратних матеріалів	qtplasmac.ext_consumables
Обрамлення	qtplasmac.ext_frame_job

Наступні виводи бітів HAL завжди створюються і можуть використовуватися користувацькими кнопками Toggle HAL Pin або Pulse HAL Pin для зміни стану виходу.

HAL Pin
qtplasmac.ext_out_0
qtplasmac.ext_out_1
qtplasmac.ext_out_2

10.8.15.5 Приховати кнопки програм

Якщо користувач має зовнішні кнопки та/або підвісний пульт, що імітує будь-яку з програмних кнопок CYCLE START, CYCLE PAUSE або CYCLE STOP, то можна приховати будь-яку або всі ці програмні кнопки графічного інтерфейсу, додавши наступні опції до розділу **[GUI_OPTIONS]** файлу <*machine_name*>.prefs:

```
b''Пb''b''pb''b''иb''b''xb''b''ob''b''вb''b''ab''b''тb''b''иb'' b''зb''b''ab''b''пb''b' ↩
    'yb''b''cb''b''кb'' = True
b''Пb''b''pb''b''иb''b''xb''b''ob''b''вb''b''ab''b''тb''b''иb'' b''пb''b''ab''b''yb''b' ↩
    'зb''b''yb'' = True
b''Пb''b''pb''b''иb''b''xb''b''ob''b''вb''b''ab''b''тb''b''иb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''pb''b''иb''b''вb''b''ab''b''нb''b''нb''b''яb'' = True
```

Для графічних інтерфейсів зі співвідношенням сторін 16:9 або 4:3 приховування кожної з цих кнопок графічного інтерфейсу призведе до появи ще двох кнопок користувача в графічному інтерфейсі.

10.8.15.6 Режим налаштування 0 Дуга OK

Режим 0 Arc OK використовує напругу дуги для встановлення сигналу Arc OK. Це досягається шляхом вибірки напруги дуги кожний цикл сервоприводу. Для встановлення сигналу Arc OK необхідно мати певну кількість послідовних зразків, які повинні знаходитися в межах заданого порогу. Ці напруги також повинні знаходитися в заданому діапазоні.

У вкладці PARAMETERS є два налаштування для встановлення діапазону:

- **OK High Volts** – це верхнє значення діапазону напруги. Значення за замовчуванням – 250 В.
- **OK Low Volts** – нижнє значення діапазону напруги. Значення за замовчуванням – 60 В.

Обидва ці значення можна змінити шляхом безпосереднього введення або за допомогою кнопок збільшення/зменшення.

Також передбачено два виводи HAL, які дозволяють користувачеві налаштовувати задане значення. Ці виводи HAL:

- `plasmac.arc-ok-counts` що є кількістю послідовних показників у межах порогового значення, необхідних для встановлення сигналу «Дуга в порядку». Значення за замовчуванням — 10.
- `plasmac.arc-ok-threshold` що є максимальним відхиленням напруги, дозволеним для дійсної напруги, що встановлює сигнал «Дуга в порядку». Значення за замовчуванням — 10.

У наступному прикладі кількість необхідних послідовних показників буде встановлено на 6:

```
setp plasmac.arc-ok-counts 6
```

Ці налаштування, якщо вони використовуються, мають бути у файлі конфігурації custom.hal.

10.8.15.7 Затримка втрати дуги

Деякі джерела живлення плазми/конфігурації верстатів можуть втратити сигнал Arc OK або тимчасово під час різання, або назавжди ближче до кінця різання, що змушує QtPlasmaC призупинити програму і повідомити про помилку «втрачено дійсну дугу».

Існує контакт HAL під назвою «plasmac.arc-lost-delay», який можна використовувати для встановлення затримки (у секундах), що запобігатиме призупиненню програми/помилці, якщо втрачений сигнал Arc OK буде відновлено або команда **M5** буде виконана до закінчення встановленого періоду затримки.

Важливо зазначити, що THC буде вимкнено та заблоковано на висоті різання, яка була встановлена на момент втрати сигналу Arc OK.

Наступний код встановить затримку 0,1 секунди:

```
setp plasmac.arc-lost-delay 0.1
```

Рекомендується, щоб користувач встановив цей PIN-код у файлі custom.hal.

Це налаштування слід використовувати лише у випадку, якщо користувач стикається з вищезазначеними симптомами. Слід також зазначити, що користувач може використовувати відповідні команди G-коду Ignore Arc OK для досягнення аналогічного результату.

10.8.15.8 Нульове вікно

Невеликі коливання напруги дуги, що відображається під час роботи машини в режимі холостого ходу, можливі залежно від багатьох різних змінних (електричний шум, неправильне налаштування THCAD тощо).

Після усунення всіх факторів, що сприяють цьому, якщо невелике коливання все ще існує, його можна усунути, розширивши вікно напруги, для якого QtPlasmaC відображатиме 0 В.

PIN-код для налаштування цього значення називається plasmac.zero-window, а значення за замовчуван встановлено на 0.1. Щоб змінити це значення, додайте PIN-код та потрібне значення до файлу custom.hal.

У наступному прикладі вікно напруги буде встановлено на відображення як 0 В від -5 В до +5 В:

```
setp plasmac.zero-window 5
```

10.8.15.9 Налаштування зондування порожнечі

Окрім налаштування **Void Slope** у PARAMETERS Tab, є два виводи HAL для точного налаштування захисту від занурення. Ці виводи HAL:

- **plasmac.void-on-cycles** що є кількістю перевищень швидкості нахилу, щоб активувати функцію проти занурення без можливості занурення. Значення за замовчуванням — 2.
- **plasmac.void-off-cycles** – кількість циклів без перевищення швидкості нахилу, що призводить до деактивації функції захисту від занурення. Значення за замовчуванням – 10.

У наступному прикладі необхідно встановити кількість циклів увімкнення на 3:

```
setp plasmac.void-on-cycles 3
```

Мета полягає в тому, щоб отримати якомога нижче значення Void Slope без помилкових спрацьовувань а потім відрегулювати цикли ввімкнення та вимкнення, щоб забезпечити чітке ввімкнення та вимкнення функції запобігання зануренню. У більшості випадків не потрібно змінювати цикли ввімкнення та вимкнення від значення за замовчуванням.

Ці налаштування, якщо вони використовуються, мають бути у файлі конфігурації custom.hal.

10.8.15.10 Максимальне зміщення

Максимальне зміщення (Max Offset) – це відстань (у міліметрах) від Z MAX_LIMIT, на яку QtPlasmaC дозволить переміщення осі Z під керуванням машини.

Контакт для регулювання цього значення називається plasmac.max-offset, а його значення за замовчуванням (у міліметрах) встановлено на 5. Щоб змінити це значення, додайте контакт і необхідне значення до файлу custom.hal. Не рекомендується використовувати значення менше 5 мм, оскільки перевищення зміщення може спричинити непередбачувані проблеми.

У наступному прикладі відстань від Z MAX_LIMIT буде встановлено на 10 мм:

```
setp plasmac.max-offset 10
```

10.8.15.11 Увімкнути вкладки під час автоматизованого руху

За замовчуванням, всі вкладки, крім MAIN Tab, вимкнені під час автоматичного переміщення. Можна ввімкнути всі вкладки, крім CONVERSATIONAL Tab, під час автоматичного переміщення, встановивши наступний HAL-пін True:

```
setp qtplasmac.tabs_always_enabled 1
```



Warning
Оператор несе відповідальність за те, щоб машина була обладнана відповідним, справним апаратним аварійним вимикачем. Якщо для навігації по графічному інтерфейсу QtPlasmaC використовується тільки сенсорний екран, зупинити автоматичний рух машини можна тільки на вкладці MAIN (ГОЛОВНА).

10.8.15.12 Блокування поштовхового руху за допомогою Z+ Jog

Можна скасувати блокування поштовхового переміщення, використовуючи графічний інтерфейс користувача або клавіатуру для поштовхового переміщення в напрямку Z+, замість того, щоб встановлювати прапорець «Скасувати поштовхове переміщення» на вкладці SETTINGS.

Це робиться шляхом зміни наступного параметра на **True** у **[GUI_OPTIONS]** файлу <*machine_name*>.p
у папці <*machine_name*>:
`Перезаблокувати поштовховий хід через Z+`

10.8.15.13 Виходи стану QtPlasmaC

Компонент plasmac HAL має контакт HAL з назвою **plasmac.state-out**, який можна використовувати для взаємодії з компонентами, запрограмованими користувачем, щоб отримати інформацію про поточний стан компонента.

Table 10.47: Різні стани, з якими може зіткнутися QtPlasmaC

Штат	Ім’я	Опис
0	IDLE	простоює та очікує команди запуску
1	PROBE_HEIGHT	переміститися вниз до висоти зонда
2	PROBE_DOWN	зонд вниз, доки не відчує матеріал
3	PROBE_UP	переміщуйте зонд вгору, доки матеріал не перестане виявлятися, це встановлює нульову висоту
4	ZERO_HEIGHT	наразі не використовується
5	PIERCE_HEIGHT	піднятися на висоту проколу
6	TORCH_ON	увімкніть ліхтарик
7	ARC_OK	зачекайте, поки дуга не буде виявлена в порядку
8	PIERCE_DELAY	зачекайте час затримки проколювання
9	PUDDLE_JUMP	починається рух xy, перемістіться на висоту стрибка калюжі
10	CUT_HEIGHT	перемістити на висоту зрізу
11	CUT_MODE_01	різання в режимі 0 або режимі 1
12	CUT_MODE_2	різання в режимі 2
13	PAUSE_AT_END	пауза руху в кінці різання
14	SAFE_HEIGHT	переміститися на безпечну висоту
15	MAX_HEIGHT	переміститися на максимальну висоту
16	END_CUT	завершити поточне скорочення
17	END_JOB	завершити поточну роботу

Table 10.47: (continued)

Штат	Ім’я	Опис
18	TORCHPULSE	імпульс пальника активний
19	PAUSED_MOTION	рух відновлення зрізу активний під час паузи
20	OHMIC_TEST	активний омічний тест
21	PROBE_TEST	тест зонда активний
22	РИСКИ	завдання списування активне
23	CONSUMABLE_CHANGE_ON	перейти до координат зміни витратних матеріалів
24	CONSUMABLE_CHANGE_OFF	повернення з координат зміни витратних матеріалів
25	CUT_RECOVERY_ON	відновлення зрізу активне
26	CUT_RECOVERY_OFF	відновлення зрізу деактивовано

Стан DEBUG призначений лише для тестування і зазвичай не зустрічається.

10.8.15.14 Налагодження QtPlasmaC Друк

Компонент HAL plasmac має вивід HAL під назвою **plasmac.debug-print**, який, якщо встановити значення 1 (true), виводитиме на термінал кожну зміну стану як засіб налагодження.

10.8.15.15 Комунікації Hypertherm PowerMax

Зв’язок можна встановити з плазмовим різаком Hypertherm PowerMax, який має порт RS485. Ця функція дозволяє автоматично встановлювати **Режим різання**, **Силу струму різання** та **Тиск газу** з **Параметрів різання** файлу матеріалу. Крім того, користувач зможе переглянути **Час роботи дуги** PowerMax у форматі hh:mm:ss на вкладці Вкладка СТАТИСТИКА.

Якщо **Тиск газу** встановлено на нуль, то PowerMax автоматично розрахує необхідний тиск на основі **Режиму різання**, **Струму різання**, типу пальника та довжини пальника.

Зміна режиму різання встановить тиск газу до нуля, що призведе до використання автоматичного режиму регулювання тиску газу.

Максимальні та мінімальні значення цих параметрів зчитуються з плазмового різака, а відповідні кнопки в меню «Параметри різання» обмежуються цими значеннями. Тиск газу не можна змінювати з нуля, поки не буде встановлено зв’язок.

Ця функція вмикається шляхом встановлення правильного імені порту для опції PM_PORT у розділі **[POWERMAX]** файлу <*machine_name*>.prefs. Якщо опція PM_PORT не встановлена у файлі <*machine_name*>.prefs, віджети, пов’язані з цією функцією, не будуть видимими.

Приклад увімкнення комунікацій Hypertherm PowerMax на USB0:

```
[POWERMAX]
Port = /dev/ttyusb0
```

Якщо користувач не впевнений у назві порту, у каталозі конфігурації є скрипт Python, який покаже всі доступні порти і який також можна використовувати для тестування зв’язку з плазмовим блоком перед увімкненням цієї функції в графічному інтерфейсі QtPlasmaC.

Щоб скористатися тестовим скриптом, виконайте такі інструкції:

Для встановлення пакета (Buildbot) введіть таку команду у вікні терміналу:

```
pmx485-test
```

Для встановлення на місці введіть такі дві команди у вікні терміналу:

```
source ~/linuxcnc-dev/scripts/rip-environment
pmx485-test
```

Одиниця виміру тиску газу (psi або bar) визначається даними, отриманими під час початкового налаштування каналу зв'язку, і потім відображається поруч із налаштуванням тиску газу в розділі «МАТЕРІАЛ» на вкладці «плазма:параметри», вкладка «ПАРАМЕТРИ».

Після встановлення зв'язку машина PowerMax перейде в дистанційний режим і на цьому етапі її можна буде керувати тільки дистанційно (через графічний інтерфейс QtPlasmaC). Зв'язок можна перевірити, спостерігаючи за дисплеєм PowerMax.

Щоб перемкнути PowerMax назад у локальний режим, користувач може зробити це одним із таких дій:

1. Вимкніть зв'язок PowerMax з вкладка ОСНОВНА
2. Закрийте LinuxCNC, що переведе PowerMax у локальний режим під час вимкнення.
3. Вимкніть PowerMax на 30 секунд, а потім знову увімкніть його.

Tip
Якщо зв'язок PowerMax активний, то вибір Mesh Mode автоматично вибере режим CPA на блоці PowerMax.

Note
Щоб використовувати функцію зв'язку PowerMax, необхідно встановити модуль Python pyserial. Якщо pyserial не встановлено, буде відображено повідомлення про помилку.

Щоб встановити pyserial, введіть таку команду у вікно терміналу:

```
sudo apt install python3-serial
```

Типова схема з'єднань connection, а також підтверджені робочі інтерфейси, наведені в додатку до цього документа.

10.8.15.16 Рухомий Пірс

Рухомий прокол дозволяє пальнику рухатися під час періоду затримки проколу. Це має ту перевагу, що дозволяє проколювати більш товсті матеріали, ніж це можливо при стаціонарному проколі. Це також може сприяти збільшенню терміну служби витратних матеріалів, оскільки такий тип руху допомагає запобігти розбризкуванню розплавленого матеріалу в сопло пальника.

Завдяки використанню M159 можна налаштувати рухоме проколювання.

Синтаксис команди M159 такий:

```
M159 Pn Qn
```

Код дії (P)	**Дія**	**Опис**	**Значення (Q)**
601	Тип проколу	0=Нормальний, 1=Хибкий, 2=Нахил	0,1,2

Код дії (P)	**Дія**	**Опис**	**Значення (Q)**
602	Затримка руху Пірса	Затримка перед початком руху по осі Z до кінцевої висоти прожига. Виражається у відсотках від затримки прожига.	Ціле число від 0 до 100
603	Висота кінця проколу	Цільова висота прожигання в кінці затримки прожигання. Зазвичай нижча за висоту прожигання. Виражається в одиницях вимірювання.	Float
604	Затримка висоти зрізу	Затримка в кінці переходу до висоти кінця проколу перед переходом до висоти різання. Виражається в секундах.	Float
605	Швидкість різьблення	Швидкість довбання. Виражається в машинних одиницях/хв.	Float
606	Відстань до виїмки	Довжина виїмки. Виражається в машинних одиницях.	Float
607	Повзуча швидкість	Швидкість повзучості, яка починає діяти після завершення строжки. Виражається в машинних одиницях.	Float
608	Відстань повзучості	Довжина повзучості. Виражається в машинних одиницях.	Float
609	Скинути	Скидає значення для кодів дій 601-608 назад до 0, повертаючись до поведінки за замовчуванням.	Не обов’язково

Доступні моделі рухомого проколювання такі:

Підтримувана модель така сама, як і та, що створена за допомогою функції «wiggle pierce» (хитання при пробиванні) програми Sheetcam. За умови прямого введення до основного різання, функція «wiggle pierce» має рухатися вперед і назад на певну відстань уздовж введення. Строго кажучи, цей прямий рух є довільним. Технічно під час затримки пробивання доступний будь-який рух по осях X/Y, і його програмування залежить від інструменту CAM або користувача.

Обмеження полягає в тому, що цей рух повинен бути завершений протягом часу затримки пробивання. Якщо це не відбудеться, то пальник перейде до нормальної висоти різання після завершення затримки пробивання і, можливо, до завершення руху коливання.

Отже, довжину коливання та швидкість подачі необхідно враховувати при розрахунку затримки прожига або розміру коливання, обмеженого швидкістю подачі та затримкою прожига.

Наприклад:

- Швидкість подачі 1080 мм/хв (18 мм/с).
- Коливальний рух 4 мм за 3 коливання.

Це означає, що довжина коливання становить 4 x 3 = 12 мм. При швидкості подачі 18 мм/с затримка прожигання повинна становити приблизно 0,7 секунди, щоб підтримувати відстань коливання на висоті прожигання.

G-код, необхідний для виклику цієї поведінки:

```
M159 P601 Q1
```

G-код, необхідний для скидання до стандартної поведінки:

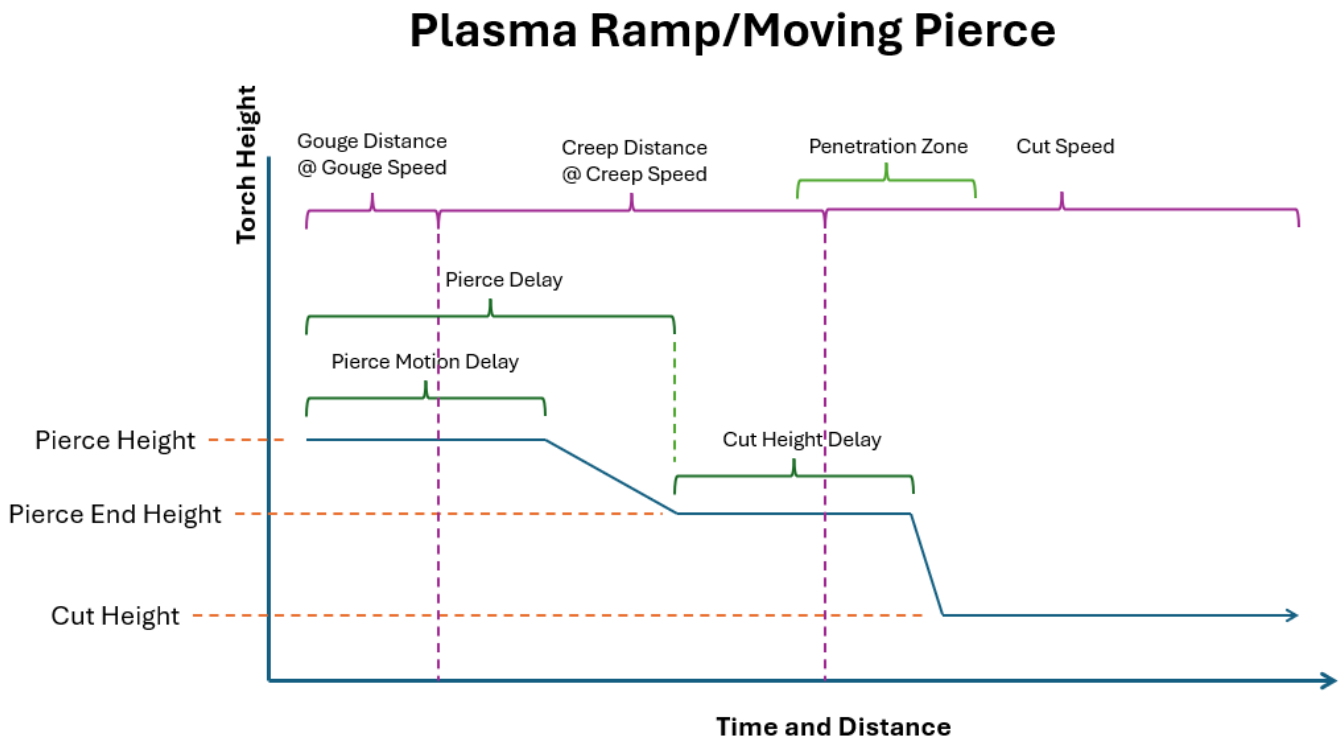
```
M159 P609
```

Пробивання з нахилом поєднує в собі ряд параметрів, що дозволяють створити похилий жолоб, який сприяє відведенню розплавленого матеріалу. Результат відведення матеріалу іноді порівнюють з «хвостом півня», оскільки він має чітко виражену спрямованість. Ретельне планування вхідної частини дозволяє відвести матеріал у безпечному для працівників та компонентів машини напрямку.

Оскільки всі елементи впливають на форму пробивання рампи, при проектуванні пробивання рампи та параметрів, що його визначають, необхідно ретельно враховувати всі ці елементи. Неминуче доведеться провести експерименти, щоб створити рецепти, які працюють з джерелом плазмової енергії, що використовується разом з матеріалом, який потрібно розрізати.

Елементи, які слід врахувати:

1. Швидкості та відстані строгання та повзучості залежно від суми затримки прожигання від матеріалу та дії затримки висоти різання.
2. Висота прожигання з матеріалу та дія висоти кінцевого прожигання стосовно затримки прожигання з матеріалу та швидкостей, що діють на різних відстанях протягом часу затримки прожигання.
3. Таблиці різання виробника плазмового джерела для типу та товщини матеріалу.



Як і у випадку з поворотним прожиганням, програмування прожигання з нахилом залежить від інструменту CAM або користувача.

Нижче наведено приклад коду для налаштування проколу рампи:

```
(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
```

Цей код показує нам таку інформацію по порядку:

1. Коментар щодо матеріальної магії.
   - Висота проколу становить 4 мм.
   - Затримка проколу становить 1 секунду.
   - Висота зрізу становить 1,5 мм.
   - Швидкість подачі на різання становить 490 мм/хв.
2. Режим встановлено на 2, що є прорізанням по рампі.
3. Затримка руху пронизування становить 50% від затримки пронизування (0,5 секунди).
4. Висота проколу 2,5 мм
5. Затримка висоти скошування становить 1 секунду.
6. Швидкість строжки становить 980 мм/хв.
7. Відстань між строжками становить 5 мм
8. Швидкість повзучого руху становить 245 мм/хв.
9. Відстань повзучості становить 3 мм.

За допомогою цієї інформації можна описати таку поведінку:

Важливо зазначити, що прискорення та уповільнення не враховуються в наступних розрахунках.

Пальник почне рух у точці проколу (на відстані 4 мм від матеріалу) зі швидкістю 980 мм/хв (16,3 мм/с) на відстань 5 мм, що займе 0,3 секунди (5 мм / 16,3 мм/с = 0,3 с) з 0,5-секундної затримки руху проколу.

Коли відстань виїмки досягається, швидкість пальника встановлюється на швидкість повзучого руху 245 мм/хв (4 мм/с) для відстані повзучого руху 3 мм. Відстань повзучого руху займе приблизно 0,7 секунди (3 мм / 4 мм/с = 0,7 с).

Висота пальника залишиться на рівні 4 мм ще протягом 0,2 секунди (0,5 с (затримка руху проколюванн - 0,3 с (відстань проколювання при швидкості проколювання) = 0,2 с), після чого пальник почне опускатися до кінцевої висоти проколювання 2,5 мм протягом решти 0,5 секунди затримки проколюван матеріалу. Оскільки залишається 0,5 секунди затримки пробивання матеріалу, а також 0,5 секунди на швидкості повзучості, відстань повзучості буде покрита одночасно з досягненням кінцевої висоти пробивання.

Коли буде досягнуто відстань повзучості, швидкість пальника буде встановлена на швидкість подачі різання матеріалу 490 мм/хв. Оскільки існує 1-секундна затримка висоти різання, яка почалася в кінці затримки пробивання матеріалу, перехід до висоти різання 1,5 мм відбудеться після закінчення решти 1 секунди затримки висоти різання.

Наведений вище текст має продемонструвати, що існує чимало можливостей для налаштування, а тонкощі можна досягти шляхом експериментів та ретельного використання різних комбінацій параметрів.

10.8.16 Інтернаціоналізація

Можна створити файли перекладу для QtPlasmaC для відображення мовою поточної локалізації.

Для створення та/або редагування файлу перекладу необхідно, щоб LinuxCNC був встановлений як запущений на місці.

Наступне припускає, що каталог LinuxCNC git — ~/linuxcnc-dev.

Усі мовні файли зберігаються в ~/linuxcnc-dev/share/screens/qtplasmac/languages.

Файл qtplasmac.py — це версія файлу графічного інтерфейсу на Python, яка використовується для перекладу.

Файли .ts – це вихідні файли перекладів. Це файли, які потрібно створити/відредагувати для кожної мови.

Файли .qm – це скомпільовані файли перекладу, що використовуються PyQt.

Директорії qtplasmac_4x3/languages та qtplasmac_9x16/languages призначені лише для посилань на файли .qm у qtplasmac/languages.

Мова визначається підкресленням та першими двома літерами локалі. Наприклад, якщо виконується італійський переклад, то це буде `_it`. У цьому документі це буде позначено як _xx, тому `qtplasmac_xx.t` у цьому документі насправді буде `qtplasmac_it.ts` для італійського перекладу.

Локаль QtPlasmaC за замовчуванням — `_en`, що означає, що будь-які файли перекладу, створені як `qtplasmac_en.*`, не будуть використані для перекладів.

Якщо будь-яка з необхідних утиліт (pyuic5, pylupdate5, linguist) не встановлена, користувачеві потрібно буде встановити необхідні інструменти розробки:

```
sudo apt install qttools5-dev-tools pyqt5-dev-tools
```

Перехід до каталогу мов:

```
cd ~/linuxcnc-dev/share/qtvcp/screens/qtplasmac/languages
```

Якщо до графічного інтерфейсу було внесено будь-які зміни тексту, виконайте наступну команду, щоб оновити файл графічного інтерфейсу Python:

```
pyuic5 ../qtplasmac.ui > qtplasmac.py
```

Користувач може створити новий файл-джерело перекладу для неіснуючої мови перекладу або змінити існуючий файл-джерело перекладу через зміни, внесені в текст файлу-джерела QtPlasmaC. Якщо ви змінюєте існуючий переклад, в якому не було змін у файлі-джерелі, цей крок не є обов’язковим.

Створення або редагування файлу .ts:

```
./langfile xx
```

Note
ця команда — це скрипт, який виконує наступне: $ pylupdate5 **.py ../**.py ../../../../../lib/python/qtvcp/lib/qtplasmac/*.py -ts qtplasmac_xx.ts

---

Редагування перекладу виконується за допомогою лінгвістичного застосунку:

```
linguist
```

1. Відкрийте файл TS та перекладіть рядки

Не обов’язково надавати переклад для кожного текстового рядка. Якщо переклад для рядка не вказано, у програмі буде використовуватися оригінальний рядок. Користувач повинен бути обережним із довжиною рядків, що з’являються на віджетах, оскільки простір обмежений. Якщо можливо, намагайтеся, щоб переклад не був довшим за оригінал.

Після завершення редагування збережіть файл:
`Файл -> Зберегти`

Потім створіть файл .qm:
`Файл -> Реліз`

Близький лінгвіст.

Потім створіть посилання на скомпільований файл .qm для інших графічних інтерфейсів QtPlasmaC.

```
$ ./langlink xx
```

Note
Ця команда — це скрипт, який створює посилання як у qtplasmac_4x3/languages, так і в qtplasmac_9x16/languages на вищезгаданий файл .qm, а потім перейменовує посилання відповідно до назви графічного інтерфейсу.

QtPlasmaC буде перекладено мовою поточної локалізації під час наступного запуску, якщо існує файл .qm цією мовою.

Користувачі можуть надсилати файли перекладів для включення до QtPlasmaC. Найпростіший спосіб — опублікувати оновлений файл qtplasmac_xx.ts на форумі, і адміністратори встановлять переклади.

Найкращий спосіб — це надіслати запит на витягнення з облікового запису користувача GitHub, як описано в документації contributing to LinuxCNC. Файли, які необхідно підтвердити, — це qtplasmac_xx.ts та qtplasmac_xx.qm у каталозі qtplasmc/languages, а також посилання в каталогах qtplasmac_4x3/languages та qtplasmac_9x16/languages.

10.8.17 Додаток

10.8.17.1 Приклади конфігурацій

Існують приклади файлів конфігурації, які використовують графічний інтерфейс QtPlasmaC для моделювання машин плазмового різання.

Їх можна знайти у селекторі LinuxCNC за адресою: Зразки конфігурацій -> sim -> qtplasmac

Доступні три версії як у метричних, так і в імперських одиницях вимірювання:

1. qtplasmac_l - формат 16:9, мінімальна роздільна здатність 1366x768
2. qtplasmac_p - формат 9:16, мінімальна роздільна здатність 786x1366
3. qtplasmac_s – формат 4:3, мінімальна роздільна здатність 1024x768

Кожна зразкова конфігурація містить спливаючу панель керування для імітації різних вхідних даних у графічний інтерфейс, таких як:

1. НАПРУГА ДУГИ
2. ОМІЧНЕ ВІДЧУТТЯ
3. ПОПЛАВКОВИЙ ВИМИКАЧ
4. РОЗІРВАНИЙ ПЕРЕМИКАЧ
5. ЕСТІЙ

10.8.17.2 Зразки NGC

У каталозі ~/linuxcnc/nc_files/examples/plasmac є кілька зразків файлів G-коду.

10.8.17.3 Специфічні G-коди QtPlasmaC

Опис	**Код**
Початок cut	`M3 $0 S1`
Кінець cut	`M5 $0`
Початок scribe	`M3 $1 S1`
Кінець scribe	`M5 $1`
Початок center spot	`M3 $2 S1`
Кінець center spot	`M5 $2`
Покладіть край усьому вищесказаному.	`M5 $-1`
Виберіть material.	`M190 P`*n* *n* позначає номер матеріалу.
Зачекайте підтвердження зміни material.	`M66 PG L3 Q`*n* + *n* – час затримки (у секундах). Це значення може знадобитися збільшити для дуже великих файлів матеріалів.
Встановіть швидкість подачі з material.	`F#<_hal[plasmac.cut-feed-rate]>`
Enable Ignore Arc OK	`M62 P1` (синхронізовано з рухом) `M64 P1` (негайно)
Disable Ignore Arc OK	`M63 P1` (синхронізовано з рухом) `M65 P1` (негайно)
Вимкнути THC	`M62 P2` (синхронізовано з рухом) `M64 P2` (негайно)
Увімкнути THC	`M63 P2` (синхронізовано з рухом) `M65 P2` (негайно)
Вимкнути Torch	`M62 P3` (синхронізовано з рухом) `M64 P3` (негайно)
Увімкнути Torch	`M63 P3` (синхронізовано з рухом) `M65 P3` (негайно)
Встановіть velocity на відсоток від швидкості подачі.	`M67 E3 Q`*n* (синхронізовано з рухом) `M68 E3 Q`*n* (негайний) *n* — відсоток, який потрібно встановити 10 — мінімум, нижче цього значення буде встановлено 100% 100 — максимум, вище цього значення буде встановлено 100% **Рекомендується мати M68 E3 Q0 як у преамбулі, так і в пост-амбулі.**
Різак компенсація - ліворуч від шляху	`G41.1 D#<_hal[plasmac.kerf-width]>`
Різак компенсація - праворуч від шляху	`G42.1 D#<_hal[plasmac.kerf-width]>`
Різак компенсація вимкнено	`G40` **Зверніть увагу, що M62–M68 недійсні, поки ввімкнено корекцію на різець.**
Різання отвори зі швидкістю подачі 60%	`#<holes> = 1` для отворів діаметром менше 32 мм (1,26 дюйма)

Опис	Код
Ріжте отвори зі швидкістю подачі 60%, вимкніть пальник на кінці отвору, продовжуйте різання отвору для нарізання.	#<holes> = 2 для отворів діаметром менше 32 мм (1,26 дюйма) надлишкова довжина різу = 4 мм (0,157 дюйма)
Різання отвори та дуги зі швидкістю подачі 60%.	#<holes> = 3 для отворів діаметром менше 32 мм (1,26 дюйма) для дуг радіусом менше 16 мм (0,63 дюйма)
Вирізати отвори та дуги зі швидкістю подачі 60%, вимкнути пальник на кінці отвору, продовжити траєкторію отвору для нарізання.	#<holes> = 4 для отворів діаметром менше 32 мм (1,26 дюйма) для дуг радіусом менше 16 мм (0,63 дюйма) над довжиною різу = 4 мм (0,157 дюйма)
Вкажіть діаметр отвор для #<отворів> = 1-4.	#<h_diameter> = *n* (*n* – діаметр, використовуйте ту саму систему одиниць, що й у решті файлу G-коду)
Вкажіть швидкість отвор для #<отворів> = 1-4.	#<h_velocity> = *n* (*n* це відсоток, встановіть відсоток поточної швидкості подачі)
Вкажіть довжину over cut.	#<oclength> = *n* (*n* – довжина, використовуйте ту саму систему одиниць, що й для решти файлу G-коду)
Вкажіть режим лише прокол.	#<pierce-only> = *n* (*n* – це режим, 0=звичайний режим різання, 1=режим лише проколювання)
Створення або редагування матеріалів. Параметри: 0 - Створити тимчасовий стандартний варіант 1 - Додати, якщо не існує 2 - Перезаписати, якщо існує, інакше додати новий	обов'язкові параметри: (o=<*option*>, nu=<*nn*>, na=<*ll*>, ph=<*nn*>, pd=<*nn*>, ch=<*nn*>, fr=<*nn*>) додаткові параметри: (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 Приклади G-коду QtPlasmaC

Опис	Приклад
Виберіть матеріал і зробіть звичайний розріз	M190 P3 M66 P3 L3 Q1 F#<_hal[plasmac.cut-feed-rate]> M3 $0 S1 . . M5 $0
Встановити швидкість на 100% від CutFeedRate	M67 E3 Q0 або M67 E3 Q100
Встановити швидкість на 60% від CutFeedRate	M67 E3 Q60
Встановити швидкість на 40% від CutFeedRate	M67 E3 Q40

Опис	**Приклад**
Вирізання отвору зі зниженою на 60% швидкістю за допомогою налаштування швидкості	G21 (метричний) G64 P0.05 M52 P1 (увімкнути адаптивну подачу) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (почати різання) G1 X0 M67 E3 Q60 (зменшити швидкість подачі до 60%) G3 I10 (отвір) M67 E3 Q100 (відновити швидкість подачі до 100%) M5 $0 (закінчити різання) G0 X0 Y0 M2 (закінчити роботу)
Вирізання отвору зі зниженою на 60% швидкістю за допомогою команди #<отвори>	G21 (метрична) G64 P0.05 M52 P1 (увімкнути адаптивну подачу) #<отвори> = 1 (зниження швидкості для отворів) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (початок різання) G1 X0 G3 I10 (отвір) M5 $0 (закінчення різання) G0 X0 Y0 M2 (закінчення роботи)
Вирізати отвір з перерізом за допомогою відключення пальника	G21 (метрична) G64 P0.05 M52 P1 (увімкнути адаптивну подачу) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (почати різання) G1 X0 M67 E3 Q60 (зменшити швидкість подачі до 60%) G3 I10 (отвір) M62 P3 (вимкнути пальник) G3 X0.8 Y6.081 I10 (продовжити рух на 4 мм) M63 P3 (дозволити увімкнути пальник) M67 E3 Q0 (відновити швидкість подачі до 100%) M5 $0 (закінчити різання) G0 X0 Y0 M2 (закінчити роботу)
Вирізання отвору з перекриттям за допомогою команди #<отвори>	G21 (метричний) G64 P0.05 M52 P1 (увімкнути адаптивну подачу) #<отвори> = 2 (надріз для отворів) F#<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (початок різання) G1 X0 G3 I10 (отвір) M5 $0 (закінчення різання) G0 X0 Y0 M2 (закінчення роботи)

Опис	Приклад
Виріжте отвір із перекриттям 6,5 мм за допомогою команди #<отвори>	G21 (метрична) G64 P0.05 M52 P1 (увімкнути адаптивну подачу) #<отвори> = 2 (надріз для отворів) <oclength> = 6.5 (6,5 мм довжина надрізу) F<_hal[plasmac.cut-feed-rate]> G0 X10 Y10 M3 $0 S1 (початок різання) G1 X0 G3 I10 (отвір) M5 $0 (закінчення різання) G0 X0 Y0 M2 (закінчення роботи)
Виберіть розписувач і виберіть пальник в кінці розписування	. +. M52 P1 (увімкнути адаптивну подачу) F#<_hal[plasmac.cut-feed-rate]> T1 M6 (вибрати різьблення) G43 H0 (застосувати зміщення) M3 $1 S1 (запустити плазмову різьбу) +. T0 M6 (вибрати пальник) G43 H0 (застосувати зміщення) G0 X0 Y0 (позиція паркування) M5 $1 (закінчити)
Визначення центру отвору.	(Потрібна невелика команда руху, інакше нічого не відбудеться) G21 (метрична система) F99999 (висока швидкість подачі) G0 X10 Y10 M3 $2 S1 (включення споттингу) G91 (режим відносної відстані) G1 X0.000001 G90 (режим абсолютної відстані) M5 $2 (споттинг вимкнено) G0 X0 Y0 G90 M2
Створити тимчасовий матеріал за замовчуванням	(o=0, nu=2, na=5mm Mild Steel 40A, ph=3.1, pd=0.1, ch=0.75, fr=3000)
Редагувати матеріал, якщо його немає, створити новий	(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 Таблиця THCAD

Mesa THCAD є поширеним способом отримання напруги дуги від плазмового різака, а також корисний для омічного зондування матеріалу під час тестування. THCAD може використовуватися для конфігурацій паралельного порту, а також для конфігурацій, що використовують апаратне забезпечення Mesa Electronics. THCAD доступний у трьох різних моделях: THCAD-5, THCAD-10 та THCAD-300.

На кожній платі THCAD є перемичка режимів, яку слід встановити в положення **UNIPOLAR**

На кожній платі THCAD є перемичка дільника частоти, яку слід встановити відповідно до типу обладнання:

Вхідний пристрій	Рекомендовані налаштування
Паралельний порт з дуже низькою затримкою	F/32
Рекомендована початкова точка паралельного порту	F/64
Паралельний порт з більшою затримкою або під час різання товстого матеріалу	F/128
Картка Mesa	F/32

Це значення потрібно ввести в PnCConf під час встановлення.

Note
При використанні паралельного порту користувачеві може знадобитися налаштувати джемпер і відповідні значення масштабування на вкладці Parameters Tab для досягнення оптимальних результатів. Симптоми можуть включати випадкові підйоми або занурення пальника під час стабільного різання. Для діагностики цих проблем можуть бути корисними графіки Halscope.

На задній панелі THCAD розташована калібрувальна наклейка, на якій зазначено:

```
   THCAD-nnn

0V  121.1 kHz
5V  925.3 kHz
```

або подібні значення, ці значення потрібно ввести в PnCConf під час встановлення.

PnCConf має записи для всіх необхідних параметрів THCAD та розрахує й налаштує будь-які необхідні параметри. Використані розрахунки такі:

Шкала напруги

```
vs = r / ((f - z) / d / v)
```

Зсув напруги

```
vo = z / d
```

r = коефіцієнт дільника (див. нижче).

f = значення повної шкали з калібрувальної наклейки.

Значення z = 0 В з калібрувальної наклейки.

d = значення з перемички вище.

v = повна шкала напруги THCAD

Коефіцієнт дільника *THCAD-5 або THCAD-10*

Якщо підключається до порту плазмового CNC, то коефіцієнт дільника вибирається з плазмового верстата. Загальний коефіцієнт, що використовується, становить 20:1.

Якщо підключення до плазмових машин здійснюється за повним напругою дуги, то типовою схемою для THCAD-10 є використання резистора 1 МОм від мінуса дуги до мінуса THCAD і резистора 1 МОм від плюса дуги до плюса THCAD. Коефіцієнт подільника отримується за формулою:

```
r = (total_resistance + 100000) / 100000
```

THCAD-300

```
r = 1
```

Important
ЯКЩО КОРИСТУВАЧ ВИКОРИСТОВУЄ ДЖЕРЕЛО ЖИВЛЕННЯ ДЛЯ ПЛАЗМИ ВЧ-СТАРТ, ТО КОЖЕН З ЦИХ ОПОРІВ ПОВИНЕН СКЛАДАТИСЯ З ДЕКІЛЬКОХ ВИСОКОВОЛЬТНИХ РЕЗИСТОРІВ.

Caution
ЯКЩО КОРИСТУВАЧ ВИКОРИСТОВУЄ ДЖЕРЕЛО ЖИВЛЕННЯ ПЛАЗМИ HF START, ТО ОМІЧНЕ ДОТИКУВАННЯ НЕ РЕКОМЕНДУЄТЬСЯ.

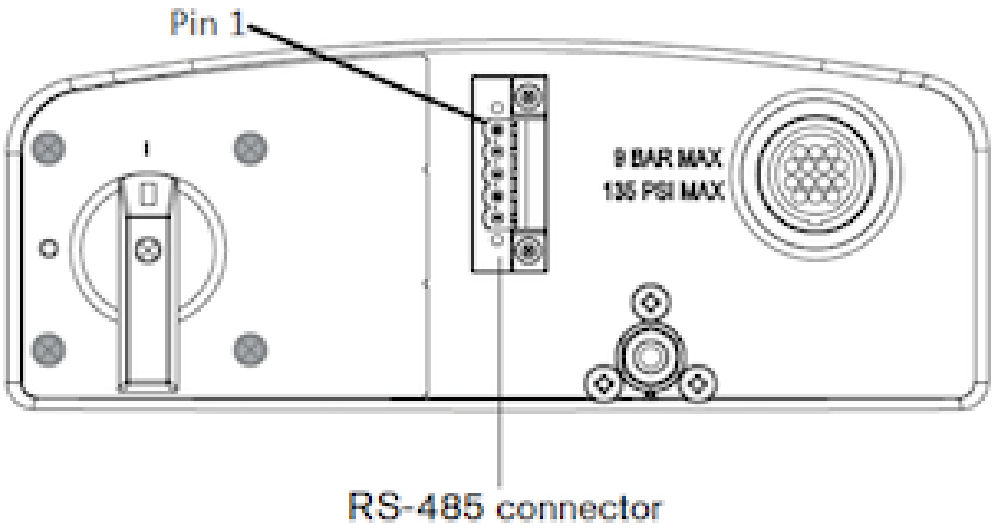
Note
Ці значення можна розрахувати за допомогою цей онлайн-калькулятор.

Note
Існує фільтр низькочастотний, який може бути корисним, якщо використовується THCAD, і на поверненій напрузі дуги є багато шуму.

10.8.17.6 З’єднання RS485

Схема підключення Hypertherm RS485 (кольори проводів всередині Hypertherm в дужках):

Підключення до контакту машини №	Підключення на платі Breakout
1 - Tx+ (Червоний)	->RXD+
2 - Tx- (Чорний)	->RXD-
3 - Rx+ (Коричневий)	->T/R+
4 - Rx- (Білий)	->T/R-
5 - GND (Зелений)	->GND



Інтерфейси RS485, які, як відомо, працюють:

Адаптер-перетворювач DTECH DT-5019 USB на RS-485:

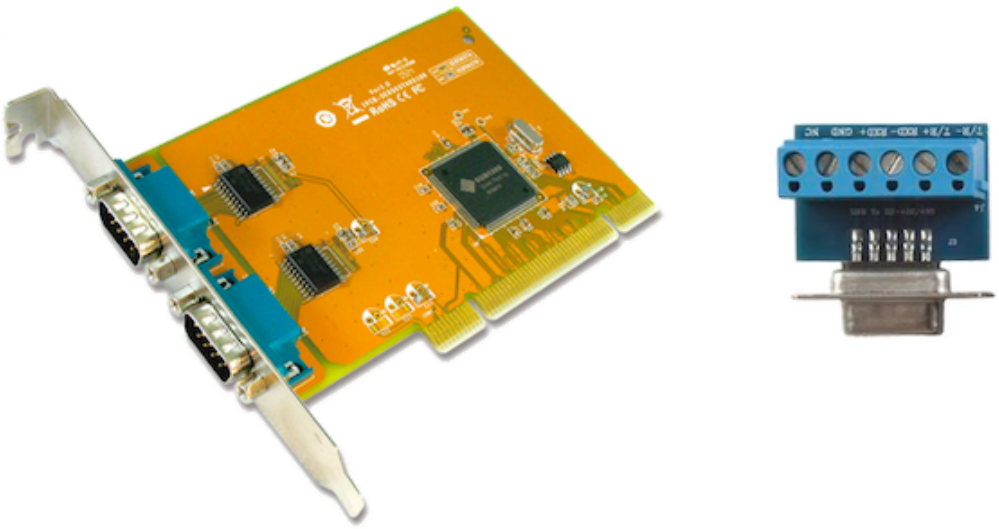


Для перетворення послідовного з’єднання материнської плати або послідовної карти (RS232) на RS485 необхідно виконати наступне:

Перетворювач DTECH RS-232 на RS-485:



Приклад послідовної карти (карта PCI Sunnix SER5037A показана з платою Breakout):

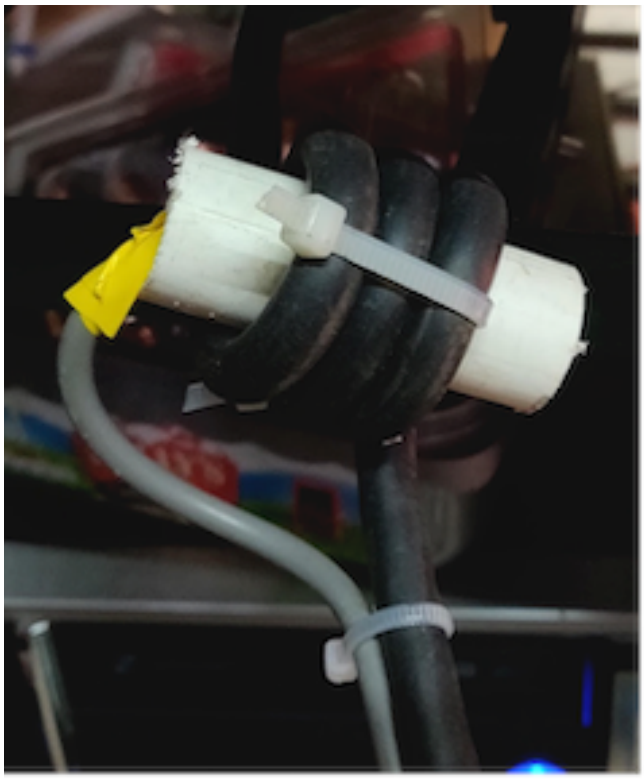


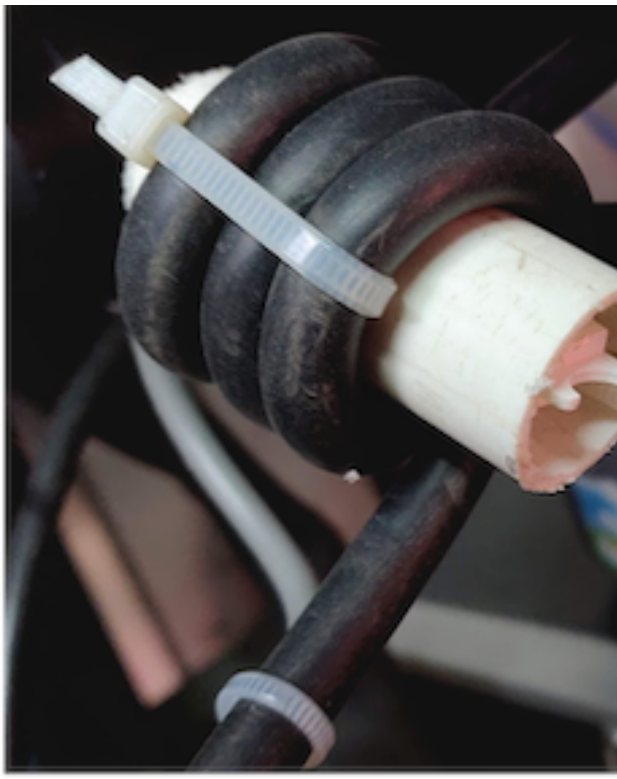
10.8.17.7 Дуга в порядку з герконовим реле

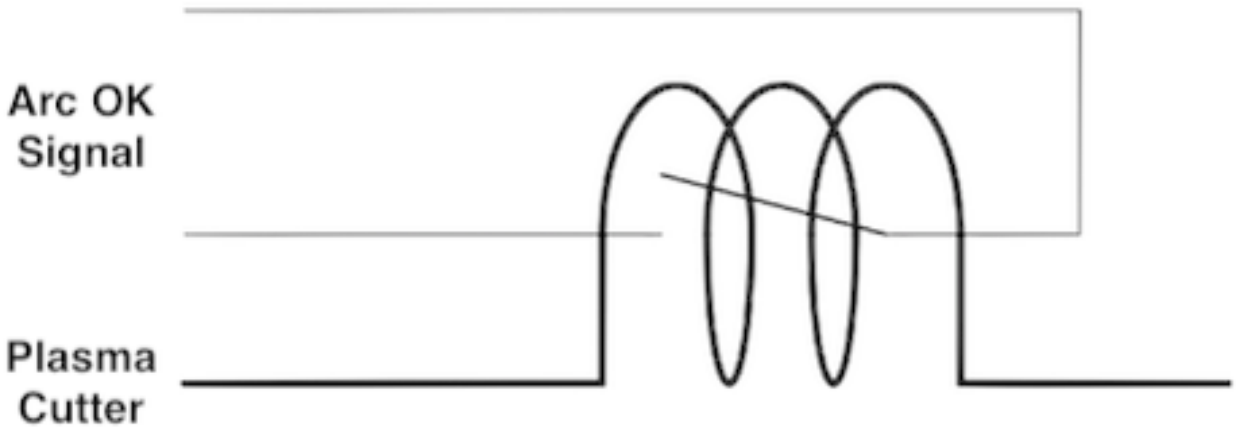
Ефективним і дуже надійним методом отримання сигналу Arc OK від джерела живлення плазми без порту ЧПУ є встановлення герконового реле всередині непроводячої трубки та обмотування і закріплення трьох витків робочого проводу навколо трубки.

Цей вузол тепер діятиме як реле, яке вмикатиметься, коли через робочий провід протікає струм, що відбувається лише тоді, коли встановлена ріжуча дуга.

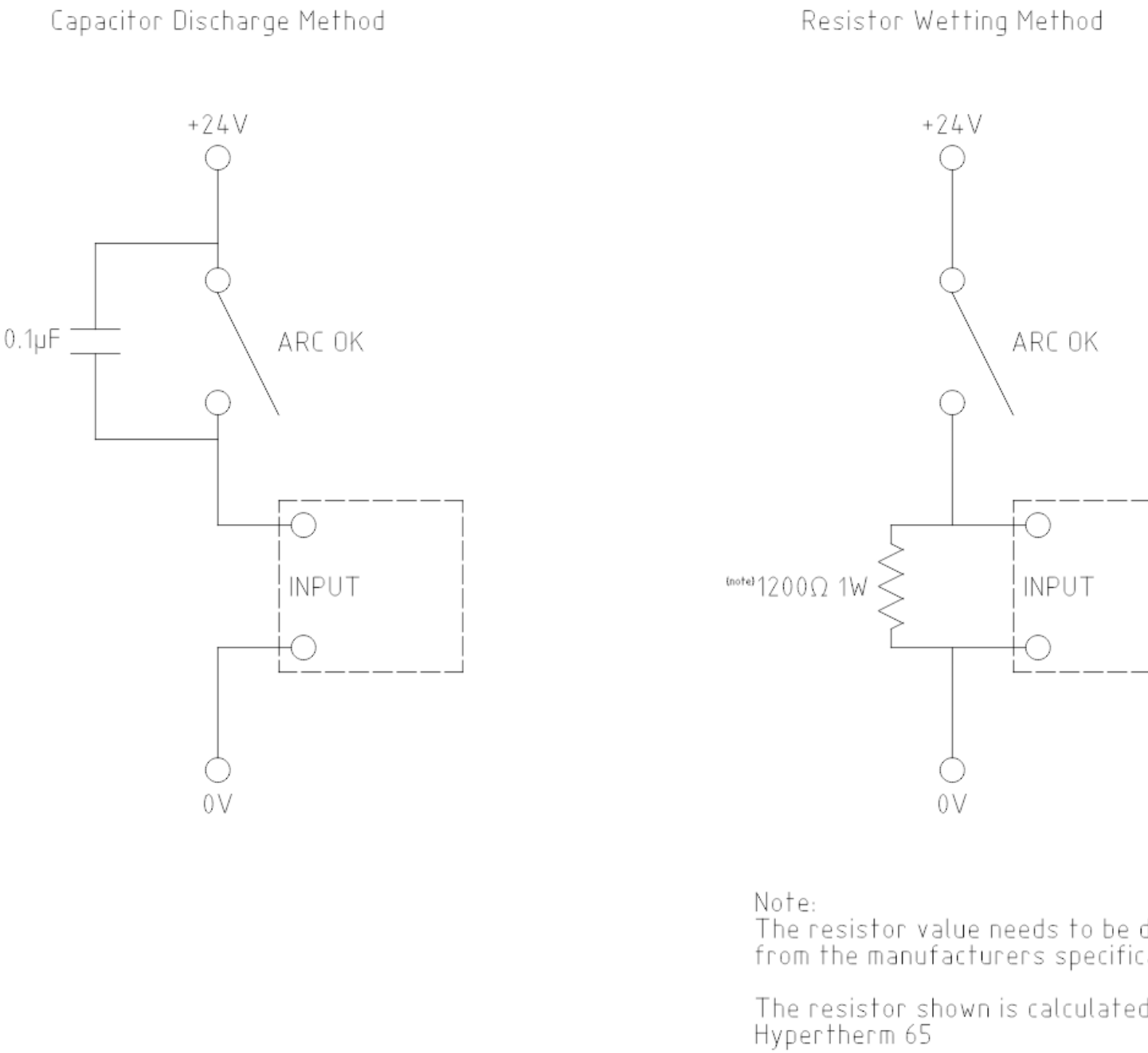
Це вимагатиме, щоб QtPlasmaC працював у режимі 1, а не в режимі 0. Див. розділи QtPlasmaC Modes для отримання додаткової інформації.






Arc OK
Signal
Plasma
Cutter

10.8.17.8 Схеми контактного навантаження



Повний опис знаходиться за адресою Contact Load.

10.8.18 Відомі проблеми

10.8.18.1 Біг по клавіатурі

Існує відома проблема з деякими комбінаціями апаратного забезпечення та клавіатур, яка може впливати на функцію автоповторення клавіатури, а отже, і на роботу клавіатури, що супроводжується періодичними зупинками та запусками під час роботи. Цю проблему можна запобігти, вимкнувши функцію автоповторення операційної системи для всіх клавіш. QtPlasmaC використовує цю функцію вимкнення за замовчуванням для всіх клавіш тільки тоді, коли MAIN Tab є видимим, за винятком

наступних випадків, коли автоповторення дозволено, коли MAIN Tab є видимим: редактор G-коду активний, MDI активний. Коли QtPlasmaC вимкнено, функція автоповторення операційної системи буде ввімкнена для всіх клавіш.

Якщо користувач бажає заборонити QtPlasmaC змінювати налаштування автоповтору операційної системи, введіть наступний параметр у розділі **[GUI_OPTIONS]** файлу <*назва_машини*>.prefs:

```
b''Ab''b''вb''b''тb''b''оb''b''мb''b''аb''b''тb''b''иb''b''чb''b''нb''b''оb'' b''пb''b' ↩
    'оb''b''вb''b''тb''b''оb''b''рb''b''юb''b''вb''b''аb''b''тb''b''иb'' b''вb''b''сb''b' ↩
    'еb'' == True
```

Ця проблема не впливає на керування підйомом за допомогою кнопок керування підйомом графічного інтерфейсу.

Note
Відключення та повторне підключення клавіатури під час активної сесії QtPlasmaC призведе до автоматичного повторного ввімкнення функції автоповторення, що може спричинити періодичні зупинки та запуски під час бігу. Користувач повинен перезапустити QtPlasmaC, щоб знову вимкнути функцію автоповторення.

10.8.18.2 NO_FORCE_HOMING

QtPlasmaC наразі не дотримується наступного розділу у файлі <*назва_машини*>.ini:

```
NO_FORCE_HOMING = 1
```

Незалежно від цього налаштування, QtPlasmaC вимагає, щоб машина була переведена в початковий стан перед виконанням команд MDI або запуском програм.

10.8.19 Внесок коду в QtPlasmaC

Виправлення помилок та вдосконалення QtPlasmaC завжди вітаються. Найкращий спосіб внести свій вклад у код — це надіслати запит на витяг (PR), що складається з одного коміту, до репозиторію LinuxCNC GitHub. Для отримання додаткової інформації про створення PR див. документацію LinuxCNC, єдиною попередньою умовою є реєстрація облікового запису GitHub. Усі PR перевіряються, а потім підтверджуються одним із розробників. Якщо вам не зручно надсилати PR, то прийнятним методом є додавання змін коду до теми на форумі LinuxCNC Forum.

Виправлення помилок приймаються як для останньої випущеної гілки, так і для головної гілки. Якщо виправлення помилки стосується обох гілок, то необхідно подати PR тільки для останньої випущеної гілки, оскільки вона буде об'єднана з головною гілкою розробником.

Покращення приймаються лише для головної гілки.

Кожен PR, за винятком змін тільки в документації QtPlasmaC, вимагає збільшення відповідного номера версії, а також оновлення історії версій. Номери версій знаходяться в наступних місцях:

Розташування	**Формат**	**Збільшується, коли**
src/hal/components/plasmac.comp	nnn	зміни коду компонентів
share/qtvcp/screens/qtplasmac/qtplasmac_handler.py	nnn.*nnn*	зміни коду компонентів *Зміни в коді графічного інтерфейсу*

Історія версій знаходиться за адресою share/qtvcp/screens/qtplasmac/versions.html.

10.8.20 Підтримка

Онлайн-допомога та підтримка доступні на сторінці розділ PlasmaC форуму LinuxCNC Forum.

Користувач може створити стислий файл, що містить повну конфігурацію машини, для полегшення діагностики несправностей, натиснувши кнопку та дотримуючись інструкцій у розділі backup. Отриманий файл можна додати до повідомлення на форумі LinuxCNC, щоб допомогти спільноті діагностувати конкретні проблеми.

10.9 MDRO GUI

10.9.1 Вступ

MDRO — це простий графічний інтерфейс для LinuxCNC, що забезпечує відображення даних з цифрових шкал (DRO). Він надає функціональність, схожу на звичайний дисплей DRO для верстатів, дозволяючи користувачеві використовувати шкали DRO на верстаті під час роботи в ручному режимі (з ручним приводом). Він найбільш корисний для ручних верстатів, таких як фрезерні верстати типу Bridgeport, оснащені DRO, які були переобладнані на CNC, але все ще мають ручне управління.

MDRO зручний для роботи з мишею та сенсорним екраном.

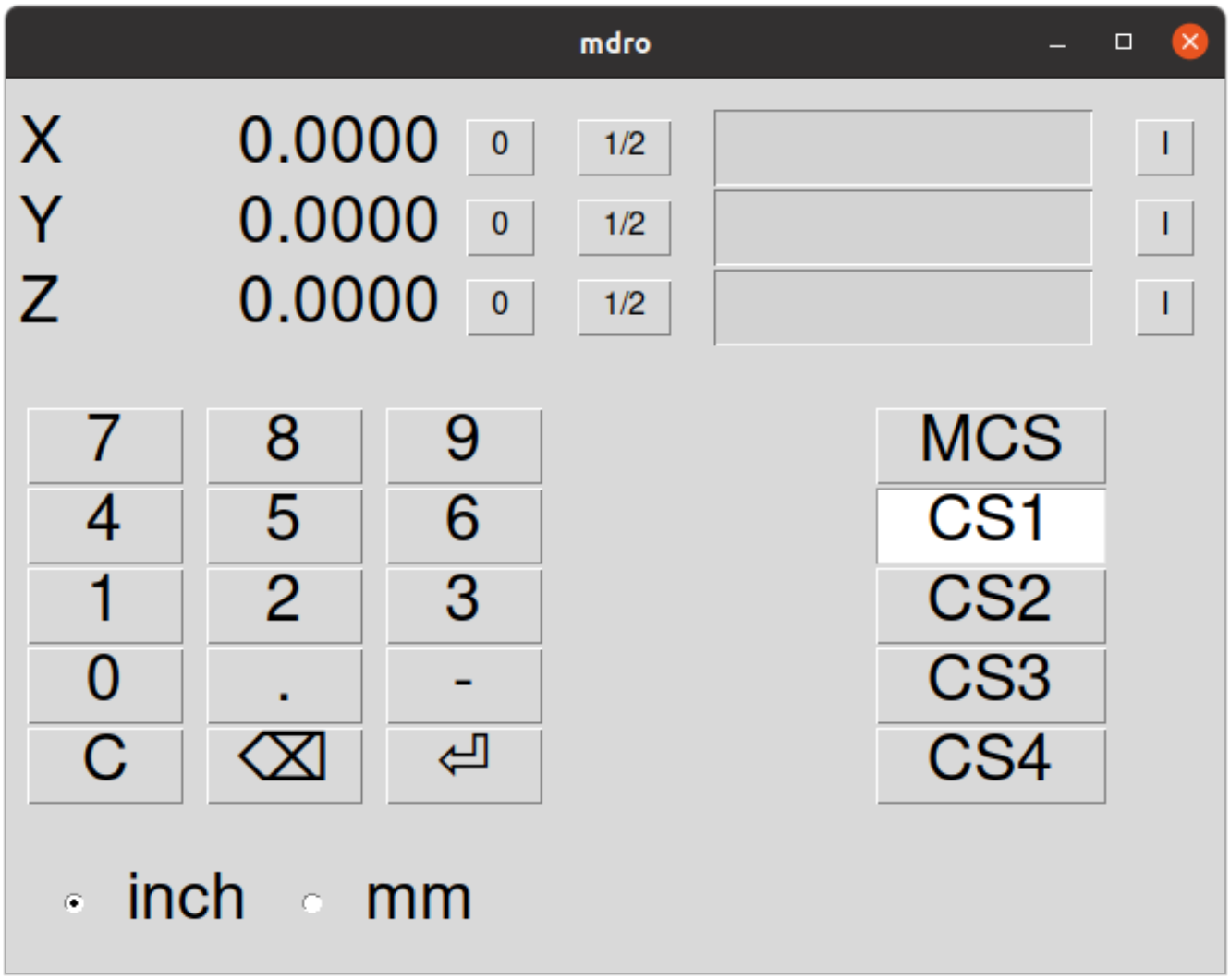


Figure 10.51: Вікно MDRO

10.9.2 Початок роботи

Якщо ваша конфігурація наразі не налаштована на використання MDRO, ви можете змінити її, відредагувавши файл INI. У розділі `[DISPLAY]` змініть рядок `DISPLAY =` на `DISPLAY = mdro`. За замовчуванням MDRO використовує осі XYZ, але це можна змінити. Встановіть розділ `[DISPLAY]` `GEOMETRY = XYZ` для 3-осьового фрезерного верстата. Токарний верстат з шкалами DRO на осях X і Z може використовувати `GEOMETRY = XZ`.

Після запуску MDRO з’являється вікно, подібне до того, що показано на малюнку Figure 10.51 відкривається вище.

10.9.2.1 Параметри INI-файлу

Інші опції, які можна включити до розділу `[DISPLAY]`, включають:

- `MDRO_VAR_FILE = <file.var>` - попереднє завантаження даних системи координат G54 - G57.

  - Попередньо завантажте файл .var. Зазвичай це файл .var, який використовується операційним кодом.

- `POINT_SIZE = <n>` - Встановити розмір тексту в пунктах.

  - Цей параметр встановлює розмір шрифту, який визначає загальний розмір вікна. Розмір за замовчуванням становить 20 пунктів, типові розміри — від 20 до 30.

- `MM = 1` Встановіть це значення, якщо шкали DRO надають дані, масштабовані в міліметрах.

10.9.2.2 Параметри командного рядка

MDRO можна запустити командою `loadusr` у HAL-файлі. Параметри, еквівалентні тим, що є у INI-файлі, можна встановити в командному рядку:

- `-l <file.var>` - попереднє завантаження даних системи координат G54 - G57.
- `-p <n>` - Встановити розмір тексту в пунктах.
- `-m` - Встановіть це значення, якщо шкали DRO надають дані, масштабовані в міліметрах.
- `<axes>` - осі для відображення. Див. `ГЕОМЕТРІЯ` вище.

10.9.2.3 Піни

Використовуючи приклад ”XYZA” для аргументу AXES, ці контакти будуть створені під час запуску MDRO:

```
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
```

У цьому прикладі перший рядок дисплея буде позначений як «X» і відображатиме дані зі шкали DRO, підключеної до виводу «mdro.axis.0». Виводи «mdro.index-enable.n» слід підключити до індексних виводів DRO, якщо DRO їх підтримує.

Контакти повинні бути підключені у файлі, вказаному в записі `POSTGUI_HALFILE` файлу INI, коли програма запускається з файлу INI. Вони можуть бути встановлені безпосередньо після команди loadusr, якщо програма запускається з файлу HAL.

10.9.3 Вікно MDRO

Вікно MDRO містить такі елементи:

- Рядок для кожної осі. Кожен рядок містить:
  - назва осі,
  - поточне значення,
  - кнопка ”z”, яка обнуляє значення,
  - кнопка ”1/2”, яка зменшує значення вдвічі,

  - поле введення, яке можна використовувати для встановлення значення, визначеного користувачем. Це поле можна встановити з клавіатури або за допомогою екранної клавіатури.
  - Кнопка «I», яка запускає операцію з індексом (див. нижче),
- клавіатура, що використовується для встановлення значень у полі введення за допомогою миші або сенсорного екрана,
- кнопки вибору системи координат:
  - Кнопка ”mcs” вибирає систему координат машини. Це необроблені значення з енкодерів, підключених до контактів `mdro.axis.n`.
  - Кнопки «cs1» - «cs4» дозволяють користувачеві вибрати одну з чотирьох визначених користувачем систем координат. Якщо програма запущена з опцією `MDRO_VAR_FILE =`, мітки будуть змінені на «g54» - «g57», а значення з вказаного файлу .var будуть завантажені заздалегідь. Зверніть увагу, що будь-які зміни значень не є постійними: файл .var ніколи не змінюється.
- Кнопки вибору дюймів/міліметрів.

10.9.4 Операції з індексами

`MDRO` підтримує шкали DRO з індексними позначками. Натисніть кнопку «I» на рядку осі, а потім поверніть вісь до індексного положення. Координати верстата будуть обнулені. Це найлегше побачити під час запуску або коли вибрано систему координат «mcs».

10.9.5 Симулятор

Найпростіший спосіб побачити, як працює `MDRO`, – це спробувати його в середовищі симуляції. Додайте цей розділ у кінець вашого HAL-файлу симуляції, зазвичай це ”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

G-code програмування

11.1 Системи координат

11.1.1 Вступ

У цьому розділі ми спробуємо розвіяти міфи про системи координат. Це дуже важлива концепція для розуміння роботи верстата з CNC, його конфігурації та використання.

Ми також покажемо, що дуже цікаво використовувати точку відліку на заготовці або деталі і змусити програму працювати з цієї точки, не беручи до уваги місце розташування деталі на столі.

У цьому розділі ви знайдете інформацію про зміщення, які використовуються в LinuxCNC. До них належать:

- Координати машини (G53)
- Дев’ять зміщень системи координат (G54-G59.3)
- Глобальні зміщення (G92) та локальні зміщення (G52)

11.1.2 Система координат машини

При запуску LinuxCNC положення кожної осі є початком координат верстата. Після повернення осі в початкове положення початок координат верстата для цієї осі встановлюється в початкове положення. Початок координат верстата є системою координат верстата, на якій базуються всі інші системи координат. G-код G53 можна використовувати для переміщення в системі координат верстата.

11.1.2.1 Переміщення координат машини: G53

Незалежно від будь-якого активного зміщення, код G53 у рядку коду вказує інтерпретатору переміститися до вказаних фактичних позицій осей (абсолютних позицій). Наприклад:

```
G53 G0 X0 Y0 Z0
```

перемістить з поточного положення в положення, де координати трьох осей верстата будуть дорівнювати нулю. Цю команду можна використовувати, якщо у вас є фіксоване положення для зміни інструменту або якщо ваш верстат має автоматичний змінювач інструменту. Цю команду також можна використовувати для очищення робочої зони та доступу до заготовки в лещатах.

G53 — це немодальна команда. Її необхідно використовувати в кожному блоці, де потрібне переміщення в системі координат верстата.

11.1.3 Системи координат

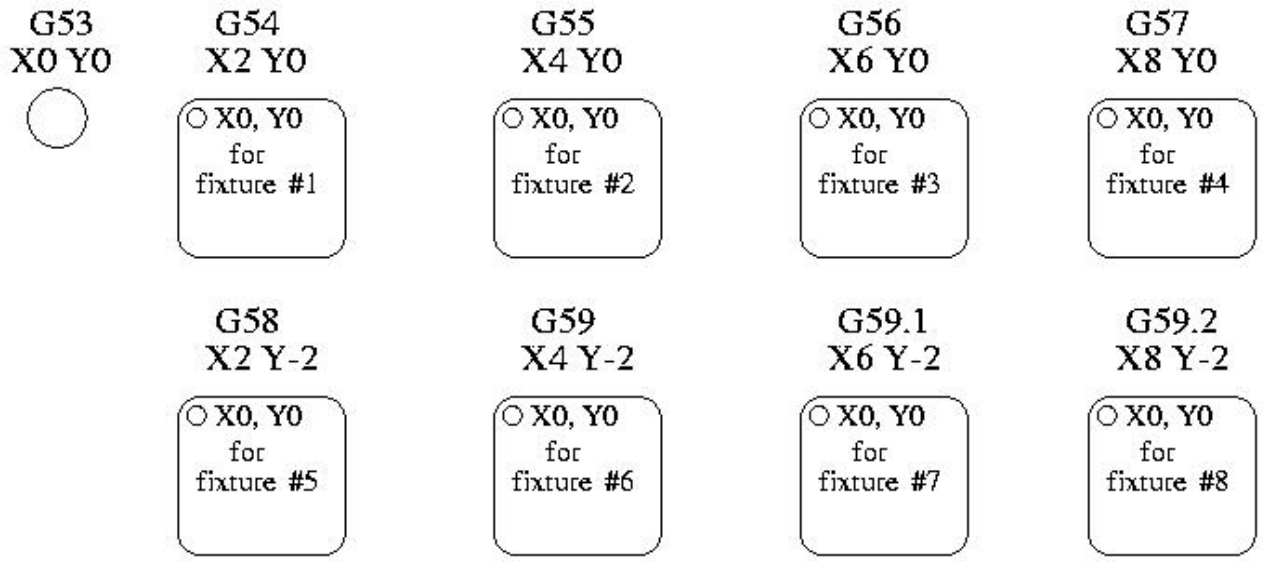


Figure 11.1: Приклад систем координат

Зміщення системи координат

- G54 - використовувати систему координат 1
- G55 - використовувати систему координат 2
- G56 - використовувати систему координат 3
- G57 - використовувати систему координат 4
- G58 - використовувати систему координат 5
- G59 - використовувати систему координат 6
- G59.1 - використовувати систему координат 7
- G59.2 - використовувати систему координат 8
- G59.3 - використовувати систему координат 9

Зсуви координатної системи використовуються для зміщення координатної системи від координатної системи верстата. Це дозволяє програмувати G-код для деталі без урахування її розташування на верстаті. Використання зсувів координатної системи дозволяє обробляти деталі в різних місцях за допомогою одного і того ж G-коду.

Значення зміщень зберігаються у файлі VAR, який запитується файлом INI під час запуску LinuxCNC. У наведеному нижче прикладі, де використовується G55, положення кожної осі для початку координат G55 зберігається у пронумерованій змінній.

У схемі VAR-файлу перша змінна номер зберігає зміщення X, друга – зміщення Y і так далі для всіх дев’яти осей. Існують такі пронумеровані набори для кожного зі зміщень системи координат.

Кожен з графічних інтерфейсів має спосіб встановлення значень для цих зміщень. Ви також можете встановити ці значення, редагуючи сам файл VAR, а потім перезапустивши LinuxCNC,

щоб LinuxCNC прочитав нові значення, однак це не є рекомендованим способом. Використання G10, G52, G92, G28.1 тощо є кращими способами встановлення змінних. У нашому прикладі ми безпосередньо відредагуємо файл, щоб G55 приймав такі значення:

Table 11.1: Приклад параметрів G55

Вісь	Змінна	Значення
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

Це слід інтерпретувати як зміщення нульових позицій G55 на X = 2 одиниці, Y = 1 одиниця та Z = -2 одиниці від абсолютного нуля.

Після присвоєння значень виклик G55 у блоці програми змістить нульову точку відліку на значення, що зберігаються. Наступний рядок перемістить кожну вісь у нове нульове положення. На відміну від G53, G54–G59.3 є модальними командами. Вони будуть діяти на всі блоки коду після того, як одна з них буде встановлена. Програма, яка може виконуватися з використанням зміщень кріплення, вимагатиме лише однієї координатної точки відліку для кожного з місць і всієї роботи, яка має бути виконана там. Наступний код наводиться як приклад створення квадрата з використанням зміщень G55, які ми встановили вище.

```
G55 ; b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b''вb''b ↩
    ''аb''b''тb''b''иb'' b''сb''b''иb''b''сb''b''тb''b''еb''b''мb''b''уb'' b''кb''b''оb''b' ↩
    'оb''b''рb''b''дb''b''иb''b''нb''b''аb''b''тb'' 2
G0 X0 Y0 Z0
G1 F2 Z-0.2000
X1
Y1
X0
Y0
G0 Z0
G54 ; b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b''вb''b ↩
    ''аb''b''тb''b''иb'' b''сb''b''иb''b''сb''b''тb''b''еb''b''мb''b''уb'' b''кb''b''оb''b' ↩
    'оb''b''рb''b''дb''b''иb''b''нb''b''аb''b''тb'' 1
G0 X0 Y0 Z0
M2
```

У цьому прикладі G54 біля кінця залишає систему координат G54 з усіма нульовими зміщеннями, так що існує модальний код для абсолютних положень осей на основі верстата. Ця програма припускає, що ми це зробили, і використовує кінцеву команду як команду для обробки нуля. Можна було б використовувати G53 і дійти до того ж результату, але ця команда не була б модальною, і будь-які команди, видані після неї, повернулися б до використання зміщень G55, оскільки ця система координат все ще була б чинною.

[source,”]

```
G54     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 1
```

```
G55     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 2
G56     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 3
G57     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 4
G58     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 5
G59     b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 6
G59.1   b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 7
G59.2   b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 8
G59.3   b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b' ↩
    'єb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb''b''иb'' b''сb''b''иb''b' ↩
    'сb''b''тb''b''еb''b''мb''b''иb'' b''кb''b''оb''b''оb''b''рb''b''дb''b''иb''b''нb''b' ↩
    'аb''b''тb'' 9
```

11.1.3.1 Система координат за замовчуванням

Ще одна змінна у файлі VAR стає важливою, коли ми думаємо про системи зміщення. Ця змінна має ім'я 5220. У файлах за замовчуванням її значення встановлено на 1.00000. Це означає, що при запуску LinuxCNC повинен використовувати першу систему координат як стандартну. Якщо ви встановите це значення на 9.00000, то при запуску і скиданні налаштувань буде використовуватися дев'ята система зміщення як стандартна. Будь-яке значення, відмінне від цілого числа (насправді десяткового) від 1 до 9, або відсутність змінної 5220 призведе до того, що LinuxCNC повернеться до значення за замовчуванням 1.00000 під час запуску.

11.1.3.2 Налаштування зміщень системи координат

Команду G10 L2x можна використовувати для встановлення зміщень системи координат:

- *G10 L2 P(1-9)* - Встановити зміщення на значення. Поточна позиція не має значення (див. G10 L2 для отримання детальнішої інформації).
- *G10 L20 P(1-9)* - Встановити зміщення (зміщення), щоб поточна позиція стала значенням (див. G10 L20 для отримання детальнішої інформації).

Note
Ми наводимо тут лише короткий огляд, повний опис дивіться в розділах G-коду.

11.1.4 Локальні та глобальні зміщення

11.1.4.1 Команда G52

G52 використовується в програмі обробки деталі як тимчасове «зсунення локальної системи координат» в системі координат заготовки. Прикладом використання є обробка декількох однакових елементів в різних місцях деталі. Для кожного елемента *G52* програмує локальну базову точку в координатах заготовки, і викликається підпрограма для обробки елемента відносно цієї точки.

Зсуви осі «G52» програмуються відносно зсувів координат заготовки «G54» до «G59.3». Як локальний зсув, «G52» застосовується після зсуву заготовки, включаючи обертання. Таким чином, елемент деталі буде оброблятися однаково на кожній деталі, незалежно від орієнтації деталі на палеті.



Caution
Як тимчасове зміщення, встановлене та скасоване в локалізованій області дії програми обробки деталі, в інших інтерпретаторах G-коду «G52» не зберігається після перезавантаження верстата, «M02» або «M30». У LinuxCNC «G52» має спільні параметри з «G92», який, з історичних причин, **зберігає** ці параметри. Див. G92 Persistence Cautions нижче.



Caution
«G52» і «G92» мають однакові регістри зміщення. Тому встановлення «G52» замінить будь-яке попереднє встановлення «G92», а «G52» збережеться після перезавантаження машини, якщо ввімкнено збереження «G92». Ці взаємодії можуть призвести до несподіваних зміщень. Див. Застереження щодо взаємодії G92 і G52 нижче.

Програмування «G52 X1 Y2» зміщує поточну систему координат заготовки по осі X на 1 і по осі Y на 2. Відповідно, на цифровому індикаторі поточні координати X і Y положення інструменту будуть зменшені на 1 і 2 відповідно. Осі, не вказані в команді, такі як Z у попередньому прикладі, не будуть змінені: будь-яке попереднє зміщення Z «G52» залишиться в силі, а в іншому випадку зміщення Z буде дорівнювати нулю.

Тимчасове локальне зміщення можна скасувати за допомогою *G52 X0 Y0*. Будь-які осі, які не обнулені явно, збережуть попереднє зміщення.

G52 використовує ті самі регістри зміщення, що й «G92», і тому «G52» видно на DRO та в попередньому перегляді з позначкою «G92».

11.1.5 Зміщення осей G92

G92 — це найбільш неправильно зрозуміла і найрозумніша команда, яку можна запрограмувати за допомогою LinuxCNC. Спосіб її роботи дещо змінився між першими версіями і поточною. Ці зміни, без сумніву, збентежили багатьох користувачів. Їх слід розглядати як команду, що створює тимчасове зміщення, яке застосовується до всіх інших зміщень.

11.1.5.1 Команди G92

«G92» зазвичай використовується у двох концептуально різних значеннях: як «зміщення глобальної системи координат» або як «зміщення локальної системи координат».

Набір команд «G92» включає:

- *G92* - Ця команда, якщо її використовувати з іменами осей, встановлює значення для змінних зміщення.
- *G92.1* - Ця команда встановлює нульові значення для змінних G92.
- *G92.2* - Ця команда призупиняє, але не обнуляє змінні G92.
- *G92.3* - Ця команда застосовує значення зміщення, які були призупинені.

Як глобальне зміщення, «G92» використовується для зміщення всіх систем координат заготовки «G54» до «G59.3». Прикладом використання є обробка декількох однакових деталей в пристосуваннях з відомими розташуваннями на палеті, але розташування палети може змінюватися між циклами або між верстатами. Кожне зміщення кріплення відносно опорної точки на палеті заздалегідь встановлюється в одній із систем координат заготовки, від «G54» до «G59.3», а «G92» використовується для «дотику» до опорної точки палети. Потім для кожної деталі вибирається відповідна система координат заготовки і виконується програма обробки деталі.

Note
Поворот системи координат заготовки «G10 R-» є специфічним для інтерпретатора «rs274ngc», а зміщення «G92» застосовується «після» повороту. При використанні «G92» як глобального зміщення повороти системи координат заготовки можуть мати несподівані результати.

Як локальна система координат, «G92» використовується як тимчасове зміщення в системі координат заготовки. Прикладом використання є обробка деталі з декількома ідентичними елементами в різних місцях. Для кожного елемента «G92» використовується для встановлення локальної точки відліку, а для обробки елемента, починаючи з цієї точки, викликається підпрограма.

Note
Не рекомендується використовувати «G92» для програмування з локальними системами координат у програмі обробки деталі. Замість цього див. «G52», локальну систему координат, яка є більш інтуїтивною, коли відоме бажане зміщення відносно заготовки, але поточне положення інструменту може бути невідомим.

Програмування *G92 X0 Y0 Z0* встановлює поточне положення інструменту на координати X0, Y0 та Z0 без руху. G92 **не** працює з абсолютними координатами верстата. Він працює з **поточного положення**.

«G92» також працює з поточного місця розташування, зміненого будь-якими іншими зміщеннями, які діють під час виклику команди «G92». Під час тестування відмінностей між робочими зміщеннями та фактичними зміщеннями було виявлено, що зміщення «G54» може скасувати «G92» і, таким чином, створити враження, що жодні зміщення не діють. Однак «G92» все ще діяла для всіх координат і створювала очікувані робочі зміщення для інших систем координат.

За замовчуванням, зміщення «G92» відновлюються після запуску верстата. Програмісти, які бажають отримати поведінку Fanuc, де зміщення «G92» очищаються при запуску верстата, після перезавантаження або закінчення програми, можуть вимкнути збереження «G92», встановивши «DISABLE_G92_PERSISTENCE = 1» в розділі «[RS274NGC]» файлу INI.

Note
Рекомендується очищати зміщення «G92» після їх використання за допомогою «G92.1» або «G92.2». При запуску LinuxCNC з увімкненою функцією збереження «G92» (за замовчуванням) будь-які зміщення в змінних «G92» будуть застосовані при поверненні осі в початкове положення. Див. Застереження щодо збереження G92 нижче.

11.1.5.2 Встановлення значень G92

Існує щонайменше два способи встановлення значень G92:

- Клацнувши правою кнопкою миші на відображенні позиції в tklinuxcnc, відкриється вікно, де можна ввести значення.
- За допомогою команди G92

Обидва працюють з поточного положення осі, яку потрібно перемістити.

Програмування «G92 X Y Z A B C U V W» встановлює значення змінних G92 таким чином, що кожна вісь приймає значення, пов’язане з її назвою. Ці значення присвоюються поточній позиції осей. Ці результати відповідають пунктам 1 і 2 документа NIST.

Команди G92 працюють з поточного положення осі та додають і віднімають правильно, щоб надати поточній позиції осі значення, призначене командою G92. Ефекти працюють, навіть якщо попередні зміщення є.

Отже, якщо вісь X наразі показує 2,0000 як своє положення, команда «G92 X0» встановить зміщення -2,0000, щоб поточне положення X стало нульовим. Команда «G92 X2» встановить зміщення 0,0000, і відображуване положення не зміниться. Команда «G92 X5.0000» встановить зміщення 3,0000, так що поточна відображувана позиція стане 5,0000.

11.1.5.3 Застереження щодо стійкості G92

За замовчуванням значення зміщення *G92* будуть збережені у файлі VAR та відновлені після скидання налаштувань або запуску верстата.

Параметри G92:

- 5210 - Прапор увімкнення/вимкнення (1.0/0.0)
- 5211 - Зміщення осі X
- 5212 - Зміщення осі Y
- 5213 - Зміщення осі Z
- 5214 - Зсув осі A
- 5215 - Зміщення осі B
- 5216 - Зміщення осі C
- 5217 - Зміщення осі U
- 5218 - Зміщення осі V
- 5219 - Зміщення осі W

де 5210 — це прапор увімкнення «G92» (1 для увімкнення, 0 для вимкнення), а 5211–5219 — це зміщення осей. Якщо ви бачите несподівані позиції в результаті заданого руху, що є наслідком збереження зміщення в попередній програмі та їхнього незнищення в кінці, видайте команду G92.1 у вікні MDI, щоб знищити збережені зміщення.

Якщо при запуску LinuxCNC у файлі VAR містяться значення G92, то значення G92 у файлі var будуть застосовані до значень поточного положення кожної осі. Якщо це вихідне положення і вихідне положення встановлено як нуль верстата, все буде правильно. Після встановлення вихідного положення за допомогою реальних перемикачів верстата або шляхом переміщення кожної осі у відоме вихідне положення і видачі команди вихідного положення осі, будуть застосовані

будь-які зміщення G92. Якщо під час повернення осі X у вихідне положення діє G92 X1, DRO покаже «X: 1.000» замість очікуваного «X: 0.000», оскільки G92 було застосовано до початку координат верстата. Якщо ви видаєте G92.1, а DRO тепер показує всі нулі, це означає, що під час останнього запуску LinuxCNC діяло зміщення G92.

Якщо ви не маєте наміру використовувати ті самі зміщення G92 у наступній програмі, найкращою практикою є видання коду G92.1 в кінці будь-яких файлів G-коду, де ви використовуєте зміщення G92.

Коли програма переривається під час обробки, в якій діють зміщення «G92», запуск призведе до їх повторного активування. В якості запобіжного заходу завжди використовуйте преамбулу для налаштування середовища відповідно до ваших очікувань. Крім того, збереження «G92» можна вимкнути, встановивши «DISABLE_G92_PERSISTENCE = 1» у розділі «[RS274NGC]» файлу INI.

11.1.5.4 Застереження щодо взаємодії G92 та G52

«G52» і «G92» мають однакові регістри зміщення. Якщо в файлі INI не вимкнено збереження «G92» (див. Команди G92), зміщення «G52» також зберігаються після перезавантаження верстата, «M02» або «M30». Зверніть увагу, що зміщення «G52», яке діє під час переривання програми, може призвести до небажаних зміщень під час виконання наступної програми. Див. Застереження щодо збереження G92 вище.

11.1.6 Приклади програм із використанням зміщень

11.1.6.1 Приклад програми з використанням зміщень координат заготовки

У цьому зразку гравірувального проєкту фрезерується набір з чотирьох кіл радіусом 0,1 приблизно у формі зірки навколо центрального кола. Ми можемо налаштувати окремий візерунок кола ось так.

```
G10 L2 P1 X0 Y0 Z0 (b''пb''b''eb''b''pb''b''eb''b''кb''b''ob''b''нb''b''ab''b''йb''b''тb''b ↩
    ''eb''b''cb''b''яb'', b''щb''b''ob'' G54 b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b' ↩
    'вb''b''лb''b''eb''b''нb''b''ob'' b''нb''b''ab'' b''нb''b''уb''b''лb''b''ьb''b''ob''b' ↩
    'вb''b''иb''b''йb'' b''cb''b''тb''b''ab''b''нb''b''ob''b''кb'')
G0 X-0.1 Y0 Z0
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

Ми можемо видати набір команд для створення зміщень для чотирьох інших кіл, подібних до цього.

```
G10 L2 P2 X0.5 (b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb'' G55 X b''нb''b''ab'' 0.5 b''дb''b''юb''b''йb''b ↩
    ''мb''b''ab'')
G10 L2 P3 X-0.5 (b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb'' G56 X b''нb''b''ab'' -0.5 b''дb''b''юb''b''йb'' ↩
    b''мb''b''ab'')
G10 L2 P4 Y0.5 (b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb'' G57 Y b''нb''b''ab'' 0.5 b''дb''b''юb''b''йb''b ↩
    ''мb''b''ab'')
G10 L2 P5 Y-0.5 (b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''нb''b''яb'' G58 Y b''нb''b''ab'' -0.5 b''дb''b''юb''b''йb'' ↩
    b''мb''b''ab'')
```

Ми об’єднали їх у наступній програмі:

```
(b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ab'' b''дb''b''лb''b''яb'' b''фb''b' ↩
    'pb''b''eb''b''зb''b''eb''b''pb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''пb'''b' ↩
    'яb''b''тb''b''иb'' b''мb''b''ab''b''лb''b''eb''b''нb''b''ьb''b''кb''b''иb''b''хb'' b' ↩
    'кb''b''іb''b''лb'' b''уb'' b''фb''b''ob''b''pb''b''мb''b''іb'' b''pb''b''ob''b''мb''b' ↩
    '6b''b''ab'')

G10 L2 P1 X0 Y0 Z0 (b''пb''b''eb''b''pb''b''eb''b''кb''b''ob''b''нb''b''ab''b''йb''b''тb''b ↩
    ''eb''b''cb''b''яb'', b''щb''b''ob'' G54 b''єb'' b''нb''b''уb''b''лb''b''ьb''b''ob''b' ↩
    'вb''b''ob''b''юb'' b''тb''b''ob''b''чb''b''кb''b''ob''b''юb'' b''вb''b''eb''b''pb''b' ↩
    'cb''b''тb''b''ab''b''тb''b''ab'')
G10 L2 P2 X0.5 (b''зb''b''cb''b''уb''b''вb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''нb''b''яb'' G55 X b''нb''b''ab'' 0,5 b''дb''b''юb''b''йb''b''мb''b''ab'')
G10 L2 P3 X-0.5 (b''зb''b''cb''b''уb''b''вb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''нb''b''яb'' G56 X b''нb''b''ab'' -0,5 b''дb''b''юb''b''йb''b''мb''b' ↩
    'ab'')
G10 L2 P4 Y0.5 (b''зb''b''cb''b''уb''b''вb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''нb''b''яb'' G57 Y b''нb''b''ab'' 0,5 b''дb''b''юb''b''йb''b''мb''b''ab'')
G10 L2 P5 Y-0.5 (b''зb''b''cb''b''уb''b''вb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''нb''b''яb'' G58 Y b''нb''b''ab'' -0,5 b''дb''b''юb''b''йb''b''мb''b' ↩
    'ab'')

G54 G0 X-0.1 Y0 Z0 (b''цb''b''eb''b''нb''b''тb''b''pb''b''ab''b''лb''b''ьb''b''нb''b''eb''  ↩
    b''кb''b''ob''b''лb''b''ob'')
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G55 G0 X-0.1 Y0 Z0 (b''пb''b''eb''b''pb''b''шb''b''eb'' b''зb''b''мb''b''іb''b''щb''b''eb'' ↩
    b''нb''b''eb'' b''кb''b''ob''b''лb''b''ob'')
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G56 G0 X-0.1 Y0 Z0 (b''дb''b''pb''b''уb''b''гb''b''eb'' b''зb''b''мb''b''іb''b''щb''b''eb'' ↩
    b''нb''b''eb'' b''кb''b''ob''b''лb''b''ob'')
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G57 G0 X-0.1 Y0 Z0 (b''тb''b''pb''b''eb''b''тb''b''єb'' b''зb''b''мb''b''іb''b''щb''b''eb'' ↩
    b''нb''b''eb'' b''кb''b''ob''b''лb''b''ob'')
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G58 G0 X-0.1 Y0 Z0 (b''чb''b''eb''b''тb''b''вb''b''eb''b''pb''b''тb''b''eb'' b''зb''b''мb'' ↩
    b''іb''b''щb''b''eb''b''нb''b''eb'' b''кb''b''ob''b''лb''b''ob'')
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0

M2
```

Тепер настав час застосувати набір зміщень G92 до цієї програми. Ви побачите, що вона працює в кожному випадку на Z0. Якщо фрезерний верстат перебував у нульовому положенні, команда G92 Z1.0000, видана на початку програми, змістила б все на один дюйм. Ви також можете змістити весь малюнок у площині XY, додавши деякі зміщення X і Y за допомогою G92. Якщо ви це зробите, вам слід додати команду G92.1 безпосередньо перед M2, яка завершує програму. Якщо ви цього не зробите, інші програми, які ви можете запустити після цієї, також використовуватиму це зміщення G92. Крім того, це збереже значення G92 при вимкненні LinuxCNC, і вони будуть

відновлені при повторному запуску.

11.1.6.2 Приклад програми з використанням зміщень G52

(Має бути написано)

11.2 Компенсація інструменту

11.2.1 Дотик вимкнено

За допомогою сенсорного екрана в інтерфейсі AXIS ви можете автоматично оновлювати таблицю інструментів.

Типові кроки для оновлення таблиці інструментів:

- Після повернення до початкового положення завантажте інструмент з *T*n *M6*, де *n* – номер інструменту.
- Перемістіть інструмент у встановлену точку за допомогою калібру або зробіть пробний розріз та виміряйте.
- Натисніть кнопку «Touch Off» на вкладці «Ручне керування» (або натисніть кнопку «End» на клавіатурі).
- Виберіть «Таблиця інструментів» у розкривному списку «Система координат».
- Введіть калібр або виміряний розмір і натисніть кнопку «OK».

Таблиця інструментів буде змінена з правильною довжиною Z, щоб DRO відображав правильне положення Z, і буде видана команда G43, щоб нова довжина Z інструменту набула чинності. Відключення таблиці інструментів доступне тільки тоді, коли інструмент завантажений з *T*n *M6*.

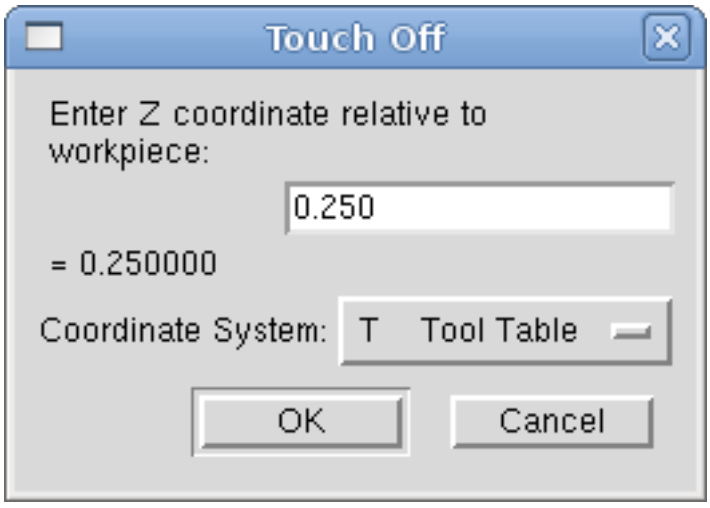


Figure 11.2: Таблиця інструментів для відключення дотику

11.2.1.1 Використання G10 L1/L10/L11

Команди G10 L1/L10/L11 можна використовувати для встановлення зміщень таблиці інструментів:

- `G10 L1 P__n__` – Встановити зміщення на значення. Поточна позиція не має значення (див. G10 L1 для отримання детальнішої інформації).
- `G10 L10 P__n__` - Встановіть зміщення (зміщення), щоб поточна позиція з приладами 1-8 стала значенням (див. G10 L10 для отримання детальнішої інформації).
- `G10 L11 P__n__` - Встановіть зміщення (зміщення), щоб поточна позиція з фіксатором 9 стала значенням (див. G10 L11 для отримання детальнішої інформації).

Note
Це лише короткий огляд, зверніться до довідника G-коду для отримання детальніших пояснень.

11.2.2 Таблиця інструментів

«Таблиця інструментів» — це текстовий файл, що містить інформацію про кожен інструмент. Файл знаходиться в тому ж каталозі, що й ваша конфігурація, і за замовчуванням має назву «tool.tbl». Ім’я файлу можна вказати за допомогою параметра INI-файлу [EMCIO]TOOL_TABLE. Інструменти можуть знаходитися в пристрої для зміни інструментів або бути змінені вручну. Файл можна редагувати за допомогою текстового редактора або оновлювати за допомогою G10 L1. Приклад формату таблиці інструментів токарного верстата див. у розділі Таблиця інструментів токарного верстата. Максимальна кількість кишень — 1000.

Для редагування таблиці інструментів можна використовувати Tool Editor або текстовий редактор. Якщо ви використовуєте текстовий редактор, переконайтеся, що ви перезавантажили таблицю інструментів у графічному інтерфейсі.

11.2.2.1 Формат таблиці інструментів

.Формат таблиці інструментів

T#	P#	X	Y	Z	A	B	C	U	V	W	Подорожі	Fwd	BA	Ori	Rem
;	(немає даних після крапки з комою)														
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

Загалом, формат рядка таблиці інструментів такий:

- T - номер інструменту (номери інструментів мають бути унікальними)
- P - номер гнізда, 1-1000 (номери гнізд мають бути унікальними, гніздо 0 представляє шпиндель)
- X..W - зміщення інструменту на заданій осі - з плаваючою комою
- D - діаметр інструменту - число з плаваючою комою, абсолютне значення
- I - передній кут (лише токарний верстат) - з плаваючою комою
- J - кут задньої частини (лише токарний верстат) - з плаваючою комою
- Q - орієнтація інструменту (лише токарний верстат) - ціле число, 0-9

- ; - початок коментаря або зауваження - текст

Номери інструментів мають бути унікальними. Рядки, що починаються з крапки з комою, ігноруються.

Одиниці вимірювання довжини, діаметра тощо - це машинні одиниці.

Ймовірно, ви захочете зберегти записи інструментів у порядку зростання, особливо якщо ви збираєтеся використовувати випадковий змінювач інструментів. Хоча таблиця інструментів дозволяє розміщувати номери інструментів у будь-якому порядку.

Один рядок може містити до 16 записів, але, швидше за все, їх буде набагато менше. Записи для T (номер інструменту) і P (номер кишені) є обов'язковими. Останній запис (примітка або коментар, що передує крапці з комою) є необов'язковим. Читати легше, якщо записи розташовані в стовпцях, як показано в таблиці, але єдиною вимогою до формату є наявність принаймні одного пробілу або табуляції після кожного запису в рядку та символу нового рядка в кінці кожного запису.

Значення записів та тип даних, які потрібно ввести в кожен з них, наведені нижче.

Номер інструменту (обов'язково)
: У стовпці «T» міститься число (ціле число без знака), яке представляє кодовий номер інструменту. Користувач може використовувати будь-який код для будь-якого інструменту, якщо ці коди є цілими числами без знака.

Номер кишені (обов'язково)
: У стовпці «P» міститься число (ціле без знака), яке позначає номер кишені (слота) змінювача інструментів, де знаходиться інструмент. Всі записи в цьому стовпці повинні бути різними.

Номери кишень зазвичай починаються з 1 і доходять до найвищого доступного номера кишені на вашому пристрої для зміни інструментів. Але не всі пристрої для зміни інструментів дотримуються цієї схеми. Номери кишень визначаються номерами, які використовує ваш пристрій для зміни інструментів для позначення кишень. Отже, все це означає, що номери кишень, які ви використовуєте, визначаються схемою нумерації, яка використовується у вашому пристрої для зміни інструментів, і номери кишень, які ви використовуєте, повинні мати сенс на вашій машині.

Номери зміщення даних (необов'язково)
: Стовпці «Зсув даних» (XYZABCUVW) містять дійсні числа, які представляють зсуви інструменту по кожній осі. Це число буде використовуватися, якщо використовуються зсуви довжини інструменту і цей інструмент вибраний. Ці числа можуть бути додатними, нульовими або від'ємними і, фактично, є повністю необов'язковими. Хоча, ймовірно, ви захочете зробити тут хоча б один запис, інакше не буде сенсу вносити запис в таблицю інструментів.

У типовому токарному верстаті, ймовірно, вам знадобиться введення для Z (зсув довжини інструменту)
У типовому токарному верстаті, ймовірно, вам знадобиться ввести значення X (зсув інструменту по осі X) і Z (зсув інструменту по осі Z). У типовому фрезерному верстаті, що використовує компенсацію діаметра фрези (компенсація фрези), вам, ймовірно, також знадобиться додати значення D (діаметр фрези). У типовому токарному верстаті, що використовує компенсацію діаметра кінчика інструменту (компенсація інструменту), вам, ймовірно, також знадобиться додати значення D (діаметр кінчика інструменту).

Токарний верстат також потребує додаткової інформації для опису форми та орієнтації інструменту. Тому, ймовірно, вам знадобляться записи для I (кут нахилу інструменту вперед) та J (кут нахилу інструменту назад). Ймовірно, вам також знадобиться запис для Q (орієнтація інструменту).

Див. розділ Інформація для користувача токарного верстата для отримання додаткової інформації.

Стовпець «Діаметр» містить дійсне число. Це число використовується тільки в тому випадку, якщо компенсація різака ввімкнена за допомогою цього інструменту. Якщо запрограмований шлях під час компенсації є краєм матеріалу, що ріжеться, це повинно бути додатне дійсне число,

що представляє виміряний діаметр інструменту. Якщо запрограмований шлях під час компенсації є шляхом інструменту, діаметр якого є номінальним, це має бути невелике число (додатне або від'ємне, але близьке до нуля), що представляє лише різницю між виміряним діаметром інструменту та номінальним діаметром. Якщо компенсація різака не використовується з інструментом, не має значення, яке число знаходиться в цій колонці.

Колонку «Коментар» можна використовувати для опису інструменту. Підходить будь-який тип опису. Ця колонка призначена лише для читачів-людей. Перед коментарем має стояти крапка з комою.

Note
У попередніх версіях LinuxCNC було два різних формати таблиці інструментів для фрезерних та токарних верстатів, але починаючи з версії 2.4.x, для всіх верстатів використовується один формат таблиці інструментів.

11.2.2.2 Інструмент IO

Програма **iocontrol**, що не працює в режимі реального часу, зазвичай використовується для управління зміною інструментів (та іншими функціями вводу-виводу для активації LinuxCNC і управління обладнанням для охолодження). Контакти HAL, що використовуються для управління інструментами, мають префікс **iocontrol.0.**.

Команда G-коду **T** активує вихідний контакт HAL `iocontrol.0.tool-prepare`. Вхідний контакт HAL, `iocontrol.0.tool-prepared`, повинен бути встановлений зовнішньою логікою HAL для завершенн підготовки інструменту, що призводить до подальшого скидання контакту підготовки інструменту.

Команда G-коду **M6** активує вихідний контакт HAL `iocontrol.0.tool-change`. Відповідний вхідний контакт HAL, `iocontrol.0.tool-prepared`, повинен бути встановлений зовнішньою логікою HAL для вказівки на завершення заміни інструменту, що призводить до подальшого скидання контакту заміни інструменту.

Доступ до даних інструменту здійснюється за допомогою впорядкованого індексу (idx), який залежить від типу змінного пристрою інструменту, визначеного параметром `[EMCIO]RANDOM_TOOLCHANG`

1. Для `RANDOM_TOOLCHANGER = 0` (0 є значенням за замовчуванням і вказує на невипадковий змінник інструментів), idx — це число, що вказує на послідовність завантаження даних інструменту.
2. Для `RANDOM_TOOLCHANGER = 1`, idx – це **поточний** номер гнізда для номера інструмента, заданого командою вибору інструмента G-коду **T*n***.

Програма вводу/виводу забезпечує вихідні контакти HAL для полегшення керування пристроєм зміни інструменту:

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. Номер інструменту *n*==0 вказує на відсутність інструменту.
2. Номер гнізда для інструмента встановлюється під час завантаження/перезавантаження даних інструменту з його джерела даних ([EMCIO]TOOL_TABLE або [EMCIO]DB_PROGRAM).
3. У команді G-коду **T*n*** (*n* != 0):

a. **iocontrol.0.tool-prep-index** = *idx* (індекс на основі послідовності завантаження даних інструменту)
b. **iocontrol.0.tool-prep-number** = *n*
c. **iocontrol.0.tool-prep-pocket** = номер кишені для *n*

4. At G-code **T0** (*n* == 0 вилучено) команда:

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** (після зміни виводу 0-->1 у файлі iocontrol.0.tool):

a. **iocontrol.0.tool-from-pocket** = номер кишені, що використовується для вилучення інструмен

1. Номер інструменту *n*==0 **не є спеціальним**.
2. Кишеня номер 0 є **особливою**, оскільки вона вказує на **шпиндель**.
3. **Поточний** номер гнізда для інструмента *n* – це індекс даних інструмента (*idx*) для інструмента *n*.
4. За командою G-коду **T*n***:

a. **iocontrol.0.tool-prep-index** = індекс даних інструменту (idx) для інструменту *n*
b. **iocontrol.0.tool-prep-number** = *n*
c. **iocontrol.0.tool-prep-pocket** = номер гнізда для інструменту *n*

5. At M-code **M6** (після зміни виводу 0-->1 у файлі iocontrol.0.tool):

a. **iocontrol.0.tool-from-pocket** = номер кишені, що використовується для вилучення інструмен

Note
При запуску **iocontrol.0.tool-from-pocket** = 0. Команда M61Q*n* (*n*!=0) не змінює **iocontrol.0.tool-from-pocket**. Команда M61Q0 (*n*==0) встановлює **iocontrol.0.tool-from-pocket** на 0.

11.2.2.3 Змінювачі інструментів

LinuxCNC підтримує три типи змінників інструментів: «ручний», «випадкове розташування» та «невипадкове або фіксоване розташування». Інформація про налаштування змінника інструментів LinuxCNC міститься в розділі EMCIO Section глави INI.

Ручний змінник інструментів Ручний змінювач інструментів (ви змінюєте інструмент вручну) розглядається як змінювач інструментів з фіксованим розташуванням. Ручна зміна інструментів може бути полегшена за допомогою конфігурації HAL, яка використовує нереальну програму **hal_manualtoolchange** і зазвичай вказується в файлі INI за допомогою операторів INI:

```
[HAL]
HALFILE = axis_manualtoolchange.hal
```

Пристрої зміни інструментів з фіксованим розташуванням Змінювачі інструментів з фіксованим розташуванням завжди повертають інструменти у фіксоване положення в змінювачі інструментів. Сюди також входять конструкції, такі як токарні револьверні головки. Коли LinuxCNC налаштований для змінювача інструментів з фіксованим розташуванням, номер «P» не використовується внутрішньо (але зчитується, зберігається і переписується) LinuxCNC, тому ви можете використовувати P для будь-якого номера обліку, який ви хочете.

Note
При використанні [EMCIO]RANDOM_TOOLCHANGER = 0 (за замовчуванням) номер кишені «P» є параметром даних інструменту, який отримується з джерела даних інструменту ([EMCIO]TOOL_TABLE або [EMCIO]DB_PROGRAM). У багатьох додатках він є фіксованим, але його можна змінити шляхом редагування [EMCIO]TOOL_TABLE або програмно, коли використовується [EMCIO]DB_PROGRAM. LinuxCNC надсилає оновлення до джерела даних ([EMCIO]TOOL_TABLE або [EMCIO]DB_PROGRAM) для G-кодів G10L1, G10L10, G10L11, M61. LinuxCNC може витягувати оновлення даних інструментів із джерела даних за допомогою команд інтерфейсу користувача (приклад на Python: `linuxcnc.command().load_tool_table()`) або за допомогою G-коду: G10L0.

Зміни інструментів випадкового розташування Випадкові змінювачі інструментів ([EMCIO]RANDOM = 1) замінюють інструмент у шпинделі на інструмент у змінювачі. З цим типом змінювача інструментів інструмент завжди буде знаходитися в іншому гнізді після заміни інструменту. Коли інструмент замінюється, LinuxCNC переписує номер гнізда, щоб відстежувати, де знаходяться інструменти. T може бути будь-яким числом, але P має бути числом, яке має сенс для верстата.

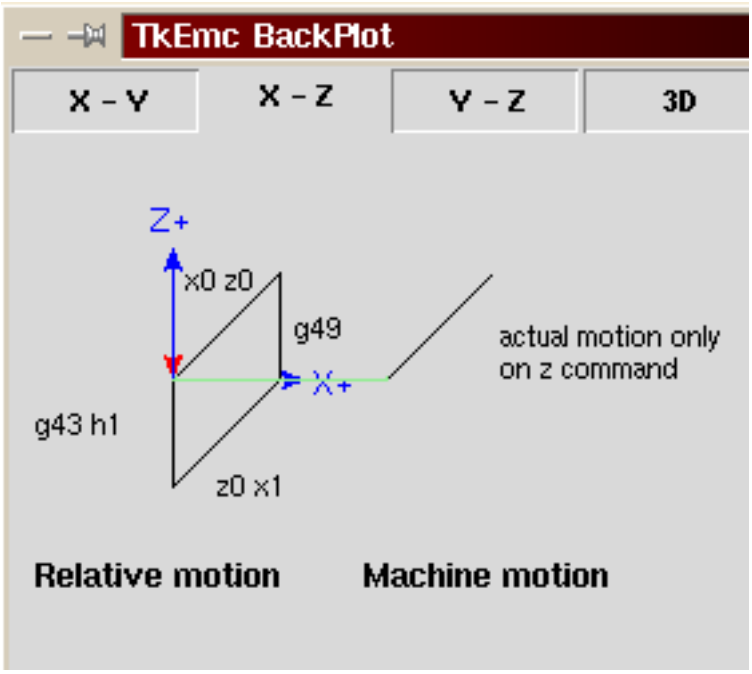
11.2.3 Компенсація довжини інструменту

Компенсації довжини інструменту вказані у вигляді додатних чисел у таблиці інструментів. Компенсац інструменту програмується за допомогою G43 H_n_, де *n* — індекс бажаного інструменту в таблиці інструментів. Всі записи в таблиці інструментів повинні бути додатними. Значення H перевіряється, воно повинно бути невід'ємним цілим числом при зчитуванні. Інтерпретатор працює наступним чином:

1. Якщо G43 H*n* запрограмовано, виконується дзвінка до функції USE_TOOL_LENGTH_OFFSET(`__lengt (де *length* є різницею довжини, прочитаною з таблиці інструментів, індексованого інструменту *n*), tool_length_offset перепоставляється в моделі налаштувань машини та значення current_z у моделі коригується. Зверніть увагу, що *n* не обов'язково має бути таким самим, як номер слоту інструменту в поточному спинділі.
2. Якщо запрограмовано G49, викликається USE_TOOL_LENGTH_OFFSET(0.0), tool_length_offset скидається до 0.0 у шаблоні налаштувань верстата, а поточне значення current_z у моделі коригується. Ефект компенсації довжини інструменту проілюстровано на знімку нижче. Зверніть увагу, що довжина інструменту віднімається від Z, щоб запрограмована контрольна точка відповідала кінчику інструменту. Зверніть також увагу, що ефект компенсації довжини є миттєвим, коли ви бачите, що компенсація є миттєвою, коли положення Z розглядається як відносна координата, але це не впливає на фактичне положення верстата, поки не буде запрограмовано рух Z.

Програма випробування довжини інструменту. Інструмент №1 має довжину один дюйм.

```
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
```



За допомогою цієї програми, у більшості випадків, верстат застосовуватиме зміщення у вигляді рампи під час руху по осях xyz після слова G43.

11.2.4 Компенсація радіуса різця

Компенсація різака дозволяє програмісту програмувати траєкторію інструменту, не знаючи точного діаметра інструменту. Єдиним застереженням є те, що програміст повинен запрограмувати рух введення так, щоб він був принаймні таким же довгим, як найбільший радіус інструменту, який може бути використаний.

Існує два можливих шляхи, якими може рухатися різак, оскільки компенсація різака може бути зліва або справа від лінії, якщо дивитися на напрямок руху різака ззаду. Щоб уявити це, уявіть, що ви стоїте на деталі і йдете за інструментом, який рухається по деталі. G41 — це ліва сторона лінії, а G42 — права сторона лінії.

Кінцева точка кожного руху залежить від наступного руху. Якщо наступний рух створює зовнішній кут, рух буде до кінцевої точки компенсованої лінії різання. Якщо наступний рух створює внутрішній кут, рух зупиниться раніше, щоб не пошкодити деталь. На наступному малюнку показано, як компенсований рух зупиняється в різних точках залежно від наступного руху.

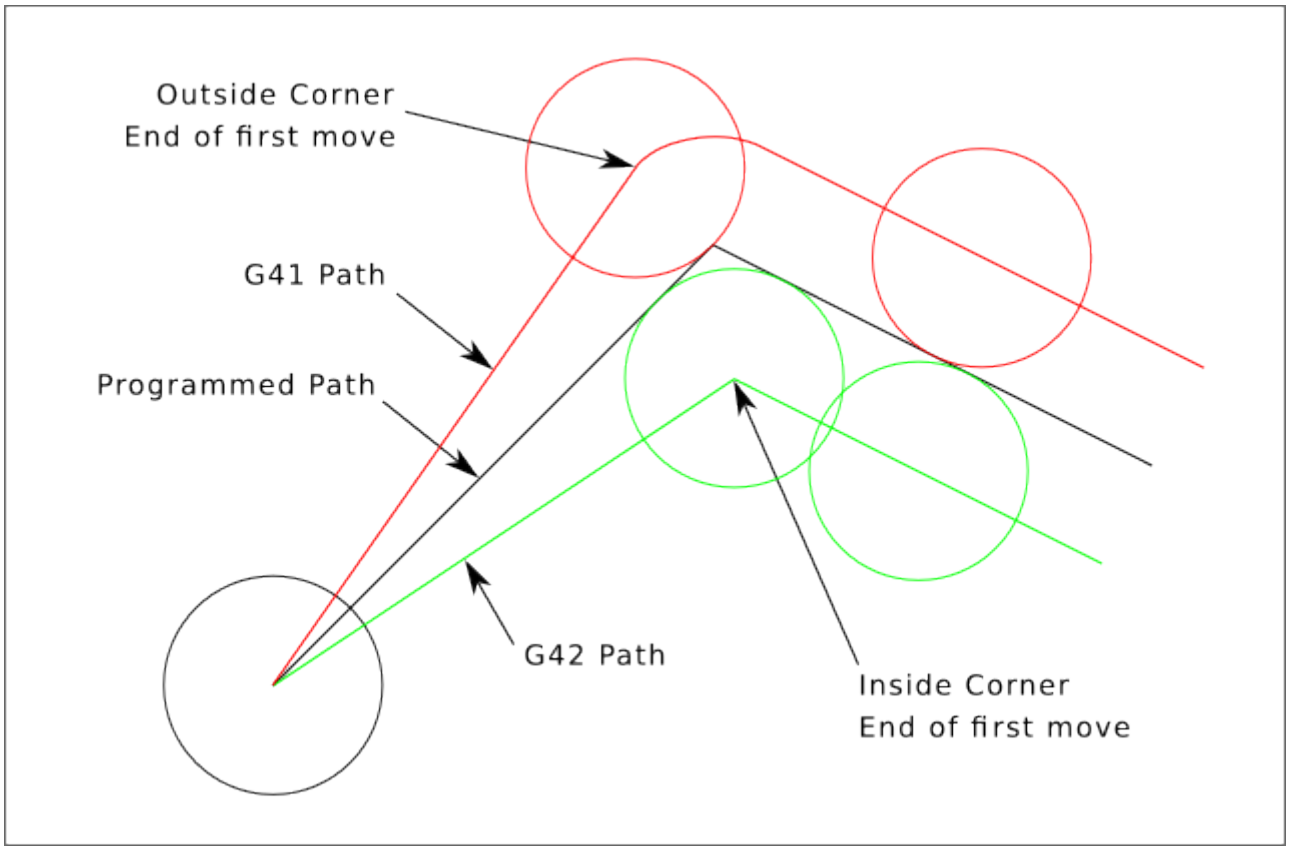


Figure 11.3: Кінцева точка компенсації

11.2.4.1 Огляд

Компенсація на різець використовує дані з таблиці інструментів для визначення необхідного зміщення. Дані можна встановити під час виконання за допомогою G10 L1.

Будь-який рух, який є достатньо довгим для виконання компенсації, буде працювати як вхідний рух. Мінімальна довжина дорівнює радіусу різака. Це може бути швидкий рух над заготовкою. Якщо після G41/42 видається кілька швидких рухів, тільки останній з них перемістить інструмент у компенсоване положення.

На наступному малюнку ви можете побачити, що рух входу компенсується праворуч від лінії. У цьому випадку центр інструменту знаходиться праворуч від X0. Якщо ви програмуєте профіль, а кінець знаходиться в точці X0, то в результаті профіль матиме нерівність через зміщення руху входу.

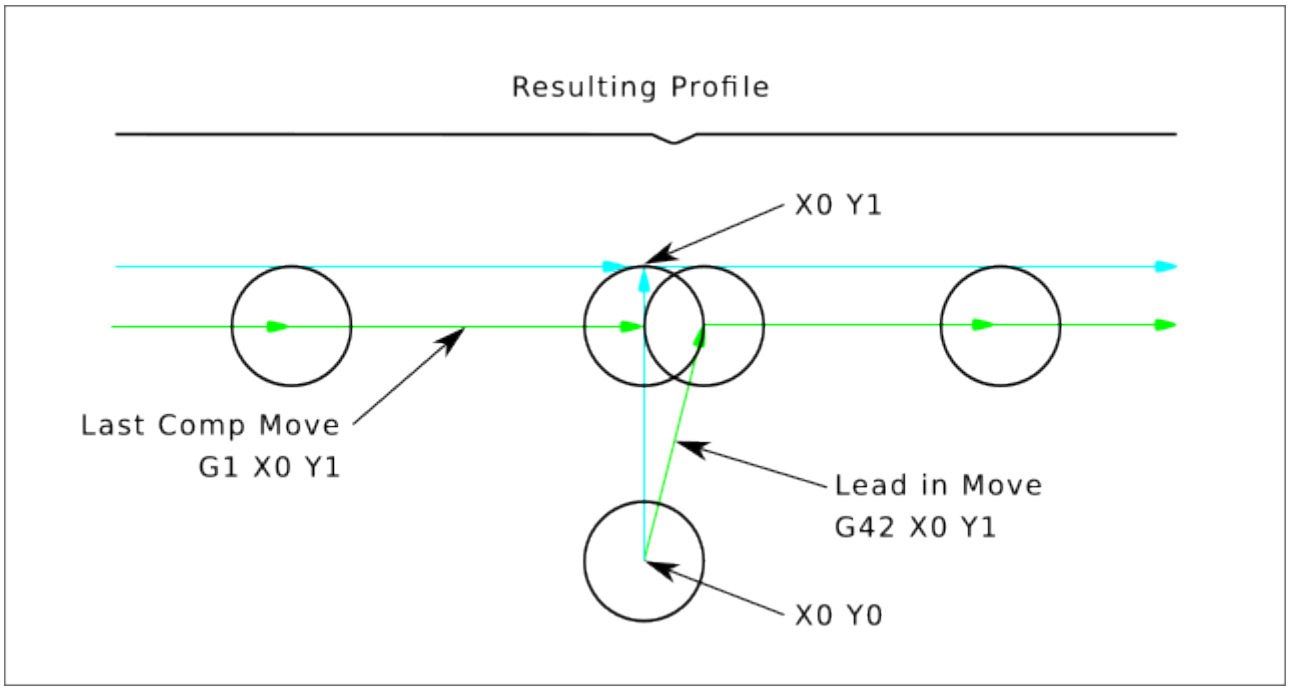


Figure 11.4: Вхідний хід

Рух по осі Z може відбуватися під час проходження контуру в площині XY. Частини контуру можуть бути пропущені шляхом відведення осі Z над деталлю та висунення осі Z у наступній початковій точці.

Швидкі рухи можна запрограмувати, навіть якщо компенсація ввімкнена.

Запустіть програму з G40, щоб переконатися, що компенсація вимкнена.

11.2.4.2 Приклади

```
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 )
```

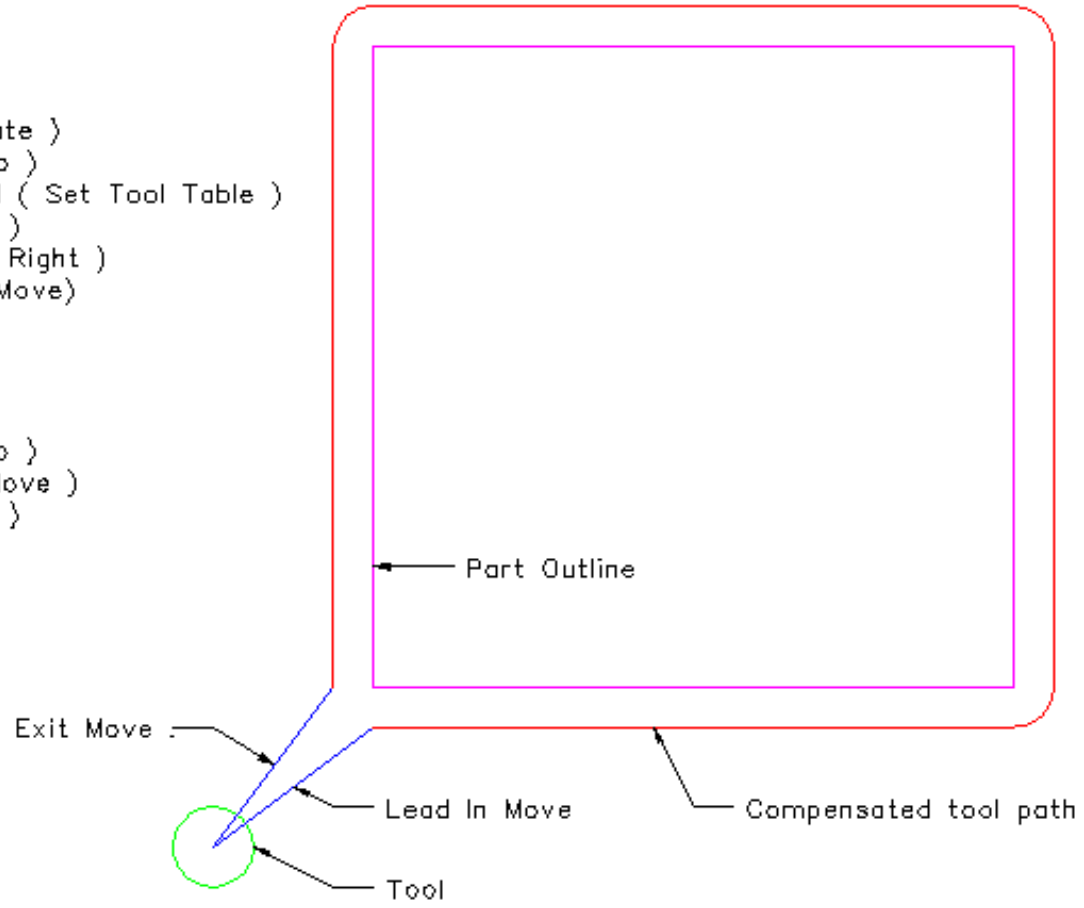


Figure 11.5: Зовнішній профіль

```
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 )
```

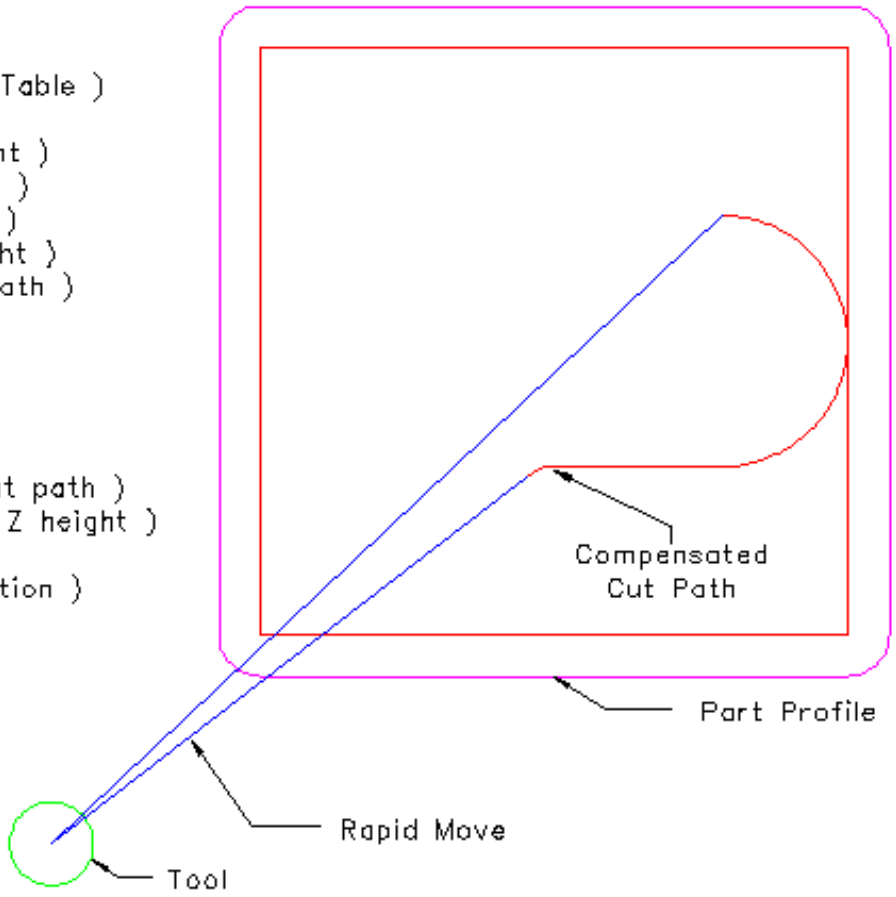


Figure 11.6: Внутрішній профіль

11.3 Графічний інтерфейс редагування інструментів

11.3.1 Огляд

Note
Елементи tooledit, описані тут, доступні, починаючи з версії 2.5.1 та пізніших. У версії 2.5.0 графічний інтерфейс не дозволяє виконувати ці налаштування.

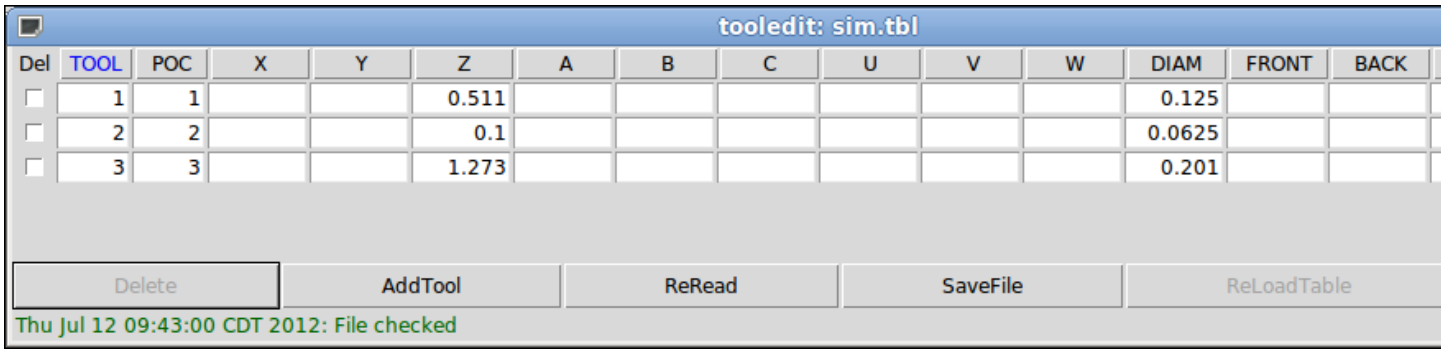


Figure 11.7: Графічний інтерфейс редагування інструментів – огляд

Програма «tooledit» може оновлювати файл таблиці інструментів з редагованими змінами за допомогою кнопки «SaveFile». Кнопка «SaveFile» оновлює системний файл, але для оновлення даних таблиці інструментів, що використовуються запущеним екземпляром LinuxCNC, потрібна окрема дія. За допомогою графічного інтерфейсу AXIS за допомогою кнопки ReloadTable можна оновити як файл, так і поточні дані таблиці інструментів, що використовуються LinuxCNC. Ця кнопка активна тільки тоді, коли верстат увімкнений і знаходиться в режимі очікування.

11.3.2 Сортування за стовпцями

Відображення таблиці інструментів можна сортувати за будь-яким стовпцем у порядку зростання, натиснувши на заголовок стовпця. Друге натискання сортує у порядку спадання. Для сортування стовпців необхідно, щоб машина була налаштована з версією Tcl за замовчуванням >= 8.5.

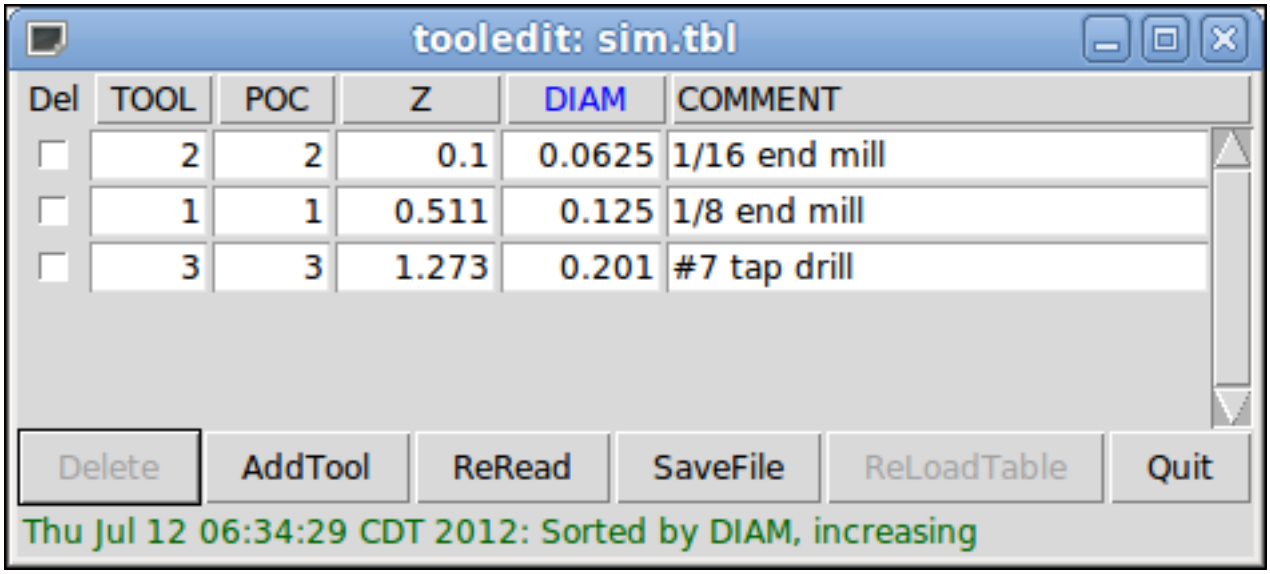


Figure 11.8: Графічний інтерфейс редагування інструментів – сортування стовпців

В Ubuntu Lucid 10.04 Tcl/Tk8.4 він встановлений за замовчуванням. Встановлення виконується наступним чином:

```
sudo apt-get install tcl8.5 tk8.5
```

Залежно від інших програм, встановлених у системі, може знадобитися ввімкнути Tcl/Tk8.5 за допомогою команд:

```
sudo update-alternatives --config tclsh  ;# b''вb''b''иb''b''бb''b''eb''b''pb''b''ib''b' ↩
    'тb''b''ьb'' b''ob''b''пb''b''цb''b''ib''b''юb'' b''дb''b''лb''b''яb'' tclsh8.5
sudo update-alternatives --config wish   ;# b''вb''b''иb''b''бb''b''eb''b''pb''b''ib''b' ↩
    'тb''b''ьb'' b''ob''b''пb''b''цb''b''ib''b''юb'' b''дb''b''лb''b''яb'' wish8.5
```

11.3.3 Вибір стовпців

За замовчуванням програма «tooledit» відображає всі можливі стовпці параметрів таблиці інструментів. Оскільки лише деякі верстати використовують усі параметри, відображення стовпців можна обмежити за допомогою наступного параметра файлу INI:

Синтаксис INI-файлу

```
[DISPLAY]
TOOL_EDITOR = tooledit column_name column_name ...
```

Приклад для стовпців Z та DIAM

```
[DISPLAY]
TOOL_EDITOR = tooledit Z DIAM
```

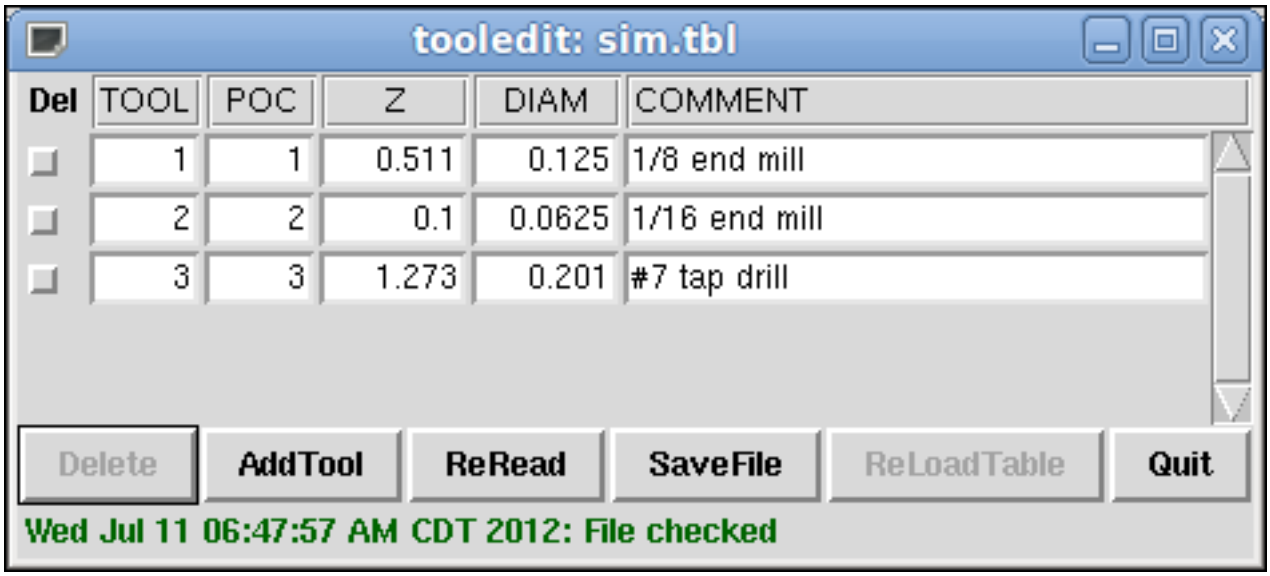


Figure 11.9: Графічний інтерфейс інструмента редагування – приклад вибору стовпців

11.3.4 Окреме використання

Програму «tooledit» також можна викликати як окрему програму. Наприклад, якщо програма знаходиться в користувацькому PATH, введення «tooledit» покаже синтаксис використання:

Окремо стояти

```
tooledit
b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':
       tooledit filename
       tooledit [column_1 ... column_n] filename
```

```
b''Дb''b''ib''b''йb''b''cb''b''нb''b''ib'' b''нb''b''ab''b''зb''b''вb''b''иb'' b''cb''b' ↩
    'тb''b''ob''b''вb''b''пb''b''цb''b''ib''b''вb'': x, y, z, a, b, c, u, v, w, diam, front, ↩
     back, orient.
```

Щоб синхронізувати окремий «tooledit» із запущеною програмою LinuxCNC, ім'я файлу має відповідати імені файлу [EMCIO]TOOL_TABLE, зазначеному у файлі INI LinuxCNC.

При використанні програми «tooledit» під час роботи LinuxCNC виконання команд G-коду або інших програм може змінити дані таблиці інструментів та файл таблиці інструментів. Зміни у файлі виявляються програмою «tooledit» і відображається повідомлення:

```
b''Пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'': b' ↩
    'Фb''b''ab''b''йb''b''лb'' b''зb''b''мb''b''ib''b''нb''b''eb''b''нb''b''ob'' b''ib''b' ↩
    'нb''b''шb''b''иb''b''мb'' b''пb''b''pb''b''ob''b''цb''b''eb''b''cb''b''ob''b''мb''
```

Таблицю інструментів «tooledit» можна оновити для зчитування зміненого файлу за допомогою кнопки «Повторно зчитати».

Таблиця інструментів вказується у файлі INI за допомогою запису:

```
[EMCIO]TOOL_TABLE = tool_table_filename
```

Файл таблиці інструментів можна редагувати за допомогою будь-якого простого текстового редактора (не текстового процесора).

Графічний інтерфейс AXIS може за потреби використовувати налаштування INI-файлу для визначення програми редактора інструментів:

```
[DISPLAY]TOOL_EDITOR = path_to_editor_program
```

За замовчуванням використовується програма під назвою «tooledit». Цей редактор підтримує всі параметри таблиці інструментів, дозволяє додавати та видаляти записи інструментів, а також виконує ряд перевірок дійсності значень параметрів.

11.4 Огляд програмування G-кодом

11.4.1 Огляд

Мова G-коду LinuxCNC базується на мові RS274/NGC. Мова G-коду базується на рядках коду. Кожен рядок (також званий «блоком») може містити команди для виконання декількох різних дій. Рядки коду можуть бути зібрані у файл для створення програми.

Типова рядок коду складається з необов'язкового номера рядка на початку, за яким слідує одне або кілька «слів». Слово складається з літери, за якою слідує число (або щось, що обчислюється як число). Слово може бути командою або аргументом команди. Наприклад, «G1 X3» — це дійсний рядок коду з двома словами. «G1» — це команда, що означає «рухатися по прямій лінії із запрограмованою швидкістю подачі до запрограмованої кінцевої точки», а «X3» надає значення аргументу (значення X повинно бути 3 в кінці руху). Більшість команд G-коду LinuxCNC починаються з G або M (для загальних і різних). Слова для цих команд називаються «G-кодами» і «M-кодами». Також поширеними є коди підпрограм, що починаються з «o-», які називаються «o-кодами».

Мова LinuxCNC не має індикатора початку програми. Однак інтерпретатор обробляє файли. Одна програма може міститися в одному файлі, або програма може бути розподілена між декількома файлами. Файл може бути розмежований відсотками наступним чином. Перший непустий рядок файлу може містити тільки знак відсотка «%», можливо оточений пробілами, а далі у файлі (зазвичай в кінці файлу) може бути подібний рядок. Розмежування файлу за допомогою відсотків

є необов’язковим, якщо файл містить «M2» або «M30», але є обов’язковим, якщо їх немає. Якщо файл має рядок з відсотками на початку, але не має його в кінці, буде видано повідомлення про помилку. Корисний вміст файлу, розмежованого за допомогою відсотків, закінчується після другого рядка з відсотками. Все, що йде після цього, ігнорується.

Мова G-коду LinuxCNC має дві команди («M2» або «M30»), кожна з яких завершує програму. Програма може закінчитися до кінця файлу. Рядки файлу, що знаходяться після кінця програми, не виконуються. Інтерпретатор навіть не читає їх.

11.4.2 Формат рядка

Допустимий рядок вхідного коду складається з наступного, у зазначеному порядку, з обмеженням максимальної (наразі 256) кількості символів, дозволених у рядку.

1. необов’язковий символ видалення блоку, яким є скісний рисочок /.
2. необов’язковий номер рядка.
3. Будь-яка кількість:
    1. слова,
    2. налаштування параметрів,
    3. коди підпрограм та
    4. коментарі.
4. маркер кінця рядка (повернення каретки або переведення рядка, або обидва).

Будь-який ввід, який явно не дозволено, є незаконним і призведе до того, що інтерпретатор сигналізує про помилку.

Пробіли та табуляції дозволені в будь-якому місці рядка коду і не змінюють значення рядка, за винятком коментарів. Це робить деякі дивно виглядаючі введення допустимими. Наприклад, рядок «G0X +0. 12 34Y 7» еквівалентний «G0 x+0.1234 Y7».

У вхідних даних дозволено використовувати порожні рядки. Їх слід ігнорувати.

Вхідні дані не враховують регістр, окрім коментарів, тобто будь-яка літера поза коментарем може бути у верхньому або нижньому регістрі, не змінюючи значення рядка.

11.4.2.1 /: Видалення блоку

Опціональний символ блокового видалення «/», розміщений на початку рядка, може використовуватис деякими користувальницькими інтерфейсами для пропускання рядків коду за потреби. У Axis комбінація клавіш Alt-m-/ вмикає та вимикає блокове видалення. Коли блокове видалення ввімкнено, всі рядки, що починаються зі знака «/», пропускаються.

В AXIS також можна ввімкнути видалення блоку за допомогою наступного значка:

Піктограма видалення блоку AXIS 

11.4.2.2 Додатковий номер рядка

Номер рядка — це літера N, за якою йде ціле число без знака, за яким може йти крапка та ще одне ціле число без знака. Наприклад, «N1234» та «N56.78» є дійсними номерами рядків. Вони можуть повторюватися або використовуватися в довільному порядку, хоча зазвичай такого використання слід уникати. Номери рядків також можуть пропускатися, і це є нормальною практикою. Номер рядка не обов'язково використовувати, але якщо він використовується, він повинен бути в правильному місці.

Note
Номери рядків не рекомендуються. Див. Найкращі практики.

11.4.2.3 Слова, параметри, підпрограми, коментарі

Слово — це літера, відмінна від N або O ("o"), за якою йде дійсне значення.

Слова можуть починатися з будь-якої літери, наведеної в таблиці нижче. Для повноти таблиця включає літери N і O, хоча, як визначено вище, номери рядків і параметри потоку програми не є словами. Деякі літери (I, J, K, L, P, R) можуть мати різне значення в різних контекстах. Літери, що позначають назви осей, не є дійсними на верстаті, який не має відповідної осі.

Table 11.3: Слова та їхні значення

Лист	Значення
A	Вісь машини
B	Вісь B машини
C	Вісь C машини
D	Номер компенсації радіуса інструменту
F	Швидкість подачі
G	Загальна функція (див. таблицю G-code Modal Groups)
H	Індекс зміщення довжини інструменту
I	Зсув X для дуг та стандартних циклів G87
J	Зміщення Y для дуг та стандартних циклів G87
K	Зміщення по осі Z для дуг та стандартних циклів G87.
	Співвідношення руху шпинделя для синхронізованих рухів G33.
L	загальне слово параметра для G10, M66 та інших
M	Різні функції (див. таблицю M-code Modal Groups)
N	Номер рядка (не рекомендується, див. Рекомендації)
O	o-коди для керування потоком програми (див. o-Codes)
P	Час витримки у стандартних циклах та з G4.
	Ключ, що використовується з G10.
Q	Збільшення подачі в стандартних циклах G73, G83
R	Радіус дуги або площина стандартного циклу
S	Швидкість шпинделя
T	Вибір інструменту
U	Вісь U машини
V	V-вісь машини
W	Вісь W машини
X	Вісь X машини
Y	Вісь Y машини
Z	Вісь Z машини

Параметри позначаються символом ”#” перед ними. Див. Parameters Section нижче.

Також називаються «о-кодами», вони забезпечують контроль потоку програми (наприклад, логіку if-else та викличні підпрограми) і повністю висвітлені на сторінці o-Codes, а також нижче в розділі Subroutine Codes and Parameters.

Note
О-коди іноді також називають О-словами.

Коментарі можна вставляти в рядок за допомогою дужок () або в кінці рядка за допомогою крапки з комою. Існують також «активні» коментарі, такі як MSG, DEBUG тощо. Див. розділ section on comments.

11.4.2.4 Маркер кінця лінії

Це будь-яка комбінація повернення каретки або переведення рядка.

11.4.3 Числа

Для (явних) чисел використовуються такі правила. У цих правилах цифра – це один символ від 0 до 9.

- Номер складається з:
  - необов’язковий знак плюс або мінус, а потім
  - від нуля до багатьох цифр, за якими, можливо, слідує
  - одна кома з комою в десятковій козі, а потім
  - від нуля до багатьох цифр – за умови, що десь у числі є хоча б одна цифра.
- Існує два види чисел:
  - Цілі числа, які не мають десяткової коми,
  - Десяткові дроби, які мають десяткову кому.
- Числа можуть мати будь-яку кількість цифр, з урахуванням обмеження довжини рядка. Однак буде збережено лише близько сімнадцяти значущих цифр (достатньо для всіх відомих застосувань).
- Ненульове число без знака, крім першого символу, вважається додатним.

Зверніть увагу, що початкові (перед десятковою крапкою і першою цифрою, відмінною від нуля) і кінцеві (після десяткової крапки і останньої цифри, відмінної від нуля) нулі допускаються, але не є обов’язковими. Число, записане з початковими або кінцевими нулями, буде мати те саме значення при читанні, якби додаткові нулі були відсутні.

Числа, що використовуються для конкретних цілей в RS274/NGC, часто обмежуються деяким скінченним набором значень або деяким діапазоном значень. У багатьох випадках десяткові числа повинні бути близькими до цілих чисел; це стосується значень індексів (наприклад, для параметрів і номерів слотів каруселі), M-кодів і G-кодів, помножених на десять. Десяткове число, яке призначене для представлення цілого числа, вважається достатньо близьким, якщо воно знаходиться в межах 0,0001 від цілого значення.

11.4.4 Параметри

Мова RS274/NGC підтримує «параметри» — те, що в інших мовах програмування називається «змінними». Існує кілька типів параметрів різного призначення та вигляду, кожен з яких описаний у наступних розділах. Єдиний тип значень, який підтримується параметрами, — це плаваюча точка; у G-коді немає типів рядків, логічних значень або цілих чисел, як в інших мовах програмування. Однак логічні вирази можна формулювати за допомогою булеві оператори («AND», «OR», «XOR» та оператори порівняння «EQ», «NE», «GT», «GE», «LT», «LE»), а також «MOD», «ROUND», *FUP* та *FIX* operators підтримують арифметику цілих чисел.

Параметри відрізняються синтаксисом, областю дії, поведінкою, коли ще не ініціалізовані, режимом, персистенцією та цільовим використанням.

Синтаксис
Існує три види синтаксичного вигляду:

- *numbered* - #4711
- *named local* - #<localvalue>
- *named global* - #<_globalvalue>

Сфера застосування
Область дії параметра може бути глобальною або локальною в межах підпрограми. Параметри підпрограми та локальні іменовані змінні мають локальну область дії. Глобальні іменовані параметри та пронумеровані параметри, починаючи з номера 31, мають глобальну область дії. RS274/NGC використовує «лексичну область дії» — у підпрограмі видимі лише локальні змінні, визначені в ній, та будь-які глобальні змінні. Локальні змінні процедури, що викликає, не видимі в процедурі, що викликається.

Поведінка неініціалізованих параметрів

- Неініціалізовані глобальні параметри та невикористані параметри підпрограми повертають значення нуль, якщо їх використовувати у виразі.
- Неініціалізовані іменовані параметри сигналізують про помилку, якщо їх використовувати у виразі.

Режим
Більшість параметрів є параметрами читання/запису і можуть бути призначені в операторі присвоювання. Однак для багатьох попередньо визначених параметрів це не має сенсу, тому вони є параметрами тільки для читання — вони можуть з’являтися у виразах, але не в лівій частині оператора присвоювання.

Наполегливість
Коли LinuxCNC вимикається, тимчасові параметри втрачають свої значення. Всі параметри, крім пронумерованих параметрів у поточному постійному діапазоні примітка:[persistent_range, Діапазон постійних параметрів може змінюватися в міру розвитку. Наразі цей діапазон становить 5161-5390. Він визначений у масиві «required_parameters» у файлі src/emc/rs274ngc/int .] є тимчасовими. Постійні параметри зберігаються у файлі .var і відновлюються до своїх попередніх значень при повторному запуску LinuxCNC. Нестабільні нумеровані параметри скидаються до нуля.

Цільове використання

- параметри користувача - пронумеровані параметри в діапазоні 31..5000, а також іменовані глобальні та локальні параметри, за винятком попередньо визначених параметрів. Вони доступні для загального зберігання значень з плаваючою комою, таких як проміжні результати, прапорці тощо, протягом усього виконання програми. Вони є читабельними/записувальними (можуть бути присвоєні значення).

- subroutine-parameters – ці параметри використовуються для зберігання фактичних параметрів, що передаються до підпрограми.
- numbered-parameters – більшість із них використовуються для доступу до зміщень систем координат.
- system-parameters – використовуються для визначення поточної запущеної версії. Вони доступні лише для читання.

11.4.4.1 Нумеровані параметри

Пронумерований параметр — це символ «#», за яким йде ціле число від 1 до (наразі) 5602. Примітка: [Інтерпретатор RS274/NGC підтримує масив пронумерованих параметрів. Його розмір визначається символом «RS274NGC_MAX_PARAMETERS» у файлі src/emc/rs274ngc/interp_internal.hh). Кількість числових параметрів також може збільшуватися в міру додавання підтримки нових параметрів у процесі розробки.]. Параметр позначається цим цілим числом, а його значенням є будь-яке число, яке зберігається в параметрі.

Значення зберігається в параметрі за допомогою оператора =; наприклад:

```
#3 = 15 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b' ↩
    'ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' 3 b''нb''b''ab'' 15)
```

Налаштування параметра набуває чинності лише після того, як будуть знайдені всі значення параметрів у тому самому рядку. Наприклад, якщо параметр 3 було попередньо встановлено на 15 і інтерпретовано рядок «#3=6 G1 X#3», відбудеться прямий рух до точки, де X дорівнює 15, а значення параметра 3 буде 6.

Символ # має пріоритет над іншими операціями, так що, наприклад, *#1+2* означає число, отримане додаванням 2 до значення параметра 1, а не значення, знайденого в параметрі 3. Звичайно, *#[1+2]* означає значення, знайдене в параметрі 3. Символ # може повторюватися; наприклад, *##2* означає значення параметра, індекс якого є (цілим) значенням параметра 2.

- *31-5000* – параметри користувача G-коду. Ці параметри є глобальними у файлі G-коду та доступні для загального використання. Нестабільні.
- *5061-5069* – Координати результату зонду G38 (X, Y, Z, A, B, C, U, V та W). Координати вказані в системі координат, в якій відбувся G38. Нестабільний.
- *5070* - G38 результат зонду: 1, якщо успішно, 0, якщо зонд не вдалося замкнути. Використовується з G38.3 та G38.5. Нестабільний.
- *5161-5169* - ”G28” Домашня сторінка для X, Y, Z, A, B, C, U, V та W. Постійне.
- *5181-5189* – ”G30” Домашня сторінка для X, Y, Z, A, B, C, U, V та W. Постійне.
- *5210* - 1, якщо наразі застосовується зміщення ”G52” або ”G92”, інакше 0. За замовчуванням постійне; нестабільне, якщо *DISABLE_G92_PERSISTENCE = 1* у розділі *[RS274NGC]* INI-файлу.
- *5211-5219* – Спільне зміщення ”G52” та ”G92” для X, Y, Z, A, B, C, U, V та W. За замовчуванням нестабільне; постійне, якщо *DISABLE_G92_PERSISTENCE = 1* у розділі *[RS274NGC]* INI-файлу.
- *5220* – Номер системи координат 1–9 для G54–G59.3. Постійно.
- *5221-5230* – Система координат 1, G54 для X, Y, Z, A, B, C, U, V, W та R. R позначає кут повороту XY навколо осі Z. Збережено.
- *5241-5250* – Система координат 2, G55 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5261-5270* – Система координат 3, G56 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5281-5290* – Система координат 4, G57 для X, Y, Z, A, B, C, U, V, W та R. Постійна.

- *5301-5310* - Система координат 5, G58 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5321-5330* – Система координат 6, G59 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5341-5350* - Система координат 7, G59.1 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5361-5370* - Система координат 8, G59.2 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5381-5390* - Система координат 9, G59.3 для X, Y, Z, A, B, C, U, V, W та R. Постійна.
- *5399* – Результат M66 – Перевірте або зачекайте на вхідні дані. Нестабільний.
- *5400* - Номер інструменту. Нестабільний.
- *5401-5409* – Зміщення інструменту для X, Y, Z, A, B, C, U, V та W. Встановлюється G43. Нестабільне.
- *5410* - Діаметр інструменту. Летючий.
- *5411* - Кут передньої частини інструменту. Летючий.
- *5412* - Кут нахилу спинки інструменту. Летючий.
- *5413* - Орієнтація інструменту. Нестабільний.
- *5420-5428* – Поточне відносне положення в активній системі координат, включаючи всі зміщення та в поточних одиницях програми для X, Y, Z, A, B, C, U, V та W, нестабільне.
- *5599* – Прапорець для керування виводом операторів (DEBUG). 1=вивід, 0=немає виводу; за замовчуванням=1. Нестабільний.

Збереження нумерованих параметрів Значення параметрів у постійному діапазоні зберігаються з часом, навіть якщо обробний центр вимкнений. LinuxCNC використовує файл параметрів для забезпечення постійності. Він управляється інтерпретатором. Інтерпретатор зчитує файл під час запуску та записує файл під час виходу.

Формат файлу параметрів показано в таблиці Parameter File Format.

Інтерпретатор очікує, що файл матиме два стовпці. Він пропускає всі рядки, які не містять рівно два числові значення. Перший стовпець повинен містити ціле число (номер параметра). Другий стовпець містить число з плаваючою комою (останнє значення цього параметра). Значення представлене в інтерпретаторі як число з плаваючою комою подвійної точності, але десяткова крапка в файлі не потрібна.

До цього файлу можна додавати параметри у визначеному користувачем діапазоні (31-5000). Такі параметри будуть зчитуватися інтерпретатором та записуватися у файл під час його завершення.

Відсутні параметри в постійному діапазоні будуть ініціалізовані нулем та записані з їхніми поточними значеннями під час наступної операції збереження.

Номери параметрів мають бути розташовані у порядку зростання. Якщо вони не у порядку зростання, буде сигналізовано помилку «Файл параметрів не в порядку зростання».

Оригінальний файл зберігається як резервна копія під час запису нового файлу.

Table 11.4: Формат файлу параметрів

Номер параметра	**Значення параметра**
5161	0.0
5162	0.0

11.4.4.2 Коди та параметри підпрограм

Коди підпрограм, або o-коди (іноді також називаються o-словами), забезпечують логіку та управління потоком у програмах NGC (як у логіці if-else). Вони називаються кодами підпрограм, оскільки вони також можуть утворювати підпрограми (як у sub-endsub).

Див. розділ про o-Codes.

Note
Якщо o-коди використовуються для формування підпрограм, то o-коди також можуть викликати ці підпрограми і передавати до 30 параметрів, які є локальними для підпрограми і мінливими. (Знову ж таки, див. o-Codes для більш повного викладу та прикладів.)

Note
Хоча дійсні як малі, так і великі літери «o-», найкращою практикою є використання малої літери «o-», оскільки вона усуває неоднозначність 0 (нуль) та O (велика o).

11.4.4.3 Іменовані параметри

Іменовані параметри працюють як пронумеровані параметри, але їх легше читати. Всі імена параметрів перетворюються в нижній регістр, а пробіли та табуляції видаляються, тому *<param>* і *<P a R am >* позначають один і той самий параметр. Іменовані параметри повинні бути укладені в знаки < >.

#<named parameter> є локальним іменованим параметром. За замовчуванням іменований параметр є локальним для області, в якій він призначений. Ви не можете отримати доступ до локального параметра поза його підпрограмою. Це означає, що дві підпрограми можуть використовувати однакові імена параметрів, не побоюючись, що одна підпрограма перезапише значення в іншій.

#<_global named parameter> є глобальним іменованим параметром. До них можна отримати доступ з викликуваних підпрограм і вони можуть встановлювати значення в підпрограмах, доступних для викликувача. Що стосується області дії, вони діють так само, як звичайні числові параметри. Вони не зберігаються у файлах.

Приклади:

Оголошення іменованої глобальної змінної

```
#<_endmill_dia> = 0.049
```

Посилання на раніше оголошену глобальну змінну

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

Змішані літерали та іменовані параметри

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]
```

Іменовані параметри з’являються, коли їм вперше присвоюється значення. Локальні іменовані параметри зникають, коли їх область дії закінчується: коли підпрограма повертається, всі її локальні параметри видаляються і більше не можуть бути використані.

Використання неіснуючого іменованого параметра у виразі або в правій частині присвоювання є помилкою. Виведення значення неіснуючого іменованого параметра за допомогою оператора DEBUG, наприклад *(DEBUG, <no_such_parameter>)*, призведе до відображення рядка #.

Глобальні параметри, а також локальні параметри, призначені на глобальному рівні, зберігають своє значення після призначення навіть після завершення програми та мають ці значення під час повторного запуску програми.

Функція EXISTS перевіряє, чи існує заданий іменований параметр.

11.4.4.4 Попередньо визначені іменовані параметри

Наступні глобальні параметри з іменами, доступні тільки для читання, використовуються для доступу до внутрішнього стану інтерпретатора та стану машини. Вони можуть використовуватися в довільних виразах, наприклад, для управління потоком програми за допомогою операторів if-then-else. Зверніть увагу, що нові predefined named parameters можна легко додати без змін у вихідному коді.

- *#<_vmajor>* – Основна версія пакета. Якщо поточна версія була 2.5.2, повернеться 2.5.
- *#<_vminor>* – Додаткова версія пакета. Якби поточна версія була 2.6.2, поверталося б 0.2.
- *#<_line>* - Порядковий номер. Якщо виконується файл G-коду, повертається номер поточного рядка.
- *#<_motion_mode>* - Повернути поточний режим руху інтерпретатора:

Режим руху	**повернене значення**
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>* - повертає значення, що позначає поточну площину:

Літак	**повернене значення**
G17	170
G18	180
G19	190
G17.1	171
G18.1	181
G19.1	191

- *#<_ccomp>* - Стан компенсації різця. Повернені значення:

Режим	повернене значення
G40	400
G41	410
G41.1	411
G41	410
G42	420
G42.1	421

- *#<_metric>* - Повертає 1, якщо G21 увімкнено, інакше 0.
- *#<_imperial>* - Повертає 1, якщо G20 увімкнено, інакше 0.
- *#<_absolute>* - Повертає 1, якщо G90 увімкнено, інакше 0.
- *#<_incremental>* - Повертає 1, якщо G91 увімкнено, інакше 0.
- *#<_inverse_time>* - Повертає 1, якщо увімкнено режим зворотної подачі (G93), інакше 0.
- *#<_units_per_minute>* - Повертає 1, якщо увімкнено режим подачі в одиницях/хвилину (G94), інакше 0.
- *#<_units_per_rev>* - Повертає 1, якщо увімкнено режим одиниць/обертання (G95), інакше 0.
- *#<_coord_system>* - Повертає число з плаваючою комою, що відповідає імені поточної системи координат (G54..G59.3). Наприклад, якщо ви використовуєте систему координат G55, то значення, що повертається, буде 550.000000, а якщо ви використовуєте G59.1, то значення, що повертається, буде 591.000000.

Режим	повернене значення
G54	540
G55	550
G56	560
G57	570
G58	580
G59	590
G59.1	591
G59.2	592
G59.3	593

- *#<_tool_offset>* - Повертає 1, якщо зміщення інструменту (G43) увімкнено, інакше 0.
- *#<_retract_r_plane>* - Повертає 1, якщо встановлено G98, інакше 0.
- *#<_retract_old_z>* - Повертає 1, якщо G99 увімкнено, інакше 0.

11.4.4.5 Системні параметри

* *<_spindle_rpm_mode>* - Повертає 1, якщо режим обертів шпинделя (G97) увімкнено, інакше 0. * *<_spindle_css_mode>* - Повертає 1, якщо режим постійної швидкості поверхні (G96) увімкнено, інакше 0. * *<_ijk_absolute_mode>* - Повертає 1, якщо режим абсолютної дугової відстані (G90.1) увімкнено, інакше 0. * *<_lathe_diameter_mode>* - Повертає 1, якщо це конфігурація токарного верстата та увімкнено режим діаметра (G7), інакше 0. * *<_lathe_radius_mode>* - Повертає 1, якщо це конфігурація токарного верстата та увімкнено режим радіуса (G8), інакше 0. * *<_spindle_on>* - Повертає 1, якщо шпиндель наразі працює (M3 або M4), інакше 0. * *<_spindle_cw>* - Повертає 1, якщо напрямок обертання шпинделя за годинниковою стрілкою (M3), інакше 0. * *<_mist>*

- Поверніть 1, якщо туман (M7) увімкнено. * <*_flood*> - Повертає 1, якщо увімкнено повінь (M8). * <*_speed_override*> - Повертає 1, якщо увімкнено корекцію подачі (M48 або M50 P1), інакше 0. * <*_feed_override*> - Повертає 1, якщо увімкнено корекцію подачі (M48 або M51 P1), інакше 0. * <*_adaptive_feed*> - Повертає 1, якщо адаптивна подача (M52 або M52 P1) увімкнена, інакше 0. * <*_feed_hold*> - Повертає 1, якщо перемикач блокування подачі увімкнено (M53 P1), інакше 0. * <*_feed*> - Повертає поточне значення F, а не фактичну швидкість подачі. * <*_rpm*> - Повертає поточне значення S, а не фактичну швидкість шпинделя. * <*_x*> - Повертає поточну відносну координату X, включаючи всі зміщення. Те саме, що й 5420. У конфігурації токарного верстата завжди повертає радіус. * <*_y*> - Повертає поточну відносну координату Y, включаючи всі зміщення. Те саме, що й 5421. * <*_z*> - Повертає поточну відносну координату Z, включаючи всі зміщення. Те саме, що й 5422. * <*_a*> - Повертає поточну відносну координату A, включаючи всі зміщення. Те саме, що й 5423. * <*_b*> - Повертає поточну відносну координату B, включаючи всі зміщення. Те саме, що й 5424. * <*_c*> - Повертає поточну відносну координату C, включаючи всі зміщення. Те саме, що й 5425. * <*_u*> - Повертає поточну відносну U-координату, включаючи всі зміщення. Те саме, що й 5426. * <*_v*> - Повертає поточну відносну V-координату, включаючи всі зміщення. Те саме, що й 5427. * <*_w*> - Повертає поточну відносну координату W, включаючи всі зміщення. Те саме, що й 5428. * <*_abs_x*> - Повертає поточну абсолютну координату X (G53) без урахування зміщень. * <*_abs_y*> - Повертає поточну абсолютну координату Y (G53) без урахування зміщень. * <*_abs_z*> - Повертає поточну абсолютну координату Z (G53) без урахування зміщень. * <*_abs_a*> - Повернути поточну абсолютну координату A (G53) без урахування зміщень. * <*_abs_b*> - Повертає поточну абсолютну координату B (G53) без урахування зміщень. * <*_abs_c*> - Повертає поточну абсолютну координату C (G53) без урахування зміщень. * <*_current_tool*> - Повернутий номер поточного інструменту в шпинделі. Те саме, що й 5400. * <*_current_pocket*> - Повертає індекс tooldata для поточного інструменту. * <*_selected_tool*> - Повернутий номер вибраного інструменту після T-коду. За замовчуванням -1. * <*_selected_pocket*> - Повертає індекс tooldata вибраної кишені після T-коду. За замовчуванням -1 (кишеня не вибрана). * <*_value*> - Повернене значення з останнього O-коду *return* або *endsub*. Значення за замовчуванням 0, якщо після *return* або *endsub* немає виразу. Ініціалізується значенням 0 під час запуску програми. * <*_value_returned*> - 1.0, якщо останній виклик O-коду *return* або *endsub* повернув значення, 0 в іншому випадку. Очищається наступним викликом O-коду. * <*_task*> - 1.0, якщо екземпляр інтерпретатора, що виконується, є частиною milltask, в іншому випадку - 0.0. Іноді необхідно по-особливому підходити до цього випадку, щоб зберегти правильний попередній перегляд, наприклад, під час перевірки успішності зонда (G38.n) шляхом перевірки #5070, що завжди призведе до помилки в інтерпретаторі попереднього перегляду (наприклад, Axis). * <*_call_level*> - Поточний рівень вкладеності процедур O-коду. Для налагодження. * #<*_remap_level*> - поточний рівень стеку перепризначення. Кожне перепризначення в блоці додає одиницю до рівня перепризначе Для налагодження.

11.4.5 Піни HAL та значення INI

Якщо ввімкнено у файлі INI, G-код має доступ до значень записів INI-файлу та виводів HAL.

- *#<_ini[section]name>* Повертає значення відповідного елемента у INI-файлі.

Наприклад, якщо INI-файл виглядає так:

```
[SETUP]
XPOS = 3.145
YPOS = 2.718
```

Ви можете звернутися до іменованих параметрів #<_ini[setup]xpos> та #<_ini[setup]ypos> у G-коді.

EXISTS можна використовувати для перевірки наявності заданої змінної INI-файлу:

```
o100 if [EXISTS[#<_ini[setup]xpos>]]
  (b''нb''b''ab''b''лb''b''ab''b''rb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', [ ↩
      setup]xpos b''ib''b''cb''b''нb''b''yb''b''єb'': #<_ini[setup]xpos>)
```

```
o100 else
  (b''нb''b''ab''b''лb''b''ab''b''гb''b''оb''b''дb''b''жb''b''еb''b''нb''b''нb''b''яb'', [ ↩
      setup]xpos b''нb''b''eb'' b''ib''b''cb''b''нb''b''yb''b''єb'')
o100 endif
```

Значення зчитується з файлу INI один раз і кешується в інтерпретаторі. Ці параметри доступні тільки для читання - присвоєння значення призведе до помилки виконання. У G-коді імена не чутливі до регістру - вони перетворюються у верхній регістр перед зверненням до файлу INI. Тому записи INI, що містять символи нижнього регістру, недоступні з G-коду.

- *#<_hal[HAL item]>* Дозволяє програмам G-коду зчитувати значення виводів HAL. Доступ до змінних є тільки для читання, єдиним способом *встановлення* виводів HAL з G-коду залишаються коди M62-M65, M67, M68 і користувацькі коди M100-M199. Зверніть увагу, що зчитане значення не оновлюється в режимі реального часу, зазвичай повертається значення, яке було на контакті під час запуску програми G-коду. Це можна обійти, примусово синхронізувавши стан. Один із способів зробити це — за допомогою фіктивної команди M66: M66E0L0

Приклад:

```
(debug, #<_hal[motion-controller.time]>)
```

Доступ до елементів HAL здійснюється лише для читання. Наразі таким чином можна отримати доступ лише до імен HAL, написаних виключно нижніми літерами.

EXISTS можна використовувати для перевірки наявності певного елемента HAL:

```
o100 if [EXISTS[#<_hal[motion-controller.time]>]]
  (b''нb''b''ab''b''лb''b''ab''b''гb''b''оb''b''дb''b''жb''b''еb''b''нb''b''нb''b''яb'', [ ↩
      motion-controller.time] b''ib''b''cb''b''нb''b''yb''b''єb'': #<_hal[motion-controller. ↩
      time]>)
o100 else
  (b''нb''b''ab''b''лb''b''ab''b''гb''b''оb''b''дb''b''жb''b''еb''b''нb''b''нb''b''яb'', [ ↩
      motion-controller.time] b''нb''b''eb'' b''ib''b''cb''b''нb''b''yb''b''єb'')
o100 endif
```

Ця функція була мотивована бажанням забезпечити більш тісний зв’язок між компонентами користувацького інтерфейсу, такими як GladeVCP і PyVCP, щоб вони могли слугувати джерелом параметрів для керування поведінкою файлів NGC. Альтернативний варіант — використання контактів M6x і їх підключення — має обмежений, не мнемонічний простір імен і є надмірно громіздким як механізм комунікації між користувацьким інтерфейсом та інтерпретатором.

11.4.6 Вирази

Вираз — це набір символів, що починається з лівої дужки «[» і закінчується відповідною правою дужкою «]». Між дужками знаходяться числа, значення параметрів, математичні операції та інші вирази. Вираз обчислюється для отримання числа. Вирази в рядку обчислюються під час читання рядка, перед виконанням будь-яких дій у рядку. Прикладом виразу є «[1 + acos[0] - [#3 ** [4.0/2]]]».

11.4.7 Бінарні оператори

Бінарні оператори з’являються тільки всередині виразів. Існує чотири основні математичні операції: додавання («+»), віднімання («-»), множення («**») і ділення («/»). Існує три логічні операції: не виключна або («OR»), виключна або («XOR») і логічна і («AND»). Восьма операція — операція модуля («MOD»). Дев’ята операція — це операція «потужність» («*»), що підносить число зліва від операції до потужності справа. Реляційні оператори — це

рівність («EQ»), нерівність («NE»), строго більше («GT»), більше або дорівнює («GE»), строго менше («LT») і менше або дорівнює («LE»).

Бінарні операції поділяються на кілька груп за їх пріоритетом. Якщо операції з різних груп пріоритету з'єднані між собою (наприклад, у виразі «[2.0 / 3 * 1.5 - 5.5 / 11.0]»), операції з вищої групи виконуються перед операціями з нижчої групи. Якщо вираз містить більше однієї операції з однієї групи (наприклад, перші «/» і «*» у прикладі), спочатку виконується операція зліва. Таким чином, приклад еквівалентний: «[[[2.0 / 3] * 1.5] - [5.5 / 11.0]]», що еквівалентно «[1.0 - 0.5]», тобто «0.5».

Логічні операції та модуль слід виконувати над будь-якими дійсними числами, а не лише над цілими. Число нуль еквівалентне логічній хибності, а будь-яке ненульове число еквівалентне логічній істині.

Table 11.9: Пріоритет операторів

Оператори	**Пріоритет**
**	*highest*
* / MOD	
+ -	
EQ NE GT GE LT LE	
І АБО Виключаюче АБО	*lowest*

11.4.8 Рівність та значення з плаваючою комою

Перевірка рівності або нерівності двох значень з подвійною точністю з плаваючою комою є за своєю суттю проблематичною. Інтерпретатор вирішує цю проблему, вважаючи значення рівними, якщо їх абсолютна різниця менше 1e-6 (це значення визначено як *TOLERANCE_EQUAL* у src/emc/rs274ngc/interp_internal.hh).

11.4.9 Функції

Доступні функції наведені в наступній таблиці. Аргументи для унарних операцій, які приймають величини кутів («COS», «SIN» і «TAN»), вимірюються в градусах. Значення, що повертаються унарними операціями, які повертають величини кутів («ACOS», «ASIN» і «ATAN»), також вимірюються в градусах.

Table 11.10: Функції G-коду

Назва функції	**Результат функції**
ATAN[arg]/[arg]	Арктангенс чотирьох квадрантів
ABS[arg]	Абсолютне значення
ACOS[arg]	Арккосинус
ASIN[arg]	Аркексинсус
COS[arg]	Косинус
EXP[arg]	e, зведене до заданого степеня
FIX[arg]	Округліть до меншого числа до цілого
FUP[arg]	Округліть до цілого числа
ROUND[arg]	Округліть до найближчого цілого числа
LN[arg]	Логарифм з основою e

Table 11.10: (continued)

Назва функції	Результат функції
SIN[arg]	Його/її
SQRT[arg]	Квадратний корінь
TAN[arg]	Тангенс
EXISTS[arg]	Перевірте іменований параметр

Функція «FIX» округляє числову пряму вліво (менш додатне або більш від'ємне), таким чином, «FIX[2.8] = 2» та «FIX[-2.8] = -3».

Операція «FUP» округляє число праворуч (більш додатне або менш від'ємне) на числовій прямій; «FUP[2.8] = 3» та «FUP[-2.8] = -2».

Функція «EXISTS» перевіряє наявність одного іменованого параметра. Вона приймає тільки один іменований параметр і повертає 1, якщо він існує, і 0, якщо він не існує. Використання пронумерованого параметра або виразу є помилкою. Ось приклад використання функції EXISTS:

```
o<test> sub
o10 b''яb''b''кb''b''щb''b''ob'' [EXISTS[#<_global>]]
    (debug, _global b''ib''b''cb''b''нb''b''yb''b''єb'' b''тb''b''ab'' b''мb''b''ab''b' ↩
        'єb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' #<_global>)
o10 b''ib''b''нb''b''ab''b''кb''b''шb''b''eb''
    (debug, _global b''нb''b''eb'' b''ib''b''cb''b''нb''b''yb''b''єb'')
o10 endif
o<test> endsub

o<test> call
#<_global> = 4711
o<test> call
m2
```

11.4.10 Повторювані елементи

Рядок може містити будь-яку кількість G-слів, але два G-слова з однієї модальної групи не можуть з'являтися в одному рядку. Див. розділ Modal Groups для отримання додаткової інформації.

Рядок може містити від нуля до чотирьох M-слів. Два M-слова з однієї модальної групи не можуть з'являтися в одному рядку.

Для всіх інших дозволених літер рядок може містити лише одне слово, що починається з цієї літери.

Якщо налаштування одного і того ж параметра повторюється в рядку, наприклад, «#3=15 #3=6», то чинним буде лише останнє налаштування. Налаштувати один і той же параметр двічі в одному рядку — це нерозумно, але не є порушенням правил.

Якщо в рядку з'являється більше одного коментаря, буде використано тільки останній; кожен з інших коментарів буде прочитано і перевірено його формат, але після цього він буде проігноровано. Очікується, що розміщення більше одного коментаря в рядку буде дуже рідкісним.

11.4.11 Порядок товарів

Три типи елементів, порядок яких може змінюватися в рядку (як зазначено на початку цього розділу), — це слово, налаштування параметра та коментар. Уявіть, що ці три типи елементів поділяються на три групи за типом.

Першу групу (слова) можна переставляти будь-яким чином, не змінюючи значення рядка.

Якщо друга група (налаштування параметрів) буде перепорядкована, значення рядка не зміниться, якщо тільки один і той самий параметр не буде встановлений більше ніж один раз. У цьому випадку буде діяти тільки останнє налаштування параметра. Наприклад, після інтерпретації рядка «#3=15 #3=6» значення параметра 3 буде 6. Якщо порядок буде змінено на «#3=6 #3=15» і рядок буде інтерпретовано, значення параметра 3 буде 15.

Якщо третя група (коментарі) містить більше одного коментаря та перевпорядковується, буде використано лише останній коментар.

Якщо кожна група зберігається в порядку або перепорядковується без зміни значення рядка, то три групи можуть бути переплетені будь-яким чином без зміни значення рядка. Наприклад, рядок «g40 g1 #3=15 (foo) #4=-7.0» має п’ять елементів і означає абсолютно те саме в будь-якому з 120 можливих порядків (наприклад, «#4=-7.0 g1 #3=15 g40 (foo)») для п’яти елементів.

11.4.12 Команди та режими роботи машини

Багато команд змушують контролер переходити з одного режиму в інший, і режим залишається активним, поки інша команда не змінить його явно або неявно. Такі команди називаються «модальними». Наприклад, якщо охолоджуюча рідина ввімкнена, вона залишається ввімкненою, поки її явно не вимкнуть. G-коди для руху також є модальними. Наприклад, якщо команда G1 (прямий рух) задана в одному рядку, вона буде виконана знову в наступному рядку, якщо в цьому рядку є одне або кілька слів осі, за винятком випадків, коли в наступному рядку задана явна команда з використанням слів осі або скасування руху.

«Немодальні» коди впливають лише на ті лінії, на яких вони зустрічаються. Наприклад, G4 (затримка) є немодальним.

11.4.13 Полярні координати

Полярні координати можна використовувати для визначення координат XY переміщення. @n — це відстань, а ^n — кут. Перевага цього полягає в тому, що для таких елементів, як кола отворів для болтів, можна дуже просто переміститися до точки в центрі кола, встановити зміщення, а потім переміститися до першого отвору і запустити цикл свердління. Полярні координати завжди відраховуються від поточної нульової позиції XY. Щоб змістити полярні координати від нуля верстата, використовуйте зміщення або виберіть систему координат.

В абсолютному режимі відстань і кут вимірюються від нульової позиції XY, а кут починається з 0 на позитивній осі X і збільшується в напрямку проти годинникової стрілки навколо осі Z. Код G1 @1^90 є таким самим, як G1 Y1.

У відносному режимі відстань і кут також враховуються від нульового положення XY, але вони є сукупними. Спочатку може бути заплутано, як це працює в інкрементальному режимі.

Наприклад, якщо у вас є наступна програма, ви можете очікувати, що вона буде квадратним візерунком:

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

З наступного рисунка видно, що результат не такий, як можна було б очікувати. Тому що ми додавали 0,5 до відстані щоразу, коли відстань від нульової точки XY збільшувалася з кожною лінією.

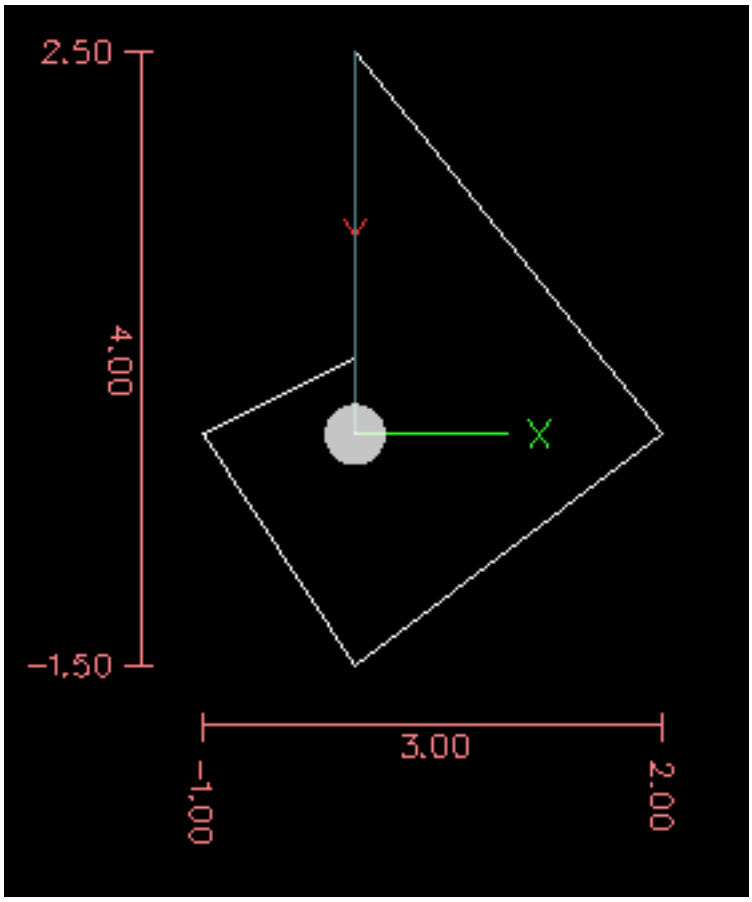


Figure 11.10: Полярна спіраль

Наступний код створить наш квадратний візерунок:

```
F100 G1 @.5 ^90
G91 ^90
^90
^90
^90
G90 G0 X0 Y0 M2
```

Як бачите, додаючи до кута лише 90 градусів щоразу, відстань до кінцевої точки однакова для кожної лінії.

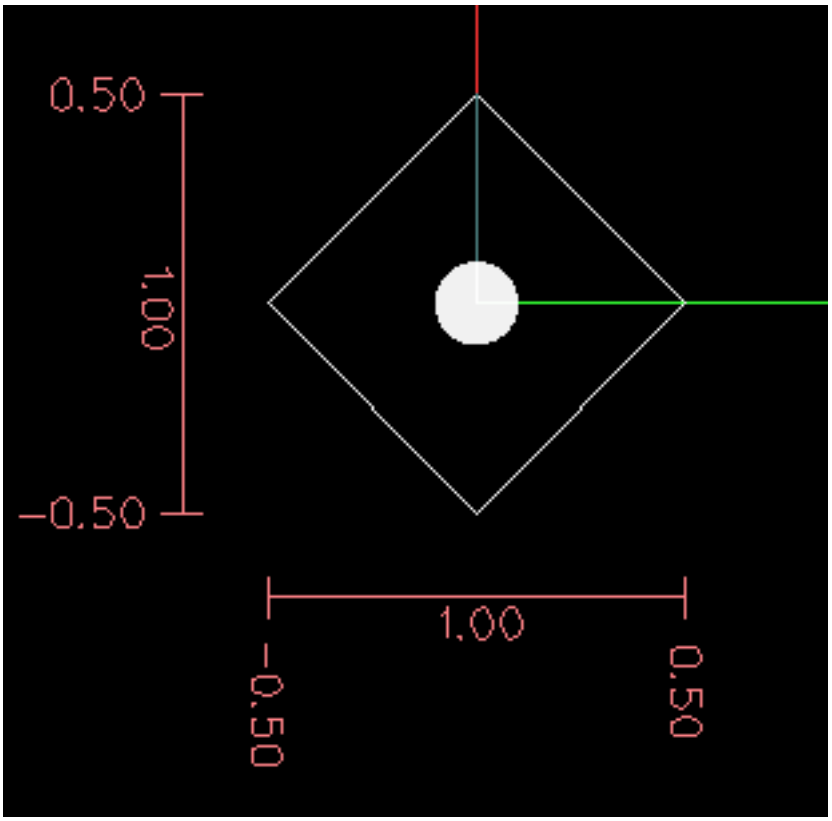


Figure 11.11: Полярна площа

Це помилка, якщо:

- Поступовий рух починається з початку координат
- Використовується поєднання полярних слів та слів X або Y

11.4.14 Модальні групи

Команди організовані в набори, які називаються «модальними групами», і лише один член модальної групи може бути чинним у будь-який момент часу. Загалом, модальна група містить команди, для яких логічно неможливо, щоб два члени були чинними одночасно – наприклад, вимірювання в дюймах проти вимірювання в міліметрах. Обробний центр може перебувати в багатьох режимах одночасно, при цьому активним є один режим з кожної модальної групи. Модальні групи показано в наступній таблиці.

Table 11.11: Модальні групи G-коду

Значення модальної групи	**Слова учасника**
Немодальні коди (Група 0)	G4, G10 G28, G30, G52, G53, G92, G92.1, G92.2, G92.3,
Рух (Група 1)	G0, G1, G2, G3, G33, G38.n, G73, G76, G80, G81 G82, G83, G84, G85, G86, G87, G88, G89
Вибір площини (Група 2)	G17, G18, G19, G17.1, G18.1, G19.1

Table 11.11: (continued)

Значення модальної групи	**Слова учасника**
Режим дистанції (Група 3)	G90, G91
Режим відстані дуги IJK (група 4)	G90.1, G91.1
Режим швидкості подачі (Група 5)	G93, G94, G95
Одиниці (Група 6)	G20, G21
Компенсація діаметра різця (Група 7)	G40, G41, G42, G41.1, G42.1
Зміщення довжини інструменту (Група 8)	G43, G43.1, G49
Режим повернення стандартних циклів (група 10)	G98, G99
Система координат (Група 12)	G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3
Режим керування (Група 13)	G61, G61.1, G64
Режим швидкості шпинделя (Група 14)	G96, G97
Режим діаметра токарного верстата (група 15)	G7, G8

Table 11.12: Модальні групи M-кодів

Значення модальної групи	**Слова учасника**
Зупинка (Група 4)	M0, M1, M2, M30, M60
Контакти вводу/виводу (Група 5)	(M62-M65 digital output), (M66 цифровий або аналоговий вхід), (M67, M68 аналоговий вихід)
Зміна інструменту (Група 6)	M6 T*n*
Шпиндель (Група 7)	M3, M4, M5
Охолоджувальна рідина (Група 8)	(M7 M8 можуть бути обидві), M9
Перемикачі керування (Група 9)	M48, M49
Визначено користувачем (Група 10)	M100-M199

Для декількох модальних груп, коли обробний центр готовий приймати команди, один член групи повинен бути активним. Для цих модальних груп існують налаштування за замовчуванням. Коли обробний центр вмикається або іншим чином переініціалізується, значення за замовчуванням автоматично вступають в силу.

Група 1, перша група в таблиці, – це група G-кодів для руху. Один з них завжди активний. Він називається поточним режимом руху.

Помилкою є розміщення G-коду з групи 1 і G-коду з групи 0 в одному рядку, якщо обидва вони використовують слова осі. Якщо G-код з групи 1, що використовує слова осі, неявним чином діє

в рядку (будучи активованим у попередньому рядку), а G-код з групи 0, що використовує слова осі, з'являється в рядку, дія G-коду з групи 1 для цього рядка призупиняється. G-коди з групи 0, що використовують слова осі, — це G10, G28, G30, G52 і G92.

Включати будь-які непов'язані слова в рядок з керуванням потоком *O*- є помилкою.

11.4.15 Коментарі

Коментарі носять суто інформативний характер і не впливають на роботу машини.

Коментарі можна додавати до рядків G-коду, щоб пояснити наміри програміста. Коментарі можна вставляти в рядок за допомогою дужок () або в кінці рядка за допомогою крапки з комою. Крапка з комою не розглядається як початок коментаря, якщо вона знаходиться в дужках.

Коментарі можуть з'являтися між словами, але не між словами та відповідним їм параметром. Отже, «S100(встановлена швидкість)F200(подача)» прийнятний, тоді як «S(швидкість)100F(подача)» – ні.

Ось приклад програми з коментарями:

```
G0 (b''Шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''пb''b''уb''b''сb''b''кb'') X1 Y1
G0 X1 Y1 (b''Шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''пb''b''уb''b''сb''b''кb''; b ↩
    ''аb''b''лb''b''еb'' b''нb''b''еb'' b''зb''b''аb''b''бb''b''уb''b''дb''b''ьb''b''тb''b' ↩
    'еb'' b''пb''b''рb''b''оb'' b''оb''b''хb''b''оb''b''лb''b''оb''b''дb''b''жb''b''уb''b' ↩
    'вb''b''аb''b''лb''b''ьb''b''нb''b''уb'' b''рb''b''іb''b''дb''b''иb''b''нb''b''уb'')
M2 ; b''Kb''b''іb''b''нb''b''еb''b''цb''b''ьb'' b''пb''b''рb''b''оb''b''гb''b''рb''b''аb''b ↩
    ''мb''b''иb''.
```

Існує кілька «активних» коментарів, які виглядають як коментарі, але викликають певні дії, наприклад «(debug,..)» або «(print,..)». Якщо в рядку є кілька коментарів, то за цими правилами інтерпретується тільки останній коментар. Отже, звичайний коментар, що йде за активним коментарем, фактично вимикає активний коментар. Наприклад, «(foo) (debug,#1)» надрукує значення параметра «#1», а «(debug,#1)(foo)» — ні.

Коментар, що починається з крапки з комою, за визначенням є останнім коментарем у цьому рядку та завжди інтерпретуватиметься відповідно до синтаксису активного коментаря.

Note
Вбудовані коментарі до O-кодів не слід використовувати, див. розділ comments для отримання додаткової інформації.

11.4.16 Повідомлення

* «(MSG,)» — відображає повідомлення, якщо «MSG» з'являється після лівої дужки і перед будь-якими іншими друкованими символами. Допускаються варіанти «MSG», що містять пробіли та символи нижнього регістру. Решта символів перед правою дужкою вважаються повідомленням. Повідомлення повинні відображатися на пристрої відображення повідомлень інтерфейсу користувача, якщо такий передбачений.

Приклад повідомлення

```
(MSG, b''Цb''b''eb'' b''пb''b''оb''b''вb''b''іb''b''дb''b''оb''b''мb''b''лb''b''eb''b''нb'' ↩
   b''нb''b''яb'')
```

11.4.17 Реєстрація зонда

* *(PROBEOPEN filename.txt)* - відкриє filename.txt та збереже в ньому 9-значну координату, що складається з XYZABCUVW кожного успішного прямого зондування. * *(PROBECLOSE)* - закриє відкритий файл журналу проб.

Для отримання додаткової інформації про зондування див. розділ G38.

11.4.18 Лісозаготівля

* *(LOGOPEN,filename.txt)* - відкриває іменований файл журналу. Якщо файл вже існує, він скорочується
* *(LOGAPPEND,filename)* - відкриває іменований файл журналу. Якщо файл вже існує, дані додаються.
* *(LOGCLOSE)* – закриває відкритий файл журналу. * *(LOG,)* – все, що знаходиться після ,, записується до файлу журналу, якщо він відкритий. Підтримує розширення параметрів, як описано нижче.

Приклади логування знаходяться у файлах G-коду *nc_files/examples/smartprobe.ngc* та *nc_files/ngcgui_li*

11.4.19 Повідомлення про скасування

* *(ABORT,)* – відображає повідомлення типу *(MSG,)* з додаванням спеціальної обробки параметрів коментарів, як описано нижче, та переривання поточного запущеного процесу.

11.4.20 Повідомлення про налагодження

* *(DEBUG,)* – відображає повідомлення типу *(MSG,)* з додаванням спеціальної обробки параметрів коментарів, як описано нижче.

11.4.21 Друк повідомлень

* *(PRINT,)* - Повідомлення виводяться на *stderr* зі спеціальною обробкою параметрів коментарів, як описано нижче.

11.4.22 Параметри коментарів

У коментарях DEBUG, PRINT та LOG значення параметрів у повідомленні розгортаються.

Наприклад: вивести іменовану глобальну змінну до stderr (вікно консолі за замовчуванням).

Приклад параметрів

```
(print,endmill dia = #<_endmill_dia>)
(print,value of variable 123 is: #123)
```

Усередині вищезазначених типів коментарів послідовності типу «#123» замінюються значенням параметра 123. Послідовності типу «#<іменований параметр>» замінюються значенням іменованого параметра. З іменованих параметрів видаляються пробіли. Отже, «#<іменований параметр>» буде перетворено на «#<іменованийпараметр>».

Номери параметрів можна форматувати, наприклад:

```
(DEBUG, value = %d#<some_value>)
```

виведе значення, округлене до цілого числа.

- %lf використовується за замовчуванням, якщо рядок форматування відсутній.
- %d = 0 десяткові дроби
- %f = чотири знаки під комою
- %.xf = x (0-9) кількість знаків під комою

Форматування буде виконано для всіх параметрів в одному рядку, якщо не буде змінено, тобто в одному рядку дозволено кілька форматів.

Рядок форматування не обов'язково має бути одразу поруч із параметром.

Якщо рядок форматування створено з неправильним шаблоном, він буде надрукований як символи.

11.4.23 Вимоги до файлу

Файл G-коду повинен містити один або декілька рядків G-коду та завершуватися символом Program End. Будь-який G-код після кінця програми не обчислюється.

Якщо код закінчення програми не використовується, використовується пара знаків відсотка «%», де перший знак відсотка знаходиться в першому рядку файлу, за ним йде один або кілька рядків G-коду, а потім другий знак відсотка. Будь-який код після другого знака відсотка не обробляється.



Warning
Використання % для обгортання файлу G-коду не дасть того ж результату, що і використання кінця програми. Машина буде перебувати в тому стані, в якому її залишила програма, що використовує %, шпиндель і охолоджуюча рідина можуть залишатися увімкненими, а такі параметри, як G90/91, залишаться такими, як їх встановила остання програма. Якщо ви не використовуєте належний преамбулу, наступна програма може запуститися в небезпечному стані.

Note
Файл має бути створений за допомогою текстового редактора, такого як Gedit, а не текстового процесора, такого як Open Office Word Processor.

11.4.24 Розмір файлу

Інтерпретатор і завдання ретельно написані таким чином, що єдиним обмеженням розміру програми обробки деталей є ємність диска. Однак інтерфейси TkLinuxCNC і Axis завантажують текст програми для відображення його користувачеві, тому обмежуючим фактором стає оперативна пам'ять. В Axis, оскільки попередній перегляд малюється за замовчуванням, час перемальовування також стає практичним обмеженням розміру програми. Попередній перегляд можна вимкнути в Axis, щоб пришвидшити завантаження великих програм обробки деталей. В Axis розділи попереднього перегляду можна вимкнути за допомогою коментарів preview control.

11.4.25 Порядок виконання G-коду

Порядок виконання елементів у рядку визначається не позицією кожного елемента в рядку, а наступним списком:

- Команди O-коду (необов'язково з коментарем, але інші слова в тому ж рядку не допускаються)

- Коментар (включно з повідомленням)
- Встановити режим швидкості подачі (G93, G94).
- Встановіть швидкість подачі (F).
- Встановіть швидкість шпинделя (S).
- Виберіть інструмент (T).
- Вхід/вихід HAL (M62-M68).
- Змінити інструмент (M6) та встановити номер інструменту (M61).
- Увімкнення або вимкнення шпинделя (M3, M4, M5).
- Зберегти стан (M70, M73), Відновити стан (M72), Визнати стан недійсним (M71).
- Увімкнення або вимкнення охолоджувальної рідини (M7, M8, M9).
- Увімкнути або вимкнути заміни (M48, M49,M50,M51,M52,M53).
- Команди, визначені користувачем (M100-M199).
- Затримка (G4).
- Встановити активну площину (G17, G18, G19).
- Встановити одиниці вимірювання довжини (G20, G21).
- Компенсація радіуса різця ввімкнена або вимкнена (G40, G41, G42)
- Компенсація довжини різця ввімкнена або вимкнена (G43, G49)
- Вибір системи координат (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
- Встановлення режиму керування траєкторією (G61, G61.1, G64)
- Встановити режим відстані (G90, G91).
- Встановити режим відведення (G98, G99).
- Перейдіть до опорної точки (G28, G30) або змініть дані системи координат (G10) або встановіть зміщення осей (G52, G92, G92.1, G92.2, G92.3).
- Виконайте рух (G0–G3, G33, G38.n, G73, G76, G80–G89), змінений (можливо) за допомогою G53.
- Стій (M0, M1, M2, M30, M60).

11.4.26 Найкращі практики G-коду

Використовуйте відповідну десяткову точність Використовуйте щонайменше 3 цифри після коми під час фрезерування в міліметрах і щонайменше 4 цифри після коми під час фрезерування в дюймах.

Зокрема, перевірки допусків дуг виконуються для 0,001 та 0,0001 відповідно до активних одиниць вимірювання.

Використовуйте послідовний білий простір G-код найрозбірливіший, коли перед словами стоїть хоча б один пробіл. Хоча дозволено вставляти пробіл посередині чисел, це не дозволено.

Використовуйте дуги з центральним форматом Дуги центрального формату (які використовують *I- J- K-* замість *R-*) поводяться більш стабільно, ніж дуги R-формату, особливо для кутів, що входять до складу, поблизу 180 або 360 градусів.

Використовуйте модальні групи набору преамбул Якщо правильне виконання вашої програми залежить від модальних налаштувань, обов'язково встановіть їх на початку програми обробки деталей. Режими можуть бути перенесені з попередніх програм та з команд MDI.

Приклад преамбули для млина

```
G17 G20 G40 G49 G54 G80 G90 G94
```

G17 використовувати площину XY, G20 дюймовий режим, G40 скасувати компенсацію діаметра, G49 скасувати зміщення довжини, G54 використовувати систему координат 1, G80 скасувати стандартні цикли, G90 режим абсолютної відстані, G94 режим подачі/хвилини.

Мабуть, найважливішим параметром модального режиму є одиниці вимірювання відстані. Якщо ви не включите G20 або G21, різні верстати будуть фрезерувати програму в різних масштабах. Інші параметри, такі як режим повернення в фіксованих циклах, також можуть бути важливими.

Не пишіть забагато речей в один рядок Ігноруйте все в розділі Order of Execution, і натомість не пишіть жодного рядка коду, який є хоч трохи неоднозначним.

Не встановлюйте та не використовуйте параметр в одному рядку Не використовуйте та не встановлюйте параметр в одному рядку, навіть якщо семантика добре визначена. Оновлення змінної до нового значення, наприклад, *#1=[#1+#2]*, є прийнятним.

Не використовуйте номери рядків Номери рядків не мають жодних переваг. Коли номери рядків повідомляються в повідомленнях про помилки, числа відносяться до номера рядка у файлі, а не до значення N-слів.

Коли переміщуються кілька систем координат Розгляньте можливість використання режиму зворотної швидкості часу.

Оскільки значення слова «F» в метрах за хвилину варіюється залежно від типу осі, що переміщується, а кількість видаленого матеріалу залежить не тільки від швидкості подачі, простіше використовувати G93, обернену швидкість часу, для видалення необхідної кількості матеріалу.

11.4.27 Лінійна та поворотна вісь

Оскільки значення F-слова в режимі подачі за хвилину залежить від того, яким осям подається команда на рух, а також оскільки кількість видаленого матеріалу залежить не тільки від швидкості подачі, для досягнення бажаної швидкості вилучення матеріалу може бути простіше використовувати режим зворотної подачі за часом G93.

11.4.28 Поширені повідомлення про помилки

- *G-code out of range* - Було використано G-код, більший за G99, діапазон G-кодів у LinuxCNC становить від 0 до 99. Не кожне число від 0 до 99 є дійсним G-кодом.
- *Unknown G-code used* - Було використано G-код, який не є частиною мови G-коду LinuxCNC.
- «i,j,k слово без Gx для його використання» – i, j та k слів мають використовуватися в тому ж рядку, що й G-код.
- «Неможливо використовувати значення осей без G-коду, який їх використовує» – значення осей не можна використовувати на рядку без активного модального G-коду або G-коду на тому ж рядку.
- «Файл закінчився без знака відсотка або кінця програми» – кожен файл G-коду повинен закінчуватис на M2 або M30 або бути перенесений на знак відсотка %.

11.5 G-Коди

11.5.1 Конвенції

Умовні позначення, що використовуються в цьому розділі

У прототипах G-коду дефіс (-) означає дійсне значення, а (<>) позначає необов’язковий елемент.

Якщо в прототипі написано «L-», то «-» часто називатиметься «числом L» і так далі для будь-якої іншої літери.

У прототипах G-коду слово «осі» означає будь-яку вісь, визначену у вашій конфігурації.

Необов’язкове значення буде записано так: <*L-* >.

Реальна вартість може бути:

- Явне число, «4»
- Вираз «[2+2]»
- Значення параметра «#88»
- Значення унарної функції, *acos[0]*

У більшості випадків, якщо вказано слова «вісь» (будь-яке або всі з «X Y Z A B C U V W»), вони вказують точку призначення.

Номери осей вказані в поточній активній системі координат, якщо явно не зазначено, що вони знаходяться в абсолютній системі координат.

Якщо слова осей необов’язкові, будь-які пропущені осі збережуть своє початкове значення.

Будь-які елементи в прототипах G-коду, які явно не описані як додаткові, є обов’язковими.

Значення, що йдуть за літерами, часто подаються у вигляді явних чисел. Якщо не вказано інше, явні числа можуть бути дійсними значеннями. Наприклад, «G10 L2» можна також записати як «G[2*5] L[1+1]». Якщо значення параметра 100 дорівнює 2, «G10 L#100» також матиме те саме значення.

Якщо в прототипі написано «L-», то «-» часто називатиметься «числом L» і так далі для будь-якої іншої літери.

11.5.2 Таблиця швидкого довідника G-коду

Код	Опис
G0	Скоординований рух з високою швидкістю
G1	Скоординований рух зі швидкістю подачі
G2 G3	Скоординований гвинтовий рух при швидкості подачі
G4	Залишатися
G5	Кубічний сплайн
G5.1	Квадратний B-сплайн
G5.2	NURBS, додати контрольну точку
G7	Режим діаметра (токарний верстат)
G8	Радіусний режим (токарний верстат)
G10 L0	Перезавантажити дані таблиці інструментів
G10 L1	Встановлення запису в таблиці інструментів

Код	Опис
G10 L10	Встановити таблицю інструментів, розраховано, заготовка
G10 L11	Встановити таблицю інструментів, розраховано, пристосування
G10 L2	Налаштування початку системи координат
G10 L20	Розраховано налаштування початку системи координат
G17 - G19.1	Вибір площини
G20 G21	Встановити одиниці вимірювання
G28 - G28.1	Перейти до попередньо визначеної позиції
G30 - G30.1	Перейти до попередньо визначеної позиції
G33	Синхронізований рух шпинделя
G33.1	Жорстке нарізання різьби
G38.2 - G38.5	Зондування
G40	Скасувати компенсацію різця
G41 G42	Компенсація різця
G41.1 G42.1	Динамічна компенсація різця
G43	Використовувати зміщення довжини інструменту з таблиці інструментів
G43.1	Динамічне зміщення довжини інструменту
G43.2	Застосувати додаткове зміщення довжини інструмента
G49	Скасувати зміщення довжини інструмента
G52	Зміщення локальної системи координат
G53	Перемістити машинні координати
G54-G59.3	Виберіть систему координат (1 - 9)
G61	Режим точного шляху
G61.1	Режим точної зупинки
G64	Режим керування траєкторією з додатковим допуском
G70	Цикл фінішної обробки токарного верстата
G71-G72	Цикл чорнової обробки токарного верстата
G73	Цикл свердління зі стружколомленням
G74	Цикл нарізання різьби лівою рукою з затримкою
G76	Багатопрохідний цикл нарізання різьби (токарний верстат)
G80	Скасувати режими руху
G81	Цикл буріння
G82	Цикл свердління з витримкою
G83	Цикл свердління з відсіканням Peck
G84	Цикл нарізання різьби правою рукою з затримкою
G85	Цикл розточування, без витримки, вихідна подача
G86	Цикл нудного свердління, зупинка, швидкий вихід
G87	Цикл зворотного розточування (ще не реалізовано)
G88	Цикл розточування, зупинка, ручний вихід (ще не реалізовано)
G89	Цикл розточування, витримка, вихідна подача
G90 G91	Режим дистанції
G90.1 G91.1	Режим дугової відстані
G92	Зміщення системи координат
G92.1 G92.2	Скасувати зміщення G92

Код	Опис
G92.3	Відновлення зміщень G92
G93 G94 G95	Режими подачі
G96 G97	Режим керування шпинделем, постійна поверхня в залежності від швидкості обертання (IPM або м/хв в залежності від об/хв)
G98 G99	Режим відведення по Z стандартному циклу

11.5.3 G0 Швидкий рух

```
G0 <axes>
```

Для швидкого руху запрограмуйте «осі G0», де всі слова осі є необов’язковими. «G0» є необов’язковим, якщо поточний режим руху є «G0». Це забезпечить скоординований рух до пункту призначення з максимальною швидкістю (або повільніше). «G0» зазвичай використовується як рух позиціонування.

11.5.3.1 Швидка швидкість

Параметр MAX_VELOCITY у розділі [TRAJ] файлу INI визначає максимальну швидкість швидкого переміщення. Максимальна швидкість швидкого переміщення може бути вищою за параметр MAX_VELOCITY окремих осей під час скоординованого переміщення. Максимальна швидкість швидкого переміщення може бути нижчою за параметр MAX_VELOCITY у розділі [TRAJ], якщо її обмежують параметр MAX_VELOCITY осі або обмеження траєкторії.

Приклад G0

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b ↩
    ''pb''b''eb''b''жb''b''иb''b''мb''b''yb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b' ↩
    'тb''b''нb''b''ob''b''їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G0 X1 Y-2.3 (b''Шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''лb''b''ib''b''нb''b''ib''b''йb'' ↩
    b''нb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b' ↩
    'яb'' b''вb''b''ib''b''дb'' b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''ob''b''гb''b' ↩
    'ob'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' X1  ↩
    Y-2.3)
M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''иb'')
```

- Див. розділи G90 та M2 для отримання додаткової інформації.

Якщо корекція різця активна, рух відрізнятиметься від вищезазначеного; див. розділ Cutter Compensation.

Якщо *G53* запрограмовано на тому ж рядку, рух також буде відрізнятися; див. розділ G53 для отримання додаткової інформації.

Шлях швидкого руху G0 може бути заокруглений при зміні напрямку та залежить від налаштувань trajectory control та максимального прискорення осей.

Це помилка, якщо:

- Літера осі не має реального значення.
- Використовується літера осі, яка не налаштована.

11.5.4 Лінійний рух G1

```
G1 axes
```

Для лінійного (прямолінійного) руху з запрограмованою feed rate (для різання або без нього) запрограмуйте «G1 *осі*», де всі слова осі є необов'язковими. «G1» є необов'язковим, якщо поточний режим руху є «G1». Це забезпечить скоординований рух до точки призначення з поточною швидкістю подачі (або повільніше).

Приклад G1

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''pb''b''eb''b ↩
    ''жb''b''иb''b''мb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ob''b' ↩
    'їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G1 X1.2 Y-3 F10 (b''лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''иb''b''йb'' b''pb''b''yb''b' ↩
    'хb'' b''зb''b''ib'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''юb'' b' ↩
    'пb''b''ob''b''дb''b''ab''b''чb''b''ib'' 10 b''вb''b''ib''b''дb'' b''пb''b''ob''b''тb''b ↩
    ''ob''b''чb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b' ↩
    'нb''b''нb''b''яb'' b''дb''b''ob'' X1.2 Y-3)
Z-2.3 (b''лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''иb''b''йb'' b''pb''b''yb''b''хb'' b' ↩
    'зb'' b''тb''b''ib''b''єb''b''юb'' b''жb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b' ↩
    'cb''b''тb''b''юb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b''вb''b''ib''b''дb'' b' ↩
    'пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b''лb''b' ↩
    'ob''b''жb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' Z-2.3)
Z1 F25 (b''лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''иb''b''йb'' b''pb''b''yb''b''хb'' b' ↩
    'зb''b''ib'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''юb'' b''пb''b' ↩
    'ob''b''дb''b''ab''b''чb''b''ib'' 25 b''вb''b''ib''b''дb'' b''пb''b''ob''b''тb''b''ob''b ↩
    ''чb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''дb''b''ob'' Z1)
M2 (b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''pb''b' ↩
    'ob''b''гb''b''pb''b''ab''b''мb''b''иb'')
```

- Див. розділи G90 та F та M2 для отримання додаткової інформації.

Якщо корекція різця активна, рух відрізнятиметься від вищезазначеного; див. розділ Cutter Compensation.

Якщо *G53* запрограмовано на тому ж рядку, рух також буде відрізнятися; див. розділ G53 для отримання додаткової інформації.

Це помилка, якщо:

- Швидкість подачі не встановлена.
- Літера осі не має реального значення.
- Використана літера осі, яка не налаштована

11.5.5 G2, G3 Дуговий рух

```
b''3b''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''ob''b''cb''b''eb''b''йb'' G2 b' ↩
    'ab''b''бb''b''ob'' G3 (b''цb''b''eb''b''нb''b''тb''b''pb''b''ab''b''лb''b''ьb''b''нb''b ↩
    ''иb''b''йb'' b''фb''b''ob''b''pb''b''мb''b''ab''b''тb'')
b''Ob''b''cb''b''ib'' G2 b''ab''b''бb''b''ob'' G3 R- (b''pb''b''ab''b''дb''b''ib''b''yb''b' ↩
    'cb''b''нb''b''иb''b''йb'' b''фb''b''ob''b''pb''b''мb''b''ab''b''тb'')
b''3b''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' G2 b''ab''b''бb''b''ob'' G3|R- <P ↩
    -> (b''пb''b''ob''b''вb''b''нb''b''ib'' b''кb''b''ob''b''лb''b''ab'')
```

Кругова або гвинтова дуга задається за допомогою *G2* (дуга за годинниковою стрілкою) або *G3* (дуга проти годинникової стрілки) при поточній feed rate. Напрямок (CW, CCW) визначається з позитивного кінця осі, навколо якої відбувається круговий рух.

Вісь кола або спіралі повинна бути паралельна осі X, Y або Z системи координат верстата. Вісь (або, що еквівалентно, площина, перпендикулярна до осі) вибирається за допомогою *G17* (вісь Z, площина XY), *G18* (вісь Y, площина XZ) або *G19* (вісь X, площина YZ). Площини «17.1», «18.1» і «19.1» наразі не підтримуються. Якщо дуга є круговою, вона лежить у площині, паралельній до вибраної площини.

Щоб запрограмувати спіраль, включіть слово осі, перпендикулярне до площини дуги, наприклад, якщо в площині «G17», включіть слово «Z». Це призведе до переміщення осі «Z» до запрограмованого значення під час кругового руху «XY».

Щоб запрограмувати дугу, яка дає більше одного повного оберту, використовуйте слово «P», вказавши кількість повних обертів плюс запрограмовану дугу. Слово «P» повинно бути цілим числом. Якщо «P» не вказано, поведінка буде такою, ніби було вказано «P1», тобто буде виконано тільки один повний або частковий оберт. Наприклад, якщо дуга 180 градусів запрограмована з P2, результатом руху буде 1 1/2 обертів. Для кожного приросту P вище 1 до запрограмованої дуги додається додаткове повне коло. Підтримуються багатообертові гвинтові дуги, які забезпечують рух, корисний для фрезерування отворів або різьблення.



Warning
Якщо крок спіралі дуже малий (менший за naive CAM tolerance), спіраль може бути перетворена на пряму лінію. Bug #222

Якщо рядок коду утворює дугу і включає рух обертової осі, обертові осі обертаються з постійною швидкістю, так що обертовий рух починається і закінчується, коли починається і закінчується рух XYZ. Рядки такого типу майже ніколи не програмуються.

Якщо корекція різця активна, рух відрізнятиметься від вищезазначеного; див. розділ Cutter Compensation.

Центр дуги може бути абсолютним або відносним, як встановлено відповідно G90.1 або G91.1.

Для визначення дуги дозволено два формати: формат центру та формат радіуса.

Це помилка, якщо:

- Швидкість подачі не встановлена.
- Слово на літеру P не є цілим числом.

11.5.5.1 Дуги формату центру

Дуги центрального формату є точнішими, ніж дуги радіусного формату, і є кращим форматом для використання.

Кінцева точка дуги разом із зміщенням до центру дуги від поточного положення використовуються для програмування дуг, які не є повним колом. Немає проблем, якщо кінцева точка дуги збігається з поточним положенням.

Зміщення до центру дуги від поточного положення та, за бажанням, кількість поворотів використовуют для програмування повних кіл.

Під час програмування дуг помилка через округлення може виникнути внаслідок використання точності менше 4 знаків після коми (0,0000) для дюймів та менше 3 знаків після коми (0,000) для міліметрів.

Режим поступового визначення відстані дуги Зміщення центру дуги – це відносна відстань від початкового положення дуги. За замовчуванням використовується режим збільшення відстані дуги.

Для дуги, меншої за 360 градусів, необхідно запрограмувати одне або декілька слів осей та одне або декілька зміщень.

Для повних кіл не потрібно програмувати жодних слів осей та одне або декілька зміщень. Слово «P» за замовчуванням дорівнює 1 і є необов'язковим.

Для отримання додаткової інформації про «Режим збільшення відстані дуги» див. розділ G91.1.

Режим абсолютної дугової відстані Зміщення центру дуги – це абсолютна відстань від поточного положення 0 осі.

Для дуг менше 360 градусів необхідно запрограмувати одне або декілька слів осей та зміщення «обидва».

Для повних кіл не потрібно програмувати слова осей та обидва зміщення. Слово «P» за замовчуванням дорівнює 1 і є необов'язковим.

Для отримання додаткової інформації про режим «Абсолютна дугова відстань» див. розділ G90.1.

Площина XY (G17)

```
G2 b''ab''b''6b''b''ob'' G3 <X- Y- Z- I- J- P->
```

- *Z* - helix
- *I* - X зміщення
- *J* - Y зміщення
- *P* - кількість обертів

XZ-площина (G18)

```
G2 b''ab''b''6b''b''ob'' G3 <X- Z- Y- I- K- P->
```

- *Y* - helix
- *I* - X зміщення
- *K* - Z зміщення
- *P* - кількість обертів

YZ-площина (G19)

```
G2 b''ab''b''6b''b''ob'' G3 <Y- Z- X- J- K- P->
```

- *X* - helix
- *J* - Y зміщення
- *K* - Z зміщення
- *P* - кількість обертів

Це помилка, якщо:

- Швидкість подачі не встановлюється за допомогою слова F.

- Зміщення не запрограмовано.
- Коли дуга проектується на вибрану площину, відстань від поточної точки до центру відрізняється від відстані від кінцевої точки до центру більш ніж на (.05 дюйма/.5 мм) АБО ((.0005 дюйма/.005 мм) І .1% радіуса).

Розшифровка повідомлення про помилку «Радіус до кінця дуги відрізняється від радіуса до початку:»

- *start* - поточна позиція
- *center* - центральне положення, розраховане за допомогою слів i, j або k
- *end* - запрограмована кінцева точка
- *r1* - радіус від початкової позиції до центру
- *r2* - радіус від кінцевого положення до центру

11.5.5.2 Приклади формату центру

Розрахунок дуг вручну іноді може бути складним. Один із варіантів — намалювати дугу за допомогою програми CAD, щоб отримати координати та зміщення. Зважаючи на зазначену вище похибку, можливо, доведеться змінити точність програми CAD, щоб отримати бажані результати. Інший варіант — обчислити координати та зміщення за допомогою формул. Як видно з наведених нижче рисунків, трикутник можна утворити з поточного положення, кінцевого положення та центру дуги.

На наступному малюнку ви можете бачити, що початкова позиція знаходиться в точці X0 Y0, а кінцева позиція — в точці X1 Y1. Центр дуги знаходиться в точці X1 Y0. Це дає нам зміщення від початкової позиції на 1 по осі X і на 0 по осі Y. У цьому випадку потрібно лише зміщення I.

Приклад Лінії G2

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (b''Дb''b''yb''b''гb''b''ab'' b''зb''b''ab'' b''гb''b''ob''b''дb''b''иb''b' ↩
    'нb''b''нb''b''иb''b''кb''b''ob''b''вb''b''ob''b''юb'' b''cb''b''тb''b''pb''b''ib''b' ↩
    'лb''b''кb''b''ob''b''юb'' b''вb'' b''пb''b''лb''b''ob''b''щb''b''иb''b''нb''b''ib'' XY)
```

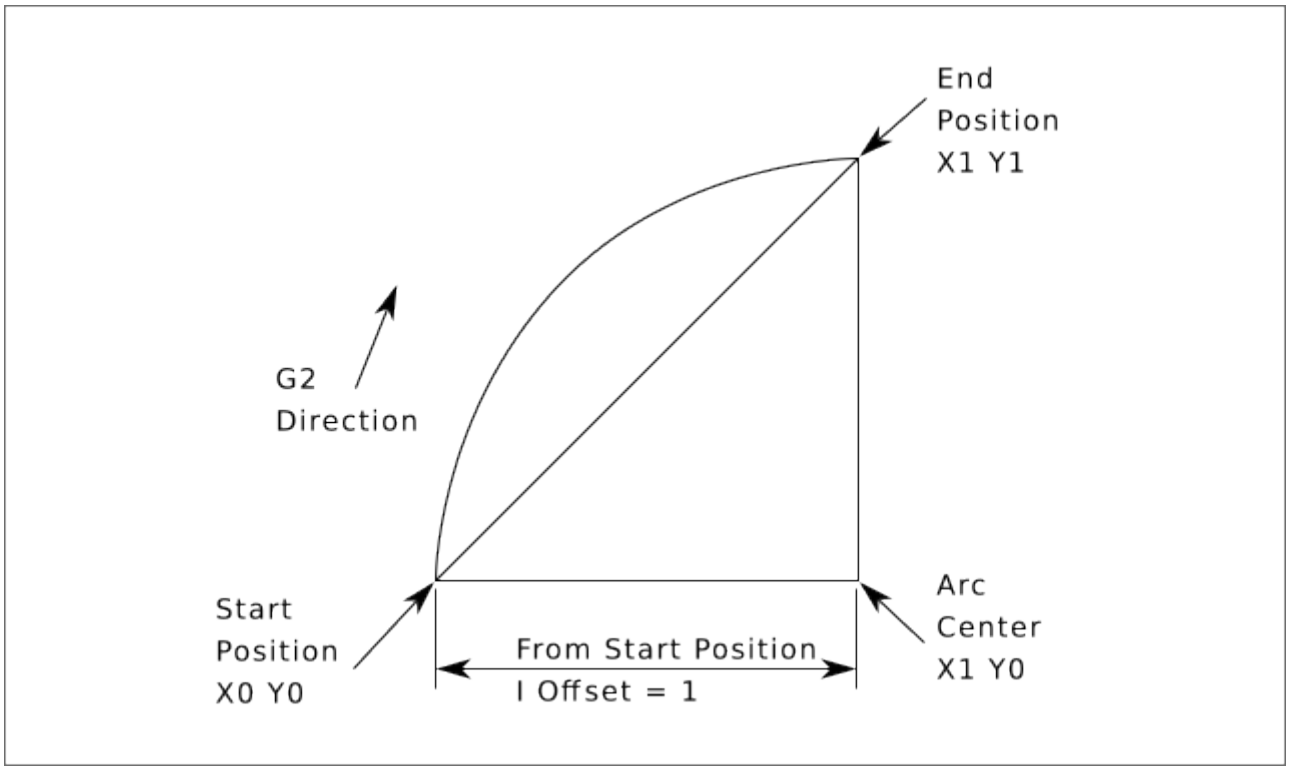


Figure 11.12: Приклад G2

У наступному прикладі ми бачимо різницю між зміщеннями для Y, якщо ми виконуємо рух G2 або G3. Для руху G2 початкове положення — X0 Y0, для руху G3 — X0 Y1. Центр дуги для обох рухів — X1 Y0,5. Для руху G2 зміщення J дорівнює 0,5, а для руху G3 — -0,5.

Приклад рядка G2-G3

```
G0 X0 Y0
G2 X0 Y1 I1 J0.5 F25 (b''дb''b''yb''b''гb''b''ab'' b''зb''b''ab'' b''гb''b''ob''b''дb''b' ↩
    'иb''b''нb''b''нb''b''иb''b''кb''b''ob''b''вb''b''ob''b''юb'' b''cb''b''тb''b''pb''b' ↩
    'ib''b''лb''b''кb''b''ob''b''юb'' b''вb'' b''пb''b''лb''b''ob''b''щb''b''иb''b''нb''b' ↩
    'ib'' XY)
G3 X0 Y0 I1 J-0.5 F25 (b''дb''b''yb''b''гb''b''ab'' b''пb''b''pb''b''ob''b''тb''b''иb'' b' ↩
    'гb''b''ob''b''дb''b''иb''b''нb''b''нb''b''иb''b''кb''b''ob''b''вb''b''ob''b''їb'' b' ↩
    'cb''b''тb''b''pb''b''ib''b''лb''b''кb''b''иb'' b''вb'' b''пb''b''лb''b''ob''b''щb''b' ↩
    'иb''b''нb''b''ib'' XY)
```

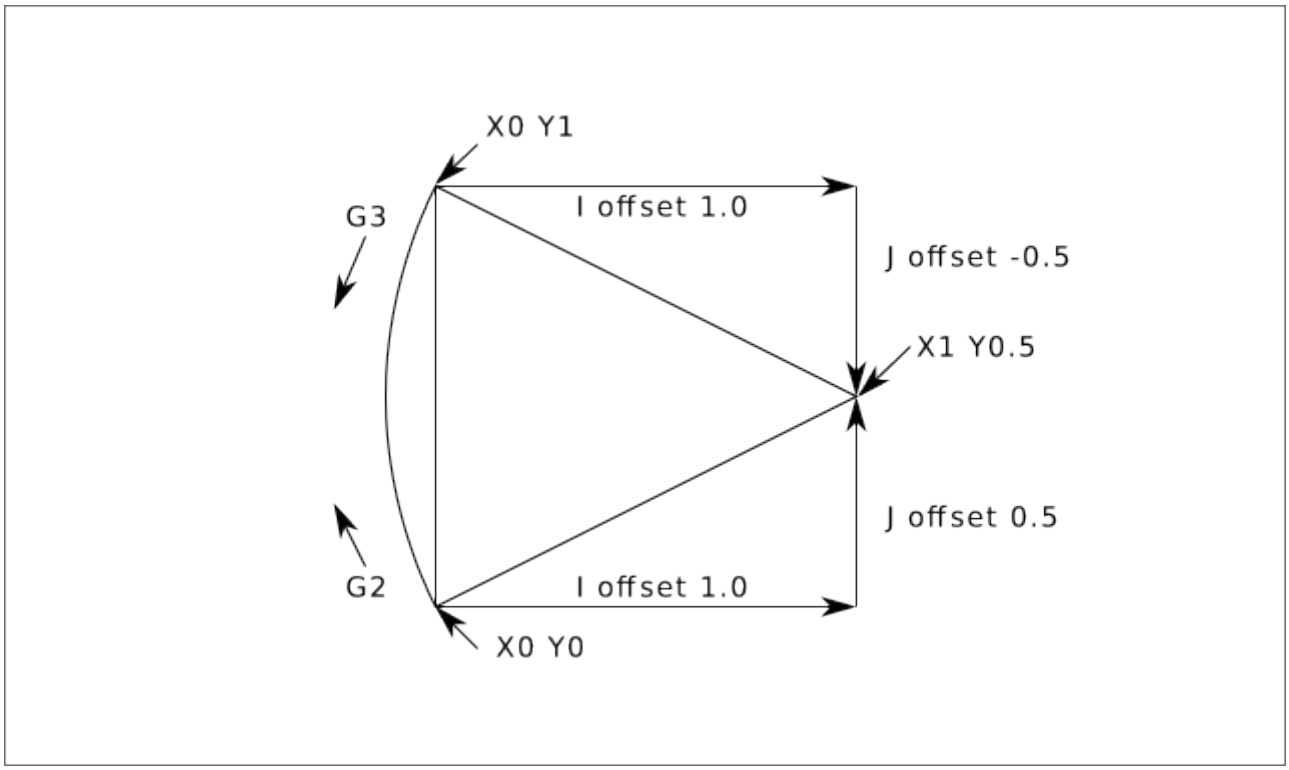


Figure 11.13: Приклад G2-G3

У наступному прикладі ми показуємо, як дуга може утворити спіраль по осі Z, додаючи слово Z.

Приклад спіралі G2

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (b''cb''b''пb''b''ib''b''pb''b''ab''b''лb''b''ьb''b''нb''b''ab'' b ↩
    ''дb''b''yb''b''гb''b''ab'' b''зb'' b''дb''b''ob''b''дb''b''ab''b''нb''b''ob''b''юb'' Z)
```

У наступному прикладі ми покажемо, як зробити більше одного повороту, використовуючи слово на літеру ”П”.

Приклад слова на букву ”П”

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

У центральному форматі радіус дуги не вказується, але його можна легко знайти як відстань від центру кола до поточної точки або кінцевої точки дуги.

11.5.5.3 Формат радіуса дуг

```
b''Ob''b''cb''b''ib'' G2 b''ab''b''6b''b''ob'' G3 R- <P->
```

- *R* - радіус від поточного положення

Не рекомендується програмувати дуги радіусного формату, які є майже повними колами або майже півколами, оскільки невелика зміна розташування кінцевої точки призведе до набагато більшої зміни розташування центру кола (а отже, і середини дуги). Ефект збільшення є настільки

великим, що похибка округлення числа може призвести до різання поза межами допуску. Наприклад, зміщення кінцевої точки дуги 180 градусів на 1% призвело до зміщення точки на 90 градусів уздовж дуги на 7%. Майже повні кола є ще гіршими. Дуги інших розмірів (в діапазоні від мінімальних до 165 градусів або від 195 до 345 градусів) є прийнятними.

У форматі радіуса вказуються координати кінцевої точки дуги у вибраній площині разом із радіусом дуги. Програма «G2» «осі» «R-» (або використовуйте «G3» замість «G2»). R — радіус. Слова осі є необов'язковими, за винятком того, що має бути використано принаймні одне з двох слів для осей у вибраній площині. Число R — це радіус. Додатний радіус означає, що дуга повертається менше ніж на 180 градусів, а від'ємний радіус означає поворот більше ніж на 180 градусів. Якщо дуга є гвинтовою, також вказується значення кінцевої точки дуги на координатній осі, паралельній осі гвинта.

Це помилка, якщо:

- обидва слова осей для осей вибраної площини пропущені
- Кінцева точка дуги така ж, як і поточна точка.

Приклад Лінії G2

```
G17 G2 X10 Y15 R20 Z5 (b''фb''b''ob''b''pb''b''мb''b''ab''b''тb'' b''pb''b''ab''b''дb''b' ↩
    'ib''b''yb''b''cb''b''ab'' b''зb'' b''дb''b''yb''b''гb''b''ob''b''юb'')
```

У наведеному вище прикладі створюється кругова або гвинтова дуга за годинниковою стрілкою (якщо дивитися від позитивної осі Z), вісь якої паралельна осі Z, що закінчується в точці X=10, Y=15 і Z=5, з радіусом 20. Якщо початкове значення Z дорівнює 5, це дуга кола, паралельна площині XY; в іншому випадку це гвинтова дуга.

11.5.6 G4 Житло

```
G4 P-
```

- *P* - секунди для затримки (число з плаваючою комою)

Число P – це час у секундах, протягом якого всі осі залишатимуться нерухомими. Число P – це число з плаваючою комою, тому можна використовувати частки секунди. G4 не впливає на шпиндель, охолоджувальну рідину та будь-які операції введення/виведення.

Приклад рядка G4

```
G4 P0.5 (b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'' 0,5 b''cb''b''eb'' ↩
    b''кb''b''yb''b''нb''b''дb''b''иb'', b''пb''b''eb''b''pb''b''шb'' b''нb''b''ib''b''жb''  ↩
    b''пb''b''pb''b''ob''b''дb''b''ob''b''вb''b''жb''b''иb''b''тb''b''иb'')
```

Це помилка, якщо:

- Число P від'ємне або не вказано.

11.5.7 Кубічний сплайн G5

```
G5 X- Y- <I- J-> P- Q-
```

- *I* - приростне зміщення по осі X від початкової точки до першої контрольної точки

- *J* - приростне зміщення по осі Y від початкової точки до першої контрольної точки
- *P* - приростне зміщення по осі X від кінцевої точки до другої контрольної точки
- *Q* - приростне зміщення по осі Y від кінцевої точки до другої контрольної точки

G5 створює кубічний B-сплайн у площині XY лише з осями X та Y. P та Q необхідно вказати для кожної команди G5.

Для першої команди G5 у серії команд G5 необхідно вказати як I, так і J. Для наступних команд G5 необхідно вказати або I, або J, або ж не вказувати жодної з них. Якщо I та J не вказані, початковий напрямок цього куба автоматично відповідатиме кінцевому напрямку попереднього куба (ніби I та J є запереченням попередніх P та Q).

Наприклад, щоб запрограмувати вигнуту N-подібну фігуру:

G5 Зразок початкового кубічного сплайна

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

Другу вигнуту N, яка плавно приєднується до цієї, тепер можна зробити без вказівки I та J:

G5 Зразок наступного кубічного сплайна

```
G5 P0 Q-3 X2 Y2
```

Це помилка, якщо:

- P та Q не є одночасно заданими.
- Вказано лише один з I або J.
- I або J не вказані в першій з серії команд G5.
- Вказано вісь, відмінну від X або Y.
- Активна площина не G17.

11.5.8 G5.1 Квадратний сплайн

```
G5.1 X- Y- I- J-
```

- *I* - приростне зміщення по осі X від початкової точки до контрольної точки
- *J* - приростне зміщення по осі Y від початкової точки до контрольної точки

G5.1 створює квадратичний B-сплайн у площині XY лише з осями X та Y. Якщо I або J не вказано, зміщення для невказаної осі буде нульовим, тому необхідно вказати одну або обидві.

Наприклад, щоб запрограмувати параболу через початок координат від X-2 Y4 до X2 Y4:

G5.1 Зразок квадратичного сплайна

```
G90 G17
G0 X-2 Y4
G5.1 X2 I2 J-8
```

Це помилка, якщо:

- як зсув I, так і зсув J не визначені або не відповідають нулю
- Вказано вісь, відмінну від X або Y
- Активна площина не G17

11.5.9 G5.2 G5.3 Блок NURBS

```
G5.2 <P-> <X- Y-> <L->
X- Y- <P->
...
G5.3
```



Warning
G5.2, G5.3 є експериментальним варіантом і не пройшов повного тестування.

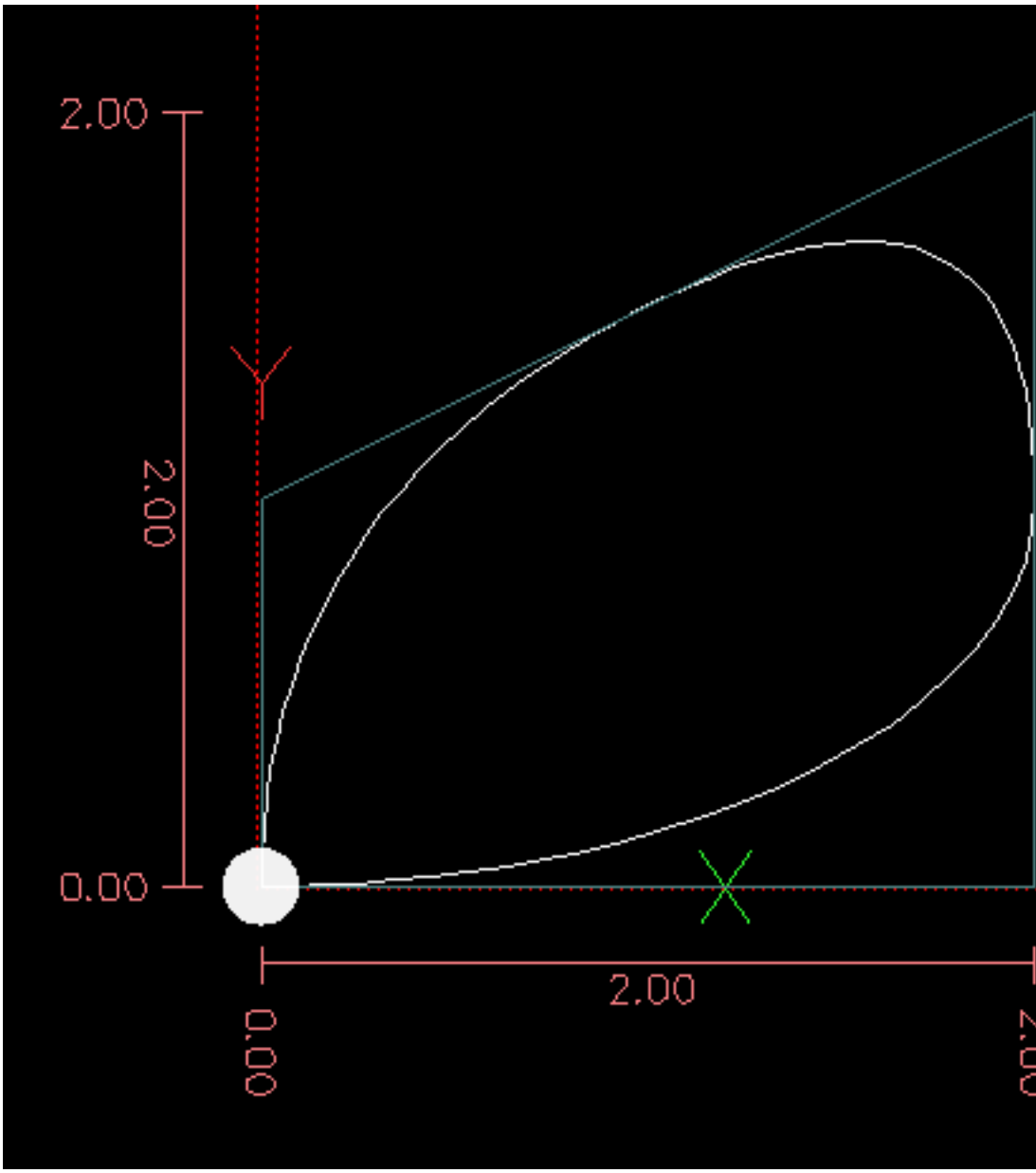
G5.2 призначений для відкриття блоку даних, що визначає NURBS, а G5.3 — для закриття блоку даних. У рядках між цими двома кодами визначаються контрольні точки кривої з відповідними «вагами» (P) і параметром (L), що визначає порядок кривої.

Поточна координата, перед першою командою G5.2, завжди береться за першу контрольну точку NURBS. Щоб встановити вагу для цієї першої контрольної точки, спочатку запрограмуйте G5.2 P- без вказівки жодних X Y.

Вага за замовчуванням, якщо P не вказано, дорівнює 1. Порядок за замовчуванням, якщо L не вказано, дорівнює 3.

Приклад G5.2

```
G0 X0 Y0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''рb''b''уb''b''хb'')
F10 (b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' b''шb''b''вb''b ↩
    ''иb''b''дb''b''кb''b''іb''b''сb''b''тb''b''ьb'' b''пb''b''оb''b''дb''b''аb''b''чb''b' ↩
    'іb'')
G5.2 P1 L3
     X0 Y1 P1
     X2 Y2 P1
     X2 Y0 P1
     X0 Y0 P2
G5.3
; b''Шb''b''вb''b''иb''b''дb''b''кb''b''іb'' b''пb''b''еb''b''рb''b''еb''b''мb''b''іb''b' ↩
    'щb''b''еb''b''нb''b''нb''b''яb'' b''пb''b''оb''b''кb''b''аb''b''зb''b''уb''b''юb''b' ↩
    'тb''b''ьb'' b''тb''b''оb''b''йb'' b''сb''b''аb''b''мb''b''иb''b''йb'' b''шb''b''лb''b' ↩
    'яb''b''хb'' b''бb''b''еb''b''зb'' b''бb''b''лb''b''оb''b''кb''b''уb'' NURBS
G0 X0 Y1
   X2 Y2
   X2 Y0
   X0 Y0
M2
```



Зразок виводу NURBS

Більше інформації про NURBS можна знайти тут:

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

11.5.10 G7 Режим діаметра токарного верстата

```
G7
```

Запрограмуйте G7 для входу в режим діаметра для осі X на токарному верстаті. У режимі діаметра ось X переміщується на токарному верстаті на відстань, що дорівнює 1/2 відстані до центру токарного верстата. Наприклад, X1 перемістить різак на 0,500” від центру токарного верстата, отримавши деталь діаметром 1”.

11.5.11 G8 Режим радіуса токарного верстата

```
G8
```

Програма G8 для входу в режим радіуса для осі X на токарному верстаті. У режимі радіуса ось X рухається на токарному верстаті на відстань від центру. Таким чином, різання в точці X1 дасть деталь діаметром 2 дюйма. G8 є стандартним налаштуванням при ввімкненні.

11.5.12 Дані таблиці інструментів для перезавантаження G10 L0

```
G10 L0
```

G10 L0 перезавантажує всі дані таблиці інструментів. Потрібно, щоб у шпинделі не було завантажено жодного інструменту.

Note
При використанні G10 L0 параметри інструменту (#5401-#5413) будуть негайно оновлені, а будь-які змінені діаметри інструменту будуть використовуватися для наступних команд компенсації радіуса різака G41,42. Існуючі значення компенсації довжини інструменту G43 залишатимуться в силі до оновлення новими командами G43.

11.5.13 Таблиця інструментів G10 L1

```
G10 L1 P- axes <R- I- J- Q->
```

- *P* - номер інструменту
- *R* - радіус інструмента
- *I* - передній кут (токарний верстат)
- *J* - задній кут (токарний верстат)
- *Q* - орієнтація (токарний верстат)

G10 L1 встановлює таблицю інструментів для номера інструмента *P* на значення слів.

Дійсний код G10 L1 перезаписує та повторно завантажує таблицю інструментів для зазначеного інструменту.

Приклад лінії G10 L1

```
G10 L1 P1 Z1.5 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
    'зb''b''мb''b''іb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''іb''b''нb''b''cb''b''тb''b' ↩
    'рb''b''уb''b''мb''b''eb''b''нb''b''тb''b''уb'' 1 b''пb''b''ob'' b''ob''b''cb''b''іb'' Z ↩
     b''вb''b''іb''b''дb'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''уb'' b''кb''b''ob''b ↩
    ''ob''b''рb''b''дb''b''иb''b''нb''b''ab''b''тb'' b''вb''b''eb''b''рb''b''cb''b''тb''b' ↩
    'ab''b''тb''b''ab'' b''нb''b''ab'' 1.5)
G10 L1 P2 R0.015 Q3 (b''пb''b''рb''b''иb''b''кb''b''лb''b''ab''b''дb'' b''тb''b''ob''b' ↩
    'кb''b''ab''b''рb''b''нb''b''ob''b''гb''b''ob'' b''вb''b''eb''b''рb''b''cb''b''тb''b' ↩
    'ab''b''тb''b''ab'': b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b' ↩
    'иb'' b''рb''b''ab''b''дb''b''іb''b''уb''b''cb'' b''іb''b''нb''b''cb''b''тb''b''рb''b' ↩
    'уb''b''мb''b''eb''b''нb''b''тb''b''уb'' 2 b''нb''b''ab'' 0.015 b''тb''b''ab'' b''ob''b' ↩
    'рb''b''іb''b''єb''b''нb''b''тb''b''ab''b''цb''b''іb''b''юb'' b''нb''b''ab'' 3)
```

Це помилка, якщо:

- Компенсація різця увімкнена

- Номер P не вказано
- Номер P не є дійсним номером інструменту з таблиці інструментів
- Число P дорівнює 0

Для отримання додаткової інформації про орієнтацію різця, що використовується словом *Q*, див. діаграму Lathe Tool Orientation.

11.5.14 G10 L2 Встановити систему координат

```
G10 L2 P- <axes R->
```

- *P* - система координат (0-9)
- *R* - обертання навколо осі Z

G10 L2 зміщує початок координат у заданій системі координат на значення слова осі. Зсув відбувається від початку координат верстата, встановленого під час повернення в початкове положення. Значення зсуву замінить усі поточні зсуви, що діють для заданої системи координат. Невикористані слова осі не змінюються.

Запрограмуйте P0 - P9, щоб вказати, яку систему координат потрібно змінити.

Table 11.14: Система координат

Значення P	**Система координат**	**G-код**
0	Активний	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

За потреби запрограмуйте R для позначення обертання осі XY навколо осі Z. Напрямок обертання — проти годинникової стрілки, якщо дивитися з позитивного кінця осі Z.

Усі слова осі є необов’язковими.

Перебування в режимі поступового збільшення відстані (G91) не впливає на *G10 L2*.

Важливі поняття:

- `G10 L2 Pn` не змінюється з поточної системи координат на ту, що визначена P, для вибору системи координат потрібно використовувати G54-59.3.
- Під час обертання виконується штовхання, вісь переміщуватиме цю вісь лише в позитивному або негативному напрямку, а не вздовж осі, що обертається.

- Якщо локальне зміщення *G52* або зміщення початку *G92* діяло до *G10 L2*, воно продовжуватиме діяти і після цього.
- Під час програмування системи координат за допомогою R, будь-які команди *G52* або *G92* будуть застосовані **після** обертання.
- Система координат, початок якої встановлюється командою «G10», може бути активною або неактивною на момент виконання «G10». Якщо вона наразі активна, нові координати набувають чинності негайно.

Це помилка, якщо:

- Число P не дорівнює цілому числу в діапазоні від 0 до 9.
- Запрограмовано вісь, яка не визначена в конфігурації.

Приклад лінії G10 L2

```
G10 L2 P1 X3.5 Y17.2
```

У наведеному вище прикладі початок першої системи координат (вибраної за допомогою «G54») встановлено як X=3,5 і Y=17,2. Оскільки вказано тільки X і Y, точка початку переміщується тільки по X і Y; інші координати не змінюються.

Приклад лінії G10 L2

```
G10 L2 P1 X0 Y0 Z0 (b''чb''b''ib''b''тb''b''кb''b''ib'' b''зb''b''мb''b''ib''b''щb''b''eb'' ↩
    b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''ob''b''cb''b''eb''b''йb'' X, Y b''тb''b' ↩
    'ab'' Z b''yb'' b''cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''ib'' b''кb''b''ob''b''ob''b ↩
    ''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' 1)
```

У наведеному вище прикладі координати XYZ системи координат 1 встановлюються на початок координат машини.

Система координат описана в розділі Coordinate System.

11.5.15 Таблиця інструментів G10 L10

```
G10 L10 P- axes <R- I- J- Q->
```

- *P* - номер інструменту
- *R* - радіус інструмента
- *I* - передній кут (токарний верстат)
- *J* - задній кут (токарний верстат)
- *Q* - орієнтація (токарний верстат)

G10 L10 змінює запис в таблиці інструментів для інструменту P таким чином, що при перезавантаженн зміщення інструменту, коли верстат знаходиться в поточному положенні і активні поточні зміщення G52 і /G92, поточні координати для даних осей стануть заданими значеннями. Осі, які не вказані в команді G10 L10, не будуть змінені. Це може бути корисно при переміщенні датчика, як описано в розділі G38.

Приклад G10 L10

```
T1 M6 G43 (b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb'' 1 b''тb''b''ab'' b' ↩
    'зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob''b''вb''b''жb''b' ↩
    'иb''b''нb''b''иb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b' ↩
    'тb''b''ab'')
G10 L10 P1 Z1.5 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
    'пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''yb'' b''пb''b''ob''b''зb''b''иb''b''цb''b' ↩
    'ib''b''юb'' b''дb''b''лb''b''яb'' Z b''нb''b''ab'' 1.5)
G43 (b''пb''b''eb''b''pb''b''eb''b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b' ↩
    'иb''b''тb''b''иb'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b' ↩
    'ob''b''вb''b''жb''b''иb''b''нb''b''иb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b' ↩
    'мb''b''eb''b''нb''b''тb''b''ab'' b''зb''b''ib'' b''зb''b''мb''b''ib''b''нb''b''eb''b' ↩
    'нb''b''ob''b''їb'' b''тb''b''ab''b''бb''b''лb''b''иb''b''цb''b''ib'' b''ib''b''нb''b' ↩
    'cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''ib''b''вb'')
M2 (b''зb''b''ab''b''кb''b''ib''b''нb''b''чb''b''иb''b''тb''b''иb'' b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''yb'')
```

- Див. розділи T та M6, а також G43/G43.1 для отримання додаткової інформації.

Це помилка, якщо:

- Компенсація різця увімкнена
- Номер P не вказано
- Номер P не є дійсним номером інструменту з таблиці інструментів
- Число P дорівнює 0

11.5.16 G10 L11 Набір інструментів Таблиця

```
G10 L11 P- axes <R- I- J- Q->
```

- *P* - номер інструменту
- *R* - радіус інструмента
- *I* - передній кут (токарний верстат)
- *J* - задній кут (токарний верстат)
- *Q* - орієнтація (токарний верстат)

G10 L11 аналогічний G10 L10, за винятком того, що замість встановлення запису відповідно до поточних зміщень, він встановлюється таким чином, що поточні координати стануть заданим значенням, якщо буде перезавантажено нове зміщення інструменту і верстат буде розміщено в системі координат G59.3 без будь-якого активного зміщення G52/G92.

Це дозволяє користувачеві встановити систему координат G59.3 відповідно до фіксованої точки на верстаті, а потім використовувати це пристосування для вимірювання інструментів без урахування інших активних наразі зміщень. Це помилка, якщо:

- Компенсація різця увімкнена
- Номер P не вказано
- Номер P не є дійсним номером інструменту з таблиці інструментів
- Число P дорівнює 0

11.5.17 G10 L20 Встановити систему координат

```
G10 L20 P- axes
```

- *P* - система координат (0-9)

G10 L20 подібний до G10 L2, за винятком того, що замість встановлення зміщення/введення на задане значення, воно встановлюється на обчислене значення, яке робить поточні координати заданим значенням.

Приклад лінії G10 L20

```
G10 L20 P1 X1.5 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
    'пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''eb'' b''пb''b''ob''b''лb''b''ob''b''жb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''ob''b''cb''b''ib'' X b''уb'' b''cb''b''иb''b''cb''b''тb''b ↩
    ''eb''b''мb''b''ib'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' 1 b ↩
    ''нb''b''ab'' 1,5)
```

Це помилка, якщо:

- Число P не дорівнює цілому числу в діапазоні від 0 до 9.
- Запрограмовано вісь, яка не визначена в конфігурації.

11.5.18 G17 - G19.1 Вибір площини

Ці коди встановлюють поточну площину наступним чином:

- *G17* - XY (за замовчуванням)
- *G18* - ZX
- *G19* - YZ
- *G17.1* - UV
- *G18.1* - WU
- *G19.1* - VW

Площини UV, WU та VW не підтримують дуги.

Гарною ідеєю є включення вибору площини у преамбулу кожного файлу G-коду.

Вплив вибору площини обговорюється в розділах G2 G3 Arcs та G81 G89.

11.5.19 Одиниці G20, G21

- *G20* - використовувати дюйми для одиниць вимірювання довжини.
- «G21» – використовувати міліметри для одиниць вимірювання довжини.

Бажано включати одиниці вимірювання у преамбулу кожного файлу G-коду.

11.5.20 G28, G28.1 Перейти/Встановити попередньо визначене положення

Warning
Використовуйте G28 лише тоді, коли ваш верстат переведений у вихідне положення з повторюваною позицією, а бажане положення G28 було збережено за допомогою G28.1.

G28 використовує значення, збережені в parameters 5161-5169, як кінцеву точку X Y Z A B C U V W, до якої потрібно переміститися. Значення параметрів є «абсолютними» координатами машини в «одиницях виміру» машини, як зазначено в файлі INI. Усі осі, визначені в файлі INI, будуть переміщені при видачі команди G28. Якщо в G28.1 не збережено жодних позицій, то всі осі перемістяться до machine origin.

- *G28* - виконує rapid move з поточної позиції до *абсолютної* позиції значень у параметрах 5161-5166.
- «G28 axes» — виконує швидкий рух до позиції, вказаної «axes», включаючи будь-які зміщення, а потім виконує швидкий рух до «абсолютної» позиції значень у параметрах 5161-5166 для всіх вказаних «axes». Будь-яка «axis», що не вказана, не рухатиметься.
- *G28.1* - зберігає поточну «абсолютну» позицію в параметрах 5161-5166.

Приклад рядка G28

```
G28 Z2.5 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' Z2.5, b''пb''b''ob''b''тb''b' ↩
    'ib''b''мb'' b''дb''b''ob'' b''пb''b''ob''b''зb''b''иb''b''цb''b''ib''b''їb'' Z, b''зb'' ↩
    b''ab''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''ob''b''їb'' b''вb'' #5163)
```

Це помилка, якщо:

- Компенсацію різця ввімкнено

11.5.21 G30, G30.1 Перейти/Встановити попередньо визначену позицію

Warning
Використовуйте G30 лише тоді, коли ваш верстат переведений у вихідне положення з повторюваною позицією, а бажане положення G30 було збережено за допомогою G30.1.

G30 працює так само, як G28, але використовує значення, збережені в parameters 5181-5189, як кінцеву точку X Y Z A B C U V W, до якої потрібно переміститися. Значення параметрів є «абсолютними» координатами верстата в «одиницях виміру» верстата, як зазначено в файлі INI. Всі осі, визначені в файлі INI, будуть переміщені при видачі команди G30. Якщо в G30.1 не збережено жодних позицій, то всі осі перемістяться до machine origin.

Note
Параметри G30 будуть використовуватися для переміщення інструменту, коли запрограмовано M6, якщо TOOL_CHANGE_AT_G30=1 знаходиться в розділі [EMCIO] INI-файлу.

- *G30* - виконує rapid move з поточної позиції до *абсолютної* позиції значень у параметрах 5181-5189.
- «G30 axes» — виконує швидкий рух до позиції, вказаної «axes», включаючи будь-які зміщення, а потім виконує швидкий рух до «абсолютної» позиції значень у параметрах 5181-5189 для всіх вказаних «axes». Будь-яка «axis», що не вказана, не рухатиметься.
- *G30.1* - зберігає поточну абсолютну позицію в параметрах 5181-5186.

Приклад рядка G30

```
G30 Z2.5 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' Z2.5, b''пb''b''ob''b''тb''b' ↩
    'ib''b''мb'' b''дb''b''ob'' b''тb''b''ob''b''чb''b''кb''b''иb'' Z, b''зb''b''ab''b''зb'' ↩
    b''нb''b''ab''b''чb''b''eb''b''нb''b''ob''b''їb'' b''вb'' #5183)
```

Це помилка, якщо:

- Компенсацію різця ввімкнено

11.5.22 G33 Синхронізований рух шпинделя

```
G33 X- Y- Z- K- $-
```

- *K* - відстань за оберт

Для синхронізованого руху шпинделя в одному напрямку введіть код «G33 X- Y- Z- K-», де K вказує відстань, пройдена в XYZ за кожен оберт шпинделя. Наприклад, якщо почати з «Z=0», «G33 Z-1 K.0625» створює рух на 1 дюйм в Z за 16 обертів шпинделя. Ця команда може бути частиною програми для створення різьби 16TPI. Інший приклад у метричній системі: «G33 Z-15 K1.5» створює рух на 15 мм, коли шпиндель обертається 10 разів для різьби 1,5 мм.

Аргумент $ (необов'язковий) встановлює, з яким шпинделем синхронізується рух (за замовчуванням дорівнює нулю). Наприклад, G33 Z10 K1 $1 переміщуватиме шпиндель синхронно зі значенням виводу HAL spindle.N.revs.

Рух, синхронізований зі шпинделем, очікує на індекс шпинделя та штифти швидкості шпинделя, тому кілька проходів вирівнюються. «G33» переміщує кінець у запрограмовану кінцеву точку. G33 можна використовувати для різання конічних різьблень або фузеї.

Усі слова осі є необов'язковими, окрім того, що принаймні одне має бути використане.

Note
Осі K відповідає лінії приводу, описаній як *X-Y-Z-*. Осі K не паралельна осі Z, якщо кінцеві точки X або Y використовуються, наприклад, під час нарізання конічної різьби.

Технічна інформація

На початку кожного проходу G33 LinuxCNC використовує швидкість шпинделя та обмеження прискорення верстата, щоб обчислити, скільки часу знадобиться Z для прискорення після імпульсу індексації, і визначає, на скільки градусів обернеться шпиндель за цей час. Потім він додає цей кут до позиції індексації та обчислює позицію Z, використовуючи скоригований кут шпинделя. Це означає, що Z досягне правильного положення одразу після завершення прискорення до належної швидкості і зможе негайно розпочати різання якісної різьби.

З'єднання HAL Для запуску руху необхідно встановити або активувати контакт «spindle.N.at-speed». Крім того, значення spindle.N.revs повинно збільшуватися на 1 за кожен оберт шпинделя,

а контакт spindle.N.index-enable повинен бути підключений до лічильника енкодера (або резольвера), який скидає індекс-активацію один раз за оберт.

Див. посібник інтегратора для отримання додаткової інформації про синхронізований рух шпинделя.

Приклад G33

```
G90 (b''pb''b''eb''b''жb''b''иb''b''мb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b ↩
    ''нb''b''ob''b''їb'' b''вb''b''іb''b''дb''b''cb''b''тb''b''ab''b''нb''b''іb'')
G0 X1 Z0.1 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b ↩
    ''іb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''вb'' b''пb''b''ob''b''лb''b''ob''b''жb''b' ↩
    'eb''b''нb''b''нb''b''яb'')
S100 M3 (b''зb''b''ab''b''пb''b''уb''b''cb''b''кb'' b''ob''b''бb''b''eb''b''pb''b''тb''b' ↩
    'ab''b''нb''b''нb''b''яb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb'')
G33 Z-2 K0.125 (b''пb''b''eb''b''pb''b''eb''b''мb''b''іb''b''щb''b''eb''b''нb''b''нb''b' ↩
    'яb'' b''ob''b''cb''b''іb'' Z b''дb''b''ob'' -2 b''зb''b''іb'' b''шb''b''вb''b''иb''b' ↩
    'дb''b''кb''b''іb''b''cb''b''тb''b''юb'' 0,125 b''нb''b''ab'' b''ob''b''бb''b''eb''b' ↩
    'pb''b''тb'')
G0 X1.25 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'іb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''іb''b''нb''b''cb''b''тb''b''pb''b''уb''b' ↩
    'мb''b''eb''b''нb''b''тb''b''уb'' b''вb''b''іb''b''дb'' b''зb''b''ab''b''гb''b''ob''b' ↩
    'тb''b''ob''b''вb''b''кb''b''иb'')
Z0.1 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''іb'' ↩
    b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' b''пb''b''ob''b''чb''b''ab''b''тb''b' ↩
    'кb''b''ob''b''вb''b''ob''b''їb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb'' Z)
M2 (b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''pb''b' ↩
    'ob''b''гb''b''pb''b''ab''b''мb''b''иb'')
```

- Див. розділи G90 та G0 та M2 для отримання додаткової інформації.

Це помилка, якщо:

- Усі слова осі пропущені.
- Шпиндель не обертається, коли виконується ця команда.
- Запитаний лінійний рух перевищує обмеження швидкості машини через швидкість шпинделя.

11.5.23 G33.1 Жорстке нарізання різьби

```
G33.1 X- Y- Z- K- I- $-
```

- *K* - відстань за оберт
- *I* - додатковий множник швидкості шпинделя для швидшого зворотного руху
- *$* - додатковий селектор шпинделя



Warning
Для Z тільки натискання препозиції XY перед викликом G33.1 і тільки використання слова Z в G33.1. Якщо вказані координати не є поточними координатами при виклику G33.1 для натискання, рух не буде вздовж осі Z, а буде скоординованим, синхронізованим з веретеном рухом від поточного місця до вказаного місця і назад.

Для жорсткого нарізання різьби (синхронізований рух шпинделя з поверненням) використовується код *G33.1 X- Y- Z- K-*, де *K-* вказує на відстань, що переміщується за кожен оберт шпинделя.

Рух жорсткого постукування складається з наступної послідовності:

- Переміщення від поточної координати до заданої координати, синхронізоване з вибраним шпинделе у заданому співвідношенні та починаючи з поточної координати за допомогою імпульсу індексу шпинделя.
- Після досягнення кінцевої точки, команда на реверс шпинделя та збільшення прискорення на коефіцієнт, встановлений множником (наприклад, з годинникової стрілки на проти годинникової стрілки).
- Продовження синхронізованого руху за межі заданої кінцевої координати, доки шпиндель фактично не зупиниться та не почне обертатися у зворотному напрямку.
- Продовження синхронізованого руху назад до початкової координати.
- Після досягнення початкової координати, команда на реверс шпинделя вдруге (наприклад, з руху проти годинникової стрілки на рух за годинниковою стрілкою).
- Продовження синхронізованого руху за межі початкової координати, доки шпиндель фактично не зупиниться та не почне обертатися у зворотному напрямку.
- **Несинхронізований** рух назад до початкової координати.

Синхронізовані зі шпинделем рухи очікують на індекс шпинделя, тому кілька проходів вишиковуються. Рухи *G33.1* завершуються у вихідній координаті.

Усі слова осі є необов’язковими, окрім того, що принаймні одне має бути використане.

Приклад G33.1

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''ab''b''бb''b ↩
    ''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b''иb''b' ↩
    'мb'')
G0 X1.000 Y1.000 Z0.100 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''вb'' b''пb''b''ob''b''чb''b' ↩
    'ab''b''тb''b''кb''b''ob''b''вb''b''eb'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b' ↩
    'нb''b''нb''b''яb'')
S100 M3 (b''yb''b''вb''b''ib''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''шb''b''пb''b' ↩
    'иb''b''нb''b''дb''b''eb''b''лb''b''ьb'', 100 b''ob''b''бb''/b''xb''b''вb'')
G33.1 Z-0.750 K0.05 (b''жb''b''ob''b''pb''b''cb''b''тb''b''кb''b''eb'' b''нb''b''ab''b' ↩
    'pb''b''ib''b''зb''b''ab''b''нb''b''нb''b''яb'' b''pb''b''ib''b''зb''b''ьb''b''бb''b' ↩
    'иb'' 20 TPI b''гb''b''лb''b''иb''b''бb''b''иb''b''нb''b''ob''b''юb'' 0.750)
M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''иb'')
```

- Див. розділи G90 та G0 та M2 для отримання додаткової інформації.

Це помилка, якщо:

- Усі слова осі пропущені.
- Шпиндель не обертається, коли виконується ця команда
- Запитаний лінійний рух перевищує обмеження швидкості машини через швидкість шпинделя

11.5.24 G38.*n* Прямий зонд

```
G38.n axes
```

- *G38.2* - зонд до заготовки, зупинка при контакті, сигналізація про помилку у разі невдачі

- *G38.3* - зонд до заготовки, зупинка при контакті
- *G38.4* - зонд віддалений від заготовки, зупинка при втраті контакту, сигналізація про помилку у разі несправності
- *G38.5* - зонд віддаліть від заготовки, зупиніться при втраті контакту



Important
Ви не зможете використовувати рух зонда, поки ваша машина не буде налаштована на надання вхідного сигналу зонда. Вхідний сигнал зонда повинен бути підключений до *motion.probe-input* у файлі .hal. G38.*n* використовує motion.probe-input для визначення, коли зонд встановив (або втратив) контакт. TRUE для закритого контакту зонда (дотик), FALSE для відкритого контакту зонда.

Програма «G38.*n* axes» для виконання прямої операції зондування. Слова осі є необов'язковими, але принаймні одне з них має бути використано. Слова осі разом визначають кінцеву точку, до якої зонд буде рухатися, починаючи з поточного місця розташування. Якщо зонд не спрацює до досягнення кінцевої точки, G38.2 і G38.4 подадуть сигнал про помилку.

Інструмент у шпинделі повинен бути зондом або контактувати з перемикачем зонда.

У відповідь на цю команду верстат переміщує контрольовану точку (яка повинна знаходитися в центрі кулі зонда) по прямій лінії з поточним feed rate у напрямку до запрограмованої точки. У режимі зворотного часу подачі швидкість подачі така, що весь рух від поточної точки до запрограмованої точки займає заданий час. Рух зупиняється (в межах обмежень прискорення верстата) при досягненні запрограмованої точки або при запитуваній зміні вхідних даних зонда, залежно від того, що відбудеться раніше.

Table 11.15: Зондування G-кодів

Код	**Цільовий стан**	**Перемістити орієнтацію**	**Сигнал помилки**
G38.2	Торкнувся	До шматка	Так
G38.3	Торкнувся	До шматка	Ні
G38.4	Недоторканий	З шматка	Так
G38.5	Недоторканий	З шматка	Ні

Після успішного зондування параметри #5061 до #5069 будуть встановлені на координати X, Y, Z, A, B, C, U, V, W розташування контрольованої точки в момент зміни стану зонда (в поточній системі координат роботи). Після невдалого зондування вони встановлюються на координати запрограмованої точки. Параметр 5070 встановлюється на 1, якщо зондування пройшло успішно, і на 0, якщо зондування не пройшло успішно. Якщо операція зондування не пройшла успішно, G38.2 і G38.4 сигналізують про помилку, виводячи повідомлення на екран, якщо вибраний графічний інтерфейс користувача підтримує таку функцію. А також зупиняючи виконання програми.

Ось приклад формули для вимірювання висоти інструменту з перетворенням зміщення Z з локальної системи координат в координати верстата, які зберігаються в таблиці інструментів. Існуюча компенсація висоти інструменту спочатку скасовується за допомогою G49, щоб уникнути її включення в розрахунок висоти, а нова висота завантажується з таблиці інструментів. Початкова позиція повинна бути достатньо високою над датчиком висоти інструменту, щоб компенсувати використання G49.

Приклад G38.2

```
G49
G38.2 Z-100 F100
#<zworkoffset> = [#[5203 + #5220 * 20] + #5213 * #5210]
G10 L1 P#5400 Z#<zworkoffset> (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b' ↩
    'тb''b''иb'' b''нb''b''ob''b''вb''b''eb'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b' ↩
    'ab'')
G43
```

Коментар у формі «(PROBEOPEN filename.txt)» відкриє файл «filename.txt» і збереже в ньому 9-значну координату, що складається з XYZABCUVW, кожного успішного прямого зондування. Файл необхідно закрити за допомогою «(PROBECLOSE)». Для отримання додаткової інформації див. розділ Comments.

Приклад файлу «smartprobe.ngc» включений (у каталозі прикладів) для демонстрації використання переміщень зонда для запису координат деталі у файл. Програма «smartprobe.ngc» може використовуватися з «ngcgui» з мінімальними змінами.

Це помилка, якщо:

- поточна точка така ж, як запрограмована точка.
- жодне слово осі не використовується
- корекція різця увімкнена
- швидкість подачі дорівнює нулю
- зонд вже знаходиться в цільовому стані

11.5.25 Компенсація G40 вимкнена

- *G40* - Вимкніть компенсацію різця. Якщо компенсація інструменту була ввімкнена, наступний рух має бути лінійним і довшим за діаметр інструменту. Можна вимкнути компенсацію, якщо вона вже вимкнена.

Приклад G40

```
; b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''eb'' b''пb''b''ob''b''лb''b''ob''b''жb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''—b'' X1 b''пb''b''ib''b''cb''b''лb''b''яb'' b''зb''b''ab'' ↩
    b''вb''b''eb''b''pb''b''шb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''eb''b''pb''b''eb''b' ↩
    'мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''зb'' b''кb''b''ob''b''мb''b''пb''b' ↩
    'eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''єb''b''юb'' b''нb''b''ab'' b''pb''b''ib''b' ↩
    'зb''b''eb''b''цb''b''ьb''
G40 (b''вb''b''иb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b ↩
    ''eb''b''нb''b''cb''b''ab''b''цb''b''ib''b''юb'')
G0 X1.6 (b''лb''b''ib''b''нb''b''ib''b''йb''b''нb''b''eb'' b''пb''b''eb''b''pb''b''eb''b' ↩
    'мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob''b''вb''b''шb''b''иb''b' ↩
    'eb'' b''зb''b''ab'' b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''иb''b''йb'' b''дb''b' ↩
    'ib''b''ab''b''мb''b''eb''b''тb''b''pb'' b''pb''b''ib''b''зb''b''цb''b''яb'')
M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''иb'')
```

Див. розділи G0 та M2 для отримання додаткової інформації.

Це помилка, якщо:

- Далі після G40 програмується дуговий рух G2/G3.
- Лінійний рух після вимкнення компенсації менший за діаметр інструменту.

11.5.26 G41, G42 Компенсація різця

```
G41 <D-> (b''лb''b''ib''b''вb''b''ob''b''pb''b''yb''b''чb'' b''вb''b''ib''b''дb'' b''зb''b' ↩
    'ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b''нb''b' ↩
    'ob''b''гb''b''ob'' b''шb''b''лb''b''яb''b''xb''b''yb'')
G42 <D-> (b''пb''b''pb''b''ab''b''вb''b''ob''b''pb''b''yb''b''чb'' b''вb''b''ib''b''дb'' b' ↩
    'зb''b''ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b' ↩
    'нb''b''ob''b''гb''b''ob'' b''шb''b''лb''b''яb''b''xb''b''yb'')
```

- *D* - номер інструменту

Слово D необов'язкове; якщо слово D відсутнє, буде використано радіус поточного завантаженого інструменту (якщо інструмент не завантажено і слово D не вказано, буде використано радіус 0).

Якщо вказано, то D-слово є номером інструменту, який слід використовувати. Зазвичай це номер інструменту в шпинделі (у цьому випадку D-слово є зайвим і вказувати його не потрібно), але це може бути будь-який дійсний номер інструменту.

Note
G41/G42 D0 є дещо особливим. Його поведінка відрізняється на верстатах з випадковою заміною інструменту та верстатах з невипадковою заміною інструменту (див. розділ Tool Change). На верстатах з невипадковою зміною інструментів «G41/G42 D0» застосовує зміщення довжини інструмента, який наразі знаходиться в шпинделі, або TLO 0, якщо в шпинделі немає інструмента. На верстатах з довільним перемиканням інструментів «G41/G42 D0» застосовує TLO інструменту T0, визначеного у файлі таблиці інструментів (або викликає помилку, якщо T0 не визначено в таблиці інструментів).

Щоб розпочати компенсацію різака зліва від профілю деталі, використовуйте G41. G41 розпочинає компенсацію різака зліва від запрограмованої лінії, якщо дивитися з позитивного кінця осі, перпендикулярної до площини.

Щоб розпочати компенсацію різака праворуч від профілю деталі, використовуйте G42. G42 розпочинає компенсацію різака праворуч від запрограмованої лінії, якщо дивитися з позитивного кінця осі, перпендикулярної до площини.

Рух підготовки має бути щонайменше таким же довгим, як радіус інструменту. Рух підготовки може бути швидким.

Компенсацію на різець можна виконати, якщо площина XY або площина XZ активна.

Команди користувача M100-M199 дозволені, коли ввімкнено компенсацію різця.

Поведінка обробного центру, коли ввімкнено корекцію різця, описана в розділі Cutter Compensation разом із прикладами коду.

Це помилка, якщо:

- Номер D не є дійсним номером інструменту або 0.
- Площина YZ активна.
- Компенсація різця увімкнена, якщо вона вже увімкнена.

11.5.27 G41.1, G42.1 Динамічна компенсація різця

```
G41.1 D- <L-> (b''лb''b''ib''b''вb''b''ob''b''pb''b''yb''b''чb'' b''вb''b''ib''b''дb'' b' ↩
    'зb''b''ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b' ↩
    'нb''b''ob''b''гb''b''ob'' b''шb''b''лb''b''яb''b''xb''b''yb'')
G42.1 D- <L-> (b''пb''b''pb''b''ab''b''вb''b''ob''b''pb''b''yb''b''чb'' b''вb''b''ib''b' ↩
    'дb'' b''зb''b''ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b' ↩
    'ab''b''нb''b''ob''b''гb''b''ob'' b''шb''b''лb''b''яb''b''xb''b''yb'')
```

- *D* - діаметр різця
- *L* - орієнтація інструменту (див. орієнтація токарного інструменту)

G41.1 та G42.1 функціонують так само, як G41 та G42, з додатковою можливістю програмування діаметра інструмента. Якщо значення слова L не вказано, воно за замовчуванням дорівнює 0.

Це помилка, якщо:

- Площина YZ активна.
- Число L не знаходиться в діапазоні від 0 до 9 включно.
- Число L використовується, коли площина XZ неактивна.
- Компенсація різця увімкнена, якщо вона вже увімкнена.

11.5.28 G43 Зміщення довжини інструменту

```
G43 <H->
```

- *H* - номер інструменту (необов'язково)
- «G43» — увімкнення компенсації довжини інструмента. G43 змінює наступні рухи, зміщуючи координати осі на величину зміщення. G43 не викликає жодних рухів. При наступному переміщенні компенсованої осі кінцевою точкою цієї осі є компенсоване положення.

G43 без слова H використовує поточний завантажений інструмент з останнього *T*n *M6*.

*G43 H*n використовує зміщення для інструменту *n*.

Активні значення компенсації довжини інструмента зберігаються в пронумерованих параметрах *5401-5409*.

Note
«G43 H0» є дещо особливим. Його поведінка відрізняється на верстатах із випадковим та невипадковим зміною інструментів (див. розділ «Tool Changers»). На верстатах з невипадковою зміною інструментів «G43 H0» застосовує зміщення довжини інструмента, який наразі знаходиться в шпинделі, або TLO 0, якщо в шпинделі немає інструмента. На верстатах з випадковою зміною інструментів «G43 H0» застосовує TLO інструмента T0, визначеного у файлі таблиці інструментів (або викликає помилку, якщо T0 не визначено в таблиці інструментів).

G43 H – Приклад рядка

```
G43 H1 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''зb''b' ↩
    'мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''ib''b''нb''b''cb''b''тb''b''pb''b' ↩
    'yb''b''мb''b''eb''b''нb''b''тb''b''ab'', b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b' ↩
    'cb''b''тb''b''ob''b''вb''b''yb''b''юb''b''чb''b''иb'' b''зb''b''нb''b''ab''b''чb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''зb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b' ↩
    'eb''b''нb''b''тb''b''ab'' 1 b''yb'' b''тb''b''ab''b''бb''b''лb''b''иb''b''цb''b''ib'' b ↩
    ''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''ib''b''вb'')
```

Це помилка, якщо:

- число H не є цілим числом, або
- число H від'ємне, або
- Номер H не є дійсним номером інструменту (хоча зауважте, що 0 є дійсним номером інструменту на верстатах із невипадковою зміною інструменту, це означає «інструмент, який наразі знаходиться в шпинделі».)

11.5.29 G43.1 Динамічне зміщення довжини інструменту

```
G43.1 axes
```

- «G43.1 осі» — зміна наступних рухів шляхом заміни поточного зміщення (зміщень) осей. G43.1 не викликає жодних рухів. Наступного разу, коли компенсована вісь буде переміщена, кінцевою точкою цієї осі буде компенсоване місце розташування.

Приклад G43.1

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''ab''b''бb''b ↩
    ''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b''иb''b' ↩
    'мb'')
T1 M6 G43 (b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb'' 1 b''ib'' b''зb''b' ↩
    'мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob''b''вb''b''жb''b''иb''b' ↩
    'нb''b''иb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b' ↩
    'yb'', Z b''зb''b''нb''b''ab''b''хb''b''ob''b''дb''b''иb''b''тb''b''ьb''b''cb''b''яb'' b ↩
    ''нb''b''ab'' b''мb''b''ab''b''шb''b''иb''b''нb''b''ib'' 0, b''ab'' DRO b''пb''b''ob''b' ↩
    'кb''b''ab''b''зb''b''yb''b''єb'' Z1.500)
G43.1 Z0.250 (b''зb''b''ab''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''пb''b''ob''b' ↩
    'тb''b''ob''b''чb''b''нb''b''eb'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b' ↩
    'яb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''yb'' b' ↩
    'нb''b''ab'' 0.250, DRO b''тb''b''eb''b''пb''b''eb''b''pb'' b''пb''b''ob''b''кb''b''ab'' ↩
    b''зb''b''yb''b''єb'' Z0.250)
M2 (b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''иb''b''тb''b''иb'' b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''yb'')
```

- Див. розділи G90 та T та M6 для отримання додаткової інформації.

Це помилка, якщо:

- Команда руху подається в тому ж рядку, що й *G43.1*

Note
G43.1 не записує в таблицю інструментів.

11.5.30 G43.2 Застосувати додаткове зміщення довжини інструменту

```
G43.2 H- or axes-
```

- *H* - номер інструменту

- *G43.2 H*n - застосовує додаткове одночасне зміщення інструменту до наступних рухів шляхом додавання зміщення(й) інструмента *n*.
- «Осі G43.2» – застосовує додаткове одночасне зміщення інструменту до наступних рухів шляхом додавання значення(й) будь-яких слів осей.

G43.2 Hn Приклад

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''ab''b''бb''b ↩
    ''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b''иb''b' ↩
    'мb'')
T1 M6 (b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb'' 1)
G43 (b''ab''b''бb''b''ob'' G43 H1 - b''зb''b''ab''b''мb''b''ib''b''нb''b''иb''b''тb''b' ↩
    'иb'' b''вb''b''cb''b''ib'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''ab'' b''зb''b' ↩
    'мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb''b''мb'' T1)
G43.2 H10 (b''тb''b''ab''b''кb''b''ob''b''жb'' b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b ↩
    ''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''ib''b''нb''b''cb''b''тb''b' ↩
    'pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''ab'' T10)
M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''иb'')
```

Приклад осей G43.2

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''ab''b''бb''b ↩
    ''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b''иb''b' ↩
    'мb'')
T1 M6 (b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''иb''b''тb''b''иb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb'' 1)
G43 (b''ab''b''бb''b''ob'' G43 H1 - b''зb''b''ab''b''мb''b''ib''b''нb''b''иb''b''тb''b' ↩
    'иb'' b''вb''b''cb''b''ib'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''yb'' b''нb''b' ↩
    'ab'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' T1)
G43.2 X0.01 Z0.02 (b''тb''b''ab''b''кb''b''ob''b''жb'' b''дb''b''ob''b''дb''b''ab''b''тb''b ↩
    ''иb'' 0,01 b''дb''b''ob'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b''нb''b''тb''b''yb'' b''пb''b' ↩
    'ob'' b''ob''b''cb''b''ib'' x b''ib'' 0,02 b''дb''b''ob'' b''зb''b''мb''b''ib''b''щb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''ib''b''нb''b''cb''b''тb''b''pb''b''yb''b''мb''b''eb''b' ↩
    'нb''b''тb''b''yb'' b''пb''b''ob'' b''ob''b''cb''b''ib'' z)
M2 (b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''иb''b''тb''b''иb'' b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''yb'')
```

Ви можете підсумувати довільну кількість зміщень, викликавши G43.2 кілька разів. Немає вбудованих припущень щодо того, які числа є зміщеннями геометрії, а які — зміщеннями зносу, або що ви повинні мати тільки одне з кожного типу.

Як і інші команди G43, G43.2 не викликає жодного руху. Наступного разу, коли компенсована вісь переміщується, кінцевою точкою цієї осі є скомпенсована точка розташування.

Це помилка, якщо:

- «H» не вказано, і зміщення осей не вказано.
- Вказано «H», а вказаний номер інструменту не існує в таблиці інструментів.
- Вказано «H» та також вказані осі.

Note
G43.2 не записує в таблицю інструментів.

11.5.31 G49 Скасувати компенсацію довжини інструменту

- *G49* - скасовує компенсацію довжини інструменту

Можна програмувати, використовуючи те саме зміщення, яке вже використовується. Також можна програмувати без зміщення довжини інструменту, якщо воно наразі не використовується.

11.5.32 Зміщення локальної системи координат G52

```
G52 axes
```

G52 використовується в програмі обробки деталі як тимчасове «зсунення локальної системи координат» в системі координат заготовки. Для отримання додаткової інформації про G92 та G52 і про те, як вони взаємодіють, див. Local and Global Offsets.

11.5.33 G53 Переміщення в координатах машини

```
G53 axes
```

Для переміщення в machine coordinate system, запрограмуйте *G53* в тому ж рядку, що і лінійне переміщення. *G53* не є модальним і повинен бути запрограмований в кожному рядку. *G0* або *G1* не потрібно програмувати в тому ж рядку, якщо один з них вже активний.

Наприклад, «G53 G0 X0 Y0 Z0» перемістить осі в початкове положення, навіть якщо поточна вибрана система координат має діючі зміщення.

Приклад G53

```
G53 G0 X0 Y0 Z0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''лb''b''ib''b''нb''b''ib''b' ↩
    'йb''b''нb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''дb''b''ob'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''уb'' b''кb''b' ↩
    'ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' b''вb''b''eb''b''pb''b''cb''b' ↩
    'тb''b''ab''b''тb''b''ab'')
G53 X2 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''лb''b''ib''b''нb''b''ib''b''йb''b' ↩
    'нb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b' ↩
    'яb'' b''дb''b''ob'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ob''b' ↩
    'їb'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb''b''иb'' X2)
```

Див. розділ G0 для отримання додаткової інформації.

Це помилка, якщо:

- G53 використовується без активних G0 або G1,
- або G53 використовується, коли ввімкнено корекцію різця.

11.5.34 G54-G59.3 Вибір системи координат

- *G54* - вибрати систему координат 1
- *G55* - вибрати систему координат 2
- *G56* - вибрати систему координат 3
- *G57* - вибрати систему координат 4
- *G58* - вибрати систему координат 5

- *G59* - вибрати систему координат 6
- *G59.1* - вибрати систему координат 7
- *G59.2* - вибрати систему координат 8
- *G59.3* - вибрати систему координат 9

Системи координат зберігають значення осей та кут повороту XY навколо осі Z у параметрах, наведених у наступній таблиці.

Table 11.16: Параметри системи координат

Вибрати	G5x	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

Це помилка, якщо:

- Вибір системи координат використовується, коли ввімкнено корекцію різця.

Див. розділ Coordinate System для огляду систем координат.

11.5.35 G61 Режим точного шляху

- *G61* – Режим точної траєкторії, рух точно за програмою. Рухи будуть сповільнюватися або зупинятися за потреби для досягнення кожної запрограмованої точки. Якщо два послідовні рухи точно колінеарні, рух не зупиниться.

11.5.36 G61.1 Режим точної зупинки

- *G61.1* - Режим точної зупинки, рух зупинятиметься в кінці кожного запрограмованого сегмента.

11.5.37 G64 Змішування контурів

```
G64 <P- <Q->>
```

- *P* - допуск змішування руху
- *Q* - наївна толерантність до кулачка
- *G64* - найкраща можлива швидкість. Без P (або значення за замовчуванням у RS274NGC) означає, що потрібно підтримувати найкращу можливу швидкість, незалежно від того, як далеко від запрограмованої точки ви опинитеся.

- *G64 P-* - Поєднання найкращої швидкості та допуску до відхилень
- *G64 P- <Q- >* змішування з допуском. Це спосіб точного налаштування системи для досягнення найкращого компромісу між швидкістю та точністю. Допуск P- означає, що фактичний шлях не буде відхилятися від запрограмованої кінцевої точки більше, ніж на P-. Швидкість буде зменшена, якщо це необхідно для збереження траєкторії. Якщо ви встановите Q на значення, відмінне від нуля, це увімкне «Наївний детектор CAM»: коли є серія лінійних рухів подачі XYZ з однаковою feed rate, які відхиляються від колінеарності менше, ніж Q-, вони об'єднуються в один лінійний рух. При рухах G2/G3 в площині G17 (XY), коли максимальне відхилення дуги від прямої лінії менше, ніж толеранс G64 P-, дуга розбивається на дві лінії (від початку дуги до середини і від середини до кінця). Ці лінії потім підлягають наївному алгоритму CAM для ліній. Таким чином, випадки лінії-дуги, дуги-дуги та дуги-лінії, а також лінії-лінії виграють від «наївного детектора CAM». Це покращує продуктивність контурування шляхом спрощення траєкторії. Можна програмувати для режиму, який вже активний. Дивіться також розділ Trajectory Control для отримання додаткової інформації про ці режими. Якщо Q не вказано, то воно буде поводитися так само, як і раніше, і використовувати значення P-. Встановіть Q на нуль, щоб вимкнути «Наївний детектор CAM».

Гарною ідеєю є включення специфікації керування шляхом у преамбулу кожного файлу G-коду.

G64 P- Q- Приклад рядка

```
G64 P0.015 Q0.015 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b ↩
    ''cb''b''лb''b''ib''b''дb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''зb''b''ab'' b' ↩
    'шb''b''лb''b''яb''b''хb''b''ob''b''мb'' b''уb'' b''мb''b''eb''b''жb''b''ab''b''хb''  ↩
    0.015 b''вb''b''ib''b''дb'' b''фb''b''ab''b''кb''b''тb''b''иb''b''чb''b''нb''b''ob''b' ↩
    'гb''b''ob'' b''шb''b''лb''b''яb''b''хb''b''уb'')
```

Значення P і Q вибираються невеликими. Зазвичай вони менші за точність верстата або точність типових деталей. Нижче наведено приклади з екстремальними значеннями P і Q для розуміння функції G64.

G64 Без значень

```
G64 (b''нb''b''ab''b''бb''b''ib''b''pb'' b''бb''b''eb''b''зb'' b''зb''b''нb''b''ab''b''чb'' ↩
    b''eb''b''нb''b''ьb'' P- b''тb''b''ab'' Q-)
```

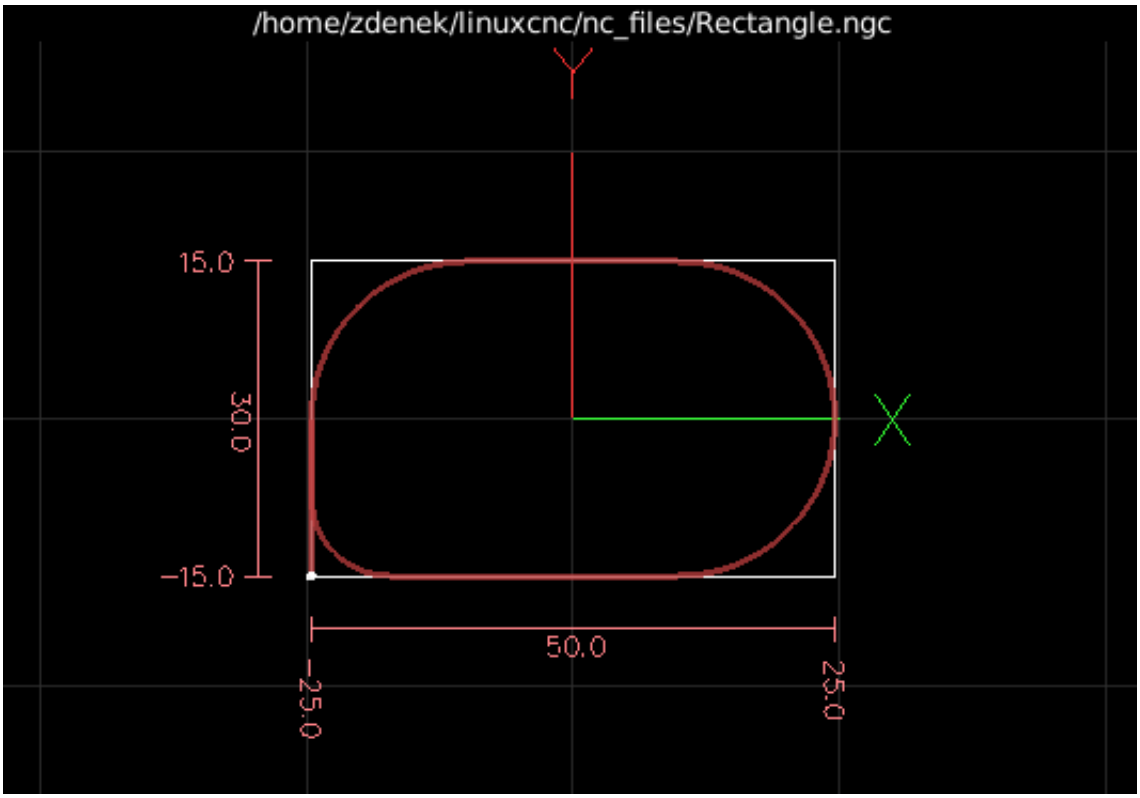


Figure 11.14: Прямокутник G64

G64 З великим значенням Q

```
G64 P0.015 Q6
```

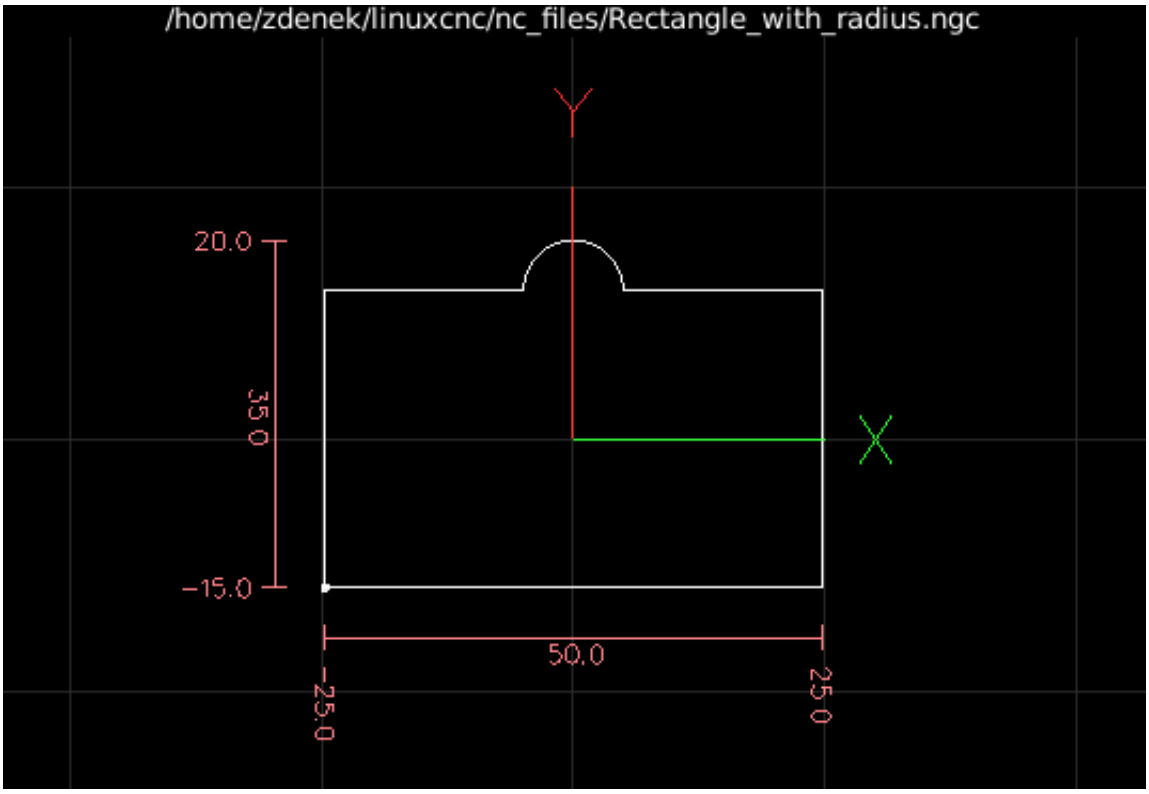


Figure 11.15: G64 Прямокутник з радіусом перед фрезеруванням

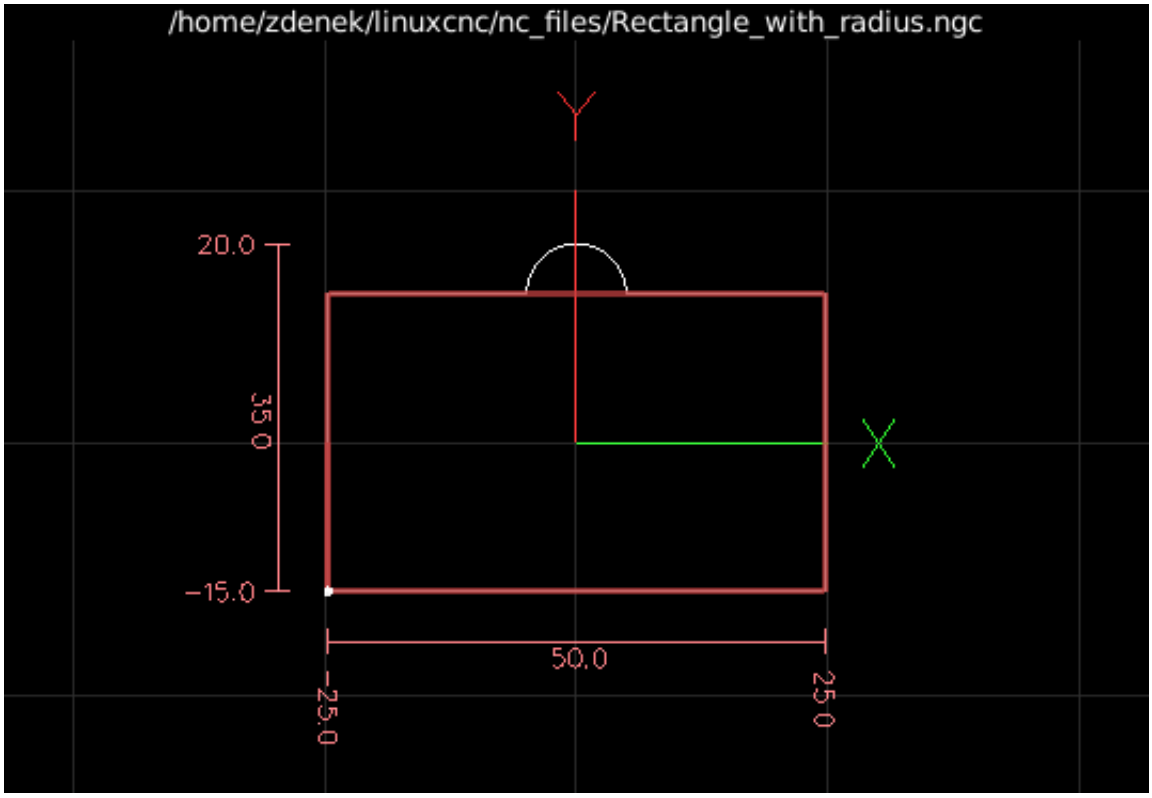


Figure 11.16: G64 Прямокутник з радіусом після фрезерування

```
G64 P0.015 Q2
```

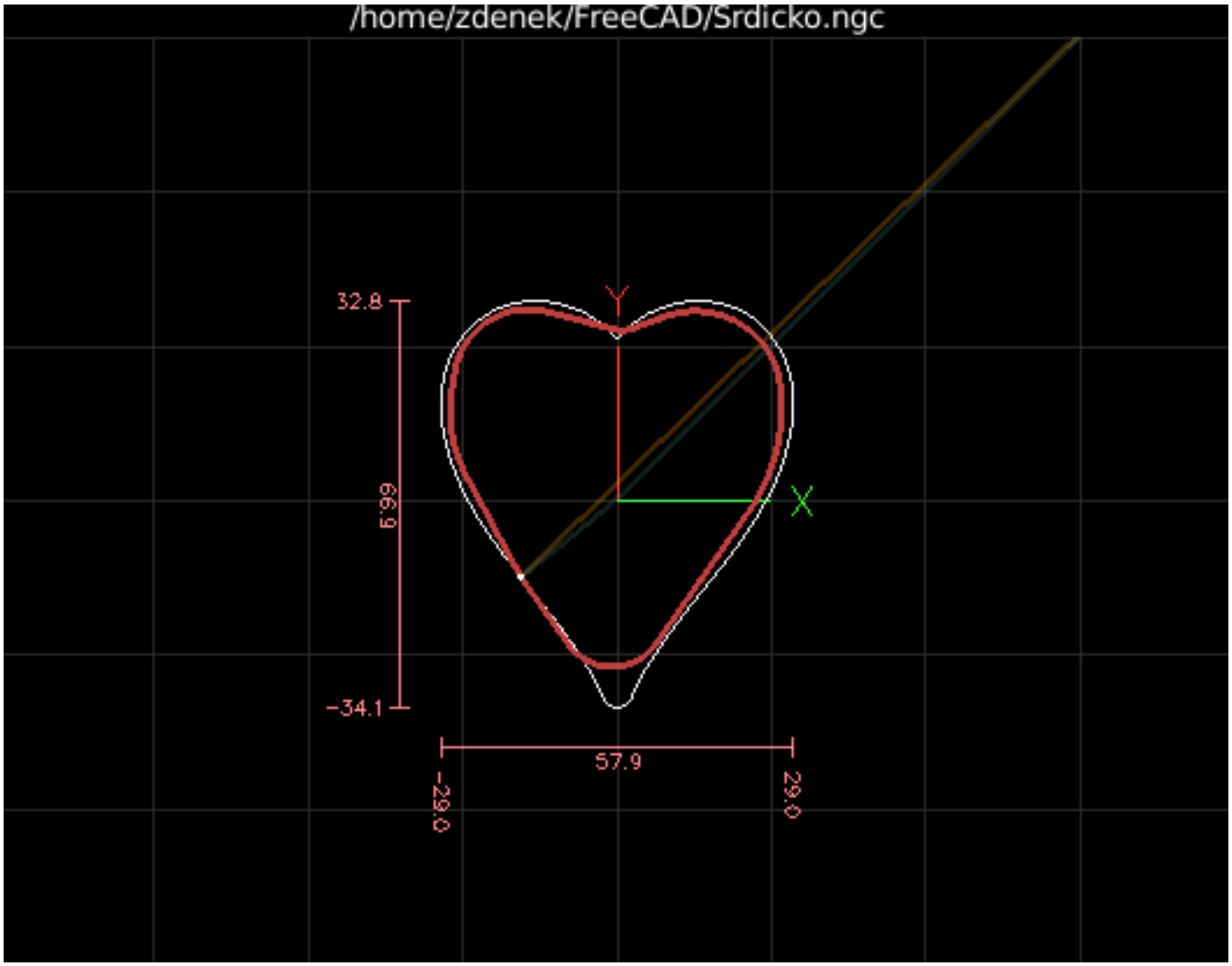


Figure 11.17: G64 Серце

11.5.38 Цикл чистової обробки токарного верстата G70

```
G70 Q- <X-> <Z-> <D-> <E-> <P->
```

- *Q* - Номер підпрограми.
- *X* - Початкова позиція X за замовчуванням дорівнює початковій позиції.
- *Z* - Початкова позиція Z за замовчуванням дорівнює початковій позиції.
- *D* - Початкова відстань профілю за замовчуванням дорівнює 0.
- *E* - Кінцева відстань профілю за замовчуванням дорівнює 0.
- *P* - Кількість перепусток для використання, за замовчуванням дорівнює 1.

Цикл *G70* призначений для використання після того, як форма профілю, задана в підпрограмі з номером Q, була вирізана за допомогою G71 або G72.

- Попередня клопотання.
  - Якщо використовуються Z або X, виконується команда rapid move до цієї позиції. Ця позиція також використовується між кожним чистовим проходом.
  - Потім виконується команда rapid move на початок профілю.
  - Шлях, заданий у Q-, виконується за допомогою команд G1 та Section 11.5.5.
  - Якщо потрібен наступний прохід, то є ще один поріг до проміжного місця, перш ніж зробити поріг до початку профілю.
  - Після останнього проходу інструмент залишається на кінці профілю, включаючи E-.
- Кілька проходів. Відстань між проходом та кінцевим профілем дорівнює (прохід-1)*(D-E)/P+E. Де прохід – номер проходу, а D, E та P – числа D/E/P.
- Відстань обчислюється з використанням початкової позиції циклу з додатною відстанню до цієї точки.
- Зігнення та фаски в профілі. У профілі можна додавати згини або фаски, див. Section 11.5.39 для отримання додаткової інформації.

Це помилка, якщо:

- Немає підпрограми, визначеної з номером, вказаним у Q.
- Шлях, заданий у профілі, не є монотонним у Z або X.
- Section 11.5.18 не використовувався для вибору площини ZX.

11.5.39 G71 G72 Цикли чорнової обробки токарного верстата

Note
Цикли G71 та G72 наразі дещо нестабільні. Див., наприклад, проблему #2939.

```
G71   Q- <X-> <Z-> <D-> <I-> <R->
G71.1 Q- <X-> <Z-> <D-> <I-> <R->
G71.2 Q- <X-> <Z-> <D-> <I-> <R->
G72   Q- <X-> <Z-> <D-> <I-> <R->
G72.1 Q- <X-> <Z-> <D-> <I-> <R->
G72.2 Q- <X-> <Z-> <D-> <I-> <R->
```

- *Q* - Номер підпрограми.
- *X* - Початкова позиція X за замовчуванням дорівнює початковій позиції.
- *Z* - Початкова позиція Z за замовчуванням дорівнює початковій позиції.
- *D* - Відстань, що залишилася до профілю, за замовчуванням дорівнює 0.
- *I* - Приріст різання за замовчуванням дорівнює 1.
- *R* - Відстань відведення за замовчуванням становить 0,5.

Цикл G71/G72 призначений для чорнової обробки профілю на токарному верстаті. Цикли G71 видаляють шари матеріалу під час переміщення у напрямку Z. Цикли G72 видаляють матеріал під час переміщення по осі X, так званий цикл торцювання. Напрямок руху такий самий, як і в траєкторії, заданій у підпрограмі. Для циклу G71 координата Z повинна змінюватися монотонно, для G72 це потрібно для осі X.

Профіль задається в підпрограмі з номером Q-. Ця підпрограма може містити команди руху G0, G1, G2 і G3. Всі інші команди ігноруються, включаючи налаштування подачі і швидкості. Команди Section 11.5.3 інтерпретуються як команди G1. Кожна команда руху може також містити опціональне число A- або C-. Якщо додано число A-, то в кінцевій точці цього руху буде вставлено заокруглення з радіусом, заданим A. Якщо цей радіус занадто великий, алгоритм завершиться з помилкою немонотонного шляху. Також можна використовувати число C-, яке дозволяє вставити фаску. Ця фаска має ті самі кінцеві точки, що й заокруглення того самого розміру, але замість дуги вставляється пряма лінія.

В абсолютному режимі U (для X) та W (для Z) можна використовувати як приростні переміщення.

Цикли G7x.1 не вирізають кишені. Цикли G7x.2 вирізають тільки після першої кишені і продовжують там, де зупинився G7x.1. Рекомендується залишити трохи додаткового матеріалу для вирізання перед циклом G7x.2, тому якщо G7x.1 використовував D1.0, G7x.2 може використовувати D0.5, і 0,5 мм буде видалено під час переходу від однієї кишені до наступної.

Звичайні цикли G7x вирізають весь профіль за один цикл.

1. Попередня клопотання.
   - Якщо використовуються Z або X, виконується rapid move до цієї позиції.
   - Після вирізання профілю інструмент зупиняється в кінці профілю, включаючи відстань, зазначену в D.
2. Число D використовується для дотримання відстані від остаточного профілю, щоб залишився матеріал для обробки.

Це помилка, якщо:

- Немає підпрограми, визначеної з номером, вказаним у Q.
- Шлях, заданий у профілі, не є монотонним у Z або X.
- Section 11.5.18 не використовувався для вибору площини ZX.
- Section 11.5.26 є активним.

11.5.40 Цикл свердління G73 зі стружколомленням

```
G73 X- Y- Z- R- Q- D- <L->
```

- *R* - положення відведення вздовж осі Z.
- *Q* - приріст дельти вздовж осі Z.
- *L* - повторити

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.

- Попередня клопотання.

  - Якщо поточна позиція Z знаходиться нижче позиції R, вісь Z виконує rapid move до позиції R.
  - Перемістити до координат X Y
- Переміщувати вісь Z лише за поточною точкою feed rate вниз на дельту або до позиції Z, залежно від того, яка з них менш глибока.
- Rapid up either the D value, the G73_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Повторюйте кроки 2 та 3, доки не буде досягнуто положення Z на кроці 2.
- Вісь Z швидко переміщується в положення R.

Це помилка, якщо:

- число Q від’ємне або дорівнює нулю.
- номер R не вказано

11.5.41 G74 Цикл нарізання різьби ліворуч із затримкою

```
G74 (X- Y- Z-) or (U- V- W-) R- L- P- $- F-
```

- *R-* - Відведення положення вздовж осі Z.
- *L-* – Використовується в інкрементальному режимі; кількість разів для повторення циклу. Див. приклади G81.
- *P-* - Час витримки (секунди).
- *$-* - Вибраний шпиндель.
- *F-* - Швидкість подачі (швидкість обертання шпинделя, помножена на відстань, пройдену за оберт (крок різьби)).



Warning
G74 не використовує синхронізований рух.

Цикл *G74* призначений для нарізання різьби з плаваючим патроном та затримкою на дні отвору.

1. Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
2. Вимкнути коригування подачі та швидкості.
3. Перемістіть вісь Z з поточною швидкістю подачі в позицію Z.
4. Зупинити вибраний шпиндель (вибраний параметром $)
5. Почніть обертання шпинделя за годинниковою стрілкою.
6. Затримайтеся на певну кількість секунд P.
7. Перемістіть вісь Z з поточною швидкістю подачі, щоб очистити Z
8. Відновлення подачі та швидкості дозволяє перевизначення до попереднього стану

Тривалість витримки визначається словом «P-» у блоці G74. Швидкість подачі «F-» дорівнює швидкості обертання шпинделя, помноженій на відстань за один оберт (крок різьби). У прикладі S100 з кроком різьби 1,25 мм за один оберт швидкість подачі становить F125.

11.5.42 Цикл нарізання різьби G76

```
G76 P- Z- I- J- R- K- Q- H- E- L- $-
```

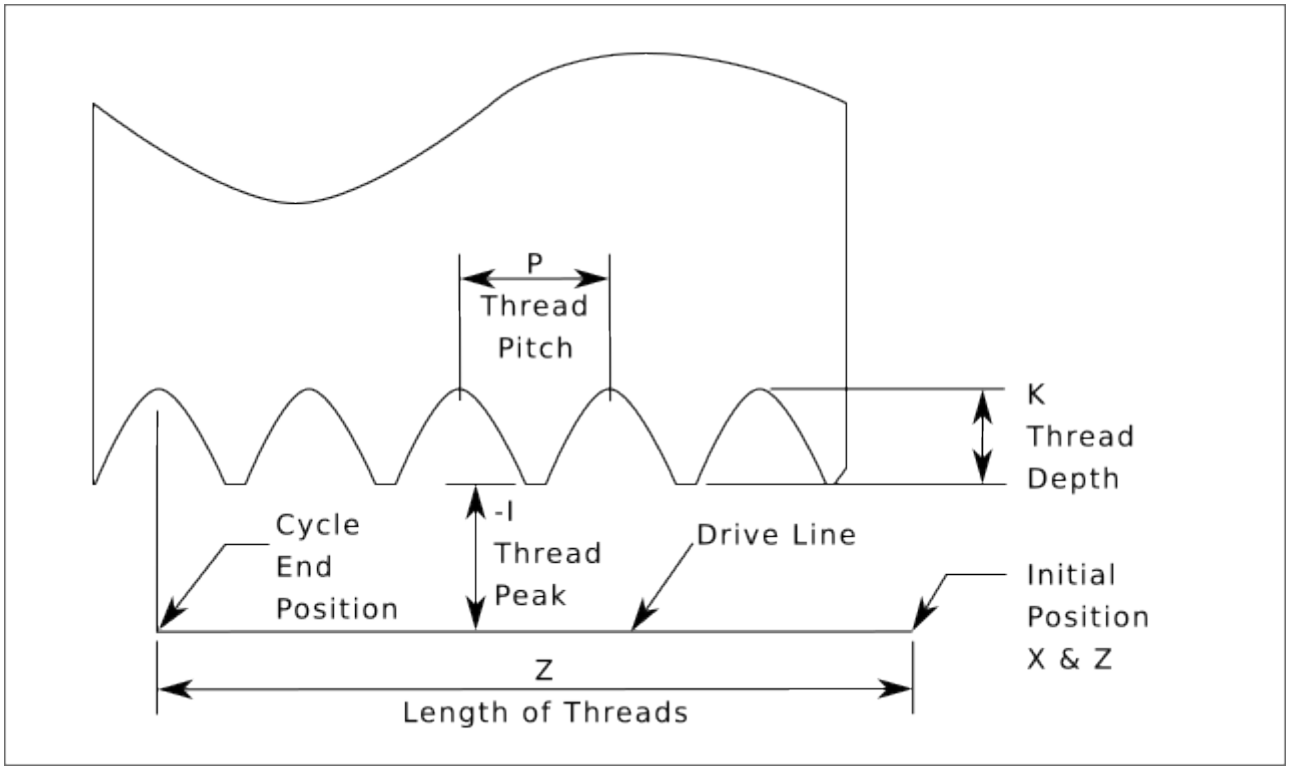


Figure 11.18: G76 Різьбонарізання

- «Лінія руху» – лінія, що проходить через початкове положення X, паралельна Z.
- «P-» – «крок різьби» на відстань за оберт.
- *Z-* – Кінцеве положення різьби. В кінці циклу інструмент буде в цьому положенні Z.

Note
Коли діє G7 «Режим діаметра токарного верстата», значення «I», «J» та «K» є вимірюваннями діаметра. Коли діє G8 «Режим радіуса токарного верстата», значення «I», «J» та «K» є вимірюваннями радіуса.

- «I-» — зміщення «піку різьби» від «лінії приводу». Від’ємні значення «I» відповідають зовнішній різьбі, а додатні значення «I» — внутрішній різьбі. Зазвичай матеріал обточується до цього розміру перед циклом «G76».
- «J-» – додатне значення, що визначає «початкову глибину різання». Перший різьбовий розріз буде «J» за позицією «вершини різьби».
- «K-» – додатне значення, що вказує на «повну глибину різьби». Остаточний розріз різьби буде на «K» далі за положення «вершини різьби».

Додаткові налаштування

- *$*- - номер шпинделя, з яким буде синхронізовано рух (за замовчуванням 0). Наприклад, якщо запрограмовано $1, рух розпочнеться після скидання `spindle.1.index-enable` і буде продовжуватися синхронно із значенням `spindle.1.revs`.
- «R-» — «зменшення глибини». «R1.0» вибирає постійну глибину для послідовних проходів різьблення. «R2.0» вибирає постійну площу. Значення від 1,0 до 2,0 вибирають зменшення глибини, але збільшення площі. Значення вище 2,0 вибирають зменшення площі. Зверніть увагу, що надмірно високі значення зменшення глибини призведуть до використання великої кількості проходів. (Зменшення глибини = спуск по етапах або сходинках.)



Warning
Непотрібно високі значення дегресії призведуть до надмірно великої кількості проходів (дегресія = поетапне занурення)

- «Q-» — «кут складеного зсуву» — це кут (у градусах), що описує, наскільки послідовні проходи повинні бути зміщені вздовж лінії руху. Це використовується для того, щоб одна сторона інструменту видаляла більше матеріалу, ніж інша. Позитивне значення «Q» призводить до того, що передній край інструменту ріже сильніше. Типові значення — 29, 29,5 або 30.
- «H-» – кількість «пружинних проходів». Пружинні проходи – це додаткові проходи на повну глибину різьби. Якщо додаткові проходи не потрібні, запрограмуйте «H0».

Вхід та вихід різьби можна запрограмувати конічними зі значеннями *E* та *L*.

- «E-» — вказує відстань уздовж лінії приводу, яка використовується для конусності. Кут конусності буде таким, що останній прохід звужується до вершини різьби на відстані, вказаній за допомогою E. «E0.2» дасть конусність для перших/останніх 0,2 одиниць довжини уздовж різьби. Для конусності 45 градусів програма E така сама, як і K.
- «L-» — вказує, на яких кінцях різьби буде зроблено конус. Програма «L0» для відсутності конуса (за замовчуванням), «L1» для вхідного конуса, «L2» для вихідного конуса або «L3» для вхідного та вихідного конусів. Конусність на вході зупиниться на лінії приводу для синхронізації з імпульсом індексу, а потім переміститься зі швидкістю feed rate до початку конусності. Без конусності на вході інструмент швидко переміститься до глибини різання, а потім синхронізується і почне різання.

Інструмент переміщується у початкові положення X та Z перед видачею G76. Положення X – це «лінія приводу», а положення Z – початок різьби.

Інструмент буде коротко зупинятися для синхронізації перед кожним проходом різьблення, тому на вході буде потрібна рельєфна канавка, якщо початок різьблення не знаходиться за кінцем матеріалу або не використовується конусний вхід.

Якщо не використовується конус виходу, рух виходу не синхронізується зі швидкістю шпинделя і буде rapid move. При низькій швидкості шпинделя рух виходу може зайняти лише невелику частину оберту. Якщо швидкість шпинделя збільшується після завершення декількох проходів, наступні рухи виходу вимагатимуть більшої частини оберту, що призведе до дуже важкого різання під час руху виходу. Цього можна уникнути, зробивши на виході рельєфну канавку або не змінюючи швидкість шпинделя під час нарізування різьби.

Остаточне положення інструменту буде на кінці «лінії приводу». Для вилучення інструменту з отвору знадобиться безпечний рух по осі Z із внутрішньою різьбою.

Це помилка, якщо:

- Активна площина не є площиною ZX.

- Інші слова осей, такі як X- або Y-, вказуються.
- Значення дегресії *R*- менше ніж 1,0.
- Не всі необхідні слова вказані.
- «P-», «J-», «K-» або «H-» – негативні.
- «E-» більше половини довжини приводного тракту.

З'єднання HAL Щоб G76 працював, у файлі HAL необхідно підключити контакти «spindle.N.at-speed» та «encoder.*n*.phase-Z» для шпинделя. Докладнішу інформацію див. у розділі «spindle» розділу «Рух».

Технічна інформація Стандартний цикл G76 базується на синхронізованому русі шпинделя G33. Для отримання додаткової інформації див. G33 Технічна інформація.

Приклад програми «g76.ngc» демонструє використання стандартного циклу G76, його можна переглянути та виконати на будь-якому верстаті за допомогою конфігурації «sim/lathe.ini».

Приклад коду G76

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

На малюнку інструмент знаходиться в кінцевому положенні після завершення циклу G76. Ви можете побачити шлях входу праворуч від Q29.5 і шлях виходу ліворуч від L2 E0.045. Білі лінії - це рухи різання.

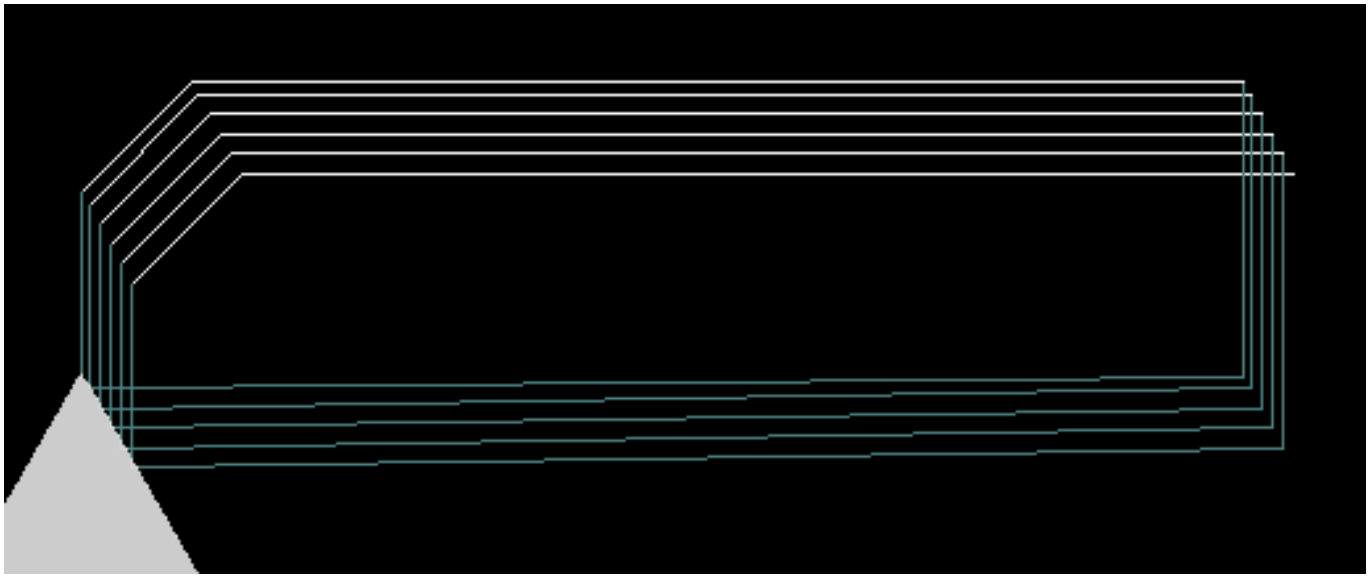


Figure 11.19: Приклад G76

11.5.43 Стандартні цикли G80-G89

У цьому розділі описано стандартні цикли від «G81» до «G89» та зупинку стандартного циклу «G80».

Всі цикли консервування виконуються відносно поточної вибраної площини. Можна вибрати будь-яку з дев'яти площин. У цьому розділі в більшості описів передбачається, що вибрано площину XY. Поведінка аналогічна, якщо вибрано іншу площину, і необхідно використовувати

правильні слова. Наприклад, у площині «G17.1» дія стандартного циклу відбувається вздовж W, а розташування або приріст задаються за допомогою U і V. У цьому випадку замість X, Y, Z у наведених нижче інструкціях слід використовувати U, V, W.

Слова осі обертання не допускаються в стандартних циклах. Коли активна площина належить до сімейства XYZ, слова осі UVW не допускаються. Аналогічно, коли активна площина належить до сімейства UVW, слова осі XYZ не допускаються.

11.5.43.1 Загальні слова

Усі стандартні цикли використовують групи X, Y, Z або U, V, W залежно від вибраної площини та слів R. Позиція R (зазвичай означає відведення) знаходиться вздовж осі, перпендикулярної до поточної вибраної площини (ось Z для площини XY тощо). Деякі стандартні цикли використовують додаткові аргументи.

11.5.43.2 Липкі слова

Для вбудованих циклів ми будемо називати число «фіксованим», якщо при використанні одного і того ж циклу в декількох рядках коду поспіль це число має бути використане в першому випадку, але є необов’язковим в інших рядках. Фіксовані числа зберігають своє значення в інших рядках, якщо вони не запрограмовані явно як інші. Число R завжди є фіксованим.

У режимі інкрементальної відстані числа X, Y і R розглядаються як інкременти від поточного положення, а Z — як інкремент від положення осі Z до початку переміщення, що включає Z. У режимі абсолютної відстані числа X, Y, R і Z є абсолютними положеннями в поточній системі координат.

11.5.43.3 Повторення циклу

Число L є необов’язковим і позначає кількість повторень. L=0 не допускається. Якщо використовуєтьс функція повторення, вона зазвичай застосовується в режимі інкрементальної відстані, так що одна і та ж послідовність рухів повторюється в декількох рівновіддалених місцях вздовж прямої лінії. Коли L- більше 1 в інкрементальному режимі з вибраною площиною XY, позиції X і Y визначаються шляхом додавання заданих чисел X і Y або до поточних позицій X і Y (при першому проході), або до позицій X і Y в кінці попереднього проходу (при повтореннях). Таким чином, якщо ви запрограмуєте «L10», ви отримаєте 10 циклів. Перший цикл буде відстанню X,Y від початкового місця розташування. Позиції R і Z не змінюються під час повторень. Число L не є фіксованим. У режимі абсолютної відстані L>1 означає «виконати один і той самий цикл у тому самому місці кілька разів». Опущення слова L еквівалентно заданню L=1.

11.5.43.4 Режим втягування

Висота руху відведення в кінці кожного повторення (названого «clear Z» в описах нижче) визначається налаштуванням режиму відведення, або до початкового положення Z (якщо воно знаходиться вище положення R і режим відведення є «G98», OLD_Z), або до положення R. Дивіться розділ G98 G99.

11.5.43.5 Помилки стандартного циклу

Це помилка, якщо:

- всі слова осі відсутні під час стандартного циклу,

- слова осі з різних груп (XYZ) (UVW) використовуються разом,
- потрібне число P, і використовується від'ємне число P,
- використовується число L, яке не є додатним цілим числом,
- рух обертової осі використовується під час стандартного циклу,
- обернена швидкість подачі активна під час стандартного циклу,
- або компенсація різця активна під час стандартного циклу.

Якщо площина XY активна, число Z є закріпленим, і це помилка, якщо:

- Номер Z відсутній, і той самий стандартний цикл ще не був активним,
- або число R менше за число Z.

Якщо інші площини активні, умови помилки аналогічні умовам XY, описаним вище.

11.5.43.6 Попереднє та проміжне клопотання

Попередній рух — це набір рухів, який є спільним для всіх циклів фрезерування. Якщо поточне положення Z знаходиться нижче положення R, вісь Z виконує rapid move до положення R. Це відбувається тільки один раз, незалежно від значення L.

Крім того, на початку першого циклу та кожного повторення виконуються такі один або два рухи:

- Швидкий рух rapid move паралельно площині XY до заданої позиції XY.
- Вісь Z швидко переміститься в положення R, якщо вона ще не знаходиться в цьому положенні.

Якщо активна інша площина, попередній та проміжний рухи аналогічні.

11.5.43.7 Навіщо використовувати фіксований цикл?

Існує щонайменше дві причини використання стандартних циклів. Перша — це економія коду. Для виконання одного отвору знадобиться кілька рядків коду.

G81 Приклад 1 демонструє, як можна використовувати готовий цикл для створення 8 отворів за допомогою десяти рядків G-коду в режимі готового циклу. Програма, наведена нижче, створить той самий набір з 8 отворів, використовуючи п'ять рядків для готового циклу. Вона не слідує точно тому самому шляху і не свердлить у тому самому порядку, що й попередній приклад. Але економічність написання програми для хорошого готового циклу повинна бути очевидною.

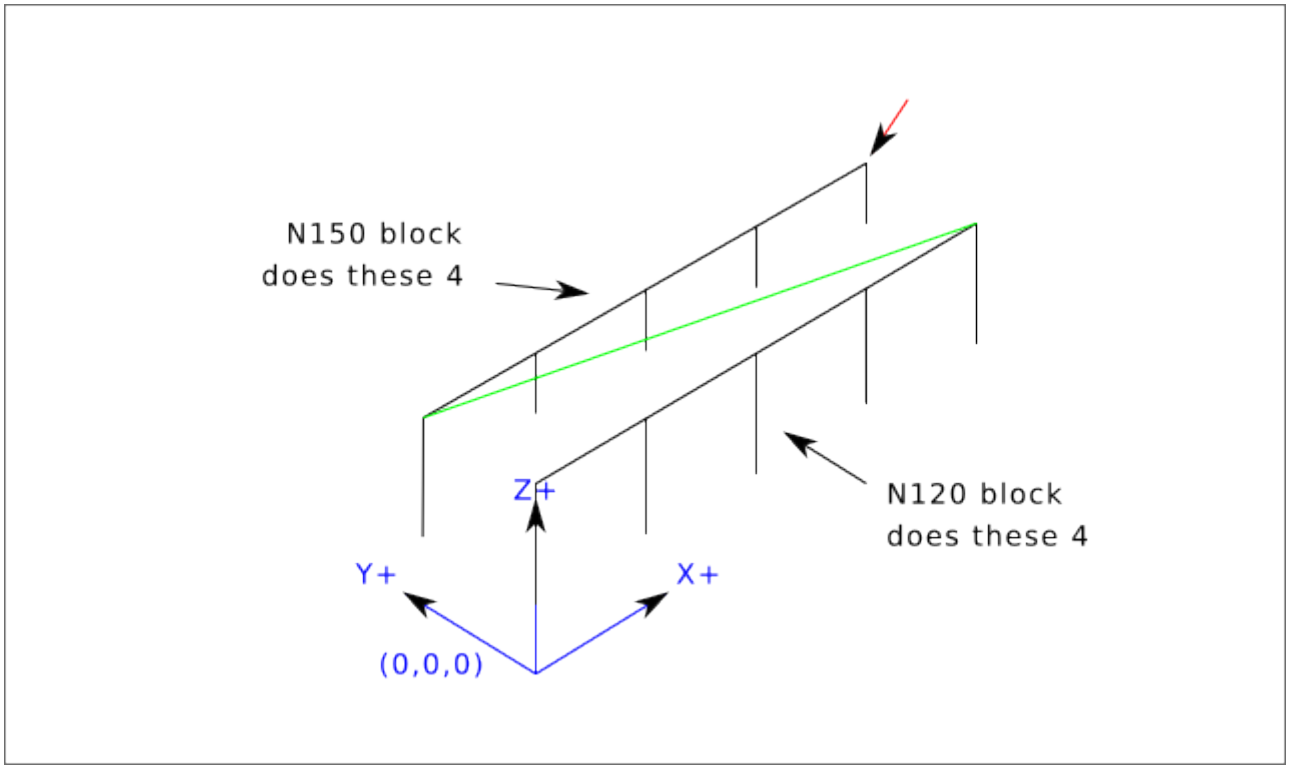
Note
Номери рядків не потрібні, але вони допомагають пояснити ці приклади.

Вісім лунок

```
N100 G90 G0 X0 Y0 Z0 (b''пb''b''eb''b''pb''b''eb''b''мb''b''іb''b''щb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''аb''b''тb'' b''дb''b' ↩
    'ob'' b''вb''b''иb''b''хb''b''іb''b''дb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b' ↩
    'лb''b''ob''b''жb''b''eb''b''нb''b''нb''b''яb'')
N110 G1 F10 X0 G4 P0.1
```

```
N120 G91 G81 X1 Y0 Z-1 R1 L4 (b''цb''b''иb''b''кb''b''лb'' b''cb''b''вb''b''eb''b''pb''b' ↩
    'дb''b''лb''b''ib''b''нb''b''нb''b''яb'')
N130 G90 G0 X0 Y1
N140 Z0
N150 G91 G81 X1 Y0 Z-0.5 R1 L4 (b''зb''b''ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb'' b''цb''b''иb''b''кb''b''лb'' b''cb''b' ↩
    'вb''b''eb''b''pb''b''дb''b''лb''b''ib''b''нb''b''нb''b''яb'')
N160 G80 (b''вb''b''иb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''зb''b''ab''b''пb''b' ↩
    'pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb'' b' ↩
    'цb''b''иb''b''кb''b''лb'')
N170 M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''иb'')
```

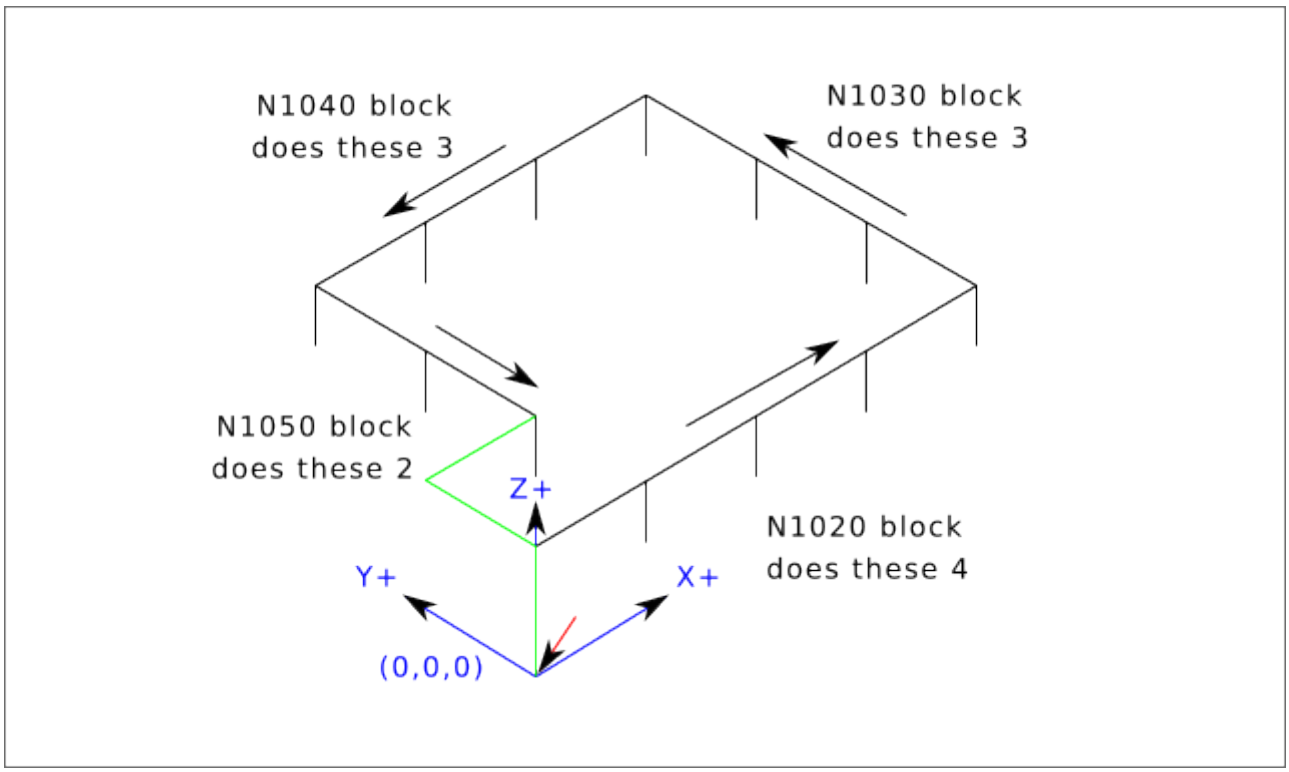
Код G98 у другому рядку вище означає, що зворотний рух буде до значення Z у першому рядку, оскільки воно вище за задане значення R.



Дванадцять отворів у квадраті Цей приклад демонструє використання слова L для повторення набору інкрементальних циклів свердління для послідовних блоків коду в одному режимі руху G81. Тут ми створюємо 12 отворів за допомогою п’яти рядків коду в режимі заданого руху.

```
N1000 G90 G0 X0 Y0 Z0 (b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''щb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'' b''дb''b' ↩
    'ob'' b''вb''b''иb''b''xb''b''ib''b''дb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b' ↩
    'лb''b''ob''b''жb''b''eb''b''нb''b''нb''b''яb'')
N1010 G1 F50 X0 G4 P0.1
N1020 G91 G81 X1 Y0 Z-0.5 R1 L4 (b''цb''b''иb''b''кb''b''лb'' b''cb''b''вb''b''eb''b''pb''b ↩
    ''дb''b''лb''b''ib''b''нb''b''нb''b''яb'')
N1030 X0 Y1 R0 L3 (b''пb''b''ob''b''вb''b''тb''b''ob''b''pb''b''иb''b''тb''b''иb'')
N1040 X-1 Y0 L3 (b''пb''b''ob''b''вb''b''тb''b''ob''b''pb''b''иb''b''тb''b''иb'')
N1050 X0 Y-1 L2 (b''пb''b''ob''b''вb''b''тb''b''ob''b''pb''b''иb''b''тb''b''иb'')
N1060 G80 (b''вb''b''иb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''цb''b''иb''b''кb''b' ↩
    'лb'')
N1070 G90 G0 X0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''pb''b''уb''b''xb'' b' ↩
    'дb''b''ob''b''дb''b''ob''b''мb''b''уb'')
```

```
N1080 Y0
N1090 Z0
N1100 M2 (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''иb'')
```



Друга причина використання стандартного циклу полягає в тому, що всі вони створюють попередні рухи та повернення, які можна передбачити та контролювати незалежно від початкової точки стандартного циклу.

11.5.44 G80 Скасувати стандартний цикл

- *G80* - Скасувати модальний рух стандартного циклу. *G80* є частиною модальної групи 1, тому програмування будь-якого іншого G-коду з модальної групи 1 також скасує стандартний цикл.

Це помилка, якщо:

- Слова осей програмуються, коли активний G80.

Приклад G80

```
G90 G81 X1 Y1 Z1.5 R2.8 (b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b''нb''b' ↩
    'иb''b''йb'' b''цb''b''иb''b''кb''b''лb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b' ↩
    'тb''b''нb''b''ob''b''їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G80 (b''вb''b''иb''b''мb''b''кb''b''нb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ob''b''cb''b ↩
    ''тb''b''ib''b''йb''b''нb''b''ob''b''гb''b''ob'' b''цb''b''иb''b''кb''b''лb''b''уb'' b' ↩
    'pb''b''уb''b''хb''b''уb'')
G0 X0 Y0 Z0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb'' ↩
    b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' b''пb''b''ob''b''чb''b''ab''b' ↩
    'тb''b''кb''b''ob''b''вb''b''ob''b''їb'' b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b' ↩
    'нb''b''ab''b''тb''b''иb'')
```

Наступний код створює таку ж кінцеву позицію та стан машини, як і попередній код.

Приклад G0

```
G90 G81 X1 Y1 Z1.5 R2.8 (b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b''нb''b' ↩
    'иb''b''йb'' b''цb''b''иb''b''кb''b''лb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b' ↩
    'тb''b''нb''b''ob''b''їb'' b''вb''b''іb''b''дb''b''cb''b''тb''b''ab''b''нb''b''іb'')
G0 X0 Y0 Z0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb'' ↩
    b''іb''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' b''пb''b''ob''b''чb''b''ab''b' ↩
    'тb''b''кb''b''ob''b''вb''b''ob''b''їb'' b''тb''b''ob''b''чb''b''кb''b''иb'' b''кb''b' ↩
    'ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb'')
```

Перевага першого набору полягає в тому, що рядок G80 чітко вимикає вбудований цикл G81. З першим набором блоків програміст повинен знову увімкнути рух за допомогою G0, як це зроблено в наступному рядку, або будь-яким іншим словом G режиму руху.

Якщо цикл не вимкнено за допомогою G80 або іншого слова руху, цикл спробує повторити себе, використовуючи наступний блок коду, що містить слово X, Y або Z. Наступний файл свердлить (G81) набір з восьми отворів, як показано в наступному підписі.

Приклад G80 1

```
N100 G90 G0 X0 Y0 Z0 (b''кb''b''ob''b''ob''b''pb''b''дb''b''иb''b''нb''b''ab''b''тb''b' ↩
    'нb''b''ab'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ab'' b''тb''b' ↩
    'ob''b''чb''b''кb''b''ab'')
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (b''зb''b''ab''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b' ↩
    'ob''b''вb''b''ab''b''нb''b''иb''b''йb'' b''цb''b''иb''b''кb''b''лb'' b''cb''b''вb''b' ↩
    'eb''b''pb''b''дb''b''лb''b''іb''b''нb''b''нb''b''яb'')
N130 X2
N140 X3
N150 X4
N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (b''вb''b''иb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''цb''b''иb''b''кb''b' ↩
    'лb'')
N210 G0 X0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''pb''b''уb''b''хb'' b''дb'' ↩
    b''ob'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''ob''b''вb''b''ob''b''їb'' b''пb''b' ↩
    'ob''b''зb''b''иb''b''цb''b''іb''b''їb'')
N220 Y0
N230 Z0
N240 M2 (b''кb''b''іb''b''нb''b''eb''b''цb''b''ьb'' b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''иb'')
```

Note
Зверніть увагу на зміну позиції Z після перших чотирьох отворів. Також це одне з небагатьох місць, де номери рядків мають певне значення, оскільки вони можуть вказати читачеві на певний рядок коду.

Використання G80 у рядку N200 є необов’язковим, оскільки G0 у наступному рядку вимкне цикл G81. Але використання G80, як показано в прикладі 1, забезпечить легше читання стандартного циклу. Без нього не так очевидно, що всі блоки між N120 і N200 належать до стандартного циклу.

11.5.45 Цикл свердління G81

```
G81 (X- Y- Z-) b''ab''b''бb''b''ob'' (U- V- W-) R- L-
```

Цикл «G81» призначений для свердління.

Цикл функціонує наступним чином:

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістіть вісь Z у поточній позиції feed rate у позицію Z.
- Вісь Z виконує rapid move для очищення осі Z.

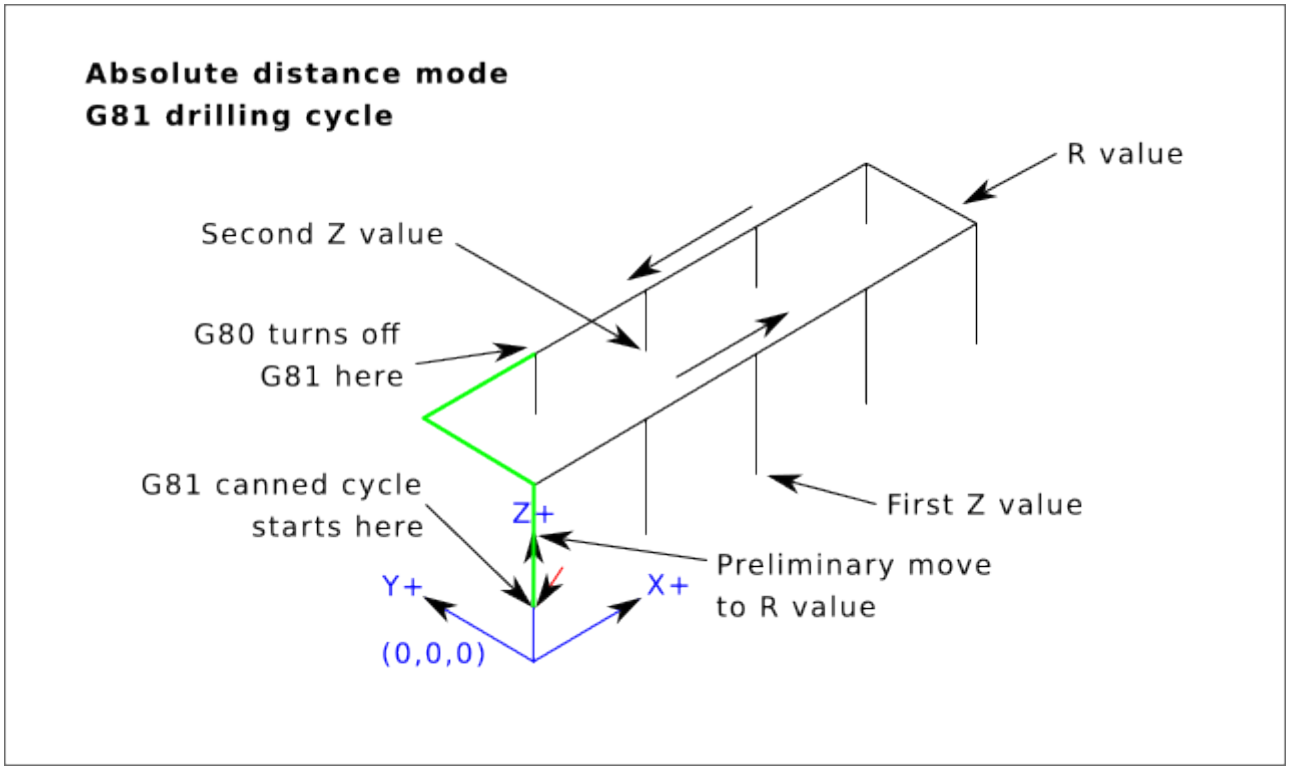


Figure 11.20: Цикл G81

Приклад 1 - Абсолютне положення G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

Припустимо, що поточна позиція (X1, Y2, Z3) і інтерпретується попередній рядок коду NC.

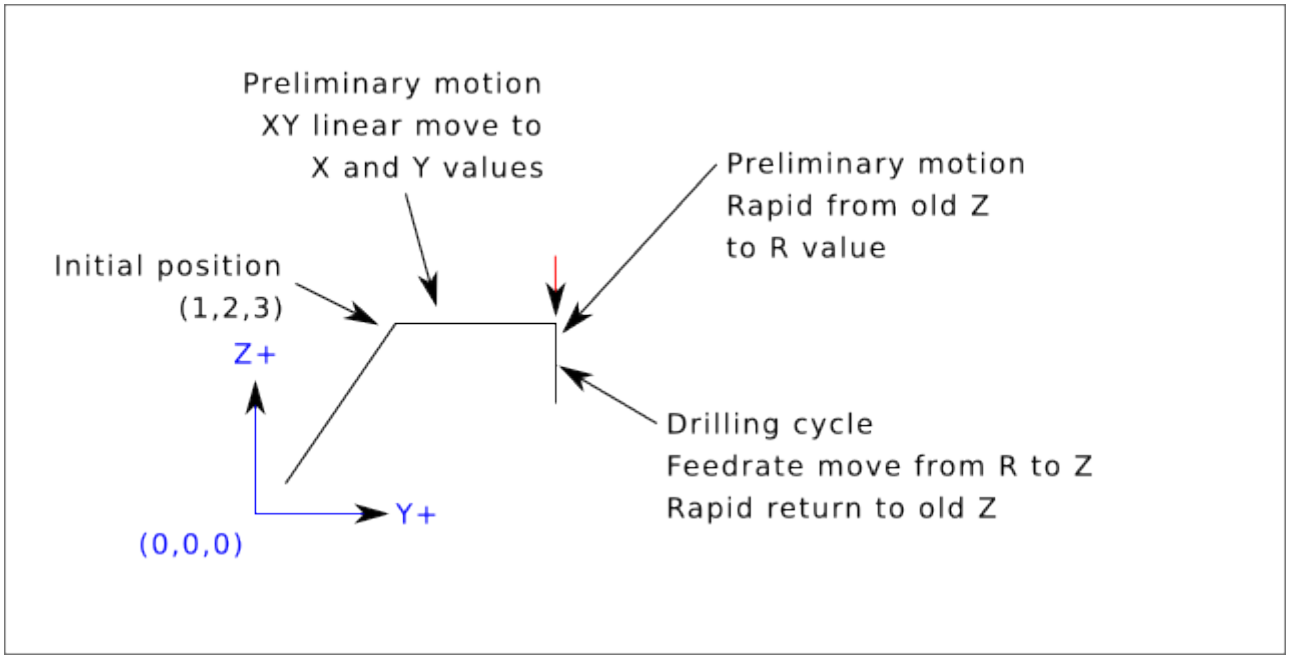
Це вимагає режиму абсолютної відстані (G90) та режиму відведення OLD_Z (G98), а також одноразового виконання циклу свердління G81.

- Значення X та позиція X дорівнюють 4.
- Значення Y та положення Y дорівнюють 5.
- Значення Z та положення Z дорівнюють 1,5.
- Значення R та чистий Z дорівнюють 2,8. OLD_Z дорівнює 3.

Відбуваються такі рухи:

- A rapid move паралельно площині XY до (X4, Y5)

- Швидкий рух паралельно осі Z до (Z2.8).
- Переміщення паралельно осі Z у точці feed rate до (Z1.5)
- Швидкий рух паралельно осі Z до (Z3)



Приклад 2 - Відносне положення G81

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Припустимо, що поточна позиція (X1, Y2, Z3) і інтерпретується попередній рядок коду NC.

Для цього необхідний режим інкрементальної відстані (G91) і режим відведення OLD_Z (G98). Також необхідно тричі повторити цикл свердління G81. Значення X дорівнює 4, значення Y дорівнює 5, значення Z дорівнює -0,6, а значення R дорівнює 1,8. Початкове положення X дорівнює 5 (=1+4), початкове положення Y дорівнює 7 (=2+5), чітке положення Z дорівнює 4,8 (=1,8+3), а положення Z дорівнює 4,2 (=4,8-0,6). OLD_Z дорівнює 3.

Перший попередній рух – це максимально швидкий рух вздовж осі Z до (X1, Y2, Z4.8), оскільки OLD_Z < очистити Z.

Перший повтор складається з 3 рухів.

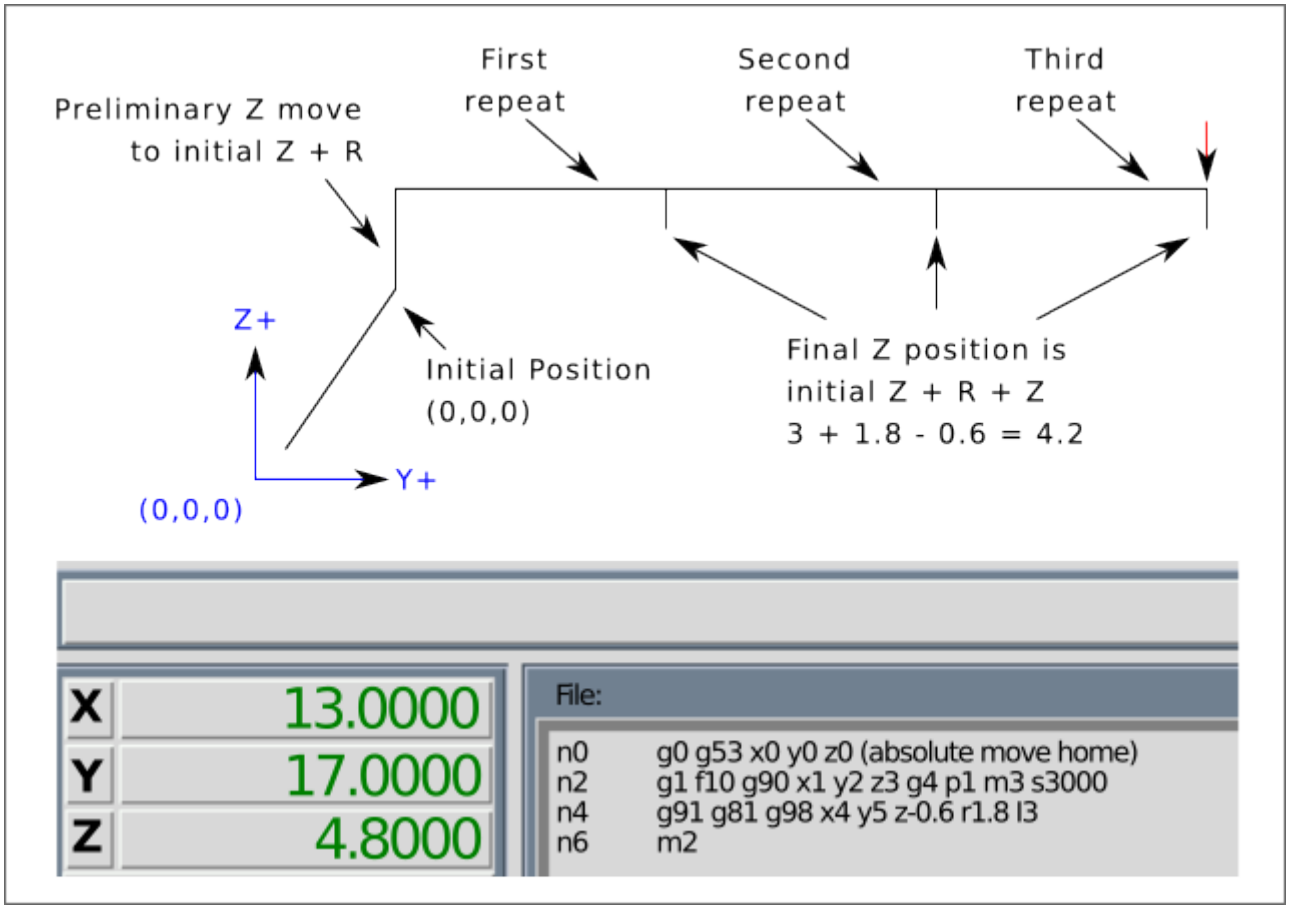
- A rapid move паралельно площині XY до (X5, Y7)
- Переміщення паралельно осі Z у точці feed rate до (Z4.2)
- Швидкий рух паралельно осі Z до (X5, Y7, Z4.8)

Другий повтор складається з 3 рухів. Позиція X скидається до 9 (=5+4), а позиція Y — до 12 (=7+5).

- A rapid move паралельно площині XY до (X9, Y12, Z4.8)
- Переміщуватися паралельно осі Z зі швидкістю подачі до (X9, Y12, Z4.2)
- Швидкий рух паралельно осі Z до (X9, Y12, Z4.8)

Третій повтор складається з 3 рухів. Позиція X скидається до 13 (=9+4), а позиція Y — до 17 (=12+5).

- A rapid move паралельно площині XY до (X13, Y17, Z4.8)
- Переміщуватися паралельно осі Z зі швидкістю подачі до (X13, Y17, Z4.2)
- Швидкий рух паралельно осі Z до (X13, Y17, Z4.8)

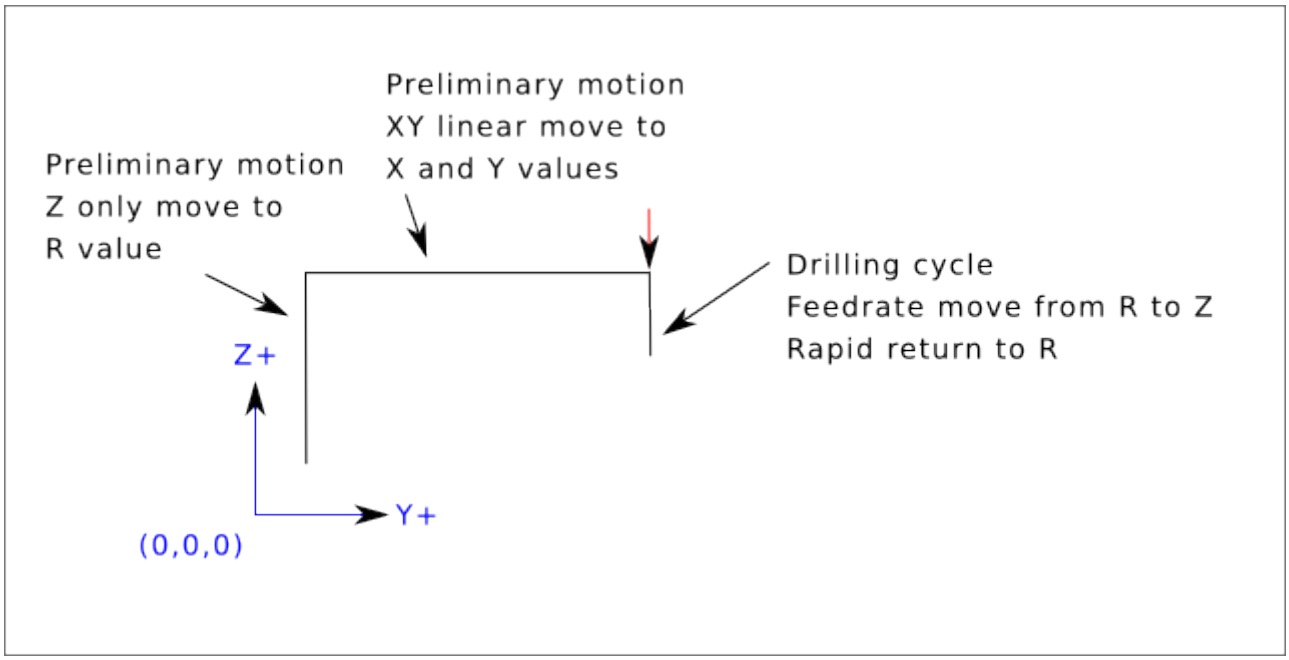


Приклад 3 - Відносне положення G81

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

Тепер припустимо, що ви виконуєте перший блок коду G81, але починаючи з (X0, Y0, Z0), а не з (X1, Y2, Z3).

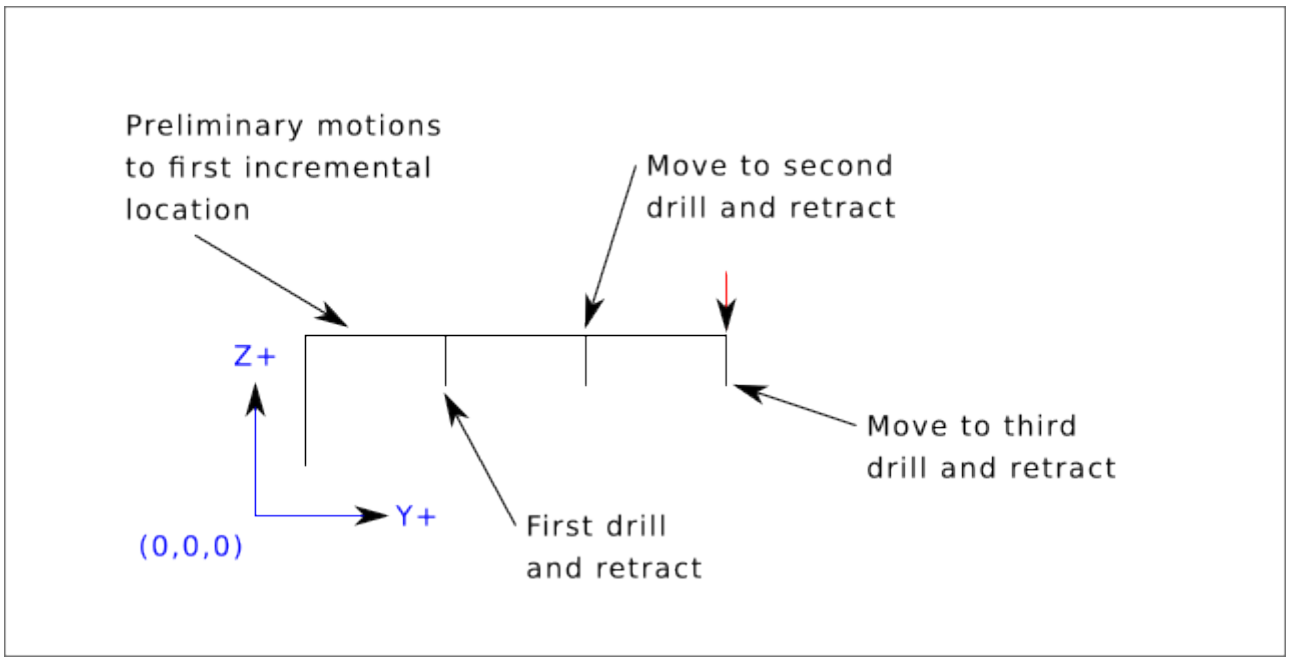
Оскільки OLD_Z менше значення R, це нічого не додає до руху, але оскільки початкове значення Z менше за значення, вказане в R, під час попередніх рухів відбудеться початковий рух Z.



Приклад 4 – Абсолютний G81 R > Z Це графік траєкторії руху для другого блоку коду g81.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Оскільки цей графік починається з (X0, Y0, Z0), інтерпретатор додає початкові значення Z0 і R1.8 і швидко переміщається в це місце. Після цього початкового переміщення по осі Z функція повторення працює так само, як у прикладі 3, причому кінцева глибина по осі Z становить 0,6 нижче значення R.



Приклад 5 - Відносне положення R > Z

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

Оскільки цей графік починається з (X0, Y0, Z0), інтерпретатор додає початкові значення Z0 і R1.8 і швидко переміщається в це місце, як у «Прикладі 4». Після цього початкового переміщення по

осі Z виконується rapid move до X4 Y5. Потім остаточна глибина по осі Z становить 0,6 нижче значення R. Функція повторення змусить Z знову переміститися в те саме місце.

11.5.46 Цикл свердління G82, затримка

```
G82 (X- Y- Z-) b''ab''b''6b''b''ob'' (U- V- W-) R- L- P-
```

Цикл «G82» призначений для свердління з затримкою на дні отвору.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістіть вісь Z у поточній позиції feed rate у позицію Z.
- Затримайтеся на певну кількість секунд P.
- Вісь Z виконує rapid move для очищення осі Z.

Рух стандартного циклу G82 виглядає так само, як і G81, але з додаванням затримки внизу руху по осі Z. Тривалість затримки визначається словом *P-* у блоці G82.

```
G90 G82 G98 X4 Y5 Z1.5 R2.8 P2
```

Це буде схоже на приклад 3 вище, тільки з додатковою затримкою 2 секунди на дні отвору.

11.5.47 Цикл свердління з відведенням зубців G83

```
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.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістити вісь Z у поточній точці feed rate вниз на дельту або до позиції Z, залежно від того, яка з них менш глибока.
- Швидке повернення до площини відведення, визначеної словом R.
- Rapid up either the D value, the G83_PECK_CLEARANCE specified in the INI file or the default of .010” / 0.254 mm.
- Повторюйте кроки 2, 3 та 4, доки не буде досягнуто положення Z на кроці 2.
- Вісь Z виконує rapid move для очищення осі Z.

Це помилка, якщо:

- число Q від’ємне або дорівнює нулю.

11.5.48 G84 Цикл нарізання різьби праворуч, затримка

```
G84 (X- Y- Z-) b''ab''b''6b''b''ob'' (U- V- W-) R- L- P- $- F-
```

- *R-* - Відведення положення вздовж осі Z.
- *L-* – Використовується в інкрементальному режимі; кількість разів для повторення циклу. Див. приклади G81.
- *P-* - Час витримки (секунди).
- *$-* - Вибраний шпиндель.
- *F-* - Швидкість подачі (швидкість обертання шпинделя, помножена на відстань, пройдену за оберт (крок різьби)).



Warning
G84 не використовує синхронізований рух.

Цикл *G84* призначений для нарізання різьби з плаваючим патроном та затримкою на дні отвору.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Вимкнути коригування подачі та швидкості.
- Перемістіть вісь Z з поточною швидкістю подачі в позицію Z.
- Зупинити вибраний шпиндель (вибраний параметром $)
- Почніть обертання шпинделя проти годинникової стрілки.
- Затримайтеся на певну кількість секунд P.
- Перемістіть вісь Z з поточною швидкістю подачі, щоб очистити Z
- Відновлення подачі та швидкості дозволяє перевизначення до попереднього стану

Тривалість витримки визначається словом «P-» у блоці G84. Швидкість подачі «F-» дорівнює швидкості обертання шпинделя, помноженій на відстань за один оберт (крок різьби). У прикладі S100 з кроком різьби 1,25 мм за один оберт швидкість подачі становить F125.

11.5.49 Цикл розточування G85, вихідна подача

```
G85 (X- Y- Z-) b''ab''b''6b''b''ob'' (U- V- W-) R- L-
```

Цикл «G85» призначений для розточування або розгортання, але може використовуватися для свердління або фрезерування.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістити вісь Z лише за поточною точкою feed rate до позиції Z.
- Відведіть вісь Z з поточною швидкістю подачі до площини R, якщо вона нижча за початкову Z.
- Відведіть зі швидкістю переміщення, щоб очистити Z.

11.5.50 G86 Цикл розточування, зупинка шпинделя, швидкий вихід

```
G86 (X- Y- Z-) b’’ab’’b’’6b’’b’’ob’’ (U- V- W-) R- L- P- $-
```

Цикл «G86» призначений для розточування. У цьому циклі використовується число P для позначення кількості секунд витримки.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістити вісь Z лише за поточною точкою feed rate до позиції Z.
- Затримайтеся на певну кількість секунд P.
- Зупинити обертання вибраного шпинделя. (Вибрано параметром $)
- Вісь Z виконує rapid move для очищення осі Z.
- Перезапустіть шпиндель у попередньому напрямку.

Це помилка, якщо:

- шпиндель не обертається до виконання цього циклу.

11.5.51 Цикл зворотного розточування G87

Цей код наразі не реалізовано в LinuxCNC. Він прийнятний, але його поведінка не визначена.

11.5.52 G88 Цикл розточування, зупинка шпинделя, ручний вихід

Цей код наразі не реалізовано в LinuxCNC. Він прийнятний, але його поведінка не визначена.

11.5.53 G89 Цикл розточування, затримка, вихідна подача

```
G89 (X- Y- Z-) b’’ab’’b’’6b’’b’’ob’’ (U- V- W-) R- L- P-
```

Цикл «G89» призначений для розточування. У цьому циклі використовується число P, де P вказує на кількість секунд затримки.

- Попередній клопотання, як описано в розділі Попереднє та проміжне клопотання.
- Перемістити вісь Z лише за поточною точкою feed rate до позиції Z.
- Затримайтеся на певну кількість секунд P.
- Відведіть вісь Z з поточною швидкістю подачі, щоб очистити ось Z.

11.5.54 G90, G91 Режим відстані

- «G90» — режим абсолютної відстані У режимі абсолютної відстані номери осей (X, Y, Z, A, B, C, U, V, W) зазвичай позначають положення в поточній активній системі координат. Винятки з цього правила описані в розділі G80 G89.
- *G91* - режим приросту відстані У режимі приросту відстані номери осей зазвичай представляють прирости від поточної координати.

Приклад G90

```
G90 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b ↩
    ''pb''b''eb''b''жb''b''иb''b''мb''b''уb'' b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b' ↩
    'тb''b''нb''b''ob''b''їb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G0 X2.5 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'ib''b''щb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' b''кb''b''ob''b''ob''b''pb''b' ↩
    'дb''b''иb''b''нb''b''ab''b''тb''b''иb'' X2.5, b''вb''b''кb''b''лb''b''юb''b''чb''b' ↩
    'ab''b''юb''b''чb''b''иb'' b''бb''b''уb''b''дb''b''ьb''-b''яb''b''кb''b''ib'' b''дb''b' ↩
    'ib''b''юb''b''чb''b''ib'' b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''нb''b''яb'')
```

Приклад G91

```
G91 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b ↩
    ''pb''b''eb''b''жb''b''иb''b''мb''b''уb'' b''пb''b''pb''b''иb''b''pb''b''ob''b''cb''b' ↩
    'тb''b''уb'' b''вb''b''ib''b''дb''b''cb''b''тb''b''ab''b''нb''b''ib'')
G0 X2.5 (b''шb''b''вb''b''иb''b''дb''b''кb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''мb''b' ↩
    'ib''b''щb''b''eb''b''нb''b''нb''b''яb'' 2.5 b''вb''b''ib''b''дb'' b''пb''b''ob''b''тb'' ↩
    b''ob''b''чb''b''нb''b''ob''b''їb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''ib''b''їb'' b' ↩
    'вb''b''зb''b''дb''b''ob''b''вb''b''жb'' b''ob''b''cb''b''ib'' X)
```

- Див. розділ G0 для отримання додаткової інформації.

11.5.55 G90.1, G91.1 Режим дугової відстані

- *G90.1* – режим абсолютної відстані для зміщень I, J та K. Коли діє G90.1, I та J повинні бути вказані за допомогою G2/3 для площини XY або J та K для площини XZ, інакше це помилка.
- *G91.1* – режим поступового збільшення відстані для зміщень I, J та K. `G91.1` Повертає I, J та K до їхньої поведінки за замовчуванням.

11.5.56 Зміщення системи координат G92

```
G92 axes
```



Warning
Використовуйте «G92» лише після того, як ваш верстат буде позиціоновано в потрібну точку.

G92 надає поточній точці бажані координати (без руху), де слова осей містять бажані номери осей. Всі слова осей є необов’язковими, але принаймні одне з них має бути використано. Якщо слово осі не використовується для певної осі, зміщення для цієї осі буде дорівнювати нулю.

При виконанні команди «G92» origins всіх систем координат зміщуються. Вони зміщуються таким чином, що значення поточної контрольної точки в активній системі координат стає заданим значенням. Всі початки систем координат (G53-G59.3) зміщуються на цю ж відстань.

G92 використовує значення, збережені в parameters 5211-5219, як значення зміщення X Y Z A B C U V W для кожної осі. Значення параметрів є «абсолютними» координатами верстата в «одиницях виміру» верстата, як зазначено в файлі INI. Усі осі, визначені в файлі INI, будуть зміщені, коли G92 активний. Якщо ось не була введена після G92, зміщення цієї осі буде дорівнювати нулю.

Наприклад, припустимо, що поточна точка знаходиться в точці X=4 і в даний момент не активне зміщення «G92». Потім програмується «G92 X7». Це переміщує всі початки координат на -3 в X, що призводить до того, що поточна точка стає X=7. Це значення -3 зберігається в параметрі 5211.

Перебування в режимі поступового збільшення відстані (G91 замість G90) не впливає на дію `G92`.

Зміщення *G92* можуть вже бути активними, коли викликається `G92`. Якщо це так, зміщення замінюється новим зміщенням, яке робить поточну точку заданою.

Це помилка, якщо всі слова осі пропущені.

LinuxCNC зберігає зміщення G92 та повторно використовує їх під час наступного запуску програми. Щоб запобігти цьому, можна запрограмувати `G92.1` (щоб стерти їх) або запрограмувати `G92.2` (щоб видалити їх - вони все одно зберігаються).

Note
Команду «G52» також можна використовувати для зміни цього зміщення; див. розділ Локальні та глобальні зміщення для отримання додаткової інформації про `G92` та `G52`, а також про те, як вони взаємодіють.

Див. розділ Coordinate System для огляду систем координат.

Див. розділ Parameters для отримання додаткової інформації.

11.5.57 G92.1, G92.2 Скидання зміщень G92

- *G92.1* - вимкнути зміщення G92 та скинути parameters 5211 - 5219 до нуля.
- *G92.2* – вимкнути зміщення G92, але залишити parameters 5211–5219 доступними.

Note
G92.1 очищує лише зміщення G92, щоб змінити зміщення системи координат G53-G59.3 у G-коді, використовуйте або G10 L2, або G10 L20.

11.5.58 G92.3 Відновлення зміщень G92

- *G92.3* - встановити зміщення `G92` на значення, збережені в параметрах з 5211 по 5219

Ви можете встановити зміщення осей в одній програмі і використовувати ті ж зміщення в іншій програмі. Програма `G92` в першій програмі. Це встановить параметри 5211 до 5219. Не використовуйте `G92.1` в решті першої програми. Значення параметрів будуть збережені при виході з першої програми і відновлені при запуску другої. Використовуйте `G92.3` на початку другої програми. Це відновить зміщення, збережені в першій програмі.

11.5.59 Режим швидкості подачі G93, G94, G95

- «G93» — це режим зворотного часу. У режимі зворотного часу подачі F-команда означає, що рух повинен бути виконаний за [один поділений на число F] хвилин. Наприклад, якщо число F дорівнює 2,0, рух повинен бути виконаний за півхвилини.

  Коли активний режим зворотної швидкості подачі, слово F повинно з'являтися в кожному рядку, що містить рух G1, G2 або G3, а слово F в рядку, що не містить G1, G2 або G3, ігнорується. Перебування в режимі зворотної швидкості подачі не впливає на рухи G0 (rapid move).

- «G94» — режим одиниць на хвилину. У режимі подачі одиниць на хвилину слово F інтерпретується як значення, що контрольована точка повинна рухатися з певною швидкістю в дюймах на хвилину, міліметрах на хвилину або градусах на хвилину, залежно від того, які одиниці довжини використовуються і яка вісь або осі рухаються.

- «G95» — режим «Одиниці на оберт». У режимі «Одиниці на оберт» команда F-word інтерпретується як вказівка на те, що контрольована точка повинна переміщатися на певну кількість дюймів за оберт шпинделя, залежно від того, які одиниці довжини використовуються і яка вісь або осі рухаються. G95 не підходить для нарізування різьби, для нарізування різьби використовуйте G33 або G76. G95 вимагає підключення spindle.N.speed-in. Фактичний шпиндель, з яким синхронізується подача, вибирається параметром $.

Це помилка, якщо:

- Режим зворотної подачі часу активний, і рядок з G1, G2 або G3 (явно чи неявно) не має F-слова.
- Нова швидкість подачі не вказана після перемикання на G94 або G95

11.5.60 Режим керування шпинделем G96, G97

```
G96 <D-> S- <$-> (b''Pb''b''eb''b''жb''b''иb''b''мb'' b''пb''b''оb''b''сb''b''тb''b''ib''b' ↩
    'йb''b''нb''b''оb''b''їb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''оb''b''сb''b''тb''b' ↩
    'ib'' b''пb''b''оb''b''вb''b''eb''b''pb''b''xb''b''нb''b''ib'')
G97 S- <$-> (b''Pb''b''eb''b''жb''b''иb''b''мb'' b''оb''b''бb''b''eb''b''pb''b''тb''b''ib'' ↩
   b''вb'' b''зb''b''аb'' b''xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb'')
```

1. *D* - максимальна швидкість обертання (об/хв), опціонально
2. *S* - швидкість шпинделя
3. *$* - шпиндель, швидкість якого буде змінюватися, необов'язково.
   - *G96 S- <D->* - вибирає постійну поверхневу швидкість *S*:
     - У футах за хвилину, якщо діє G20,
     - або метрів за хвилину, якщо діє G21.

При використанні G96 переконайтеся, що X0 в поточній системі координат (включаючи зміщення та довжину інструменту) є центром обертання, інакше LinuxCNC не забезпечить бажану швидкість поверхні. G96 не залежить від режиму радіуса або діаметра.

Щоб досягти режиму CSS на вибраних шпинделях, запрограмуйте послідовні команди G96 для кожного шпинделя перед видачею M3.

- «G97» вибирає режим обертів за хвилину.

Приклад рядка G96

```
G96 D2500 S250 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' CSS ↩
    b''зb'' b''мb''b''ab''b''кb''b''cb''b''иb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''иb''b' ↩
    'мb''b''иb'' b''ob''b''бb''b''eb''b''pb''b''тb''b''ab''b''мb''b''иb'' 2500 b''тb''b' ↩
    'ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''юb'' b''ob''b''бb''b' ↩
    'pb''b''ob''b''бb''b''кb''b''иb'' b''пb''b''ob''b''вb''b''eb''b''pb''b''xb''b''нb''b' ↩
    'ib'' 250)
```

Це помилка, якщо:

- S не вказано з G96
- Подача задається в режимі G96, коли шпиндель не обертається

11.5.61 G98, G99 Рівень повернення стандартного циклу

Коли шпиндель відводить назад під час стандартних циклів, є два варіанти на вибір:

- *G98* - відведення в положення, в якому вісь знаходилася безпосередньо перед запуском цієї серії з одного або кількох суміжних стандартних циклів.
- *G99* - відведення в положення, задане словом R стандартного циклу.

Запрограмуйте «G98», і готовий цикл буде використовувати положення Z перед готовим циклом як положення повернення Z, якщо воно вище за значення R, вказане в циклі. Якщо воно нижче, буде використовуватися значення R. Слово R має різне значення в режимі абсолютної відстані та режимі інкрементальної відстані.

G98 Відведення до початку

```
G0 X1 Y2 Z3
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

Код G98 у другому рядку вище означає, що зворотний рух буде до значення Z у першому рядку, оскільки воно вище за вказане значення R.

«Початкова» (G98) площина скидається щоразу, коли режим циклічного руху припиняється, як явно (G80), так і неявно (будь-який код руху, що не є циклом). Перемикання між режимами циклу (наприклад, з G81 на G83) НЕ скидає «початкову» площину. Під час серії циклів можна перемикатися між G98 і G99.

11.6 M-Коди

11.6.1 Таблиця швидкого довідника M-кодів

Код	Опис
M0 M1	Пауза програми
M2 M30	Кінець програми
M60	Пауза зміни піддону
M3 M4 M5	Управління шпинделем
M6	Зміна інструменту
M7 M8 M9	Контроль охолоджувальної рідини
M19	Орієнтний шпиндель

Код	Опис
M48 M49	Увімкнення/вимкнення корекції подачі та шпинделя
M50	Керування перевизначенням подачі
M51	Керування перемиканням шпинделя
M52	Адаптивне керування подачею
M53	Контроль зупинки подачі
M61	Встановити номер поточного інструменту
m62-m65	Вихідний контроль
M66	Контроль введення
M67	Керування аналоговим виходом
M68	Керування аналоговим виходом
M70	Зберегти стан модального вікна
M71	Анулювати збережений модальний стан
M72	Відновити модальний стан
M73	Зберегти стан модального вікна автовідновлення
M98 M99	Виклик та повернення з підпрограми
M100-M199	Користувацькі M-коди

11.6.2 M0, M1 Пауза програми

* *M0* - тимчасово призупинити запущену програму. LinuxCNC залишається в автоматичному режимі, тому MDI та інші ручні дії не активовані. Натискання кнопки відновлення перезапустить програму з наступного рядка. * «M1» — тимчасово призупинити виконання програми, якщо ввімкнено опціональний вимикач зупинки. LinuxCNC залишається в автоматичному режимі, тому MDI та інші ручні дії не ввімкнені. Натискання кнопки відновлення призведе до перезапуску програми з наступного рядка.

Note
Програмувати *M0* та *M1* у режимі MDI можна, але ефект, ймовірно, не буде помітним, оскільки нормальною поведінкою в режимі MDI є зупинка після кожного рядка введення.

11.6.3 Кінець програми M2, M30

* *M2* - завершення програми. Натискання `Cycle Start` (”R” в графічному інтерфейсі Axis) перезапустит програму з початку файлу. * «M30» – замінити палетні човники та завершити програму. Натискання кнопки «Cycle Start» запустить програму з початку файлу.

Обидві ці команди мають такі ефекти:

- Перехід з автоматичного режиму на режим ручного введення даних.
- Зміщення початку координат встановлені за замовчуванням (наприклад, «G54»).
- Вибрана площина встановлюється як площина XY (наприклад, *G17*).
- Режим відстані встановлено на абсолютний режим (наприклад, «G90»).
- Режим швидкості подачі встановлено на одиниці за хвилину (наприклад, *G94*).

- Коригування подачі та швидкості встановлене на УВІМК. (наприклад, «M48»).
- Компенсацію на різець вимкнено (як у випадку з «G40»).
- Шпиндель зупинено (як у випадку з *M5*).
- Поточний режим руху встановлено на подачу (наприклад, *G1*).
- Охолоджувальна рідина вимкнена (як у випадку з *M9*).

Note
Рядки коду після M2/M30 не будуть виконані. Натискання кнопки Cycle Start запустить програму з початку файлу.



Warning
Використання % для обгортання G-коду не виконує ту саму дію, що й «Кінець програми». Див. розділ File Requirements для отримання додаткової інформації про те, чого не виконує використання %.

11.6.4 Пауза зміни піддону M60

* «M60» — обмін палетними челноками, а потім тимчасове призупинення програми, що виконується (незалежно від налаштування додаткового вимикача зупинки). Натискання кнопки запуску циклу призведе до поновлення програми з наступного рядка.

11.6.5 M3, M4, M5 Шпиндельне керування

* *M3 [$n]* - запустіть вибраний шпиндель за годинниковою стрілкою на швидкості *S*. * *M4 [$n]* - запустіть вибраний шпиндель проти годинникової стрілки на швидкості *S*. * *M5 [$n]* - зупинити вибраний шпиндель.

Використовуйте $ для роботи з певними шпинделями. Якщо $ пропущено, команди за замовчуванням працюють зі шпинделем 0. Використовуйте $-1 для роботи з усіма активними шпинделями.

У цьому прикладі шпинделі 0, 1 та 2 будуть запущені одночасно з різними швидкостями:

```
S100 $0
S200 $1
S300 $2
M3 $-1
```

У цьому прикладі шпиндель 1 буде обертатися в зворотному напрямку, але інші шпинделі залишаться обертатися вперед:

```
M4 $1
```

І це зупинить шпиндель 2, а інші шпинделі залишать обертатися:

```
M5 $2
```

Якщо символ $ пропущено, то поведінка буде точно такою ж, як і для одношпиндельного верстата.

Можна використовувати «M3» або «M4», якщо швидкість шпинделя S встановлена на нуль. У цьому випадку (або якщо перемикач перевищення швидкості увімкнений і встановлений на нуль) шпиндель не почне обертатися. Якщо пізніше швидкість шпинделя буде встановлена вище нуля (або перемикач перекриття буде увімкнений), шпиндель почне обертатися. Можна використовувати «M3» або «M4», коли шпиндель вже обертається, або «M5», коли шпиндель вже зупинився.

11.6.6 Зміна інструменту M6

11.6.6.1 Ручна зміна інструменту

Якщо завантажено компонент HAL «hal_manualtoolchange», M6 зупинить шпиндель і запропонує користувачеві замінити інструмент відповідно до останнього запрограмованого номера «T-». Докладні інформацію про hal_manualtoolchange див. у розділі Manual Tool Change.

11.6.6.2 Зміна інструменту

Щоб замінити інструмент у шпинделі з поточного інструменту на останній вибраний інструмент (за допомогою слова T — див. розділ Вибір інструменту), запрограмуйте «M6». Після завершення заміни інструменту:

- Шпиндель зупиниться.
- Інструмент, який було вибрано (за допомогою слова T у тому ж рядку або у будь-якому рядку після попередньої зміни інструменту), буде знаходитися у шпинделі.
- Якщо вибраний інструмент не був у шпинделі перед зміною інструмента, інструмент, який був у шпинделі (якщо він був), буде поміщений назад у магазин пристрою зміни інструментів.
- Якщо налаштовано у файлі INI, деякі позиції осей можуть зміщуватися під час видачі M6. Див. розділ EMCIO для отримання додаткової інформації про опції зміни інструменту.
- Жодних інших змін не буде внесено. Наприклад, охолоджувальна рідина продовжуватиме подавати під час зміни інструменту, якщо її не вимкнути за допомогою «M9».

Note
Слово «T-» – це ціле число, що позначає номер гнізда інструмента в каруселі (не його індекс).



Warning
Зміщення довжини інструмента не змінюється командою «M6», використовуйте «G43» після «M6», щоб змінити зміщення довжини інструмента.

Зміна інструменту може включати рух осі. Можна (але це не доцільно) запрограмувати зміну інструменту, який вже знаходиться в шпинделі. Немає нічого страшного, якщо у вибраному слоті немає інструменту; в цьому випадку шпиндель буде порожнім після зміни інструменту. Якщо останнім був вибраний слот нуль, то після зміни інструменту в шпинделі точно не буде інструменту. Змінник інструменту повинен бути налаштований для виконання зміни інструменту в HAL і, можливо, ClassicLadder.

11.6.7 M7, M8, M9 Контроль охолоджувальної рідини

* *M7* - Увімкніть розпилювач охолоджувальної рідини. M7 керує контактом iocontrol.0.coolant-mist. * *M8* - Увімкнути протікання охолоджувальної рідини. M8 керує контактом iocontrol.0.coolant-flood. * *M9* - Вимкніть обидва M7 та M8.

Підключіть один або обидва контакти керування охолоджуючою рідиною до HAL, перш ніж M7 або M8 керуватимуть виходом. M7 та M8 можна використовувати для ввімкнення будь-якого виходу за допомогою G-коду.

Ви можете використовувати будь-яку з цих команд, незалежно від поточного стану охолоджувальної рідини.

11.6.8 Шпиндель M19 Orient

```
M19 R- Q- [P-] [$-]
```

- *R* Позиція для повороту від 0, дійсний діапазон 0-360 градусів
- *Q* Кількість секунд очікування до завершення орієнтації. Якщо spindle.N.is-oriented не стає true протягом таймауту Q, виникає помилка.
- *P* Напрямок повороту в потрібне положення.
  - Поворот на «0» для найменшого кутового переміщення (за замовчуванням)
  - *1* завжди обертайте за годинниковою стрілкою (те саме, що й напрямок M3)
  - *2* завжди обертайте проти годинникової стрілки (як у напрямку M4)
- *$* Шпиндель для орієнтації (фактично визначає лише, які контакти HAL передають команди положення шпинделя)

M19 – це команда модальної групи 7, як і M3, M4 та M5. M19 очищається будь-якою з M3, M4, M5.

Орієнтація шпинделя вимагає квадратурного енкодера з індексом для визначення положення вала шпинделя та напрямку обертання.

Налаштування INI у розділі [RS274NGC]:

- ORIENT_OFFSET = 0-360 (фіксоване зміщення в градусах, додане до слова M19 R)
- Піни HAL
  - *spindle.N.orient-angle* (out float) Бажана орієнтація шпинделя для M19. Значення параметра R-слова M19 плюс значення параметра INI [RS274NGC]ORIENT_OFFSET.
  - *spindle.N.orient-mode* (out s32) Бажаний режим обертання шпинделя. Відображає слово параметра M19 P, за замовчуванням = 0.
  - *spindle.N.orient* (out bit) Вказує на початок циклу орієнтації шпинделя. Встановлюється за допомогою M19. Скидається за допомогою будь-якої з M3, M4, M5. Якщо помилка орієнтації шпинделя не дорівнює нулю під час значення «орієнтація шпинделя істинна», команда M19 завершується невдачею з повідомленням про помилку.
  - *spindle.N.is-oriented* (in bit) Контакт підтвердження орієнтації шпинделя. Завершує цикл орієнтації. Якщо орієнтація шпинделя була справжньою, коли було підтверджено орієнтацію шпинделя, контакт орієнтації шпинделя очищується, а контакт блокування шпинделя підтверджує Також підтверджується контакт гальма шпинделя.
  - *spindle.N.orient-fault* (in s32) Введено код помилки для орієнтаційного циклу. Будь-яке значення, відмінне від нуля, призведе до переривання орієнтаційного циклу.
  - *spindle.N.locked* (out bit) Повний штифт орієнтований на шпиндель. Очищається будь-яким з M3, M4, M5.

11.6.9 Керування швидкістю та подачею M48, M49

* *M48* - увімкнути елементи керування швидкістю шпинделя та швидкістю подачі. * *M49* - вимкніть обидва елементи керування.

Ці команди також приймають необов’язковий параметр $, щоб визначити, з яким шпинделем вони працюють.

Можна вмикати або вимикати елементи керування, якщо вони вже ввімкнені або вимкнені. Докладніше див. у розділі Feed Rate.

Їх також можна перемикати окремо за допомогою «M50» та «M51», див. нижче.

11.6.10 Керування корекцією подачі M50

* *M50 <P1>* - увімкнути керування коригуванням швидкості подачі. P1 необов'язковий. * *M50 P0* - вимкнути керування швидкістю подачі.

Якщо цю функцію вимкнено, корекція подачі не матиме жодного впливу, і рух виконуватиметься із запрограмованою швидкістю подачі (якщо адаптивна корекція швидкості подачі не активна).

11.6.11 M51 Керування корекцією швидкості шпинделя

* *M51 <P1> <$->*- увімкнути керування корекцією швидкості шпинделя для вибраного шпинделя. P1 є необов'язковим. * *M51 P0 <$->* - вимкнути програму керування корекцією швидкості шпинделя.

Якщо функція перевизначення швидкості шпинделя вимкнена, швидкість шпинделя не змінюватиметь і відповідатиме точно заданому в програмі значенню S-слова (описаному в розділі Швидкість шпинделя).

11.6.12 Адаптивне керування подачею M52

* *M52 <P1>* - використовуйте адаптивну подачу. P1 необов'язковий. * «M52 P0» – припинити використання адаптивної подачі.

Коли адаптивна подача ввімкнена, деякі зовнішні вхідні значення використовуються разом із значенням заміщення подачі в інтерфейсі користувача та заданою швидкістю подачі для встановлення фактичної швидкості подачі. У LinuxCNC для цієї мети використовується контакт HAL «motion.adaptive feed». Діапазон для «motion.adaptive-feed» визначається значенням MAX_FEED_OVERRIDE в розділі [DISPLAY] файлу ini і буде обмежений діапазоном від -MAX_FEED_OVERRIDE до MAX_FEED_OVE 0 еквівалентно утриманню подачі.

Note
Використання негативного адаптивного живлення для зворотного ходу є новою функцією і поки що не дуже добре протестовано. Передбачуване використання — для плазмових різаків та дротових електроіскрових верстатів, але не обмежується такими застосуваннями.

11.6.13 M53 Керування зупинкою подачі

* *M53 <P1>* - увімкнути перемикач зупинки подачі. P1 є опціональним. Увімкнення перемикача зупинки подачі дозволить перервати рух за допомогою управління зупинкою подачі. У LinuxCNC для цієї мети використовується контакт HAL *motion.feed-hold*. Значення *true* призведе до зупинки руху, коли *M53* активний. * *M53 P0* - вимкніть перемикач зупинки подачі. Стан «motion.feed-hold» не впливатиме на подачу, коли M53 неактивний.

11.6.14 M61 Встановити поточний інструмент

* «M61 Q-» — зміна поточного номера інструменту в режимі MDI або ручному режимі без заміни інструменту. Одне з застосувань — коли ви вмикаєте LinuxCNC з інструментом, що знаходиться в шпинделі, ви можете встановити номер цього інструменту без заміни інструменту.



Warning
Зміщення довжини інструмента не змінюється командою *M61*, використовуйте *G43* після *M61*, щоб змінити зміщення довжини інструмента.

Це помилка, якщо:

- Q- не дорівнює 0 або більше

11.6.15 M62 - M65 Цифрове керування виходом

* *M62 P-* - увімкнути цифровий вихід, синхронізований з рухом. * *M63 P-* - вимкнути цифровий вихід, синхронізований з рухом. * *M64 P-* - негайно увімкніть цифровий вихід. * *M65 P-* - негайно вимкніть цифровий вихід.

Слово P визначає номер цифрового виходу. Слово P може мати значення від 0 до стандартного значення 3. При необхідності кількість входів/виходів можна збільшити за допомогою параметра num_dio під час завантаження контролера руху. Докладнішу інформацію див. у розділі Motion.

Команди M62 і M63 будуть поставлені в чергу. Наступні команди, що посилаються на той самий номер виходу, замінять старі налаштування. Можна вказати більше ніж одну зміну виходу, видавши більше ніж одну команду M62/M63.

Фактична зміна зазначених виходів відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходів у черзі не відбудуться. Найкраще завжди програмувати G-код руху (G0, G1 тощо) одразу після M62/63.

Сигнали M64 та M65 виникають одразу після отримання контролером руху. Вони не синхронізовані з рухом і порушують змішування.

Note
M62-65 не працюватиме, якщо відповідні контакти motion.digital-out-*nn* не підключені у вашому HAL-файлі до виходів.

11.6.16 M66 Очікування введення

```
M66 P- | E- <L->
```

- *P-* - визначає номер цифрового входу від 0 до 3. (Налаштовується з аргументу motmod num_dio)
- *E-* - визначає номер аналогового входу від 0 до 3. (Налаштовується з аргументу motmod num_aio)
- *L-* - визначає тип режиму очікування.
  - *Mode 0: IMMEDIATE* - без очікування, повертається негайно. Поточне значення вхідного сигналу зберігається в параметрі №5399
  - *Mode 1: RISE* - очікує, поки вибраний вхід виконає подію підвищення.
  - *Mode 2: FALL* - очікує, поки вибраний вхід виконає подію падіння.
  - *Mode 3: HIGH* - очікує, поки вибраний вхід перейде у стан HIGH.
  - *Mode 4: LOW* - очікує, поки вибраний вхід перейде у стан НИЗЬКОГО рівня.
- *Q-* - визначає час очікування в секундах. Якщо час очікування перевищено, очікування переривається а змінна #5399 зберігає значення -1. Значення Q ігнорується, якщо L-слово дорівнює нулю (IMMEDIATE). Значення Q, що дорівнює нулю, є помилкою, якщо L-слово не дорівнює нулю.
- Для аналогового входу дозволений лише режим 0.

Приклади ліній M66

```
M66 P0 L3 Q5 (b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'' b''дb''b' ↩
    'ob'' 5 b''cb''b''eb''b''кb''b''уb''b''нb''b''дb'', b''пb''b''ob''b''кb''b''иb'' b''цb'' ↩
    b''иb''b''фb''b''pb''b''ob''b''вb''b''иb''b''йb'' b''вb''b''хb''b''ib''b''дb'' 0 b''уb'' ↩
    b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''тb''b''ьb''b''cb''b''яb'')
```

Очікування введення M66 зупиняє подальше виконання програми, доки не відбудеться вибрана подія (або запрограмований тайм-аут).

Помилкою є програмування M66 як за допомогою слова P, так і слова E (тобто вибір як аналогового, так і цифрового входу). У LinuxCNC ці входи не контролюються в режимі реального часу, тому їх не слід використовувати для додатків, де важливий час реагування.

Кількість операцій вводу/виводу можна збільшити за допомогою параметра num_dio або num_aio під час завантаження контролера руху. Див. розділ Motion для отримання додаткової інформації.

Note
M66 не працюватиме, якщо відповідні контакти motion.digital-in-*nn* або motion.analog-in-*nn* не підключені у вашому HAL-файлі до входу.

Приклад HAL-з'єднання

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

11.6.17 Синхронізований аналоговий вихід M67

```
M67 E- Q-
```

- *M67* - встановити аналоговий вихід, синхронізований з рухом.
- *E-* - вихідне число в діапазоні від 0 до 3 (регулюється аргументом motmod num_aio)
- *Q-* - – значення, яке потрібно встановити (встановіть значення 0, щоб вимкнути).

Фактична зміна зазначених виходів відбудеться на початку наступної команди руху. Якщо наступної команди руху немає, зміни виходів у черзі не відбудуться. Найкраще завжди програмувати G-код руху (G0, G1 тощо) одразу після M67. M67 функціонує так само, як M62-63.

Кількість операцій вводу/виводу можна збільшити за допомогою параметра num_dio або num_aio під час завантаження контролера руху. Див. розділ Motion для отримання додаткової інформації.

Note
M67 не працюватиме, якщо відповідні контакти motion.analog-out-*nn* у вашому HAL-файлі не підключені до виходів.

11.6.18 Аналоговий вихід M68, негайний

```
M68 E- Q-
```

- *M68* - негайно налаштуйте аналоговий вихід.
- *E-* - вихідне число в діапазоні від 0 до 3. (Налаштовується з аргументу motmod num_aio)

- *Q*- - – значення, яке потрібно встановити (встановіть значення 0, щоб вимкнути).

Вихідні дані M68 надходять одразу після їх отримання контролером руху. Вони не синхронізовані з рухом і порушують змішування. M68 функціонує так само, як і M64-65..

Кількість операцій вводу/виводу можна збільшити за допомогою параметра num_dio або num_aio під час завантаження контролера руху. Див. розділ Motion для отримання додаткової інформації.

Note
M68 не працюватиме, якщо відповідні контакти motion.analog-out-*nn* у вашому HAL-файлі не підключені до виходів.

11.6.19 M70 Зберегти стан модального вікна

Щоб явно зберегти модальний стан на поточному рівні виклику, запрограмуйте *M70*. Після збереження модального стану за допомогою *M70* його можна відновити саме до цього стану, виконавши *M72*.

Пара інструкцій «M70» та «M72» зазвичай використовується для захисту програми від ненавмисних модальних змін у підпрограмах.

M70 Збережений стан

Збережений стан складається з:

- поточні налаштування G20/G21 (імперські/метричні системи)
- вибрана площина (G17/G18/G19 G17.1,G18.1,G19.1)
- стан компенсації різця (G40, G41, G42, G41.1, G42,1)
- режим відстані - відносний/абсолютний (G90/G91)
- режим подачі (G93/G94,G95)
- поточна система координат (G54-G59.3)
- Стан компенсації довжини інструменту (G43, G43.1, G49)
- режим втягування (G98,G99)
- режим шпинделя (G96-css або G97-RPM)
- режим дугової відстані (G90.1, G91.1)
- режим радіуса/діаметра токарного верстата (G7,G8)
- режим керування траєкторією (G61, G61.1, G64)
- струм подачі та швидкості (значення *F* та *S*)
- Стан шпинделя (M3, M4, M5) - увімкнено/вимкнено та напрямок
- статус туману (M7) та повені (M8)
- налаштування корекції швидкості (M51) та корекції подачі (M50)
- адаптивне налаштування подачі (M52)
- налаштування блокування подачі (M53)

Зверніть увагу, що, зокрема, режим руху (G1 тощо) НЕ відновлюється.

current call level означає або:

- виконання в основній програмі. На рівні основної програми існує єдине місце зберігання стану; якщо послідовно виконується кілька команд «M70», при виконанні команди «M72» відновлюється тільки останній збережений стан.
- виконання в підпрограмі G-коду. Стан, збережений за допомогою *M70* в підпрограмі, поводиться точно так само, як локальний іменований параметр - на нього можна посилатися тільки в межах виклику цієї підпрограми за допомогою *M72*, а після виходу з підпрограми параметр зникає.

Рекурсивний виклик підпрограми вводить новий рівень викликів.

11.6.20 M71 Анулювання збереженого модального стану

Модальний стан, збережений за допомогою *M70* або *M73* на поточному рівні виклику, є недійсним (його більше не можна відновити).

Наступний «M72» на тому ж рівні виклику не спрацює.

Якщо виконати підпрограму, яка захищає модальний стан за допомогою *M73*, наступний return або endsub **не** відновить модальний стан.

Корисність цієї функції сумнівна. На неї не слід покладатися, оскільки вона може зникнути.

11.6.21 M72 Відновлення модального стану

Стан модального вікна, збережений за допомогою коду *M70>>*, можна відновити, виконавши код *M72*.

Обробка G20/G21 здійснюється окремо, оскільки подача інтерпретується по-різному залежно від G20/G21: якщо одиниці довжини (мм/дюйм) мають бути змінені операцією відновлення, «M72» спочатку відновить режим відстані, а потім усі інші параметри, включаючи подачу, щоб забезпечити інтерпретацію значення подачі з правильними налаштуваннями одиниць виміру.

Виконання команди «M72» без попередньої операції збереження «M70» на цьому рівні є помилкою.

Наступний приклад демонструє збереження та явне відновлення модального стану навколо виклику підпрограми за допомогою *M70* та *M72*. Зверніть увагу, що підпрограма *imperialsub* не «усвідомлює» функції M7x і може використовуватися без змін:

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
g20 (b''ib''b''мb''b''пb''b''eb''b''pb''b''cb''b''ьb''b''кb''b''иb''b''йb'')
g91 (b''вb''b''ib''b''дb''b''нb''b''ob''b''cb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b ↩
    ''иb''b''мb'')
F5 (b''нb''b''иb''b''зb''b''ьb''b''кb''b''ab'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ab'')
S300 (b''нb''b''иb''b''зb''b''ьb''b''кb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ib'' ↩
    b''cb''b''тb''b''ьb'' b''ob''b''бb''b''eb''b''pb''b''тb''b''ab''b''нb''b''нb''b''яb'')
(b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'вb'' b''пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ib'', b' ↩
    'cb''b''тb''b''ab''b''нb'' b''зb''b''ab''b''pb''b''ab''b''зb'':)
o<showstate> call
O<imperialsub> endsub
```

```
; b''гb''b''ob''b''лb''b''ob''b''вb''b''нb''b''ab'' b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''ab''
g21 (b''мb''b''eb''b''тb''b''pb''b''иb''b''чb''b''нb''b''ab'' b''cb''b''иb''b''cb''b''тb''b ↩
    ''eb''b''мb''b''ab'')
g90 (b''ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''ab'' b''cb''b''иb''b''cb''b ↩
    ''тb''b''eb''b''мb''b''ab'')
f200 (b''вb''b''иb''b''cb''b''ob''b''кb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b''іb'' ↩
    b''cb''b''тb''b''ьb'')
S2500 (b''вb''b''иb''b''cb''b''ob''b''кb''b''ab'' b''шb''b''вb''b''иb''b''дb''b''кb''b' ↩
    'іb''b''cb''b''тb''b''ьb'')

(b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'вb'' b''ob''b''cb''b''нb''b''ob''b''вb''b''нb''b''ob''b''мb''b''уb'', b''cb''b''тb''b' ↩
    'ab''b''нb'' b''зb''b''ab''b''pb''b''ab''b''зb'' :)
o<showstate> call

M70 (b''зb''b''бb''b''eb''b''pb''b''eb''b''жb''b''eb''b''нb''b''нb''b''яb'' b''cb''b''тb''b ↩
    ''ab''b''нb''b''уb'' b''ab''b''бb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ab'' b''нb''b' ↩
    'ab'' b''гb''b''лb''b''ob''b''бb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b''уb'' b' ↩
    'pb''b''іb''b''вb''b''нb''b''іb'')
O<imperialsub> b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''
M72 (b''яb''b''вb''b''нb''b''eb'' b''вb''b''іb''b''дb''b''нb''b''ob''b''вb''b''лb''b''eb''b ↩
    ''нb''b''нb''b''яb'' b''cb''b''тb''b''ab''b''нb''b''уb'')

(b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'', b' ↩
    'нb''b''ab''b''зb''b''ab''b''дb'' b''уb'' b''гb''b''ob''b''лb''b''ob''b''вb''b''нb''b' ↩
    'eb'' b''мb''b''eb''b''нb''b''юb'', b''тb''b''eb''b''пb''b''eb''b''pb'' b''cb''b''тb''b' ↩
    'ab''b''нb'':)
o<showstate> call
m2
```

11.6.22 M73 Збереження та автовідновлення модального стану

Щоб зберегти модальний стан у підпрограмі та відновити стан у підпрограмі *endsub* або будь-якому шляху *return*, запрограмуйте *M73*.

Переривання виконання програми в підпрограмі, яка має операцію *M73*, **не** відновить стан.

Також, звичайний кінець (*M2*) головної програми, яка містить *M73*, **не** відновить стан.

Рекомендується використовувати на початку підпрограми O-word, як показано в наступному прикладі. Використання *M73* таким чином дозволяє створювати підпрограми, які потребують зміни модального стану, але захищають програму, що їх викликає, від ненавмисних модальних змін. Зверніть увагу на використання predefined named parameters в підпрограмі *showstate*.

```
O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsub> sub
M73 (b''зb''b''бb''b''eb''b''pb''b''eb''b''гb''b''тb''b''иb'' b''cb''b''тb''b''ab''b''нb''  ↩
    b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''уb'' b''вb'' b''пb''b''ob''b''тb''b''ob''b' ↩
    'чb''b''нb''b''ob''b''мb''b''уb'' b''кb''b''ob''b''нb''b''тb''b''eb''b''кb''b''cb''b' ↩
    'тb''b''іb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''уb'', b''вb''b''іb''b''дb''b' ↩
    'нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b''pb''b''иb'' b''пb''b''ob''b''вb''b' ↩
    'eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''іb'' b''ab''b''бb''b''ob'' endsub)
g20 (imperial)
g91 (b''вb''b''іb''b''дb''b''нb''b''ob''b''cb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b ↩
    ''иb''b''мb'')
F5 (b''нb''b''иb''b''зb''b''ьb''b''кb''b''ab'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ab'')
```

```
S300 (b''нв''b''ив''b''зв''b''ьв''b''кв''b''ав'' b''шв''b''вв''b''ив''b''дв''b''кв''b''ів'' ↩
    b''св''b''тв''b''ьв'' b''ов''b''бв''b''ев''b''рв''b''тв''b''ав''b''нв''b''нв''b''яв'')
(debug, b''вв'' b''пв''b''ів''b''дв''b''пв''b''рв''b''ов''b''гв''b''рв''b''ав''b''мв''b' ↩
    'ів'', b''св''b''тв''b''ав''b''нв'' b''зв''b''ав''b''рв''b''ав''b''зв'':)
o<showstate> call

; b''Пв''b''рв''b''ив''b''мв''b''ів''b''тв''b''кв''b''ав'' - b''тв''b''ув''b''тв'' b''нв''b ↩
    ''ев'' b''пв''b''ов''b''тв''b''рв''b''ів''b''бв''b''ев''b''нв'' M72 - b''нв''b''ав''b' ↩
    'св''b''тв''b''ув''b''пв''b''нв''b''ив''b''йв'' b''кв''b''ів''b''нв''b''цв''b''ев''b' ↩
    'вв''b''ив''b''йв'' b''пв''b''ів''b''дв''b''зв''b''ав''b''гв''b''ов''b''лв''b''ов''b' ↩
    'вв''b''ов''b''кв'' b''ав''b''бв''b''ов''
; b''яв''b''вв''b''нв''b''ев'' «b''пв''b''ов''b''вв''b''ев''b''рв''b''нв''b''ев''b''нв''b' ↩
    'нв''b''яв''» b''вв''b''ів''b''дв''b''нв''b''ов''b''вв''b''ив''b''тв''b''ьв'' b''св''b' ↩
    'тв''b''ав''b''нв'' b''ав''b''бв''b''ов''b''нв''b''ев''b''нв''b''тв''b''ав''
O<imperialsub> endsub

; b''гв''b''ов''b''лв''b''ов''b''вв''b''нв''b''ав'' b''пв''b''рв''b''ов''b''гв''b''рв''b' ↩
    'ав''b''мв''b''ав''
g21 (b''мв''b''ев''b''тв''b''рв''b''ив''b''чв''b''нв''b''ав'')
g90 (b''ав''b''бв''b''св''b''ов''b''лв''b''юв''b''тв''b''нв''b''ав'')
f200 (b''вв''b''ив''b''св''b''ов''b''кв''b''ав'' b''шв''b''вв''b''ив''b''дв''b''кв''b''ів'' ↩
    b''св''b''тв''b''ьв'')
S2500 (b''вв''b''ив''b''св''b''ов''b''кв''b''ив''b''йв'' b''ов''b''бв''b''ов''b''рв''b' ↩
    'ов''b''тв'')
(b''нв''b''ав''b''лв''b''ав''b''гв''b''ов''b''дв''b''жв''b''ев''b''нв''b''нв''b''яв'', b' ↩
    'вв'' b''ов''b''св''b''нв''b''ов''b''вв''b''нв''b''ів''b''йв'', b''св''b''тв''b''ав''b' ↩
    'нв'' b''зв''b''ав''b''рв''b''ав''b''зв'':)
b''вв''b''ив''b''кв''b''лв''b''ив''b''кв'' o<showstate>
b''вв''b''ив''b''кв''b''лв''b''ив''b''кв'' o<imperialsub>
(b''нв''b''ав''b''лв''b''ав''b''гв''b''ов''b''дв''b''жв''b''ев''b''нв''b''нв''b''яв'', b' ↩
    'нв''b''ав''b''зв''b''ав''b''дв'' b''вв'' b''ов''b''св''b''нв''b''ов''b''вв''b''нв''b' ↩
    'ів''b''йв'', b''св''b''тв''b''ав''b''нв'' b''зв''b''ав''b''рв''b''ав''b''зв'':)
b''вв''b''ив''b''кв''b''лв''b''ив''b''кв'' o<showstate>
m2
```

11.6.23 M98 і M99

Інтерпретатор підтримує головні та підпрограми у стилі Fanuc за допомогою M-кодів *M98* та *M99*. Див. Програми у стилі Fanuc.

11.6.23.1 Вибіркове відновлення модального стану

Виконання *M72* або повернення з підпрограми, яка містить *M73*, відновить all збережений модальний стан.

Якщо необхідно зберегти лише деякі аспекти модального стану, альтернативним варіантом є використання predefined named parameters, локальних параметрів та умовних операторів. Ідея полягає в тому, щоб запам'ятати режими, які необхідно відновити на початку підпрограми, і відновити їх перед виходом. Ось приклад, заснований на фрагменті *nc_files/tool-length-probe.ngc*:

```
O<measure> sub   (b''ib''b''нв''b''св''b''тв''b''рв''b''ув''b''мв''b''ев''b''нв''b''тв'' b' ↩
    'вв''b''ив''b''мв''b''ів''b''рв''b''юв''b''вв''b''ав''b''нв''b''нв''b''яв'')
;
#<absolute> = #<_absolute>  (b''зв''b''ав''b''пв''b''ав''b''мв''b''яв''b''тв''b''ав''b' ↩
    'тв''b''ив'' b''вв'' b''лв''b''ов''b''кв''b''ав''b''лв''b''ьв''b''нв''b''ів''b''йв'' b' ↩
    'зв''b''мв''b''ів''b''нв''b''нв''b''ів''b''йв'', b''яв''b''кв''b''щв''b''ов'' b''бв''b' ↩
    'ув''b''лв''b''ов'' b''вв''b''св''b''тв''b''ав''b''нв''b''ов''b''вв''b''лв''b''ев''b' ↩
    'нв''b''ов'' G90)
```

```
;
g30 (b''пв''b''eb''b''pb''b''eb''b''мв''b''ив''b''кв''b''ab''b''чв'' b''вв''b''ив''b''щв''b ↩
    ''eb'')
g38.2 z0 f15 (b''вв''b''ив''b''мв''b''iв''b''pb''b''юв''b''вв''b''ab''b''нв''b''нв''b' ↩
    'яв'')
g91 g0z.2 (b''вв''b''ив''b''мв''b''кв''b''нв''b''yв''b''тв''b''ив'' b''пв''b''eb''b''pb''b' ↩
    'eb''b''мв''b''ив''b''кв''b''ab''b''чв'')
#1000=#5063 (b''зв''b''бв''b''eb''b''pb''b''eb''b''гв''b''тв''b''ив'' b''дв''b''ов''b''вв'' ↩
    b''жв''b''ив''b''нв''b''yв'' b''iв''b''нв''b''cв''b''тв''b''pb''b''yв''b''мв''b''eb''b' ↩
    'нв''b''тв''b''yв'' b''вв''b''iв''b''дв''b''лв''b''iв''b''кв''b''yв'')
(b''дв''b''pb''b''yв''b''кв'', b''дв''b''ов''b''вв''b''жв''b''ив''b''нв''b''ab'' b''вв''b' ↩
    'iв''b''дв''b''лв''b''iв''b''кв''b''yв'' b''cв''b''тв''b''ab''b''нв''b''ов''b''вв''b' ↩
    'ив''b''тв''b''ьв'' #1000)
;
O<restore_abs> b''яв''b''кв''b''щв''b''ов'' [#<absolute>]
    g90 (b''вв''b''iв''b''дв''b''нв''b''ов''b''вв''b''ив''b''тв''b''ив'' G90, b''тв''b' ↩
        'iв''b''лв''b''ьв''b''кв''b''ив'' b''яв''b''кв''b''щв''b''ов'' b''вв''b''iв''b''нв'' ↩
         b''бв''b''yв''b''вв'' b''вв''b''cв''b''тв''b''ab''b''нв''b''ов''b''вв''b''лв''b' ↩
        'eb''b''нв''b''ив''b''йв'' b''пв''b''pb''b''ив'' b''вв''b''xв''b''ов''b''дв''b' ↩
        'iв'':)
O<restore_abs> endif
;
O<measure> endsub
```

11.6.24 Команди, визначені користувачем M100-M199

```
M1-- <P- Q->
```

- *M1--* - ціле число в діапазоні від 100 до 199.
- *P-* - число, що передається до файлу як перший параметр.
- *Q-* - число, що передається до файлу як другий параметр.

Note
Після створення нового файлу «M1nn» необхідно перезапустити графічний інтерфейс, щоб він був у курсі нового файлу, інакше ви отримаєте помилку «Невідомий m-код».

Зовнішня програма з іменем від «M100» до «M199» (без розширення, велика літера M, знаходиться в каталозі, вказаному параметром «[DISPLAY] PROGRAM_PREFIX» файлу INI) виконується з опціональн значеннями P і Q як двома аргументами.

Виконання файлу G-коду призупиняється до виходу зовнішньої програми. Якщо зовнішня програма виходить з кодом виходу, відмінним від «0», виконання програми G-коду зупиняється. Можна використовувати будь-який дійсний виконуваний файл. Файл повинен знаходитися в шляху пошуку, вказаному в конфігурації файлу INI. Докладнішу інформацію про шляхи пошуку див. у розділі Display.

Після створення нової програми M1*nn* графічний інтерфейс користувача слід перезапустити, щоб нова програма враховувалася, інакше виникне помилка «Невідомий M-код».



Warning
Не використовуйте текстовий редактор для створення або редагування файлів. Текстовий редактор залишає невидимі коди, які можуть спричинити проблеми та завадити роботі файлів bash або python. Для створення або редагування файлів використовуйте текстовий редактор, наприклад Geany в Debian або Notepad++ в інших операційних системах.

Помилка «Використано невідомий M-код» означає одну з наступних ситуацій:

- Вказана користувацька команда не існує.
- Файл не є виконуваним файлом.
- Ім’я файлу має розширення.
- Ім’я файлу не відповідає цьому формату Mnnn, де nnn = від 100 до 199.
- В назві файлу використовувалася мала літера M.

Наприклад, щоб відкрити і закрити затискач, який керується контактом паралельного порту, використовуючи файл скрипта bash з використанням M101 і M102. Створіть два файли з іменами M101 і M102. Перед запуском LinuxCNC встановіть їх як виконувані файли (зазвичай це робиться правою кнопкою миші/властивості/дозволи). Переконайтеся, що контакт паралельного порту не підключений до нічого в файлі HAL.

Приклад файлу M101

```
#!/bin/bash
# b’’фb’’b’’ab’’b’’йb’’b’’лb’’ b’’дb’’b’’лb’’b’’яb’’ b’’вb’’b’’вb’’b’’іb’’b’’мb’’b’’кb’’b’ ↩
    ’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’кb’’b’’оb’’b’’нb’’b’’тb’’b’’аb’’b’’кb’’b’’тb’’b’ ↩
    ’уb’’ 14 b’’цb’’b’’аb’’b’’нb’’b’’гb’’b’’оb’’b’’вb’’b’’оb’’b’’гb’’b’’оb’’ b’’зb’’b’’аb’’b ↩
    ’’тb’’b’’вb’’b’’оb’’b’’pb’’b’’ab’’ parport
halcmd setp parport.0.pin-14-out True
exit 0
```

Приклад файлу M102

```
#!/bin/bash
# b’’нb’’b’’аb’’b’’пb’’b’’иb’’b’’лb’’b’’оb’’b’’кb’’ b’’дb’’b’’лb’’b’’яb’’ b’’вb’’b’’иb’’b’ ↩
    ’мb’’b’’кb’’b’’нb’’b’’еb’’b’’нb’’b’’нb’’b’’яb’’ b’’шb’’b’’тb’’b’’иb’’b’’фb’’b’’тb’’b’ ↩
    ’ab’’ b’’пb’’b’’ab’’b’’pb’’b’’пb’’b’’оb’’b’’pb’’b’’тb’’b’’ab’’ 14 b’’дb’’b’’лb’’b’’яb’’  ↩
    b’’вb’’b’’іb’’b’’дb’’b’’кb’’b’’pb’’b’’иb’’b’’тb’’b’’тb’’b’’яb’’ b’’цb’’b’’аb’’b’’нb’’b’ ↩
    ’гb’’b’’оb’’b’’вb’’b’’оb’’b’’гb’’b’’оb’’ b’’зb’’b’’аb’’b’’кb’’b’’pb’’b’’иb’’b’’вb’’b’ ↩
    ’ab’’b’’чb’’b’’ab’’
halcmd setp parport.0.pin-14-out False
exit 0
```

Щоб передати змінну до файлу M1nn, ви використовуєте опції P та Q ось так:

```
M100 P123.456 Q321.654
```

Приклад файлу M100

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

Щоб відобразити графічне повідомлення та зупинити його до закриття вікна повідомлення, використов
програму графічного відображення, таку як Eye of Gnome, для відображення графічного файлу.
Після закриття програма відновить роботу.

Приклад файлу M110

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

Щоб відобразити графічне повідомлення та продовжити обробку файлу G-коду, додайте до команди амперсанд.

Приклад відображення M110 та продовження роботи

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

11.7 Коди O

11.7.1 Використання O-кодів

O-коди забезпечують контроль потоку в NC-програмах. Кожен блок має відповідний номер, який використовується після O. Необхідно ретельно підбирати O-номери. O-коди використовують букву «O», а не цифру нуль як перший символ у номері, наприклад O100 або o100. O-коди також іноді називають o-словами, і ці терміни є взаємозамінними.

Note
Використання малої літери «o» полегшує відмінність від 0, який, можливо, був надрукований з помилкою. Наприклад, «o100» легше побачити, ніж «O100», що це не «0».

11.7.2 Нумерація

Нумеровані o-коди повинні мати унікальний номер для кожної підпрограми,

Приклад нумерації

```
(b''пb''b''ob''b''чb''b''ab''b''тb''b''ob''b''кb'' o100)
o100 sub
(b''зb''b''вb''b''eb''b''pb''b''нb''b''ib''b''тb''b''ьb'' b''yb''b''вb''b''ab''b''гb''b' ↩
    'yb'', b''щb''b''ob'' b''бb''b''лb''b''ob''b''кb'' if-endif b''вb''b''иb''b''кb''b''ob'' ↩
    b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb'' b''ib''b''нb''b''шb''b''иb''b' ↩
    'йb'' b''нb''b''ob''b''мb''b''eb''b''pb'')
  (b''пb''b''ob''b''чb''b''ab''b''тb''b''ob''b''кb'' o110)
  o110 if [#2 GT 5]
    (b''тb''b''yb''b''тb'' b''яb''b''кb''b''иb''b''йb''b''cb''b''ьb'' b''кb''b''ob''b' ↩
        'дb'')
  (b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' o110)
  o110 endif
  (b''тb''b''yb''b''тb'' b''щb''b''eb'' b''яb''b''кb''b''иb''b''йb''b''cb''b''ьb'' b''кb''b ↩
      ''ob''b''дb'')
(b''кb''b''ib''b''нb''b''eb''b''цb''b''ьb'' o100)
o100 endsub
```

11.7.3 Коментарі

Коментарі в тому ж рядку, що й слово на літеру «o», не слід використовувати, оскільки така поведінка може змінитися в майбутньому.

Поведінка невизначена, якщо:

- Один і той самий номер використовується для кількох блоків.
- Інші слова використовуються в рядку зі словом на літеру «o».
- Коментарі використовуються в рядку зі словом, що починається з літери «o».

11.7.4 Підпрограми

Підпрограми починаються з *oNNN sub* і закінчуються *oNNN endsub*. Рядки між *oNNN sub* і *oNNN endsub* не виконуються, поки підпрограма не буде викликана за допомогою *oNNN call*. Кожна підпрограма повинна використовувати унікальний номер.

Приклад підпрограми

```
o100 sub
  G53 G0 X0 Y0 Z0 (b''шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''пb''b''eb''b''pb''b ↩
      ''eb''b''xb''b''ib''b''дb'' b''дb''b''ob'' b''мb''b''ab''b''шb''b''иb''b''нb''b''нb''b ↩
      ''ob''b''гb''b''ob'' b''дb''b''ob''b''мb''b''yb'')
o100 endsub

(b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ab'')
o100 call
M2
```

Див. розділи G53, G0 та M2 для отримання додаткової інформації.

O- Повернення Всередині підпрограми можна виконати *o-return*. Це негайно повертає викликаючу програму, так само, як якщо б було виявлено *o-endsub*.

Приклад повернення O-

```
o100 sub
  (b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b''иb''b''тb''b''иb'', b''чb''b''иb'' b ↩
      ''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' b''№b''2 b''бb''b''ib''b''лb'' ↩
     b''ьb''b''шb''b''иb''b''йb'' b''зb''b''ab'' 5)
  o110 b''яb''b''кb''b''щb''b''ob'' [#2 GT 5]
    (b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb''b''cb''b''яb'' b''дb''b ↩
       ''ob'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''yb'' b''пb''b''ib''b''дb''b' ↩
       'пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb'', b''яb''b''кb''b''щb''b''ob'' ↩
        b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b''кb''b''ab'' b''єb'' b''ib''b' ↩
       'cb''b''тb''b''иb''b''нb''b''нb''b''ob''b''юb'')
    o100 b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb''
  o110 endif
    (b''цb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''нb''b''yb''b''єb''b''тb''b''ьb''b''cb''b ↩
       ''яb'' b''тb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''яb''b''кb''b''щb''b''ob'' b' ↩
       'пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' #2 b''нb''b''eb'' b''бb''b' ↩
       'ib''b''лb''b''ьb''b''шb''b''иb''b''йb'' b''зb''b''ab'' 5)
    (DEBUG, b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' 1 b''єb'' [#1])
o100 endsub
```

Див. розділи Бінарні оператори та Параметри для отримання додаткової інформації.

O- Виклик *o- Call* приймає до 30 необов'язкових аргументів, які передаються до підпрограми як *#1*, *#2* , ..., #N. Параметри від #N+1 до #30 мають те саме значення, що і в контексті виклику. Після повернення з підпрограми значення параметрів від #1 до #30 (незалежно від кількості аргументів) будуть відновлені до значень, які вони мали до виклику. Параметри #1 - #30 є локальними для підпрограми.

Оскільки *1 2 3* розбирається як число 123, параметри мають бути взяті в квадратні дужки. Наступна команда викликає підпрограму з 3 аргументами:

Приклад виклику O

```
o100 sub
  (b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b''иb''b''тb''b''иb'', b''чb''b''иb'' b ↩
      ''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' b''№b''2 b''бb''b''ib''b''лb'' ↩
     b''ьb''b''шb''b''иb''b''йb'' b''зb''b''ab'' 5)
  o110 if [#2 GT 5]
```

```
    (b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb''b''cb''b''яb'' b''дb''b ↩
        ''ob'' b''пb''b''ob''b''чb''b''ab''b''тb''b''кb''b''yb'' b''пb''b''іb''b''дb''b' ↩
        'пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb'', b''яb''b''кb''b''щb''b''ob'' ↩
         b''пb''b''eb''b''pb''b''eb''b''вb''b''іb''b''pb''b''кb''b''ab'' b''cb''b''пb''b' ↩
        'pb''b''ab''b''вb''b''жb''b''нb''b''яb'')
    o100 b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''яb''
  o110 endif
    (b''цb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''нb''b''yb''b''єb''b''тb''b''ьb''b''cb''b ↩
        ''яb'' b''тb''b''іb''b''лb''b''ьb''b''кb''b''иb'' b''яb''b''кb''b''щb''b''ob'' b' ↩
        'пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' b''№b''2 b''нb''b''eb'' b' ↩
        'бb''b''іb''b''лb''b''ьb''b''шb''b''иb''b''йb'' b''зb''b''ab'' 5)
    (DEBUG, b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' 1 b''єb'' [b''№b''1])
    (DEBUG, b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' 3 b''єb'' [b''№b''3])
o100 endsub

o100 call [100] [2] [325]
```

Тіла підпрограм не можуть бути вкладеними. Вони можуть бути викликані тільки після того, як вони визначені. Вони можуть бути викликані з інших функцій і можуть викликати себе рекурсивно, якщо це має сенс. Максимальний рівень вкладення підпрограм становить 10.

Підпрограми можуть змінювати значення параметрів вище #30, і ці зміни будуть видимі для викличного коду. Підпрограми також можуть змінювати значення global named parameters (тобто параметрів, імена яких починаються з символу підкреслення "_").

11.7.4.1 Нумеровані програми в стилі Fanuc

Пронумеровані програми (як основні, так і підпрограми), виклик «M98» і повернення «M99» M-кодів, а також їх відповідні семантичні відмінності є альтернативою описаним вище підпрограмам rs274ngc, що забезпечують сумісність з Fanuc та іншими контролерами верстатів.

Нумеровані програми ввімкнено за замовчуванням, і їх можна вимкнути, додавши `DISABLE_FANUC_STYL = 1` до розділу `[RS274NGC]` файлу `.ini`.

Note
Пронумеровані визначення та виклики основних і підпрограм відрізняються від традиційних rs274ngc як за синтаксисом, так і за виконанням. Щоб зменшити ймовірність плутанини, інтерпретатор видасть помилку, якщо визначення одного стилю будуть змішані з викликами іншого.

Простий приклад нумерованої підпрограми

```
o1 (b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' 1)    ; b''Ob''b''cb''b''нb''b''ob''b ↩
    ''вb''b''нb''b''ab'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''ab'' 1, «b' ↩
    'Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb'' 1»
M98 P100          ; b''Bb''b''иb''b''кb''b''лb''b''иb''b''кb'' b''пb''b''іb''b''дb''b''пb'' ↩
    b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb'' 100
M30               ; b''Kb''b''іb''b''нb''b''eb''b''цb''b''ьb'' b''ob''b''cb''b''нb''b''ob'' ↩
    b''вb''b''нb''b''ob''b''їb'' b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb''

o100              ; b''Пb''b''ob''b''чb''b''ab''b''тb''b''ob''b''кb'' b''пb''b''іb''b''дb'' ↩
    b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb'' 100
  G53 G0 X0 Y0 Z0 ; b''Шb''b''вb''b''иb''b''дb''b''кb''b''иb''b''йb'' b''пb''b''eb''b''pb'' ↩
      b''eb''b''xb''b''іb''b''дb'' b''дb''b''ob'' b''вb''b''иb''b''xb''b''іb''b''дb''b''нb'' ↩
      b''ob''b''їb'' b''пb''b''ob''b''зb''b''иb''b''цb''b''іb''b''їb'' b''вb''b''eb''b''pb'' ↩
      b''cb''b''тb''b''ab''b''тb''b''ab''
M99               ; b''Пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''яb''  ↩
    b''зb'' b''пb''b''іb''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb''  ↩
    100
```

o1 (Назва) Опціональний початковий блок основної програми присвоює основній програмі номер «1». Деякі контролери трактують опціональний коментар у дужках як назву програми, в даному прикладі «Приклад 1», але це не має особливого значення в інтерпретаторі rs274ngc.

M98 P- <L-> Викликати підпрограму з номером. Блок `M98 P100` аналогічний традиційній синтаксичній конструкції `o100 call`, але може використовуватися тільки для виклику наступної підпрограми з номером, визначеної за допомогою `o100`...`M99`. Опціональне слово *L* визначає кількість циклів.

M30 Головну програму необхідно завершити командою `M02` або `M30` (або `M99`; див. нижче).

Початок визначення підпрограми O- Позначає початок нумерованого визначення підпрограми. Блок `o100` подібний до `o100 sub`, за винятком того, що його потрібно розмістити у файлі пізніше, ніж блок виклику `M98 P100`.

Повернення M99 з нумерованої підпрограми Блок `M99` аналогічний традиційному синтаксису `o100 endsub`, але може завершувати тільки пронумеровану програму (`o100` у цьому прикладі) і не може завершувати підпрограму, що починається з синтаксису `o100 sub`.

Виклик підпрограми `M98` відрізняється від виклику `o` rs274ngc наступним чином:

- Пронумерована підпрограма повинна йти після виклику `M98` у файлі програми. Інтерпретатор викличе помилку, якщо підпрограма передує блоку виклику.
- Параметри `#1`, `#2`, ..., `#30` є глобальними і доступними в пронумерованих підпрограмах, подібно до параметрів з вищими номерами в традиційних викликах. Зміни цих параметрів у підпрограмі є глобальними і зберігаються після повернення підпрограми.
- `M98` Виклики підпрограм не мають повертаного значення.
- Блоки виклику підпрограми `M98` можуть містити опціональне L-слово, що визначає кількість повторень циклу. Без L-слова підпрограма буде виконана тільки один раз (еквівалентно `M98 L1`). Блок `M98 L0` не буде виконувати підпрограму.

У рідкісних випадках M-код `M99` може використовуватися для завершення основної програми, де він позначає «нескінченну програму». Коли інтерпретатор досягає `M99` в основній програмі, він повертається до початку файлу і продовжує виконання з першого рядка. Прикладом використання нескінченної програми є цикл розігріву машини; блок видалення програми end `/M30` може бути використаний для зупинки циклу в зручний момент, коли оператор готовий.

Повний приклад нумерованої підпрограми

```
o1                              ; b''Ob''b''cb''b''нb''b''ob''b''вb''b''нb''b''ab'' b''пb''b ↩
    ''pb''b''ob''b''rb''b''pb''b''ab''b''мb''b''ab'' 1
  #1 = 0
  (PRINT,X MAIN BEGIN:  1=#1)
  M98 P100 L5                   ; b''Bb''b''иb''b''кb''b''лb''b''иb''b''кb'' b''пb''b''ib''b ↩
      ''дb''b''пb''b''pb''b''ob''b''rb''b''pb''b''ab''b''мb''b''иb'' 100
  (PRINT,X MAIN END:  1=#1)
M30                             ; b''3b''b''ab''b''вb''b''eb''b''pb''b''шb''b''иb''b''тb''b' ↩
    'иb'' b''ob''b''cb''b''нb''b''ob''b''вb''b''нb''b''yb'' b''пb''b''pb''b''ob''b''rb''b' ↩
    'pb''b''ab''b''мb''b''yb''

o100                            ; b''Пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''rb''b''pb''b' ↩
    'ab''b''мb''b''ab'' 100
  #1 = [#1 + 1]
  M98 P200 L5                   ; b''Bb''b''иb''b''кb''b''лb''b''иb''b''кb'' b''пb''b''ib''b ↩
      ''дb''b''пb''b''pb''b''ob''b''rb''b''pb''b''ab''b''мb''b''иb'' 200
  (PRINT,>> o100:  #1)
M99                             ; b''Пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''зb'' b''пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''rb''b''pb''b''ab''b' ↩
    'мb''b''иb'' 100
```

```
o200                            ; b''Пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b' ↩
    'ab''b''мb''b''ab'' 200
  #1 = [#1 + 0.01]
  (PRINT,>>>> o200:  #1)
M99                             ; b''Пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''зb'' b''пb''b''ib''b''дb''b''пb''b''pb''b''ob''b''гb''b''pb''b''ab''b' ↩
    'мb''b''иb'' 200
```

У цьому прикладі параметр #1 ініціалізується значенням 0. Підпрограма o100 викликається п’ять разів у циклі. Вбудована в кожен виклик o100, підпрограма o200 викликається п’ять разів у циклі, загалом 25 разів.

Зверніть увагу, що параметр #1 є глобальним. В кінці основної програми, після оновлень у межах o100 та o200, його значення дорівнюватиме 5.25.

11.7.5 Цикл

Цикл «while» має дві структури: «while/endwhile» і «do/while». У кожному випадку цикл завершується, коли умова «while» оцінюється як хибна. Різниця полягає в тому, коли виконується умова перевірки. Цикл «do/while» виконує код у циклі, а потім перевіряє умову перевірки. Цикл «while/endwhile» спочатку виконує перевірку.

Приклад завершення в той час як

```
(b''нb''b''ab''b''мb''b''ab''b''лb''b''юb''b''йb''b''тb''b''eb'' b''фb''b''ib''b''гb''b' ↩
    'уb''b''pb''b''уb'' b''уb'' b''вb''b''иb''b''гb''b''лb''b''яb''b''дb''b''ib'' b''пb''b' ↩
    'иb''b''лb''b''кb''b''иb'')
G0 X1 Y0 (b''пb''b''eb''b''pb''b''eb''b''мb''b''ib''b''cb''b''тb''b''ib''b''тb''b''ьb''b' ↩
    'cb''b''яb'' b''уb'' b''вb''b''иb''b''хb''b''ib''b''дb''b''нb''b''eb'' b''пb''b''ob''b' ↩
    'лb''b''ob''b''жb''b''eb''b''нb''b''нb''b''яb'')
#1 = 0 (b''пb''b''pb''b''иb''b''cb''b''вb''b''ob''b''йb''b''тb''b''eb'' b''пb''b''ab''b' ↩
    'pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''уb'' #1 b''зb''b''нb''b''ab''b''чb''b''eb''b' ↩
    'нb''b''нb''b''яb'' 0)
F25 (b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''ib''b''тb''b''ьb'' b''шb''b''вb''b ↩
    ''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b' ↩
    'ib'')
o101 while [#1 LT 10]
  G1 X0
  G1 Y[#1/10] X1
  #1 = [#1+1] (b''зb''b''бb''b''ib''b''лb''b''ьb''b''шb''b''иb''b''тb''b''иb'' b''лb''b' ↩
      'ib''b''чb''b''иb''b''лb''b''ьb''b''нb''b''иb''b''кb'' b''тb''b''eb''b''cb''b''тb''b' ↩
      'ib''b''вb'')
o101 endwhile
M2 (b''зb''b''ab''b''вb''b''eb''b''pb''b''шb''b''иb''b''тb''b''иb'' b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''уb'')
```

Приклад виконання While

```
#1 = 0 (b''пb''b''pb''b''иb''b''cb''b''вb''b''ob''b''їb''b''тb''b''иb'' b''пb''b''ab''b' ↩
    'pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''уb'' #1 b''зb''b''нb''b''ab''b''чb''b''eb''b' ↩
    'нb''b''нb''b''яb'' 0)
o100 do
  (debug, b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb'' 1 = #1)
  o110 if [#1 EQ 2]
    #1 = 3 (b''пb''b''pb''b''иb''b''cb''b''вb''b''ob''b''їb''b''тb''b''иb'' b''пb''b''ab''b ↩
        ''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''уb'' #1 b''зb''b''нb''b''ab''b''чb''b' ↩
        'eb''b''нb''b''нb''b''яb'' 3)
    (msg, #1 b''пb''b''pb''b''иb''b''cb''b''вb''b''ob''b''єb''b''нb''b''ob'' b''зb''b''нb'' ↩
        b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' 3)
    o100 continue (b''пb''b''eb''b''pb''b''eb''b''йb''b''тb''b''иb'' b''дb''b''ob'' b''пb'' ↩
        b''ob''b''чb''b''ab''b''тb''b''кb''b''уb'' b''цb''b''иb''b''кb''b''лb''b''уb'')
```

```
  o110 endif
  (b''тb''b''уb''b''тb'' b''яb''b''кb''b''иb''b''йb''b''сb''b''ьb'' b''кb''b''оb''b''дb'')
  #1 = [#1 + 1] (b''зb''b''бb''b''іb''b''лb''b''ьb''b''шb''b''иb''b''тb''b''иb'' b''лb''b' ↩
      'іb''b''чb''b''иb''b''лb''b''ьb''b''нb''b''иb''b''кb'' b''тb''b''еb''b''сb''b''тb''b' ↩
      'уb'')
o100 while [#1 LT 3]
(msg, b''Цb''b''иb''b''кb''b''лb'' b''зb''b''аb''b''вb''b''еb''b''рb''b''шb''b''еb''b''нb'' ↩
    b''оb''!)
M2
```

У циклі while оператор «o-break» негайно виходить із циклу, а оператор «o-continue» негайно переходить до наступної оцінки умови «while». Якщо вона все ще є істинною, цикл починається знову з початку. Якщо вона є хибною, цикл завершується.

11.7.6 Умовний

Умова «if» складається з групи операторів з однаковим номером «o», які починаються з «if» і закінчуються «endif». Опціональні умови «elseif» і «else» можуть бути між початковим «if» і кінцевим «endif».

Якщо умова «if» має значення true, то виконується група операторів, що йдуть після «if» до наступного рядка умови.

Якщо умова «if» оцінюється як хибна, то умови «elseif» оцінюються по порядку, поки одна з них не буде оцінена як істинна. Якщо умова «elseif» є істинною, то виконуються оператори, що йдуть за «elseif» аж до наступного умовного рядка. Якщо жодна з умов «if» або «elseif» не оцінюється як істинна, то виконуються оператори, що йдуть за «else». Коли умова оцінюється як істинна, інші умови в групі більше не оцінюються.

Приклад Endif

```
(b''яb''b''кb''b''щb''b''оb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb'' b' ↩
    '№b''31 b''дb''b''оb''b''рb''b''іb''b''вb''b''нb''b''юb''b''єb'' 3, b''вb''b''сb''b' ↩
    'тb''b''аb''b''нb''b''оb''b''вb''b''лb''b''еb''b''нb''b''иb''b''мb'' S2000)
o101 if [#31 EQ 3]
  S2000
o101 endif
```

Якщо ElseIf Else EndIf Приклад

```
(b''яb''b''кb''b''щb''b''оb'' b''пb''b''аb''b''рb''b''аb''b''мb''b''еb''b''тb''b''рb'' b' ↩
    '№b'' 2 b''бb''b''іb''b''лb''b''ьb''b''шb''b''иb''b''йb'' b''зb''b''аb'' 5, b''вb''b' ↩
    'сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' F100)
o102 b''яb''b''кb''b''щb''b''оb'' [b''№b'' 2 GT 5]
  F100
o102 b''іb''b''нb''b''аb''b''кb''b''шb''b''еb'' b''яb''b''кb''b''щb''b''оb'' [b''№b'' 2 LT  ↩
    2]
(b''іb''b''нb''b''аb''b''кb''b''шb''b''еb'', b''яb''b''кb''b''щb''b''оb'' b''пb''b''аb''b' ↩
    'рb''b''аb''b''мb''b''еb''b''тb''b''рb'' b''№b'' 2 b''мb''b''еb''b''нb''b''шb''b''иb''b' ↩
    'йb'' b''зb''b''аb'' 2, b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b ↩
    ''иb'' F200)
  F200
(b''іb''b''нb''b''аb''b''кb''b''шb''b''еb'', b''яb''b''кb''b''щb''b''оb'' b''пb''b''аb''b' ↩
    'рb''b''аb''b''мb''b''еb''b''тb''b''рb'' b''№b'' 2 b''сb''b''тb''b''аb''b''нb''b''оb''b' ↩
    'вb''b''иb''b''тb''b''ьb'' b''вb''b''іb''b''дb'' 2 b''дb''b''оb'' 5, b''вb''b''сb''b' ↩
    'тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' F150)
o102 else
  F150
o102 endif
```

Кілька умов можна перевірити за допомогою операторів *elseif*, доки шлях *else* нарешті не буде виконано, якщо всі попередні умови хибні:

Приклад If Elseif Else Endif

```
(b''яв''b''кв''b''щв''b''ов'' b''пв''b''ав''b''рв''b''ав''b''мв''b''ев''b''тв''b''рв'' b' ↩
    '№b'' 2 b''бв''b''ів''b''лв''b''ьв''b''шв''b''ив''b''йв'' b''зв''b''ав'' 5, b''вв''b' ↩
    'св''b''тв''b''ав''b''нв''b''ов''b''вв''b''ив''b''тв''b''ив'' F100)
o102 b''яв''b''кв''b''щв''b''ов'' [b''№b'' 2 GT 5]
  F100
(b''ів''b''нв''b''ав''b''кв''b''шв''b''ев'', b''яв''b''кв''b''щв''b''ов'' b''пв''b''ав''b' ↩
    'рв''b''ав''b''мв''b''ев''b''тв''b''рв'' b''№b'' 2 b''мв''b''ев''b''нв''b''шв''b''ев''  ↩
    2, b''вв''b''св''b''тв''b''ав''b''нв''b''ов''b''вв''b''ив''b''тв''b''ив'' F200)
o102 elseif [b''№b'' 2 LT 2]
  F20
(b''пв''b''ав''b''рв''b''ав''b''мв''b''ев''b''тв''b''рв'' b''№b'' 2 b''зв''b''нв''b''ав''b' ↩
    'хв''b''ов''b''дв''b''ив''b''тв''b''ьв''b''св''b''яв'' b''вв'' b''дв''b''ів''b''ав''b' ↩
    'пв''b''ав''b''зв''b''ов''b''нв''b''ів'' b''вв''b''ів''b''дв'' 2 b''дв''b''ов'' 5)
o102 else
  F200
o102 endif
```

11.7.7 Повторити

Команда «repeat» виконає оператори всередині repeat/endrepeat задану кількість разів. У цьому прикладі показано, як можна фрезерувати діагональний ряд фігур, починаючи з поточної позиції.

Приклад з «повтором»

```
(b''Фв''b''рв''b''ев''b''зв''b''ев''b''рв''b''ув''b''вв''b''ав''b''нв''b''нв''b''яв'' 5 b' ↩
    'дв''b''ів''b''ав''b''гв''b''ов''b''нв''b''ав''b''лв''b''ьв''b''нв''b''ив''b''хв'' b' ↩
    'фв''b''ов''b''рв''b''мв'')
G91 (b''Ів''b''нв''b''кв''b''рв''b''ев''b''мв''b''ев''b''нв''b''тв''b''ав''b''лв''b''ьв''b' ↩
    'нв''b''ив''b''йв'' b''рв''b''ев''b''жв''b''ив''b''мв'')
o103 b''пв''b''ов''b''вв''b''тв''b''ов''b''рв''b''ев''b''нв''b''нв''b''яв'' [5]
... (b''вв''b''св''b''тв''b''ав''b''вв''b''ив''b''тв''b''ив'' b''кв''b''ов''b''дв'' b''фв'' ↩
    b''рв''b''ев''b''зв''b''ев''b''рв''b''ув''b''вв''b''ав''b''нв''b''нв''b''яв'' b''тв''b' ↩
    'ув''b''тв'')
G0 X1 Y1 (b''дв''b''ів''b''ав''b''гв''b''ов''b''нв''b''ав''b''лв''b''ьв''b''нв''b''ев'' b' ↩
    'пв''b''ев''b''рв''b''ев''b''мв''b''ів''b''щв''b''ев''b''нв''b''нв''b''яв'' b''дв''b' ↩
    'ов'' b''нв''b''ав''b''св''b''тв''b''ув''b''пв''b''нв''b''ов''b''їв'' b''пв''b''ов''b' ↩
    'зв''b''ив''b''цв''b''ів''b''їв'')
o103 b''кв''b''ів''b''нв''b''цв''b''ев''b''вв''b''ев'' b''пв''b''ов''b''вв''b''тв''b''ов''b ↩
    ''рв''b''ев''b''нв''b''нв''b''яв''
G90 (b''Ав''b''бв''b''св''b''ов''b''лв''b''юв''b''тв''b''нв''b''ив''b''йв'' b''рв''b''ев''b ↩
    ''жв''b''ив''b''мв'')
```

11.7.8 Непрямий

o-число може бути задане параметром та/або розрахунком.

Приклад непрямого втручання

```
o[#101+2] call
```

Обчислення значень у словах на літеру «O» Щоб отримати додаткові відомості про обчислення значень, див. наступні розділи:

- Parameters

- Expressions
- Бінарні оператори
- Functions

11.7.9 Виклик файлів

Щоб викликати окремий файл з іменем підпрограми, файл повинен мати таку саму назву, як і ваш виклик, і містити sub та endsub у файлі. Файл повинен знаходитися в каталозі, на який вказує *PROGRAM_PREFIX* або *SUBROUTINE_PATH* в файлі INI. Ім'я файлу може містити тільки **малі** літери, цифри, тире та підкреслення. Файл підпрограми з іменем може містити тільки одне визначення підпрограми.

Приклад іменованого файлу

```
o<myfile> call
```

Приклад нумерованого файлу

```
o123 call
```

У викликаному файлі необхідно включити oxxx sub та endsub, а сам файл має бути коректним.

Приклад викликаного файлу

```
(b''ib''b''мb'''b''яb'' b''фb''b''ab''b''йb''b''лb''b''yb'' myfile.ngc)
o<myfile> sub
  (code here)
o<myfile> endsub
M2
```

Note
Імена файлів складаються тільки з малих літер, тому *o<MyFile>* інтерпретатор перетворює на *o<myfile>*. Більше інформації про шлях пошуку та параметри шляху пошуку наведено в розділі конфігурації INI.

11.7.10 Значення, що повертаються підпрограмою

Підпрограми можуть необов'язково повертати значення за допомогою необов'язкового виразу в операторі *endsub* або *return*.

Приклад повернутого значення

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

Значення, яке повертає підпрограма, зберігається в *<_value>* predefined named parameter , а попередньо визначений параметр *<_value_returned>* встановлюється на 1, щоб вказати, що значення було повернуто. Обидва параметри є глобальними і очищаються безпосередньо перед наступним викликом підпрограми.

11.7.11 Помилки

Наступні оператори викликають повідомлення про помилку та переривають роботу інтерпретатора:

- `return` або `endsub` поза межами визначення підрядка
- мітка для `repeat`, яка визначена в іншому місці
- мітка на `while`, яка визначена в іншому місці та не посилається на `do`
- мітка для `if`, визначена в іншому місці
- невизначена мітка для `else` або `elseif`
- мітка для `else`, `elseif` або `endif` не вказує на відповідний `if`
- мітка на `break` або `continue`, яка не вказує на відповідний `while` або `do`
- мітка на `endrepeat` або `endwhile` без посилання на відповідний `while` або `repeat`

Щоб ці помилки не були фатальними попередженнями на stderr, встановіть біт 0x20 в ініціальній опції маски `[RS274NGC]FEATURE=`.

11.8 Інші коди

11.8.1 F: Встановити швидкість подачі

«Fx» – встановити швидкість подачі на «x». «x» зазвичай вимірюється в машинних одиницях (дюймах або міліметрах) за хвилину.

Застосування швидкості подачі здійснюється відповідно до опису в розділі Feed Rate, за винятком випадків, коли діє «режим зворотної швидкості подачі» або «режим подачі на оберт», в яких швидкість подачі визначається відповідно до опису в розділі G93 G94 G95.

11.8.2 S: Встановлення швидкості шпинделя

Sx [$n] - встановити швидкість шпинделя на *x* обертів за хвилину (RPM), а додатковий $ встановити швидкість шпинделя для конкретного шпинделя. Без $ команда за замовчуванням використовуватиме spindle.0.

Шпиндель (шпинделі) або вибраний шпиндель буде обертатися з цією швидкістю, коли діє команда «M3» або «M4». Можна програмувати команду S незалежно від того, обертається шпиндель чи ні. Якщо перемикач перевищення швидкості увімкнено і не встановлено на 100 %, швидкість буде відрізнятися від запрограмованої.

Програмувати S0 можна, шпиндель не обертатиметься, якщо це зробити.

Це помилка, якщо:

- число S є від’ємним.

Як описано в розділі «gcode:g84, цикл правого нарізування з витримкою», якщо активний цикл свердління «G84» (нарізування) і потенціометри швидкості та подачі ввімкнені, буде використовуватис той, що має найнижче значення. Швидкість обертання та швидкість подачі залишатимуться синхронізованими. У цьому випадку швидкість може відрізнятися від запрограмованої, навіть якщо потенціометр корекції швидкості встановлений на 100%.

11.8.3 T: Вибрати інструмент

Tx - підготуйтеся до переходу на інструмент «x».

Інструмент не змінюється, поки не буде запрограмовано «M6» (див. розділ M6). Слово T може з'являтися в тому ж рядку, що й «M6», або в попередньому рядку. Немає нічого страшного, якщо слова T з'являються в двох або більше рядках без зміни інструменту. Тільки останнє слово T набуде чинності під час наступної зміни інструменту.

Note
Коли LinuxCNC налаштований для невипадкового змінювача інструментів (див. запис для RANDOM_TOOLCHANGER у розділі EMCIO Section), «T0» отримує спеціальне оброблення: жоден інструмент не буде вибрано. Це корисно, якщо ви хочете, щоб шпиндель був порожнім після зміни інструменту.

Note
Коли LinuxCNC налаштований для випадкового змінювача інструментів (див. запис для RANDOM_TOOLCHANGER у розділі EMCIO Section), «T0» не отримує жодного спеціального ставлення: T0 є дійсним інструментом, як і будь-який інший. Зазвичай T0 використовується на верстаті з випадковим перемикачем інструментів для відстеження порожнього гнізда, щоб він працював як верстат з невипадковим перемикачем інструментів і розвантажував шпиндель.

Це помилка, якщо:

- використовується від'ємне число T,
- Використовується номер T, якого немає у файлі таблиці інструментів (за винятком того, що T0 на невипадкових змінниках інструментів **приймається**, як зазначено вище).

На деяких верстатах карусель рухається, коли програмується слово T, одночасно з обробкою. На таких верстатах програмування слова T за кілька рядків до зміни інструменту заощадить час. Загальною практикою програмування для таких верстатів є розміщення слова T для наступного інструменту, який буде використовуватися, в рядку після зміни інструменту. Це максимізує час, доступний для руху каруселі.

Швидкі переміщення після «T<n>» не відображатимуться на попередньому перегляді AXIS до завершення переміщення подачі. Це стосується верстатів, які переміщуються на великі відстані для зміни інструменту, наприклад токарні верстати. Спочатку це може бути дуже заплутаним. Щоб вимкнути цю функцію для поточної програми інструменту, введіть G1 без будь-якого переміщення після «T<n>».

11.9 Приклади G-коду

Після встановлення LinuxCNC кілька файлів-зразків буде розміщено в папці /nc_files. Перед запуском переконайтеся, що файл-зразок підходить для вашої машини.

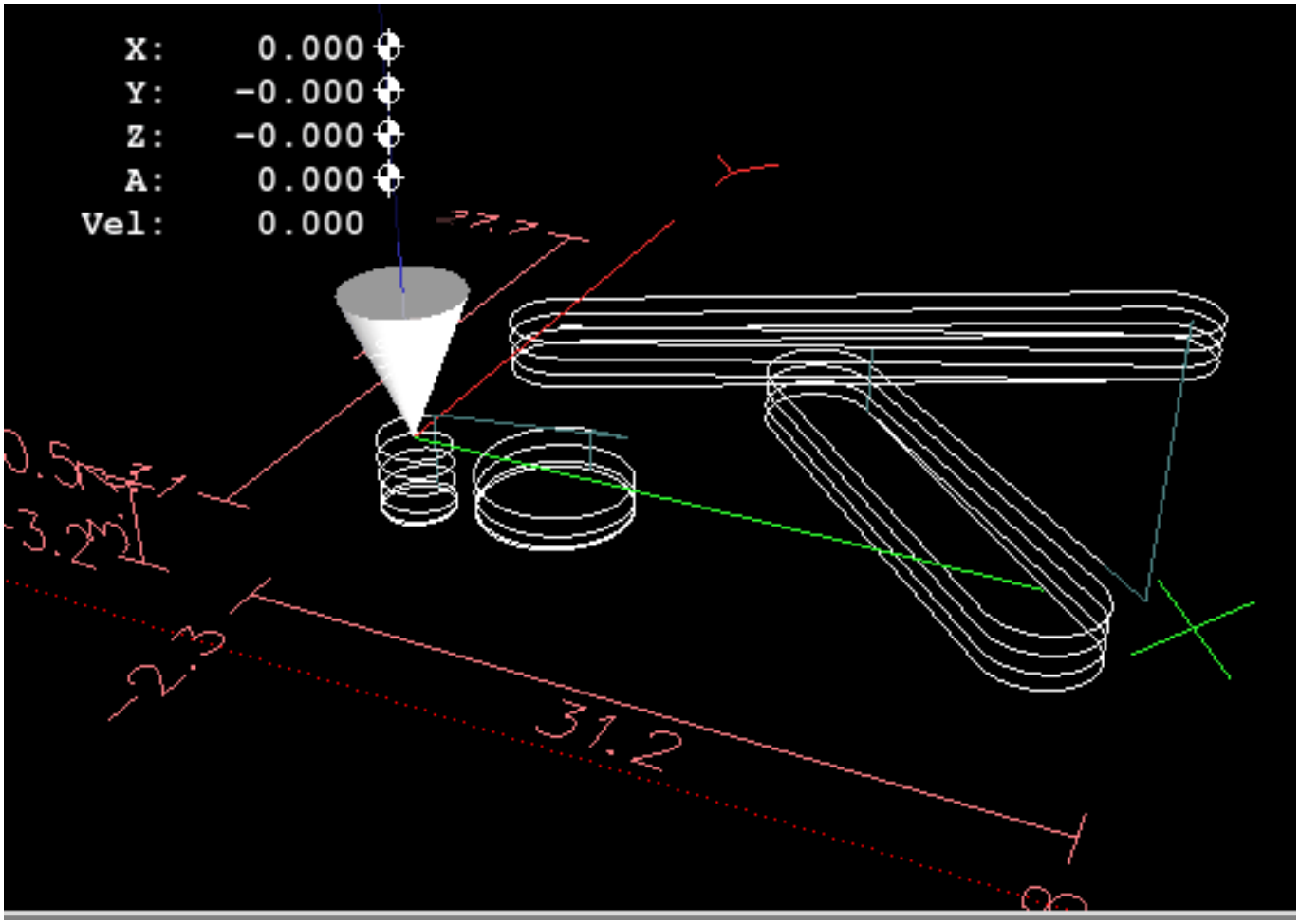
11.9.1 Приклади млинів

11.9.1.1 Фрезерування гвинтових отворів

- Ім'я файлу: useful-subroutines.ngc
- Опис: Підпрограма для фрезерування отвору з використанням параметрів.

11.9.1.2 Пазування

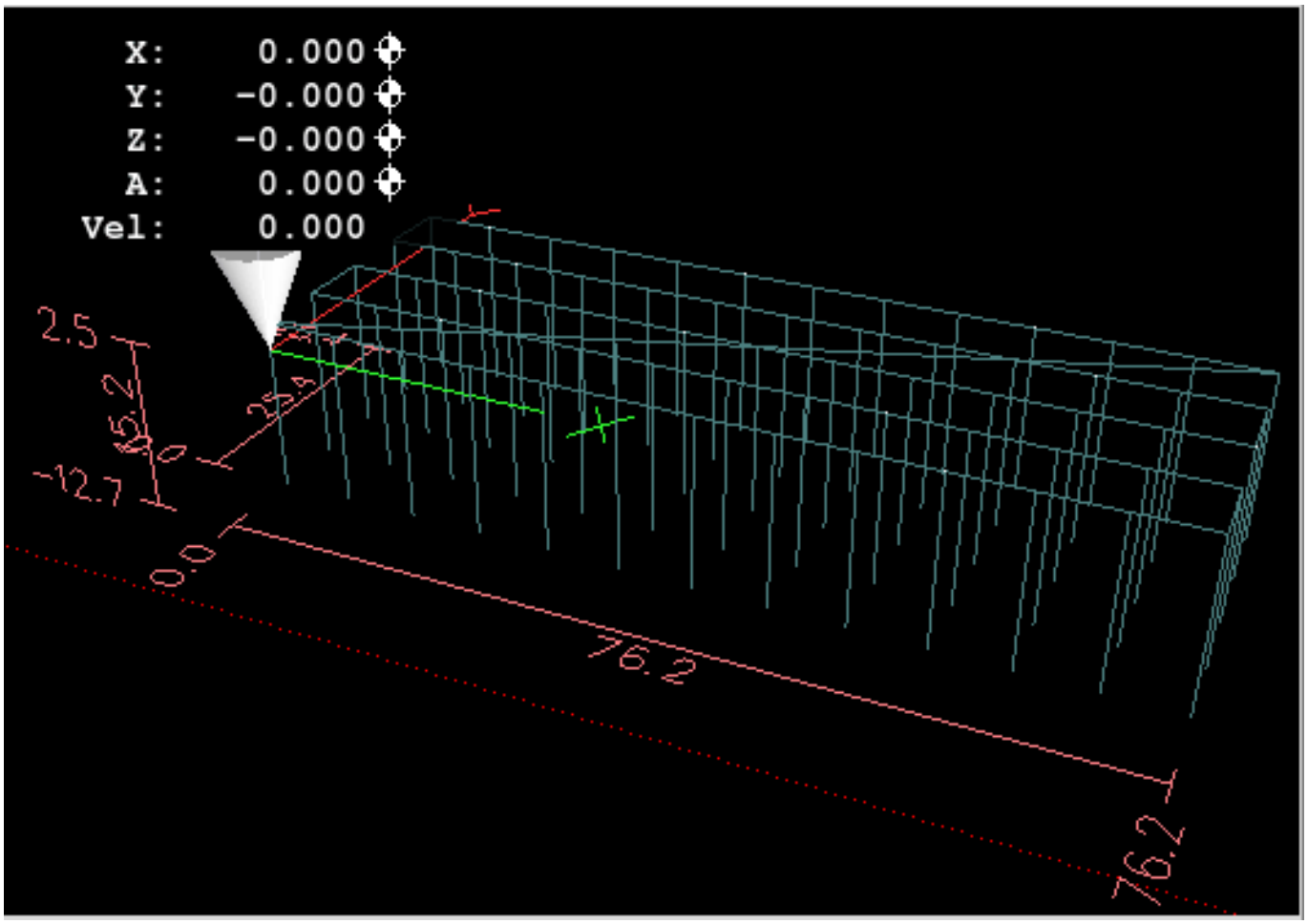
- Ім’я файлу: useful-subroutines.ngc
- Опис: Підпрограма для фрезерування паза з використанням параметрів.



11.9.1.3 Сітковий зонд

- Ім’я файлу: gridprobe.ngc
- Опис: Прямокутне зондування

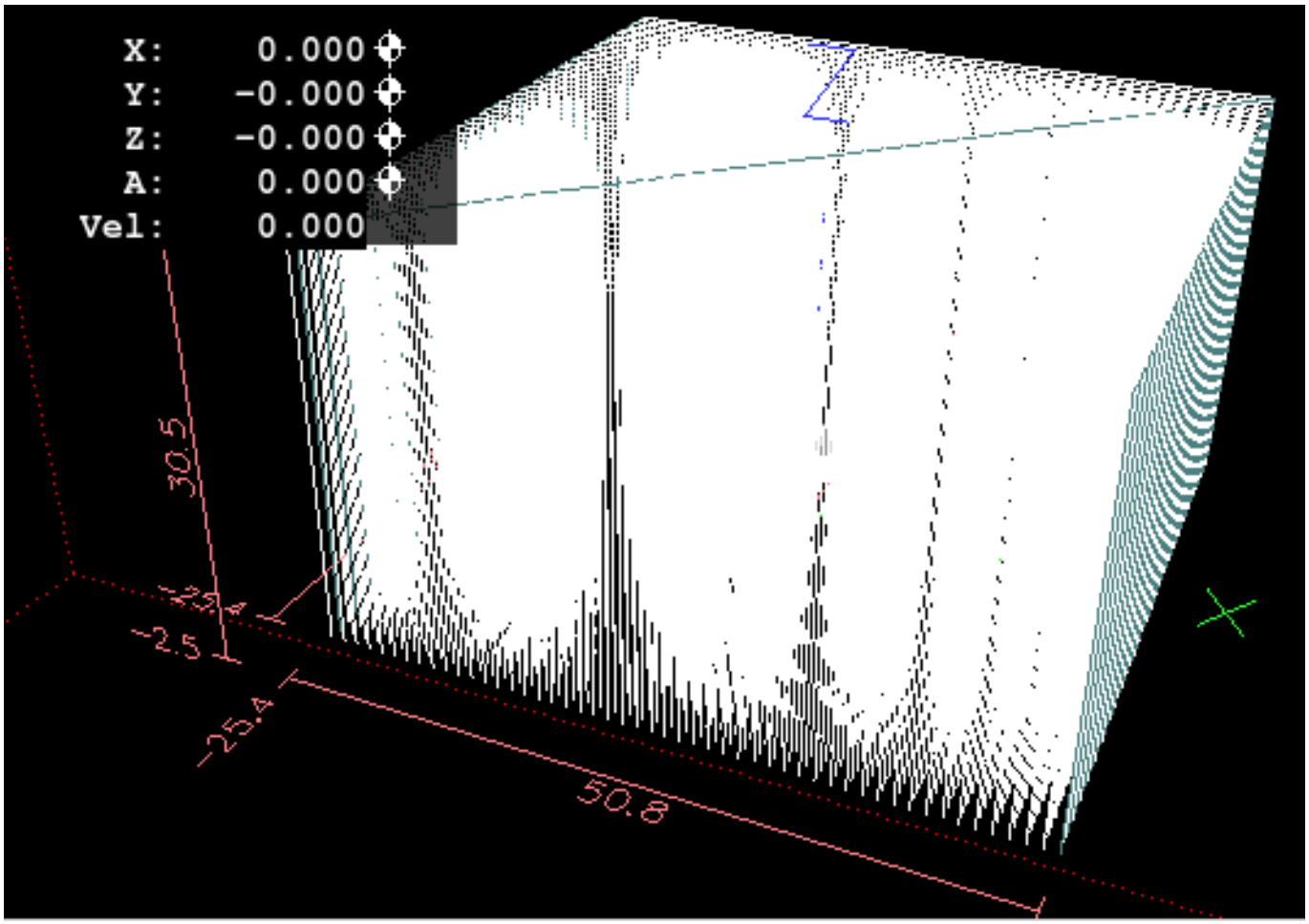
Ця програма багаторазово зондує звичайну сітку XY та записує зондовану локацію у файл *probe-results.txt* у тому ж каталозі, що й файл .ini.



11.9.1.4 Розумний зонд

- Ім’я файлу: smartprobe.ngc
- Опис: Прямокутне зондування

Ця програма багаторазово зондує звичайну сітку XY та записує зондовану локацію у файл *probe-results.txt* у тому ж каталозі, що й файл .ini. Це покращено порівняно з файлом зонду сітки.



11.9.1.5 Зонд довжини інструменту

- Ім’я файлу: tool-length-probe.ngc
- Опис: Вимірювання довжини інструменту

Ця програма показує приклад автоматичного вимірювання довжини інструменту за допомогою перемикача, підключеного до входу датчика. Це корисно для верстатів без інструментальних тримачів, де довжина інструменту змінюється при кожному вставленні.

11.9.1.6 Зонд для отворів

- Ім’я файлу: probe-hole.ngc
- Опис: Знаходження центру та діаметра отвору.

Програма демонструє, як знайти центр отвору, виміряти діаметр отвору та записати результати.

11.9.1.7 Компенсація різця

- Ім’я файлу: comp-g1.ngc
- Опис: Рухи в’їзду та виїзду з компенсацією радіуса інструменту.

Ця програма демонструє особливість траєкторії інструменту без та з компенсацією радіуса інструменту
Радіус інструменту береться з таблиці інструментів.

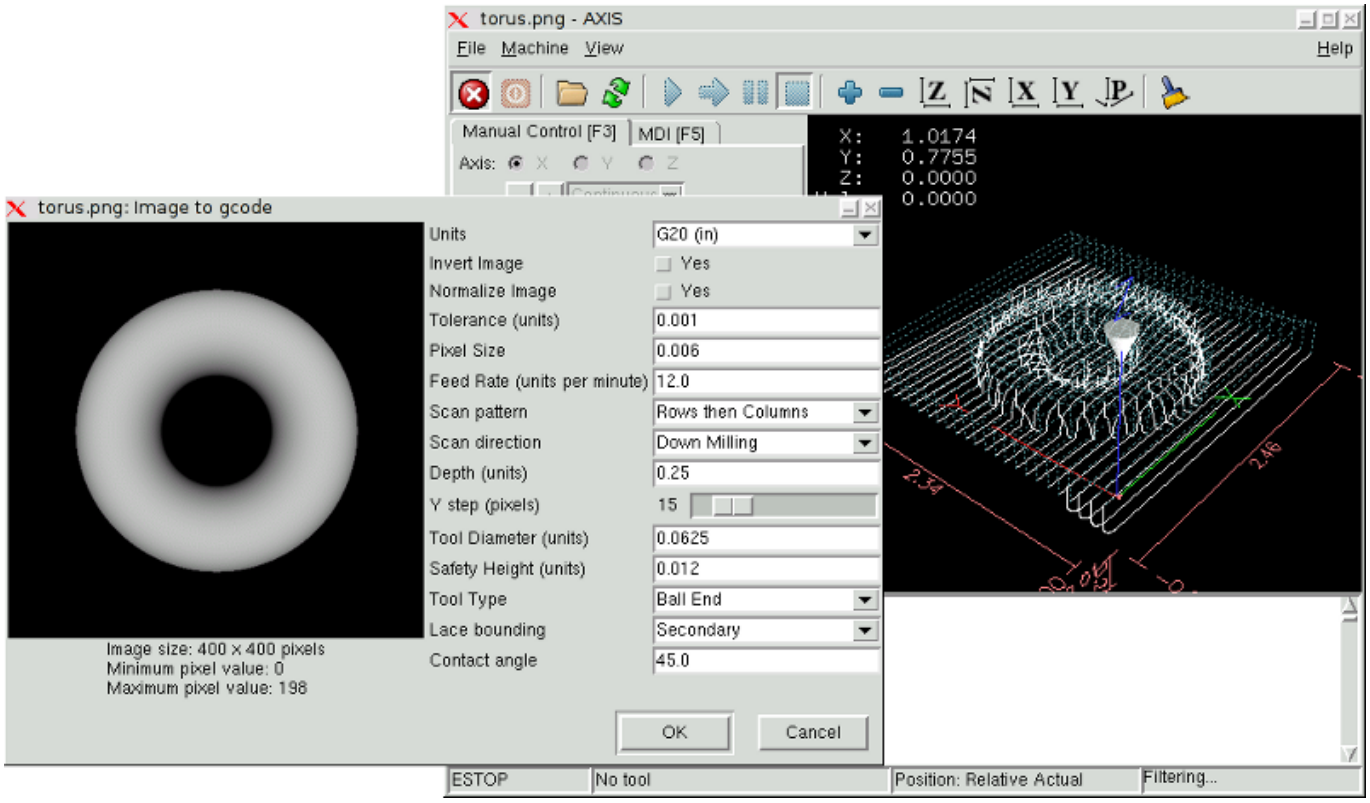
11.9.2 Приклади токарних верстатів

11.9.2.1 Різьблення

- Ім’я файлу lathe-g76.ngc
- Опис: Обробка торця, нарізання різьби та відрізка.

У цьому файлі показано приклад нарізання різьби на токарному верстаті з використанням параметрів.

11.10 Зображення в G-код



11.10.1 Що таке карта глибини?

Карта глибини — це зображення у градаціях сірого, де яскравість кожного пікселя відповідає глибині (або висоті) об’єкта в кожній точці.

11.10.2 Інтеграція зображення в G-код з інтерфейсом користувача AXIS

Додайте наступні рядки до розділу *[FILTER]* вашого INI-файлу, щоб AXIS автоматично викликав перетворення зображення на G-код під час відкриття зображення PNG, GIF або JPEG:

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

Стандартний файл конфігурації *sim/axis.ini* вже підготовлений таким чином.

11.10.3 Використання `image-to-gcode`

Запустіть перетворення зображення на G-код, відкривши файл зображення в AXIS або викликавши перетворення зображення на G-код з терміналу таким чином:

```
image-to-gcode torus.png > torus.ngc
```

Перевірте всі налаштування в правій колонці, а потім натисніть OK, щоб створити G-код. Залежно від розміру зображення та вибраних опцій, це може зайняти від декількох секунд до декількох хвилин. Якщо ви завантажуєте зображення в AXIS, G-код буде автоматично завантажений і попередньо переглянутий після завершення перетворення зображення в G-код. У AXIS натискання кнопки «Перезавантажити» знову відобразить екран опцій перетворення зображення в G-код, що дозволить вам їх налаштувати.

11.10.4 Посилання на опцію

11.10.4.1 Одиниці

Вказує, чи використовувати G20 (дюйми) чи G21 (мм) у згенерованому G-коді та як одиниці вимірювання для кожного параметра з позначкою «(одиниці)».

11.10.4.2 Інвертувати зображення

Якщо «ні», чорний піксель – це найнижча точка, а білий піксель – найвища точка. Якщо «так», чорний піксель – це найвища точка, а білий піксель – найнижча точка.

11.10.4.3 Нормалізувати зображення

Якщо «так», найтемніший піксель перепризначається на чорний, найсвітліший піксель — на білий.

11.10.4.4 Розгорнути межу зображення

Якщо «None», вхідне зображення використовується без змін, і деталі, що знаходяться на самих краях зображення, можуть бути обрізані. Якщо «White» або «Black», то з усіх боків додається рамка з пікселів, рівна діаметру інструменту, і деталі, що знаходяться на самих краях зображення, не будуть обрізані.

11.10.4.5 Толерантність (одиниці)

Коли низка точок знаходиться в межах «допустимого відхилення» від прямої лінії, вони відображаютьс як пряма лінія. Збільшення допустимого відхилення може призвести до покращення якості контурного малювання в LinuxCNC, але також може призвести до видалення або розмиття дрібних деталей зображення.

11.10.4.6 Розмір пікселя (одиниці)

Один піксель у вхідному зображенні буде дорівнювати цій кількості одиниць — зазвичай це число набагато менше 1,0. Наприклад, щоб фрезерувати об’єкт розміром 2,5x2,5 дюйма з файлу зображення розміром 400x400, використовуйте розмір пікселя 0,00625, оскільки 2,5 / 400 = 0,00625.

11.10.4.7 Швидкість подачі при зануренні (одиниць за хвилину)

Швидкість подачі для початкового руху врізання.

11.10.4.8 Швидкість подачі (одиниць за хвилину)

Швидкість подачі для інших частин траєкторії.

11.10.4.9 Швидкість шпинделя (об/хв)

S-код швидкості шпинделя, який слід ввести у файл G-коду.

11.10.4.10 Шаблон сканування

Можливі шаблони сканування:

- Рядки
- Колонки
- Рядки, потім стовпці
- Спочатку стовпці, потім рядки

11.10.4.11 Напрямок сканування

Можливі напрямки сканування:

- Позитивне: Почніть фрезерування з низького значення осі X або Y та рухайтеся до високого значення осі X або Y.
- Від’ємне: Почніть фрезерування з високого значення осі X або Y та рухайтеся до низького значення осі X або Y.
- Чергування: Почніть з того ж кінця переміщення по осі X або Y, на якому закінчився останній рух. Це зменшує кількість рухів переміщення.
- Фрезерування вгору: Почніть фрезерування з нижніх точок, рухаючись до верхніх точок.
- Фрезерування вниз: Почніть фрезерування з найвищих точок, рухаючись до найнижчих точок.

11.10.4.12 Глибина (одиниці)

Верх матеріалу завжди знаходиться на точці *Z = 0*. Найглибший розріз у матеріалі знаходиться на точці *Z = -глибина*.

11.10.4.13 Перехід (пікселі)

Відстань між сусідніми рядками або стовпцями. Щоб знайти кількість пікселів для заданої відстані в одиницях, обчисліть «відстань/розмір пікселя» і округліть до найближчого цілого числа. Наприклад, якщо «розмір пікселя = 0,006» і бажаний крок «відстань = 0,015», то використовуйте крок 2 або 3 пікселі, оскільки «0,015/0,006 = 2,5».

11.10.4.14 Діаметр інструменту

Діаметр ріжучої частини інструменту.

11.10.4.15 Безпечна висота

Висота, на яку потрібно переміститися для траверсних рухів. image-to-gcode завжди припускає, що верх матеріалу знаходиться на точці *Z=0*.

11.10.4.16 Тип інструменту

Форма ріжучої частини інструменту. Можливі форми інструменту:

- Кульовий кінець
- Плоский кінець
- 45 градусів ”vee”
- 60 градусів ”vee”

11.10.4.17 Мереживо облямівка

Це визначає, чи пропускаються області, які є відносно плоскими вздовж рядка або стовпця. Цей параметр має сенс лише тоді, коли фрезеруються як рядки, так і стовпці. Можливі параметри обмеження:

- Немає: Рядки та стовпці повністю фрезеруються.
- Додаткове: Під час фрезерування у другому напрямку ділянки, які не мають сильного нахилу в цьому напрямку, пропускаються.
- Повний: під час фрезерування в першому напрямку пропускаються ділянки, які мають сильний нахил у другому напрямку. Під час фрезерування в другому напрямку пропускаються ділянки, які не мають сильного нахилу в цьому напрямку.

11.10.4.18 кут контакту

Коли для параметра «Обмежувальні межі мережі» значення не має значення «Немає», схили, більші за «Кут контакту», вважаються «сильними» схилами, а схили, менші за цей кут, вважаються «слабкими» схилами.

11.10.4.19 Зміщення та глибина чорнової обробки за прохід

Image-to-gcode може опціонально виконувати чорнові проходи. Глибина послідовних чорнових проходів визначається параметром «Глибина чорнування за прохід». Наприклад, введення значення 0,2 призведе до виконання першого чорнового проходу з глибиною 0,2, другого чорнового проходу з глибиною 0,4 і так далі, доки не буде досягнута повна глибина зображення. Жодна частина операції чорнової обробки не буде наближатися до кінцевої деталі ближче, ніж «Зсув чорнової обробки». На наступному малюнку показано фрезерування високої вертикальної деталі. На цьому зображенні глибина чорнової обробки за один прохід становить 0,2 дюйма, а зсув чорнової обробки — 0,1 дюйма.

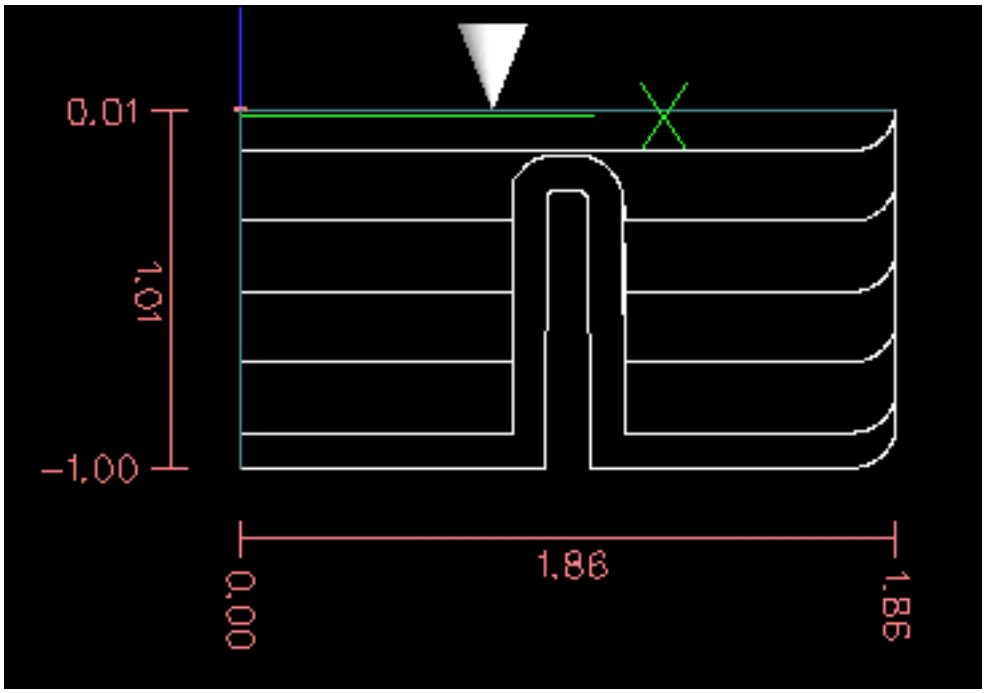


Figure 11.21: Грубі паси та фінальний пас

11.11 Відмінності RS274/NGC

11.11.1 Зміни з RS274/NGC

Відмінності, що змінюють значення програм RS274/NGC

Розташування після зміни інструменту
: У LinuxCNC верстат не повертається у вихідне положення після заміни інструменту. Ця зміна була внесена, оскільки новий інструмент може бути довшим за старий, і переміщення у вихідне положення верстата може призвести до того, що кінчик інструменту буде занадто низьким.

Параметри зсуву – це одиниці INI-файлу
: У LinuxCNC значення, що зберігаються в параметрах для початкових положень G28 і G30, систем координат P1...P9 і зміщення G92, виражені в «одиницях файлу INI». Ця зміна була внесена, оскільки в іншому випадку значення положення змінювалося залежно від того, чи був активним G20 або G21 під час програмування G28, G30, G10 L2 або G92.3.

Довжина/діаметри інструментального столу вказані в одиницях вимірювання INI-файлу
: У LinuxCNC довжина інструменту (зсув) і діаметр у таблиці інструментів вказуються тільки в одиницях виміру INI-файлу. Ця зміна була внесена, оскільки в іншому випадку довжина інструменту і його діаметр змінювалися б залежно від того, чи був активним G20 або G21 при запуску режимів G43, G41, G42. Це унеможливлювало виконання G-коду в невластивих для верстата одиницях виміру, навіть якщо G-код був простим і правильно сформованим (починався з G20 або G21 і не змінював одиниці виміру протягом програми), без зміни таблиці інструментів.

G84, G87 не реалізовано
: G84 та G87 наразі не реалізовані, але можуть бути додані до майбутньої версії LinuxCNC.

G28, G30 зі словами осей
Коли G28 або G30 програмується тільки з деякими словами осі, LinuxCNC переміщує тільки зазначені осі. Це є звичайним для інших систем управління верстатами. Щоб перемістити деякі осі до проміжної точки, а потім перемістити всі осі до заздалегідь визначеної точки, напишіть два рядки коду G:

```
G0 X- Y- (b’’ob’’b’’cb’’b’’ib’’ b’’дb’’b’’лb’’b’’яb’’ b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’мb’’b’ ↩
    ’ib’’b’’щb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ b’’дb’’b’’ob’’ b’’пb’’b’’pb’’b’’ob’’b’’мb’’b’ ↩
    ’ib’’b’’жb’’b’’нb’’b’’ob’’b’’їb’’ b’’тb’’b’’ob’’b’’чb’’b’’кb’’b’’иb’’)
G28 (b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’мb’’b’’ib’’b’’щb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ b’’вb’’b ↩
    ’’cb’’b’’ib’’b’’xb’’ b’’ob’’b’’cb’’b’’eb’’b’’йb’’ b’’дb’’b’’ob’’ b’’пb’’b’’ob’’b’’пb’’b’ ↩
    ’eb’’b’’pb’’b’’eb’’b’’дb’’b’’нb’’b’’ьb’’b’’ob’’ b’’вb’’b’’иb’’b’’зb’’b’’нb’’b’’ab’’b’ ↩
    ’чb’’b’’eb’’b’’нb’’b’’ob’’b’’їb’’ b’’тb’’b’’ob’’b’’чb’’b’’кb’’b’’иb’’)
```

11.11.2 Доповнення до RS274/NGC

Відмінності, які не змінюють значення програм RS274/NGC

G33, G76 коди потоків
Ці коди не визначені в RS274/NGC.

G38.2
Кінчик зонда не відводиться назад після руху G38.2. Цей рух відведення може бути додано в майбутній версії LinuxCNC.

G38.3...G38.5
Ці коди не визначені в RS274/NGC

O-коди
Ці коди не визначені в RS274/NGC

Перевизначення M50...M53
Ці коди не визначені в RS274/NGC

M61..M66
Ці коди не визначені в RS274/NGC

G43, G43.1
Негативна довжина інструменту

У специфікації RS274/NGC зазначено, що «очікується», що всі довжини інструментів будуть додатними. Однак G43 працює для від’ємних довжин інструментів.

Токарні інструменти

Компенсація довжини інструменту G43 може зміщувати інструмент як за вимірами X, так і за вимірами Z. Ця функція корисна в першу чергу на токарних верстатах.

Динамічні довжини інструментів

LinuxCNC дозволяє вказувати обчислену довжину інструменту за допомогою G43.1 I K.

G41.1, G42.1
LinuxCNC дозволяє вказати діаметр інструменту та, якщо в режимі токарного верстата, орієнтацію в G-коді. Формат: G41.1/G42.1 D L, де D – діаметр, а L (якщо вказано) – орієнтація токарного інструменту.

G43 без слова на букву “H”
У NGC це не дозволяється. У LinuxCNC це встановлює зміщення довжини для поточного завантаженого інструменту. Якщо інструмент не завантажений, це є помилкою. Ця зміна була внесена, щоб користувач не мав вказувати номер інструменту в двох місцях для кожної зміни інструменту, а також тому, що це відповідає способу роботи G41/G42, коли слово D не вказано.

Осі U, V та W

LinuxCNC дозволяє використовувати верстати з 9 осями, визначаючи додатковий набір з 3 лінійних осей, відомих як U, V та W

Chapter 12

Віртуальна панель керування

12.1 PyVCP

12.1.1 Вступ

PyVCP, віртуальний контрольний модуль Python, розроблений для того, щоб інтегратор міг налаштовува інтерфейс AXIS за допомогою кнопок та індикаторів для виконання спеціальних завдань.

Панелі керування апаратними засобами можуть використовувати багато контактів вводу/виводу та бути дорогими. Саме тут віртуальні панелі керування мають перевагу, оскільки створення PyVCP нічого не коштує.

Віртуальні панелі керування можуть використовуватися для тестування або моніторингу, щоб тимчасово замінити реальні пристрої вводу-виводу під час налагодження логіки релейно-контактних схем, або для імітації фізичної панелі перед її побудовою та підключенням до плати вводу-виводу.

На наступному малюнку показано багато віджетів PyVCP.

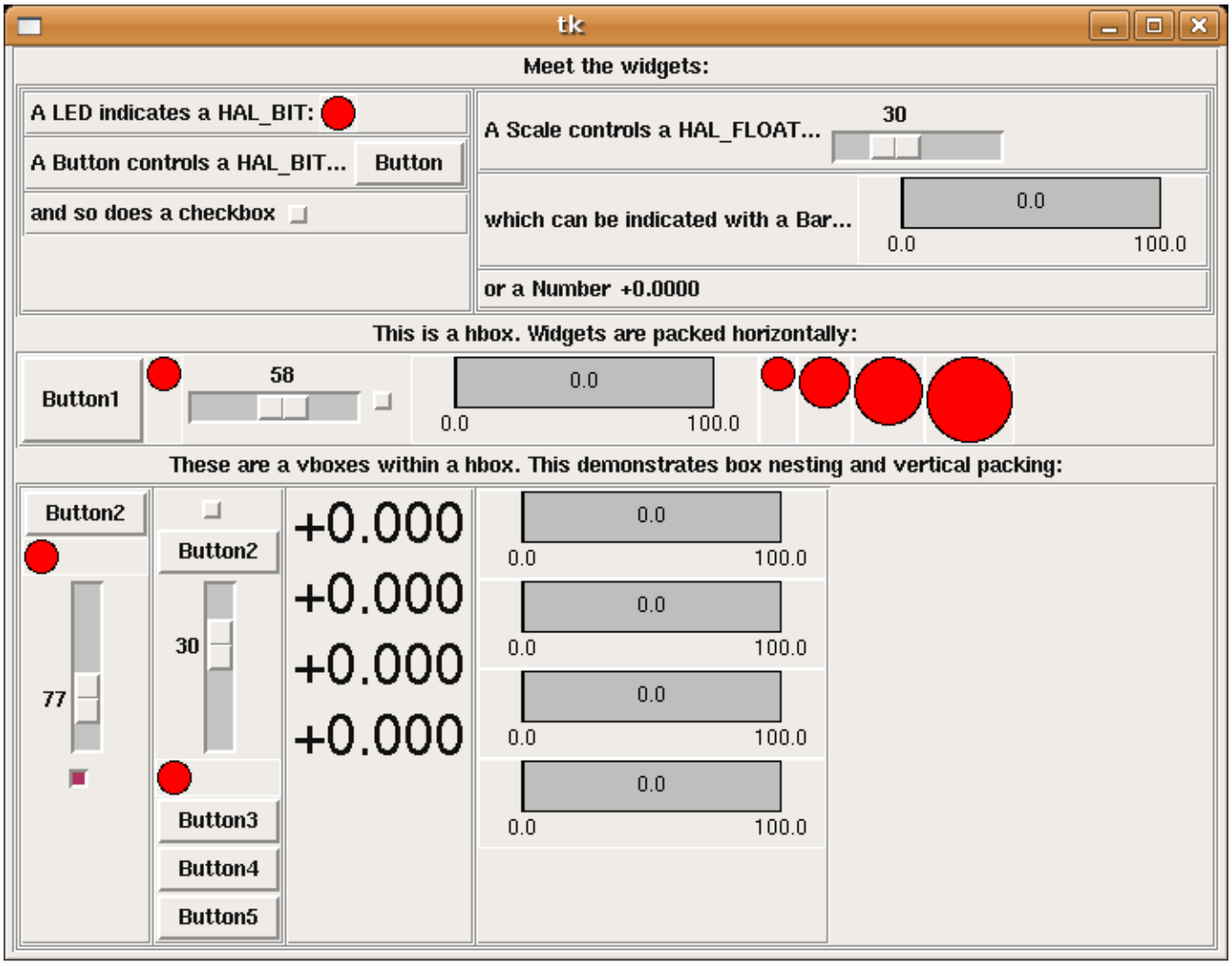


Figure 12.1: Презентація віджетів PyVCP

12.1.2 Панельне будівництво

Макет панелі PyVCP визначається за допомогою XML-файлу, який містить теги віджетів між <pyvcp> та </pyvcp>. Наприклад:

```
<pyvcp>
    <label text="This is a LED indicator"/>
    <led/>
</pyvcp>
```



Figure 12.2: Простий приклад світлодіодної панелі PyVCP

Якщо ви помістите цей текст у файл з назвою tiny.xml та запустите

```
halcmd loadusr pyvcp -c mypanel tiny.xml
```

PyVCP створить для вас панель, яка включає два віджети: мітку з текстом «Це світлодіодний індикатор» та світлодіод, що використовується для відображення стану сигналу HAL BIT. Він також створить компонент HAL з назвою «mypanel» (усі віджети на цій панелі підключені до контактів, що починаються з «mypanel»). Оскільки тег <halpin> не був присутній всередині тегу <led>, PyVCP автоматично назве контакт HAL для віджета світлодіода mypanel.led.0

Список віджетів, їхніх тегів і опцій дивіться у widget reference нижче.

Після створення панелі підключення сигналів HAL до та від контактів PyVCP виконується за допомогою halcmd:

```
net <signal-name> <pin-name> <opt-direction> <opt-pin-name>signal-name
```

Якщо ви новачок у HAL, розділ основ HAL у посібнику інтегратора – гарне місце для початку.

12.1.3 Безпека

Частини файлів PyVCP оцінюються як код Python і можуть виконувати будь-які дії, доступні програмам Python. Використовуйте XML-файли PyVCP лише з джерела, якому ви довіряєте.

12.1.4 AXIS

Оскільки AXIS використовує той самий набір інструментів графічного інтерфейсу користувача (Tkinter), що й PyVCP, можна додати панель PyVCP у правій частині або внизу інтерфейсу користувача AXIS. Неможливо відобразити панель одночасно в обох цих місцях. Типовий приклад наведено нижче.

На додаток до або замість відображення панелі PyVCP, як описано вище, можна відобразити одну або кілька панелей PyVCP у вигляді вбудованих вкладок у графічному інтерфейсі AXIS. Це можна зробити, виконавши наступні дії в розділі [DISPLAY] файлу INI:

```
EMBED_TAB_NAME    = Spindle
EMBED_TAB_COMMAND = pyvcp spindle.xml
```

Текстова позначка вкладки AXIS відображатиме «Шпиндель».

12.1.4.1 Приклад панелі

Помістіть файл PyVCP XML, що описує панель, у той самий каталог, де знаходиться файл INI. Припустимо, ми хочемо відобразити поточну швидкість шпинделя за допомогою віджета Bar. Помістіть наступне у файл під назвою spindle.xml:

```
<pyvcp>
    <label>
        <text>”Spindle speed:”</text>
    </label>
    <bar>
        <halpin>”spindle-speed”</halpin>
        <max_>5000</max_>
    </bar>
</pyvcp>
```

Тут ми створили панель з етикеткою та віджетом Bar, вказали, що контакт HAL, підключений до Bar, повинен називатися «spindle-speed», і встановили максимальне значення смуги на 5000 (див. widget reference нижче для всіх опцій). Щоб AXIS розпізнав цей файл і викликав його під час запуску, нам потрібно вказати наступне в розділі `[DISPLAY]` файлу INI:

```
PYVCP = spindle.xml
```

Якщо панель має відображатися внизу інтерфейсу користувача AXIS, нам потрібно вказати наступне в розділі `[DISPLAY]` INI-файлу:

```
PYVCP_POSITION = BOTTOM
```

Будь-яке значення, відмінне від BOTTOM, або пропуск цієї змінної розмістить панель PYVCP праворуч.

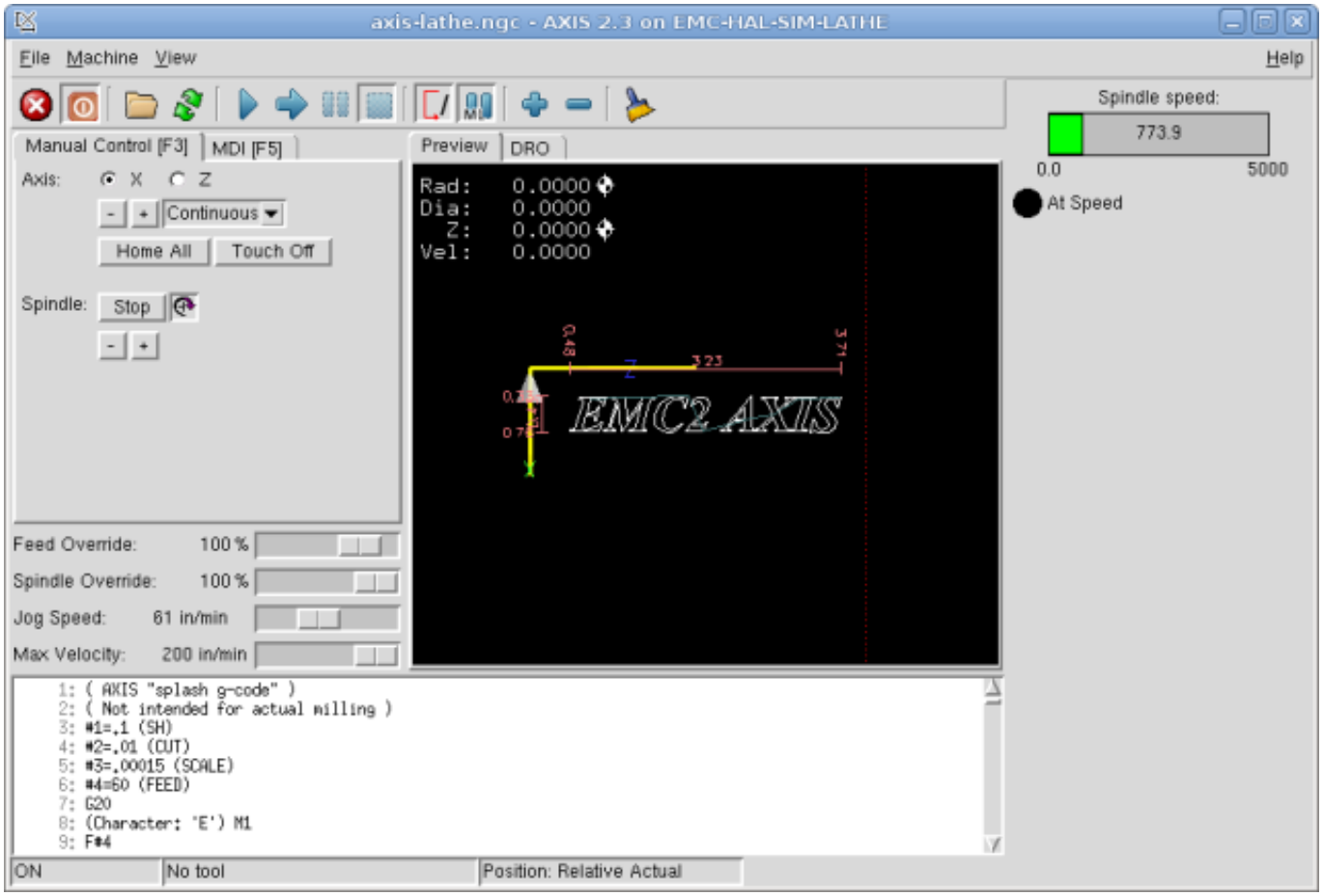
Щоб наш віджет фактично відображав швидкість обертання шпинделя, його потрібно підключити до відповідного сигналу HAL. Файл HAL, який буде запущений після запуску AXIS і PyVCP, можна вказати в розділі [HAL] файлу INI:

```
POSTGUI_HALFILE = spindle_to_pyvcp.hal
```

Ця зміна призведе до запуску команд HAL, зазначених у *spindle_to_pyvcp.hal*. У нашому прикладі вміст може виглядати так:

```
net spindle-rpm-filtered => pyvcp.spindle-speed
```

припускаючи, що сигнал під назвою «spindle-rpm-filtered» вже існує. Зверніть увагу, що під час роботи разом з AXIS усі виводи HAL віджетів панелі PyVCP мають імена, що починаються з «pyvcp.», а всі виводи HAL вбудованих віджетів вкладок PyVCP починаються з імені, вказаного як EMBED_TAB_NAME, перетвореного в нижній регістр.



Ось як має виглядати щойно створена панель PyVCP в AXIS. Конфігурація *sim/lathe* вже налаштована таким чином.

12.1.5 Окремо стояти

У цьому розділі описано, як панелі PyVCP можуть відображатися окремо з контролером верстата LinuxCNC або без нього.

Щоб завантажити окрему панель PyVCP за допомогою LinuxCNC, використовуйте ці команди:

```
loadusr -Wn mypanel pyvcp -g WxH+X+Y -c mypanel <path/>panel_file.xml
```

Ви б використовували це, якщо б вам потрібна плаваюча панель або панель з графічним інтерфейсом, відмінним від AXIS.

- *-Wn panelname* - змушує HAL чекати, поки компонент *panelname* закінчить завантаження (*стане готовим* в термінології HAL), перш ніж обробляти інші команди HAL. Це важливо, оскільки панелі PyVCP експортують контакти HAL, а інші компоненти HAL потребують їх наявності для підключення. Зверніть увагу на велику літеру W і малу літеру n. Якщо ви використовуєте опцію -Wn, ви повинні використовувати опцію -c для присвоєння імені панелі.
- *pyvcp < -g> < -c> panel.xml* - створює панель з опціональною геометрією та/або назвою панелі з XML-файлу панелі. Файл panel.xml може мати будь-яку назву, що закінчується на .xml. Файл .xml - це файл, що описує, як створити панель. Ви повинні додати ім'я шляху, якщо панель не знаходиться в каталозі, де знаходиться скрипт HAL.
- *-g <WxH><+X+Y>* - визначає геометрію, яка буде використовуватися при побудові панелі. Синтаксис: *Ширина x Висота + X Анкер + Y Анкер*. Ви можете встановити розмір, положення або і те, і інше. Точка прив'язки — це верхній лівий кут панелі. Приклад: -g 250x500+800+0. Це встановлює ширину панелі 250 пікселів, висоту 500 пікселів і прив'язує її до X800 Y0.
- *-c назва_панелі* – повідомляє PyVCP, як називати компонент, а також назву вікна. Назва панелі може бути будь-якою без пробілів.

Щоб завантажити «окрему» панель PyVCP без LinuxCNC, використовуйте цю команду:

```
loadusr -Wn mypanel pyvcp -g 250x500+800+0 -c mypanel mypanel.xml
```

Мінімальна команда для завантаження панелі PyVCP:

```
loadusr pyvcp mypanel.xml
```

Ви б використовували це, якщо вам потрібна панель без контролера верстата LinuxCNC, наприклад, для тестування або автономного DRO.

Команда loadusr використовується, коли ви також завантажуєте компонент, який зупинить закриття HAL до завершення завантаження. Якщо ви завантажили панель, а потім завантажили Classic Ladder за допомогою *loadusr -w classicladder*, CL утримуватиме HAL відкритим (а також панель) доти, доки ви не закриєте CL. «-Wn» вище означає очікування, поки компонент «-Wn "name"» буде готовий. («name» може бути будь-яким іменем. Зверніть увагу на велику літеру W і малу літеру n.) -c вказує PyVCP створити панель з іменем «panelname» за допомогою інформації в «panel_file_name.xml». Ім'я «panel_file_name.xml» може бути будь-яким, але повинно закінчуватися на .xml — це файл, який описує, як створити панель. Ви повинні додати ім'я шляху, якщо панель не знаходиться в каталозі, в якому знаходиться скрипт HAL.

Додаткова команда, яку можна використовувати, якщо потрібно, щоб панель зупинила HAL від продовження команд / завершення роботи. Після завантаження будь-яких інших компонентів остання команда HAL має бути такою:

```
waitusr panelname
```

Це вказує HAL чекати закриття компонента «panelname», перш ніж продовжувати виконувати команди HAL. Зазвичай це встановлюється як остання команда, щоб HAL завершував роботу після закриття панелі.

12.1.6 Віджети

Сигнали HAL бувають двох типів: біти та числа. Біти — це сигнали увімкнення/вимкнення. Числа можуть бути типу «float», «s32», «u32», «s64» або «u64». Більш детальну інформацію про типи даних HAL див. у розділі HAL Data. Віджет PyVCP може або відображати значення сигналу за допомогою індикатора, або змінювати значення сигналу за допомогою елемента керування. Таким чином, існує чотири класи віджетів PyVCP, які можна підключити до сигналу HAL. П’ятий клас допоміжних віджетів дозволяє організувати та позначити панель.

- Віджети для індикації бітових сигналів: `led`, `rectled`.
- Віджети для керування бітовими сигналами: `button`, `checkbutton`, `radiobutton`.
- Віджети для індикації сигналів «число»: `number`, `s32`, `u32`, `bar`, `meter`.
- Віджети для керування «числовими» сигналами: `spinbox`, `scale`, `jogwheel`.
- Допоміжні віджети: `hbox`, `vbox`, `table`, `label`, `labelframe`.

12.1.6.1 Синтаксис

Кожен віджет коротко описується, потім вказано використану розмітку та зроблено знімок екрана. Усі теги всередині основного тегу віджета є необов’язковими.

12.1.6.2 Загальні примітки

Наразі підтримується синтаксис як на основі тегів, так і на основі атрибутів. Наприклад, наступні фрагменти XML обробляються однаково:

```
<led halpin=”my-led”/>
```

і

```
<led><halpin>”my-led”</halpin></led>
```

Коли використовується синтаксис на основі атрибутів, для перетворення значення атрибута на значення Python використовуються такі правила:

1. Якщо перший символ атрибута є одним із наведених нижче, він обчислюється як вираз Python: `{([”’` .
2. Якщо рядок прийнято функцією int(), значення обробляється як ціле число.
3. Якщо рядок приймається функцією float(), значення обробляється як число з плаваючою комою.
4. В іншому випадку рядок приймається як рядок.

Коли використовується синтаксис на основі тегів, текст усередині тегу завжди обчислюється як вираз Python.

Наведені нижче приклади демонструють поєднання форматів.

Коментарі Щоб додати коментар, використовуйте синтаксис XML для коментарів.

```
<!-- My Comment -->
```

Редагування XML-файлу Відредагуйте XML-файл за допомогою текстового редактора. У більшості випадків ви можете клацнути правою кнопкою миші на файлі та вибрати «відкрити за допомогою текстового редактора» або щось подібне.

Кольори

Кольори можна вказати за допомогою кольорів X11 rgb за назвою «gray75» або шістнадцятковим числом «#0000ff». Повний список знаходиться тут https://sedition.com/perl/rgb.html.

Поширені Кольори (кольори з цифрами позначають відтінки цього кольору)

- білий
- чорний
- синій та синій1 - 4
- блакитний та блакитний1 - 4
- зелений та зелений1 - 4
- жовтий та жовтий1 - 4
- червоний та червоний1 - 4
- фіолетовий та пурпурний1 - 4
- сірий та сірий0 - 100

Піни HAL Контакти HAL забезпечують можливість «підключення» віджета до чогось. Після створення контакту HAL для віджета його можна «підключити» до іншого контакту HAL за допомогою команди «net» у файлі .hal. Докладнішу інформацію про команду «net» див. у розділі HAL Commands.

12.1.6.3 Мітка

Мітка – це спосіб додавання тексту до панелі.

- *<label></label>* - створює мітку.
- *<text>”текст”</text>* – текст, який потрібно вставити в підпис; порожній підпис можна використов як роздільник для вирівнювання інших об’єктів.
- *(”Helvetica”,20)* - вкажіть шрифт і розмір тексту.
- *<relief>FLAT</relief>* - вкажіть рамку навколо мітки («ПЛОСКАЯ», «ПІДНЯТА», «ЗАТОПЛАТА»), за замовчуванням — «ПЛОСКАЯ».
- *<bd>_n_</bd>* - де *n* — ширина межі, коли використовуються межі типу «ПІДНЯТІ» або «ЗАГНУТІ».
- *<padx>_n_</padx>* - де *n* – це кількість додаткового горизонтального простору.
- *<pady>_n_</pady>* - де *n* – це кількість додаткового вертикального простору.

Мітка має додатковий PIN-код для відключення, який створюється під час додавання *<disable_pin>Tru*

```
<label>
    <text>”b’’Цb’’b’’eb’’ b’’мb’’b’’ib’’b’’тb’’b’’кb’’b’’ab’’:”</text>
    <font>(”Helvetica”,20)</font>
</label>
```

Наведений вище код створив цей приклад:



Figure 12.3: Приклад простої етикетки

12.1.6.4 Багатомітка

Розширення текстової мітки.

Вибір текстової мітки, може відображати до 6 підписів міток, коли активовано відповідний бітовий контакт.

Приєднайте кожен контакт легенди до сигналу та отримайте описову мітку, коли сигнал має значення TRUE.

Якщо більше одного позначення має значення TRUE, буде відображено позначення з найбільшим номером «TRUE».

Якщо PIN-код відключення створено з параметром *<disable_pin>True</disable_pin>*, і для цього PIN-кода встановлено значення true, мітка змінить свій стан на сірий.

```
<multilabel>
    <legends>[”Label1”, ”Label2”, ”Label3”, ”Label4”, ”Label5”, ”Label6”]</legends>
    <font>(”Helvetica”,20)</font>
    <disable_pin>True</disable_pin>
</multilabel>
```

У наведеному вище прикладі буде створено такі контакти.

```
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
```

Якщо у вас є більше однієї мультимітки, створені піни збільшуватимуть число ось так: *pyvcp.multilabel.*

12.1.6.5 LEDs

Світлодіод використовується для індикації стану halpin-розряду (бітового). Колір світлодіода буде on_color, коли halpin має значення true, і off_color в іншому випадку.

- *<led></led>* - створює круглий світлодіод

- *<rectled></rectled>* - створює прямокутний світлодіод
- *<halpin>*name*</halpin>* – *назва* виводу, за замовчуванням – *led*.n, де *n* – ціле число, яке збільшується на 1 для кожного світлодіода.
- *<size>*n*</size>* - *n* — розмір світлодіода в пікселях, за замовчуванням — 20.
- *<on_color>*color*</on_color>* - встановлює колір світлодіода на *color*, коли вивід має значення true. За замовчуванням — «зелений». Див. розділ colors для отримання додаткової інформації.
- *<off_color>*color*</off_color>* - встановлює колір світлодіода на *color*, коли контакт перебуває у стані хибності. За замовчуванням — «червоний».
- *<height>*n*</height>* - встановлює висоту світлодіода в пікселях.
- *<width>*n*</width>* - встановлює ширину світлодіода в пікселях.
- *<disable_pin>false</disable_pin>* - коли значення true, до світлодіода додається контакт відключення
- *<disabled_color>*color*</disabled_color>* - встановлює колір світлодіода на *color*, коли контакт вимкнено.

Круглий світлодіод

```
<led>
    <halpin>”my-led”</halpin>
    <size>50</size>
    <on_color>”green”</on_color>
    <off_color>”red”</off_color>
</led>
```

Наведений вище код створив цей приклад:

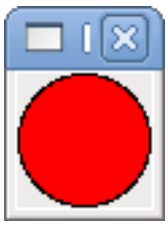


Figure 12.4: Приклад круглого світлодіода

Прямокутний світлодіод Це варіант віджета «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>
```

Наведений вище код створив цей приклад. Також показано вертикальну рамку з рельєфом.



Figure 12.5: Простий приклад прямокутного світлодіода

12.1.6.6 Кпонки

Кнопка використовується для керування виводом BIT. Вивід буде встановлений у стан True, коли кнопка натиснута і утримується, і буде встановлений у стан False, коли кнопка відпущена. Кнопки можуть використовувати наступні опціональні параметри.

- *<padx>*n*</padx>* - де *n* – це кількість додаткового горизонтального простору.
- *<pady>*n*</pady>* - де *n* – це кількість додаткового вертикального простору.
- *<activebackground>”*color*”</activebackground>* - курсор над кольором встановлено на *color*.
- *<fg>”*color*”</fg>* - колір переднього плану встановлено на *color*.
- *<bg>”*color*”</bg>* - колір фону встановлено на *color*.
- *<disable_pin>True</disable_pin>* – вимкнути закріплення.

Текстова кнопка Текстова кнопка керує «бітовим» halpin. Halpin має значення false (хибність), доки кнопка не буде натиснута, тоді воно стає істинним. Кнопка є тимчасовою кнопкою.

Текстова кнопка має необов’язковий PIN-код для відключення, який створюється після додавання <disable_pin>True</disable_pin>.

```
<button>
    <halpin>”ok-button”</halpin>
    <text>”OK”</text>
</button>
<button>
    <halpin>”abort-button”</halpin>
    <text>”Abort”</text>
</button>
```

Наведений вище код створив цей приклад:



Figure 12.6: Приклад простих кнопок

Кнопка перевірки Перемикач керує бітом halpin. Halpin буде встановлений у значення True, коли перемикач увімкнено, і у значення false, коли перемикач вимкнено. Перемикач є кнопкою типу toggle. Перемикачі можуть бути спочатку встановлені у значення TRUE або FALSE у полі initval. Також автоматично створюється контакт changepin, який може перемикати перемикач через HAL, якщо пов'язане значення змінено, для віддаленого оновлення дисплея.



Figure 12.7: Невизначена кнопка



Figure 12.8: Позначена кнопка

Приклад коду кнопки-перевірки

```
<checkbutton>
    <halpin>"coolant-chkbtn"</halpin>
    <text>"Coolant"</text>
    <initval>1</initval>
</checkbutton>
<checkbutton>
    <halpin>"chip-chkbtn"</halpin>
    <text>"Chips    "</text>
    <initval>0</initval>
</checkbutton>
```

Наведений вище код створив цей приклад:

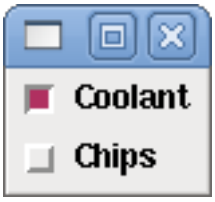


Figure 12.9: Простий приклад кнопки-перевірки

Кнопка перевірки охолоджувальної рідини позначена.
Зверніть увагу на додаткові пробіли в тексті «Стрічки», щоб вирівняти кнопки перевірки.

Перемикач Радіо-кнопка встановить один з галпінів як істинний. Інші галпіни встановлюються як хибні. Поле initval може бути встановлено для вибору значення за замовчуванням при відображенні панелі. Тільки одна радіо-кнопка може бути встановлена як TRUE (1), або тільки галпін з найвищим номером, встановлений як TRUE, матиме це значення.

```
<radiobutton>
    <choices>["one","two","three"]</choices>
    <halpin>"my-radio"</halpin>
    <initval>0</initval>
</radiobutton>
```

Наведений вище код створив цей приклад:

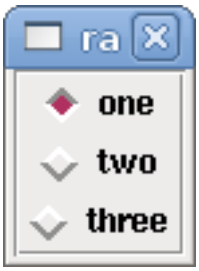


Figure 12.10: Простий приклад радіокнопки

Зверніть увагу, що контакти HAL у наведеному вище прикладі будуть називатися my-radio.one, my-radio.two та my-radio.three. На зображенні вище «one» є вибраним значенням. Використовуйте тег `<orient>HORIZONTAL</orient>` для горизонтального відображення.

12.1.6.7 Числові дисплеї

Числові відображення можуть використовувати такі параметри форматування

- `<font>("Font Name",n)</font>`, де *n* – розмір шрифту.
- `<width>_n_</width>`, де *n* – загальна ширина використаного простору.
- `<justify>_pos_</justify>`, де *pos* має значення LEFT, CENTER або RIGHT (не працює).
- `<padx>_n_</padx>`, де *n* – це кількість додаткового горизонтального простору.
- `<pady>_n_</pady>`, де *n* – це кількість додаткового вертикального простору.

Номер Віджет чисел відображає значення сигналу з плаваючою комою.

```
<number>
    <halpin>"my-number"</halpin>
    <font>("Helvetica",24)</font>
    <format>"+4.4f"</format>
</number>
```

Наведений вище код створив цей приклад:



Figure 12.11: Простий приклад числа

- *<font>* - — це специфікація типу та розміру шрифту Tkinter. Один шрифт, який відображатиметься щонайменше до розміру 200, — це «courier 10 pitch», тому для справді великого віджета Number ви можете вказати:

```
<font>("courier 10 pitch",100)</font>
```

- *<format>* - — це формат у стилі C, який визначає, як відображається число.

Номер s32 Віджет s32 number відображає значення числа s32. Синтаксис такий самий, як у *number*, за винятком імені, яке є <s32>. Переконайтеся, що ширина достатня для відображення найбільшого числа, яке ви плануєте використовувати.

```
<s32>
    <halpin>”my-number”</halpin>
    <font>(”Helvetica”,24)</font>
    <format>”6d”</format>
    <width>6</width>
</s32>
```

Наведений вище код створив цей приклад:

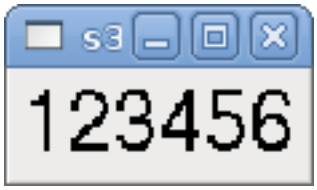


Figure 12.12: Простий приклад номера s32

Номер u32 Віджет числа u32 відображає значення числа u32. Синтаксис такий самий, як і для «число», за винятком назви <u32>.

Бар Віджет «Смуга» відображає значення сигналу FLOAT як графічно за допомогою смуги, так і числово. Колір смуги можна встановити як один колір для всього діапазону (за замовчуванням використовується колір заповнення) або встановити зміну кольору залежно від значення halpin (діапазони range1, range2 та range3 повинні бути встановлені всі; якщо ви хочете лише 2 діапазони, встановіть для них один і той самий колір).

- <halpin>”my-bar”</halpin> (текст), отримує та встановлює назву піна: pyvcp.my-bar.
- <min_>0</min_> (число), встановлює мінімальний масштаб.
- <max_>140</max_> (число), встановлює максимальний масштаб.
- <format>”3.1f”</format> (текст), встановлює формат числа за допомогою форматування чисел Python.
- <bgcolor>”grey”</bgcolor> (текст), встановлює колір фону.
- <fillcolor>”red”</fillcolor> (текст), встановлює колір заливки.
- <range1>0,100,”green”</range1> (число, число, текст), встановлює перший діапазон і колір.
- <range2>101,135,”orange”</range2> (число, число, текст), встановлює перший діапазон і колір.
- <range3>136, 150,”red”</range3> число, число, текст), встановлює перший діапазон і колір.
- <canvas_width>200</canvas_width> (число), встановлює загальну ширину.
- <canvas_height>50</canvas_height> (число), встановлює загальну висоту.
- <bar_height>30</bar_height> (число), встановлює висоту смуги, має бути меншою за canvas_height.
- <bar_width>150</bar_width> (число), встановлює ширину смуги, має бути меншою за ширину_канав

```
<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>
```

Наведений вище код створив цей приклад:



Figure 12.13: Простий приклад бару

Метр Вимірювач відображає значення сигналу FLOAT за допомогою традиційного індикатора годинникового типу.

```
<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>
```

Наведений вище код створив цей приклад:

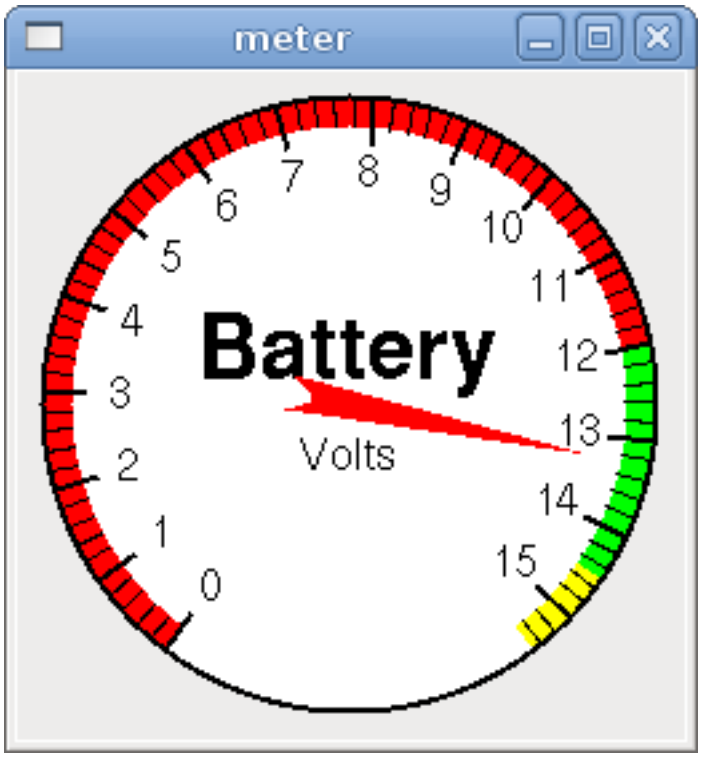


Figure 12.14: Простий приклад лічильника

12.1.6.8 Числові вводи

Спінбокс Спінбокс керує контактом FLOAT. Ви можете збільшити або зменшити значення контакту, натискаючи на стрілки або вказуючи на спінбокс і прокручуючи коліщатко миші. Якщо поле param_pin встановлено на TRUE(1), буде створено контакт, який можна використовувати для встановлення спінбоксу на початкове значення та віддаленого змінення його значення без введення 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>
```

Наведений вище код створив цей приклад:



Figure 12.15: Простий приклад спінбоксу

Масштаб Шкала контролює плаваючий або s32-контакт. Ви можете збільшити або зменшити значення контакту, перетягнувши повзунок або вказавши на шкалу і прокрутивши коліщатко миші. До «halpin» буде додано «-f» і «-i», щоб сформувати плаваючий і s32-контакти. Ширина — це ширина повзунка у вертикальному напрямку та висота повзунка у горизонтальному напрямку. Якщо поле param_pin встановлено на TRUE(1), буде створено контакт, який можна використовувати для встановлення початкового значення спінбокса та віддаленого змінення його значення без введення 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>
```

Наведений вище код створив цей приклад:

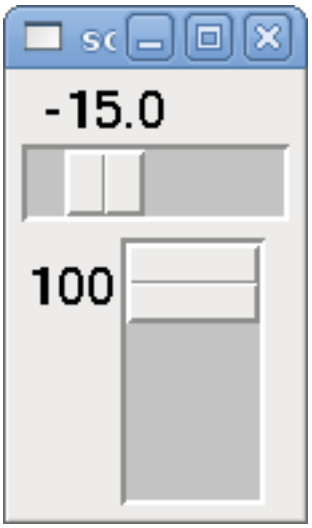


Figure 12.16: Приклад простого масштабу

Note
Зверніть увагу, що за замовчуванням відображається значення "min", навіть якщо воно більше за "max", окрім випадків, коли "min" є від'ємним числом.

Циферблат Dial виводить HAL float і реагує як на колесо миші, так і на перетягування. Двічі клацніть лівою кнопкою миші, щоб збільшити роздільну здатність, і двічі клацніть правою кнопкою

миші, щоб зменшити роздільну здатність на один цифру. Вихід обмежений мінімальним і максимальни значеннями. <cpr> — це кількість поділок на зовнішній стороні кільця (будьте обережні з великими числами). Якщо поле param_pin встановлено на TRUE(1), буде створено контакт, який можна використовувати для встановлення початкового значення спінбокса та віддаленого змінення його значення без введення HID.

```
<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>
```

Наведений вище код створив цей приклад:

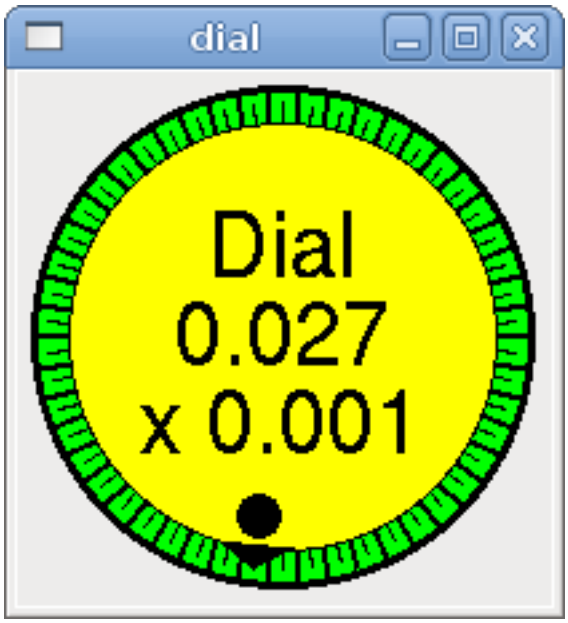


Figure 12.17: Приклад простого набору номера

Джоґвіст Джог-колесо імітує справжнє колесо, виводячи значення FLOAT, яке рахує вгору або вниз під час обертання колеса, або шляхом перетягування круговим рухом, або шляхом прокручування колеса миші.

Додаткові теги: * *<text>”My Text”</text>* відображає текст * *<bgcolor>”grey”</bgcolor>* *<fillcolor>”green”</fillcolor>* фонові та активні кольори * *<scale_pin>1</scale_pin>* створює текст масштабу та пін `FLOAT.scale` для відображення масштабу поперечного перетину * *<clear_pin>1</clear* створює `DRO` та контакт `BIT.reset` для скидання DRO. Потрібен scale_pin для масштабованого DRO. Shift+клацання також скидає DRO

```
<jogwheel>
    <halpin>”my-wheel”</halpin>
    <cpr>45</cpr>
```

```
    <size>250</size>
</jogwheel>
```

Наведений вище код створив цей приклад:

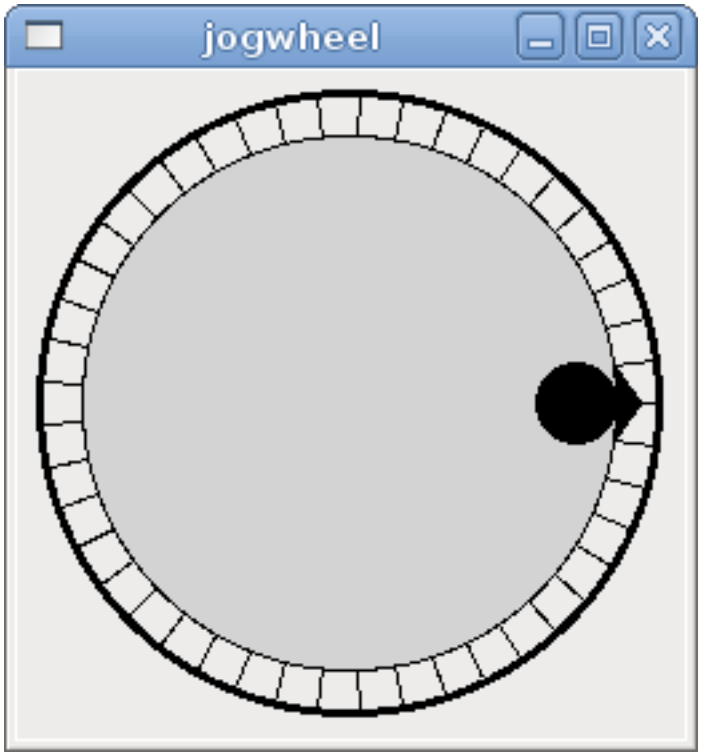


Figure 12.18: Простий приклад джогового колеса

12.1.6.9 Зображення

Для відображення зображень використовується тільки формат .gif. Всі зображення повинні бути однакового розміру. Зображення повинні знаходитися в тому ж каталозі, що і файл INI (або в поточному каталозі, якщо програма запускається з командного рядка за допомогою halrun/halcmd).

Біт зображення «image_bit» перемикається між двома зображеннями, встановлюючи halpin у значення true або false.

```
<image name='fwd' file='fwd.gif'/>
<image name='rev' file='rev.gif'/>
<vbox>
    <image_bit halpin='selectimage' images='fwd rev'/>
</vbox>
```

Цей приклад було створено з наведеного вище коду. Використовуючи два файли зображень fwd.gif та rev.gif. FWD відображається, коли *selectimage* має значення false, а REV відображається, коли *selectimage* має значення true.



Figure 12.19: Приклад хибного вибору зображення



Figure 12.20: Вибрати зображення Правдивий приклад

Зображення u32 *image_u32* аналогічний *image_bit*, за винятком того, що ви маєте практично необмежену кількість зображень і *вибираєте* зображення, встановлюючи halpin на ціле значення, де 0 відповідає першому зображенню в списку зображень, 1 — другому зображенню тощо.

```
<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>
```

Наведений вище код створив наступний приклад, додавши зображення stb.gif.



Figure 12.21: Простий приклад image_u32 з halpin=0



Figure 12.22: Простий приклад image_u32 з halpin=1



Figure 12.23: Простий приклад image_u32 з halpin=2

Зверніть увагу, що за замовчуванням використовується мінімум, навіть якщо він встановлений вище за максимум, якщо тільки немає від'ємного мінімуму.

12.1.6.10 Контейнери

Контейнери – це віджети, які містять інші віджети. Контейнери використовуються для групування інших віджетів.

Межі Межі контейнерів визначаються двома тегами, що використовуються разом. Тег <relief> визначає тип межі, а <bd> – її ширину.

<relief>_type_</relief>
: Де *тип* має значення FLAT (ПЛОСКИЙ), SUNKEN (ЗАГНУТИЙ), RAISE (ПІДНЯТИЙ), GROOVE (БАЗОВИЙ) або RIDGE (ГРЕБЕНЬ).

<bd>_n_</bd>
: Де *n* – ширина межі.

```
<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>
```

Наведений вище код створив цей приклад:



Figure 12.24: Вітрина кордонів контейнерів

Заповнити Заповнення контейнера визначається тегом <*boxfill fill=""*/>. Дійсні записи: none, x, y та обидва. Заповнення x – це горизонтальне заповнення, а заповнення y – вертикальне заповнення

<boxfill fill ="style"/>
Де «стиль» має значення none, x, y або обидва. За замовчуванням x для Vbox та y для Hbox.

Якір Контейнерні якорі вказуються за допомогою тегу <boxanchor anchor=""/>. Якір вказує, де розмістити кожного веденого в його ділянці. Допустимі значення: center, n, s, e, w (для центру, півночі, півдня, сходу та заходу). Також допустимі комбінації, такі як sw, se, nw та ne.

<boxanchor anchor="position"/>
Де «позиція» – це center, n, s, e, w, ne, nw, se або sw. За замовчуванням – center.

Розгорнути Розгортання контейнера визначається логічним тегом <boxexpand expand=""/>. Дійсні записи: "yes", "no".

<boxexpand expand="boolean"/>
Де «boolean» означає або «так», або «ні». За замовчуванням — «так».

Hbox Використовуйте Hbox, коли хочете розмістити віджети горизонтально один біля одного.

```
<hbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>"a hbox:"</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</hbox>
```

Наведений вище код створив цей приклад:

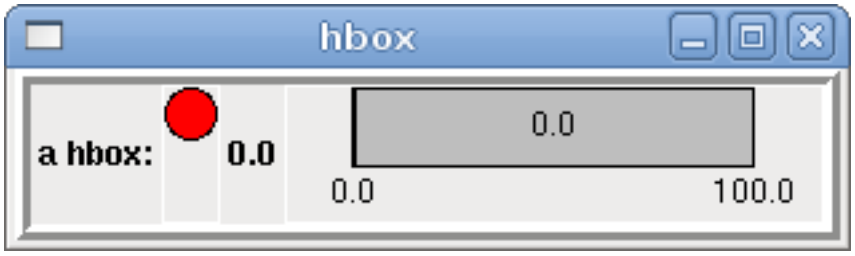


Figure 12.25: Простий приклад hbox

Всередині Hbox ви можете використовувати теги *<boxfill fill=””/>*, *<boxanchor anchor=””/>* та *<boxexpand expand=””/>*, щоб вибрати, як елементи в полі будуть поводитися при зміні розміру вікна. За замовчуванням для Hbox використовуються значення *fill=”y”*, *anchor=”center”*, *expand=”yes”*.

Vbox Використовуйте Vbox, коли хочете розмістити віджети вертикально один на одному.

```
<vbox>
    <relief>RIDGE</relief>
    <bd>6</bd>
    <label><text>”a vbox:”</text></label>
    <led></led>
    <number></number>
    <bar></bar>
</vbox>
```

Наведений вище код створив цей приклад:

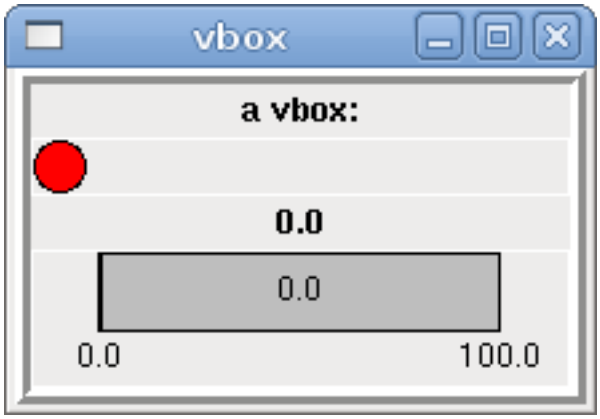


Figure 12.26: Простий приклад VBox

Усередині Vbox ви можете використовувати теги *<boxfill fill=””/>*, *<boxanchor anchor=””/>* та *<boxexpand expand=””/>*, щоб вибрати, як елементи в полі будуть поводитися при зміні розміру вікна. За замовчуванням для Vbox використовуються значення *fill=”x”*, *anchor=”center”*, *expand=”yes”*.

Рамка етикетки Рамка для етикетки — це рамка з пазом та етикеткою у верхньому лівому куті.

```
<labelframe text=”b’’Mb’’b’’ib’’b’’тb’’b’’кb’’b’’ab’’: b’’3b’’b’’гb’’b’’pb’’b’’yb’’b’’пb’’b ↩
    ’’ob’’b’’вb’’b’’ab’’b’’нb’’b’’ib’’ b’’cb’’b’’вb’’b’’ib’’b’’тb’’b’’лb’’b’’ob’’b’’дb’’b’ ↩
    ’ib’’b’’ob’’b’’дb’’b’’иb’’”>
```

```
<labelframe text="Group Title">
    <font>("Helvetica",16)</font>
    <hbox>
    <led/>
    <led/>
    </hbox>
</labelframe>
```

Наведений вище код створив цей приклад:



Figure 12.27: Простий приклад рамки етикетки

Таблиця Таблиця — це контейнер, який дозволяє розміщувати елементи у вигляді сітки рядків і стовпців. Кожен рядок починається тегом «<tablerow/>». Вміщений віджет може охоплювати рядки або стовпці за допомогою тегу «<tablespan rows= cols=/>». Сторони комірок, до яких «прилипають» вміщені віджети, можна встановити за допомогою тегу «<tablesticky sticky=/>». Таблиця розширюється за допомогою гнучких рядків і стовпців.

Приклад коду таблиці

```
<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>
```

Наведений вище код створив цей приклад:

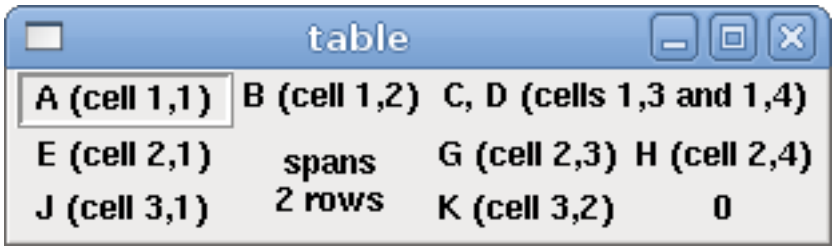


Figure 12.28: Приклад таблиці

Вкладки Інтерфейс із вкладками може заощадити чимало місця.

```
<tabs>
    <names> [”spindle”,”green eggs”]</names>
</tabs>
<tabs>
    <names>[”b’’Шb’’b’’пb’’b’’иb’’b’’нb’’b’’дb’’b’’еb’’b’’лb’’b’’ьb’’”, ”b’’Зb’’b’’еb’’b’ ↩
        ’лb’’b’’еb’’b’’нb’’b’’іb’’ b’’яb’’b’’йb’’b’’цb’’b’’яb’’”, ”b’’Шb’’b’’иb’’b’’нb’’b’ ↩
        ’кb’’b’’ab’’”]</names>
    <vbox>
        <label>
            <text>”b’’Шb’’b’’вb’’b’’иb’’b’’дb’’b’’кb’’b’’іb’’b’’сb’’b’’тb’’b’’ьb’’ b’’шb’’b ↩
                ’’пb’’b’’иb’’b’’нb’’b’’дb’’b’’еb’’b’’лb’’b’’яb’’:”</text>
        </label>
        <bar>
            <halpin>”spindle-speed”</halpin>
            <max_>5000</max_>
        </bar>
    </vbox>
    <vbox>
        <label>
            <text>”(b’’цb’’b’’еb’’ b’’вb’’b’’кb’’b’’лb’’b’’аb’’b’’дb’’b’’кb’’b’’аb’’ b’ ↩
                ’зb’’b’’еb’’b’’лb’’b’’еb’’b’’нb’’b’’иb’’b’’хb’’ b’’яb’’b’’єb’’b’’цb’’b’ ↩
                ’ьb’’)”</text>
        </label>
    </vbox>
    <vbox>
        <label>
            <text>”(b’’цb’’b’’яb’’ b’’вb’’b’’кb’’b’’лb’’b’’аb’’b’’дb’’b’’кb’’b’’аb’’ b’ ↩
                ’пb’’b’’оb’’b’’рb’’b’’оb’’b’’жb’’b’’нb’’b’’яb’’)”</text>
        </label>
    </vbox>
</tabs>
```

Наведений вище код створив цей приклад, що показує кожну вибрану вкладку.



Figure 12.29: Приклад простих вкладок

12.2 Приклади PyVCP

12.2.1 AXIS

Щоб створити панель PyVCP для використання з інтерфейсом AXIS, підключеним праворуч від AXIS, вам потрібно виконати такі основні дії.

- Створіть XML-файл, який містить опис вашої панелі, та помістіть його в каталог конфігурації.
- Додайте запис PyVCP до розділу [DISPLAY] INI-файлу разом з вашим ім’ям XML-файлу.
- Додайте запис POSTGUI_HALFILE до розділу [HAL] INI-файлу з назвою вашого HAL-файлу Postgui.
- Додайте посилання на піни HAL для вашої панелі у файл postgui.hal, щоб «підключити» вашу панель PyVCP до LinuxCNC.

12.2.2 Плаваючі панелі

Щоб створити плаваючі панелі PyVCP, які можна використовувати з будь-яким інтерфейсом, потрібно виконати такі основні дії.

- Створіть XML-файл, який містить опис вашої панелі, та помістіть його в каталог конфігурації.
- Додайте рядок loadusr до вашого HAL-файлу для завантаження кожної панелі.
- Додайте посилання на піни HAL для вашої панелі у файл postgui.hal, щоб «підключити» вашу панель PyVCP до LinuxCNC.

Нижче наведено приклад команди loadusr для завантаження двох панелей PyVCP та надання кожній з них імені, щоб імена з’єднань у HAL були відомі.

```
loadusr -Wn btnpanel pyvcp -c btnpanel panel1.xml
loadusr -Wn sppanel pyvcp -c sppanel panel2.xml
```

Параметр -Wn змушує HAL «Чекати на завантаження імені» перед продовженням.

Команда pyvcp -c змушує PyVCP присвоїти ім’я панелі.

Піни HAL з panel1.xml будуть названі btnpanel.<_назва_піна_>.

Піни HAL з panel2.xml будуть названі sppanel.<_назва_піна_>.

Переконайтеся, що рядок loadusr знаходиться перед будь-якими мережами, які використовують виводи PyVCP.

12.2.3 Приклад кнопок поворотного перемикання

У цьому прикладі ми створимо панель PyVCP з кнопками переміщення для X, Y і Z. Ця конфігурація буде побудована на основі конфігурації, згенерованої майстром StepConf. Спочатку запускаємо майстер StepConf і налаштовуємо нашу машину, на сторінці «Advanced Configuration Options» (Розширені параметри конфігурації) робимо кілька виборів, щоб додати порожню панель PyVCP, як показано на наступному малюнку. Для цього прикладу ми назвали конфігурацію «pyvcp_xyz» на сторінці «Basic Machine Information» (Основна інформація про машину) майстра StepConf.

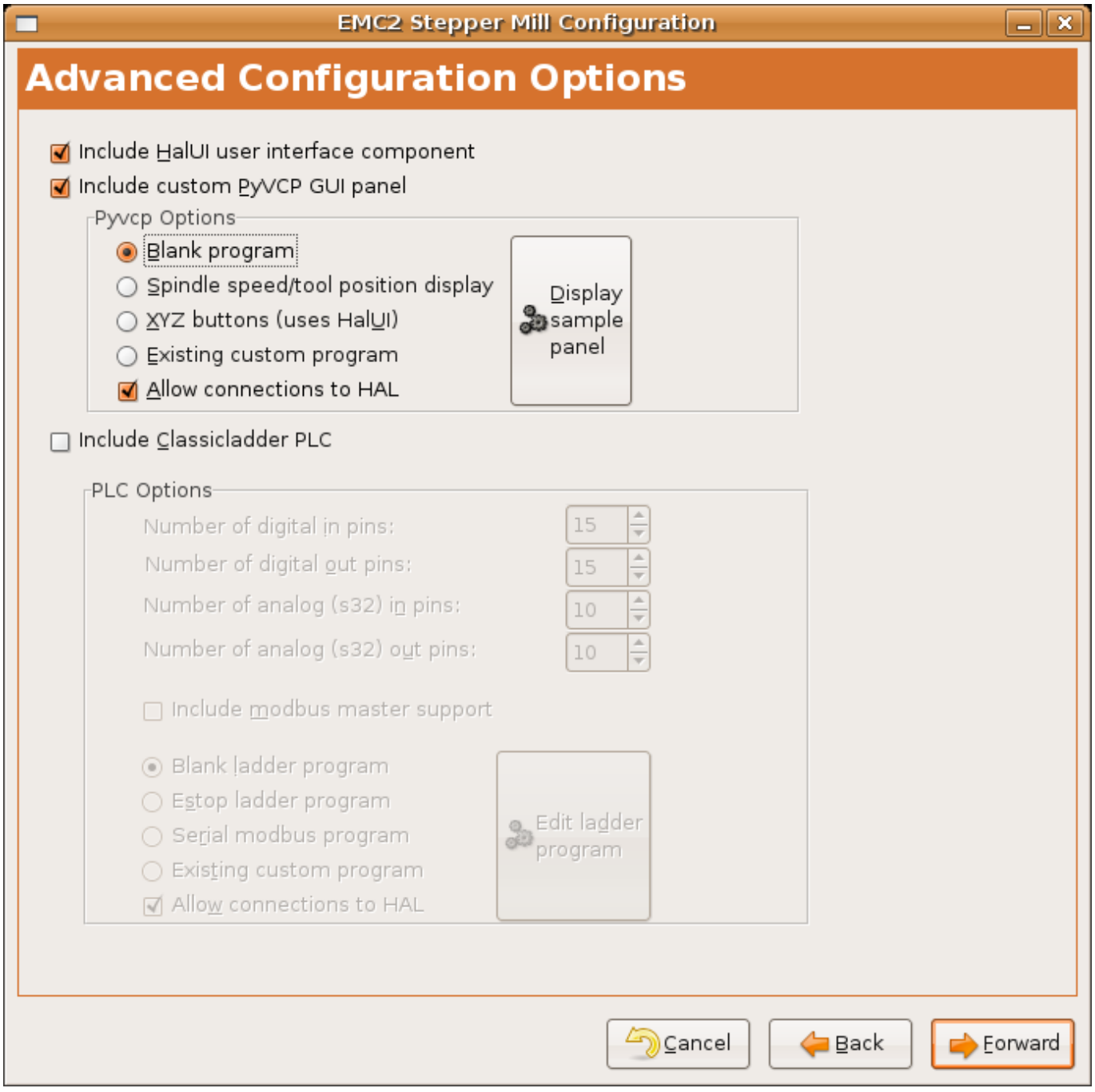


Figure 12.30: Майстер налаштування XYZ

Майстер StepConf створить кілька файлів і розмістить їх у каталозі `linuxcnc/configs/pyvcp_xyz`. Якщо ви залишили посилання для створення позначеним, посилання на ці файли буде відображатися на вашому робочому столі.

12.2.3.1 Створення віджетів

Відкрийте файл custompanel.xml, клацнувши на ньому правою кнопкою миші та вибравши «відкрити в текстовому редакторі». Між тегами <pyvcp></pyvcp> ми додамо віджети для нашої панелі.

Перегляньте розділ «Довідка з віджетів PyVCP» цього посібника для отримання детальнішої інформації про кожен віджет documentation des widgets.

У вашому файлі custompanel.xml ми додамо опис віджетів.

```
<pyvcp>
  <labelframe text="Jog Buttons">
    <font>("Helvetica",16)</font>

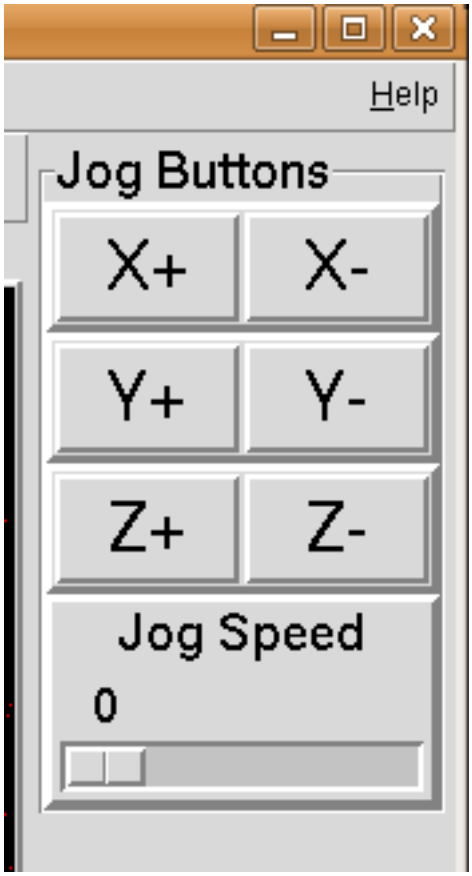
      <!-- the X jog buttons -->
      <hbox>
      <relief>RAISED</relief>
      <bd>3</bd>
      <button>
        <font>("Helvetica",20)</font>
        <width>3</width>
        <halpin>"x-plus"</halpin>
        <text>"X+"</text>
      </button>
      <button>
        <font>("Helvetica",20)</font>
        <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>
```

Після додавання вищезазначеного ви отримаєте панель PyVCP, яка виглядає так, як показано на малюнку праворуч від AXIS. Вона виглядає гарно, але не виконує жодних функцій, доки ви не «підключите» кнопки до halui. Якщо під час спроби запуску з’явиться помилка, прокрутіть вікно до кінця, і, як правило, ви побачите, що це помилка правопису або синтаксису.



12.2.3.2 Встановлюйте зв’язки

Щоб зробити необхідні підключення, відкрийте файл custom_postgui.hal та додайте наступне.

```
# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''кb''b''нb''b' ↩
    'ob''b''пb''b''кb''b''иb'' X PyVCP
net my-jogxminus halui.axis.x.minus <= pyvcp.x-minus
net my-jogxplus halui.axis.x.plus <= pyvcp.x-plus

# b''зb''b''єb''b''дb''b''нb''b''аb''b''йb''b''тb''b''eb'' b''кb''b''нb''b''ob''b''пb''b' ↩
    'кb''b''иb'' Y PyVCP
net my-jogyminus halui.axis.y.minus <= pyvcp.y-minus
net my-jogyplus halui.axis.y.plus <= pyvcp.y-plus

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''кb''b''нb''b' ↩
    'ob''b''пb''b''кb''b''иb'' Z PyVCP
net my-jogzminus halui.axis.z.minus <= pyvcp.z-minus
net my-jogzplus halui.axis.z.plus <= pyvcp.z-plus

# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''ib''b''тb''b''ьb'' b''пb''b''ob''b' ↩
    'вb''b''зb''b''уb''b''нb''b''ob''b''кb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ob''b' ↩
    'cb''b''тb''b''ib'' b''дb''b''жb''b''ob''b''гb''b''уb''b''вb''b''аb''b''нb''b''нb''b' ↩
    'яb'' PyVCP
net my-jogspeed halui.axis.jog-speed <= pyvcp.jog-speed-f
```

Після скидання аварійної зупинки, переведення в режим ручного керування та переміщення повзунка швидкості ручного керування на панелі PyVCP на значення, більше за нуль, кнопки ручного керування PyVCP повинні працювати. Ви не можете керувати ручним керуванням під час виконання файлу G-коду, під час паузи або під час вибору вкладки MDI.

12.2.4 Порт-тестер

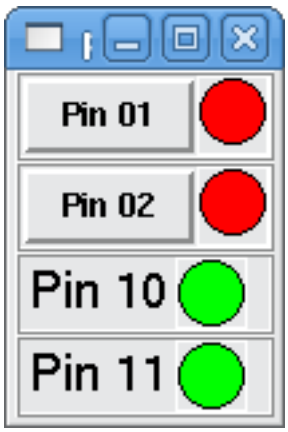
У цьому прикладі показано, як створити простий тестер паралельних портів за допомогою PyVCP та HAL.

Спочатку створіть файл ptest.xml з наступним кодом для створення опису панелі.

```
<!-- 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>
```

Це створить наступну плаваючу панель, яка містить кілька вхідних та кілька вихідних контактів.



Щоб запустити команди HAL, необхідні для запуску всього, ми поміщаємо наступний код у наш файл ptest.hal.

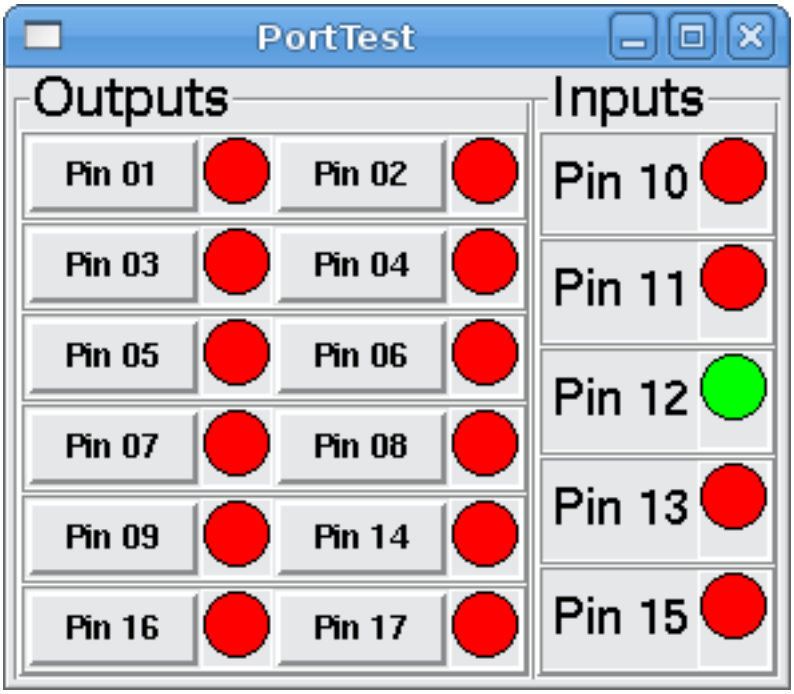
```
loadrt hal_parport cfg=”0x378 out”
loadusr -Wn ptest pyvcp -c ptest ptest.xml
loadrt threads name1=porttest period1=1000000
addf parport.0.read porttest
addf parport.0.write porttest
net pin01 ptest.btn01 parport.0.pin-01-out ptest.led-01
net pin02 ptest.btn02 parport.0.pin-02-out ptest.led-02
net pin10 parport.0.pin-10-in ptest.led-10
```

```
net pin11 parport.0.pin-11-in ptest.led-11
start
```

Щоб запустити HAL-файл, ми використовуємо наступну команду з вікна терміналу.

```
~$ halrun -I -f ptest.hal
```

На наступному малюнку показано, як може виглядати повна панель.



Щоб додати решту контактів паралельного порту, просто змініть файли XML та HAL.

Щоб показати піни після запуску скрипта HAL, використовуйте таку команду в командному рядку halcmd:

```
halcmd: b''пb''b''ob''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''шb''b''тb''b''иb''b' ↩
    'фb''b''тb''
b''Kb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''нb''b''ib'' b''кb''b' ↩
    'ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'':
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
```

Це покаже вам, які контакти ввімкнені, а які — вимкнені, а також будь-які з’єднання.

12.2.5 Вимірювач обертів GS2

У наведеному нижче прикладі використовується драйвер Automation Direct GS2 VDF і відображаються оберти за хвилину та інша інформація на панелі PyVCP. Цей приклад базується на прикладі GS2 у розділі «Приклади апаратного забезпечення» цього посібника.

12.2.5.1 Панель

Щоб створити панель, ми додаємо наступний код до XML-файлу.

```
<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>

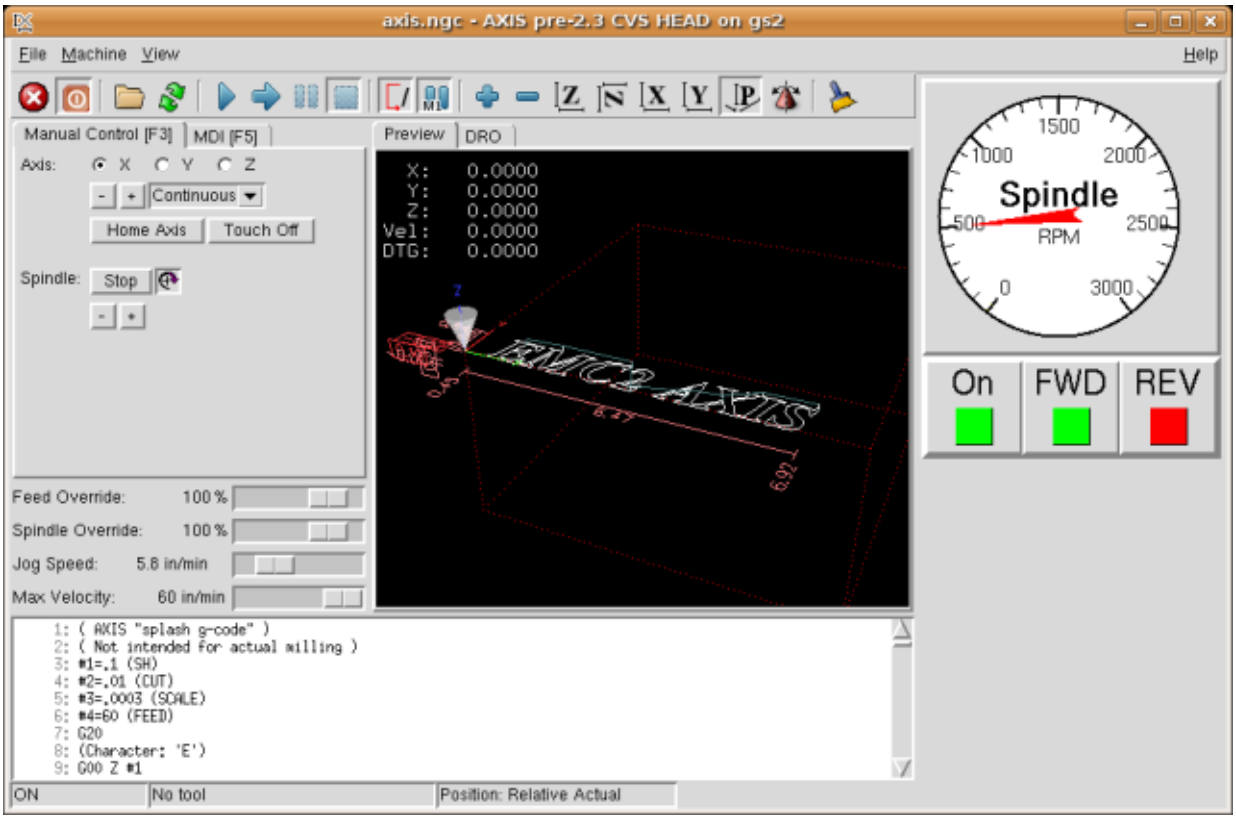
  <!-- the On Led -->
  <hbox>
  <relief>RAISED</relief>
  <bd>3</bd>
  <vbox>
  <relief>RAISED</relief>
  <bd>2</bd>
  <label>
  <text>”On”</text>
  <font>(”Helvetica”,18)</font>
  </label>
  <width>5</width>
   <hbox>
  <label width=”2”/> <!-- used to center the led -->
  <rectled>
  <halpin>”on-led”</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>
```

Вищезазначене дає нам панель PyVCP, яка виглядає наступним чином.



12.2.5.2 Зв’язки

Щоб це запрацювало, ми додаємо наступний код до файлу custom_postgui.hal.

```
# b''вb''b''іb''b''дb''b''оb''b''бb''b''рb''b''аb''b''жb''b''еb''b''нb''b''нb''b''яb'' b' ↩
    'оb''b''бb''b''еb''b''рb''b''тb''b''іb''b''вb'' b''зb''b''аb'' b''хb''b''вb''b''иb''b' ↩
    'лb''b''иb''b''нb''b''уb'' b''нb''b''аb'' b''оb''b''сb''b''нb''b''оb''b''вb''b''іb'' b' ↩
    'чb''b''аb''b''сb''b''тb''b''оb''b''тb''b''иb'' * b''оb''b''бb''b''еb''b''рb''b''тb''b' ↩
    'іb''b''вb'' b''зb''b''аb'' b''хb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb'' b''нb''b' ↩
    'аb'' b''Гb''b''цb''
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

# b''зb''b''аb''b''пb''b''уb''b''сb''b''тb''b''иb''b''тb''b''иb'', b''кb''b''еb''b''рb''b' ↩
    'оb''b''вb''b''аb''b''нb''b''иb''b''йb''
net gs2-run => pyvcp.on-led

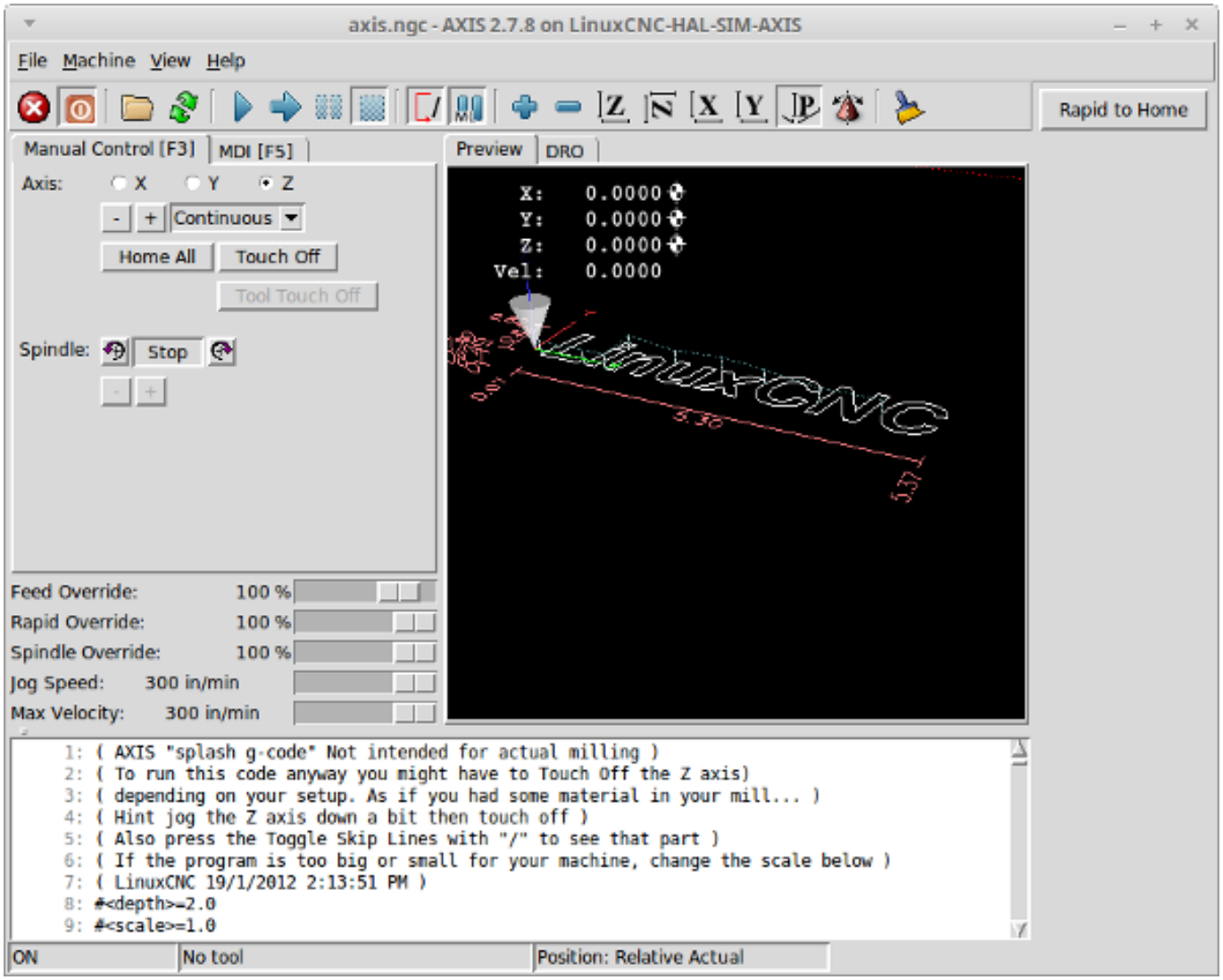
# b''сb''b''вb''b''іb''b''тb''b''лb''b''оb''b''дb''b''іb''b''оb''b''дb''b''нb''b''иb''b' ↩
    'йb'' b''пb''b''еb''b''рb''b''еb''b''мb''b''иb''b''кb''b''аb''b''чb'' b''уb''b''пb''b' ↩
    'еb''b''рb''b''еb''b''дb''
net gs2-fwd => pyvcp.fwd-led

# b''сb''b''вb''b''іb''b''тb''b''лb''b''оb''b''дb''b''іb''b''оb''b''дb''b''нb''b''иb''b' ↩
    'йb'' b''іb''b''нb''b''дb''b''иb''b''кb''b''аb''b''тb''b''оb''b''рb''
net running-rev spindle-vfd.spindle-rev => pyvcp.rev-led
```

Деякі рядки можуть потребувати пояснень. Рядок fwd led використовує сигнал, створений у файлі custom.hal, тоді як rev led потребує використання біта spindle-rev. Ви не можете двічі пов’язати біт spindle-fwd, тому використовуйте сигнал, до якого він був пов’язаний.

12.2.6 Кнопка швидкого переходу додому

У цьому прикладі створюється кнопка на бічній панелі PyVCP, після натискання якої всі осі повертаються в початкове положення. У цьому прикладі припускається, що у вас немає панелі PyVCP.



У каталозі конфігурації створіть XML-файл. У цьому прикладі він має назву «rth.xml». У файл «rth.xml» додайте наступний код для створення кнопки.

```
<pyvcp>
<!-- rapid to home button example -->
<button>
<halpin>”rth-button”</halpin>
<text>”Rapid to Home”</text>
</button>
</pyvcp>
```

Відкрийте ваш INI-файл за допомогою текстового редактора та в розділі [DISPLAY] додайте наступний рядок. Це те, що завантажує панель PyVCP.

```
PYVCP = rth.xml
```

Якщо у вашому INI-файлі немає розділу [HALUI], створіть його та додайте наступну MDI-команду.

```
MDI_COMMAND = G53 G0 X0 Y0 Z0
```

Note
Інформація про G-коди G53 та G0.

У розділі [HAL], якщо у вас немає файлу графічного інтерфейсу post, додайте наступний код та створіть файл з назвою *postgui.hal*.

```
POSTGUI_HALFILE = postgui.hal
```

У файлі *postgui.hal* додайте наступний код, щоб пов’язати кнопку PyVCP з командою MDI.

```
net rth halui.mdi-command-00 <= pyvcp.rth-button
```

Note
Інформація про команду net

12.3 GladeVCP: Віртуальна панель керування Glade

12.3.1 Що таке GladeVCP?

GladeVCP — це компонент LinuxCNC, який додає можливість додавати нову панель інтерфейсу користувача до інтерфейсів користувача LinuxCNC, таких як:

- AXIS
- Доторкливий
- G-екран
- GMOCCAPY

На відміну від PyVCP, GladeVCP не обмежується відображенням і налаштуванням контактів HAL, оскільки в коді Python можна виконувати довільні дії — фактично, за допомогою GladeVCP і Python можна створити повний користувацький інтерфейс LinuxCNC.

GladeVCP використовує редактор інтерфейсу користувача WYSIWYG Glade, який спрощує створення візуально привабливих панелей. Він базується на зв’язках PyGObject з багатим набором віджетів GTK3, і фактично всі ці віджети можуть бути використані в додатку GladeVCP - не тільки спеціалізовані віджети для взаємодії з HAL і LinuxCNC, які описані тут.

12.3.1.1 PyVCP проти GladeVCP: короткий огляд

Обидва підтримують створення панелей з «віджетами HAL» – елементами інтерфейсу користувача, такими як світлодіоди, кнопки, повзунки тощо, значення яких пов’язані з виводом HAL, який, у свою чергу, взаємодіє з рештою LinuxCNC.

PyVCP:

- Набір віджетів: використовує віджети TkInter.

- Створення інтерфейсу користувача: цикл ”редагування XML-файлу / виконання результату / оцінка вигляду”.
- Немає підтримки для вбудовування обробки користувацьких подій.
- Взаємодія з LinuxCNC поза межами вводу/виводу HAL не підтримується.

GladeVCP:

- Набір віджетів: спирається на набір віджетів GTK3.
- Створення інтерфейсу користувача: використовує WYSIWYG-редактор інтерфейсу користувача Glade.
- Будь-яка зміна виводу HAL може бути спрямована на зворотний виклик до визначеного користувачем обробника подій Python.
- Будь-який сигнал GTK (натискання клавіші/кнопки, вікно, введення/виведення, таймер, мережеві події) може бути пов’язаний з користувацькими обробниками в Python.
- Пряма взаємодія з LinuxCNC: виконання довільних команд, наприклад, ініціювання команд MDI для виклику підпрограми G-коду, а також підтримка операцій зміни стану через віджети дій.
- Кілька незалежних панелей GladeVCP можуть бути запущені на різних вкладках.
- Розділення зовнішнього вигляду та функціональності інтерфейсу користувача: зміна зовнішнього вигляду без зміни коду.

12.3.2 Короткий огляд панелі-прикладу

Вікна панелей GladeVCP можна запускати у трьох різних конфігураціях:

- завжди видимий, інтегрований в AXIS з правого боку, точно так само, як і панелі PyVCP,
- як вкладка в AXIS, Touchy, Gscreen або GMOCCAPY; в AXIS це створить третю вкладку, окрім вкладок Preview та DRO, які потрібно викликати явно,
- як окреме вікно верхнього рівня, яке можна іконіфікувати/деіконіфікувати незалежно від головного вікна.

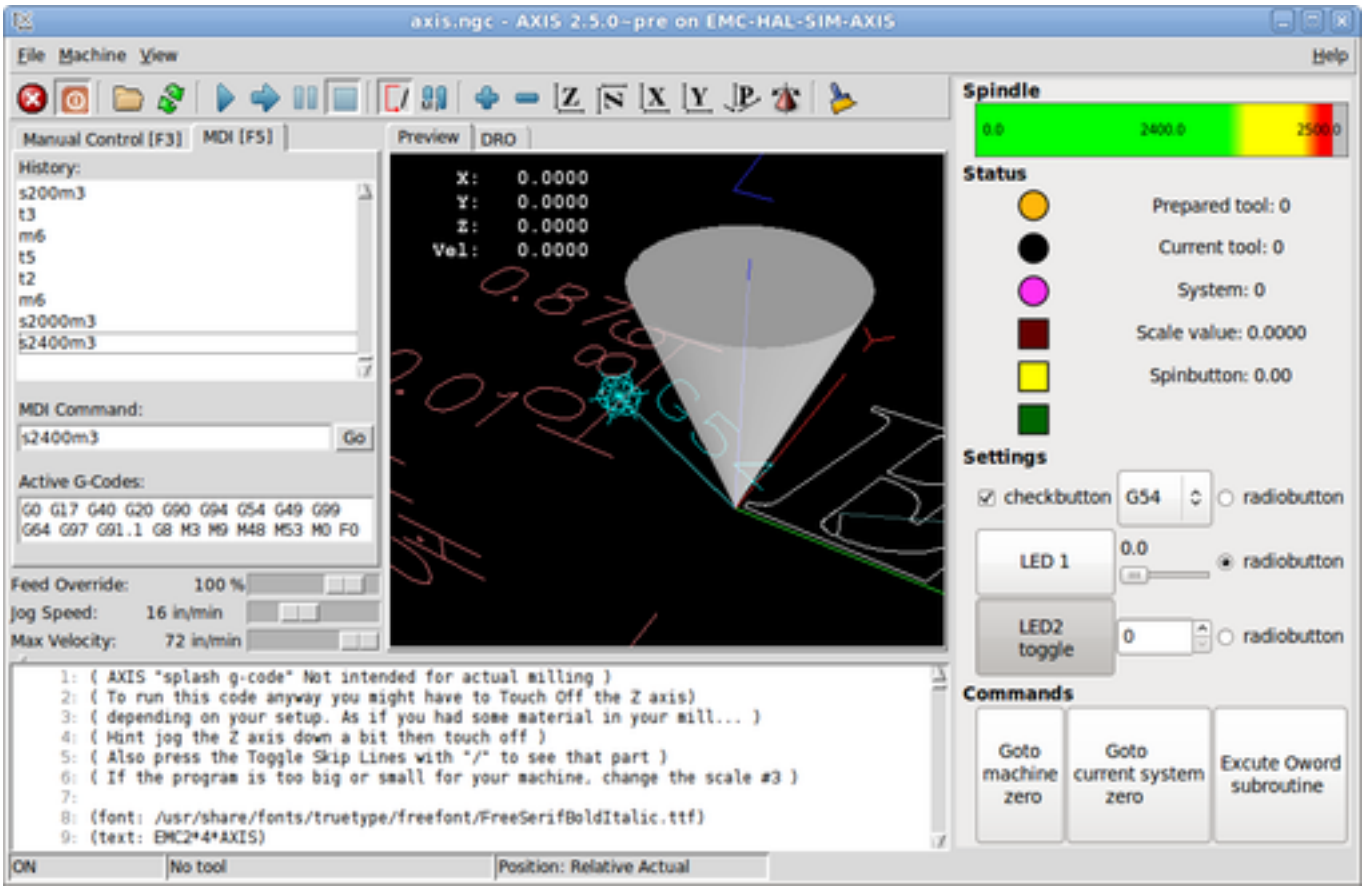
Встановлено LinuxCNC Якщо ви використовуєте встановлену версію LinuxCNC, наведені нижче приклади знаходяться у configuration picker у гілці *Зразки конфігурацій > програми > GladeVCP*.

Оформлення замовлення через Git Наведені нижче інструкції застосовуються лише якщо ви використовуєте git checkout. Відкрийте термінал і перейдіть до каталогу, створеного git, а потім виконайте команди, як показано.

Note
Щоб наступні команди працювали на вашому git checkout, спочатку потрібно запустити *make*, потім *sudo make setuid*, а потім . *./scripts/rip-environment*. Більше інформації про git checkout можна знайти на сторінці LinuxCNC wiki.

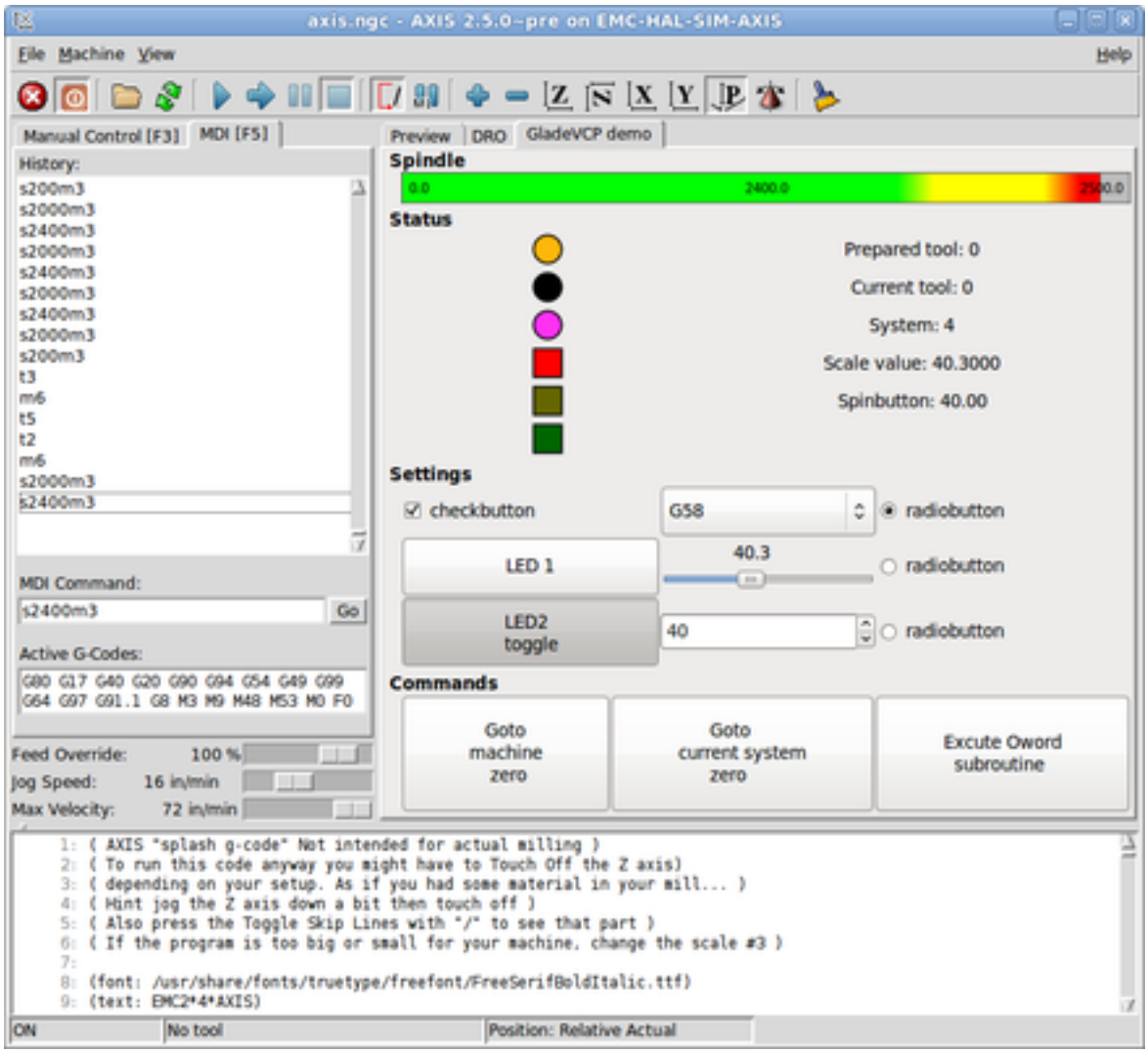
Запустіть зразок панелі GladeVCP, інтегрованої в AXIS, подібно до PyVCP, наступним чином:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_panel.ini
```



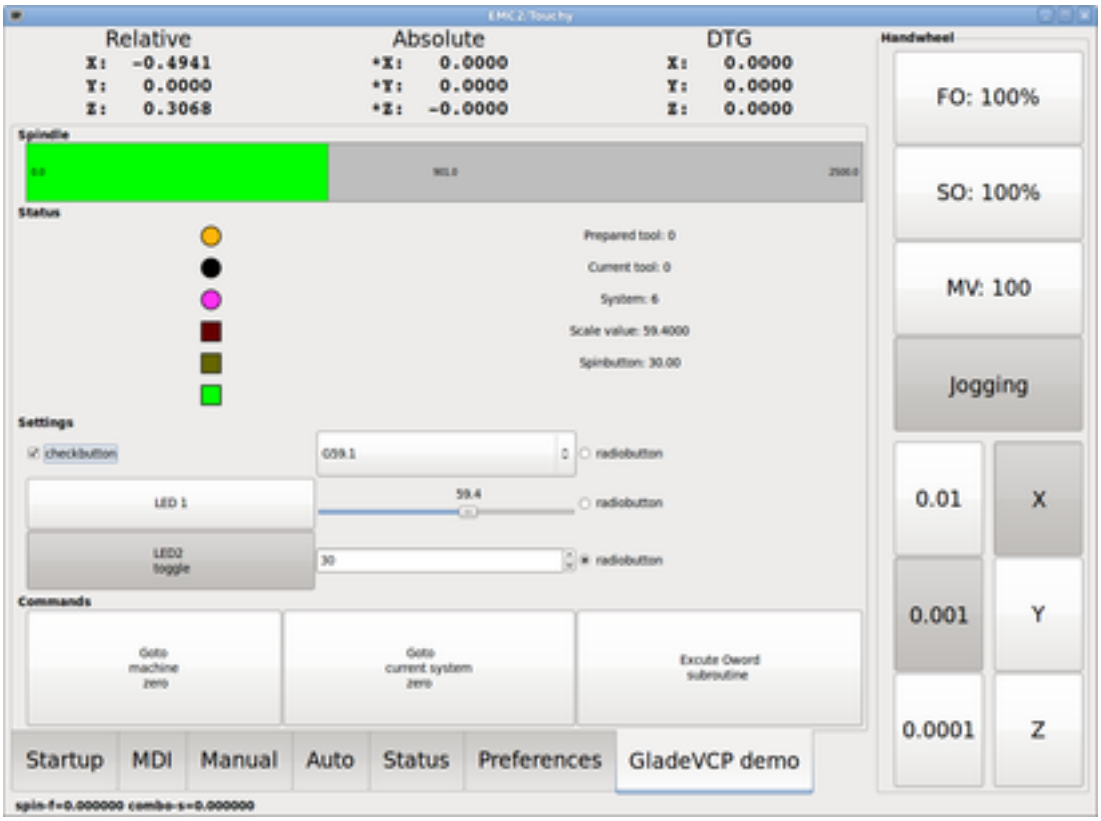
Запустіть ту саму панель, але як вкладку всередині AXIS:

```
$ cd configs/sim/axis/gladevcp
$ linuxcnc gladevcp_tab.ini
```



Щоб запустити цю панель усередині «Touchy»:

```
$ cd configs/sim/touchy/gladevcp
$ linuxcnc gladevcp_touchy.ini
```



Функціонально ці налаштування ідентичні - вони відрізняються лише вимогами до розміру екрану та видимістю. Оскільки можна запускати кілька компонентів GladeVCP паралельно (з різними іменами компонентів HAL), можливі також змішані налаштування - наприклад, панель з правого боку та одна або кілька вкладок для рідше використовуваних частин інтерфейсу.

12.3.2.1 Огляд прикладу панелі

Під час запуску configs/sim/axis/gladevcp_panel.ini або configs/sim/axis/gladevcp_tab.ini, перейдіть до «Show HAL Configuration» (Показати конфігурацію HAL) — ви знайдете компонент HAL `gladevcp` і зможете спостерігати за значеннями його контактів під час взаємодії з віджетами на панелі. Налаштування HAL можна знайти в «configs/axis/gladevcp/manual-example.hal».

Приклад панелі має дві рамки внизу. Панель налаштована таким чином, що скидання ESTOP активує рамку «Налаштування», а ввімкнення машини активує рамку «Команди» внизу. Віджети HAL у рамці «Налаштування» пов’язані зі світлодіодами та мітками у рамці «Статус», а також з поточним і підготовленим номером інструменту — пограйтеся з ними, щоб побачити ефект. Виконання команд «T<номер інструменту>» та «M6» у вікні MDI змінить поля поточного та підготовленого номера інструменту.

Кнопки в рамці «Команди» є «віджетами дій MDI» — натискання на них призведе до виконання команди MDI в інтерпретаторі. Третя кнопка «Виконати підпрограму Oword» є розширеним прикладом — вона бере кілька значень виводів HAL з рамки «Налаштування» і передає їх як параметри до підпрограми Oword. Фактичні параметри, отримані підпрограмою, відображаються командами «(DEBUG,)» — див. «../../nc_files/oword.ngc» для тіла підпрограми.

Щоб побачити, як панель інтегрована в AXIS, дивіться оператор *[DISPLAY]GLADEVCP* у файлі configs/sim/axis/gladevcp/gladevcp_panel.ini, оператор ’ [DISPLAY]EMBED*» у файлі configs/sim/axis/gladevcp/gladevcp_tab.ini та оператори «[HAL]POSTGUI_HALFILE» у файлах configs/sim/axis/gladevcp/gladevcp_tab.ini та configs/sim/axis/gladevcp/gladevcp_panel.ini.

12.3.2.2 Огляд опису інтерфейсу користувача

Користувацький інтерфейс створюється за допомогою редактора Glade UI — щоб його ознайомитися, потрібно встановити Glade. Щоб редагувати користувацький інтерфейс, виконайте команду

```
$ glade configs/axis/gladevcp/manual-example.ui
```

На новіших системах необхідна програма glade може називатися glade-gtk2.

У центральному вікні відображається зовнішній вигляд інтерфейсу користувача. Усі об'єкти інтерфейсу користувача та об'єкти підтримки знаходяться у правому верхньому вікні, де ви можете вибрати конкретний віджет (або натиснувши на нього в центральному вікні). Властивості вибраного віджета відображаються і можуть бути змінені в правому нижньому вікні.

Щоб побачити, як команди MDI передаються з віджетів MDI Action, перегляньте віджети, перелічені в розділі «Actions» у верхньому правому вікні, а також у правому нижньому вікні, на вкладці «General», у властивості «MDI command».

12.3.2.3 Вивчення зворотного виклику Python

Подивіться, як зворотний виклик Python інтегровано в приклад:

- У Glade дивіться віджет мітки `hits` (звичайний віджет GTK+).
- У віджеті `button1` перегляньте вкладку «Сигнали» та знайдіть сигнал «натиснуто», пов'язаний з обробником «on_button_press».
- У hitcounter.py подивіться на метод *on_button_press* та як він встановлює властивість label в об'єкті *hits*.

Це лише торкається концепції — механізм зворотного виклику буде розглянуто детальніше в розділі Програмування GladeVCP.

12.3.3 Створення та інтеграція користувацького інтерфейсу Glade

12.3.3.1 Передумова: встановлення Glade

Щоб переглянути або змінити файли інтерфейсу користувача Glade, вам потрібно встановити Glade 3.38.2 або пізнішої версії — він потрібен не лише для запуску панелі GladeVCP. Якщо команда «glade» відсутня, встановіть її за допомогою команди:

```
$ sudo apt install glade
```

Потім перевірте встановлену версію, яка має бути дорівнює або перевищувати 3.6.7:

```
$ glade --version
```

Glade містить внутрішній інтерпретатор Python, і підтримується тільки Python 3. Це стосується Debian Bullseye, Ubuntu 21 і Mint 21 або пізніших версій. Старіші версії не працюватимуть, ви отримаєте помилку Python.

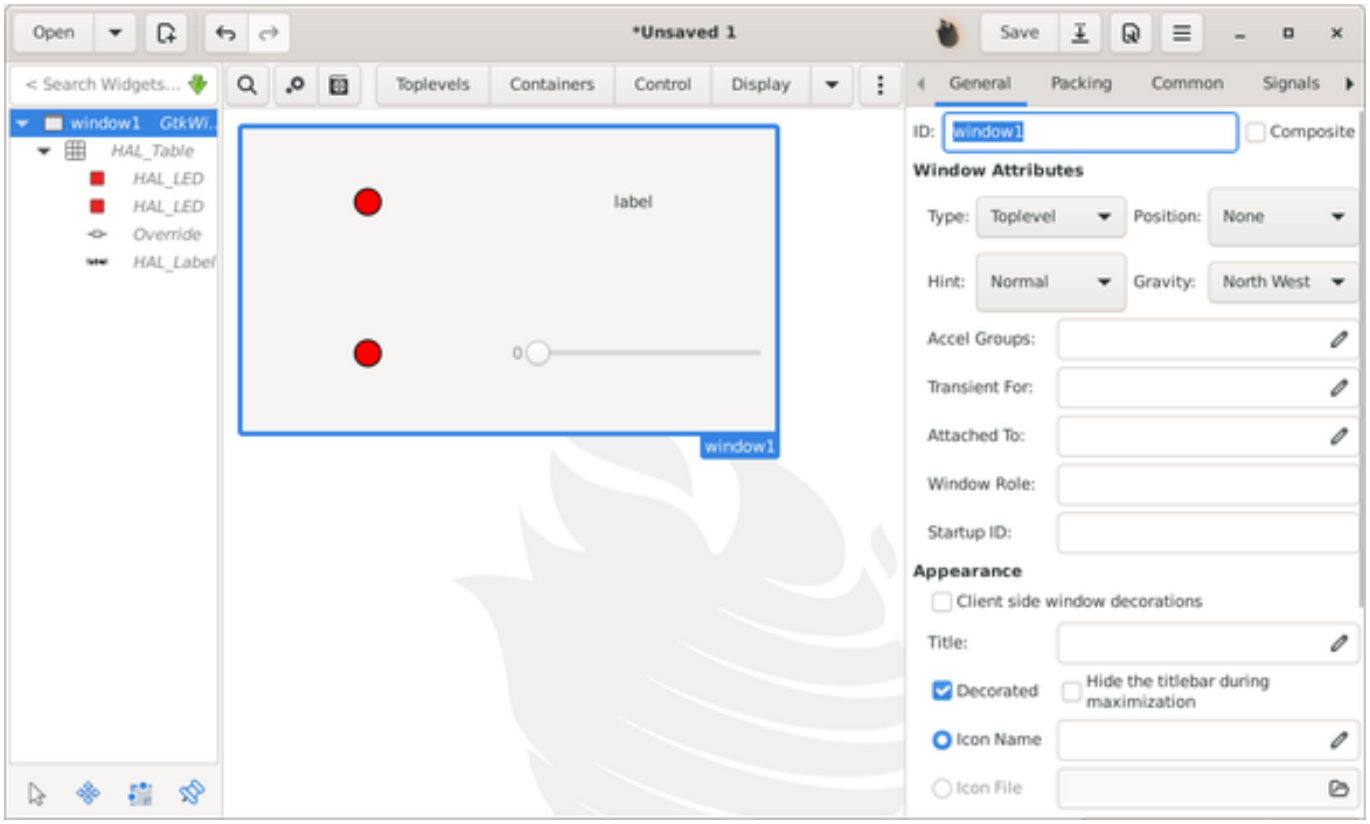
12.3.3.2 Запуск Glade для створення нового інтерфейсу користувача

У цьому розділі описано лише початкові кроки, специфічні для LinuxCNC. Більш детальну інформацію та підручник з Glade можна знайти на сайті http://glade.gnome.org. Деякі поради та підказки щодо Glade також можна знайти на сайті YouTube.

Або змініть існуючий компонент інтерфейсу, запустивши `glade <файл>.ui`, або запустіть новий, просто виконавши команду `glade` з оболонки.

- Якщо LinuxCNC не було встановлено з пакета, середовище оболонки LinuxCNC потрібно налаштувати за допомогою команди `source <linuxcncdir>/scripts/rip-environment`, інакше Glade не знайде специфічні для LinuxCNC віджети.
- Коли вас запитають про незбережені налаштування, просто прийміть значення за замовчуванням і натисніть «Закрити».
- На панелі інструментів «Toplevels» виберіть «GtkWindow» (перший запис) як вікно верхнього рівня. Встановіть «window1» як ідентифікатор у правій панелі під вкладкою «General». Це найменування важливе, оскільки GladeVCP залежить від нього.
- За допомогою кнопки з трьома крапками ви можете знайти специфічні віджети LinuxCNC.
- Додайте контейнер, такий як HAL_Box або HAL_Table з *HAL Python*, до фрейму.
- Виберіть та розмістіть деякі елементи, такі як світлодіод, кнопка тощо, в контейнері.

Це виглядатиме так:



Glade має тенденцію писати багато повідомлень у вікно оболонки, які в основному можна ігнорувати. Виберіть «Файл»→«Зберегти як», дайте йому ім'я, наприклад «myui.ui», і переконайтеся, що він збережений як файл «GtkBuilder» (кнопка в лівому нижньому куті діалогового вікна «Зберегти»). GladeVCP також правильно обробляє старий формат «libglade», але немає сенсу його використовувати. Стандартним розширенням файлів GtkBuilder є «.ui».

12.3.3.3 Тестування панелі

Тепер ви готові спробувати (поки запущено LinuxCNC, наприклад, AXIS) за допомогою:

```
gladevcp myui.ui
```

GladeVCP створює компонент HAL з іменем, що відповідає базовому імені файлу інтерфейсу користувача — в даному випадку «myui» — якщо це не замінено опцією `-c <ім'я компонента>`. Якщо ви використовуєте AXIS, просто спробуйте «Показати конфігурацію HAL» і перевірте його контакти.

Ви можете задатися питанням, чому віджети, що містять «HAL_Hbox» або «HAL_Table», відображаються сірим кольором (неактивними). Контейнери HAL мають пов'язаний з ними контакт HAL, який за замовчуванням вимкнений, що призводить до того, що всі віджети, що містяться в них, відображаються як неактивні. Типовим випадком використання є пов'язування цих контактів контейнерів HAL із сигналом «halui.machine.is-on» або одним із сигналів «halui.mode», щоб забезпечити активність деяких віджетів лише в певному стані.

Щоб просто активувати контейнер, виконайте команду HAL `setp gladevcp.<ім'я-контейнера> 1`.

12.3.3.4 Підготовка файлу команд HAL

Рекомендований спосіб зв'язування контактів HAL у панелі GladeVCP — зібрати їх в окремому файлі з розширенням «.hal». Цей файл передається через опцію `POSTGUI_HALFILE=` у розділі `HAL` вашого файлу INI.



Caution
Не додавайте командний файл GladeVCP HAL до розділу AXIS `[HAL]HALFILE=` ini, це не матиме бажаного ефекту – див. наступні розділи.

12.3.3.5 Інтеграція в AXIS, як-от PyVCP

Розмістіть панель GladeVCP на правій бічній панелі, вказавши наступне у файлі INI:

```
[DISPLAY]
# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''пb''b''ab''b''нb''b''eb''b''лb''b''ьb''  ↩
    GladeVCP b''тb''b''ab''b''мb'', b''дb''b''eb'' b''pb''b''ab''b''нb''b''ib''b''шb''b' ↩
    'eb'' b''pb''b''ob''b''зb''b''мb''b''ib''b''щb''b''yb''b''вb''b''ab''b''вb''b''cb''b' ↩
    'яb'' PyVCP:
GLADEVCP= -u ./hitcounter.py ./manual-example.ui

[HAL]
# b''Kb''b''ob''b''мb''b''ab''b''нb''b''дb''b''иb'' HAL b''дb''b''лb''b''яb'' b''кb''b' ↩
    'ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''ib''b''вb'' GladeVCP b''yb'' b' ↩
    'вb''b''кb''b''лb''b''ab''b''дb''b''цb''b''ib'' b''мb''b''ab''b''юb''b''тb''b''ьb'' b' ↩
    'вb''b''иb''b''кb''b''ob''b''нb''b''yb''b''вb''b''ab''b''тb''b''иb''b''cb''b''яb'' b' ↩
    'чb''b''eb''b''pb''b''eb''b''зb'' POSTGUI_HALFILE
POSTGUI_HALFILE =  ./manual-example.hal

[RS274NGC]
# b''Tb''b''yb''b''тb'' b''pb''b''ob''b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''ib'' b' ↩
    'cb''b''yb''b''бb''b''тb''b''иb''b''тb''b''pb''b''иb'' Oword, b''cb''b''пb''b''eb''b' ↩
    'цb''b''иb''b''фb''b''ib''b''чb''b''нb''b''ib'' b''дb''b''лb''b''яb'' b''дb''b''eb''b' ↩
    'мb''b''ob''b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' gladevcp
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Ім’я компонента HAL за замовчуванням для програми GladeVCP, запущеної з опцією GLADEVCP, таке: `gladevcp`.

Командний рядок, який фактично виконує AXIS у вищезгаданій конфігурації, виглядає наступним чином:

```
halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID} -u ./hitcounter.py ./manual- ↩
    example.ui
```

Ви можете додавати тут довільні параметри `gladevcp`, якщо вони не конфліктують із вищезазначеними параметрами командного рядка.

Можна створити власну назву компонента HAL, додавши параметр `-c`:

```
[DISPLAY]
# b’’дb’’b’’ob’’b’’дb’’b’’ab’’b’’тb’’b’’иb’’ b’’пb’’b’’ab’’b’’нb’’b’’eb’’b’’лb’’b’’ьb’’  ↩
    GladeVCP b’’тb’’b’’ab’’b’’мb’’, b’’дb’’b’’eb’’ b’’pb’’b’’ab’’b’’нb’’b’’іb’’b’’шb’’b’ ↩
    ’eb’’ b’’pb’’b’’ob’’b’’зb’’b’’мb’’b’’іb’’b’’щb’’b’’уb’’b’’вb’’b’’ab’’b’’вb’’b’’cb’’b’ ↩
    ’яb’’ PyVCP:
GLADEVCP= -c example -u ./hitcounter.py ./manual-example.ui
```

Командний рядок, який фактично виконує AXIS для вищезазначеного:

```
halcmd loadusr -Wn example gladevcp -c example -x {XID} -u ./hitcounter.py ./manual-example ↩
    .ui
```

Note
Позначення файлів, такі як ./hitcounter.py, ./manual-example.ui тощо, вказують, що файли знаходяться в тому ж каталозі, що й файл INI. Можливо, вам доведеться скопіювати їх до свого каталогу (або вказати правильний абсолютний чи відносний шлях до файлу(-ів)).

Note
Параметр `[RS274NGC]SUBROUTINE_PATH=` встановлюється лише для того, щоб панель прикладів могла знайти підпрограму Oword (oword.ngc) для віджета MDI Command. У вашій конфігурації він може бути непотрібним. Відносний шлях ../../nc_files/gladevcp_lib побудований для роботи з каталогами, скопійованими за допомогою конфігураційного селектора, та при використанні налаштування «run-in-place».

12.3.3.6 Вбудовування як вкладка

Для цього відредагуйте ваш INI-файл та додайте до розділів DISPLAY та HAL наступне:

```
[DISPLAY]
# b’’дb’’b’’ob’’b’’дb’’b’’ab’’b’’йb’’b’’тb’’b’’eb’’ b’’пb’’b’’ab’’b’’нb’’b’’eb’’b’’лb’’b’ ↩
    ’ьb’’ GladeVCP b’’яb’’b’’кb’’ b’’вb’’b’’кb’’b’’лb’’b’’ab’’b’’дb’’b’’кb’’b’’уb’’ b’’пb’’b ↩
    ’’ob’’b’’pb’’b’’уb’’b’’чb’’ b’’іb’’b’’зb’’ «b’’Пb’’b’’ob’’b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’ ↩
    ’дb’’b’’нb’’b’’іb’’b’’йb’’ b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’гb’’b’’лb’’b’’яb’’b’’дb’’/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]
# b’’Kb’’b’’ob’’b’’мb’’b’’ab’’b’’нb’’b’’дb’’b’’иb’’ HAL b’’дb’’b’’лb’’b’’яb’’ b’’кb’’b’ ↩
    ’ob’’b’’мb’’b’’пb’’b’’ob’’b’’нb’’b’’eb’’b’’нb’’b’’тb’’b’’іb’’b’’вb’’ GladeVCP b’’уb’’ b’ ↩
    ’вb’’b’’кb’’b’’лb’’b’’ab’’b’’дb’’b’’цb’’b’’іb’’ b’’мb’’b’’ab’’b’’юb’’b’’тb’’b’’ьb’’ b’ ↩
    ’вb’’b’’иb’’b’’кb’’b’’ob’’b’’нb’’b’’уb’’b’’вb’’b’’ab’’b’’тb’’b’’иb’’b’’cb’’b’’яb’’ b’ ↩
    ’чb’’b’’eb’’b’’pb’’b’’eb’’b’’зb’’ POSTGUI_HALFILE
```

```
POSTGUI_HALFILE =  ./gladevcp/manual-example.hal

[RS274NGC]
# b''Tb''b''yb''b''тb'' b''pb''b''ob''b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''ib'' b' ↩
    'cb''b''yb''b''бb''b''тb''b''иb''b''тb''b''pb''b''иb'' Oword, b''cb''b''пb''b''eb''b' ↩
    'цb''b''иb''b''фb''b''ib''b''чb''b''нb''b''ib'' b''дb''b''лb''b''яb'' b''дb''b''eb''b' ↩
    'мb''b''ob''b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' gladevcp
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Зверніть увагу на спосіб запуску команди tab за допомогою `halcmd loadusr` — це гарантує, що *POSTGUI_HALFILE* буде запущено лише після того, як компонент HAL буде готовий. У рідкісних випадках ви можете запустити тут команду, яка використовує вкладку, але не має пов'язаного компонента HAL. Таку команду можна запустити без «halcmd loadusr», і це означає для AXIS, що йому не потрібно чекати на компонент HAL, оскільки його немає.

Змінюючи назву компонента у наведеному вище прикладі, зверніть увагу, що назви, що використовуют
в `-Wn <компонент>` та `-c <компонент>`, мають бути ідентичними.

Спробуйте, запустивши AXIS — поруч із вкладкою DRO має з'явитися нова вкладка під назвою «GladeVCP demo». Виберіть цю вкладку, і ви побачите приклад панелі, який чудово вписується в AXIS.

Note
Переконайтеся, що файл інтерфейсу користувача є останнім параметром, переданим до Glade-VCP в обох операторах `GLADEVCP=` та `EMBED_TAB_COMMAND=`.

12.3.3.7 Інтеграція в Touchy

Щоб додати вкладку GladeVCP до «Touchy», відредагуйте свій INI-файл наступним чином:

```
[DISPLAY]
# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''пb''b''ab''b''нb''b''eb''b''лb''b''ьb''  ↩
    GladeVCP b''яb''b''кb'' b''вb''b''кb''b''лb''b''ab''b''дb''b''кb''b''yb''
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=gladevcp -c gladevcp -x {XID} -u ./hitcounter.py -H ./gladevcp-touchy.hal ↩
     ./manual-example.ui

[RS274NGC]
# b''Tb''b''yb''b''тb'' b''pb''b''ob''b''зb''b''мb''b''ib''b''щb''b''eb''b''нb''b''ib'' b' ↩
    'cb''b''yb''b''бb''b''тb''b''иb''b''тb''b''pb''b''иb'' Oword, b''cb''b''пb''b''eb''b' ↩
    'цb''b''иb''b''фb''b''ib''b''чb''b''нb''b''ib'' b''дb''b''лb''b''яb'' b''дb''b''eb''b' ↩
    'мb''b''ob''b''вb''b''eb''b''pb''b''cb''b''ib''b''їb'' gladevcp
SUBROUTINE_PATH = ../../nc_files/gladevcp_lib
```

Note
Позначення файлів, такі як ./hitcounter.py, ./manual-example.ui тощо, вказують, що файли знаходяться в тому ж каталозі, що й файл INI. Можливо, вам доведеться скопіювати їх до свого каталогу (або вказати правильний абсолютний чи відносний шлях до файлу(-ів)).

Зверніть увагу на такі відмінності в налаштуваннях вкладки AXIS:

- Файл команд HAL дещо змінено, оскільки «Touchy» не використовує компоненти «halui», тому його сигнали недоступні, а також було використано деякі скорочення.
- Немає опції INI 'POSTGUI_HALFILE=", але передача файлу команд HAL у рядку 'EMBED_TAB_COMM
є прийнятною.
- Заклинання *halcmd loaduser -Wn ...* не потрібне.

12.3.3.8 Завантаження вбудованих панелей

У системі є вбудовані панелі, які завантажуються дещо іншим способом.
До назви панелі не додається розширення файлу.
Якщо для панелі потрібен файл користувача (опція -u), вона завантажиться автоматично.
Назви вбудованих панелей можна переглянути, виконавши лише команду gladevcp у терміналі.

Завантаження вбудованої панелі зонда verser:

```
gladevcp gtk_verser_probe
```

Вбудовування те саме, без розширення файлу, але інші опції підходять:

```
[DISPLAY]
# b''дb''b''ob''b''дb''b''ab''b''йb''b''тb''b''eb'' b''пb''b''ab''b''нb''b''eb''b''лb''b' ↩
    'ьb'' GladeVCP b''яb''b''кb'' b''вb''b''кb''b''лb''b''ab''b''дb''b''кb''b''уb'' b''пb''b ↩
    ''ob''b''pb''b''уb''b''чb'' b''іb''b''зb'' «b''Пb''b''ob''b''пb''b''eb''b''pb''b''eb''b' ↩
    'дb''b''нb''b''іb''b''йb'' b''пb''b''eb''b''pb''b''eb''b''гb''b''лb''b''яb''b''дb''/DRO» ↩
    :
EMBED_TAB_NAME=GladeVCP demo
EMBED_TAB_COMMAND=halcmd loadusr -Wn gladevcp gladevcp -c gladevcp -x {XID}  ↩
    gtk_verser_probe
```

12.3.4 Параметри командного рядка GladeVCP

Ось параметри командного рядка GladeVCP:
(Див. також `man gladevcp`.)
Якщо ви введете gladevcp у терміналі, ось що ви побачите:

```
b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':  ↩
    gladevcp [options] myfile.ui
b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'':  ↩
    gladevcp [options] built_in_panel_name

b''Пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'':
  -h, --help      b''пb''b''ob''b''кb''b''ab''b''зb''b''ab''b''тb''b''иb'' b''цb''b''eb'' b ↩
      ''пb''b''ob''b''вb''b''іb''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b' ↩
      'дb''b''ob''b''вb''b''іb''b''дb''b''кb''b''иb'' b''тb''b''ab'' b''вb''b''иb''b''йb''b' ↩
      'тb''b''иb''
  -c NAME         b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
      'іb''b''мb''b''яb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b' ↩
      'ab'' b''яb''b''кb'' NAME. b''Зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b' ↩
      'уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'' b''вb''b''иb''b''кb''b''ob''b''pb''b' ↩
      'иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''бb''b' ↩
      'ab''b''зb''b''ob''b''вb''b''ab'' b''нb''b''ab''b''зb''b''вb''b''ab'' b''фb''b''ab''b' ↩
      'йb''b''лb''b''уb'' b''іb''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b''cb''b' ↩
      'уb'' b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b''ab''b''чb''b''ab''.
  -d              b''Уb''b''вb''b''іb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''вb''b' ↩
      'иb''b''вb''b''eb''b''дb''b''eb''b''нb''b''нb''b''яb'' b''іb''b''нb''b''фb''b''ob''b' ↩
      'pb''b''мb''b''ab''b''цb''b''іb''b''їb'' b''дb''b''лb''b''яb'' b''нb''b''ab''b''лb''b' ↩
      'ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb''.
  -g GEOMETRY     b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
      'гb''b''eb''b''ob''b''мb''b''eb''b''тb''b''pb''b''іb''b''юb'' WIDTHxHEIGHT+XOFFSET+ ↩
      YOFFSET. b''Зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''вb''b''кb''b' ↩
      'ab''b''зb''b''ab''b''нb''b''іb'' b''вb''
                  b''пb''b''іb''b''кb''b''cb''b''eb''b''лb''b''яb''b''xb'', XOFFSET/YOFFSET ↩
                       b''вb''b''іb''b''дb''b''лb''b''іb''b''чb''b''уb''b''юb''b''тb''b' ↩
                      'ьb''b''cb''b''яb'' b''вb''b''іb''b''дb'' b''вb''b''eb''b''pb''b''xb'' ↩
                      b''нb''b''ьb''b''ob''b''гb''b''ob'' b''лb''b''іb''b''вb''b''ob''b' ↩
                      'гb''b''ob'' b''кb''b''уb''b''тb''b''ab''
```

```
                b''eb''b''кb''b''pb''b''ab''b''нb''b''yb''. b''Bb''b''иb''b''кb''b''ob''b ↩
                    ''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''йb''b''тb''b''eb''  ↩
                    -g WIDTHxHEIGHT b''дb''b''лb''b''яb'' b''вb''b''cb''b''тb''b''ab''b' ↩
                    'нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''лb''b''иb''b ↩
                    ''шb''b''eb'' b''pb''b''ob''b''зb''b''мb''b''іb''b''pb''b''yb'' b' ↩
                    'ab''b''бb''b''ob'' -g
                +XOFFSET+YOFFSET b''дb''b''лb''b''яb'' b''вb''b''cb''b''тb''b''ab''b' ↩
                    'нb''b''ob''b''вb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''лb''b''иb''b ↩
                    ''шb''b''eb'' b''пb''b''ob''b''лb''b''ob''b''жb''b''eb''b''нb''b''нb'' ↩
                    b''яb''.
-H FILE          b''вb''b''иb''b''кb''b''ob''b''нb''b''ab''b''тb''b''иb'' b''кb''b''ob''b' ↩
    'мb''b''ab''b''нb''b''дb''b''иb'' hal b''зb'' FILE b''зb''b''ab'' b''дb''b''ob''b' ↩
    'пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' halcmd b''пb''b''іb''b''cb''b''лb''b' ↩
    'яb'' b''тb''b''ob''b''гb''b''ob'', b''яb''b''кb''
                b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb'' b''нb''b' ↩
                    'ab''b''лb''b''ab''b''шb''b''тb''b''ob''b''вb''b''ab''b''нb''b''иb''b' ↩
                    'йb'' b''іb'' b''гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''йb'' b' ↩
                    'дb''b''ob'' b''pb''b''ob''b''бb''b''ob''b''тb''b''иb''.
-i               b''Уb''b''вb''b''іb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''вb''b' ↩
    'иb''b''вb''b''eb''b''дb''b''eb''b''нb''b''нb''b''яb'' b''іb''b''нb''b''фb''b''ob''b' ↩
    'pb''b''мb''b''ab''b''цb''b''іb''b''їb''
-m MAXIMUM       b''Пb''b''pb''b''иb''b''мb''b''yb''b''cb''b''ob''b''вb''b''ob'' b''мb''b' ↩
    'ab''b''кb''b''cb''b''иb''b''мb''b''іb''b''зb''b''yb''b''вb''b''ab''b''тb''b''иb'' b' ↩
    'вb''b''іb''b''кb''b''нb''b''ob'' b''пb''b''ab''b''нb''b''eb''b''лb''b''іb''
-q               b''Уb''b''вb''b''іb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''тb''b' ↩
    'іb''b''лb''b''ьb''b''кb''b''иb'' b''вb''b''иb''b''вb''b''eb''b''дb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''іb''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''іb''b''їb'' b' ↩
    'пb''b''pb''b''ob'' b''пb''b''ob''b''мb''b''иb''b''лb''b''кb''b''иb''
-r GTK_RC        b''Пb''b''pb''b''ob''b''чb''b''иb''b''тb''b''ab''b''тb''b''иb'' b''вb''b' ↩
    'лb''b''ab''b''cb''b''нb''b''иb''b''йb'' b''фb''b''ab''b''йb''b''лb'' GTK rc b''дb''b' ↩
    'лb''b''яb'' b''нb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b' ↩
    'нb''b''яb'' b''cb''b''тb''b''иb''b''лb''b''юb'' b''вb''b''іb''b''дb''b''жb''b''eb''b' ↩
    'тb''b''ab''
-t THEME         b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b' ↩
    'тb''b''eb''b''мb''b''yb'' gtk. b''Зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b' ↩
    'чb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'' b''вb''b''иb''b''кb''b''ob''b' ↩
    'pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''нb''b''ab'' b''тb''b''eb''b''мb''b''ab''
-x XID           b''Пb''b''eb''b''pb''b''eb''b''пb''b''іb''b''дb''b''кb''b''лb''b''юb''b' ↩
    'чb''b''иb''b''тb''b''иb'' gladevcp b''дb''b''ob'' b''іb''b''cb''b''нb''b''yb''b''юb'' ↩
    b''чb''b''ob''b''гb''b''ob'' b''вb''b''іb''b''кb''b''нb''b''ab'' XID b''зb''b''ab''b' ↩
    'мb''b''іb''b''cb''b''тb''b''ьb''
                b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b''нb''b''нb''b''яb'' b''нb''b' ↩
                    'ob''b''вb''b''ob''b''гb''b''ob'' b''вb''b''іb''b''кb''b''нb''b''ab''  ↩
                    b''вb''b''eb''b''pb''b''xb''b''нb''b''ьb''b''ob''b''гb''b''ob'' b' ↩
                    'pb''b''іb''b''вb''b''нb''b''яb''
--xid            b''пb''b''eb''b''pb''b''eb''b''пb''b''іb''b''дb''b''кb''b''лb''b''юb''b' ↩
    'чb''b''иb''b''тb''b''иb'' b''вb''b''іb''b''кb''b''нb''b''ob'' b''дb''b''ob'' b''пb''b ↩
    ''лb''b''ab''b''гb''b''іb''b''нb''b''ab'' b''тb''b''ab'' b''вb''b''іb''b''дb''b''пb''b ↩
    ''pb''b''ab''b''вb''b''иb''b''тb''b''иb'' b''нb''b''ob''b''мb''b''eb''b''pb'' xid b' ↩
    'пb''b''лb''b''ab''b''гb''b''іb''b''нb''b''ab'' b''дb''b''ob''
                b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b''нb''b''ob''b' ↩
                    'гb''b''ob'' b''вb''b''иb''b''вb''b''ob''b''дb''b''yb''
-u FILE          b''Bb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b' ↩
    'yb''b''вb''b''ab''b''тb''b''иb'' FILEs b''яb''b''кb'' b''дb''b''ob''b''дb''b''ab''b' ↩
    'тb''b''кb''b''ob''b''вb''b''іb'' b''мb''b''ob''b''дb''b''yb''b''лb''b''іb'', b''вb''b ↩
    ''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''іb'' b''кb''b''ob''b''pb''b''иb''b' ↩
    'cb''b''тb''b''yb''b''вb''b''ab''b''чb''b''eb''b''мb'', b''зb'' b''ob''b''бb''b''pb''b ↩
    ''ob''b''бb''b''нb''b''иb''b''кb''b''ab''b''мb''b''иb''
-U USEROPT       b''пb''b''eb''b''pb''b''eb''b''дb''b''ab''b''тb''b''иb'' USEROPTs b''дb'' ↩
    b''ob'' b''мb''b''ob''b''дb''b''yb''b''лb''b''іb''b''вb'' Python
-v               b''Уb''b''вb''b''іb''b''мb''b''кb''b''нb''b''yb''b''тb''b''иb'' b''дb''b' ↩
```

```
      'eb''b''тb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''вb''b''иb''b''вb''b''ib''b' ↩
      'дb'' b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b' ↩
      'яb''
  --always_above  b''3b''b''ab''b''пb''b''pb''b''ob''b''cb''b''иb''b''тb''b''иb'' b''вb''b' ↩
      'ib''b''кb''b''нb''b''ob'', b''щb''b''ob''b''бb'' b''вb''b''ob''b''нb''b''ob'' b''зb'' ↩
      b''ab''b''вb''b''жb''b''дb''b''иb'' b''бb''b''уb''b''лb''b''ob'' b''нb''b''ab''b''дb'' ↩
       b''ib''b''нb''b''шb''b''иb''b''мb''b''иb'' b''вb''b''ib''b''кb''b''нb''b''ab''b''мb'' ↩
      b''иb''
  -
-ini=INI_PATH  b''шb''b''лb''b''яb''b''xb'' b''дb''b''ob'' ini

[GladeVCP-][CRITICAL]  b''Дb''b''ob''b''cb''b''тb''b''уb''b''пb''b''нb''b''ib'' b''вb''b' ↩
    'бb''b''уb''b''дb''b''ob''b''вb''b''ab''b''нb''b''ib'' b''пb''b''ab''b''нb''b''eb''b' ↩
    'лb''b''ib'' VCP: (gladevcp:205)
['gtk_verser_probe', 'gtk_little_probe']
```

12.3.5 Розуміння процесу запуску GladeVCP

Кроки інтеграції, описані вище, виглядають дещо складними, і це так. Тому корисно зрозуміти процес запуску LinuxCNC та те, як це пов'язано з GladeVCP.

Звичайний процес запуску LinuxCNC виконує наступне:

- Середовище реального часу запущено.
- Усі компоненти HAL завантажено.
- Компоненти HAL пов'язані між собою за допомогою скриптів .hal cmd.
- `task`, `iocontrol` і, зрештою, запускається користувацький інтерфейс.
- До появи GladeVCP припущення було таким: до моменту запуску інтерфейсу користувача весь HAL завантажений, налаштований та готовий до роботи.

Впровадження GladeVCP призвело до наступної проблеми:

- Панелі GladeVCP потрібно вбудувати в головне вікно графічного інтерфейсу.
- Панелі GladeVCP потрібно вбудувати в головне вікно графічного інтерфейсу, наприклад, AXIS, Touchy, Gscreen або GMOCCAPY (вбудоване вікно або як вбудована вкладка).
- Це вимагає запуску головного графічного інтерфейсу, перш ніж вікно GladeVCP можна буде підключити до головного графічного інтерфейсу.
- Однак, GladeVCP також є компонентом HAL і створює власні виводи HAL.
- Як наслідок, усі HAL-з'єднання, що включають виводи GladeVCP HAL як джерело або призначення, повинні виконуватися **після** налаштування графічного інтерфейсу.

Це є метою `POSTGUI_HALFILE`. Ця опція INI перевіряється графічними інтерфейсами користувача. Якщо графічний інтерфейс користувача виявляє цю опцію, він запускає відповідний файл HAL після налаштування будь-якої вбудованої панелі GladeVCP. Однак він не перевіряє, чи панель GladeVCP фактично використовується, і в цьому випадку файл HAL cmd просто запускається у звичайному режимі. Отже, якщо ви НЕ запускаєте GladeVCP через `GLADEVCP` або `EMBED_TAB` тощо, а пізніше в окремому вікні оболонки або за допомогою іншого механізму, командний файл HAL у `POSTGUI_HALFILE` буде виконано занадто рано. Припускаючи, що тут є посилання на штифти GladeVCP, це призведе до помилки з повідомленням про те, що компонент GladeVCP HAL недоступний.

Отже, якщо ви запускаєте GladeVCP з окремого вікна оболонки (тобто не запускається графічним інтерфейсом у вбудованому режимі):

- Ви не можете покладатися на опцію INI `POSTGUI_HALFILE`, яка призводить до виконання команд HAL «у потрібний момент часу», тому закоментуйте це у файлі INI.
- Явно передайте файл команд HAL, який посилається на піни GladeVCP, до GladeVCP за допомогою опції `-H <файл halcmd>` (див. попередній розділ).

12.3.6 Довідник з віджетів HAL

GladeVCP включає в себе набір віджетів Gtk з приєднаними HAL-контактами, які називаються HAL-віджетами і призначені для управління, відображення або іншої взаємодії з шаром HAL LinuxCNC. Вони призначені для використання з редактором користувальницького інтерфейсу Glade. При правильній установці HAL-віджети повинні з'явитися в групі віджетів Glade «HAL Python». Багато полів, специфічних для HAL, в розділі «Загальні» Glade мають пов'язані підказки, що з'являються при наведенні курсору миші.

Сигнали HAL бувають двох типів: біти та числа. Біти — це сигнали увімкнення/вимкнення. Числа можуть бути типу «float», «s32» або «u32». Більш детальну інформацію про типи даних HAL дивіться в HAL manual. Віджети GladeVCP можуть або відображати значення сигналу за допомогою індикатора, або змінювати значення сигналу за допомогою елемента управління. Таким чином, існує чотири класи віджетів GladeVCP, які ви можете підключити до сигналу HAL. Інший клас допоміжних віджетів дозволяє організувати та позначити вашу панель.

- Віджети для індикації "бітових" сигналів: HAL_LED
- Віджети для керування "бітовими" сигналами: HAL_Button HAL_RadioButton HAL_CheckButton
- Віджети для індикації сигналів "число": HAL_Label, HAL_ProgressBar, HAL_HBar та HAL_VBar, HAL_Meter
- Віджети для керування сигналами «числа»: HAL_SpinButton, HAL_HScale та HAL_VScale, Jog Wheel, Speed Control
- Віджети чутливого керування: State_Sensitive_Table HAL_Table та HAL_HBox
- Попередній перегляд траєкторії інструменту: HAL_Gremlin
- Віджети для відображення положення осей: DRO Widget, Combi DRO Widget
- Віджети для обробки файлів: Вибір файлу IconView
- Віджети для відображення/редагування всіх зміщень осей: OffsetPage
- Віджети для відображення/редагування всіх зміщень інструментів: Tooloffset editor
- Віджет для відображення та редагування G-коду: HAL_Sourceview
- Віджет для введення MDI та відображення історії: MDI History

12.3.6.1 Найменування віджетів та пінів HAL

Більшість віджетів HAL мають один пов'язаний пін HAL з тим самим ім'ям HAL, що й у віджета (glade: General→Name).

Винятками з цього правила наразі є:

- «HAL_Spinbutton» та «HAL_ComboBox», які мають два виводи: `<назвавіджета>-f` (число з плаваючою комою) та `<назвавіджета>-s` (s32)
- «HAL_ProgressBar», який має вхідний контакт `<назвавіджета>-значення` та вхідний контакт `<назвавіджета>-шкала`.

12.3.6.2 Атрибути та методи Python віджетів HAL

Віджети HAL є екземплярами GtKWidgets і, отже, успадковують методи, властивості та сигнали відповідного класу GtkWidget. Наприклад, щоб з'ясувати, які методи, властивості та сигнали, пов'язані з GtkWidget, має «HAL_Button», перегляньте опис https://lazka.github.io/pgi-docs/# Gtk-3.0/classes/Button.html#Gtk.Button[GtkButton] в PyGObject API Reference.

Простий спосіб дізнатися про спадковість певного віджета HAL такий: запустіть glade, розмістіть віджет у вікні та виберіть його; потім виберіть вкладку «Signals» (Сигнали) у вікні «Properties» (Властивості). Наприклад, вибравши віджет «HAL_LED», ви побачите, що «HAL_LED» походить від «GtkWidget», який, у свою чергу, походить від «GtkObject» і, зрештою, від «GObject».

Повну ієрархію класів можна переглянути, викликавши GtkInspector у графічному інтерфейсі Glade, вибравши віджет і натиснувши Control-Shift-I. Якщо Inspector не відкривається, його можна увімкнути з терміналу, ввівши:

```
gsettings set org.gtk.Settings.Debug enable-inspector-keybinding true
```

Інспектор також зручний для тестування змін стилю CSS «на льоту», а також для визначення всіх властивостей та сигналів, доступних для віджета.

Віджети HAL також мають кілька атрибутів Python, специфічних для HAL:

hal_pin
: Базовий об'єкт HAL pin у Python, якщо віджет має один тип pin

hal_pin_s, hal_pin_f
: Піни s32 та float віджетів *HAL_Spinbutton* та *HAL_ComboBox* – зверніть увагу, що ці віджети не мають атрибута *hal_pin*!

hal_pin_scale
: Вхідний пін з плаваючою комою віджета *HAL_ProgressBar*, що представляє максимальне абсолютне значення вхідних даних.

Існує кілька специфічних для HAL методів віджетів HAL, але єдиний релевантний метод:

<halpin>.get()
: Отримати значення поточного виводу HAL, де *<halpin>* – це відповідна назва виводу HAL, зазначена вище.

12.3.6.3 Налаштування значень піна та віджета

Як правило, якщо вам потрібно встановити значення віджета виводу HAL з коду Python, зробіть це, викликавши базовий «сеттер» Gtk (наприклад, `set_active()`, `set_value()`). Не намагайтеся встановити значення пов'язаного виводу безпосередньо за допомогою `halcomp[pinname] = value`, оскільки віджет не помітить цю зміну!

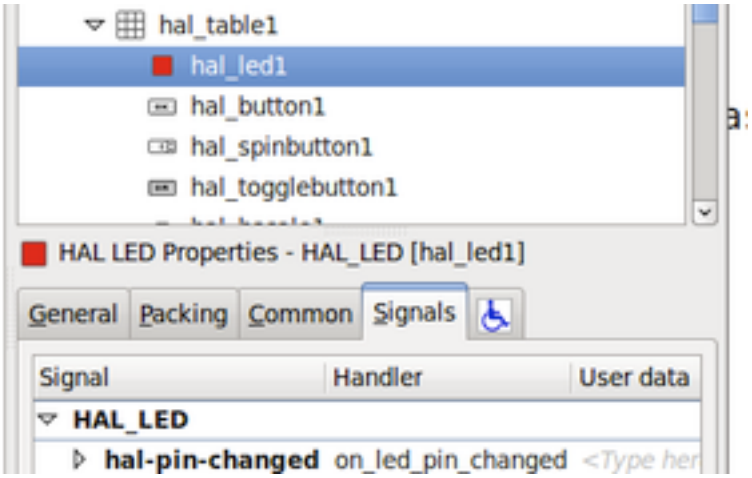
Може виникнути спокуса «встановити вхідні контакти віджета HAL» програмно. Зверніть увагу, що це суперечить основній меті вхідного контакту — він повинен бути пов'язаний із сигналами, що генеруються іншими компонентами HAL, і реагувати на них. Хоча наразі в HAL Python немає захисту від запису на вхідні контакти, це не має сенсу. Однак для тестування ви можете використовувати `setp _pinname_ _value_` у відповідному файлі HAL.

Цілком прийнятно встановлювати значення виходу HAL-контакту за допомогою `halcomp[pinname] = value`, за умови, що цей HAL-контакт не пов'язаний з віджетом, тобто був створений методом `hal_glib.GPin(halcomp.newpin(<name>,<type>,<direction>))` (приклад див. у GladeVCP Programming).

12.3.6.4 Сигнал зміни виводу Hal

Подієве програмування означає, що інтерфейс користувача повідомляє ваш код, коли «щось відбувається» — через зворотний виклик, наприклад, коли натиснуто кнопку. Вихідні віджети HAL (ті, що відображають значення виводу HAL), такі як LED, Bar, VBar, Meter тощо, підтримують сигнал `hal-pin-changed`, який може викликати зворотний виклик у вашому коді Python, коли — ну, вивід HAL змінює своє значення. Це означає, що більше немає потреби в постійному опитуванні змін виводу HAL у вашому коді, віджети роблять це у фоновому режимі і повідомляють вам про це.

Ось приклад того, як встановити сигнал `hal-pin-changed` для HAL_LED у редакторі інтерфейсу Glade:



Приклад у `configs/apps/gladevcp/complex` показує, як це обробляється в Python.

12.3.6.5 Кпонки

Ця група віджетів походить від різних кнопок Gtk і складається з віджетів HAL_Button, HAL_ToggleButto HAL_RadioButton та CheckButton. Усі вони мають один вихідний BIT-контакт, названий так само, як і віджет. Кнопки не мають додаткових властивостей порівняно з базовими класами Gtk.

- HAL_Button: миттєва дія, стан не зберігається. Важливий сигнал: `натиснуто`
- HAL_ToggleButton, HAL_CheckButton: зберігає стан увімкнено/вимкнено. Важливий сигнал: `toggled`
- HAL_RadioButton: група типу «один з багатьох». Важливий сигнал: `toggled` (для кожної кнопки).
- Важливі поширені методи: `set_active()`, `get_active()`
- Важливі властивості: `label`, `image`

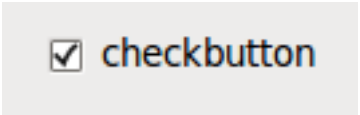


Figure 12.31: Кнопка перевірки

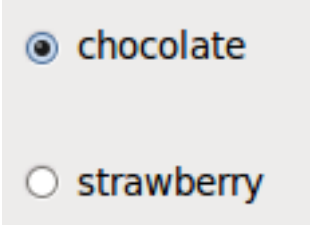


Figure 12.32: Перемикачі

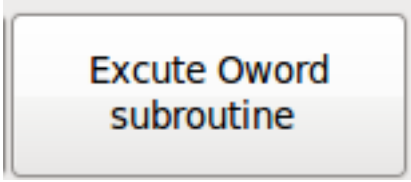


Figure 12.33: Кнопка перемикання

Tip
Визначення груп радіокнопок у Glade:

- Виберіть активну кнопку за замовчуванням.
- У діалоговому вікні «Вибрати радіокнопку в цьому проекті» іншої кнопки «Загальні→Група» виберіть назву активної кнопки за замовчуванням.

Див. `configs/apps/gladevcp/by-widget/` для отримання інформації про програми GladeVCP та файл інтерфейсу користувача для роботи з радіокнопками.

12.3.6.6 Масштаби

HAL_HScale та HAL_VScale походять від GtkHScale та GtkVScale відповідно.

<widgetname>
: вихідний штифт FLOAT

<widgetname>-s
: вихідний контакт s32

Щоб зробити шкалу корисною в Glade, додайте «Налаштування» (Загальне → Налаштування → Нове або існуюче налаштування) і відредагуйте об’єкт налаштування. Він визначає значення за замовчуванням/мінімальне/максимальне/кроку. Також встановіть налаштування «Розмір сторінки» і «Крок сторінки» на нуль, щоб уникнути попереджень.

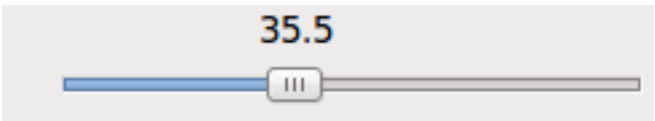


Figure 12.34: Приклад HAL_HScale

12.3.6.7 SpinButton

HAL SpinButton походить від GtkSpinButton та містить два контакти:

<widgetname>-f
вихідний штифт FLOAT

<widgetname>-s
вихідний контакт s32

Щоб бути корисними, кнопки спіну потребують значення налаштування, наприклад, шкали, див. вище.

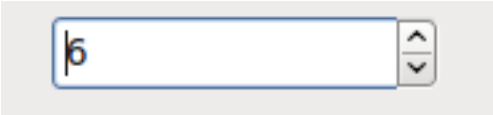


Figure 12.35: Приклад SpinButton

12.3.6.8 Hal_Dial

Віджет hal_dial імітує колесо прокрутки або регулятор налаштування.
Його можна керувати за допомогою миші. Ви можете просто використовувати коліщатко миші, коли курсор миші знаходиться над віджетом Hal_Dial, або утримувати ліву кнопку миші і рухати курсор по колу, щоб збільшити або зменшити значення.
Подвійним клацанням лівою або правою кнопкою можна збільшити або зменшити коефіцієнт масштабування.

- Проти годинникової стрілки = зменшення кількості
- За годинниковою стрілкою = збільшення кількості
- Колесо вгору = збільшення кількості
- Коліщатко вниз = зменшення кількості
- Подвійне клацання лівою кнопкою миші = масштаб x10
- Подвійне клацання правою кнопкою миші = масштаб /10

`hal_dial` експортує значення лічильника у вигляді виводів HAL:

<widgetname>
вихідний контакт s32

<widgetname>-масштабований
вихідний штифт FLOAT

<widgetname>-дельта-масштаб
вихідний штифт FLOAT

`hal_dial` має такі властивості:

cpr
Встановлює кількість обертів за оберт, допустимі значення в діапазоні від 25 до 360
за замовчуванням = 100

show_counts
Встановіть це значення на False, якщо ви хочете приховати відображення підрахунків посередині віджета.
за замовчуванням = True

мітка
Встановіть вміст мітки, яка може відображатися над значенням підрахунку.
Якщо мітка довша за 15 символів, вона буде скорочена до 15 символів.
за замовчуванням = порожнє

center_color
Це дозволяє змінити колір колеса. Використовується кольоровий рядок GDK.
за замовчуванням = #bdefbdefbdef (сірий)

count_type_shown
Доступні три підрахунки: 0) Необроблені підрахунки СЛР 1) Масштабовані підрахунки 2) Дельта-масштабовані підрахунки.
за замовчуванням = 1

- Підрахунок базується на вибраній серцево-легеневій реанімації – він рахуватиме як позитивні, так і негативні значення. Доступний як контакт s32.
- Масштабований підрахунок – це підрахунок серцево-легеневої реанімації (СЛР), помножений на шкалу – він може бути як додатним, так і від'ємним.
  Якщо ви зміните шкалу, вихід негайно відобразить зміну. Він доступний як вивід FLOAT.
- Дельта-масштабований підрахунок – це ЗМІНА підрахунку серцево-легеневої реанімації, помножена на масштаб.
  Якщо змінити масштаб, масштабуватимуться лише підрахунки після цієї зміни, а потім додадуться до поточного значення.
  Доступний як вивід FLOAT.

scale_adjustable
Встановіть для цього параметра значення «Хибно», якщо ви хочете заборонити зміну масштабу подвійним клацанням на віджеті.
Якщо для цього параметра значення «Хибно», коефіцієнт масштабування не відображатиметься на віджеті.
за замовчуванням = «Так»

масштаб
Встановіть це значення для масштабування підрахунків.
за замовчуванням = 1.0

Існують способи безпосереднього керування віджетом за допомогою Python.

Використання goobject для встановлення перелічених вище властивостей:

```
[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)
```

Існують методи Python:

- `[widget name].get_value()`
  Поверне значення counts як ціле число s32
- `[widget name].get_scaled_value()`
  Поверне значення counts як число з плаваючою комою

- `[widget name].get_delta_scaled_value()`
  Повертає значення counts як число з плаваючою комою
- `[widget name].set_label(”string”)`
  Встановлює вміст мітки за допомогою ”рядка”

Випромінюються два сигнали GObject:

- `count_changed`
  Викликається, коли лічильник віджета змінюється, наприклад, через прокручування коліщатком.
- `scale_changed`
  Викликається, коли масштаб віджета змінюється, наприклад, після подвійного клацання.

Підключіться до них ось так:

```
[widget name].connect(’count_changed’, [count function name])
[widget name].connect(’scale_changed’, [scale function name])
```

Функції зворотного виклику використовуватимуть цей шаблон:

```
def [count function name](widget, count,scale,delta_scale):
```

Це поверне: віджет, поточну кількість, масштаб та дельта-масштаб цього віджета.

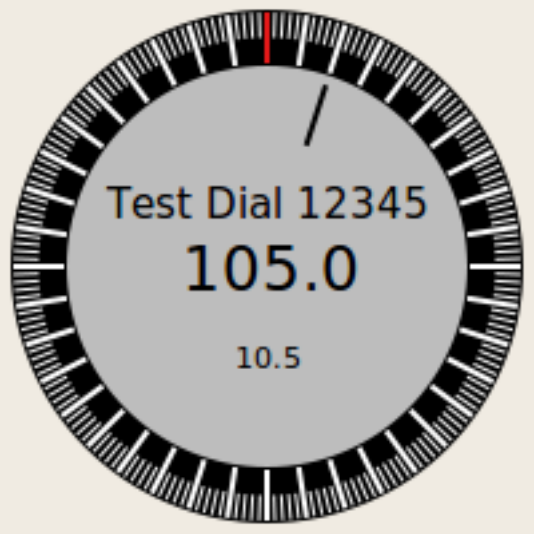


Figure 12.36: Приклад Hal_Dial

12.3.6.9 Джог-колесо

Віджет «jogwheel» імітує справжнє колесо прокрутки. Ним можна керувати за допомогою миші. Ви можете просто використовувати коліщатко миші, коли курсор миші знаходиться над віджетом JogWheel, або натиснути ліву кнопку миші і рухати курсор по колу, щоб збільшити або зменшити значення.

- Проти годинникової стрілки = зменшення кількості
- За годинниковою стрілкою = збільшення кількості
- Колесо вгору = збільшення кількості

- Коліщатко вниз = зменшення кількості

Оскільки переміщення миші методом перетягування може бути швидшим, ніж оновлення віджета, ви можете втратити рахунок, якщо будете рухатися занадто швидко. Рекомендується використовувати коліщатко миші, а метод перетягування застосовувати лише для дуже грубих рухів.

`jogwheel` експортує значення лічильника як вивід HAL:

<widgetname>-s
вихідний контакт s32

`jogwheel` має такі властивості:

розмір
Встановлює розмір віджета в пікселях, допустимі значення в діапазоні від 100 до 500, за замовчуванням = 200

cpr
Встановлює кількість обертів за оберт, допустимі значення в діапазоні від 25 до 100, за замовчуванням = 40

show_counts
Встановіть для цього значення «Хибно», якщо ви хочете приховати відображення підрахунків посередині віджета.

мітка
Встановіть вміст мітки, яка може відображатися над значенням лічильника. Мета полягає в тому, щоб дати користувачеві уявлення про використання цього колеса прокрутки. Якщо мітка довша за 12 символів, вона буде скорочена до 12 символів.

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання GObject для встановлення перелічених вище властивостей:

```
[widget name].set_property(”size”,int(value))
[widget name].set_property(”cpr”,int(value))
[widget name].set_property(”show_counts, True)
```

Існує два методи Python:

- `[widget name].get_value()`
  Поверне значення counts як ціле число
- `[widget name].set_label(”string”)`
  Встановлює вміст мітки за допомогою ”рядка”

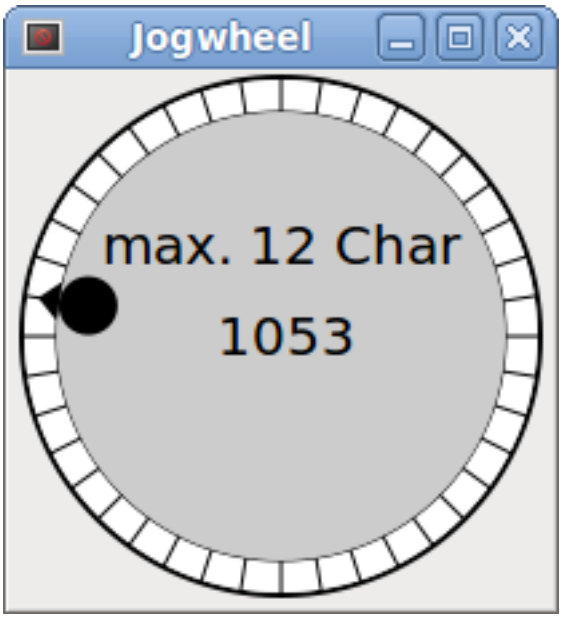


Figure 12.37: Приклад поворотного колеса

12.3.6.10 Контроль швидкості

`speedcontrol` це віджет, спеціально створений для керування налаштуваннями за допомогою сенсорного екрана. Він замінює звичайний віджет масштабування, який важко переміщати по сенсорному екрану.

Значення регулюється за допомогою двох кнопок для збільшення або зменшення значення. Крок збільшення змінюватиметься, поки кнопка натиснута. Значення кожного кроку збільшення, а також час між двома змінами можна встановити за допомогою властивостей віджета.

`speedcontrol` пропонує деякі HAL-піни:

<widgetname>-значення
out float pin
Показане значення віджета.

<widgetname>-масштабоване значення
out float pin
Показано значення, поділене на значення шкали, це дуже корисно, якщо швидкість відображаєть
в одиницях/хв, але LinuxCNC очікує, що вона буде в одиницях/секунда.

<widgetname>-масштаб
in float pin
Шкала, яку потрібно застосувати.
За замовчуванням – 60.

<widgetname>-збільшення
in bit pin
Поки контакт увімкнено, значення збільшуватиметься.
Дуже зручно з підключеним миттєвим перемикачем.

<widgetname>-зменшення
in bit pin
Поки контакт увімкнено, значення зменшуватиметься.
Дуже зручно з підключеним миттєвим перемикачем.

`speedcontrol` має такі властивості:

висота
Integer
Висота віджета в пікселях.
Допустимі значення від 24 до 96.
За замовчуванням – 36.

значення
Float
Початкове значення для встановлення.
Допустимі значення знаходяться в діапазоні від 0,001 до 99999,0.
Значення за замовчуванням – 10,0.

мін
Float
Мінімальне дозволене значення.
Дозволені значення: від 0,0 до 99999,0.
За замовчуванням: 0,0.
Якщо ви зміните це значення, приріст буде скинуто до значення за замовчуванням, тому може знадобитися встановити новий приріст.

макс.
Float
Максимально допустиме значення.
Допустимі значення: від 0,001 до 99999,0.
За замовчуванням: 100,0.
Якщо ви зміните це значення, приріст буде скинуто до значення за замовчуванням, тому може знадобитися встановити новий приріст.

приріст
Float
Встановлює застосовуване приростання для кожного клацання миші.
Допустимі значення: від 0,001 до 99999,0 та -1.
Значення за замовчуванням: -1, що дає 100 приростів від мінімуму до максимуму.

inc_speed
Integer
Встановлює затримку таймера для збільшення швидкості, утримуючи натиснутими кнопки.
Допустимі значення від 20 до 300.
За замовчуванням – 100.

unit
String
Встановлює одиницю вимірювання, яка відображатиметься на смузі після значення.
Дозволено будь-який рядок.
Значення за замовчуванням — ”“.

колір
Color
Встановлює колір смужки.
Дозволено будь-який шістнадцятковий колір.
Значення за замовчуванням — ”#FF8116”.

шаблон
String
Текстовий шаблон для відображення значення. Використовується форматування Python.
Будь-який дозволений формат.
Значення за замовчуванням — ”%.1f”.

do_hide_button
Boolean
Показувати чи приховувати кнопку збільшення або зменшення.
True (Так) чи False (Хибно).
За замовчуванням = False (Хибно).

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання GObject для встановлення перелічених вище властивостей:

```
[widget name].set_property(”do_hide_button”,bool(value))
[widget name].set_property(”color”,”#FF00FF”)
[widget name].set_property(”unit”, ”mm/min”)
etc.
```

Також існують методи Python для зміни віджета:

```
[widget name].set_adjustment(gtk-adjustment)
```

Ви можете призначити існуюче налаштування елементу управління, таким чином легко замінити існуючі повзунки без значних змін у коді. Зверніть увагу, що після зміни налаштування може знадобитися встановити нове збільшення, оскільки воно буде скинуто до значення за замовчуванням (100 кроків від MIN до MAX):

- `[widget name].get_value()`
  Повертає значення counts як число з плаваючою комою
- `[widget name].set_value(float(value))`
  Встановлює для віджета задане значення
- `[widget name].set_digits(int(value))`
  Встановлює цифри значення, яке буде використано
- `[widget name].hide_button(bool(value))`
  Приховати або показати кнопку

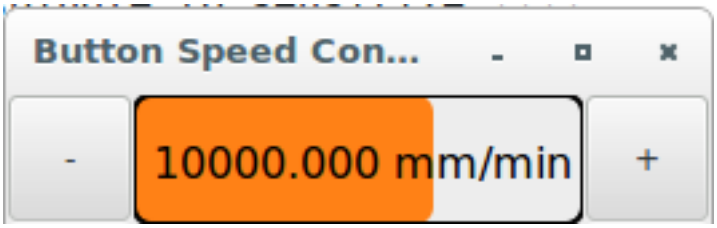


Figure 12.38: Приклад керування швидкістю

12.3.6.11 Мітка

`hal_label` — це простий віджет, заснований на GtkLabel, який представляє значення HAL-піна у визначеному користувачем форматі.

label_pin_type
Тип HAL піна (0:s32, 1:float, 2:u32), див. також підказку «Загальне→Тип піна HAL» (зверніть увагу, що це відрізняється від PyVCP, який має три віджети міток, по одному для кожного типу).

text_template
Визначає текст, що відображається — рядок формату Python для перетворення значення виводу в текст. За замовчуванням використовується `%s` (значення перетворюються за допомогою функції str()), але може містити будь-який аргумент, допустимий для методу format() Python. Приклад: `Distance: %.03f` відобразить текст і значення виводу з 3 десятковими цифрами, доповненими нулями для виводу FLOAT.

12.3.6.12 Контейнери

- HAL_HideTable
- HAL_Table
- State_Sensitive_Table
- HAL_HBox (застарілий)

Ці контейнери призначені для використання для знечутливості (засірення) або приховування їхніх дочірніх об’єктів.
Знечутливі дочірні об’єкти не реагуватимуть на ввід.

HAL_HideTable
Має один вхідний контакт HAL BIT, який контролює, чи приховані його дочірні віджети.

Закріпити: , <Panel_basename>.<widgetname>
in bit pin
Якщо PIN-код низький, то дочірні віджети видно, що є станом за замовчуванням.

HAL_Table і HAL_Hbox
Мають один вхідний контакт HAL BIT, який контролює, чи є їхні дочірні віджети чутливими чи ні.

Закріпити: , <Panel_basename>.<widgetname>
in bit pin
Якщо PIN-код низький, то дочірні віджети неактивні, що є станом за замовчуванням.

State_Sensitive_Table
Відповідає на стан інтерпретатора LinuxCNC.
Опціонально можна вибрати відповідь на «must-be-all-homed», «must-be-on» та «must-be-idle».
Ви можете комбінувати їх. При Estop завжди буде нечутливим.
(Не має контакту).

Warning
HAL_Hbox є застарілим — використовуйте HAL_Table.
Якщо поточні панелі використовують його, це не призведе до збою. Ви просто більше не знайдете його в редакторі GLADE.
У майбутніх версіях GladeVCP цей віджет може бути повністю вилучені, і тоді вам доведеться оновити панель.

Tip
Якщо ви виявите, що якась частина вашої програми GladeVCP «сіра» (нечутлива), перевірте, чи не встановлений або не підключений контакт HAL_Table.

12.3.6.13 LED

`hal_led` імітує справжній індикаторний світлодіод.
Він має один вхідний бітовий контакт, який контролює його стан: увімкнено або вимкнено.

Світлодіоди мають кілька властивостей, які визначають їхній зовнішній вигляд та відчуття:

on_color
: Рядок, що визначає колір світлодіода під час увімкнення.
Може бути будь-якою дійсною назвою gdk.Color.
Не працює на Ubuntu 8.04.

off_color
: Рядок, що визначає колір світлодіода у вимкненому стані.
Може бути будь-якою дійсною назвою gdk.Color або спеціальним значенням `dark`. `dark` означає, що колір вимкнено буде встановлено на значення 0,4 для кольору ввімкнено.
Не працює на Ubuntu 8.04.

pick_color_on, pick_color_off
: Кольори для станів УВІМК. та ВИМК.
Вони можуть бути представлені як рядки `#RRRRGGGGBBBB` та є необов'язковими властивостями, що мають пріоритет над `on_color` та `off_color`.

led_size
: Радіус світлодіода (для квадрата - половина сторони світлодіода)

led_shape
: Форма світлодіода.
Допустимі значення: 0 для круглої, 1 для овальної та 2 для квадратної форми.

led_blink_rate
: Якщо встановлено і світлодіод увімкнено, то він блимає.
Період блимання дорівнює "led_blink_rate", вказаному в мілісекундах.

create_hal_pin
: Вибрати/зняти вибір зі створення виводу HAL для керування світлодіодом.
Без створення виводу HAL світлодіодом можна керувати за допомогою функції Python.

Як вхідний віджет, LED також підтримує сигнал `hal-pin-changed`. Якщо ви хочете отримувати повідомлення у своєму коді, коли змінюється HAL-контакт світлодіода, підключіть цей сигнал до обробника, наприклад `on_led_pin_changed`, і надайте обробник наступним чином:

```
def on_led_pin_changed(self,hal_led,data=None):
    print("on_led_pin_changed() - HAL pin value:",hal_led.hal_pin.get())
```

Це буде викликано на будь-якому фронті сигналу, а також під час запуску програми для повідомлення поточного значення.

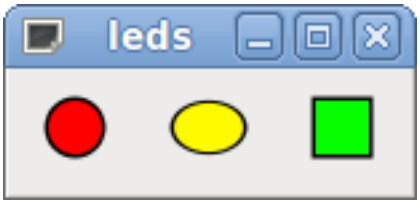


Figure 12.39: Приклад світлодіодів

12.3.6.14 Індикатор прогресу

Note
Цей віджет може зникнути.
Натомість використовуйте віджети HAL_HBar та HAL_VBar.

`HAL_ProgressBar` походить від gtk.ProgressBar та має два вхідні виводи HAL з плаваючою комою:

<widgetname>
поточне значення, яке буде відображатися

<widgetname>-масштаб
максимальне абсолютне значення вхідного сигналу

`HAL_ProgressBar` має такі властивості:

масштаб
Шкала значень.
Встановлює максимальне абсолютне значення вхідного значення. Те саме, що й встановлення виводу <widgetname>.scale.
Число з плаваючою комою, діапазон від -2^{24} до $+2^{24}$.

green_limit
Нижня межа зеленої зони

yellow_limit
Нижня межа жовтої зони

red_limit
Нижня межа червоної зони

text_template
Текстовий шаблон для відображення поточного значення `<назвавіджета>`.
Для dict `{"value":value}` можна використовувати форматування Python.

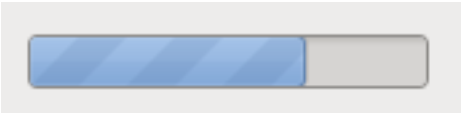


Figure 12.40: Приклад HAL_ProgressBar

12.3.6.15 Комбінований список

`HAL_ComboBox` походить від gtk.ComboBox. Він дозволяє вибирати значення зі спадного списку.

`HAL_ComboBox` експортує два піни HAL:

<widgetname>-f
Поточне значення, тип FLOAT

<widgetname>-s
Поточне значення, тип s32

`HAL_ComboBox` має таку властивість, яку можна встановити в Glade:

колона
Індекс стовпця.
Тип s32.
Допустимий діапазон від -1 до 100.
Значення за замовчуванням -1.

У режимі за замовчуванням цей віджет встановлює піни на індекс вибраного запису списку. Тож, якщо ваш віджет має три мітки, він може приймати лише значення 0, 1 та 2.

У режимі стовпця (column > -1) значення, що відображається, вибирається з масиву ListStore, як визначено в Glade. Зазвичай визначення вашого віджета матиме два стовпці в ListStore: один із текстом, що відображається в спадному меню, та цілочисельне або дійсне значення, яке використовується для цього вибору.

У `configs/apps/by-widget/combobox.{py,ui}` наведено приклад, який використовує режим стовпця для вибору значення з плаваючою комою зі ListStore.

Якщо ви, як і я, не знаєте, як редагувати ComboBox ListStores та CellRenderer, дивіться https://youtu.be/-Z5_F-rW2cL8.

12.3.6.16 Бари

Віджети `HAL_Bar` та `HAL_VBar` для горизонтальних та вертикальних стовпчиків, що представляють значення з плаваючою комою.

`HAL_Bar` та `HAL_VBar` мають по одному вхідному виводу FLOAT HAL.

`HAL_Bar` та `HAL_VBar`, обидва смужки мають такі властивості:

інвертувати
Поміняйте напрямки min та max.
Інвертований HBar зростає справа наліво, інвертований VBar — зверху вниз.

min, max
Мінімальне та максимальне значення бажаного діапазону. Якщо поточне значення знаходиться поза цим діапазоном, це не є помилкою.

показати обмеження
Використовується для вибору/зняття вибору тексту обмежень на смузі.

нуль
Нульова точка діапазону.
Якщо вона знаходиться в межах мінімального/максимального діапазону, то смуга буде зростати від цього значення, а не від лівого (або правого) боку віджета.
Корисно для відображення значень, які можуть бути як додатними, так і від'ємними.

force_width, force_height
Примусова ширина або висота віджета.
Якщо не встановлено, розмір буде визначатися з упаковки або з фіксованого розміру віджета, а смужка заповнить усю область.

text_template
Як і в Label, встановлює формат тексту для мінімальних/максимальних/поточних значень.
Можна використовувати для вимкнення відображення значень.

значення
Встановлює значення для відображення стовпчика відповідно до введеного значення.
Використовується лише для тестування в редакторі GLADE.
Значення буде встановлено з виводу HAL.

цільове значення
Встановлює цільовий рядок на введене значення.
Використовується лише для тестування в редакторі GLADE.
Значення можна встановити у функції Python.

target_width
Ширина лінії, що позначає цільове значення.

bg_color
Колір фону (неактивного) смужки.

target_color
Колір цільової лінії.

z0_color, z1_color, z2_color
Кольори різних зон значень.
Значення за замовчуванням: `green`, `yellow` та `red`.
Опис зон дивіться у властивостях `z*_border`.

z0_border, z1_border
Визначте верхні межі колірних зон.
За замовчуванням увімкнено лише одну зону. Якщо ви хочете більше однієї зони, встановіть `z0_border` та `z1_border` на бажані значення, щоб зона 0 заповнювалася від 0 до першої межі, зона 1 заповнювалася від першої до другої межі, а зона 2 — від останньої межі до 1.
Межі встановлюються як дроби.
Діапазон допустимих значень — від 0 до 1.

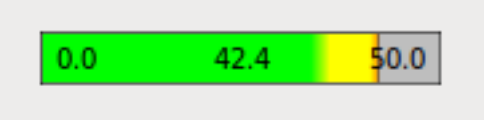


Figure 12.41: Горизонтальна смуга

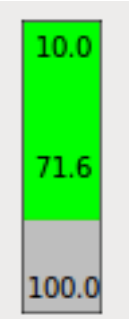


Figure 12.42: Вертикальна смуга

12.3.6.17 Метр

`HAL_Meter` — це віджет, схожий на PyVCP-метр — він представляє значення з плаваючою комою.

`HAL_Meter` має один вхідний контакт FLOAT HAL.

HAL-метр має такі властивості:

min, max
Мінімальне та максимальне значення бажаного діапазону.
Якщо поточне значення знаходиться поза цим діапазоном, це не є помилкою.

force_size
Примусовий діаметр віджета.
Якщо не встановлено, розмір буде визначатися з упаковки або з фіксованого розміру віджета, а метр заповнить весь доступний простір з урахуванням співвідношення сторін.

text_template
Як і в Label, встановлює текстовий формат для поточного значення.
Можна використовувати для вимкнення відображення значення.

мітка
Велика етикетка над центром лічильника.

підмітка
Невелика етикетка під центром лічильника.

bg_color
Колір фону лічильника.

z0_color, z1_color, z2_color
Кольори різних зон значень.
Значення за замовчуванням: `green`, `yellow` та `red`.
Опис зон дивіться у властивостях `z*_border`.

z0_border, z1_border
Визначте верхні межі колірних зон.
За замовчуванням увімкнено лише одну зону. Якщо ви хочете більше однієї зони, встановіть `z0_border` та `z1_border` на бажані значення, щоб зона 0 заповнювалася від мінімального до першого межі, зона 1 заповнювалася від першого до другого межі, а зона 2 — від останнього межі до максимального.
Межі встановлюються як значення в діапазоні мінімум-максимум.



Figure 12.43: Приклади вимірювачів HAL

12.3.6.18 HAL_Graph

Цей віджет призначений для відображення значень з плином часу.

12.3.6.19 Попередній перегляд траєкторії інструменту Gremlin для файлів NGC

Gremlin — це віджет попереднього перегляду сюжету, схожий на вікно попереднього перегляду AXIS. Він передбачає роботу в середовищі LinuxCNC, такому як AXIS або Touchy. Для підключення до нього перевіряє змінну середовища INI_FILE_NAME. Gremlin відображає поточний файл NGC — він відстежує зміни та перезавантажує файл ngc, якщо ім'я файлу в AXIS/Touchy змінюється. Якщо ви запускаєте його в додатку GladeVCP, коли LinuxCNC не працює, ви можете отримати трасування, оскільки віджет Gremlin не може знайти стан LinuxCNC, наприклад поточне ім'я файлу.

Gremlin не експортує жодних HAL-пінів.

Гремлін має такі властивості:

enable_dro
Це відображає падіння на графіку.
За замовчуванням = true.

show_velocity
Тут відображається швидкість інструменту.
За замовчуванням = true.

use_commanded
Це вибирає значення DRO для використання: командні або фактичні.
За замовчуванням = true.

metric_units
Це вибирає одиниці вимірювання DRO: метричні або імперські.
За замовчуванням = true.

show_rapids
Це вказує плотеру показати швидкі рухи.
За замовчуванням = true.

show_dtg_
Це вибирає, щоб DRO відображав значення залишкової відстані.
За замовчуванням = true.

use_relative
Це вибирає, щоб DRO відображав значення відносно координат системи користувача або машини.
За замовчуванням = true.

show_live_plot
Це вказує плотеру, чи потрібно малювати, чи ні.
За замовчуванням = true.

show_limits
Це вказує плотеру показати межі можливостей машини.
За замовчуванням = true.

show_lathe_radius
Це вибирає відображення DRO для осі X у радіусі або діаметрі, якщо в режимі токарного верстата (можна вибрати у файлі INI з LATHE = 1).
За замовчуванням = true.

show_extents_option
Це вказує плотеру показати межі машини.
За замовчуванням = true.

show_tool
Це вказує плотеру намалювати інструмент.
За замовчуванням = true.

show_program
Показує програму G-коду.
За замовчуванням = True

use_joints_mode
Використовується в нетривіальних машинах (наприклад, роботах).
За замовчуваннямt = false.

grid_size
Встановлює розмір сітки (відображається лише у вікнах X, Y та Z).
За замовчуванням — 0

use_default_controls
Це вимикає стандартні елементи керування мишею.
Це найбільш корисно при використанні сенсорного екрану, оскільки стандартні елементи керування не працюють належним чином. Ви можете програмно додати елементи керування за допомогою Python та техніки обробки файлів.
За замовчуванням = true.

вид
Може бути будь-яким із `x`, `y`, `y2`, `z`, `z2`, `p` (перспектива).
За замовчуванням використовується `z` вигляд.

enable_dro
Тип = boolean.
Чи малювати DRO на графіку чи ні.
За замовчуванням = true.

mouse_btn_mode
Тип = integer.
Обробка кнопок миші: призводить до різних функцій кнопки:

- 0 = за замовчуванням: поворот ліворуч, переміщення посередині, масштабування праворуч
- 1 = масштабування ліворуч, рух посередині, поворот праворуч
- 2 = ліворуч, поворот посередині, масштабування праворуч
- 3 = масштабування ліворуч, поворот посередині, переміщення праворуч
- 4 = ліворуч, масштабування посередині, поворот праворуч
- 5 = поворот ліворуч, масштабування посередині, рух праворуч
- 6 = ліворуч, середній зум, правий зум

Режим 6 рекомендується для плазмових та токарних верстатів, оскільки для таких машин обертання не потрібне.

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання GObject для встановлення перелічених вище властивостей:

```
[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)
```

Існують методи 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)
```

Підказки

- Якщо встановити всі параметри побудови графіка на значення false, але show_offsets на значення true, то замість графічного графіка отримаєте сторінку зміщень.
- Якщо ви отримаєте відстань масштабування перед зміною подання, а потім скинете відстань масштабування, це набагато зручніше для користувача.
- Якщо ви оберете елемент у попередньому перегляді, вибраний елемент буде використано як центральна точка обертання

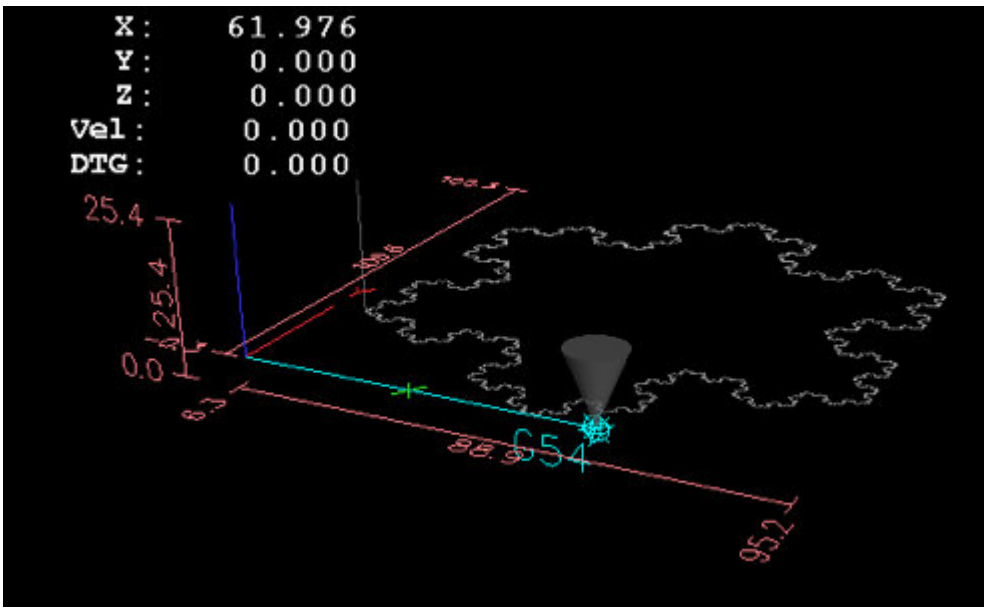


Figure 12.44: Приклад Гремліна

12.3.6.20 HAL_Offset

Віджет `HAL_Offset` використовується для відображення зміщення однієї осі.

`HAL_Offset` має такі властивості:

display_units_mm
Відображати в метричних одиницях.

joint_number
Використовується для вибору осі (технічно, який шарнір) відображається.
На верстаті Trivialkins (фрезерному, токарному, фрезерному) номер осі та шарніра:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W
```

mm_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.

imperial_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.

reference_type

```
0:G5x 1:tool 2:G92 3:b''Ob''b''6b''b''eb''b''pb''b''тb''b''ab''b''нb''b''нb''b''яb'' b' ↩
    'нb''b''ab''b''вb''b''кb''b''ob''b''лb''b''ob'' Z
```

12.3.6.21 DRO віджет

Віджет DRO використовується для відображення поточного положення осі.

Він має такі властивості:

display_units_mm
Використовується для перемикання одиниць відображення між метричною та імперською системами. Значення за замовчуванням – False.

фактичний
Виберіть фактичне (зворотне) положення або задане положення. За замовчуванням встановлено значення «True».

діаметр
Відображення діаметра для токарного верстата. За замовчуванням False.

mm_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю. Значення за замовчуванням — "%10.3f".

imperial_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю. Значення за замовчуванням — "%9.4f".

joint_number
Використовується для вибору осі (технічно, який шарнір) відображається. За замовчуванням – 0.
На верстаті Trivialkins (фрезерний, токарний, фрезерний) номер осі в порівнянні з номером шарніра:

```
0:X 1:Y 2:Z 3:A 4:B 5:C 6:U 7:V 8:W +
```

reference_type

- 0 = `absolute` (походження машини).
- 1 = `relative` (до поточного початку координат користувача - G5x).
- 2 = `distance-to-go` (відносно поточного початку координат користувача). За замовчуванням is 0.

font_family
Вкажіть сімейство шрифтів, наприклад, mono. За замовчуванням використовується sans. Якщо шрифт не існує, буде використано поточний системний шрифт. За замовчуванням використовується sans.

font_size
Вкажіть розмір шрифту від 8 до 96. За замовчуванням – 26.

font_weight
Вкажіть товщину шрифту. Виберіть світліший, звичайний, жирний або жирніший. За замовчуванн вибрано жирний шрифт.

unhomed_color
Колір тексту, коли його не розміщено, вказано як колір Gdk.RGBA. За замовчуванням – червоний, Gdk.RGBA(червоний=1.000000, зелений=0.000000, синій=0.000000, альфа=1.000000)

homed_color
Колір тексту, коли вказано колір Gdk.RGBA, визначається як колір Gdk.RGBA. За замовчуванням – зелений, Gdk.RGBA(червоний=0.000000, зелений=0.501961, синій=0.000000, альфа=1.000000)

Підказки

- Якщо ви хочете, щоб відображення було вирівняно по правому краю, встановіть для горизонтальн вирівнювання значення «Кінець».
- Фон віджета насправді прозорий, тому якщо ви розмістите його над зображенням, цифри DRO будуть відображатися поверх нього без фону. Для цього існує спеціальна техніка. Дивіться анімовані діаграми функцій нижче.
- Віджет DRO — це модифікований віджет міток gtk. Таким чином, багато з того, що можна зробити з міткою gtk, можна зробити і з віджетом DRO.
- Властивості шрифту також можна встановити з таблиці стилів CSS, яка має найвищий пріоритет і замінить значення, встановлені властивостями GObject.

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання GObject для встановлення перелічених вище властивостей

```
[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”)
```

```
# b’’Лb’’b’’eb’’b’’rb’’b’’шb’’b’’eb’’ b’’чb’’b’’иb’’b’’тb’’b’’ab’’b’’тb’’b’’иb’’ b’’кb’’b’ ↩
    ’ob’’b’’лb’’b’’ьb’’b’’ob’’b’’pb’’b’’иb’’, b’’вb’’b’’иb’’b’’кb’’b’’лb’’b’’иb’’b’’кb’’b’ ↩
    ’ab’’b’’юb’’b’’чb’’b’’иb’’ b’’фb’’b’’yb’’b’’нb’’b’’кb’’b’’цb’’b’’ib’’b’’юb’’:
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"))
```

Використання таблиці стилів CSS для налаштування властивостей шрифту
Кольори можна вказати в одному з кількох форматів, усі вони вказуватимуть один і той самий колір: червоний, *#ff0000, *rgb(255,0,0) або rgba(255,0,0,255).

Кольори можуть позначатися разом:

```
.dro_unhomed {color: magenta}
.dro_homed {color: cyan}
```

або окремо за назвою віджета:

```
#[widget name].dro_unhomed {color: magenta}
#[widget name].dro_homed {color: cyan}
```

На інші властивості стилю потрібно посилатися за назвою віджета:

```
#[widget name], #[widget name], #[widget name] {
    font-family: mono;
    font-size: 60px;
    font-weight: lighter;
}
```

Існує два методи Python

```
[widget name].set_dro_inch()
[widget name].set_dro_metric()
```

12.3.6.22 Віджет Combi_DRO

Віджет «Combi_DRO» використовується для відображення поточного значення, відносного положення осі та відстані, що залишається до кінця, в одному DRO.
При натисканні на DRO порядок DRO буде змінюватися.
У відносному режимі буде відображатися фактична система координат.

`Combi_DRO` має такі властивості:

joint_number
Використовується для вибору осі (технічно, який шарнір) відображається.
На верстаті Trivialkins (фрезерному, токарному, фрезерному) номери осей/шаблонів:

`0:X 1:Y 2:Z etc.`

фактичний
Виберіть фактичне (зворотний зв'язок) або командне положення.

metric_units
Використовується для перемикання одиниць відображення між метричною та імперською системами вимірювання.

auto_units
Одиниці вимірювання перемикатимуться між метричною та імперською системами залежно від того, чи активний G-код – G20 або G21.
За замовчуванням встановлено значення TRUE (ІСТИНА).

діаметр
Відображати положення як діаметр чи радіус.
У режимі діаметра DRO відображатиме значення з'єднання, помножене на 2.

mm_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.
Значення за замовчуванням — "%10.3f".

imperial_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.
Значення за замовчуванням — "%9.4f".

homed_color
Колір переднього плану номерів DRO, якщо з'єднання має вихідне положення.
За замовчуванням – зелений.

unhomed_color
Колір переднього плану номерів DRO, якщо з'єднання не має опорної точки.
За замовчуванням – червоний.

abs_color
Колір фону індикатора DRO, якщо головний індикатор DRO показує абсолютні координати.
За замовчуванням – синій.

krel_color
Колір фону індикатора DRO, якщо головний індикатор DRO показує відносні координати.
За замовчуванням – чорний.

dtg_color
Колір фону індикатора, якщо головний індикатор показує відстань, що залишилася.
За замовчуванням – жовтий.

font_size
Розмір шрифту для великих чисел, для малих буде в 2,5 рази меншим.
Значення має бути цілим числом у діапазоні від 8 до 96.
За замовчуванням – 25.

toggle_readout
Лівою кнопкою миші можна перемикати режими відображення DRO [«Rel», «Abs», «DTG»].
Знявши прапорець, можна вимкнути цю функцію. Перемикання можна виконати за допомогою `[widget name].toggle_readout()`.
Значення має бути булевим.
За замовчуванням — TRUE.

cycle_time
Час, протягом якого DRO очікує між двома опитуваннями.
Це налаштування слід змінювати лише в тому випадку, якщо ви використовуєте більше 5 DRO одночасно, тобто в 6-осьовій конфігурації, щоб уникнути надмірного уповільнення основної програми DRO.
Значення повинно бути цілим числом у діапазоні від 100 до 1000. FIXME unit=ms ?
За замовчуванням – 150.

Використання GObject для встановлення перелічених вище властивостей:

```
[b''нb''b''ab''b''зb''b''вb''b''ab'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ab'']. ↩
   set_property(property, value)
```

Існує кілька методів Python для керування віджетом:

- `[назва віджета].set_to_inch(state)`
  Встановлює для DRO відображення імперських одиниць.
  `state` = boolean (True або False)
  За замовчуванням – FIXME.
- `[widget name].set_auto_units(state)`
  Якщо значення True, то DRO змінюватиме одиниці вимірювання відповідно до активного G-коду (G20 / G21).
  `state` = boolean (True або False)
  За замовчуванням значення True.
- `[widget name].set_to_diameter(state)`
  Якщо значення True (Істина), указівник відображатиме діаметр, а не радіус, тобто значення осі, помножене на 2 (особливо необхідно для токарних верстатів).
  `state` = boolean (True або False)
  За замовчуванням False.
- `[widget name].toggle_readout()`
  Змінює порядок DRO у віджеті.
- `[widget name].change_axisletter(letter)`
  Змінює автоматично задану літеру осі.
  Дуже корисно для зміни укаження токарного верстата з «X» на «R» або «D».
  `letter` = string
- `[widget name].get_order()`
  Повертає порядок DRO у віджеті, який використовується переважно для підтримки їхньої узгоджено
  Порядок також буде передано разом із сигналом клацання.
  Повертає список, що містить порядок.
- `[widget name].set_order(order)`
  Встановлює порядок DRO, головним чином використовується для підтримки їхньої узгодженості.
  `order` = об’єкт списку, має бути одним із:
  - `[”Rel”, ”Abs”, ”DTG”]` (за замовчуванням)
  - `[”DTG”, ”Rel”, ”Abs”]`
  - `[”Abs”, ”DTG”, ”Rel”]`
- `[widget name].get_position()`
  Повертає позицію DRO у вигляді списку чисел з плаваючою комою.
  Порядок не залежить від порядку, що відображається в DRO, і буде задано як `[Абсолютний, відносний, DTG]`.

  - `Absolute` = координати машини, залежно від фактичної властивості, нададуть фактичне або командне положення.
  - `Relative` = будуть координатами фактичної системи координат.
  - `DTG` = відстань, що залишилася.
    Здебільшого буде 0, оскільки цю функцію не слід використовувати під час руху машини через часові затримки.

Віджет видаватиме такі сигнали:

- `clicked`
  Цей сигнал випромінюється, коли користувач натискає на віджет Combi_DRO.
  Він надсилає такі дані:
  - `widget` = widget object
    Об’єкт віджета, який надсилає сигнал.
  - `joint_number` = integer
    Спільний номер DRO, де «0:X 1:Y 2:Z тощо».
  - `order` = list object
    Порядок DRO у цьому віджеті.
    Цей порядок може бути використаний для встановлення такого ж порядку для інших віджетів Combi_DRO. `[widget name].set_order(order)`.
- `units_changed`
  Цей сигнал випромінюється, якщо змінюються одиниці вимірювання DRO.
  Він надсилатиме такі дані:
  - `widget` = widget object
    Об’єкт віджета, який надсилає сигнал.
  - `metric_units` = boolean
    Істина, якщо DRO відображає метричні одиниці, Хибність — у випадку імперських одиниць.
- `system_changed`
  Цей сигнал випромінюється, якщо змінюються одиниці вимірювання DRO.
  Він надсилатиме такі дані:
  - `widget` = widget object
    Об’єкт віджета, який надсилає сигнал.
  - `system` = string
    Фактична система координат. Буде однією з G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3 або Rel, якщо жодної не було вибрано взагалі, що відбуватиметься лише в Glade без запущеного LinuxCNC.

Ось деяка інформація, яку можна отримати за допомогою команд, яка може бути вам цікавою:

- `[widget name].system`
  Фактична система, як зазначено в сигналі system_changed.
- `[widget name].homed`
  Істина, якщо з’єднання має хоум-код.
- `[widget name].machine_units`
  0, якщо імперські, 1, якщо метричні.



Figure 12.45: Приклад: Три Combi_DRO у вікні

```
X = b''Bb''b''ib''b''дb''b''нb''b''ob''b''cb''b''нb''b''иb''b''йb'' b''pb''b''eb''b''жb''b' ↩
    'иb''b''мb''
Y = b''Ab''b''бb''b''cb''b''ob''b''лb''b''юb''b''тb''b''нb''b''иb''b''йb'' b''pb''b''eb''b' ↩
    'жb''b''иb''b''мb''
Z = b''Pb''b''eb''b''жb''b''иb''b''мb'' DTG
```

12.3.6.23 IconView (Вибір файлу)

Це зручний для сенсорного екрана віджет для вибору файлу та зміни каталогів.

`IconView` віджет має такі властивості:

icon_size
Встановлює розмір відображеної значка.
Допустимі значення – цілі числа в діапазоні від 12 до 96.
За замовчуванням – 48.

start_dir
Встановлює каталог, з якого починається показ віджета вперше.
Має бути рядком, що містить дійсний шлях до каталогу.
Значення за замовчуванням — ”/”.

jump_to_dir
Встановлює каталог «перейти до», який вибирається відповідною кнопкою в нижньому списку кнопок (5-та кнопка, рахуючи зліва).
Повинно бути рядком, що містить дійсний шлях до каталогу.
За замовчуванням — «\~».

filetypes
Встановлює фільтр файлів для відображення об’єктів.
Має бути рядком, що містить список розширень, розділених комами, для відображення.
За замовчуванням — ”ngc,py”.

чорний порядок
Встановлює порядок сортування відображеної піктограми.
Має бути цілим числом від 0 до 3, де:

- 0 = ASCENDING (відсортовано за іменами файлів)
- 1 = DESCENDING (відсортовано за іменами файлів)
- 2 = FOLDERFIRST (спочатку показати папки, потім файли), за замовчуванням
- 3 = FILEFIRST (спочатку показати файли, потім папки)

Використання GObject для встановлення перелічених вище властивостей:

```
[widget name].set_property(property,Value)
```

Існують методи Python для керування віджетом:

- `[widget name].show_buttonbox(state)`
  Якщо False, нижня кнопка буде прихована.
  Це корисно в користувацьких екранах зі спеціальним розташуванням кнопок, щоб не змінювати макет графічного інтерфейсу. Хорошим прикладом цього є GMOCCAPY.
  `state` = boolean (True або False).
  За замовчуванням True.
- `[widget name].show_filelabel(state)`
  Якщо True, буде показано мітку файлу (між вікном IconView і нижньою кнопкою).
  Приховування цієї мітки може заощадити місце, але її відображення дуже корисно для налагодження
  `state` = boolean (True або False).
  За замовчуванням True.
- `[widget name].set_icon_size(iconsize)`
  Встановлює розмір значка.
  Має бути цілим числом у діапазоні від 12 до 96.
  За замовчуванням = 48.
- `[widget name].set_directory(directory)`
  Дозволяє встановити каталог для відображення.
  `directory` = рядок (дійсний шлях до файлу).
- `[widget name].set_filetypes(filetypes)`
  Встановлює фільтр файлів, який буде використано.
  Будуть відображатися лише файли з заданими розширеннями.
  `filetypes` = рядок, що містить список розширень, розділених комами.
  За замовчуванням = ”ngc,py”.
- `[widget name].get_selected()`
  Повертає шлях до вибраного файлу або `None`, якщо каталог було вибрано.
- `[widget name].refresh_filelist()`
  Оновлює список файлів.
  Потрібно, якщо ви додаєте файл без зміни каталогу.

Якщо поле кнопки було приховано, ви можете отримати доступ до функцій цієї кнопки через сигнали її натискання, такі як:

```
[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”)
```

Віджет видаватиме такі сигнали:

- `selected`
  Цей сигнал випромінюється, коли користувач вибирає значок.
  Він повертає рядок, що містить шлях до файлу, якщо файл було вибрано, або `None`, якщо було вибрано каталог.

- `sensitive`
  Цей сигнал випромінюється, коли кнопки змінюють свій стан з чутливого на нечутливий або навпаки.
  Цей сигнал корисний для синхронізації навколишнього графічного інтерфейсу користувача з кнопкою віджета. Дивіться GMOCCAPY як приклад.
  Він поверне **buttonname** і новий **state**:

  - `buttonname` може бути одним з `btn_home`, `btn_dir_up`, `btn_sel_prev`, `btn_sel_next`, `btn_jump_to` або `btn_select`.
  - `state` – це логічне значення, яке прийматиме значення True або False.

- `exit`
  Цей сигнал випромінюється, коли натискається кнопка виходу для закриття IconView.
  Найчастіше потрібен, якщо програма запускається як окрема.

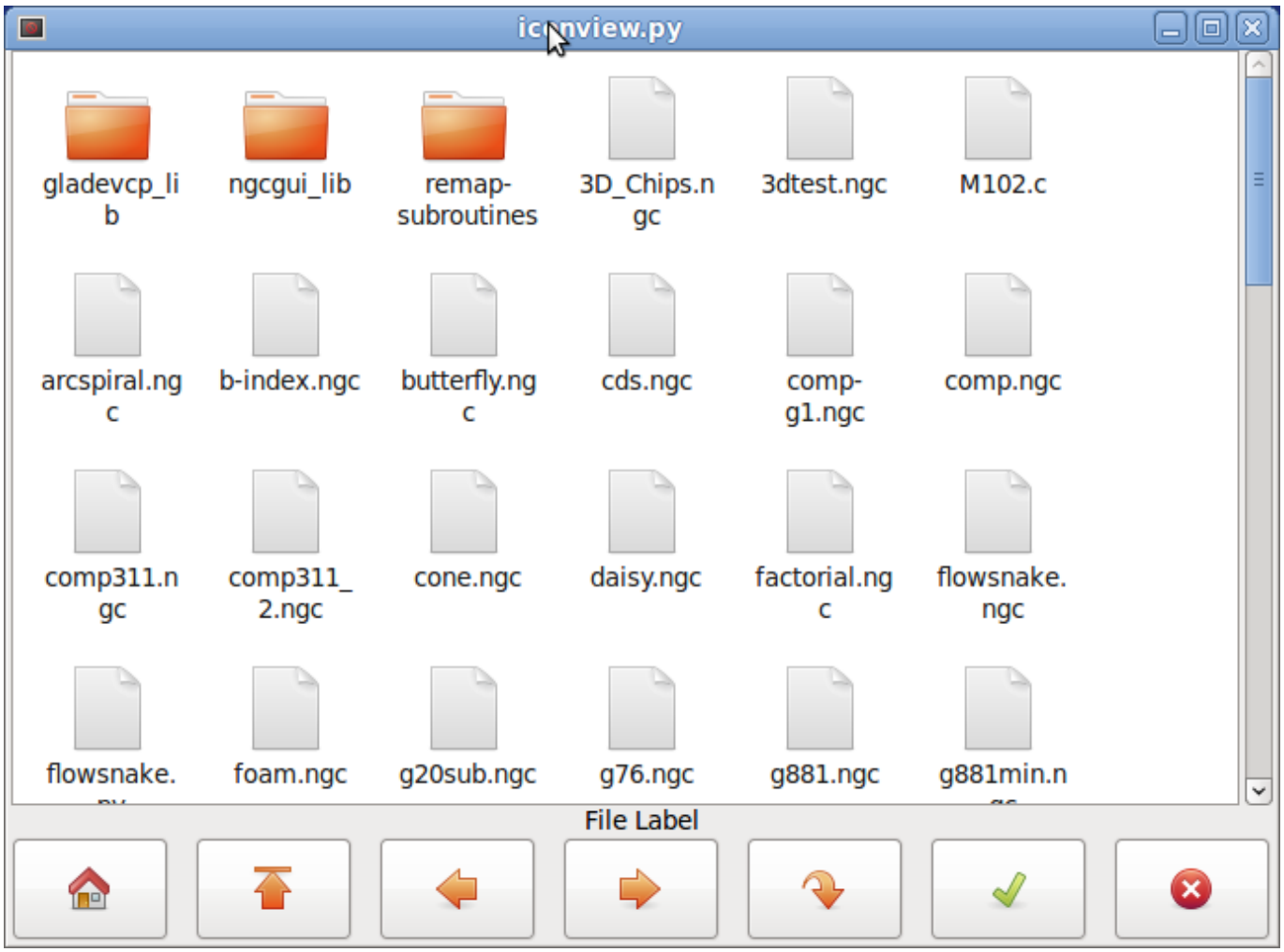


Figure 12.46: Приклад перегляду значків

12.3.6.24 Віджет калькулятора

Це простий віджет калькулятора, який можна використовувати для числового введення.
Ви можете попередньо налаштувати відображення та отримати результат або це попередньо встановлене значення.

`calculator` має такі властивості:

is_editable
Це дозволяє вводити дані з клавіатури.

шрифт
Це дозволяє налаштувати шрифт відображення.

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання goobject для встановлення перелічених вище властивостей:

```
[widget name].set_property(”is_editable”,True)
[widget name].set_property(”font”,”sans 25”)
```

Існують методи Python:

- `[widget name].set_value(2.5)`
  Це попередньо налаштовує дисплей і записує його.
- `[назва віджета].set_font(”sans 25”)`
- `[widget name].set_editable(True)`
- `[widget name].get_value()`
  Повертає обчислене значення – число з плаваючою комою.
- `[назва віджета].set_editable(True)`
- `[widget name].get_preset_value()`
  Повертає записане значення: число з плаваючою комою.

12.3.6.25 Віджет Tooleditor

Це віджет `tooleditor` для відображення та модифікації файлу інструменту.
У режимі токарного верстата він відображає знос та зміщення інструменту окремо.
Знос позначається номером інструменту вище 10000 (стиль Fanuc).
Він перевіряє поточний файл раз на секунду, щоб побачити, чи оновив його LinuxCNC.

Note
LinuxCNC вимагає перепризначення викликів інструментів для фактичного використання зміщень зносу.

`tooleditor` має такі властивості:

шрифт
Відображати шрифт для використання

hide_columns
Це приховає вказані стовпці.
Стовпці позначені (у порядку) таким чином: `s,t,p,x,y,z,a,b,c,u,v,w,d,i,j,q`.
Ви можете приховати будь-яку кількість стовпців, включаючи вибір та коментарі.

lathe_display_type
Показати формат токарного верстата

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання goobject для встановлення перелічених вище властивостей:

```
[widget name].set_properties('hide_columns','uvwijq')
```

Це приховає стовпці uvwij та q і покаже всі інші.

Існують методи Python:

- `[widget name].set_visible("ijq",False)`
  Приховав би стовпці ij та Q, а решту залишив би як є.
- `[widget name].set_filename(path_to_file)`
  Встановлює та завантажує файл інструменту.
- `[widget name].reload(None)`
  Перезавантажує поточний файл інструментів.
- `[widget name].set_font('sans 16,tab='1')`
  Встановлює шрифт (Pango) для вкладки, заголовка стовпця та даних інструменту.
  Параметри `all_offsets`, `wear_offsets`, `tool_offsets` можна встановити одночасно, додавши 1, 2 та/або 3 до рядка вкладки.
  За замовчуванням встановлюються всі вкладки.
- `[widget name].set_title_font('sans 16,tab='1')`
  Встановлює шрифт (Pango) тільки для заголовків стовпців.
  Параметри `all_offsets`, `wear_offsets`, `tool_offsets` можна встановити одночасно, додавши 1, 2 та/або 3 до рядка табуляції.
  За замовчуванням встановлені всі табуляції.
- `[widget name].set_tab_font('sans 16,tab='1')`
  Встановлює шрифт (Pango) тільки на вкладках.
  Параметри `all_offsets`, `wear_offsets`, `tool_offsets` можна встановити одночасно, додавши 1, 2 та/або 3 до рядка вкладки.
  За замовчуванням встановлено всі вкладки.
- `[widget name].set_col_visible("abcUVW", False, tab='1')`
  Це приховає (False) стовпці abcuvw на вкладці 1 (all_offsets)
- `[widget name].set_lathe_display(value)`
  Приховує або показує вкладки зносу та зміщення інструменту, що використовуються для токарних верстатів
- `[widget name].get_toolinfo(toolnum)`
  Повертає масив інформації про інструмент із запитуваним номером інструмента або поточним інструментом, якщо номер інструмента не вказано.
  Повертає None, якщо інструмент не знайдено в таблиці або якщо поточного інструмента немає.
- `[widget name].hide_buttonbox(self, True)`
  Метод «Зручність» для приховування кнопок.
  Ви повинні викликати його після show_all().

- `[widget name].get_selected_tool()`
  Повернути номер вибраного користувачем (виділеного) інструмента.
- `[widget name].set_selected_tool(toolnumber)`
  Вибирає (підсвічує) запитуваний інструмент.

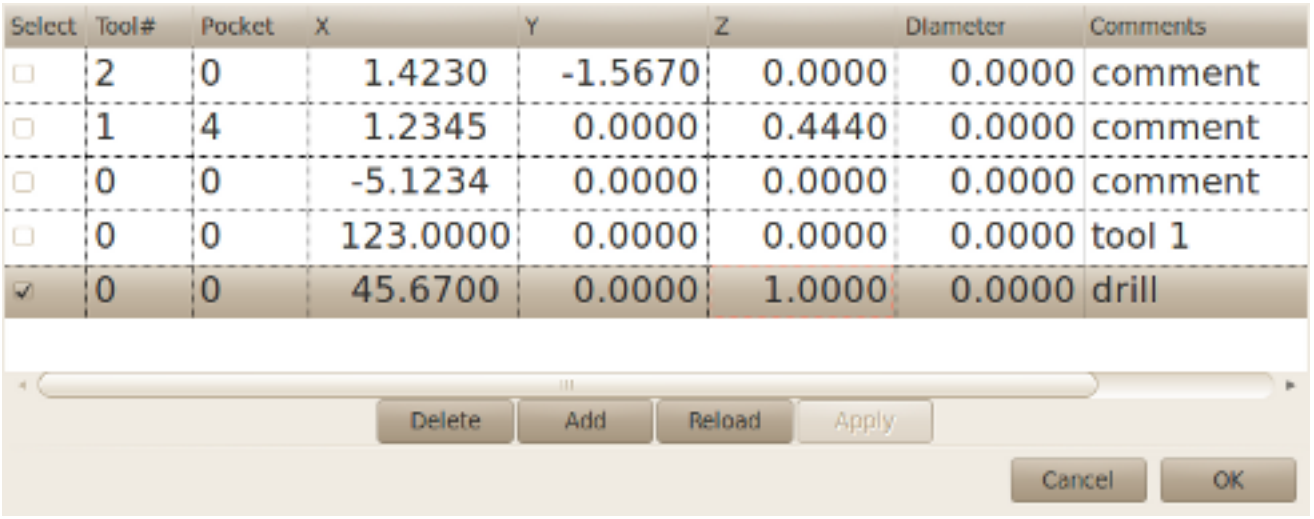


Figure 12.47: Приклад Tooleditor

12.3.6.26 Зсувна сторінка

Віджет «Offsetpage» використовується для відображення/редагування зміщень всіх осей.
Він має зручні кнопки для обнулення зміщень G92 і Rotation-Around-Z.
Ви можете вибрати режим редагування тільки тоді, коли верстат увімкнений і знаходиться в режимі очікування.
У цей час ви можете безпосередньо редагувати зміщення в таблиці. Зніміть позначку з кнопки редагування, щоб «OffsetPage» відобразив зміни.

Він має такі властивості:

display_units_mm
: Відображення в метричних одиницях

hide_columns
: Список стовпців без пробілів, які потрібно приховати. Стовпці позначені (по порядку) так: `xyzabcuvwt`.
Ви можете приховати будь-який зі стовпців.

hide_rows
: Список рядків без пробілів, які потрібно приховати.
Рядки позначені (по порядку) так: `0123456789abc`.
Ви можете приховати будь-який із рядків.

шрифт
: Встановлює тип і розмір шрифту тексту.

highlight_color
: Під час редагування це колір підсвічування.

foreground_color
Коли `OffsetPage` виявляє активну систему координат користувача, він використовуватиме цей колір для тексту.

mm_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.

imperial_text_template
Ви можете використовувати форматування Python для відображення позиції з різною точністю.

Існує кілька способів безпосереднього керування віджетом за допомогою Python.

Використання goobject для встановлення перелічених вище властивостей:

```
[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”)
```

Існують методи Python для керування віджетом:

- `[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)`
- `[назва віджета].set_font(”sans 20”)`
- `[widget name].set_highlight_color(”violet”)`
- `[widget name].set_foreground_color(”yellow”)`
- `[widget name].mark_active(”G55”)`
  Дозволяє безпосередньо встановити рядок для виділення, наприклад, якщо ви хочете використовува власні елементи керування навігацією. Див. розділ про GMOCCAPY.
- `[widget name].selection_mask = (”Tool”,”Rot”,”G5x”)`
  Ці рядки НЕ можна вибрати в режимі редагування.
- `[widget name].set_names([[’G54’,’Default’],[”G55”,”Vice1”],[’Rot’,’Rotational’]])`
  Це дозволяє вам встановити текст стовпця «T» кожного/будь-якого рядка.
  Це список пар «зсув-ім’я/ім’я користувача».
  Текст за замовчуванням такий самий, як і назва зсуву.
- `[widget name].get_names()`
  Це повертає список пар рядок-ключове слово/ім’я користувача.
  Стовпець з іменем користувача можна редагувати, тому збереження цього списку зручне для користувача.
  Див. `set_names` вище.

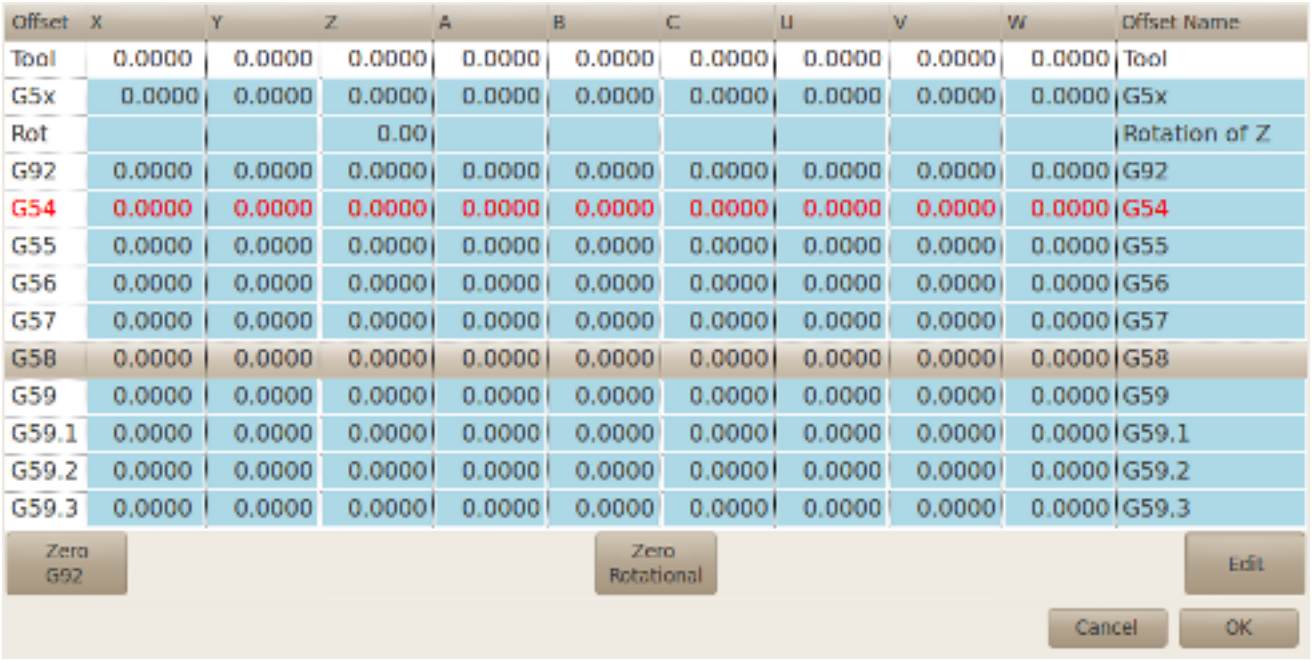


Figure 12.48: Приклад зсуву сторінки

12.3.6.27 Віджет HAL_sourceview

Це призначено для відображення та простого редагування G-коду. Він шукає `.ngc`, що підсвічує специфікації у `~/share/gtksourceview-4/language-specs/`. Поточний рядок, що виконується, буде підсвічено.

За допомогою зовнішнього коду Python glue він може:

- Пошук тексту, скасування та повторення змін.
- Використовується для вибору рядка програми.

Існують методи Python для керування віджетом:

- `[widget name].redo()`
  Повторити один рівень змін.
- `[widget name].undo()`
  Скасувати один рівень змін
- `[widget name].text_search(direction=True,mixed_case=True,text='G92')`
  Пошук уперед (напрямок = True) або назад,
  Пошук зі змішаним регістром (mixed_case = True) або точним збігом
- `[widget name].set_line_number(linenumber)`
  Встановлює виділення рядка.
  Використовує номери рядків з вихідного коду.
- `[widget name].get_line_number()`
  Повертає поточний виділений рядок.
- `[widget name].line_up()`
  Переміщує виділений рядок на один рядок угору.

- `[widget name].line_down()`
  Переміщує виділений рядок на один рядок вниз.
- `[widget name].load_file('filename')`
  Завантажує файл.
  Використання параметра «Немає» (не рядок імені файлу) призведе до перезавантаження тієї ж програми.
- `[widget name].get_filename()`
  Опис FIXME

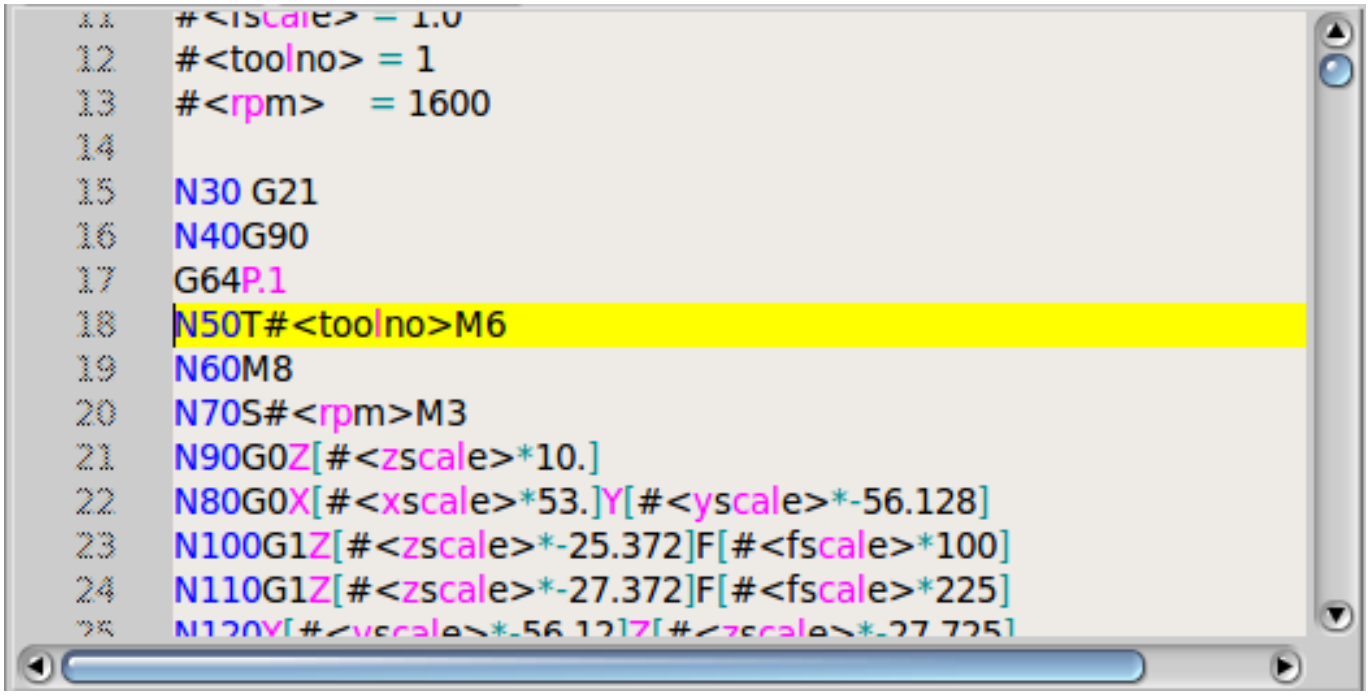


Figure 12.49: Приклад перегляду джерела

12.3.6.28 Історія MDI

Це призначено для відображення та введення кодів MDI.
Він автоматично стає сірим, коли MDI недоступний, наприклад, під час аварійної зупинки та виконання програми.

font_size_tree
Ціле число від 8 до 96.
Змінить розмір шрифту за замовчуванням для деревоподібного представлення на вибране значення.

font_size_entry
Ціле число від 8 до 96.
Змінить розмір шрифту за замовчуванням на вибране значення.

use_double_click
Булеве значення True вмикає функцію подвійного клацання мишею, і подвійне клацання на записі призведе до виконання цієї команди.
Не рекомендується використовувати цю функцію на реальних машинах, оскільки подвійне клацання на неправильному записі може призвести до небезпечних ситуацій.

Використання goobject для встановлення перелічених вище властивостей:

```
[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 Анімовані діаграми функцій: віджети HAL у растровому зображенні

Для деяких додатків може бути бажано мати фонове зображення, наприклад функціональну діаграму, і розміщувати віджети у відповідних місцях цього зображення. Хорошим поєднанням є встановлення растрового фонового зображення, наприклад, з файлу .png, встановлення фіксованого розміру вікна GladeVCP та використання віджета Glade Fixed для розміщення віджетів на цьому зображенні. Код для наведеного нижче прикладу можна знайти в `configs/apps/gladevcp/animated-b`

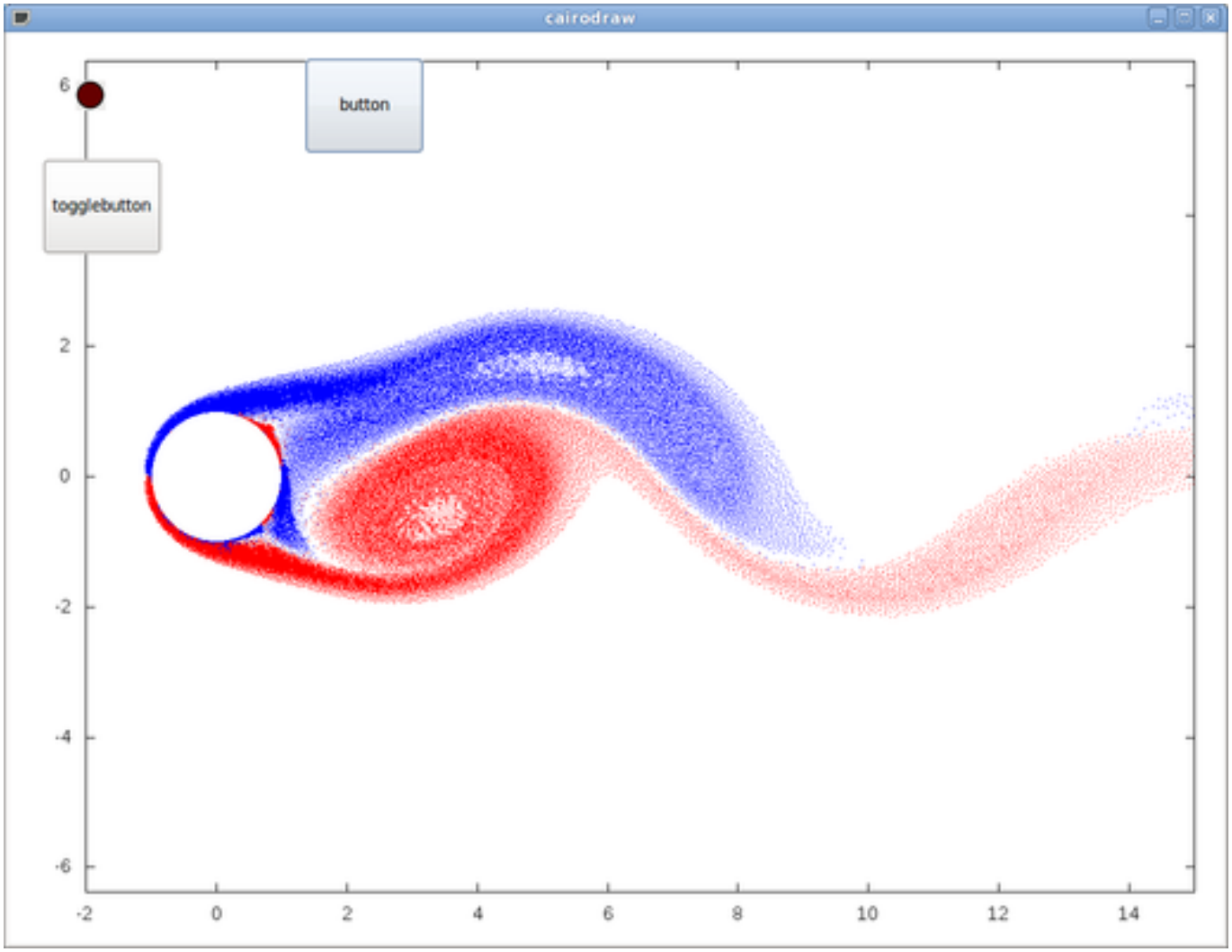


Figure 12.50: Приклад віджетів HAL у растровому зображенні

12.3.7 Довідник з віджетів дій

GladeVCP містить колекцію «готових дій» під назвою **Віджети дій VCP** для редактора інтерфейсу користувача Glade.

Note
Окрім віджетів HAL, які взаємодіють з контактами HAL, дії VCP взаємодіють з LinuxCNC та інтерпретатором G-коду.

Віджети дій VCP походять від віджета `Gtk.Action`.

Коротко про віджет «Дія»:

- Це об’єкт, доступний у Glade
- Сам по собі він не має візуального вигляду
- Його призначення: пов’язати видимий, чутливий компонент інтерфейсу користувача, такий як меню, кнопка інструмента, кнопка, з командою. Див. властивість цих віджетів «Загальні→Пов’яза
- «Запланована дія» буде виконана, коли буде запущено відповідний компонент інтерфейсу користува (натискання кнопки, клік меню тощо).
- Це забезпечує простий спосіб виконання команд без звернення до програмування на Python.

Зовнішній вигляд дій VCP у Glade приблизно такий:

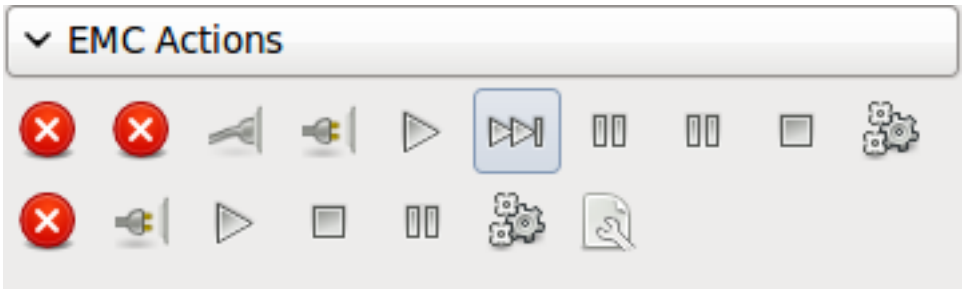


Figure 12.51: Віджети дій

Підказки під час наведення курсора надають опис.

12.3.7.1 Віджети дій VCP

Віджети дій VCP – це віджети одноразового типу. Вони реалізують одну дію та призначені для використання в простих кнопках, пунктах меню або групах радіо/перемикачів.

12.3.7.2 VCP Action Python

Цей віджет використовується для виконання невеликого довільного коду Python.

У командному рядку можуть використовуватися спеціальні ключові слова для доступу до важливих функцій.

- *ACTION* для доступу до бібліотеки команд ACTION.
- `GSTAT` для доступу до бібліотеки повідомлень про статус Gstat.
- *INFO* для доступу до зібраних даних з INI-файлу.
- *HAL* для доступу до модуля Python HAL linuxcnc.

- `STAT` для доступу до необробленого стану LinuxCNC через модуль LinuxCNC Python.
- `CMD` для доступу до команд LinuxCNC через модуль LinuxCNC Python.
- `EXT` для доступу до функцій файлу обробника, якщо такі є.
- `linuxcnc` для доступу до модуля LinuxCNC Python.
- `self` для доступу до екземпляра віджета.
- *dir* для доступу до списку атрибутів обробників.

Є варіанти

- виберіть, коли віджет буде активним,
- встановити режим перед виконанням команди.

Приклад команди для простого виведення повідомлення в термінал:

```
print('action activated')
```

Приклад команди для вимкнення машини:

```
CMD.state(linuxcnc.STATE_OFF)
```

Приклад команди для виклику функції-обробника, яка передає дані:

```
EXT.on_button_press(self, 100)
```

Ви можете використовувати крапку з комою для розділення кількох команд;

```
print('b''Bb''b''иb''b''мb''b''кb''b''нb''b''уb''b''тb''b''иb'' b''мb''b''аb''b''шb''b' ↩
    'иb''b''нb''b''уb''');CMD.state(linuxcnc.STATE_OFF)
```

Більше інформації про INFO та ACTION можна знайти тут: Модулі бібліотек GladeVCP.

Більше інформації про GStat можна знайти тут: GStat.

12.3.7.3 Віджети VCP ToggleAction

Це **бімодальні** віджети. Вони реалізують дві дії або використовують другий стан (зазвичай «натиснутий») для вказівки, що в даний момент виконується дія. Дії перемикання призначені для використання в «ToggleButtons», «ToggleToolButtons» або для перемикання елементів меню. Простим прикладом є кнопка перемикання «ESTOP».

Наразі доступні такі віджети:

- Перемикач `ESTOP` надсилає команди `ESTOP` або `ESTOP_RESET` до LinuxCNC залежно від свого стану.
- Перемикач `ON/OFF` надсилає команди `STATE_ON` та `STATE_OFF`.
- `Pause/Resume` надсилає команди `AUTO_PAUSE` або `AUTO_RESUME`.

Наведені нижче дії перемикання мають лише одну пов'язану команду та використовують стан `натиснуто`, щоб вказати, що запитувана операція виконується:

- Перемикач `Run` надсилає команду `AUTO_RUN` та чекає в натиснутому стані, поки інтерпретатор знову не буде в режимі очікування.
- Перемикач `Stop` неактивний, доки інтерпретатор не перейде в активний стан (не почне виконувати G-код), а потім не дозволить користувачеві надіслати команду `AUTO_ABORT`.
- Перемикач `MDI` надсилає задану MDI-команду та очікує її завершення в неактивному стані `натиснутого`.

12.3.7.4 Віджети Action_MDI Toggle та Action_MDI

Ці віджети забезпечують можливість виконання довільних команд MDI.
Віджет `Action_MDI` не чекає на завершення команди, як це робить перемикач `Action_MDI`, який залишається вимкненим до завершення команди.

12.3.7.5 Простий приклад: Виконання команди MDI при натисканні кнопки

`configs/apps/gladevcp/mdi-command-example/whoareyou.ui` це файл інтерфейсу Glade, який містить основні функції:

1. Відкрийте його в Glade та вивчіть, як це робиться.
2. Запустіть AXIS, а потім запустіть його з вікна терміналу за допомогою `gladevcp whoareyou.ui`.
3. Дивіться дію `hal_action_mdi1` та її властивість `MDI command` — вона просто виконує `(MSG, "Hi, I'm an VCP_Action_MDI")`, тому в AXIS має з'явитися спливаюче повідомлення, ось так:

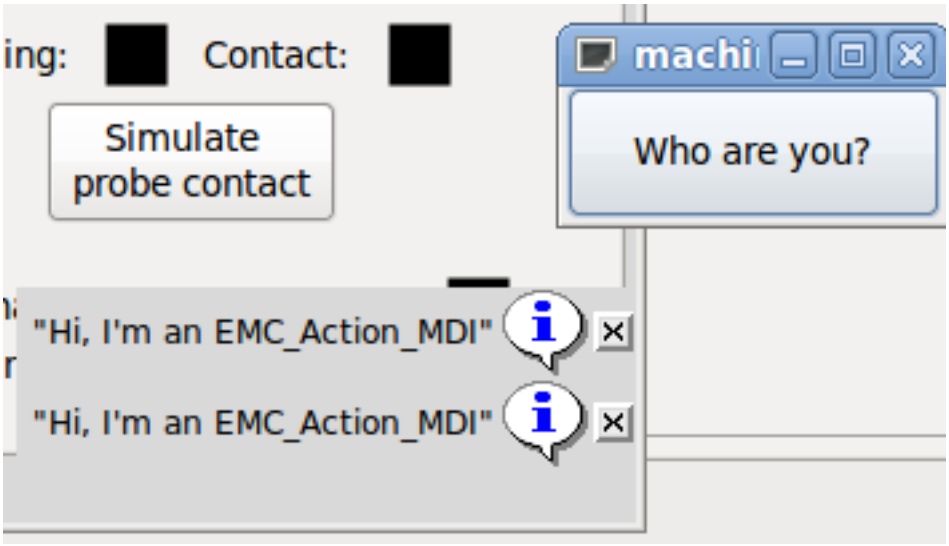


Figure 12.52: Простий приклад Action_MDI

Ви помітите, що кнопка, пов'язана з дією Action_MDI, буде сірою, якщо машина вимкнена, знаходиться в режимі аварійної зупинки або якщо працює інтерпретатор. Вона автоматично стане активною, коли машина буде увімкнена, вийде з режиму аварійної зупинки, а програма буде в режимі очікування.

12.3.7.6 Передача параметрів за допомогою віджетів Action_MDI та ToggleAction_MDI

Опціонально, рядки `MDI command` можуть мати параметри, які замінюються перед тим, як вони передаються інтерпретатору. Наразі параметрами можуть бути імена виводів HAL у компоненті GladeVCP. Ось як це працює:

- Припустимо, у вас є «HAL SpinBox» з назвою `speed`, і ви хочете передати його поточне значення як параметр у команді MDI.

- HAL SpinBox матиме висновок HAL з плаваючою точкою під назвою speed-f (див. опис HalWidgets).
- Щоб підставляти це значення в команду MDI, вставте назву виводу HAL, що міститься в ній, ось так: `${pin-name}`
- Для вищезгаданого HAL SpinBox ми могли б використовувати `(MSG, "Швидкість: ${speed-f}")`, щоб просто показати, що відбувається.

Приклад файлу інтерфейсу користувача — `configs/apps/gladevcp/mdi-command-example/speed.ui`. Ось що ви отримаєте після його запуску:

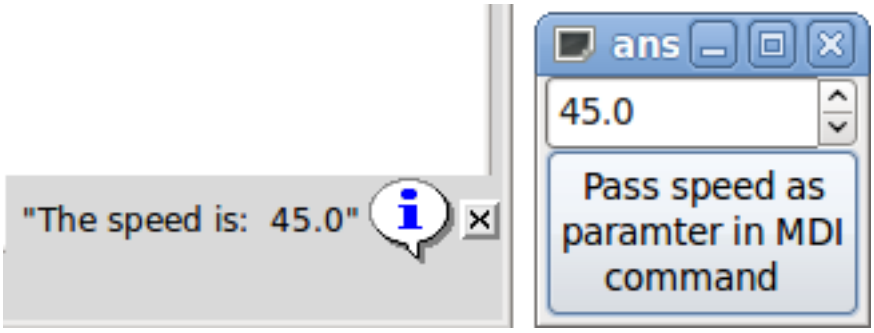


Figure 12.53: Приклад передачі параметрів Action_MDI

12.3.7.7 Розширений приклад: передача параметрів підпрограмі з буквою "O"

Цілком нормально викликати підпрограму з літерою O в команді MDI та передавати значення виводів HAL як фактичні параметри. Приклад файлу інтерфейсу користувача знаходиться в `configs/apps/gladevcp/mdi-command-example/owordsub.ui`.

Розмістіть `nc_files/gladevcp_lib/oword.ngc`, щоб AXIS міг його знайти, та запустіть `gladevcp owordsub.ui` з вікна терміналу. Це виглядає так:

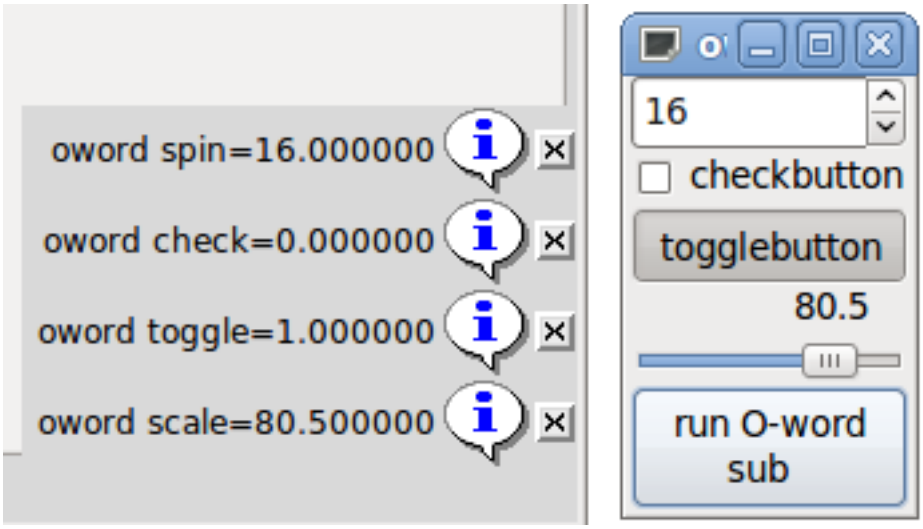


Figure 12.54: Розширений приклад Action_MDI

12.3.7.8 Підготовка до дії MDI та очищення після неї

Інтерпретатор G-коду LinuxCNC має єдиний глобальний набір змінних, таких як подача, швидкість шпинделя, відносний/абсолютний режим та інші. Якщо ви використовуєте команди G-коду або підпрограми O-word, деякі з цих змінних можуть бути змінені командою або підпрограмою - наприклад, підпрограма зондування, швидше за все, встановить досить низьке значення подачі. Без додаткових запобіжних заходів ваше попереднє налаштування подачі буде перезаписано значенням підпрограми зондування.

Щоб впоратися з цим несподіваним і небажаним побічним ефектом певної підпрограми O-word або оператора G-code, що виконується за допомогою LinuxCNC ToggleAction_MDI, ви можете пов'язати обробники pre-MDI і post-MDI з певним LinuxCNC ToggleAction_MDI. Ці обробники є опціональними і надають можливість зберегти будь-який стан перед виконанням дії MDI, а потім відновити його до попередніх значень. Імена сигналів — `mdi-command-start` та `mdi-command-stop`; імена обробників можна встановити в Glade, як і будь-які інші обробники.

Ось приклад того, як значення каналу може бути збережено та відновлено такими обробниками (зверніть увагу, що канали команд та статусу LinuxCNC доступні як `self.linuxcnc` та `self.stat` через клас 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
```

Тільки віджет перемикання `Action_MDI` підтримує ці сигнали.

Note
У пізнішій версії LinuxCNC будуть доступні нові M-коди M70-M72. Вони значно спростять збереження стану перед викликом підпрограми та відновлення стану після повернення.

12.3.7.9 Використання об'єкта LinuxCNC Stat для обробки змін статусу

Багато дій залежать від стану LinuxCNC - чи знаходиться він у ручному, MDI або автоматичному режимі? Чи програма працює, призупинена або перебуває в режимі очікування? Ви не можете запустити команду MDI під час виконання програми G-коду, тому це потрібно врахувати. Багато дій LinuxCNC самостійно враховують це, і відповідні кнопки та пункти меню деактивуються, коли операція є неможливою.

При використанні обробників подій Python, які знаходяться на нижчому рівні, ніж Actions, потрібно самостійно подбати про обробку залежностей статусу. Для цього існує віджет LinuxCNC Stat: для пов'язування змін статусу LinuxCNC з обробниками подій.

LinuxCNC Stat не має видимого компонента — ви просто додаєте його до свого інтерфейсу за допомогою Glade. Після додавання ви можете пов'язати обробники з його наступними сигналами:

- state-related:
  - `state-estop`: випромінюється, коли виникає умова аварійної зупинки,
  - `state-estop-reset`: випромінюється при скиданні машини,
  - `state-on`: випромінюється, коли машина ввімкнена,
  - `state-off`: випромінюється, коли машина вимкнена.

- mode-related:
  - `mode-manual`: видається, коли LinuxCNC переходить у ручний режим,
  - `mode-mdi`: видається, коли LinuxCNC переходить у режим MDI,
  - `mode-auto`: видається, коли LinuxCNC переходить у автоматичний режим,
- пов’язаний з інтерпретатором: видається, коли інтерпретатор G-коду переходить у цей режим
  - `interp-run`
  - `interp-idle`
  - `interp-paused`
  - `interp-reading`
  - `interp-waiting`
  - `file-loaded`
  - `line-changed`
- пов’язаний з перенаправленням: видається, коли LinuxCNC перенаправлено додому чи ні
  - `all-homed`
  - `not-all-homed`

12.3.8 Програмування GladeVCP

12.3.8.1 Дії, визначені користувачем

Більшість наборів віджетів та пов’язані з ними редактори користувацького інтерфейсу підтримують концепцію зворотних викликів, тобто функцій у коді, написаному користувачем, які виконуються, коли в інтерфейсі користувача «щось відбувається» — такі події, як клацання мишею, введення символів, рух миші, події таймера, приховування та відображення вікна тощо.

Віджети виводу HAL зазвичай відображають події типу вводу, такі як натискання кнопки, на зміну значення пов’язаного виводу HAL за допомогою такого - заздалегідь визначеного - зворотного виклику. У PyVCP це єдиний тип обробки подій, що підтримується - більш складні дії, такі як виконання команд MDI для виклику підпрограми G-коду, не підтримуються.

У GladeVCP зміни виводів HAL є лише одним із типів загального класу подій (званих сигналами) у GTK+. Більшість віджетів можуть генерувати такі сигнали, а редактор Glade підтримує асоціювання таких сигналів з іменами методів або функцій Python.

Якщо ви вирішили використовувати дії, визначені користувачем, ваше завдання полягає в тому, щоб написати модуль Python, методи класу якого — або, у простому випадку, просто функції — можуть бути використані в Glade як обробники подій. GladeVCP надає можливість імпортувати ваші модулі під час запуску і автоматично пов’язує ваші обробники подій із сигналами віджетів, як це встановлено в описі інтерфейсу користувача Glade.

12.3.8.2 Основна бібліотека

Існує три бібліотеки функцій, які можна використовувати для програмування GladeVCP.

- «Інформація»: збирає дані з INI-файлу.
- *Action*: Колекція функцій для зміни станів LinuxCNC.
- *Status*: Повідомляє про стан LinuxCNC. Він обгортається навколо *Gstat*.

Імпорт та створення екземплярів бібліотек:

```
b''зb'' gladevcp.core b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' b''Ib''b''нb''b''фb''b' ↩
    'ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''яb'', b''Дb''b''ib''b''яb''

ACTION = Action()
INFO = Info()
```

Використання функцій бібліотеки:

```
print(INFO.MACHINE_IS_METRIC)
ACTION.SET_ERROR_MESSAGE('b''Щb''b''ob''b''cb''b''ьb'' b''пb''b''ib''b''шb''b''лb''b''ob''  ↩
    b''нb''b''eb'' b''тb''b''ab''b''кb''')
```

Більш детальну інформацію можна знайти тут: GladeVCP Libraries modules. Існує зразок конфігурації, який демонструє використання основної бібліотеки з віджетами Python GladeVCP та файлом обробника Python. Спробуйте завантажити *sim/axis/gladevcp/gladevcp_panel_tester*.

12.3.8.3 Приклад: додавання користувацьких зворотних викликів у Python

Це лише мінімальний приклад для передачі ідеї — деталі викладено в решті цього розділу.

GladeVCP може не лише маніпулювати або відображати піни HAL, але й писати звичайні обробники подій на Python. Це можна використовувати, серед іншого, для виконання команд MDI. Ось як це зробити:

Напишіть модуль Python подібним чином і збережіть його, наприклад, як handlers.py:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

У Glade визначте кнопку або кнопку HAL, виберіть вкладку «Сигнали» та у властивостях GtkButton виберіть рядок «натиснуто». Введіть там «on_button_press» та збережіть файл Glade.

Потім додайте опцію *-u handlers.py* до командного рядка GladeVCP. Якщо ваші обробники подій розподілені по кількох файлах, просто додайте кілька опцій *-u <pyfilename>*.

Тепер натискання кнопки має змінити її мітку, оскільки вона встановлена у функції зворотного виклику.

Що робить прапор `+-u+`: всі функції Python у цьому файлі збираються і налаштовуються як потенційні обробники зворотного виклику для ваших віджетів Gtk — на них можна посилатися з вкладок «Сигнали» Glade. Обробники зворотного виклику викликаються з конкретним екземпляром об'єкта як параметром, як екземпляр GtkButton вище, тому ви можете застосувати будь-який метод GtkButton звідти.

Або зробіть щось корисніше, наприклад, виклик команди MDI!

12.3.8.4 Події зміни значення HAL

Вхідні віджети HAL, як і світлодіод, автоматично пов'язують стан свого виводу HAL (увімкнено/вимкнен з оптичним виглядом віджета (світиться/не світиться світлодіод).

Крім цієї вбудованої функції, можна пов'язати зворотний виклик зміни з будь-яким контактом HAL, включаючи контакти попередньо визначених віджетів HAL. Це добре відповідає структурі типового віджета, що керується подіями: кожна дія, будь то клацання мишею, натискання клавіші,

закінчення часу таймера або зміна значення контакту HAL, генерує зворотний виклик і обробляється тим самим ортогональним механізмом.

Для визначених користувачем виводів HAL, не пов'язаних з певним віджетом HAL, назва сигналу — «value-changed». Див. розділ Додавання виводів HAL нижче для отримання детальної інформації.

Віджети HAL постачаються із попередньо визначеним сигналом під назвою «hal-pin-changed». Докладніше див. у розділі Віджети HAL.

12.3.8.5 Модель програмування

Загальний підхід такий:

- Створіть свій інтерфейс користувача за допомогою Glade та встановіть обробники сигналів там, де ви хочете, щоб дії були пов'язані з віджетом.
- Напишіть модуль Python, який містить викликані об'єкти (див. «моделі обробників» нижче).
- Передайте шлях до вашого модуля до GladeVCP за допомогою опції *-u <модуль>*.
- GladeVCP імпортує модуль, перевіряє його на наявність обробників сигналів та підключає їх до дерева віджетів.
- Виконується основний цикл подій.

Для простих завдань достатньо визначити функції, названі на честь обробників сигналів Glade. Вони будуть викликані, коли відповідна подія відбудеться в дереві віджетів. Ось простий приклад - він передбачає, що сигнал «pressed» кнопки Gtk або HAL пов'язаний з функцією зворотного виклику «on_button_press»:

```
nhits = 0
def on_button_press(gtkobj,data=None):
    global nhits
    nhits += 1
    gtkobj.set_label("hits: %d" % nhits)
```

Додайте цю функцію до файлу Python та виконайте її наступним чином:

```
gladevcp -u <myhandler>.py mygui.ui
```

Зверніть увагу, що зв'язок між обробниками має проходити через глобальні змінні, що погано масштабується та є абсолютно непітонічним. Саме тому ми розробили модель обробника на основі класів.

Ідея полягає в наступному: обробники пов'язані з методами класу. Базові класи інстанціюються та перевіряються під час запуску GladeVCP і пов'язуються з деревом віджетів як обробники сигналів. Отже, завдання полягає в тому, щоб написати:

- Одне або декілька визначень класів з одним або кількома методами, в одному модулі або розділених на кілька модулів,
- функція *get_handlers* у кожному модулі, яка повертатиме список екземплярів класів до GladeVCP - назви їхніх методів будуть пов'язані з обробниками сигналів.

Ось приклад мінімального модуля обробника, визначеного користувачем:

```
class MyCallbacks :
    def on_this_signal(self,obj,data=None):
        print("this_signal happened, obj=",obj)

def get_handlers(halcomp,builder,useropts):
    return [MyCallbacks ()]
```

Тепер *on_this_signal* буде доступний як обробник сигналів для вашого дерева віджетів.

Для панелі GladeVCP, яка реагує на вхідні сигнали HAL, може бути важливо, щоб код обробника міг визначити, що панель GladeVCP наразі активна і відображається. Наприклад, панель всередині інтерфейсу Touchy може потребувати виконання дії, коли перемикач, підключений до touchy.cycle-start, активований (так само, як і вбудовані вкладки по-різному реагують на одну і ту ж кнопку). Щоб це стало можливим, з графічного інтерфейсу користувача (на момент написання статті — тільки Touchy) на вбудовану вкладку надсилається сигнал. Сигнал має тип «Gladevcp», а два надсилаються повідомлення — «Visible» і «Hidden». (Зверніть увагу, що сигнали мають фіксовану довжину 20 символів, тому для порівняння слід використовувати тільки перші символи, звідси [:7] нижче). Приклад обробника для цих сигналів:

```
    # b''Цb''b''eb'' b''пb''b''eb''b''pb''b''eb''b''xb''b''ob''b''пb''b''лb''b''юb''b''єb'' ↩
         b''нb''b''ab''b''шb''b''ib'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b' ↩
        'лb''b''eb''b''нb''b''нb''b''яb'' b''зb'' b''ib''b''нb''b''шb''b''ob''b''їb'' b' ↩
        'пb''b''pb''b''ob''b''гb''b''pb''b''ab''b''мb''b''иb''
    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

    # b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''eb''b''нb''b''нb''b''яb'' b''дb'' ↩
        b''ob'' b''кb''b''лb''b''ib''b''єb''b''нb''b''тb''b''cb''b''ьb''b''кb''b''иb''b' ↩
        'xb'' b''пb''b''ob''b''дb''b''ib''b''йb'' b''зb'' b''гb''b''pb''b''ab''b''фb''b' ↩
        'ib''b''чb''b''нb''b''ob''b''гb''b''ob'' b''ib''b''нb''b''тb''b''eb''b''pb''b''фb''b ↩
        ''eb''b''йb''b''cb''b''yb'' b''xb''b''ob''b''cb''b''тb''b''ab''
    def on_map_event(self, widget, data=None):
        top = widget.get_toplevel()
        print("map event")
        top.connect('client-event', self.event)
```

Якщо під час перевірки модуля GladeVCP знаходить функцію `get_handlers`, він викликає її наступним чином:

```
get_handlers(halcomp,builder,useropts)
```

Аргументи такі:

- `halcomp` - стосується компонента HAL, що знаходиться в процесі розробки,
- `builder` - дерево віджетів — результат читання визначення інтерфейсу користувача (або посилання на об'єкт типу GtkBuilder, або libglade),
- `useropts` - список рядків, зібраних з опції `-U <користувачі>` командного рядка GladeVCP.

Потім GladeVCP перевіряє список екземплярів класів і отримує їхні імена методів. Відповідні імена методів підключаються до дерева віджетів як обробники сигналів. Враховуються тільки імена методів, які не починаються з символу «_» (підкреслення).

Зверніть увагу, що незалежно від того, чи використовуєте ви libglade чи новий формат GtkBuilder для вашого інтерфейсу Glade, до віджетів завжди можна звернутися як `builder.get_object(<widgetna`
Також повний список віджетів доступний як `builder.get_objects()` незалежно від формату інтерфейсу.

12.3.8.6 Послідовність ініціалізації

Важливо знати, в якому стані викликається ваша функція `get_handlers()`, щоб розуміти, що в цьому стані можна робити, а що ні. Спочатку модулі імпортуються та ініціалізуються в порядку

командного рядка. Після успішного імпорту функція `get_handlers()` викликається в такому стані:

- Дерево віджетів створено, але ще не реалізовано (жодна функція верхнього рівня `window.show()` ще не виконана).
- Компонент HAL halcomp налаштовано, і всі піни віджетів HAL вже додано до нього.
- Можна безпечно додавати більше пінів HAL, оскільки `halcomp.ready()` на даний момент ще не викликано, тому ви можете додати свої власні піни, наприклад, у методі класу `__init__()`.

Після імпорту всіх модулів та вилучення назв методів, виконуються такі кроки:

- Усі відповідні імена методів будуть підключені до дерева віджетів за допомогою `connect_signals()/s` (залежно від типу імпортованого інтерфейсу користувача - GtkBuilder чи старий формат libglade).
- Компонент HAL завершується за допомогою halcomp.ready().
- Якщо як аргумент було передано ідентифікатор вікна, дерево віджетів переоформлюється для запуску в цьому вікні, а вікно верхнього рівня Glade window1 відкидається (див. FAQ).
- Якщо файл команди HAL було передано з параметром `-H halfile`, він виконується за допомогою halcmd.
- Виконується головний цикл Gtk.

Отже, коли ваш клас обробника ініціалізується, всі віджети існують, але ще не реалізовані (не відображаються на екрані). Компонент HAL також не готовий, тому доступ до значень контактів у вашому методі `__init__()` є небезпечним.

Якщо ви хочете, щоб при запуску програми виконувався зворотний виклик після того, як доступ до контактів HAL стане безпечним, підключіть обробник до сигналу реалізації вікна верхнього рівня window1 (що, можливо, є його єдиною реальною метою). На цьому етапі GladeVCP завершує всі завдання налаштування, файл HAL запущено, і GladeVCP готується увійти в головний цикл Gtk.

12.3.8.7 Кілька зворотних викликів з однаковим ім’ям

У межах класу імена методів повинні бути унікальними. Однак можна передавати до GladeVCP за допомогою get_handlers() кілька екземплярів класу з методами, що мають однакові імена. Коли з’являється відповідний сигнал, ці методи викликаються в порядку визначення — модуль за модулем, а в межах модуля — в порядку, в якому екземпляри класу повертаються за допомогою `get_handlers()`.

12.3.8.8 Прапорець GladeVCP `-U <користувачі>`

Замість того, щоб розширювати GladeVCP для будь-якої можливої опції, яка потенційно може бути корисною для класу обробника, ви можете використовувати прапор -U <useroption> (повторюючи його, якщо бажаєте). Цей прапор збирає список рядків <useroption>. Цей список передається до функції get_handlers() (аргумент useropts). Ваш код може вільно інтерпретувати ці рядки на свій розсуд. Одним із можливих варіантів використання є передача їх до функції Python exec у вашій функції `get_handlers()` наступним чином:

```
debug = 0
...
def get_handlers(halcomp,builder,useropts):
    ...
```

```
    global debug # b''пb''b''pb''b''иb''b''пb''b''yb''b''cb''b''кb''b''ab''b''юb''b''чb''b' ↩
        'иb'', b''щb''b''ob'' b''ib''b''cb''b''нb''b''yb''b''єb'' b''гb''b''лb''b''ob''b' ↩
        'бb''b''ab''b''лb''b''ьb''b''нb''b''ab'' b''зb''b''мb''b''ib''b''нb''b''нb''b''ab''
    for cmd in useropts:
        exec cmd in globals()
```

Таким чином, ви можете передавати довільні оператори Python до вашого модуля за допомогою опції `gladevcp -U`, наприклад:

```
gladevcp -U debug=42 -U "print 'debug=%d' % debug" ...
```

Це має встановити debug на 2 та підтвердити, що ваш модуль справді це зробив.

12.3.8.9 Постійні змінні в GladeVCP

Неприємним аспектом GladeVCP у його попередній версії та PyVCP є те, що ви можете змінювати значення та контакти HAL за допомогою текстового введення, повзунків, полів вибору, перемикачів тощо, але їхні налаштування не зберігаються і не відновлюються при наступному запуску LinuxCNC — вони починають працювати зі значеннями за замовчуванням, встановленими в панелі або визначенні віджета.

GladeVCP має простий у використанні механізм для збереження та відновлення стану віджетів HAL та змінних програми (фактично будь-якого атрибута екземпляра типу int, float, bool або string).

Цей механізм використовує популярний формат INI-файлу для збереження та перезавантаження постійних атрибутів.

Збереження, версії програми та перевірка підпису Уявіть, що ви перейменовуєте, додаєте або видаляєте віджети в Glade: .INI-файл, що залишився від попередньої версії програми, або зовсім інший користувацький інтерфейс не зможуть правильно відновити стан, оскільки змінні та типи могли змінитися.

GladeVCP виявляє цю ситуацію за допомогою сигнатури, яка залежить від усіх імен та типів об'єктів, що зберігаються та підлягають відновленню. У разі невідповідності сигнатури створюється новий файл INI із стандартними налаштуваннями.

12.3.8.10 Використання постійних змінних

Якщо ви хочете, щоб будь-який стан віджета Gtk, значення виводу PIN-кодів та/або атрибути класу вашого класу-обробника зберігалися під час викликів, виконайте такі дії:

- Імпортуйте модуль `gladevcp.persistence`.
- Визначте, які атрибути екземпляра та їхні значення за замовчуванням ви хочете зберегти, якщо такі є.
- Визначте, стан яких віджетів слід зберегти.
- Опишіть ці рішення у методі `\__init()__` вашого класу-обробника через вкладений словник наступним чином:

```
def __init__(self, halcomp,builder,useropts):
    self.halcomp = halcomp
    self.builder = builder
    self.useropts = useropts
    self.defaults = {
```

```
        # b''нв''b''ав''b''св''b''тв''b''ув''b''пв''b''нв''b''ів'' b''ів''b''мв''b''ев''b' ↩
            'нв''b''ав'' b''бв''b''ув''b''дв''b''ув''b''тв''b''ьв'' b''зв''b''бв''b''ев''b' ↩
            'рв''b''ев''b''жв''b''ев''b''нв''b''ів''/b''вв''b''ів''b''дв''b''нв''b''ов''b' ↩
            'вв''b''лв''b''ев''b''нв''b''ів'' b''яв''b''кв'' b''ав''b''тв''b''рв''b''ив''b' ↩
            'бв''b''ув''b''тв''b''ив'' b''мв''b''ев''b''тв''b''ов''b''дв''b''ув''
        # b''мв''b''ев''b''хв''b''ав''b''нв''b''ів''b''зв''b''мв'' b''зв''b''бв''b''ев''b' ↩
            'рв''b''ев''b''жв''b''ев''b''нв''b''нв''b''яв''/b''вв''b''ів''b''дв''b''нв''b' ↩
            'ов''b''вв''b''лв''b''ев''b''нв''b''нв''b''яв'' b''єв'' b''св''b''тв''b''рв''b' ↩
            'ов''b''гв''b''ов'' b''тв''b''ив''b''пв''b''ів''b''зв''b''ов''b''вв''b''ав''b' ↩
            'нв''b''ив''b''мв'' - b''тв''b''ив''b''пв'' b''зв''b''мв''b''ів''b''нв''b''нв''b ↩
            ''ив''b''хв'' b''бв''b''ув''b''дв''b''ев'' b''пв''b''ов''b''хв''b''ів''b''дв''b' ↩
            'нв''b''ив''b''мв'' b''вв''b''ів''b''дв'' b''тв''b''ив''b''пв''b''ув''
        # b''ів''b''нв''b''ів''b''цв''b''ів''b''ав''b''лв''b''ів''b''зв''b''ав''b''цв''b' ↩
            'ів''b''йв''b''нв''b''ов''b''гв''b''ов'' b''зв''b''нв''b''ав''b''чв''b''ев''b' ↩
            'нв''b''нв''b''яв''. b''Нв''b''ав''b''рв''b''ав''b''зв''b''ів'' b''пв''b''ів''b' ↩
            'дв''b''тв''b''рв''b''ив''b''мв''b''ув''b''юв''b''тв''b''ьв''b''св''b''яв'' b' ↩
            'тв''b''ав''b''кв''b''ів'' b''тв''b''ив''b''пв''b''ив'': int, float, bool, ↩
            string
        IniFile.vars : { 'nhits' : 0, 'a': 1.67, 'd': True ,'c' : "a string"},
        # b''щв''b''ов''b''бв'' b''зв''b''бв''b''ев''b''рв''b''ев''b''гв''b''тв''b''ив''/b' ↩
            'вв''b''ів''b''дв''b''нв''b''ов''b''вв''b''ив''b''тв''b''ив'' b''вв''b''ев''b' ↩
            'св''b''ьв'' b''св''b''тв''b''ав''b''нв'' b''вв''b''ів''b''дв''b''жв''b''ев''b' ↩
            'тв''b''ав'', b''яв''b''кв''b''ив''b''йв'' b''мв''b''ов''b''жв''b''ев'' b''мв''b ↩
            ''ав''b''тв''b''ив'' b''св''b''ев''b''нв''b''св'', b''дв''b''ов''b''дв''b''ав''b ↩
            ''йв''b''тв''b''ев'' b''цв''b''ев'':
        IniFile.widgets : widget_defaults(builder.get_objects())
        # b''рв''b''ов''b''зв''b''ув''b''мв''b''нв''b''ов''b''юв'' b''ав''b''лв''b''ьв''b' ↩
            'тв''b''ев''b''рв''b''нв''b''ав''b''тв''b''ив''b''вв''b''ов''b''юв'' b''мв''b' ↩
            'ов''b''жв''b''ев'' b''бв''b''ув''b''тв''b''ив'' b''зв''b''бв''b''ев''b''рв''b' ↩
            'ев''b''жв''b''ев''b''нв''b''нв''b''яв'' b''лв''b''ив''b''шв''b''ев'' b''св''b' ↩
            'тв''b''ав''b''нв''b''ув'' b''вв''b''св''b''ів''b''хв'' b''вв''b''ів''b''дв''b' ↩
            'жв''b''ев''b''тв''b''ів''b''вв'' b''вв''b''ив''b''вв''b''ов''b''дв''b''ув'' HAL ↩
            :
        # IniFile.widgets: widget_defaults(select_widgets(self.builder.get_objects(), ↩
            hal_only=True,output_only = True)),
    }
```

Потім пов'яжіть INI-файл з цим дескриптором:

```
self.ini_filename = __name__ + '.ini'
self.ini = IniFile(self.ini_filename,self.defaults,self.builder)
self.ini.restore_state(self)
```

Після `restore_state()`, self матиме встановлені атрибути, якщо виконати наступне:

```
self.nhits = 0
self.a = 1.67
self.d = True
self.c = "a string"
```

Зверніть увагу, що типи зберігаються під час відновлення. У цьому прикладі припускається, що INI-файл не існував або мав значення за замовчуванням із self.defaults.

Після цього заклинання ви можете використовувати такі методи IniFile:

ini.save_state(obj)
: Зберігає атрибути objs згідно зі словником `IniFile.vars` та стан віджета, як описано в `IniFile.widgets` у `self.defaults`.

ini.create_default_ini()
: Створіть INI-файл зі значеннями за замовчуванням.

ini.restore_state(obj)
Відновіть вихідні виводи HAL та атрибути obj, як збережено/ініціалізовано за замовчуванням, як зазначено вище.

12.3.8.11 Збереження стану під час вимкнення GladeVCP

Щоб зберегти стан віджета та/або змінної після виходу, виконайте такі дії:

- Виберіть якийсь внутрішній віджет (тип не має значення, наприклад, таблиця).
- На вкладці «Сигнали» виберіть «GtkObject». У першому стовпці має відображатися сигнал «знищення».
- Додайте назву обробника, наприклад, «on_destroy», до другого стовпця.
- Додайте обробник Python, як показано нижче:

```
import gtk
...
def on_destroy(self,obj,data=None):
    self.ini.save_state(self)
```

Це збереже стан та належним чином вимкне GladeVCP, незалежно від того, чи панель вбудована в AXIS, чи є окремим вікном.



Caution
Не використовуйте `window1` (вікно верхнього рівня) для підключення події «destroy». Через спосіб взаємодії панелі GladeVCP з AXIS, якщо панель вбудована в AXIS, **window1 не буде правильно отримувати події destroy**. Однак, оскільки під час вимкнення всі віджети знищуються, підійде будь-який. Рекомендується: використовуйте віджет другого рівня — наприклад, якщо у вашій панелі є контейнер таблиці, використовуйте його.

Наступного разу, коли ви запустите програму GladeVCP, віджети повинні відображатися в тому ж стані, в якому програма була закрита.



Caution
Рядок *GtkWidget* має подібну подію *destroy-event* – **не використовуйте її для підключення до обробника *on_destroy*, це не працюватиме** – переконайтеся, що ви використовуєте подію *destroy* з рядка *GtkObject*.

12.3.8.12 Збереження стану при натисканні Ctrl-C

За замовчуванням реакція GladeVCP на подію Ctrl-C полягає у простому виході — без збереження стану. Щоб переконатися, що цей випадок враховано, додайте виклик обробника `on_unix_signal`, який автоматично викликатиметься при Ctrl-C (фактично при сигналах SIGINT і SIGTERM). Приклад:

```
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 Ручне редагування INI-файлів (.ini)

Ви можете це зробити, але зверніть увагу, що значення в self.defaults замінять ваші зміни, якщо у ваших змінах буде синтаксична або типова помилка. Якщо помилка буде виявлена, консольне повідомлення підкаже про те, що сталося, а пошкоджений файл inifile буде перейменований з додаванням суфікса .BAD. Наступні пошкоджені файли INI замінять попередні файли .BAD.

12.3.8.14 Додавання контактів HAL

Якщо вам потрібні піни HAL, які не пов'язані з певним віджетом HAL, додайте їх наступним чином:

```
import hal_glib
...
# b''yb'' b''вb''b''ab''b''шb''b''ob''b''мb''b''yb'' b''кb''b''лb''b''ab''b''cb''b''ib'' b' ↩
    'ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab'' __init__():
self.example_trigger = hal_glib.GPin(halcomp.newpin('example-trigger', hal.HAL_BIT, hal. ↩
    HAL_IN))
```

Щоб отримати зворотний виклик при зміні значення цього піна, пов'яжіть зворотний виклик `value-change` з цим піном, додайте:

```
self.example_trigger.connect('value-changed', self._on_example_trigger_change)
```

та визначте метод зворотного виклику (або функцію, у цьому випадку пропустіть параметр `self`):

```
# b''пb''b''pb''b''иb''b''мb''b''ib''b''тb''b''кb''b''ab'' '_' - b''цb''b''eb''b''йb'' b' ↩
    'мb''b''eb''b''тb''b''ob''b''дb'' b''нb''b''eb'' b''бb''b''yb''b''дb''b''eb'' b''вb''b' ↩
    'иb''b''дb''b''иb''b''мb''b''иb''b''мb'' b''дb''b''лb''b''яb'' b''дb''b''eb''b''pb''b' ↩
    'eb''b''вb''b''ab'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ib''b''вb''
def _on_example_trigger_change(self,pin,userdata=None):
    print("pin value changed to:" % (pin.get()))
```

12.3.8.15 Додавання таймерів

Оскільки GladeVCP використовує віджети Gtk, які базуються на базовому класі PyGObject, доступний повний функціонал GLib. Ось приклад зворотного виклику таймера:

```
def _on_timer_tick(self,userdata=None):
    ...
    return True # b''щb''b''ob''b''бb'' b''пb''b''eb''b''pb''b''eb''b''зb''b''ab''b''пb''b' ↩
        'yb''b''cb''b''тb''b''иb''b''тb''b''иb'' b''тb''b''ab''b''йb''b''мb''b''eb''b''pb''; ↩
         b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb'' False b''дb''b' ↩
        'лb''b''яb'' b''пb''b''ob''b''cb''b''тb''b''pb''b''ib''b''лb''b''yb''
...
# b''дb''b''eb''b''мb''b''ob''b''нb''b''cb''b''тb''b''pb''b''yb''b''вb''b''ab''b''тb''b' ↩
    'иb'' b''пb''b''ob''b''вb''b''ib''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''фb''b''ob''b' ↩
    'нb''b''ob''b''вb''b''иb''b''йb'' b''тb''b''ab''b''йb''b''мb''b''eb''b''pb'' b''—b'' b' ↩
    'тb''b''ob''b''чb''b''нb''b''ib''b''cb''b''тb''b''ьb'' b''вb''b''ib''b''дb''b''лb''b' ↩
    'ib''b''кb''b''yb'' b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''ьb'' b' ↩
    'ob''b''дb''b''нb''b''yb'' b''cb''b''eb''b''кb''b''yb''b''нb''b''дb''b''yb''
# b''дb''b''лb''b''яb'' b''шb''b''вb''b''иb''b''дb''b''шb''b''ob''b''гb''b''ob'' b''тb''b' ↩
    'ab''b''йb''b''мb''b''eb''b''pb''b''ab'' (b''зb'' b''тb''b''ob''b''чb''b''нb''b''ib''b' ↩
    'cb''b''тb''b''ю'' 100 b''мb''b''cb'') b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b' ↩
    'cb''b''тb''b''ob''b''вb''b''yb''b''йb''b''тb''b''eb'' b''цb''b''eb'':
# GLib.timeout_add(100, self._on_timer_tick,userdata) # 10Hz
GLib.timeout_add_seconds(1, self._on_timer_tick)
```

12.3.8.16 Програмне налаштування властивостей віджета HAL

У Glade властивості віджетів зазвичай встановлюються фіксовано під час редагування. Однак ви можете встановлювати властивості віджетів під час виконання, наприклад, із значень файлу INI, що зазвичай робиться в коді ініціалізації обробника. Також можна встановлювати властивості із значень виводів HAL.

У наступному прикладі (припускаючи, що віджет HAL Meter називається `meter`) мінімальне значення лічильника встановлюється з параметра INI-файлу під час запуску, а максимальне значення встановлюється за допомогою контакту HAL, що призводить до динамічного переналаштування шкали віджета:

```
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() # b''пb''b''pb''b''иb''b''мb''b''уb''b''cb''b''ob''b''вb''b ↩
            ''ob'' b''пb''b''eb''b''pb''b''eb''b''мb''b''ab''b''лb''b''юb''b''вb''b''ab''b' ↩
            'тb''b''иb'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''

    def __init__(self, halcomp,builder,useropts):
        self.builder = builder

        # HAL-b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb'' b''іb''b''зb'' b''зb''b' ↩
            'вb''b''ob''b''pb''b''ob''b''тb''b''нb''b''иb''b''мb'' b''вb''b''иb''b''кb''b' ↩
            'лb''b''иb''b''кb''b''ob''b''мb'' b''пb''b''pb''b''иb'' b''зb''b''мb''b''іb''b' ↩
            'нb''b''ib''.
        # b''Kb''b''ob''b''лb''b''иb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b' ↩
            'яb'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''уb'' b''зb''b''мb''b' ↩
            'іb''b''нb''b''юb''b''єb''b''тb''b''ьb''b''cb''b''яb'', b''вb''b''иb''b''кb''b' ↩
            'ob''b''нb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''зb''b''вb''b''ob''b' ↩
            'pb''b''ob''b''тb''b''нb''b''иb''b''йb'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
            'кb''.
        self.max_value = hal_glib.GPin(halcomp.newpin('max-value',  hal.HAL_FLOAT, hal. ↩
            HAL_IN))
        self.max_value.connect('value-changed', self._on_max_value_change)

        inifile = linuxcnc.ini(os.getenv("INI_FILE_NAME"))
        mmin = float(inifile.find("METER", "MIN") or 0.0)
        self.meter = self.builder.get_object('meter')
        self.meter.min = mmin


def get_handlers(halcomp,builder,useropts):
    return [HandlerClass(halcomp,builder,useropts)]
```

12.3.8.17 Зворотний виклик зі зміною значення за допомогою hal_glib

GladeVCP використовує бібліотеку hal_glib, яку можна використовувати для підключення сигналу "спостерігача" до вхідного виводу HAL.
Цей сигнал можна використовувати для реєстрації функції, яка викликається при зміні стану виводу HAL.

Потрібно імпортувати модулі `hal_glib` та `hal`:

```
import hal_glib
import hal
```

Потім зробіть пін-код і підключіть сигнал «зміни значення» (спостерігач) до виклику функції:

```
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)
```

І маємо функцію, яку потрібно викликати:

```
    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()))

        # b''цb''b''eb'' b''мb''b''ob''b''жb''b''нb''b''ab'' b''вb''b''иb''b''кb''b''лb''b' ↩
            'иb''b''кb''b''ab''b''тb''b''иb'' b''пb''b''ob''b''зb''b''ab'' b''фb''b''yb''b' ↩
            'нb''b''кb''b''цb''b''ib''b''єb''b''юb''
        self.example_trigger.get()
```

12.3.8.18 Приклади та створення власної програми GladeVCP

Відвідайте `linuxcnc_root_directory/configs/apps/gladevcp` для прикладів запуску та стартових ігор для ваших власних проектів.

12.3.9 Найчастіші запитання

1. *Я отримую неочікувану подію initar у своїй функції-обробнику одразу після запуску. Що це?*

   Це наслідок того, що у вашому файлі Glade UI властивість window1 Visible встановлена на True, а також переприв'язування вікна GladeVCP до AXIS або touchy. Створюється дерево віджетів GladeVCP, включаючи вікно верхнього рівня, а потім «перепризначається в AXIS», залишаючи це вікно верхнього рівня без батьків. Щоб уникнути наявності цього непотрібного порожнього вікна, воно відключається (стає невидимим), що є причиною сигналу відключення, який ви отримуєте. Пропоноване виправлення: встановіть для window1.visible значення False та ігноруйте початкову подію відключення.

2. *Моя програма GladeVCP запускається, але вікно не з'являється там, де я очікую?*

   Вікно, яке AXIS виділяє для GladeVCP, отримає «природний розмір» усіх своїх дочірніх віджетів разом. Запит розміру (ширини та/або висоти) є завданням дочірнього віджета. Однак не всі віджети запитують ширину, більшу за 0, наприклад, віджет Graph у його поточному вигляді. Якщо у вашому файлі Glade є такий віджет і саме він визначає макет, ви можете явно встановити його ширину. Зверніть увагу, що встановлення властивостей ширини та висоти window1 у Glade не має сенсу, оскільки це вікно буде втратити батьківський елемент під час переприв'язки, а отже, його геометрія не матиме впливу на макет (див. вище). Загальне правило: якщо ви вручну запускаєте файл UI за допомогою «gladevcp <uifile>» і його вікно має розумну геометрію, воно також має правильно відображатися в AXIS.

3. *Я хочу миготливий світлодіод, але він не мигає*

   Я поставив галочку, щоб він блимав з інтервалом 100 мс. Він не блимає, і я отримую попередження при запуску: Попередження: значення «0» типу «gint» є недійсним або виходить за межі діапазону для властивості «led-blink-rate» типу «gint»? Це, схоже, помилка Glade. Просто перезапишіть поле частоти блимання і збережіть знову — у мене це працює.

4. *Моя панель GladeVCP в AXIS не зберігає стан після закриття AXIS, хоча я визначив обробник on_destroy, пов'язаний із сигналом знищення вікна*

   Швидше за все, цей обробник пов'язаний з window1, який через переприв'язку не може бути використаний для цієї мети. Будь ласка, пов'яжіть обробник on_destroy із сигналом destroy внутрішнього вікна. Наприклад, у мене є ноутбук у window1, і я пов'язав on_destroy із сигналом destroy ноутбука, і це працює нормально. Це не працює для window1.

5. *Я хочу встановити колір фону або текст віджета HAL_Label залежно від значення його виводу HAL*

   Дивіться приклад у configs/apps/gladevcp/colored-label. Налаштування кольору фону віджета GtkLabel (а HAL_Label походить від GtkLabel) є дещо складним. Віджет GtkLabel не має власного об'єкта вікна з міркувань продуктивності, а колір фону можуть мати лише об'єкти вікна. Рішенням є розміщення Label у контейнері EventBox, який має вікно, але є невидимим — див. файл coloredlabel.ui.

Я визначив віджет `hal_spinbutton` у Glade та встановив властивість `value` за замовчування

Це пов'язано з помилкою в старій версії Gtk, що поширювалася з Ubuntu 8.04 та 10.04, і, ймовірно, стосується всіх віджетів, що використовують регулювання. Обхідний шлях, згаданий, наприклад, в http://osdir.com/ml/gtk-app-devel-list/2010-04/msg00129.html, не встановлю надійно значення HAL pin, краще встановити його явно в обробнику сигналів `on_realize` під час створення віджета. Дивіться приклад у `configs/apps/gladevcp/by-widget/spinbutton.{ui,p`

12.3.10 Усунення несправностей

- Переконайтеся, що у вас встановлено розробницьку версію LinuxCNC. Вам більше не потрібен файл axisrc, про це згадувалося на старій вікі-сторінці GladeVCP.
- Запустіть GladeVCP або AXIS з терміналу. Якщо ви отримаєте помилки Python, перевірте, чи все ще існує файл `/usr/lib/python2.6/dist-packages/hal.so`, окрім нового `/usr/lib/python2.6/di` (зверніть увагу на підкреслення); якщо так, видаліть файл `hal.so`. Він був замінений файлом hal.py в тому ж каталозі і заважає механізму імпорту.
- Якщо ви використовуєте функцію запуску на місці, виконайте команду «make clean», щоб видалити будь-який випадково залишковий файл hal.so, а потім виконайте команду «make».
- Якщо ви використовуєте віджети *HAL_table* або *HAL_HBox*, майте на увазі, що з ними пов'язаний HAL-пін, який за замовчуванням вимкнено. Цей пін контролює, чи активні дочірні елементи цих контейнерів.

12.3.11 Примітка щодо впровадження: Обробка ключів в AXIS

Ми вважаємо, що обробка ключів працює нормально, але оскільки це новий код, ми розповідаємо про нього, щоб ви могли бути уважні до проблем; будь ласка, повідомляйте нам про помилки або дивну поведінку. Ось історія:

AXIS використовує набір віджетів TkInter. Додатки GladeVCP використовують віджети Gtk і працюють в окремому контексті процесу. Вони підключаються до AXIS за допомогою протоколу Xembed. Це дозволяє дочірньому додатку, такому як GladeVCP, правильно вписуватися у вікно батьківського додатка і, теоретично, мати інтегровану обробку подій.

Однак це передбачає, що як батьківська, так і дочірня програма належним чином підтримують протокол Xembed, що робить Gtk, але не робить TkInter. Наслідком цього є те, що певні клавіші не будуть належним чином передаватися з панелі GladeVCP до AXIS за всіх обставин. Однією з таких ситуацій був випадок, коли фокус був на віджеті Entry або SpinButton: у цьому випадку,

наприклад, клавіша Escape не передавалася б до AXIS і не спричиняла б переривання, як це мало б бути, що могло б мати катастрофічні наслідки.

Тому ключові події в GladeVCP обробляються явно і вибірково пересилаються до AXIS, щоб уникнути виникнення таких ситуацій. Детальніше див. функцію `keyboard_forward()` у файлі `lib/python/gladev`

12.3.12 Додавання користувацьких віджетів

У вікі LinuxCNC є інформація про додавання користувацьких віджетів до GladeVCP. Посилання: Користувацькі віджети GladeVCP

12.3.13 Допоміжні програми GladeVCP

Підтримка надається для самостійно встановлених додатків GladeVCP, які відповідають розміщенню системних каталогів, визначеному елементами LINUXCNC_AUX_GLADEVCP та LINUXCNC_AUX_EXAM про які повідомляє скрипт linuxcnc_var:

```
$ linuxcnc_var LINUXCNC_AUX_GLADEVCP
/usr/share/linuxcnc/aux_gladevcp
$ linuxcnc_var LINUXCNC_AUX_EXAMPLES
/usr/share/linuxcnc/aux_examples
```

Системний каталог, визначений LINUXCNC_AUX_GLADEVCP (/usr/share/linuxcnc/aux_gladevcp), визначає розташування файлів Python, сумісних з GladeVCP, та відповідних підкаталогів. Файл Python імпортується під час запуску GladeVCP і стає доступним для наступних додатків GladeVCP, включаючи вбудоване використання в підтримуючих графічних інтерфейсах користувача.

Системний каталог, визначений LINUXCNC_AUX_EXAMPLES (/usr/share/linuxcnc/aux_examples), визначає розташування підкаталогів з прикладами конфігурацій, що використовуються для допоміжни програм. Дивіться розділ getting-started/running-linuxcnc для «Додавання елементів вибору конфігурац

Для тестування можна вказати специфікацію часу виконання допоміжних програм за допомогою експортованої змінної середовища: GLADEVCP_EXTRAS. Ця змінна повинна бути списком шляхів до одного або декількох каталогів конфігурації, розділених символом (:). Зазвичай ця змінна встановлюється в оболонці, що запускає `linuxcnc`, або в скрипті запуску користувача `~/.profile`. Приклад:

```
export GLADEVCP_EXTRAS=~/mygladevcp:/opt/othergladevcp
```

Файли, знайдені в каталогах, вказаних за допомогою змінної середовища GLADEVCP_EXTRAS, замінюють файли з однаковою назвою в підкаталогах системного каталогу, вказаного за допомогою LINUXNC_AUX_GLADEVCP (наприклад, /usr/share/linuxcnc/aux_gladevcp). Це положення дозволяє розробнику тестувати програму, експортуючи GLADEVCP_EXTRAS для визначення приватного каталогу програми без видалення системного каталогу програми. Повідомлення про відхилені дублікати виводяться на stdout.

Note
Для підтримки допоміжних програм GladeVCP потрібен модуль Python під назвою «importlib». Цей модуль може бути недоступний у старіших інсталяціях, таких як Ubuntu-Lucid.

12.4 Модулі бібліотеки GladeVCP

Бібліотеки — це попередньо створені модулі Python, які надають GladeVCP додаткові функції. Таким чином, ви можете вибрати потрібні функції, але вам не доведеться самостійно створювати поширені.

12.4.1 Інформація

Info — це бібліотека для збору та фільтрації даних з INI-файлу.

Доступні дані та значення за замовчуванням:

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = int(self.INI.find("KINS","JOINTS")or 0)
AVAILABLE_AXES = ['X','Y','Z']
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:'X',1:'Y',2:'Z'}
GET_JOG_FROM_NAME = {'X':0,'Y':1,'Z':2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.find(section, "TYPE") or "LINEAR"
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 b''ob''b''дb''b''иb''b''нb''b''иb''b''цb''b''ьb'' b''зb''b' ↩
    'ab'' b''xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb''
MIN_LINEAR_JOG_VEL = 60 b''ob''b''дb''b''иb''b''нb''b''иb''b''цb''b''ьb'' b''зb''b''ab'' b' ↩
    'xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb''
MAX_LINEAR_JOG_VEL = 300 b''ob''b''дb''b''иb''b''нb''b''иb''b''цb''b''ьb'' b''зb''b''ab'' b ↩
    ''xb''b''вb''b''иb''b''лb''b''иb''b''нb''b''уb''

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = int(self.INI.find("TRAJ", "SPINDLES") or 1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
```

```
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY

# b''ib''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''яb'' b''пb''b''pb''b' ↩
    'ob'' b''дb''b''ib''b''ab''b''лb''b''ob''b''гb''b''ob''b''вb''b''eb'' b''вb''b''ib''b' ↩
    'кb''b''нb''b''ob'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b' ↩
    'нb''b''нb''b''яb'' b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''yb''b''вb''b''ab''b' ↩
    'чb''b''ab''
USRMESS_BOLDTEXT = self.INI.findall("DISPLAY", "MESSAGE_BOLDTEXT")
USRMESS_TEXT = self.INI.findall("DISPLAY", "MESSAGE_TEXT")
USRMESS_TYPE = self.INI.findall("DISPLAY", "MESSAGE_TYPE")
USRMESS_PINNAME = self.INI.findall("DISPLAY", "MESSAGE_PINNAME")
USRMESS_DETAILS = self.INI.findall("DISPLAY", "MESSAGE_DETAILS")
USRMESS_ICON = self.INI.findall("DISPLAY", "MESSAGE_ICON")
ZIPPED_USRMESS =

self.GLADEVCP = (self.INI.find("DISPLAY", "GLADEVCP")) or None

# b''вb''b''6b''b''yb''b''дb''b''ob''b''вb''b''ab''b''нb''b''ab'' b''ib''b''нb''b''фb''b' ↩
    'ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''яb'' b''пb''b''pb''b''ob'' b''пb''b''pb''b' ↩
    'ob''b''гb''b''pb''b''ab''b''мb''b''yb''
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)
```

Є деякі «допоміжні функції» – здебільшого використовуються для підтримки віджетів

```
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)
```

Щоб імпортувати ці модулі, додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

b''зb'' b''ib''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''їb'' b''пb''b' ↩
    'pb''b''ob'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' gladevcp.core
```

Щоб створити екземпляр модуля для використання у файлі обробника, додайте цей код Python до розділу створення екземплярів:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''»  ↩
    **** #
########################################
```

```
INFO = Info()
```

Для доступу до даних INFO використовуйте цей загальний синтаксис:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print('Metric based')
```

12.4.2 Дія

Ця бібліотека використовується для керування контролером руху LinuxCNC. Вона намагається приховати випадкові деталі та додати зручні методи для розробників.

Щоб імпортувати ці модулі, додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

b''Дb''b''ib''b''яb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' b''зb'' gladevcp. ↩
    core
```

Щоб створити екземпляр модуля для його використання, додайте цей код Python до розділу створення екземплярів:

```
##########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''»  ↩
    **** #
##########################################

ACTION = Action()
```

Щоб отримати доступ до команд дій, використовуйте загальний синтаксис, наприклад:

```
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(b''cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''ab'')

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(b''нb''b''ob''b''мb''b''eb''b''pb'' b''cb''b''yb''b''гb''b''лb''b' ↩
    'ob''b''6b''b''ab'')

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)
```

Є деякі «допоміжні функції» – здебільшого використовуються для підтримки цієї бібліотеки

```
get_jog_info (num)
jnum_check(num)
ensure_mode(modes)
open_filter_program(filename, filter)
```

12.5 QtVCP

QtVCP — це **інфраструктура для створення користувацьких екранів CNC або панелей керування для LinuxCNC**.

Він відображає файл *.ui, створений за допомогою редактора екранів Qt Designer, та поєднує його з _програмуванням на Python* для створення екрана графічного інтерфейсу для роботи з верстатом з ЧПК.

QtVCP повністю *налаштовується*: ви можете додавати різні кнопки, світлодіоди стану тощо, або додавати код Python для ще дрібнішого налаштування.

12.5.1 Вітрина

Кілька прикладів екранів та віртуальних панелей керування, вбудованих у QtVCP:

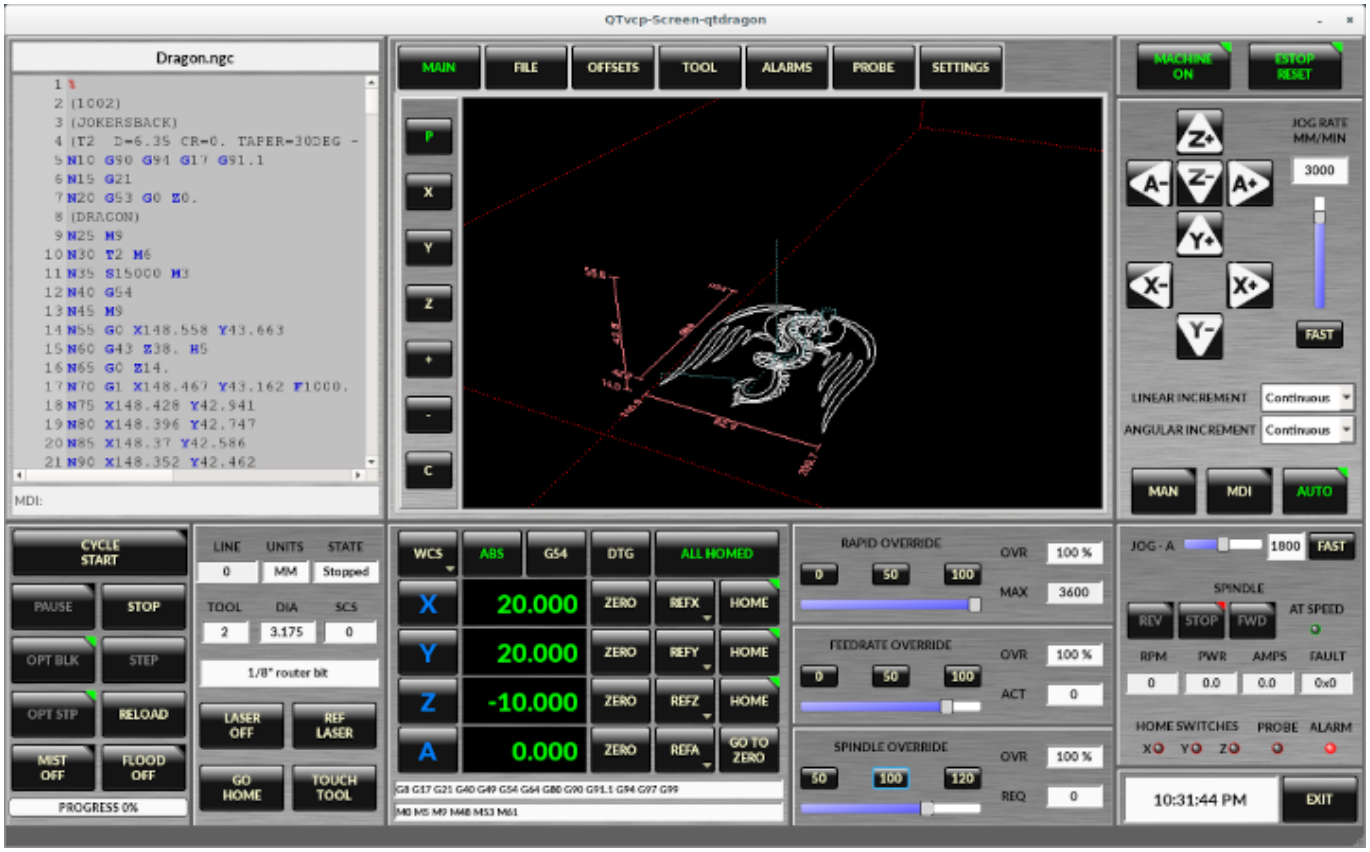


Figure 12.55: QtDragon - Зразок для 3/4 осі



Figure 12.56: QtDefault - 3-осьовий приклад

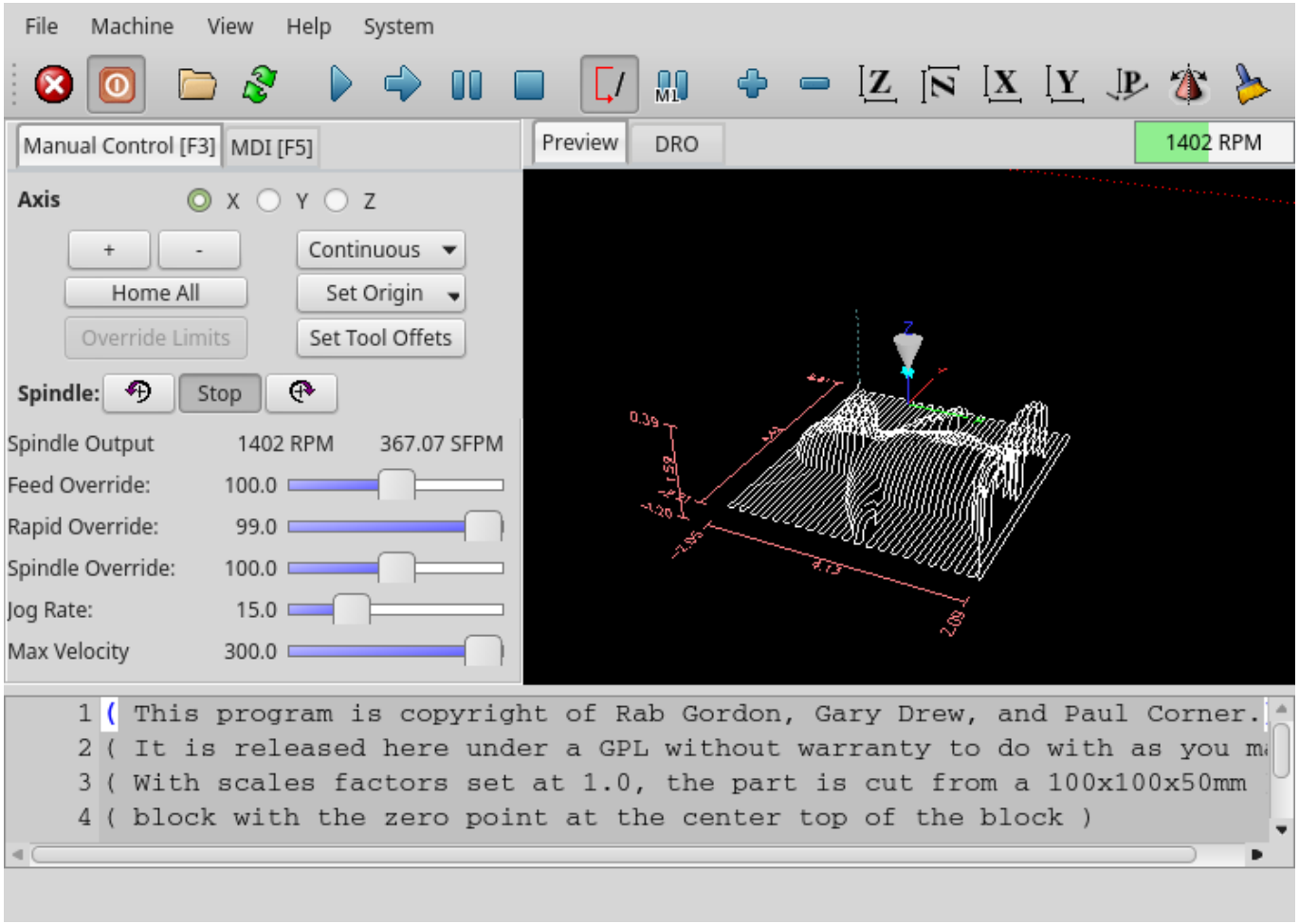


Figure 12.57: QtAxis - Приклад самоналаштування осі

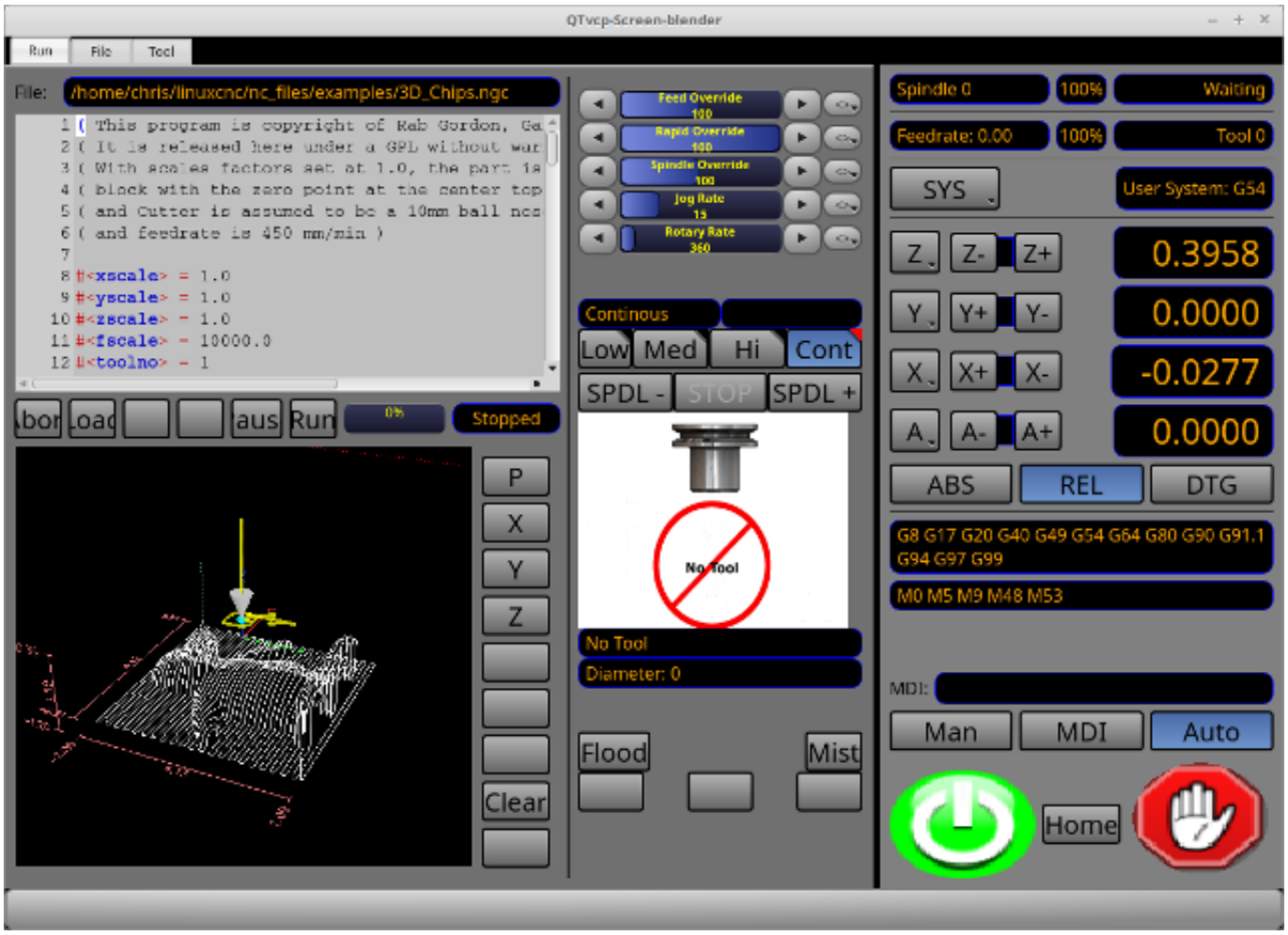


Figure 12.58: Blender - 4-осьовий зразок

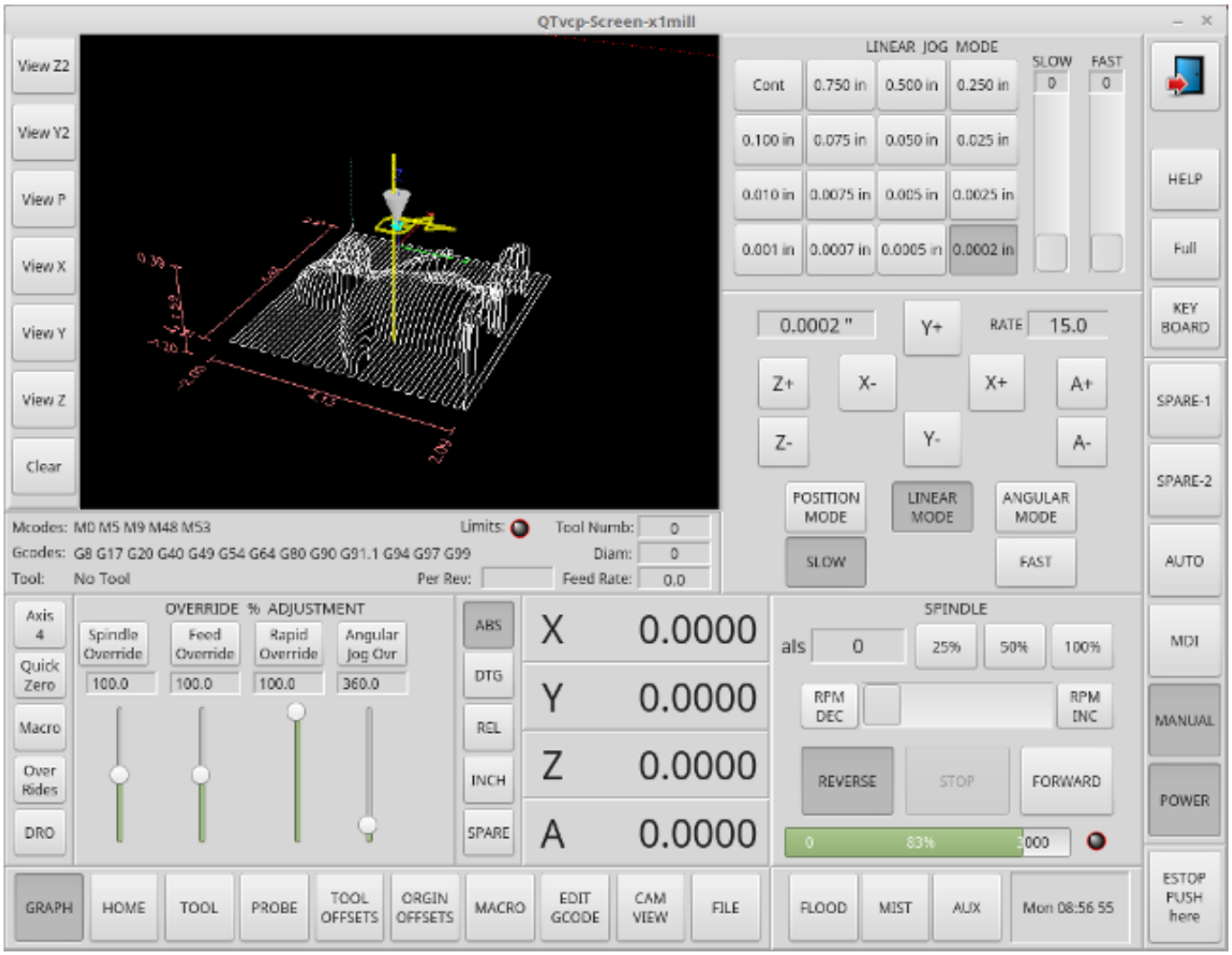


Figure 12.59: X1mill - 4-осьовий зразок



Figure 12.60: cam_align - Вирівнювання камери VCP

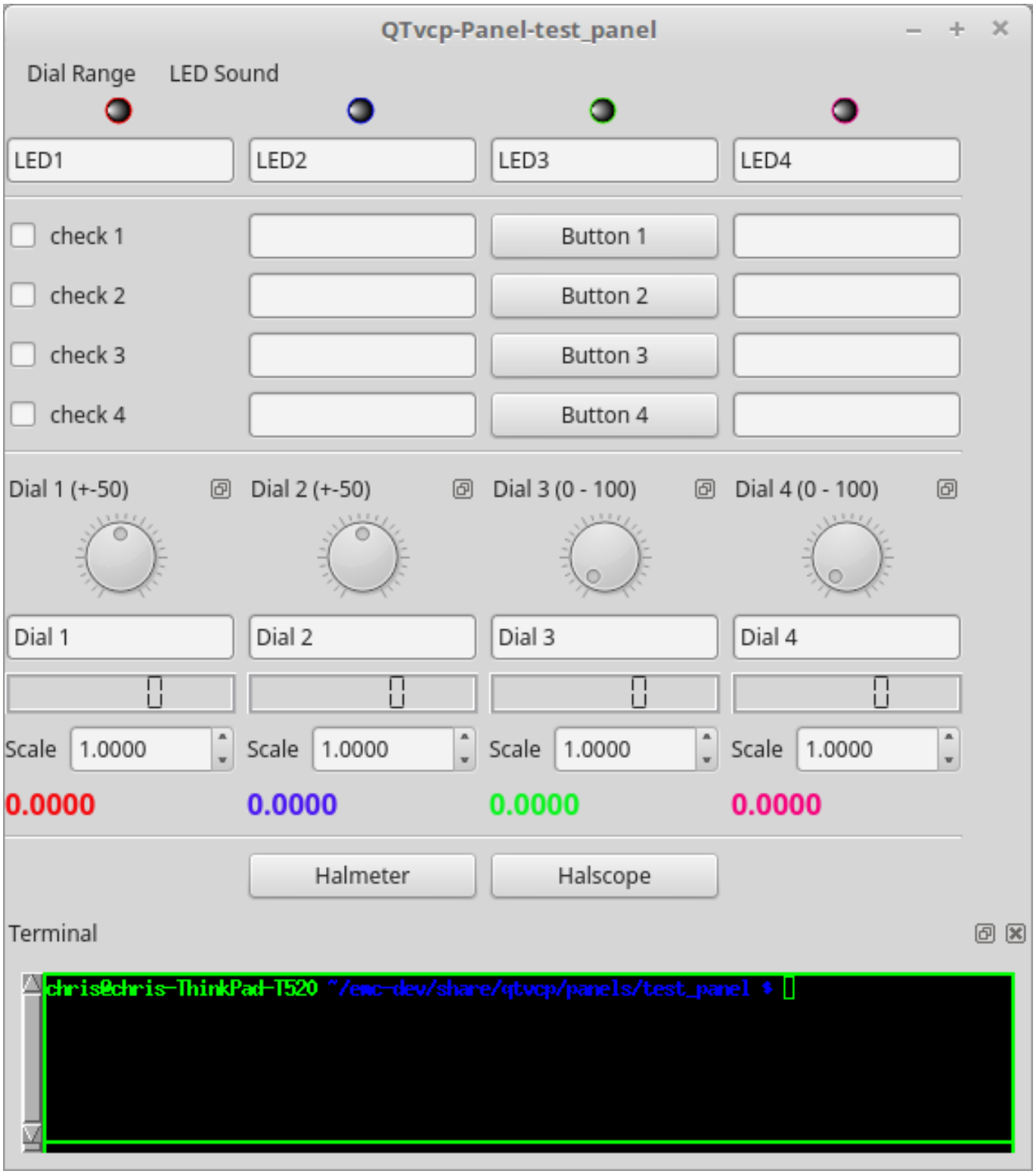


Figure 12.61: test_panel - Тестова панель VCP

12.5.2 Огляд

Для додавання налаштувань використовуються два файли, окремо або в поєднанні:

- **Файл інтерфейсу користувача**, який є *XML* файлом, створеним за допомогою графічного редактора *Qt Designer*.
- Файл **обробника**, який є текстовим файлом коду *Python*.

Зазвичай QtVCP використовує стандартний файл інтерфейсу користувача та обробника, але ви можете вказати QtVCP використовувати «локальні» файли інтерфейсу користувача та обробника. «Локальний» файл — це файл, який знаходиться в папці *configuration* і визначає решту вимог до комп'ютера.

Додавання власної панелі праворуч або вкладки не обмежується лише додаванням власної панелі, оскільки QtVCP використовує *Qt Designer* (редактор) та *PyQt5* (набір інструментів для віджетів).

QtVCP має деякі додаткові **спеціальні віджети та дії LinuxCNC**.
Існують спеціальні віджети для з'єднання віджетів сторонніх розробників з контактами HAL.
Можна створювати відповіді віджетів, підключаючи сигнали до коду Python у файлі обробника.

12.5.2.1 Віджети QtVCP

QtVCP використовує віджети **інструментарію PyQt5** для інтеграції з LinuxCNC.

Віджет — це *загальна назва для об'єктів інтерфейсу користувача*, таких як кнопки та мітки в PyQt5.

Ви можете вільно використовувати будь-які доступні **віджети за замовчуванням** у редакторі Qt Designer.

Також існують **спеціальні віджети**, створені для LinuxCNC, які спрощують інтеграцію.
Вони розділені на три заголовки в лівій частині редактора:

- Один призначений для *віджетів лише HAL*;
- Один призначений для _віджетів керування _CNC;
- Один призначений для *віджетів діалогу*.

Ви можете вільно змішувати їх будь-яким чином на своїй панелі.

Дуже важливим віджетом для управління CNC є віджет **`ScreenOptions`**: він не додає нічого візуально на екран, але дозволяє вибирати важливі деталі, замість того, щоб кодувати їх у файлі обробника.

12.5.2.2 Налаштування INI

Якщо ви використовуєте QtVCP для створення екрана керування рухом CNC (а не панелі на основі HAL), у файлі INI, в розділі `[DISPLAY]`, додайте рядок із таким шаблоном:

```
DISPLAY = qtvcp <options> <screen_name>
```

> **Note**
> Усі `<options>` мають стояти перед `<screen_name>`.

Опції

- `-d` Налагодження триває.
- `-i` Увімкнути вивід інформації.

- `-v` Увімкнути детальний вивід налагодження.
- `-q` Увімкнути лише вивід налагодження помилок.
- `-a` Встановити вікно завжди зверху.
- `-c NAME` Назва компонента HAL. За замовчуванням використовується назва файлу інтерфейсу користувача.
- `-g GEOMETRY` Встановити геометрію WIDTHxHEIGHT+XOFFSET+YOFFSET. Значення вказані в пікселях, XOFFSET/YOFFSET відлічуються від верхнього лівого кута екрану. Використовуйте -g WIDTHxHEIGHT для встановлення лише розміру або -g +XOFFSET+YOFFSET для встановлення лише положення. Приклад: `-g 200x400+0+100`
- `-H FILE` Виконайте оператори hal з ФАЙЛУ за допомогою halcmd після того, як компонент буде налаштовано та готовий до роботи.
- `-m` Максимізувати вікно.
- `-f` Повноекранне вікно.
- `-t THEME` За замовчуванням використовується системна тема
- `-x XID` Вбудувати у вікно X11, яке не підтримує вбудовування.
- `--push_xid` Надіслати ідентифікатор вікна X11 QtVCP на стандартний вивід; для вбудовування.
- `-u USERMOD` Шлях до файлу обробника-замінника.
- `-o USEROPTS` Передати рядок до файлу обробника QtVCP у змінній списку `self.w.USEROPTIONS_`. Може бути декілька -o.

<screen_name> `<screen_name>` – це *базова назва файлів .ui та _handler.py*. Якщо `<screen_name>` відсутнє, буде завантажено екран за замовчуванням.

QtVCP припускає, що файл інтерфейсу користувача та файл обробника використовують **однакове базове ім'я**. QtVCP спочатку шукатиме файли у каталозі конфігурації LinuxCNC, який було запущено, а потім у системній папці шкір, що містить стандартні екрани.

Час циклу

```
[DISPLAY]
CYCLE_TIME = 100
GRAPHICS_CYCLE_TIME = 100
HALPIN_CYCLE = 100
```

Налаштовує швидкість реагування оновлень графічного інтерфейсу в мілісекундах. За замовчуванням 100, доступний діапазон 50–200.

Віджети, графіку та оновлення PIN-кодів HAL можна налаштувати окремо.

Якщо час оновлення встановлено неправильно, екран може перестати реагувати або стати дуже тріскатим.

12.5.2.3 Файл інтерфейсу Qt Designer

Файл Qt Designer — це текстовий файл, організований у стандарті *XML*, який описує **макет та віджети** екрана.

PyQt5 використовує цей файл для створення дисплея та реагування на ці віджети.

Редактор Qt Designer значно спрощує створення та редагування цього файлу.

12.5.2.4 Файли обробників

Файл-обробник — це файл, що містить код *Python*, який **додається до стандартних процедур QtVCP**.

Файл-обробник дозволяє *змінювати значення за замовчуванням* або *додавати логіку* до екрана QtVCP без необхідності змінювати основний код QtVCP. Таким чином, ви можете мати **власну поведінку**.

Якщо файл обробника присутній, буде завантажено його. Дозволено **лише один файл**.

12.5.2.5 Модулі бібліотек

QtVCP, у вбудованому вигляді, виконує лише відображення екрана та реагує на віджети. Для отримання додаткових **готових налаштувань поведінки** доступні бібліотеки (знаходяться в `lib/python/qtvcp/lib` у файлі встановлення RIP LinuxCNC).

Бібліотеки — це попередньо створені *модулі Python*, які **додають функції** до QtVCP. Таким чином, ви можете вибрати потрібні функції, але вам не доведеться самостійно створювати поширені.

Такі бібліотеки включають:

- `audio_player`
- `aux_program_loader`
- `keybindings`
- `message`
- `preferences`
- `notify`
- `virtual_keyboard`
- `machine_log`

12.5.2.6 Теми

Теми – це спосіб змінити **зовнішній вигляд** віджетів на екрані.

Наприклад, *колір* або *розмір* кнопок і повзунків можна змінити за допомогою тем.

Тема Windows використовується за замовчуванням для екранів.
Тема System використовується за замовчуванням для панелей.

Щоб переглянути доступні теми, їх можна завантажити, виконавши таку команду в терміналі:

```
qtvcp -d -t <theme_name>
```

QtVCP також можна налаштувати за допомогою *Qt-таблиць стилів (QSS)* за допомогою CSS.

12.5.2.7 Локальні файли

Якщо такі є, то замість стандартних файлів інтерфейсу користувача будуть завантажені локальні файли UI/QSS/Python у папці конфігурації.

Локальні файли UI/QSS/Python дозволяють використовувати власні налаштовані дизайни, а не екрани за замовчуванням.

QtVCP шукатиме папку з назвою <screen_name> (у запущеній папці конфігурації, яка містить INI-файл).

У цій папці QtVCP завантажить будь-який з доступних наступних файлів:

- `<screen_name>.ui`,
- `<screen_name>_handler.py`, і
- `<screen_name>.qss`.

12.5.2.8 Зміна стокових екранів

Існує *три способи* налаштування екрана/панелі.

Таблиці стилів можна використовувати для **встановлення властивостей Qt**. Якщо віджет використовує властивості, то їх зазвичай можна змінювати за допомогою таблиць стилів.

Приклад віджета з відповідними налаштуваннями таблиці стилів.

```
State_LED #name_of_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
  }
```

Ми можемо змусити QtVCP завантажити підкласову версію стандартного файлу обробника. У цьому файлі ми можемо маніпулювати оригінальними функціями або додавати нові.
Підкласування означає, що наш файл обробника спочатку завантажує оригінальний файл обробника і додає до нього наш новий код - фактично патч змін.
Це корисно для зміни/додавання поведінки, зберігаючи при цьому стандартні оновлення обробника з репозиторіїв LinuxCNC.

Можливо, вам все одно доведеться скористатися діалоговим вікном копіювання обробника, щоб скопіювати оригінальний файл обробника та вирішити, як його виправити. Див. розділ «користувацьки файл обробника».

У папці config повинна бути папка для екранів з назвою «<CONFIG FOLDER>/qtvcp/screens/<SCREEN NAME>/».
Додайте туди файл патча з назвою <ORIGINAL SCREEN NAME>_handler.py,
наприклад, для QtDragon файл буде називатися «qtdragon_handler.py».

Ось приклад додавання штифтів повороту осі X на екран, такий як QtDragon:

```
import sys
import importlib
from qtvcp.core import Path, Qhal, Action
PATH = Path()
QHAL = Qhal()
ACTION = Action()
```

```
# b''ob''b''тb''b''pb''b''иb''b''мb''b''ab''b''тb''b''иb'' b''пb''b''ob''b''cb''b''иb''b' ↩
    'лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b''ob''b''pb''b''иb''b''гb''b''іb''b' ↩
    'нb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''фb''b''ab''b''йb''b''лb'' b''ob''b' ↩
    'бb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab'', b''щb''b''ob''b''бb'' b''мb''b' ↩
    'иb'' b''мb''b''ob''b''гb''b''лb''b''иb'' b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b' ↩
    'иb'' b''йb''b''ob''b''гb''b''ob'' b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb''b' ↩
    'ob''b''мb''
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

# b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''уb''b''тb''b''иb'' b''нb''b''ab''b''шb'' b' ↩
    'ob''b''бb'''b''єb''b''кb''b''тb''-b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''иb''b' ↩
    'кb'' b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb''b''уb'' b''дb''b''ob'' QtVCP
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb'' HandlerClass, b''яb''b''кb''b''иb''b' ↩
    'йb'' b''бb''b''уb''b''лb''b''ob'' b''іb''b''мb''b''пb''b''ob''b''pb''b''тb''b''ob''b' ↩
    'вb''b''ab''b''нb''b''ob'' b''вb''b''иb''b''щb''b''eb''
class UserHandlerClass(HandlerClass):
    # b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''пb''b''ob''b''вb''b''іb''b''дb''b''ob'' ↩
        b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''тb''b''eb''b''pb''b''мb''b''іb''b' ↩
        'нb''b''ab''b''лb''b''уb'', b''щb''b''ob''b''бb'' b''мb''b''иb'' b''зb''b''нb''b' ↩
        'ab''b''лb''b''иb'', b''щb''b''ob'' b''цb''b''eb'' b''зb''b''ab''b''вb''b''ab''b' ↩
        'нb''b''тb''b''ab''b''жb''b''иb''b''лb''b''ob''b''cb''b''яb''
    print('\nCustom subclassed handler patch loaded.\n')

    def init_pins(self):
        # b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''тb''b''иb'' b''фb''b''уb''b' ↩
            'нb''b''кb''b''цb''b''іb''b''юb'' init_pins b''ob''b''pb''b''иb''b''гb''b''іb''b ↩
            ''нb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''ob''b''бb''b''pb''b' ↩
            'ob''b''бb''b''нb''b''иb''b''кb''b''ab''
        super().init_pins()

        # b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''шb''b''тb''b''иb''b''фb''b''тb''b' ↩
            'иb'' b''пb''b''ob''b''вb''b''ob''b''pb''b''ob''b''тb''b''нb''b''ob''b''гb''b' ↩
            'ob'' b''мb''b''eb''b''хb''b''ab''b''нb''b''іb''b''зb''b''мb''b''уb'' b''пb''b' ↩
            'ob'' b''ob''b''cb''b''іb'' X
        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))
```

Інший файл Python може бути використаний для **додавання команд** на екран після аналізу файлу обробника. Це може бути корисно для внесення незначних змін, при цьому зберігаючи стандартні оновлення обробника з репозиторіїв LinuxCNC.

Note
Патчування обробників — це кращий спосіб додавання змін, патчування екземплярів — це вуду чорної магії, і це тут з міркувань застарілих кодів.

У файлі *INI* під заголовком `[DISPLAY]` додайте **`USER_COMMAND_FILE = _PATH_`**
PATH може бути будь-яким дійсним шляхом. Можна використовувати ~ для домашнього каталогу або `WORKINGFOLDER` чи `CONFIGFOLDER`, щоб позначити ці каталоги в QtVCP, наприклад:

```
[DISPLAY]
USER_COMMAND_FILE = CONFIGFOLDER/<screen_name_added_commands>
```

Якщо запис не знайдено в *INI*, QtVCP шукатиме в **шляху за замовчуванням**. Шлях за замовчуванням знаходиться в каталозі конфігурації як прихований файл із використанням базового імені екрана та rc, наприклад, **CONFIGURATION DIRECTORY/.<screen_name>rc**.

Цей файл буде зчитано та виконано як код Python у **контексті файлу обробника**.

Можна посилатися тільки на локальні функції та локальні атрибути.
На глобальні бібліотеки, визначені у файлі обробника екрана, можна посилатися шляхом імпортування файлу обробника.
Зазвичай вони позначаються великими літерами без попереднього self.
self посилається на функції класу вікна.
self.w зазвичай посилається на віджети.

Те, що можна використовувати, може відрізнятися залежно від екрана та циклу розробки.

Простий приклад Зверніться до головного вікна, щоб змінити заголовок (не відображатиметься, якщо для зміни заголовка використовуються записи INI).

```
self.w.setWindowTitle('b''Mb''b''ib''b''йb'' b''тb''b''eb''b''cb''b''тb'' b''нb''b''ab'' b' ↩
    'тb''b''иb''b''тb''b''уb''b''лb''')
```

Приклад розширеного виправлення екземпляра Це може працювати з файлом обробника екрана QtDragon.
Тут ми показуємо, як додавати нові функції та перевизначати існуючі.

```
# b''Пb''b''ob''b''тb''b''pb''b''ib''b''бb''b''нb''b''ob'' b''cb''b''тb''b''вb''b''ob''b' ↩
    'pb''b''иb''b''тb''b''иb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
    'pb'' b''пb''b''ab''b''тb''b''чb''b''ab''.
# b''пb''b''ob''b''cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'': https://ruivieira.dev/ ↩
    python-monkey-patching-for-readability.html
import types

# b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''уb''b''йb''b''тb''b''eb'' b''фb''b''ab''b' ↩
    'йb''b''лb'' b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab'', b''щb''b' ↩
    'ob''b''бb'' b''ob''b''тb''b''pb''b''иb''b''мb''b''ab''b''тb''b''иb'' b''пb''b''ob''b' ↩
    'cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b''йb''b''ob''b''гb''b' ↩
    'ob'' b''бb''b''ib''b''бb''b''лb''b''ib''b''ob''b''тb''b''eb''b''кb''b''иb''.
# b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''уb''b''вb''b' ↩
    'ab''b''тb''b''иb'' <screenname>_handler
b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''уb''b''вb''b''ab''b''тb''b''иb''  ↩
    qtdragon_handler b''яb''b''кb'' hdlr

# b''Цb''b''eb'' b''нb''b''ab''b''cb''b''пb''b''pb''b''ab''b''вb''b''дb''b''ib'' b''нb''b' ↩
    'eb''b''ob''b''бb''b''мb''b''eb''b''жb''b''eb''b''нb''b''ab'' b''фb''b''уb''b''нb''b' ↩
    'кb''b''цb''b''ib''b''яb'' b''зb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b' ↩
    'pb''b''ob''b''мb'' «obj».
# b''Bb''b''иb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''тb''b''eb'' b' ↩
    'цb''b''юb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''юb'' b''бb''b''eb''b''зb'' b' ↩
    'зb''b''вb''b''иb''b''чb''b''ab''b''йb''b''нb''b''ob''b''гb''b''ob'' b''пb''b''ob''b' ↩
    'пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob''b''гb''b''ob'' «self.».
# b''Цb''b''eb'' b''тb''b''ob''b''мb''b''уb'', b''щb''b''ob'' b''мb''b''иb'' b''нb''b''eb'' ↩
     b''бb''b''уb''b''дb''b''eb''b''мb''b''ob'' b''вb''b''cb''b''тb''b''ab''b''вb''b''лb''b' ↩
    'яb''b''тb''b''иb'' b''їb''b''їb'' b''вb'' b''ob''b''pb''b''иb''b''гb''b''ib''b''нb''b' ↩
    'ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b' ↩
    'лb''b''яb''b''pb'' b''кb''b''лb''b''ab''b''cb''b''уb'' b''ob''b''бb''b''pb''b''ob''b' ↩
    'бb''b''нb''b''иb''b''кb''b''ab''.
# b''Bb''b''ob''b''нb''b''ab'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''лb''b' ↩
    'иb''b''кb''b''ab''b''тb''b''иb''b''cb''b''яb'' b''тb''b''ib''b''лb''b''ьb''b''кb''b' ↩
    'иb'' b''зb'' b''кb''b''ob''b''дb''b''уb'' b''вb'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b ↩
    ''фb''b''ab''b''йb''b''лb''b''ib''.
```

```
def test_function(obj):
    print(dir(obj))

# b''Цb''b''eb'' b''нb''b''ob''b''вb''b''ab'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b' ↩
    'яb'', b''яb''b''кb''b''уb'' b''мb''b''иb'' b''дb''b''ob''b''дb''b''ab''b''мb''b''ob'' b ↩
    ''дb''b''ob'' b''іb''b''сb''b''нb''b''уb''b''юb''b''чb''b''ob''b''гb''b''ob'' b''eb''b' ↩
    'кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''уb'' b''кb''b''лb''b''ab''b' ↩
    'сb''b''уb'' b''ob''b''бb''b''рb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab''.
# b''Зb''b''вb''b''eb''b''рb''b''нb''b''іb''b''тb''b''ьb'' b''уb''b''вb''b''ab''b''гb''b' ↩
    'уb'', b''щb''b''ob'' b''вb''b''ob''b''нb''b''ab'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
    'кb''b''ab''b''єb'' b''нb''b''eb''b''ob''b''бb''b''мb''b''eb''b''жb''b''eb''b''нb''b' ↩
    'уb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''юb'' b''зb'' 'self' b''яb''b''кb'' b' ↩
    'пb''b''ab''b''рb''b''ab''b''мb''b''eb''b''тb''b''рb''b''ob''b''мb''. 'self' b''єb'' b' ↩
    'єb''b''дb''b''иb''b''нb''b''иb''b''мb'' b''дb''b''ob''b''сb''b''тb''b''уb''b''пb''b' ↩
    'нb''b''иb''b''мb'' b''гb''b''лb''b''ob''b''бb''b''ab''b''лb''b''ьb''b''нb''b''иb''b' ↩
    'мb'' b''пb''b''ob''b''сb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb''b''мb''.
# b''Bb''b''ob''b''нb''b''ob'' b''пb''b''ob''b''сb''b''иb''b''лb''b''ab''b''єb''b''тb''b' ↩
    'ьb''b''сb''b''яb'' b''нb''b''ab'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b' ↩
    'яb''b''рb'' b''вb''b''іb''b''кb''b''нb''b''ab''.
def on_keycall_F10(self,event,state,shift,cntrl):
    if state:
        print ('F10')
        test_function(self)

# b''Цb''b''eb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b''иb''b''кb''b''ob''b''рb''b''иb''b' ↩
    'сb''b''тb''b''ob''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb''b''сb''b''яb'' b''дb''b' ↩
    'лb''b''яb'' b''зb''b''ab''b''мb''b''іb''b''нb''b''иb'' b''іb''b''сb''b''нb''b''уb''b' ↩
    'юb''b''чb''b''ob''b''їb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''їb'' b''вb'' b' ↩
    'іb''b''сb''b''нb''b''уb''b''юb''b''чb''b''ob''b''мb''b''уb'' b''eb''b''кb''b''зb''b' ↩
    'eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''іb'' b''кb''b''лb''b''ab''b''сb''b''уb'' b' ↩
    'ob''b''бb''b''рb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab''.
# b''Зb''b''вb''b''eb''b''рb''b''нb''b''іb''b''тb''b''ьb'' b''уb''b''вb''b''ab''b''гb''b' ↩
    'уb'', b''щb''b''ob'' b''мb''b''иb'' b''тb''b''ab''b''кb''b''ob''b''жb'' b''вb''b''иb''b ↩
    ''кb''b''лb''b''иb''b''кb''b''ab''b''єb''b''мb''b''ob'' b''кb''b''ob''b''пb''b''іb''b' ↩
    'юb'' b''ob''b''рb''b''иb''b''гb''b''іb''b''нb''b''ab''b''лb''b''ьb''b''нb''b''ob''b' ↩
    'їb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''їb''.
# b''Цb''b''eb'' b''пb''b''ob''b''кb''b''ab''b''зb''b''уb''b''єb'', b''яb''b''кb'' b''рb''b ↩
    ''ob''b''зb''b''шb''b''иb''b''рb''b''иb''b''тb''b''иb'' b''іb''b''сb''b''нb''b''уb''b' ↩
    'юb''b''чb''b''уb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''юb'' b''дb''b''лb''b' ↩
    'яb'' b''вb''b''иb''b''кb''b''ob''b''нb''b''ab''b''нb''b''нb''b''яb'' b''дb''b''ob''b' ↩
    'дb''b''ab''b''тb''b''кb''b''ob''b''вb''b''иb''b''хb'' b''фb''b''уb''b''нb''b''кb''b' ↩
    'цb''b''іb''b''йb''.
def on_keycall_F11(self,event,state,shift,cntrl):
    if state:
        self.on_keycall_F11_super(event,state,shift,cntrl)
        print ('Hello')

# b''Mb''b''иb'' b''пb''b''ob''b''сb''b''иb''b''лb''b''ab''b''єb''b''мb''b''ob''b''сb''b' ↩
    'яb'' b''нb''b''ab'' b''бb''b''іb''b''бb''b''лb''b''іb''b''ob''b''тb''b''eb''b''кb''b' ↩
    'уb'' KEYBIND, b''яb''b''кb''b''ab'' b''бb''b''уb''b''лb''b''ab'' b''іb''b''нb''b''сb''b ↩
    ''тb''b''ab''b''нb''b''цb''b''іb''b''йb''b''ob''b''вb''b''ab''b''нb''b''ab'' b''вb'' b' ↩
    'ob''b''рb''b''иb''b''гb''b''іb''b''нb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b' ↩
    'уb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''іb'' b''кb''b' ↩
    'лb''b''ab''b''сb''b''уb'' b''ob''b''бb''b''рb''b''ob''b''бb''b''нb''b''иb''b''кb''b' ↩
    'ab'',
# b''дb''b''ob''b''дb''b''ab''b''вb''b''шb''b''иb'' b''дb''b''ob'' b''нb''b''eb''b''їb'' ' ↩
   hdlr.' (b''зb'' imp).
# b''Цb''b''яb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''яb'' b''пb''b''ob''b''вb''b' ↩
    'іb''b''дb''b''ob''b''мb''b''лb''b''яb''b''єb'' KEYBIND b''вb''b''иb''b''кb''b''лb''b' ↩
    'иb''b''кb''b''ab''b''тb''b''иb'' 'on_keycall_F10' b''пb''b''рb''b''иb'' b''нb''b''ab''b ↩
    ''тb''b''иb''b''сb''b''кb''b''ab''b''нb''b''нb''b''іb'' F10.
hdlr.KEYBIND.add_call('Key_F10','on_keycall_F10')
```

```
# b''Tb''b''yb''b''tb'' b''mb''b''ib'' b''vb''b''ib''b''pb''b''rb''b''ab''b''vb''b''lb''b' ↩
    'yb''b''єb''b''mb''b''ob'' b''ob''b''rb''b''ib''b''gb''b''ib''b''nb''b''ab''b''lb''b' ↩
    'ьb''b''nb''b''ib''b''йb'' b''фb''b''ab''b''йb''b''lb'' b''ob''b''бb''b''rb''b''ob''b' ↩
    'бb''b''nb''b''ib''b''kb''b''ab'', b''щb''b''ob''b''бb'' b''дb''b''ob''b''дb''b''ab''b' ↩
    'tb''b''ib'' b''nb''b''ob''b''vb''b''yb'' b''фb''b''yb''b''nb''b''kb''b''цb''b''ib''b' ↩
    'юb'',
# b''яb''b''kb''b''ab'' b''vb''b''ib''b''kb''b''lb''b''ib''b''kb''b''ab''b''єb'' b''nb''b' ↩
    'ab''b''шb''b''yb'' b''nb''b''ob''b''vb''b''yb'' b''фb''b''yb''b''nb''b''kb''b''цb''b' ↩
    'ib''b''юb'' (b''зb'' b''tb''b''ab''b''kb''b''ob''b''юb'' b''жb'' b''nb''b''ab''b''зb''b ↩
    ''vb''b''ob''b''юb''), b''vb''b''ib''b''зb''b''nb''b''ab''b''чb''b''eb''b''nb''b''yb'' b ↩
    ''vb'' b''цb''b''ьb''b''ob''b''mb''b''yb'' b''фb''b''ab''b''йb''b''lb''b''ib''.
self.on_keycall_F10 = types.MethodType(on_keycall_F10, self)

# b''Tb''b''yb''b''tb'' b''mb''b''ib'' b''vb''b''ib''b''зb''b''nb''b''ab''b''чb''b''ab''b' ↩
    'єb''b''mb''b''ob'' b''kb''b''ob''b''пb''b''ib''b''юb'' b''ob''b''pb''b''ib''b''gb''b' ↩
    'ib''b''nb''b''ab''b''lb''b''ьb''b''nb''b''ob''b''їb'' b''фb''b''yb''b''nb''b''kb''b' ↩
    'цb''b''ib''b''їb'' 'on_keycall_F11',
# b''щb''b''ob''b''бb'' b''mb''b''ib'' b''mb''b''ob''b''gb''b''lb''b''ib'' b''vb''b''ib''b' ↩
    'kb''b''lb''b''ib''b''kb''b''ab''b''tb''b''ib'' b''їb''b''їb'' b''пb''b''ib''b''зb''b' ↩
    'nb''b''ib''b''шb''b''eb''. b''Mb''b''ib'' b''mb''b''ob''b''жb''b''eb''b''mb''b''ob'' b' ↩
    'vb''b''ib''b''kb''b''ob''b''pb''b''ib''b''cb''b''tb''b''ob''b''vb''b''yb''b''vb''b' ↩
    'ab''b''tb''b''ib'' b''бb''b''yb''b''дb''b''ьb''-b''яb''b''kb''b''yb'' b''дb''b''ib''b' ↩
    'йb''b''cb''b''nb''b''yb'', b''nb''b''eb''b''vb''b''ib''b''kb''b''ob''b''pb''b''ib''b' ↩
    'cb''b''tb''b''ab''b''nb''b''yb'' b''nb''b''ab''b''зb''b''vb''b''yb'' b''фb''b''yb''b' ↩
    'nb''b''kb''b''цb''b''ib''b''їb''.
# b''Hb''b''ab''b''mb'' b''пb''b''ob''b''tb''b''pb''b''ib''b''бb''b''nb''b''ob'' b''зb''b' ↩
    'pb''b''ob''b''бb''b''ib''b''tb''b''ib'' b''цb''b''eb'' b''пb''b''eb''b''pb''b''eb''b' ↩
    'дb'' b''пb''b''eb''b''pb''b''eb''b''vb''b''ib''b''зb''b''nb''b''ab''b''чb''b''eb''b' ↩
    'nb''b''nb''b''яb''b''mb'' b''ob''b''pb''b''ib''b''gb''b''ib''b''nb''b''ab''b''lb''b' ↩
    'ьb''b''nb''b''ob''b''їb'' b''фb''b''yb''b''nb''b''kb''b''цb''b''ib''b''їb''.
self.on_keycall_F11_super = self.on_keycall_F11

# b''Tb''b''yb''b''tb'' b''mb''b''ib'' b''cb''b''tb''b''vb''b''ob''b''pb''b''юb''b''єb''b' ↩
    'mb''b''ob'' b''пb''b''pb''b''ib''b''kb''b''lb''b''ab''b''дb'' b''пb''b''ab''b''tb''b' ↩
    'чb''b''yb'' b''дb''b''lb''b''яb'' b''ob''b''pb''b''ib''b''gb''b''ib''b''nb''b''ab''b' ↩
    'lb''b''ьb''b''nb''b''ob''b''gb''b''ob'' b''фb''b''ab''b''йb''b''lb''b''yb'' b''ob''b' ↩
    'бb''b''pb''b''ob''b''бb''b''nb''b''ib''b''kb''b''ab'', b''щb''b''ob''b''бb'' b''пb''b' ↩
    'eb''b''pb''b''eb''b''зb''b''ab''b''пb''b''ib''b''cb''b''ab''b''tb''b''ib'' b''ib''b' ↩
    'cb''b''nb''b''yb''b''юb''b''чb''b''yb'' b''фb''b''yb''b''nb''b''kb''b''цb''b''ib''b' ↩
    'юb''
# b''щb''b''ob''b''бb'' b''vb''b''kb''b''ab''b''зb''b''ab''b''tb''b''ib'' b''nb''b''ab'' b' ↩
    'nb''b''ab''b''шb''b''yb'' b''nb''b''ob''b''vb''b''yb'' b''фb''b''yb''b''nb''b''kb''b' ↩
    'цb''b''ib''b''юb'' (b''зb'' b''tb''b''ab''b''kb''b''ob''b''юb'' b''жb'' b''nb''b''ab''b ↩
    ''зb''b''vb''b''ob''b''юb''), b''vb''b''ib''b''зb''b''nb''b''ab''b''чb''b''eb''b''nb''b' ↩
    'yb'' b''vb'' b''цb''b''ьb''b''ob''b''mb''b''yb'' b''фb''b''ab''b''йb''b''lb''b''ib''.
self.on_keycall_F11 = types.MethodType(on_keycall_F11, self)


# b''дb''b''ob''b''дb''b''ab''b''tb''b''ib'' b''nb''b''ob''b''vb''b''ib''b''йb'' b''пb''b' ↩
    'ib''b''nb''-b''kb''b''ob''b''дb'' b''nb''b''ab'' b''eb''b''kb''b''pb''b''ab''b''nb'':

# b''зb''b''vb''b''ob''b''pb''b''ob''b''tb''b''nb''b''ib''b''йb'' b''vb''b''ib''b''kb''b' ↩
    'lb''b''ib''b''kb'' pin b''дb''b''lb''b''яb'' b''vb''b''ib''b''vb''b''eb''b''дb''b''eb'' ↩
   b''nb''b''nb''b''яb'' b''cb''b''tb''b''ab''b''nb''b''yb''
def new_pin_changed(data):
    print(data)

# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''lb''b''ьb''b''nb''b''ab'' b''фb''b''yb''b' ↩
    'nb''b''kb''b''цb''b''ib''b''яb'', b''яb''b''kb''b''ab'' b''vb''b''ib''b''kb''b''lb''b' ↩
    'ib''b''kb''b''ab''b''єb''b''tb''b''ьb''b''cb''b''яb'' b''дb''b''ob'' b''tb''b''ob''b' ↩
    'rb''b''ob'', b''яb''b''kb'' b''kb''b''ob''b''mb''b''пb''b''ob''b''nb''b''eb''b''nb''b' ↩
```

```
    ’тb’’ HAL b’’бb’’b’’уb’’b’’дb’’b’’еb’’ b’’гb’’b’’оb’’b’’тb’’b’’оb’’b’’вb’’b’’иb’’b’ ↩
    ’йb’’.
# b’’Tb’’b’’уb’’b’’тb’’ b’’мb’’b’’иb’’ b’’вb’’b’’иb’’b’’кb’’b’’оb’’b’’рb’’b’’иb’’b’’сb’’b’ ↩
    ’тb’’b’’аb’’b’’лb’’b’’иb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’’юb’’ b’’дb’’b’ ↩
    ’лb’’b’’яb’’ b’’дb’’b’’оb’’b’’дb’’b’’аb’’b’’вb’’b’’аb’’b’’нb’’b’’нb’’b’’яb’’ b’’бb’’b’ ↩
    ’іb’’b’’тb’’b’’оb’’b’’вb’’b’’оb’’b’’гb’’b’’оb’’ b’’вb’’b’’хb’’b’’іb’’b’’дb’’b’’нb’’b’ ↩
    ’оb’’b’’гb’’b’’оb’’ b’’вb’’b’’иb’’b’’вb’’b’’оb’’b’’дb’’b’’уb’’ b’’зb’’ 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)

# b’’Tb’’b’’уb’’b’’тb’’ b’’мb’’b’’иb’’ b’’вb’’b’’иb’’b’’пb’’b’’рb’’b’’аb’’b’’вb’’b’’лb’’b’ ↩
    ’яb’’b’’єb’’b’’мb’’b’’оb’’ b’’оb’’b’’рb’’b’’иb’’b’’гb’’b’’іb’’b’’нb’’b’’аb’’b’’лb’’b’ ↩
    ’ьb’’b’’нb’’b’’иb’’b’’йb’’ b’’фb’’b’’аb’’b’’йb’’b’’лb’’ b’’оb’’b’’бb’’b’’рb’’b’’оb’’b’ ↩
    ’бb’’b’’нb’’b’’иb’’b’’кb’’b’’аb’’, b’’щb’’b’’оb’’b’’бb’’ b’’дb’’b’’оb’’b’’дb’’b’’аb’’b’ ↩
    ’тb’’b’’иb’’ b’’нb’’b’’оb’’b’’вb’’b’’уb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’’іb’’b’ ↩
    ’юb’’,
# b’’яb’’b’’кb’’b’’аb’’ b’’вb’’b’’иb’’b’’кb’’b’’лb’’b’’иb’’b’’кb’’b’’аb’’b’’єb’’ b’’нb’’b’ ↩
    ’аb’’b’’шb’’b’’уb’’ b’’нb’’b’’оb’’b’’вb’’b’’уb’’ b’’фb’’b’’уb’’b’’нb’’b’’кb’’b’’цb’’b’ ↩
    ’іb’’b’’юb’’ (b’’зb’’ b’’тb’’b’’аb’’b’’кb’’b’’оb’’b’’юb’’ b’’жb’’ b’’нb’’b’’аb’’b’’зb’’b ↩
    ’’вb’’b’’оb’’b’’юb’’), b’’вb’’b’’иb’’b’’зb’’b’’нb’’b’’аb’’b’’чb’’b’’еb’’b’’нb’’b’’уb’’ b ↩
    ’’вb’’ b’’цb’’b’’ьb’’b’’оb’’b’’мb’’b’’уb’’ b’’фb’’b’’аb’’b’’йb’’b’’лb’’b’’іb’’.
self.after_override__ = types.MethodType(after_override__, self)
```

Якщо ви бажаєте **змінити стандартний екран** з повним контролем, *скопіюйте його інтерфейс користувача та файл обробника до папки конфігурації*.

Для цього є панель QtVCP:

- Відкрийте термінал і виконайте таку команду:

```
qtvcp copy
```

- Виберіть екран і папку призначення в діалоговому вікні
- Якщо ви бажаєте **назвати свій екран** інакше, ніж ім’я вбудованого екрана за замовчуванням, змініть *basename* у полі редагування.
- У папці config повинна бути папка; для екранів: з назвою «<CONFIG FOLDER>/qtvcp/screens/»; для панелей: з назвою «<CONFIG FOLDER>/qtvcp/panels/». Додайте папки, якщо їх немає, і скопіюйте туди свою папку/файли.
- Перевірте, щоб скопіювати всі файли
- Видаліть файли, які не хочете змінювати, щоб використовувати оригінальні файли.

12.5.3 Панелі VCP

QtVCP можна використовувати для створення панелей керування, які взаємодіють з **HAL**.

12.5.3.1 Вбудовані панелі

Доступно кілька **вбудованих панелей HAL**.

У терміналі введіть `qtvcp <return>`, щоб побачити список:

test_panel
Колекція корисних віджетів для тестування компонентів HAL, включаючи озвучування стану світлодіодів.

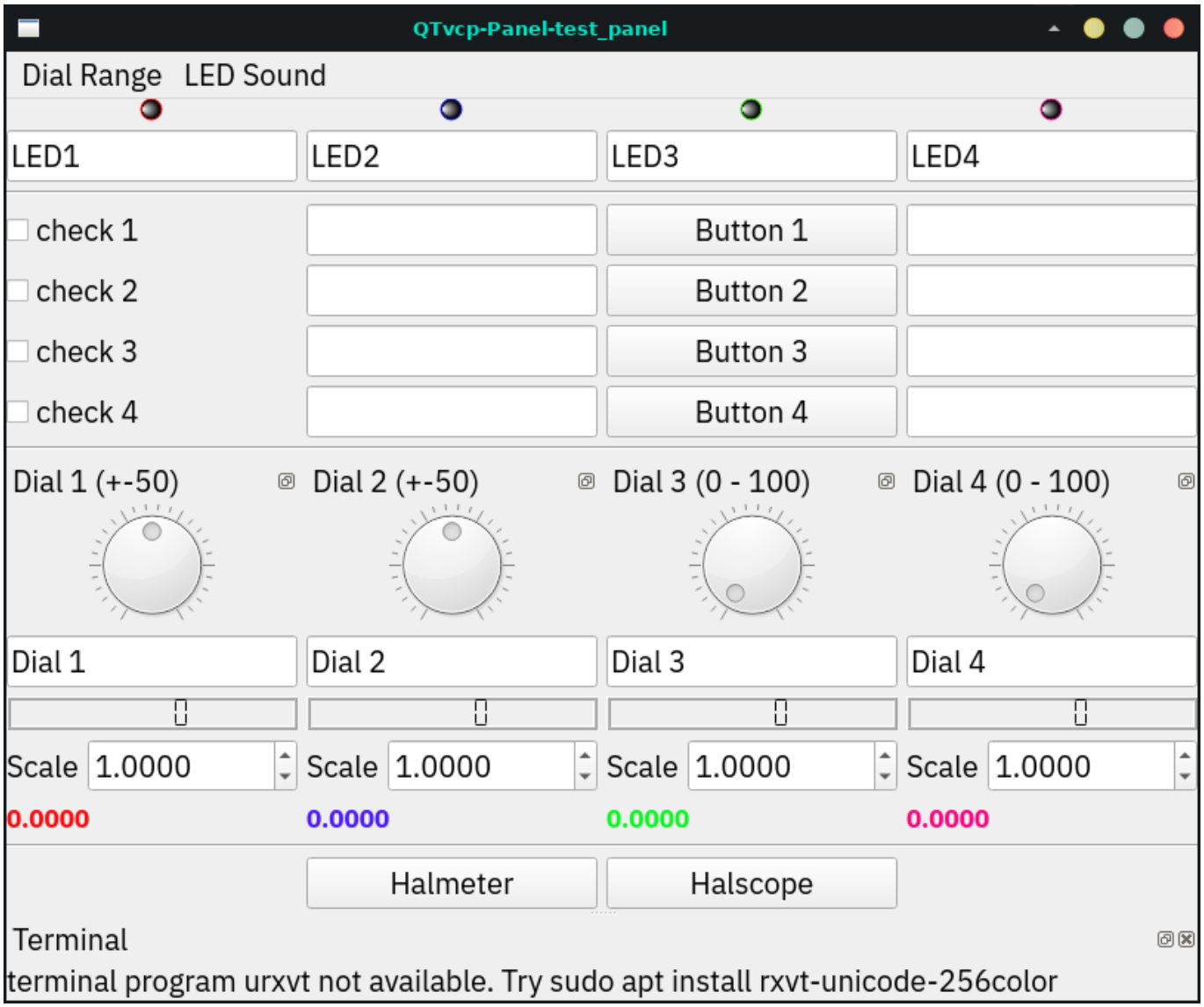


Figure 12.62: Вбудована панель тестування HAL QtVCP

cam_align
Віджет відображення камери для вирівнювання повороту.



Figure 12.63: cam_align - Вирівнювання камери VCP

sim_panel
: Невелика панель керування для імітації керування бігом MPG тощо.
Для імітації конфігурацій.



Figure 12.64: Вбудована панель QtVCP Sim

vismach_mill_xyz
: 3D OpenGL-вигляд 3-осьового фрезерного верстата.

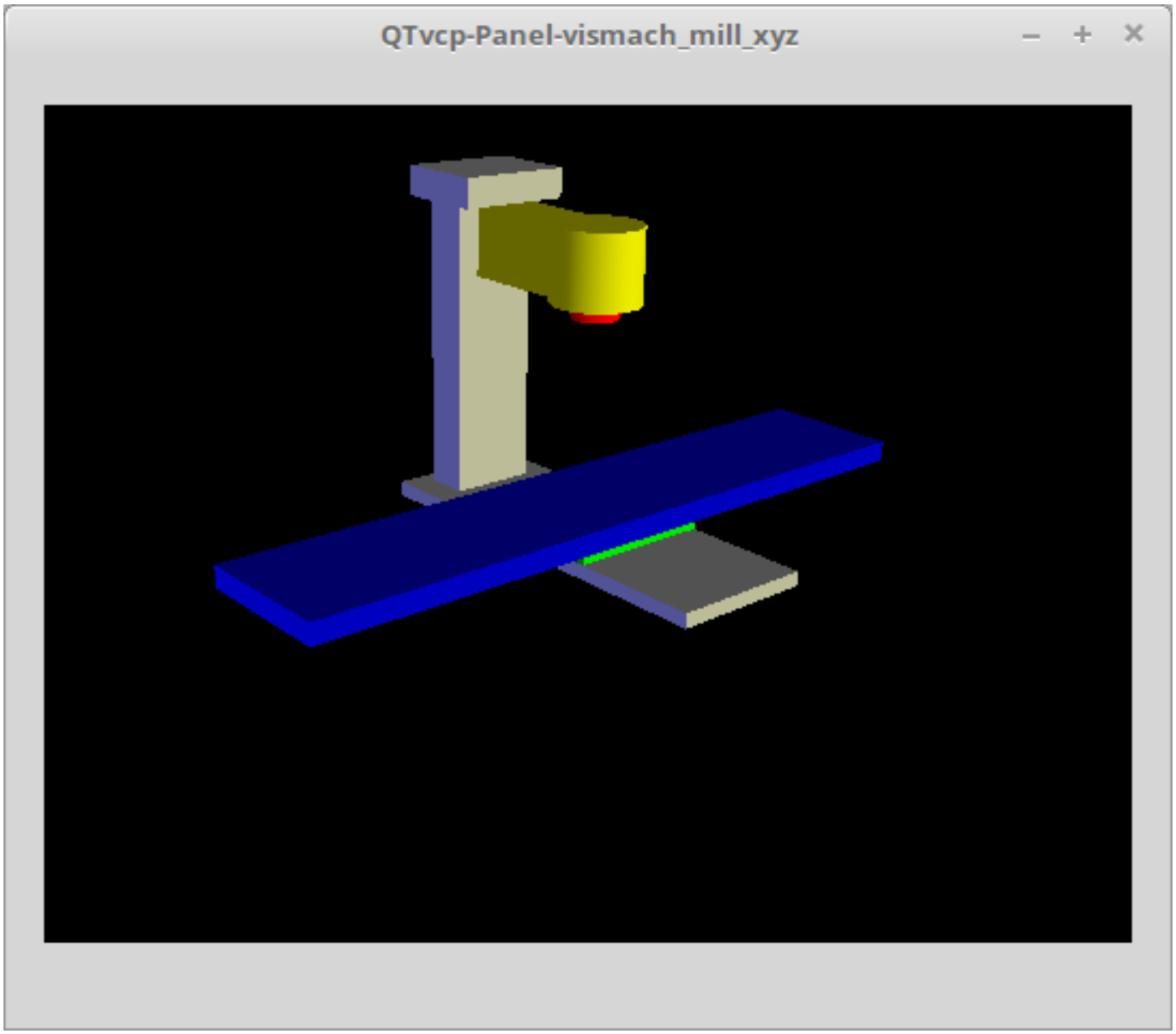


Figure 12.65: QtVismach - Вбудована панель 3-осьового фрезерного верстата

Ви можете завантажити їх з терміналу або з HAL-файлу за допомогою цієї базової команди:

```
loadusr qtvcp test_panel
```

Але частіше ось так:

```
loadusr -Wn test_panel qtvcp test_panel
```

Таким чином, HAL чекатиме, поки будуть створені контакти HAL, перш ніж продовжити.

12.5.3.2 Спеціальні панелі

Звісно, ви можете **створити власну панель та завантажити її**.

Якщо ви створили файл інтерфейсу користувача з назвою `my_panel.ui` та файл HAL з назвою `my_panel.hal`, ви завантажили б їх з терміналу за допомогою:

```
halrun -I -f my_panel.hal
```

Приклад HAL-файлу, що завантажує панель QtVCP

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''іb''b''вb'' b''рb''b' ↩
    'eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''чb''b''ab''b''cb''b''уb''
loadrt threads
loadrt classicladder_rt

# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''уb''b''вb''b''ab''b''тb''b' ↩
    'иb'' b''пb''b''рb''b''ob''b''гb''b''рb''b''ab''b''мb''b''иb'', b''щb''b''ob'' b''нb''b' ↩
    'eb'' b''пb''b''рb''b''ab''b''цb''b''юb''b''юb''b''тb''b''ьb'' b''уb'' b''рb''b''eb''b' ↩
    'ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b''уb'' b''чb''b''ab''b''cb''b''іb''
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # <t>\coref{1}{CO1-1}</t>

# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''иb'' b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb''
addf classicladder.0.refresh thread1


# b''зb'''b''єb''b''дb''b''нb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''нb''b''тb''b''ab''b' ↩
    'кb''b''тb''b''иb''
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

# b''пb''b''ob''b''чb''b''ab''b''тb''b''иb'' b''тb''b''eb''b''мb''b''уb''
start
```

❶ У цьому випадку ми завантажуємо `qtvcp` за допомогою **`-Wn`**, що очікує завершення завантаження панелі перед продовженням виконання наступної команди HAL.
Це робиться для того, щоб *переконатися, що створені панеллю контакти HAL дійсно готові*, якщо вони використовуються в решті файлу.

12.5.4 Створіть простий користувацький екран з чистого аркуша

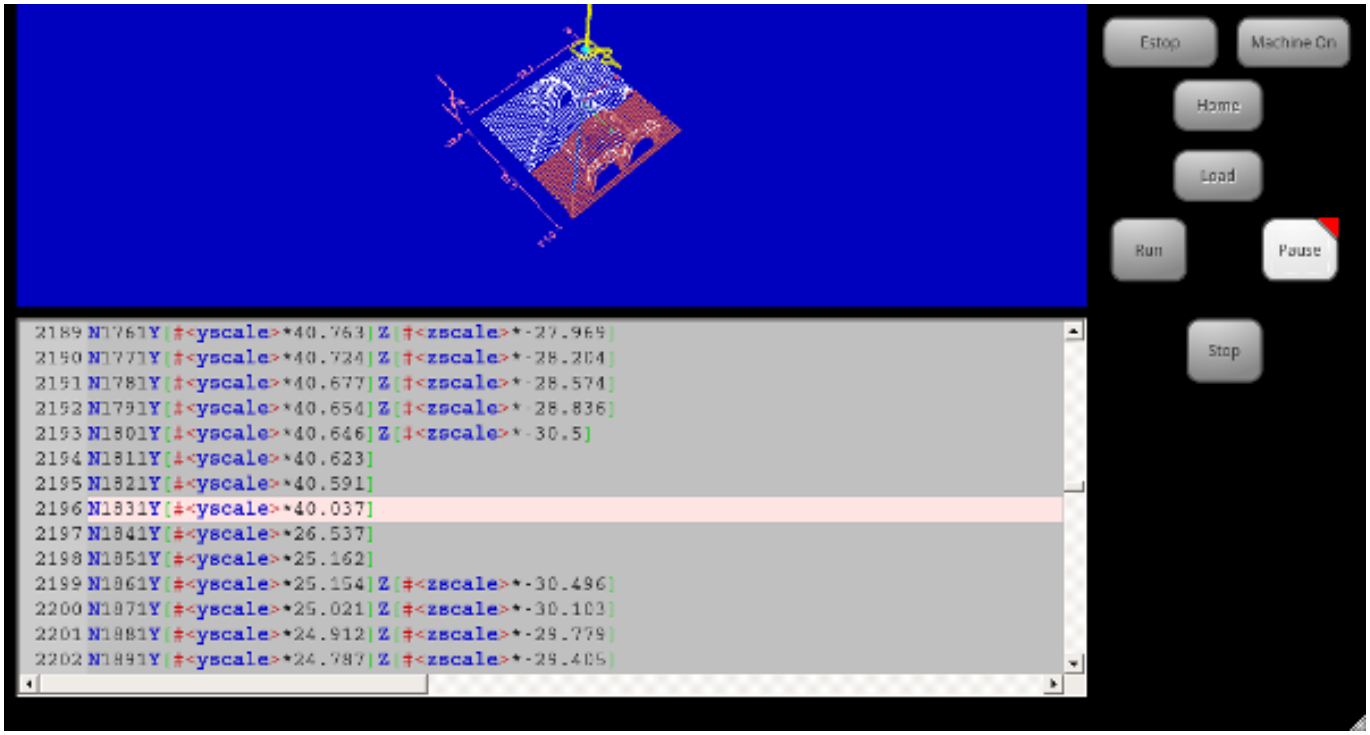


Figure 12.66: Потворний користувацький екран QtVCP

12.5.4.1 Огляд

Щоб створити панель або екран:

- Використовуйте Qt Designer, щоб створити дизайн, який вам подобається, та збережіть його у папці конфігурації з вибраним ім’ям, що закінчується на `.ui`
- Змініть INI-файл конфігурації, щоб завантажити QtVCP за допомогою вашого нового файлу `.ui`.
- Потім підключіть усі необхідні контакти HAL у файлі HAL.

12.5.4.2 Включити віджети LinuxCNC у Qt Designer

Встановлення Qt Designer Спочатку у вас має бути встановлено **Qt Designer**.
Наступні команди мають додати його до вашої системи або скористатися вашим менеджером пакетів для цього:

```
sudo apt-get install qttools5-dev-tools qttools5-dev libpython3-dev
```

Додати посилання qtvcp_plugin.py до шляху пошуку Qt Designer Потім вам потрібно додати посилання на `qtvcp_plugin.py` в одній з папок, у яких Qt Designer шукатиме.

У *RIP* версії LinuxCNC `qtvcp_plugin.py` буде:

’~/LINUXCNC_PROJECT_NAME/lib/python/qtvcp/plugins/qtvcp_plugin.py’

Для версії з *встановленим* пакетом має бути:

```
'usr/lib/python2.7/qtvcp/plugins/qtvcp_plugin.py' b''ab''b''6b''b''ob''
'usr/lib/python2.7/dist-packages/qtvcp/plugins/qtvcp_plugin.py'
```

Створіть символічне посилання на вищезгаданий файл і перемістіть його в одне з місць, де Qt Designer виконує пошук.

Qt Designer шукає посилання в цих двох місцях (виберіть одне):

```
'/usr/lib/x86_64-linux-gnu/qt5/plugins/designer/python' b''ab''b''6b''b''ob''
'~/.designer/plugins/python'
```

Можливо, вам знадобиться створити папку `plugins/python`.

Запустіть Qt Designer:

- Для *RIP-встановлення*:
  Відкрийте термінал, встановіть середовище для LinuxCNC <1>, потім завантажте Qt Designer <2> за допомогою:

```
. scripts/rip-environment  (1)
designer -qt=5             (2)
```

- Для *встановлення пакета*:
  Відкрийте термінал і введіть:

```
designer -qt=5
```

Якщо все пройде добре, Qt Designer запуститься, і ви побачите доступні для вибору віджети LinuxCNC ліворуч.

12.5.4.3 Зберіть екранний файл `.ui`

Створити віджет MainWindow Під час першого запуску Qt Designer відображається діалогове вікно
«Нова форма». Виберіть _«Головне вікно» та натисніть кнопку _«Створити».
Відображається віджет _«Головне вікно».

Ми збираємося зробити це вікно певного розміру, який не можна змінювати:

Встановити мінімальний та максимальний розмір `MainWindow`

- Візьміть кут вікна та змініть його розмір до відповідного розміру, скажімо, 1000x600.

- Клацніть правою кнопкою миші на вікні та виберіть «Встановити *мінімальний розмір*».
- Зробіть це ще раз і встановіть *максимальний розмір*.

Розмір нашого зразка віджета тепер не можна буде змінювати.

Додайте віджет ScreenOptions Перетягніть віджет `ScreenOptions` будь-де в головне вікно.

Цей віджет не додає нічого візуального, але налаштовує деякі **загальні опції**.

Рекомендується завжди *додавати цей віджет перед будь-яким іншим*.

Клацніть правою кнопкою миші на головному вікні, не на віджеті `ScreenOptions`, та встановіть вертикальне розташування *layout*, щоб зробити `ScreenOptions` повнорозмірним.

Додати вміст панелі Праворуч розташована панель із вкладками для *редактора властивостей* та *інспектора об'єктів*.

В інспекторі об'єктів натисніть на *ScreenOptions*.
Потім перейдіть до редактора властивостей і під заголовком *ScreenOptions* увімкніть **filedialog_optic**

Перетягніть віджет **GCodeGraphics** *widget* та **GcodeEditor** *widget*.
Розмістіть та змініть їх розмір на свій розсуд, залишивши місце для кнопок.

Додати кнопки дій Додайте 7 кнопок дій до головного вікна.

Якщо двічі клацнути кнопку, можна додати текст.
Відредагуйте підписи кнопок для «Стоп», «Увімкнення машини», «Додому», «Завантаження», «Виконати», «Пауза» та «Стоп».

Кнопки дій *за замовчуванням не виконують жодної дії*, тому нам потрібно змінити властивості для визначених функцій. Ви можете редагувати властивості:

- безпосередньо в *редакторі властивостей* у правій частині Qt Designer, або
- зручно, подвійне клацання лівою кнопкою миші на кнопці запускає *діалогове вікно властивостей*, яке дозволяє вибирати дії, відображаючи лише відповідні для дії дані.

Спочатку ми опишемо зручний спосіб:

- Клацніть правою кнопкою миші на кнопці «Увімкнення машини» та виберіть *Встановити дії*.
- Коли відобразиться діалогове вікно, скористайтеся випадаючим списком, щоб перейти до пункту `КЕРУВАННЯ МАШИНОЮ - Машина ввімкнена`.
- У цьому випадку для цієї дії немає опції, тому виберіть «OK».

Тепер кнопка вмикатиме машину при натисканні.

А тепер прямий шлях за допомогою редактора властивостей Qt Designer:

- Виберіть кнопку «Увімкнути машину».
- Перейдіть до Редактора властивостей у правій частині Qt Designer.
- Прокрутіть униз, доки не знайдете заголовок *ActionButton*.
- У списку властивостей та значень ви побачите прапорець дії `machine_on`.

Тепер кнопка керуватиме вмиканням/вимиканням машини.

Зробіть те саме для всіх інших кнопок, додавши:

- За допомогою кнопки «Головна» нам також потрібно змінити властивість `joint_number` на `-1`. Це вказує контролеру *перевести в початкове положення всі осі*, а не певну вісь.
- За допомогою кнопки «Пауза»:
  - Під заголовком `Indicated_PushButton` перевірте `indicator_option`.
  - Під заголовком `QAbstactButton` поставте галочку навпроти `checkable`.

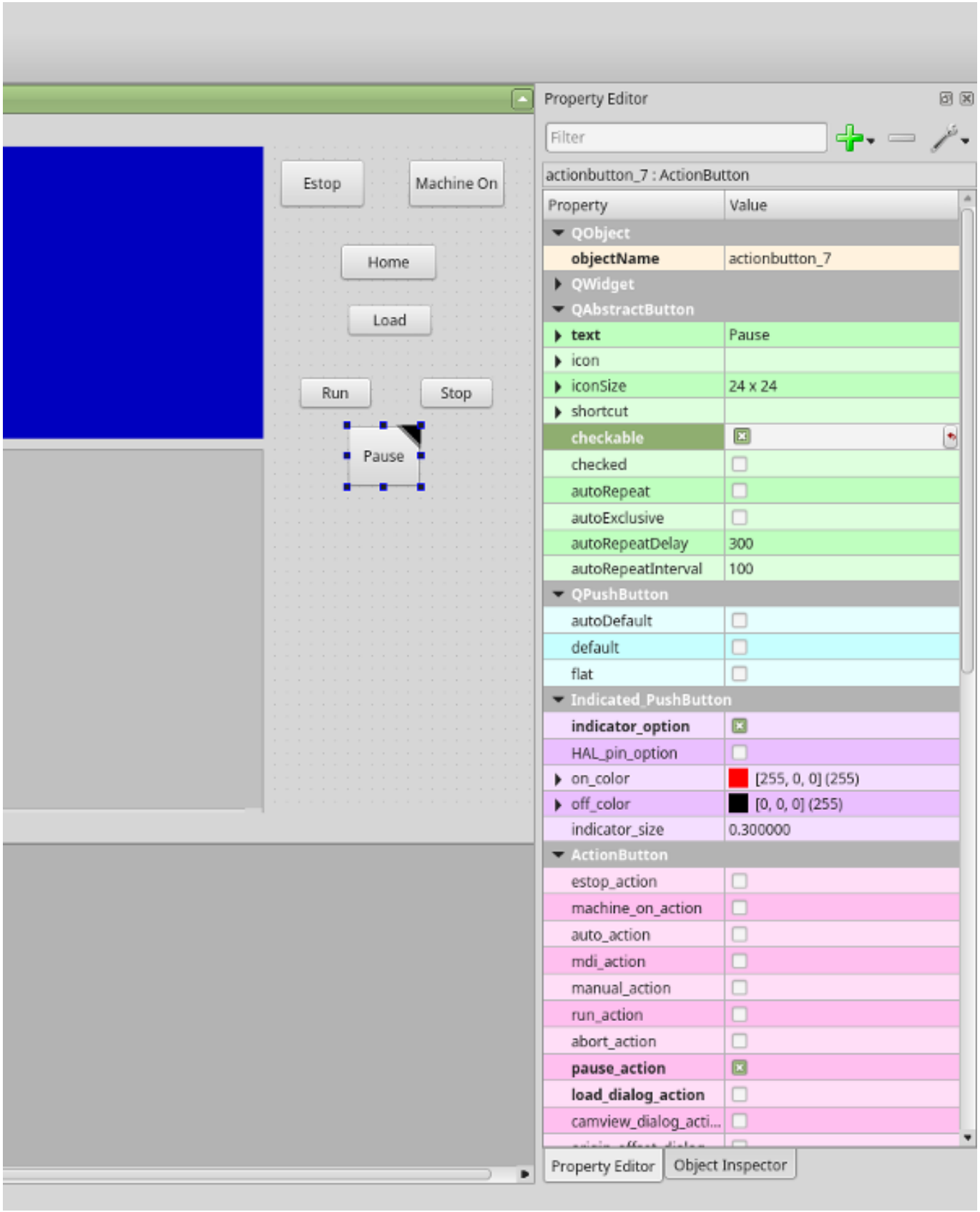


Figure 12.67: Qt Designer: Вибір властивостей кнопки паузи

Збережіть файл .ui Потім нам потрібно зберегти цей дизайн як `tester.ui` у папці `sim/qtvcp`.

Ми зберігаємо його як *tester*, оскільки це ім'я файлу, яке розпізнає QtVCP і використовуватиме вбудований файл-обробник для його відображення.

12.5.4.4 Файл обробника

Файл обробника **обов'язковий**.

Це дозволяє писати налаштування на Python.

Наприклад, *елементи керування клавіатурою* зазвичай записуються у файлі обробника.

У цьому прикладі вбудований файл `tester_handler.py` використовується автоматично: він виконує мінімум необхідних дій для відображення екрана, визначеного `tester.ui`, та базової роботи з клавіатурою.

12.5.4.5 Конфігурація INI

[DISPLAY] Розділ Якщо ви використовуєте QtVCP для створення екрана керування CNC, у файлі *INI* під заголовком `[DISPLAY]` встановіть:

```
DISPLAY = qtvcp <screen_name>
```

`<screen_name>` – це *базова назва* файлів `.ui` та `_handler.py`.

У нашому прикладі вже є конфігурація симулятора під назвою tester, яку ми використовуватимемо для відображення нашого тестового екрана.

[HAL] Розділ Якщо на вашому екрані використовуються *віджети з виводами HAL*, тоді вам потрібно **підключити їх у файлі HAL**.

QtVCP шукає у файлі *INI*, під заголовком `[HAL]`, записи нижче:

`POSTGUI_HALFILE=<filename>`
: Зазвичай `<filename>` буде `+<screen_name>_postgui.hal+`, але може бути будь-яким допустимим іменем файлу.
Ви можете мати *кілька рядків `POSTGUI_HALFILE`* в INI: кожен з них буде виконуватися один за одним у порядку їх появи.
Ці команди *виконуються після побудови екрану*, що гарантує доступність контактів HAL віджета.

`POSTGUI_HALCMD=<command>`
: `<command>` може бути будь-якою дійсною командою HAL.
У файлі INI може бути кілька рядків `POSTGUI_HALCMD`: кожна з них буде виконана послідовно в порядку їх появи.
Щоб гарантувати доступність контактів HAL віджета, виконуються такі команди:

- *після того, як екран буде побудовано*,
- *після виконання всіх POSTGUI_HALFILE*.

У нашому прикладі немає контактів HAL для підключення.

12.5.5 Детальний опис файлу обробника

Файли обробників використовуються для *створення користувацьких елементів керування за допомогою Python*.

12.5.5.1 Огляд

Ось приклад файлу обробника.

Для зручності обговорення його розбито на розділи.

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################
import sys
import os
import linuxcnc

b''зb'' PyQt5 b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' 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

# b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb'' b' ↩
    'вb''b''eb''b''дb''b''eb''b''нb''b''нb''b''яb'' b''жb''b''уb''b''pb''b''нb''b''ab''b' ↩
    'лb''b''уb''
from qtvcp import logger
LOG = logger.getLogger(__name__)

# b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''pb''b''ib''b' ↩
    'вb''b''eb''b''нb''b''ьb'' b''жb''b''уb''b''pb''b''нb''b''ab''b''лb''b''уb'' b''дb''b' ↩
    'лb''b''яb'' b''цb''b''ьb''b''ob''b''гb''b''ob'' b''мb''b''ob''b''дb''b''уb''b''лb''b' ↩
    'яb''
#LOG.setLevel(logger.INFO) # b''Ob''b''дb''b''иb''b''нb'' b''зb'' b''нb''b''ab''b''cb''b' ↩
    'тb''b''уb''b''пb''b''нb''b''иb''b''хb'' b''пb''b''уb''b''нb''b''кb''b''тb''b''ib''b' ↩
    'вb'': b''Hb''b''Ab''b''Лb''b''Ab''b''Гb''b''Ob''b''Дb''b''Жb''b''Eb''b''Hb''b''Hb''b' ↩
    'Яb'', b''Ib''b''Hb''b''Фb''b''Ob''b''Pb''b''Mb''b''Ab''b''Цb''b''Ib''b''Яb'', b''Пb''b' ↩
    'Ob''b''Пb''b''Eb''b''Pb''b''Eb''b''Дb''b''Жb''b''Eb''b''Hb''b''Hb''b''Яb'', b''Пb''b' ↩
    'Ob''b''Mb''b''Иb''b''Лb''b''Kb''b''Ab'', b''Kb''b''Pb''b''Иb''b''Tb''b''Иb''b''Чb''b' ↩
    'Hb''b''Иb''b''Йb''

##########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''»  ↩
    **** #
##########################################

KEYBIND = Keylookup()
STATUS = Status()
ACTION = Action()
###################################
# **** b''Cb''b''Eb''b''Kb''b''Цb''b''Ib''b''Яb'' b''Kb''b''Лb''b''Ab''b''Cb''b''Уb'' b' ↩
    'Xb''b''Eb''b''Hb''b''Дb''b''Лb''b''Eb''b''Pb'' **** #
###################################

class HandlerClass:

    ########################
    # **** b''Ib''b''Hb''b''Ib''b''Цb''b''Ib''b''Ab''b''Лb''b''Ib''b''3b''b''Ab''b''Цb''b' ↩
        'Ib''b''Яb'' **** #
    ########################
    # b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''дb''b''ob''b''зb''b''вb''b''ob'' ↩
        b''лb''b''яb''b''юb''b''тb''b''ьb'' b''дb''b''ob''b''cb''b''тb''b''уb''b''пb'' b' ↩
        'дb''b''ob'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''зb'' b' ↩
        'фb''b''ab''b''йb''b''лb''b''ib''b''вb'' QtVCP
```

```
    # b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b' ↩
        'ib'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''тb''b''ab'' b''пb''b' ↩
        'ib''b''нb''b''иb'' hal b''щb''b''eb'' b''нb''b''eb'' b''cb''b''тb''b''вb''b''ob''b' ↩
        'pb''b''eb''b''нb''b''ib''
    def __init__(self, halcomp,widgets,paths):
        self.hal = halcomp
        self.w = widgets
        self.PATHS = paths

    ########################################
    # b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Cb''b''Пb''b''Eb''b''Цb''b''Ib''b''Ab'' ↩
        b''Лb''b''Ьb''b''Нb''b''Иb''b''Хb'' b''Фb''b''Уb''b''Нb''b''Kb''b''Цb''b''Ib''b' ↩
        'Йb''              #
    ########################################

    # b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b' ↩
        'ib'':
    # b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''ib''b''нb''b''cb''b''тb''b''ab'' ↩
        b''нb''b''цb''b''ib''b''юb''b''юb''b''тb''b''ьb''b''cb''b''яb''.
    # b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'' HAL b''пb''b''ob''b''бb''b' ↩
        'уb''b''дb''b''ob''b''вb''b''ab''b''нb''b''ib'', b''ab''b''лb''b''eb'' HAL b''нb''b' ↩
        'eb'' b''гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''йb'' b''дb''b''ob'' b''pb''b' ↩
        'ob''b''бb''b''ob''b''тb''b''иb''.
    # b''Tb''b''уb''b''тb'' b''вb''b''иb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''юb''b' ↩
        'єb''b''тb''b''eb'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'' HAL b' ↩
        'ab''b''бb''b''ob'' b''ib''b''нb''b''ib''b''цb''b''ib''b''ab''b''лb''b''ib''b''зb''b ↩
        ''уb''b''єb''b''тb''b''eb'' b''cb''b''тb''b''ab''b''нb'' b''вb''b''ib''b''дb''b' ↩
        'жb''b''eb''b''тb''b''ib''b''вb'' b''тb''b''ob''b''щb''b''ob''.
    def initialized__(self):
        pass

    def processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):
        # b''пb''b''ib''b''дb'' b''чb''b''ab''b''cb'' b''нb''b''ab''b''бb''b''ob''b''pb''b' ↩
            'уb'' b''тb''b''eb''b''кb''b''cb''b''тb''b''уb'' b''вb'' MDI b''мb''b''иb'' b' ↩
            'нb''b''eb'' b''хb''b''ob''b''чb''b''eb''b''мb''b''ob'', b''щb''b''ob''b''бb'' b ↩
            ''кb''b''ob''b''мb''b''бb''b''ib''b''нb''b''ab''b''цb''b''ib''b''їb'' b''кb''b' ↩
            'лb''b''ab''b''вb''b''ib''b''шb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b' ↩
            'ab''b''лb''b''иb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''їb'',
        # b''тb''b''ob''b''мb''b''уb'' b''мb''b''иb'' b''бb''b''eb''b''зb''b''пb''b''ob''b' ↩
            'cb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob'' b''фb''b''ib''b''кb''b' ↩
            'cb''b''уb''b''єb''b''мb''b''ob'' b''тb''b''ab'' b''ob''b''бb''b''pb''b''ob''b' ↩
            'бb''b''лb''b''яb''b''єb''b''мb''b''ob'' b''пb''b''ob''b''дb''b''ib''b''їb''.
        # b''Ob''b''дb''b''нb''b''ab''b''кb'' b''мb''b''иb'' b''хb''b''ob''b''чb''b''eb''b' ↩
            'мb''b''ob'', b''щb''b''ob''b''бb'' ESC, F1 b''тb''b''ab'' F2 b''вb''b''иb''b' ↩
            'кb''b''лb''b''иb''b''кb''b''ab''b''лb''b''иb'' b''фb''b''уb''b''нb''b''кb''b' ↩
            'цb''b''ib''b''їb'' b''кb''b''ob''b''мb''b''бb''b''ib''b''нb''b''ab''b''цb''b' ↩
            'ib''b''йb'' b''кb''b''лb''b''ab''b''вb''b''ib''b''шb''
        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):

            # b''пb''b''ob''b''шb''b''уb''b''кb'' b''вb''b''eb''b''pb''b''хb''b''нb''b' ↩
                'ьb''b''ob''b''гb''b''ob'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ab'' ↩
                 b''зb'' b''уb''b''cb''b''ib''b''хb'' b''вb''b''ib''b''дb''b''жb''b''eb''b' ↩
                'тb''b''ib''b''вb'', b''яb''b''кb''b''ib'' b''ob''b''тb''b''pb''b''иb''b' ↩
                'мb''b''ab''b''лb''b''иb'' b''пb''b''ob''b''дb''b''ib''b''юb''
            # b''пb''b''ob''b''тb''b''ib''b''мb'' b''пb''b''eb''b''pb''b''eb''b''вb''b' ↩
                'ib''b''pb''b''кb''b''ab'', b''чb''b''иb'' b''цb''b''eb'' b''тb''b''ob''b' ↩
                'йb'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb'', b''дb''b''ob'' b''яb''b' ↩
                'кb''b''ob''b''гb''b''ob'' b''мb''b''иb'' b''хb''b''ob''b''чb''b''eb''b' ↩
                'мb''b''ob'' b''нb''b''ab''b''пb''b''pb''b''ab''b''вb''b''иb''b''тb''b''иb'' ↩
                 b''пb''b''ob''b''дb''b''ib''b''їb'' b''нb''b''ab''b''тb''b''иb''b''cb''b' ↩
                'кb''b''ab''b''нb''b''нb''b''яb'' b''кb''b''лb''b''ab''b''вb''b''ib''b''шb''
```

```
            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):
                    # b''яb''b''кb''b''щb''b''ob'' b''вb'' b''pb''b''yb''b''чb''b''нb''b' ↩
                        'ob''b''мb''b''yb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''ib'' b' ↩
                        'вb''b''иb''b''кb''b''ob''b''нb''b''yb''b''єb''b''мb''b''ob'' b' ↩
                        'нb''b''ab''b''шb''b''ib'' b''кb''b''ob''b''мb''b''бb''b''ib''b' ↩
                        'нb''b''ab''b''цb''b''ib''b''їb'' b''кb''b''лb''b''ab''b''вb''b' ↩
                        'ib''b''шb'' - b''ib''b''нb''b''ab''b''кb''b''шb''b''eb''
                    # b''нb''b''ab''b''дb''b''cb''b''иb''b''лb''b''ab''b''тb''b''иb'' b' ↩
                        'пb''b''ob''b''дb''b''ib''b''їb'' b''дb''b''ob'' b''вb''b''ib''b' ↩
                        'дb''b''жb''b''eb''b''тb''b''ab'' G-b''кb''b''ob''b''дb''b''yb''
                    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

        # b''дb''b''ob''b''бb''b''pb''b''eb'', b''яb''b''кb''b''щb''b''ob'' b''мb''b''иb''  ↩
            b''cb''b''юb''b''дb''b''иb'' b''дb''b''ib''b''йb''b''шb''b''лb''b''иb'', b''тb'' ↩
            b''ob'' b''cb''b''пb''b''pb''b''ob''b''бb''b''yb''b''йb''b''тb''b''eb'' b''кb''b ↩
            ''ob''b''мb''b''бb''b''ib''b''нb''b''ab''b''цb''b''ib''b''їb'' b''кb''b''лb''b' ↩
            'ab''b''вb''b''ib''b''шb''
        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

    ########################
    # b''3b''b''Bb''b''Ob''b''Pb''b''Ob''b''Tb''b''Hb''b''Ib'' b''Bb''b''Иb''b''Kb''b''Лb'' ↩
        b''Иb''b''Kb''b''Иb'' b''3b''b''Ib'' b''Cb''b''Tb''b''Ab''b''Tb''b''Уb''b''Cb''b' ↩
        'Уb'' #
    ########################

    ######################
```

```
    # b''Зb''b''Bb''b''Ob''b''Pb''b''Ob''b''Tb''b''Hb''b''Ib'' b''Bb''b''Иb''b''Kb''b''Лb'' ↩
        b''Иb''b''Kb''b''Иb'' b''Зb'' b''Фb''b''Ob''b''Pb''b''Mb''b''Иb'' #
    ########################

    ####################
    # b''Зb''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Hb''b''Ib'' b''Фb''b''Уb''b''Hb''b''Kb'' ↩
        b''Цb''b''Ib''b''Їb'' #
    ####################

    # b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''аb''b''нb''b''нb''b''яb'' b''мb'' ↩
        b''іb''b''жb'' b''кb''b''лb''b''аb''b''вb''b''іb''b''шb''b''аb''b''мb''b''иb'' b' ↩
        'зb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''мb''b''иb'' b''пb''b''pb'' ↩
        b''иb''b''вb''b''яb''b''зb''b''кb''b''иb'' b''кb''b''лb''b''аb''b''вb''b''іb''b' ↩
        'шb''
    # b''пb''b''ob''b''дb''b''вb''b''ob''b''їb''b''тb''b''иb'' b''шb''b''вb''b''иb''b''дb'' ↩
        b''кb''b''іb''b''cb''b''тb''b''ьb'', b''яb''b''кb''b''щb''b''ob'' b''шb''b''вb''b' ↩
        'иb''b''дb''b''кb''b''іb''b''cb''b''тb''b''ьb'' b''єb'' b''пb''b''pb''b''аb''b''вb'' ↩
        b''дb''b''ob''b''юb''
    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)

    ####################
    # b''Пb''b''Pb''b''Иb''b''Bb''b'''b''b''Яb''b''Зb''b''Kb''b''Ab'' b''Kb''b''Лb''b''Ab'' ↩
        b''Bb''b''Ib''b''Шb'' b''Bb''b''Иb''b''Kb''b''Лb''b''Иb''b''Kb''b''Ib''b''Bb'' #
    ####################

    # b''Уb''b''пb''b''pb''b''аb''b''вb''b''лb''b''іb''b''нb''b''нb''b''яb'' b''мb''b''аb'' ↩
        b''шb''b''иb''b''нb''b''ob''b''юb''
    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()

    # b''Лb''b''іb''b''нb''b''іb''b''йb''b''нb''b''иb''b''йb'' b''бb''b''іb''b''гb'' b' ↩
        'пb''b''іb''b''дb''b''тb''b''юb''b''пb''b''цb''b''eb''b''мb''
```

```
    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)

    ############################
    # **** b''зb''b''ab''b''кb''b''лb''b''юb''b''чb''b''нb''b''иb''b''йb'' b''зb''b''ab''b' ↩
        'xb''b''ib''b''дb'' **** #
    ############################

    ##############################
    # b''кb''b''ob''b''дb'' b''нb''b''eb''b''ob''b''бb''b''xb''b''ib''b''дb''b''нb''b''ob'' ↩
        b''гb''b''ob'' b''кb''b''лb''b''ab''b''cb''b''yb'' b''кb''b''ob''b''тb''b''лb''b' ↩
        'ab'' #
    ##############################

    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)

################################
# b''нb''b''eb''b''ob''b''бb''b''xb''b''ib''b''дb''b''нb''b''иb''b''йb'' b''кb''b''ob''b' ↩
    'дb'' b''кb''b''ob''b''тb''b''лb''b''ab''-b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b' ↩
    'иb''b''кb''b''ab'' #
################################

def get_handlers(halcomp,widgets,paths):
     return [HandlerClass(halcomp,widgets,paths)]
```

12.5.5.2 Розділ ІМПОРТ

Цей розділ призначений для **імпорту необхідних модулів бібліотеки** для вашого екрана.

Типовим було б імпортувати бібліотеки QtVCP *keybinding, Status* та *Action*.

12.5.5.3 Розділ «СТВОРЕННЯ ЕКЗЕМПЛЯРІВ БІБЛІОТЕК»

Створюючи екземпляри бібліотек тут, ми **створюємо глобальне посилання**.

Ви можете помітити це за командами, які не мають перед собою `self`.

За домовленістю ми *пишемо з великої літери назви бібліотек, на які посилаються в усьому світі*.

12.5.5.4 Розділ класу обробника

Користувацький код розміщено *в класі, щоб QtVCP міг його використовувати*.

Це визначення класу обробника.

12.5.5.5 Розділ INICIALIZE

Як і всі бібліотеки Python, функція **`+__init__+`** викликається під час *першого створення екземпляра* бібліотеки.

Тут ви налаштуєте *defaults*, а також *reference variables* та *global variables*.

Посилання на віджети на даний момент недоступні.

Змінні `halcomp`, `widgets` та `paths` надають доступ до компонента HAL, віджетів та інформації про шляхи QtVCP відповідно.

12.5.5.6 Розділ СПЕЦІАЛЬНИХ ФУНКЦІЙ

Існує кілька *спеціальних функцій*, які QtVCP шукає у файлі обробника. Якщо QtVCP їх знаходить, він їх викликає, якщо ні, то тихо проігнорує.

`class_patch__(self):`
: *Класове патчування*, також відоме як *мавпяче патчування*, дозволяє **перекривати виклики функцій в імпортованому модулі**.
Класове патчування повинно бути виконано *до інстанціювання модуля*, і воно *модифікує всі інстанції*, створені після цього.
Прикладом може бути патчування викликів кнопок з редактора G-коду для виклику функцій у файлі обробника.
Функція класового патчування, переозначена тут, буде викликатися з екземпляром HandlerClass як «self», а не з екземпляром патчованого класу. Це може ускладнити доступ до функцій/змінних патчованого класу.
При класовому патчуванні поза класом HandlerClass виклик функції буде використовувати екземпляр патчованого класу як «self».

`initialized__(self):`
: Ця функція *викликається після того, як віджети та виводи HAL будуть побудовані*.
Ви можете маніпулювати віджетами та виводами HAL або додавати більше виводів HAL тут.
Зазвичай може бути

- перевірені та встановлені налаштування,
- стилі, застосовані до віджетів,
- Стан підключеного до функцій LinuxCNC.
- будуть додані комбінації клавіш.

pre_hal_init__(self):
Ця функція викликається перед викликом функції hal_init_ для HAL-іфікованих віджетів. Деякі зміни властивостей необхідно внести перед викликом HAL_init для віджета.

after_override__(self):
Ця функція викликається після завантаження додаткового файлу перевизначення, але до завантаження додаткового файлу HAL або встановлення компонента HAL у стан готовності.

processed_key_event__(self,receiver,event,is_pressed,key,code,shift,cntrl):
Ця функція викликається для полегшення *перемикання клавіш* тощо. Використовуючи бібліотеку _`keybinding`, її можна легко використовувати для додавання функцій, пов’язаних з натисканнями клавіш.

keypress_event__(self,receiver, event):
Ця функція повертає **необроблені події натискання клавіш**. Вона має *пріоритет* над_ обробленою_подією_клавіші.

keyrelease_event__(receiver, event):
Ця функція видає **необроблені події звільнення ключів**. Вона має *пріоритет* над_ обробленою_подією_ключів.

before_loop__(self):
Ця функція *викликається безпосередньо перед входом у цикл подій Qt*. У цей момент усі віджети/бібліотеки/код ініціалізації завершилися, і екран вже відображається.

system_shutdown_request__(self):
Якщо ця функція присутня, вона **перекриває звичайну функцію, що викликається для повного вимкнення системи**.
Вона може використовуватися для виконання *підготовчих дій перед вимкненням*.

Bibliography

[1] + Система Linux не вимкнеться, якщо використовується ця функція, ви повинні зробити це самостійно. QtVCP/LinuxCNC завершить роботу без попередження, як тільки ця функція повернеться.

closing_cleanup__(self):
Ця функція *викликається безпосередньо перед закриттям екрана*. Її можна використовувати для очищення перед закриттям.

12.5.5.7 Розділ зворотних викликів стану

За домовленістю, саме тут слід розміщувати функції, які є **зворотними викликами з визначень STATUS**.

12.5.5.8 ЗВОРОТНІ ВИКЛИКИ З РОЗДІЛУ ФОРМИ

За домовленістю, саме тут слід розміщувати функції, які є **зворотними викликами віджетів, підключених до MainWindow** у редакторі Qt Designer.

12.5.5.9 Розділ ЗАГАЛЬНИХ ФУНКЦІЙ

За домовленістю, саме тут ви розміщуєте свої **загальні функції**.

12.5.5.10 Розділ ПРИВ'ЯЗКИ КЛАВІШ

Якщо ви *використовуєте бібліотеку `keybinding`,* саме тут ви розміщуєте свої **користувацькі процедури виклику клавіш**.

Сигнатура функції:

```
def on_keycall_KEY(self,event,state,shift,cntrl):
    if state:
        self.do_something_function()
```

KEY код (з бібліотеки клавіатурних скорочень) для потрібної клавіші.

12.5.5.11 Розділ ЗАКРИТТЯ ПОДІЇ

Розміщення тут функції **`closeEvent` перехоплюватиме події закриття**.

Це *замінює будь-яку попередньо визначену функцію `closeEvent`* з QtVCP.

```
def closeEvent(self, event):
    self.do_something()
    event.accept()
```

Note
Зазвичай краще використовувати спеціальну функцію `closing_cleanup__`.

12.5.6 Підключення віджетів до коду Python

Можна підключати віджети до коду Python за допомогою **сигналів та слотів**.

Таким чином ви можете:

- *Надайте нові функції віджетам LinuxCNC,* або
- *Використовуйте стандартні віджети Qt для керування LinuxCNC*.

12.5.6.1 Огляд

У редакторі Qt Designer:

- Ви *створюєте слоти функцій користувача*
- Ви *підключаєте слоти до віджетів за допомогою сигналів*.

У файлі обробника:

- Ви *створюєте функції слота,* визначені в Qt Designer.

12.5.6.2 Використання Qt Designer для додавання слотів

Коли ви завантажите екран у Qt Designer, додайте на екран просту кнопку `PushButton`.
Ви можете змінити назву кнопки на щось цікаве, наприклад, `test_button`.

Існує два способи редагування з’єднань - це графічний спосіб.

- У верхній панелі інструментів Qt Designer є кнопка для редагування сигналів. Після натискання на неї, якщо натиснути і утримувати кнопку, з’явиться стрілка (схожа на сигнал заземлення з електричної схеми).
- Пересуньте цю стрілку до частини головного вікна, на якій немає віджетів.
- З’явиться діалогове вікно «Налаштування з’єднань».
  - Список ліворуч містить доступні сигнали з віджета.
  - Список праворуч містить доступні слоти в головному вікні, і ви можете до них додавати нові.
- Виберіть сигнал `clicked()` — це зробить сторону слотів доступною.
- Натисніть «Редагувати» у списку слотів.
- З’явиться діалогове вікно «Слоти/сигнали головного вікна».
- У списку слотів зверху є значок «+» – натисніть на нього.
- Тепер ви можете редагувати нову назву слота.
- Видаліть назву за замовчуванням `slot()` та змініть її на `test_button()`.
- Натисніть кнопку «OK».
- Ви повернетеся до діалогового вікна «Налаштування підключень».
- Тепер ви можете вибрати новий слот у списку слотів.
- Потім натисніть «OK» та збережіть файл.

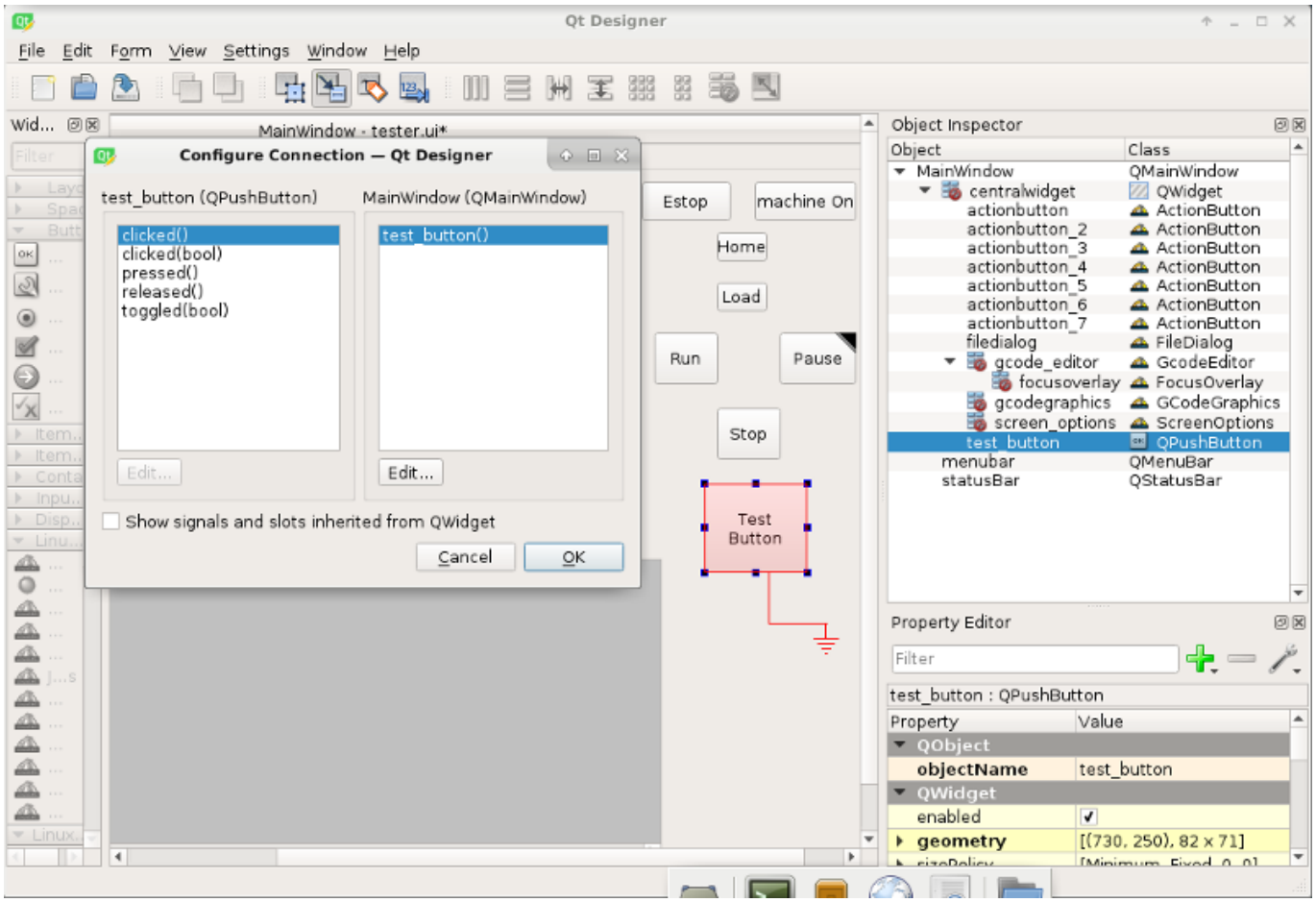


Figure 12.68: Вибір сигналу/слота в Qt Designer

12.5.6.3 Зміни обробника Python

Тепер вам потрібно **додати функцію до файлу обробника**.

Сигнатура функції — **`def slot_name(self):`**.

Для нашого прикладу ми додамо код для виведення назви віджета:

```
def test_button(self):
    name = self.w.sender().text()
    print(name)
```

Додайте цей код під розділом з назвою:

```
########################
# b''зb''b''вb''b''оb''b''рb''b''оb''b''тb''b''нb''b''іb'' b''вb''b''иb''b''кb''b''лb''b' ↩
    'иb''b''кb''b''иb'' b''зb'' b''фb''b''оb''b''рb''b''мb''b''иb'' b''№b''
########################
```

Насправді не має значення, де в класі обробника ви розміщуєте команди, але за домовленістю це саме те місце, де їх слід розміщувати.

Збережіть файл обробника.

Тепер, коли ви завантажуєте екран і натискаєте кнопку, у терміналі має вивести назву кнопки.

12.5.7 Більше інформації

Вбудовані віртуальні панелі керування QtVCP

QtVCP Widgets

QtVCP Libraries

Qt Vismach

Фрагменти коду файлу обробника QtVCP

QtVCP Development

QtVCP Віджети для конструктора Qt з користувацьким використанням

12.6 Віртуальні панелі керування QtVCP

QtVCP можна використовувати для **створення панелей керування**, які взаємодіють з *HAL*.

12.6.1 Вбудовані віртуальні панелі керування

Доступно кілька **вбудованих панелей HAL**.

У терміналі введіть `qtvcp list`, щоб побачити список.

12.6.1.1 `копія`

Використовується для **копіювання вбудованого коду екранів/панелей VCP/QtVismach QtVCP до папки**, щоб його можна було *налаштувати*.

У термінальному запуску:

```
qtvcp copy
```

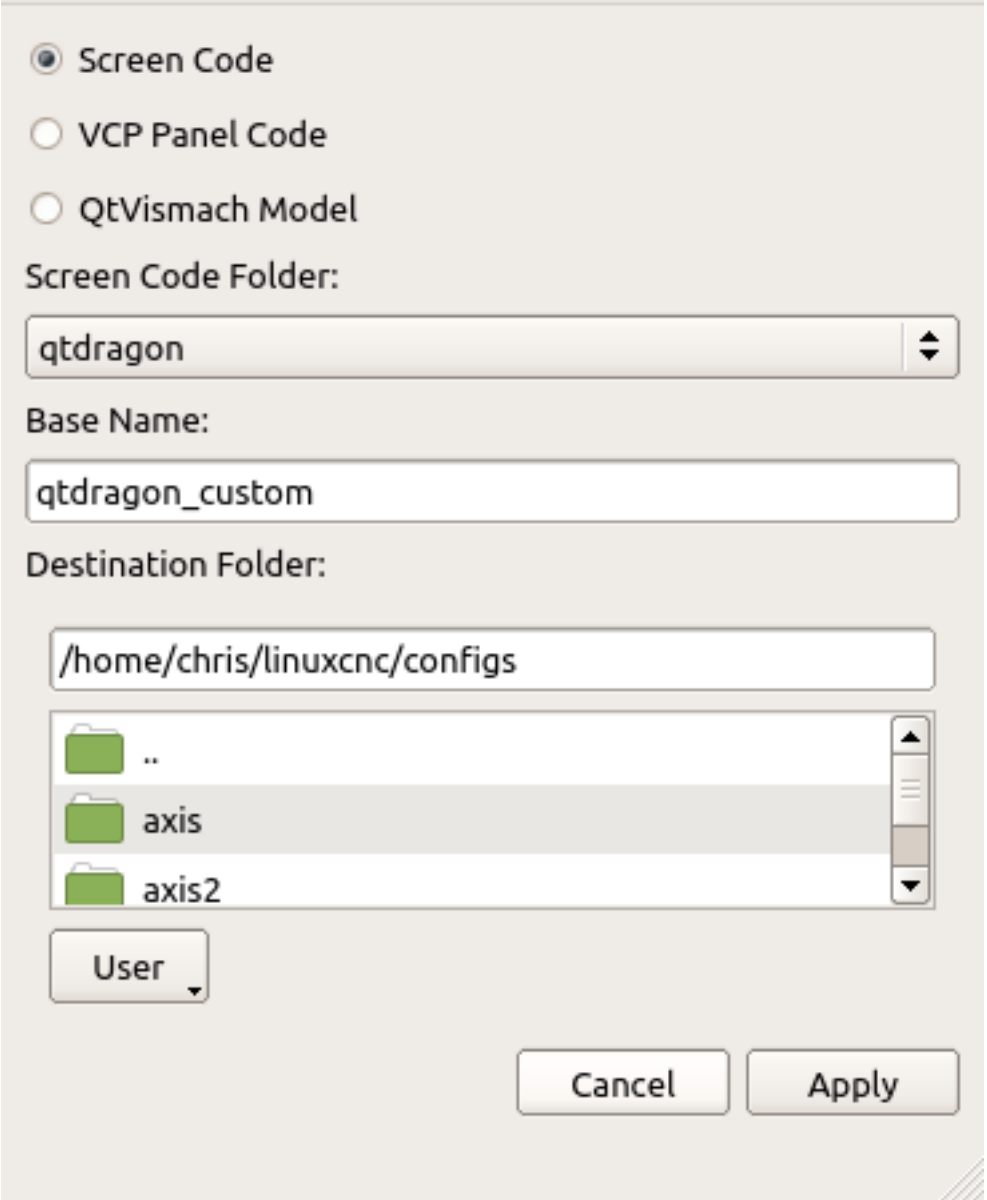
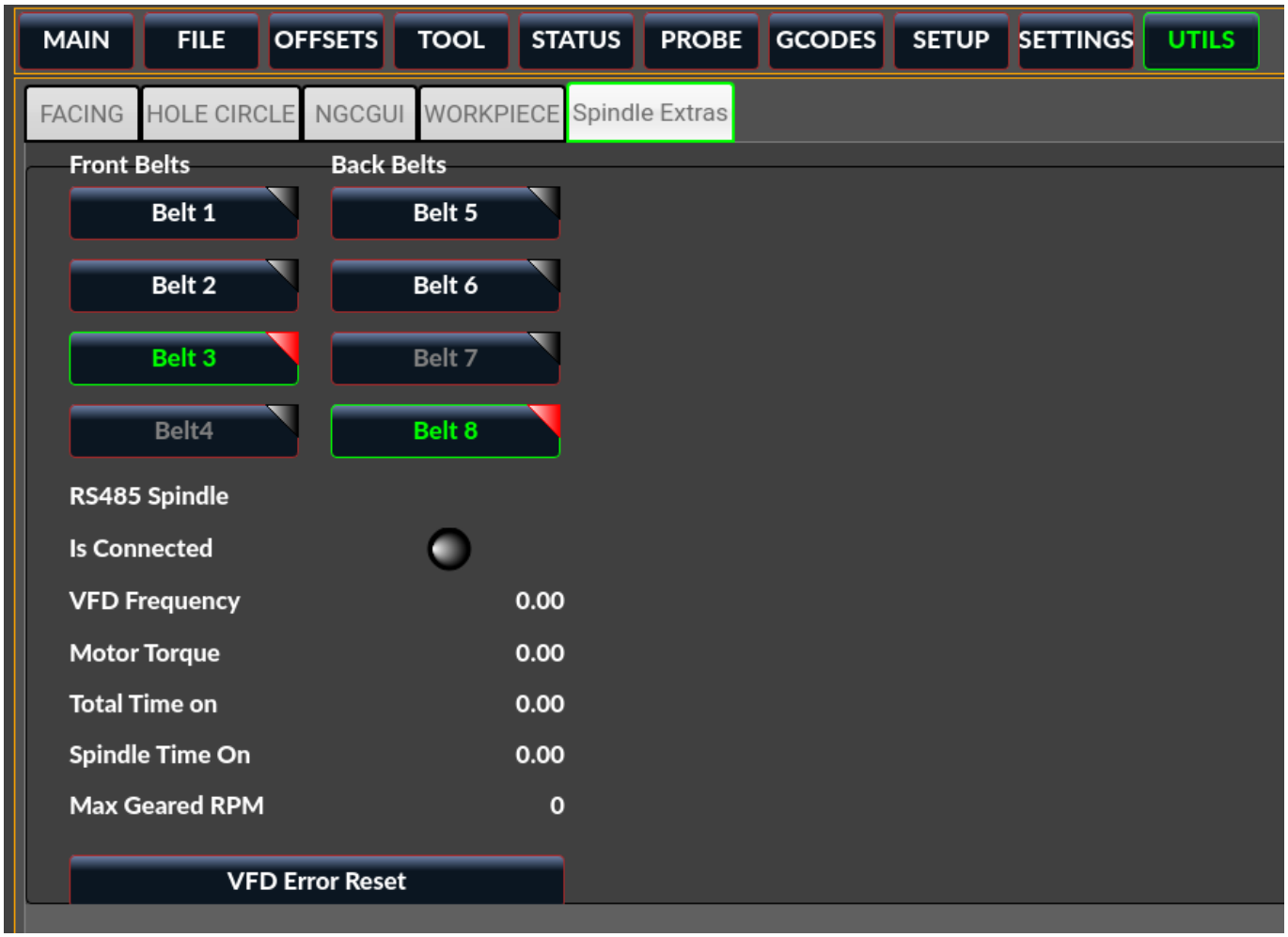


Figure 12.69: Діалогове вікно `копіювання` QtVCP - екран, панель VCP або панель копіювання коду QtVismach

12.6.1.2 spindle_belts

Ця панель призначена для відображення додаткових даних частотного перетворювача RS485, а також для налаштування 4-шківного шпиндельного приводу з 2 ременями за допомогою серії кнопок.



Крім того, це також корисний шаблон для вашої власної панелі, оскільки він містить:

- Відображення додаткових даних HAL
- Кнопки та групи кнопок
- Динамічні зміни стану кнопок увімкнено/вимкнено залежно від стану інших кнопок
- Збереження даних у файлі qtdragon.prefs
- Кнопка користувача для скидання налаштувань частотно-регульованого перетворювача (VFD)

Змініть цю панель відповідно до своїх потреб. Використовуються найпоширеніші функції. Перевага використання панелей полягає в тому, що вони відокремлюють ваш власний код відображення від основного коду qtdragon, тому оновлення системи не порушить ваші налаштування.

- Шпиндельний привід (наприклад VFDMOD)
- Спеціальний компонент, який масштабує частоту частотного перетворювача для отримання бажаної швидкості шпинделя.
- Шпиндель з ремінним приводом, який використовує два ремені та проміжний натяжний шків, подібно до свердлильного верстата.
- Підключіть вхідні контакти qtdragon.belts.<назва-кода> у вашому HAL-файлі postgui.

Ремені розділені на дві групи кнопок: передні ремені та задні ремені. Вони пронумеровані відповідно до таблички на машині. Кнопки в групі є взаємовиключними, тобто в групі можна вибрати тільки одну.

Крім того, з таким механізмом неможливо встановити обидва ремені на одному рівні, оскільки на один шків натяжного ролика не можна встановити два ремені. Тому, якщо вибрано один ремінь, його протилежна кнопка стає неактивною. Наприклад, якщо вибрано ремінь 3, ремінь 7 стає неактивним.

Додайте ці рядки до розділу [DISPLAY] у вашому INI-файлі.
Приклад tab_location стосується екрана QtDragon.

```
EMBED_TAB_NAME=Spindle Extras
EMBED_TAB_COMMAND=qtvcp spindle_belts
EMBED_TAB_LOCATION=tabWidget_utilities
```

Ось як завантажити `spindle_belt` зі скрипта HAL:

```
loadusr qtvcp spindle_belts
```

Налаштування панелі:

- Скопіюйте файли, розташовані в /user/share/qtvcp/qtdragon/panels/belts, до: ~/linuxcnc/configs/<my_c
  (для цього можна скористатися діалоговою панеллю копіювання)
- Редагувати belts.ui за допомогою дизайнера.
- Редагуйте belts_handler.py за допомогою текстового редактора
- Підключіть відповідні контакти у файлі postgui.hal
- Переконайтеся, що ваш файл postgui завантажено вашим INI-файлом.

Щоб отримати детальнішу інформацію, зверніться до документації QtVCP та QtDragon. Файл обробника Python також надає корисний шаблон для будь-якої користувацької панелі.

12.6.1.3 `test_dial`

- Ця панель має **циферблат, який регулює вихідні контакти S32 та Float HAL**.
- Діапазон циферблата можна налаштувати у випадаючому меню.
- Вивід можна масштабувати за допомогою `spinbox`.
- Для автоматичного вибору та підключення до сигналу можна використовувати `combobox`.

```
loadusr qtvcp test_dial
```

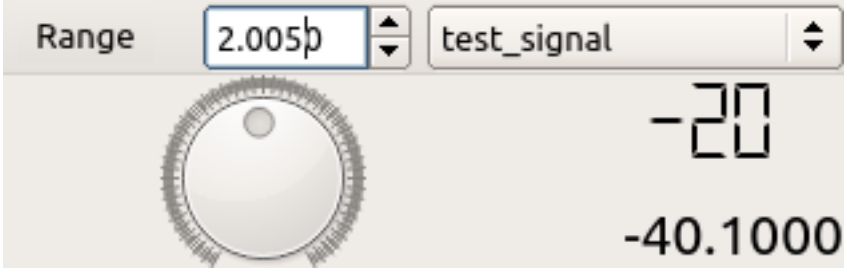


Figure 12.70: QtVCP `test_dial` Панель - Тестовий циферблат VCP

12.6.1.4 test_button

- Ця панель має **кнопку, яка встановлює PIN-код HAL**.
- Кнопку можна вибрати як *минуточасну* або _кнопку_перемикання.
- Буде створено HAL-пін, який відповідає стану кнопки.
- Колір *індикатора* кнопки можна налаштувати у випадаючому меню.
- Вивід або сигнал HAL можна вибрати для відстеження стану кнопки.
- Ви можете додати більше кнопок з випадаючого меню.
- Ви можете завантажити Halmeter з випадаючого меню.
- Ви можете завантажити тестовий світлодіод з випадаючого меню.
- Кнопку можна від’єднати від основних вікон.

Ось як завантажити `test_button` зі скрипта HAL:

```
loadusr qtvcp test_button
loadusr qtvcp -o 4 test_button
```

Перемикач `-o` встановлює кількість кнопок, з яких починається завантаження панелі. Якщо завантаження відбувається безпосередньо з терміналу, пропустіть `loadusr`.

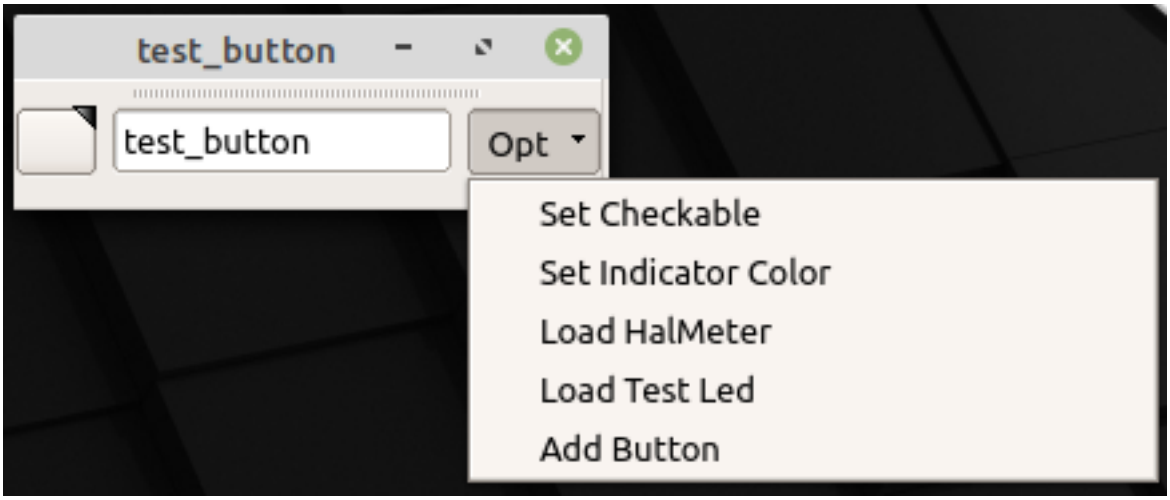


Figure 12.71: QtVCP `test_button` – Тестування кнопки VCP

12.6.1.5 test_led

- Ця панель має **світлодіод, який можна вибрати для відстеження виводів/сигналів бітів HAL**.
- Колір світлодіодів можна налаштувати у випадаючому меню.
- Текстове поле та стан можна вивести як мовлення, якщо вибрано звук.
- «Випадаючий список» можна використовувати для автоматичного вибору та підключення до виводу/сигналу.
- Ви можете додати більше світлодіодів з випадаючого меню.

- Світлодіод можна від’єднати від основних вікон.

Ось як завантажити `test_led` зі скрипта HAL:

```
loadusr qtvcp test_led
loadusr qtvcp -o 4 test_led
```

Перемикач `-o` встановлює кількість світлодіодів, з яких починається завантаження панелі. Якщо завантаження відбувається безпосередньо з терміналу, пропустіть `loadusr`.

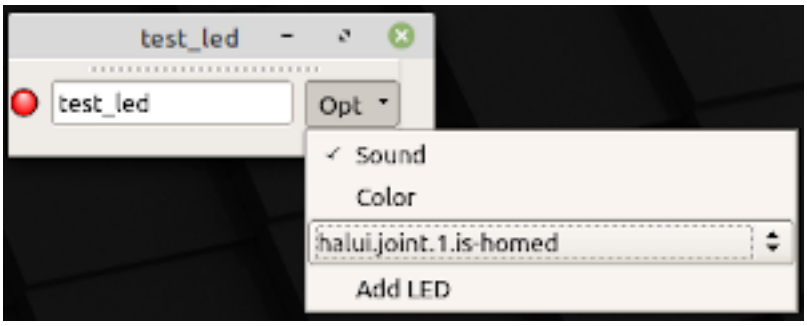


Figure 12.72: QtVCP `test_dial` Панель - Тестовий світлодіод VCP

12.6.1.6 `test_panel`

Колекція корисних віджетів для тестування компонента HAL, включаючи озвучування стану світлодіода.

```
loadusr qtvcp test_panel
```

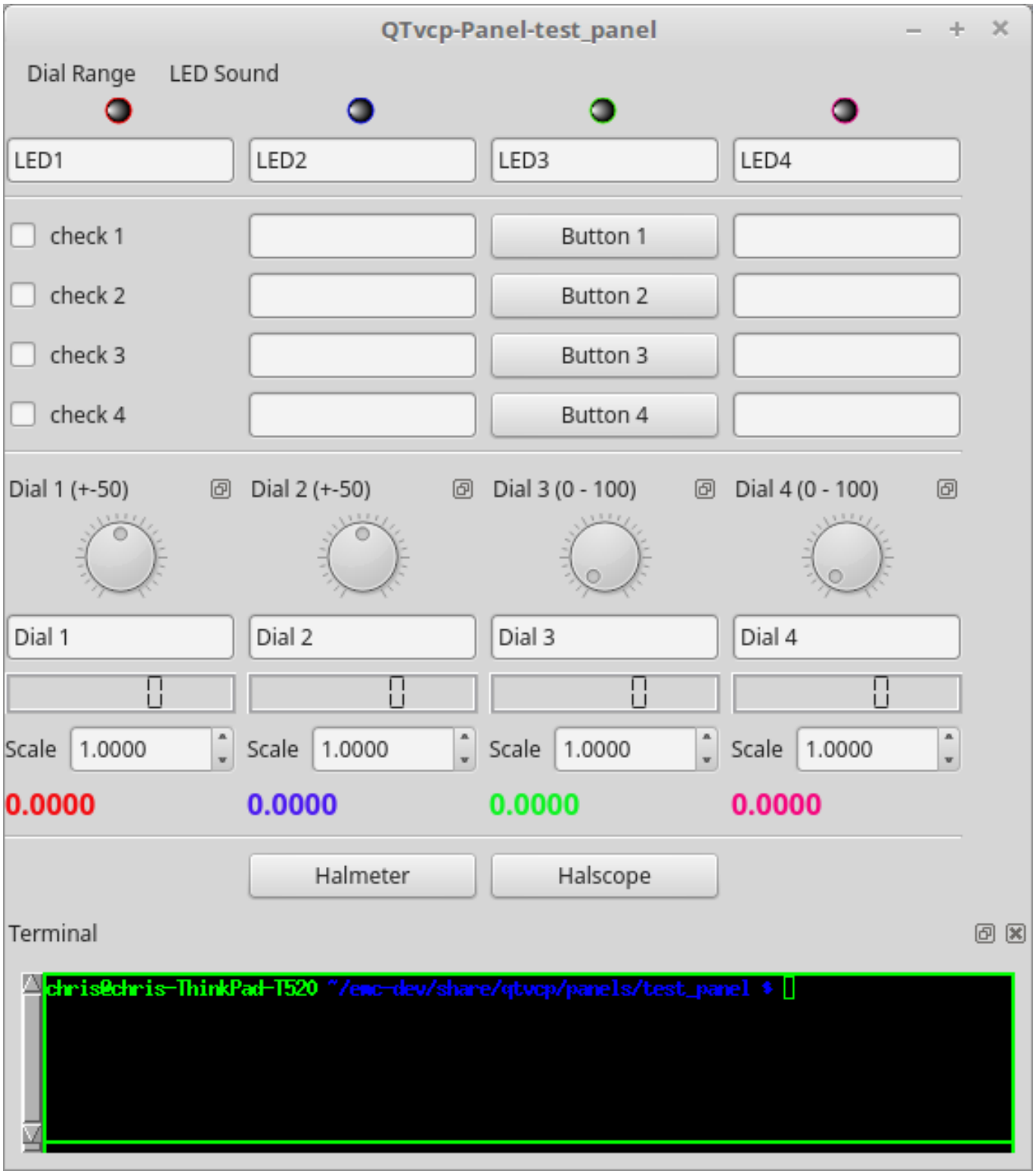


Figure 12.73: QtVCP `test_panel` - Панель тестування компонентів HAL

12.6.1.7 `cam_align`

Віджет відображення камери для вирівнювання повороту.



Figure 12.74: Панель QtVCP `cam_align` – панель вирівнювання на основі камери

Застосування Додайте ці рядки до INI-файлу:

```
[DISPLAY]
EMBED_TAB_NAME = cam_align
EMBED_TAB_COMMAND = halcmd loadusr -Wn qtvcp_embed qtvcp -d -c qtvcp_embed -x {XID}  ↩
    cam_align
# b''Нb''b''ab''b''cb''b''тb''b''уb''b''пb''b''нb''b''иb''b''йb'' b''рb''b''яb''b''дb''b' ↩
    'ob''b''кb'' b''пb''b''ob''b''тb''b''рb''b''іb''b''бb''b''еb''b''нb'', b''яb''b''кb''b' ↩
    'щb''b''ob'' b''вb''b''бb''b''уb''b''дb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b' ↩
    'cb''b''яb'' b''вb'' GMOCCAPY
EMBED_TAB_LOCATION = ntb_preview
```

Note
Усі `<options>` мають стояти перед назвою панелі `cam_align`.

Параметри Qtvcp

- `-c NAME` Назва компонента HAL. За замовчуванням використовується назва файлу інтерфейсу користувача.
- `-d` Налагодження ввімкнено. або вилучення для відсутності мінімального виводу
- `-v` Увімкнено детальне налагодження. Ви можете знайти всі доступні роздільні здатності.
- `-x {XID}` використовується для вбудовування в AXIS або Gmoccapy
- `-o <опція>` Параметри, що передаються до cam_align. можуть використовувати кілька записів -o

Параметри вирівнювання Cam_align Ось доступні опції -o:

- `size=400,400` Розмір вбудованого вікна (ширина, висота)
- `imagesize=300,300` Розмір зображення у вікні (ширина, висота)
- `rotincr=5` Встановлює крок обертання перехрестя прицілу. (градуси)
- `xscale=100` Масштабує зображення по осі X. Від'ємне значення призведе до перевертання зображення по осі X (у відсотках)
- `yscale=100` Масштабує зображення по осі Y. Від'ємне значення призведе до перевертання зображення по осі Y (у відсотках)
- `camnumber=1` Встановлює, яку системну камеру використовувати
- `api=V4L2` Встановлює бібліотеку OpenCV для камери
- `res=1280,720` Встановлює запитувану роздільну здатність (ширину, висоту)

Наприклад, ви можете додати ширину та висоту вікна, крок повороту та номер камери з INI за допомогою опцій -o.

```
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
```

Керування мишею:

- одинарний клік лівою кнопкою миші - збільшити обертання перехрестя на один крок
- одинарний клік правою кнопкою миші - зменшити обертання перехрестя на один крок
- одинарне клацання середньою кнопкою миші — перемикання між кроками обертання
- Утримуйте ліву кнопку миші та прокручуйте - прокручуйте, масштабуючи камеру
- утримуйте праву кнопку миші та прокручуйте - кут повороту перехрестя прокручування
- лише прокручування мишею - діаметр кола прокручування
- подвійне клацання лівою кнопкою миші - скинути масштаб
- подвійний клік правою кнопкою миші - скинути обертання
- подвійне клацання середньою кнопкою миші - скинути діаметр кола

Щоб використовувати верхні кнопки, потрібно призначити команду (або підпрограму). Це може виглядати так:

```
[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
```

Де перша команда відноситься до кнопки «SET origin», а друга — до кнопки «GOTO Origin». Зверніть увагу, що кома та текст після неї є необов'язковими — вони замінять текст кнопки за замовчуванням.
Ці кнопки є кнопками дій QtVCP і підпорядковуються цим правилам.

12.6.1.8 sim_panel

Невелика панель управління для **імітації елементів управління бігом MPG тощо** для імітованих конфігурацій.
MPG, кнопки вибору та кнопки управління експортують контакти HAL для підключення до linuxcnc.
Групи вибору та управління можна приховати, якщо вони не потрібні, за допомогою опції «-o hide=».
«groupBoxControl» та «groupBoxSelection» — це назви віджетів, які можна приховати.
Якщо ви хочете приховати обидва, використовуйте між ними кому без пробілів.
Опція «-a» зробить панель завжди активною над усіма вікнами.

```
loadusr qtvcp sim_panel
```

Тут ми завантажуємо панель без кнопок вибору MPG та з опцією «завжди зверху».

```
loadusr qtvcp -a -o hide=groupBoxSelection sim_panel
```



Figure 12.75: QtVCP `sim_panel` - Імітована панель керування для тестування екрану.

12.6.1.9 `tool_dialog`

Діалогове вікно ручної зміни інструменту з описом інструменту.

```
loadusr -Wn tool_dialog qtvcp -o speak_on -o audio_on tool_dialog
```

Опції:

- `-o notify_on` - *використовуйте діалогові вікна сповіщень на робочому столі* замість власних вікон QtVCP.
- `-o audio_on` - *відтворювати звук під час зміни інструменту*

- `-o speak_on` - *озвучити оголошення про зміну інструменту*

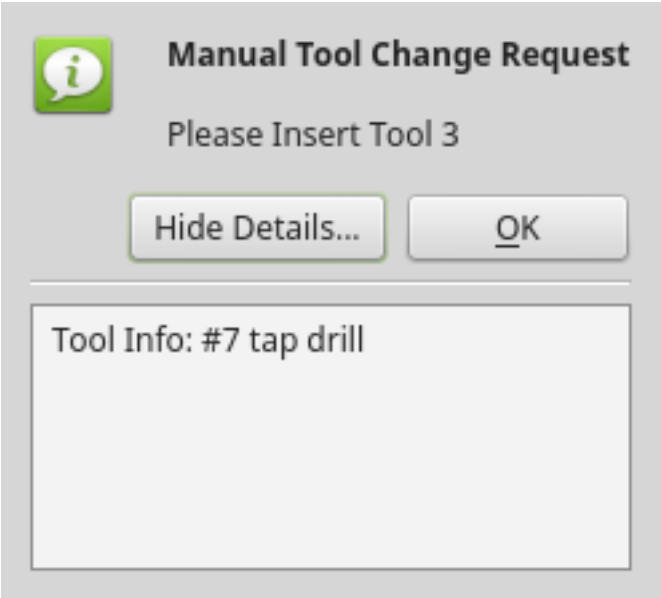


Figure 12.76: QtVCP `tool_dialog` - Діалогове вікно ручної зміни інструменту

12.6.2 `vismach` Панелі 3D-моделювання

Ці панелі є попередньо створеними симуляціями поширених типів машин.
Їх також можна вбудовувати в інші екрани, такі як AXIS або GMOCCAPY.

12.6.2.1 QtVCP `vismach_mill_xyz`

3D OpenGL-вигляд *3-осьового фрезерного верстата*.

```
loadusr qtvcp vismach_mill_xyz
```

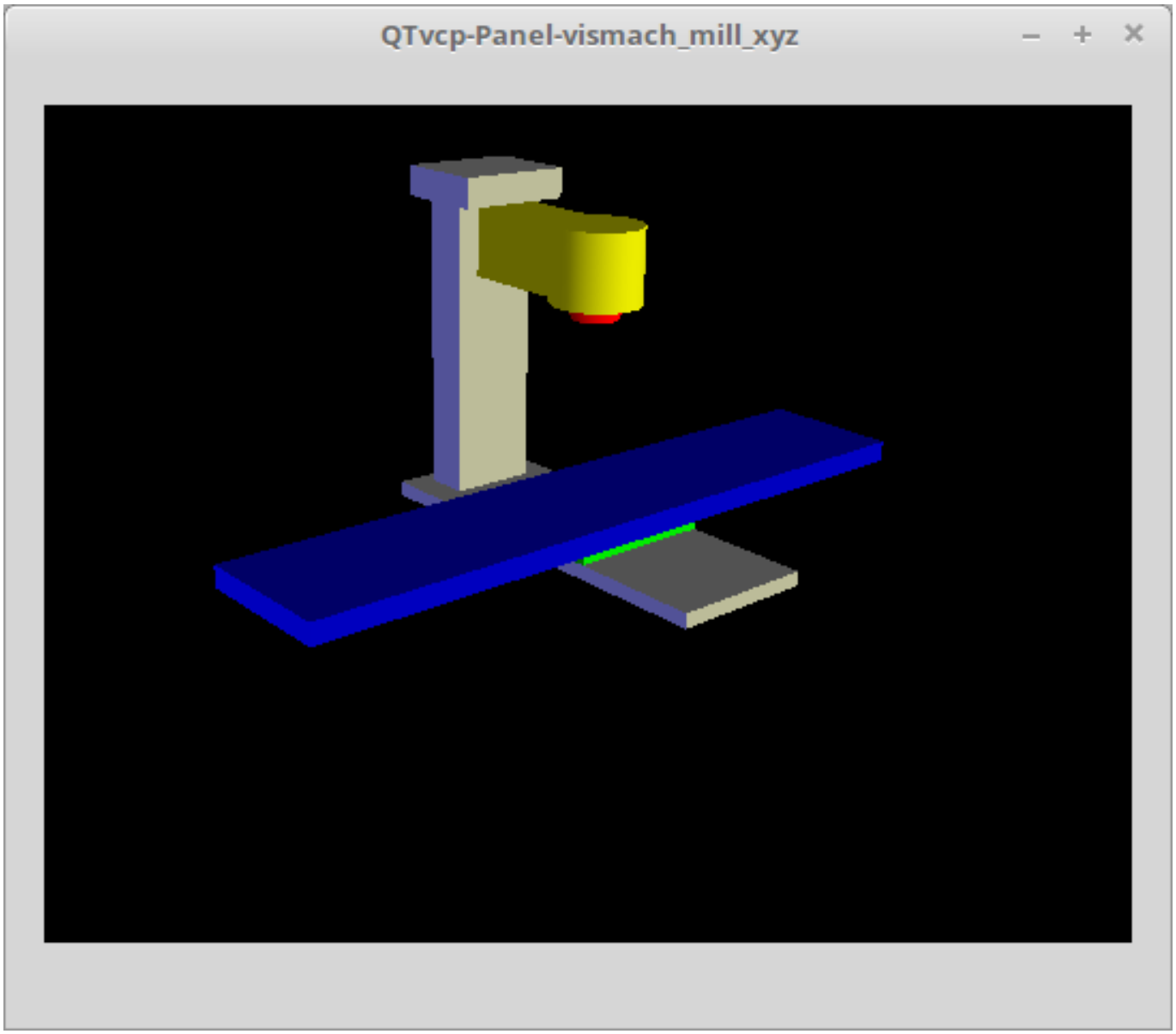


Figure 12.77: QtVCP `vismach_mill_xyz` - Панель 3D-вигляду 3-осьового фрезерного верстата

12.6.2.2 QtVCP `vismach_router_atc`

3D-вигляд OpenGL фрезерного верстата з 3-осьовим маршрутизатором і портальною платформою. Ця панель показує, як визначити та підключити деталі моделі у файлі обробника, а не імпортувати готову модель із бібліотеки vismach QtVCP.

```
loadusr qtvcp vismach_router_atc
```

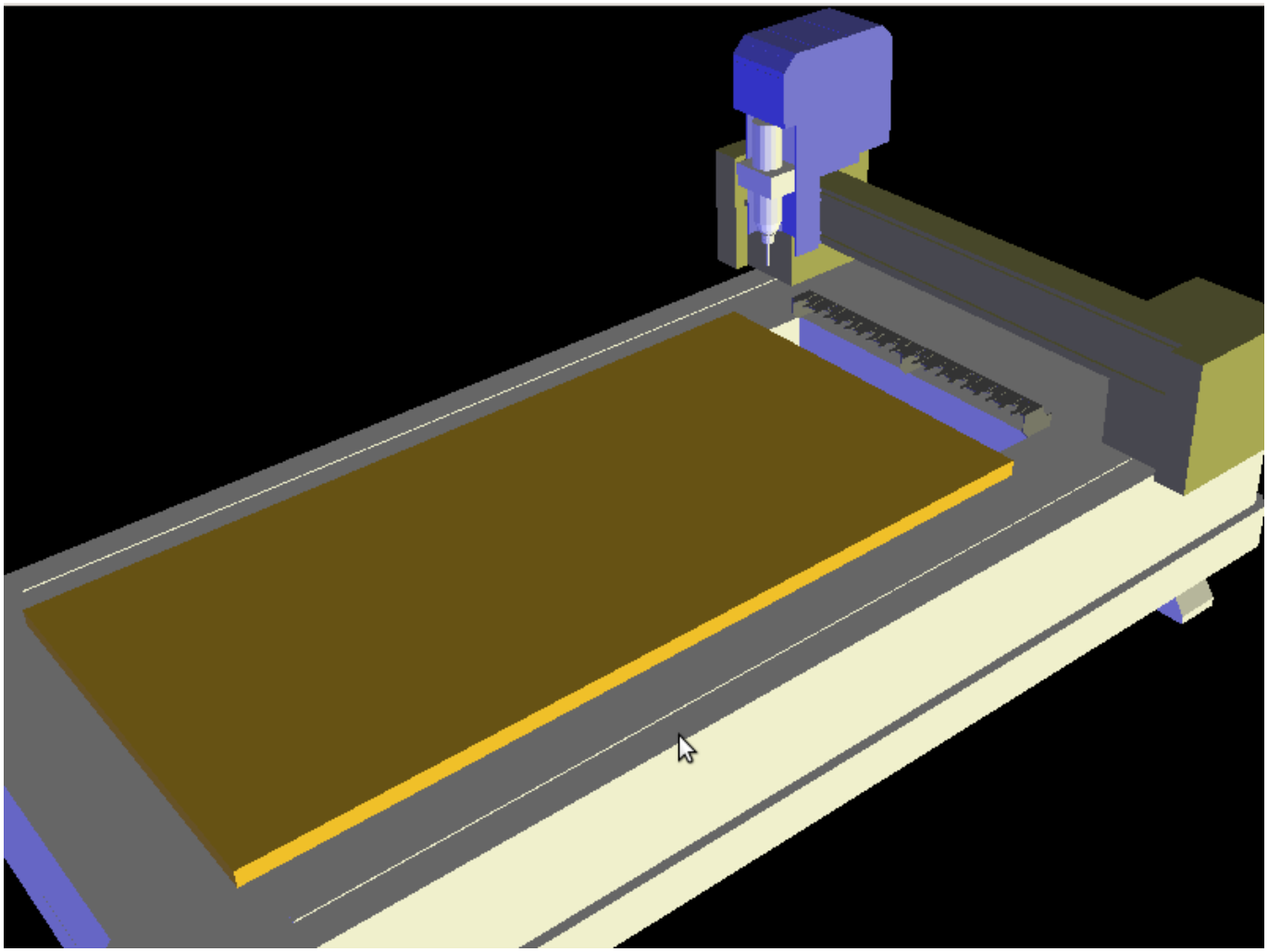


Figure 12.78: QtVCP `vismach_router_atc` - Панель 3D-вигляду 3-осьового портального фрезерного верстата

12.6.2.3 QtVCP `vismach_scara`

3D OpenGL-вигляд фрезерного верстата на базі SCARA.

```
loadusr qtvcp vismach_scara
```

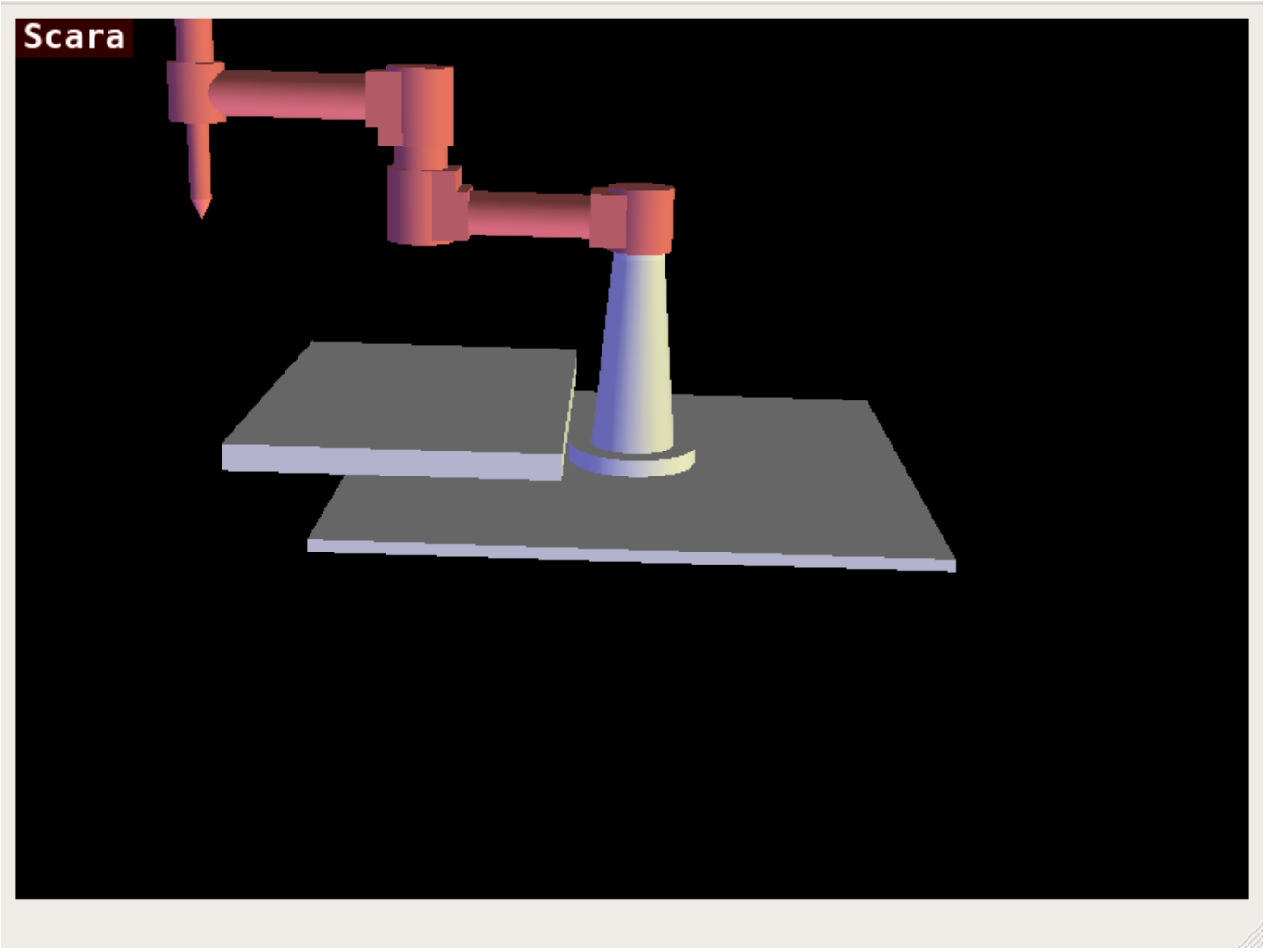


Figure 12.79: QtVCP `vismach_scara` - Панель 3D-вигляду SCARA Mill

12.6.2.4 QtVCP `vismach_millturn`

3D-зображення OpenGL *3-осьового фрезерного верстата з віссю A/шпинделем*.

```
loadusr qtvcp vismach_millturn
```

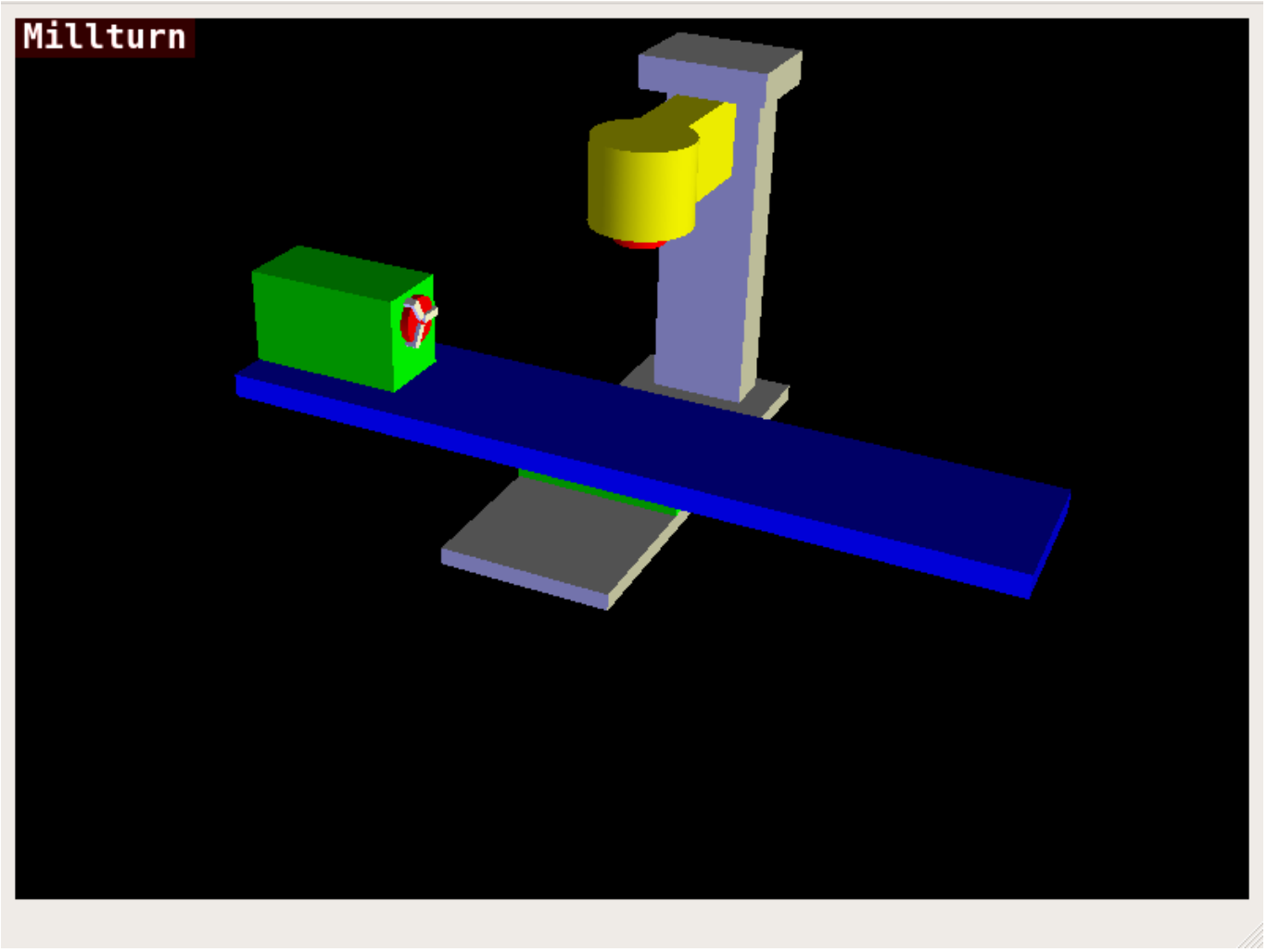


Figure 12.80: QtVCP `vismach_millturn` - Панель 3D-вигляду 4-осьового фрезерування/токарного верстата

12.6.2.5 QtVCP `vismach_mill_5axis_gantry`

3D OpenGL-вигляд *5-осьового портального фрезерного верстата*.

```
loadusr qtvcp vismach_mill_5axis_gantry
```

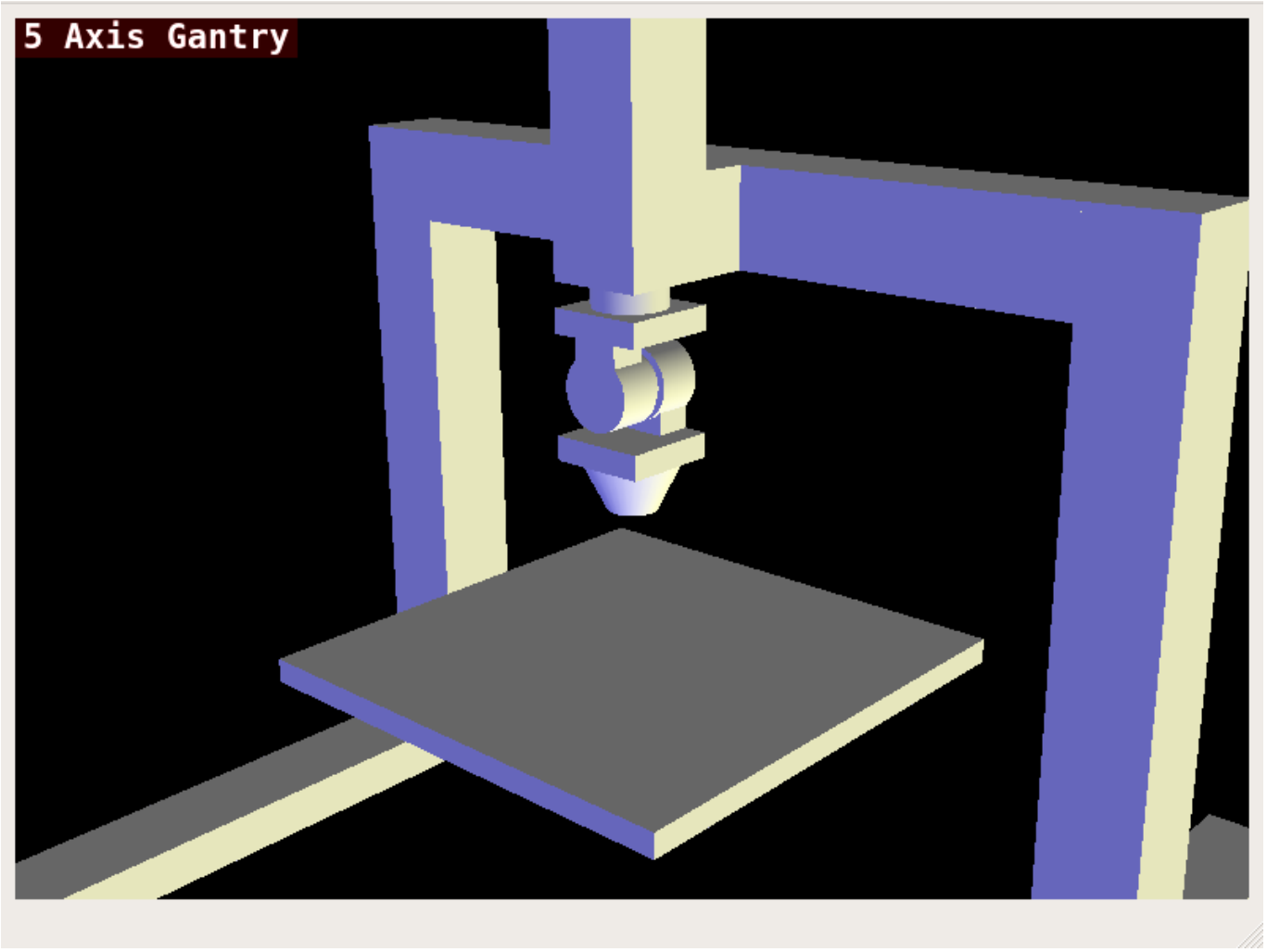


Figure 12.81: QtVCP `vismach_mill_5axis_gantry` - Панель 3D-вигляду 5-осьового портального фрезерного верстата

12.6.2.6 QtVCP `vismach_fanuc_200f`

3D-вигляд *6-суглобової роботизованої руки* у форматі OpenGL.

```
loadusr qtvcp vismach_fanuc_200f
```

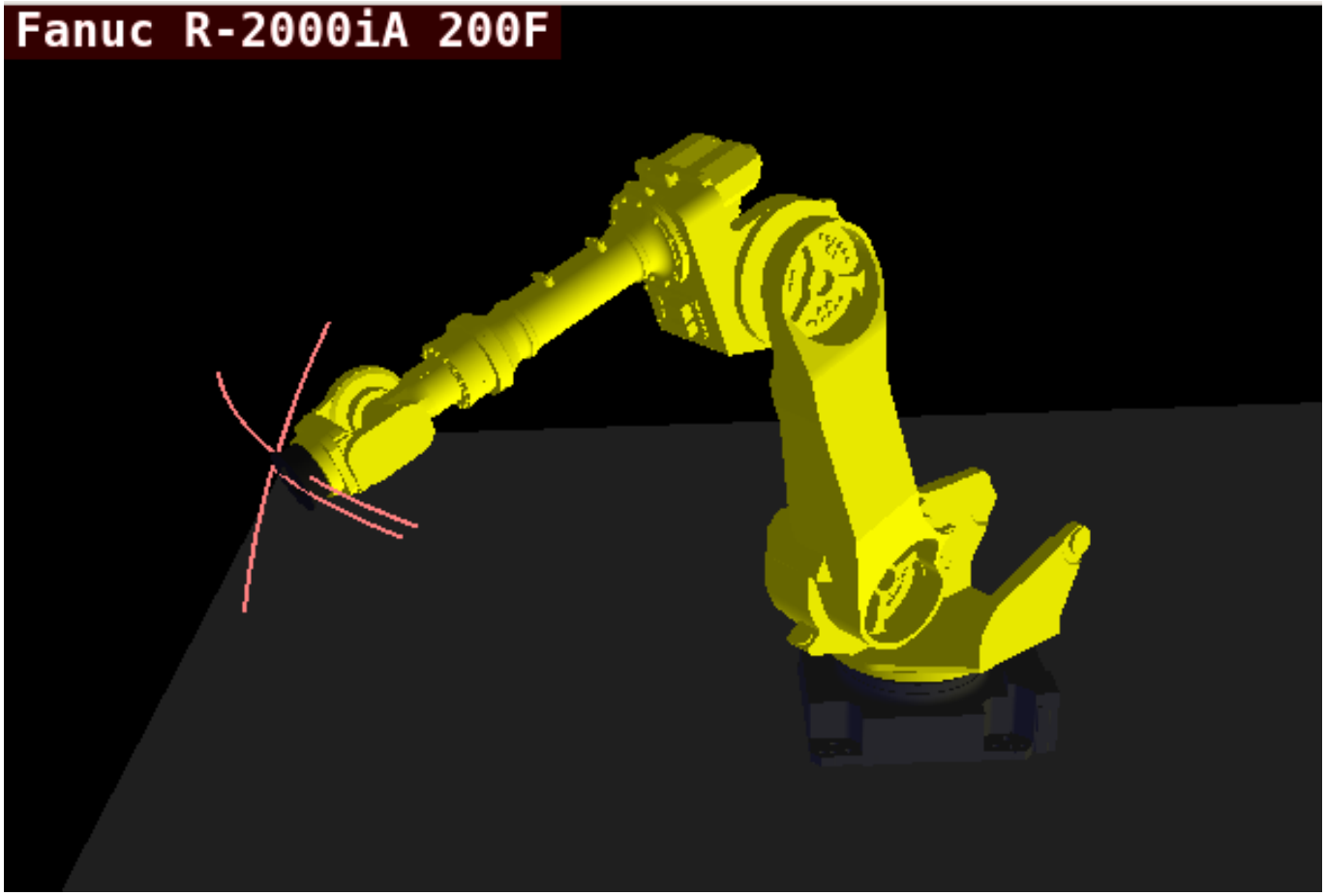


Figure 12.82: QtVCP `vismach_fanuc_200f` - 6-суглобова роботизована рука

12.6.3 Користувацькі віртуальні панелі керування

Звісно, ви можете **створити власну панель та завантажити її**.

Якщо ви створили файл інтерфейсу користувача з назвою `my_panel.ui` та файл HAL з назвою `my_panel.hal`, ви завантажили б їх з терміналу за допомогою:

```
halrun -I -f my_panel.hal
```

Приклад HAL-файлу, що завантажує панель QtVCP

```
# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'кb''b''ob''b''мb''b''пb''b''ob''b''нb''b''eb''b''нb''b''тb''b''іb''b''вb'' b''рb''b' ↩
    'eb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b''чb''b''ab''b''cb''b''уb''
loadrt threads
loadrt classicladder_rt

# b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b''ab''b''жb''b''уb''b''вb''b''ab''b''тb''b' ↩
    'иb'' b''пb''b''рb''b''ob''b''гb''b''рb''b''ab''b''мb''b''иb'', b''щb''b''ob'' b''нb''b' ↩
    'eb'' b''пb''b''рb''b''ab''b''цb''b''юb''b''юb''b''тb''b''ьb'' b''уb'' b''рb''b''eb''b' ↩
    'ab''b''лb''b''ьb''b''нb''b''ob''b''мb''b''уb'' b''чb''b''ab''b''cb''b''іb''
loadusr classicladder
loadusr -Wn my_panel qtvcp my_panel.ui  # <t>\coref{1}{CO2-3}</t>

# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''иb'' b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b''уb''
```

```
addf classicladder.0.refresh thread1


# b''зb''b''єb''b''дb''b''нb''b''аb''b''тb''b''иb'' b''кb''b''оb''b''нb''b''тb''b''аb''b' ↩
    'кb''b''тb''b''иb''
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

# b''пb''b''оb''b''чb''b''аb''b''тb''b''иb'' b''тb''b''еb''b''мb''b''уb''
start
```

(1), (1) У цьому випадку ми завантажуємо `qtvcp` за допомогою **`-Wn`**, що очікує завершення завантаження панелі перед продовженням виконання наступної команди HAL.
Це робиться для того, щоб *переконатися, що створені панеллю контакти HAL дійсно готові*, якщо вони використовуються в решті файлу.

12.6.4 Вбудовування віртуальних панелей керування QtVCP в екрани QtVCP

Панелі QtVCP можна вбудовувати в більшість екранів QtVCP, що дозволяє уникнути таких проблем, як переміщення фокуса, що може бути проблемою при вбудовуванні в ненативні середовища.

12.6.4.1 Команди вбудовування

Типовий екран, такий як QtDragon, шукатиме команди для вбудовування панелі у файлі INI під заголовком [DISPLAY].

```
[DISPLAY]
EMBED_TAB_NAME=Embedding demo
EMBED_TAB_COMMAND=qtvcp simple_hal
EMBED_TAB_LOCATION=tabWidget_utilities
```

EMBED_TAB_NAME
: зазвичай буде заголовком вкладки.

EMBED_TAB_LOCATION
: буде специфічним для екрана та визначатиме tabWidget або stackedWidget для вбудовування.

EMBED_TAB_COMMAND
: - це команда, яка використовується для завантаження панелі. Для вбудованих панелей першим словом завжди буде «qtvcp», а останнім - назва панелі, яку потрібно завантажити. Ви також можете передати параметри панелі за допомогою перемикачів -o в командному рядку між «qtvcp» і назвою панелі. Панель буде слідувати налаштуванням режиму налагодження головного екрану.

12.6.4.2 Розташування вбудованих панелей

Існують панелі, які входять до складу LinuxCNC. Щоб переглянути список, відкрийте термінал, введіть «qtvcp» і натисніть клавішу «Return».
Ви отримаєте довідку та список вбудованих екранів і панелей.
Виберіть будь-яке ім’я зі списку панелей і додайте його до запису COMMAND після «qtvcp».

Шлях пошуку вбудованої панелі — «share/qtvcp/panels/PANELNAME».
У версіях LinuxCNC, що запускаються на місці, та встановлених версіях вони знаходяться в різних місцях системи.

12.6.4.3 Розташування користувацьких панелей

Також можна вбудовувати власні панелі — модифіковані вбудовані панелі або нові панелі, створені користувачем.
Під час завантаження панелей QtVCP шукає в папках конфігурації шлях до «qtvcp/panels/PANELNAME/PANELNAME.ui».
«PANNELNAME» — це будь-який дійсний рядок без пробілів. Якщо шлях не знайдено, то шукається вбудований шлях до файлу.
QtVCP виконає той самий процес для опціонального файлу обробника: «qtvcp/panels/PANELNAME/PANELNAME_handler.py»

12.6.4.4 Поради щодо програмування обробників

У файлі обробника екрану посилання, яке використовується для вікна, є «self.w».
У панелях QtVCP це посилання буде посилатися на вікно панелі.
Щоб посилатися на головне вікно, використовуйте «self.w.MAIN». Якщо ваша панель має працювати незалежно та бути вбудованою, ви повинні перехоплювати помилки від посилань на об'єкти, які недоступні. (Зверніть увагу, що об'єкти головного екрану недоступні в незалежній панелі.)

Наприклад, це використовуватиме файл налаштувань панелі, якщо такий є.

```
try:
    belt_en = self.w.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

Це використовуватиме файл налаштувань головного екрана, якщо такий є.

```
try:
    belt_en = self.w.MAIN.PREFS_.getpref('Front_Belt_enabled', 1, int, 'SPINDLE_EXTRAS')
except:
    belt_en = 1
```

12.6.4.5 Поради щодо дизайнерських віджетів

Під час використання параметра команди Python у віджетах кнопок дій вбудованої панелі:

ПРИМЕР
стосується вікна панелі. Наприклад, `INSTANCE.my_panel_handler_function_call(True)`

MAIN_INSTANCE
посилається на головне вікно екрана. Наприклад, `MAIN_INSTANCE.my_main_screen_handler_funct`

Якщо панель не вбудована, обидва посилаються на вікно панелі.

12.6.4.6 Патчування обробників - створення підкласів вбудованих панелей

Ми можемо змусити QtVCP завантажити підкласову версію стандартного файлу обробника. У цьому файлі ми можемо маніпулювати оригінальними функціями або додавати нові.
Підкласування означає, що наш файл обробника спочатку завантажує оригінальний файл обробника і додає до нього наш новий код - фактично патч змін.
Це корисно для зміни/додавання поведінки, зберігаючи при цьому стандартні оновлення обробника з репозиторіїв LinuxCNC.

Можливо, вам все одно доведеться скористатися діалоговим вікном копіювання обробника, щоб скопіювати оригінальний файл обробника та вирішити, як його виправити.

У папці config повинна бути папка для панелі з назвою «<CONFIG FOLDER>/qtvcp/panels/<PANEL NAME>/».
Додайте туди файл патча з назвою <ORIGINAL PANEL NAME>_handler.py,
тобто для cam_align файл буде називатися «cam_align_handler.py».

Ось приклад зміни кольору кола в cam_align:

```
import sys
import os
import importlib
from PyQt5.QtCore import Qt
from qtvcp.core import Path

PATH = Path()

# b''ob''b''тb''b''pb''b''иb''b''мb''b''ab''b''тb''b''иb'' b''пb''b''ob''b''cb''b''иb''b' ↩
    'лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b''ob''b''pb''b''иb''b''гb''b''іb''b' ↩
    'нb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''йb'' b''фb''b''ab''b''йb''b''лb'' b''ob''b' ↩
    'бb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab'', b''щb''b''ob''b''бb'' b''мb''b' ↩
    'иb'' b''мb''b''ob''b''гb''b''лb''b''иb'' b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b' ↩
    'иb'' b''йb''b''ob''b''гb''b''ob'' b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb''b' ↩
    'ob''b''мb''
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

# b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''yb''b''тb''b''иb'' b''нb''b''ab''b''шb'' b' ↩
    'ob''b''бb'''b''єb''b''кb''b''тb''-b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''иb''b' ↩
    'кb'' b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb''b''yb'' b''дb''b''ob'' Qtvcp
def get_handlers(halcomp, widgets, paths):
    return [UserHandlerClass(halcomp, widgets, paths)]

# b''пb''b''іb''b''дb''b''кb''b''лb''b''ab''b''cb'' HandlerClass, b''іb''b''мb''b''пb''b' ↩
    'ob''b''pb''b''тb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb'' b''вb''b''иb''b''щb''b' ↩
    'eb''
class UserHandlerClass(HandlerClass):
    print('Custom subclassed panel handler loaded\n')

    def initialized__(self):
        # call original handler initialized function
        super().initialized__()

        # b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''нb''b''ab''b''шb''b''yb'' b''нb''b' ↩
            'ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''
        self.w.camview.circle_color = Qt.green
```

12.7 Віджети QtVCP

`Qtscreen` використовує *QtVCP віджети* для інтеграції з LinuxCNC.

Віджет — це загальна назва для *об'єктів інтерфейсу*, таких як кнопки та мітки в PyQt.

Ви можете вільно використовувати будь-які доступні **віджети за замовчуванням** у редакторі *Qt Designer*.

Також існують **спеціальні віджети**, створені для LinuxCNC, які спрощують інтеграцію. Вони розділені на дві частини, розташовані у правій частині редактора:

- Один призначений для **віджетів лише HAL**.
- Інший призначений для **віджетів керування CNC**.

Ви можете вільно змішувати їх будь-яким чином на своїй панелі.

Note
Цей опис властивостей віджетів може легко застаріти через подальший розвиток та брак людей, які б писали документацію (хороший спосіб віддячити проекту). Остаточні описи можна знайти в source code.

12.7.1 Віджети лише для HAL

Ці віджети зазвичай мають *HAL-піни* та **не реагують на контролер машини**.

12.7.1.1 `CheckBox` Віджет

Цей віджет дозволяє користувачеві **встановити прапорець, щоб встановити значення HAL pin у значення «true» або «false»**.

Він базується на *QCheckButton* PyQt.

12.7.1.2 *DetachTabWidget* - Віджет контейнера зі знімними користувачем панелями

Цей контейнерний віджет працює так само, як QTabWidget — він відображає кілька панелей по черзі з вкладками для вибору.
Якщо двічі клацнути вкладку або перетягнути її, вона від'єднається від головного вікна.
Коли вкладка від'єднана, її вміст розміщується в QDialog.
Вкладку можна знову приєднати, закривши діалогове вікно або двічі клацнувши на її рамці.

Він базується на QTabWidget від PyQt.

12.7.1.3 `DoubleScale` - Віджет введення кнопки спінінга

Цей віджет — віджет **введення за допомогою кнопки обертання**, який використовується для *встановлення s32 та float HAL-піну*.

Він має внутрішній *коефіцієнт масштабування*, встановлений за замовчуванням на 1, який можна встановити програмно або за допомогою QtSignal.

Слот `setInput` може бути підключений до цілочисельного або плаваючого сигналу.

`[HALLabelName].setInput(some_value)`
Це виклик функції для зміни внутрішнього коефіцієнта масштабування.

Виводи HAL будуть встановлені на значення *внутрішньої шкали, помножене на значення, що відображається віджетом*.

12.7.1.4 `FocusOverlay` - Віджет накладання фокуса

Цей віджет розміщує **кольорове накладання на екран**, зазвичай під час відображення діалогового вікна.

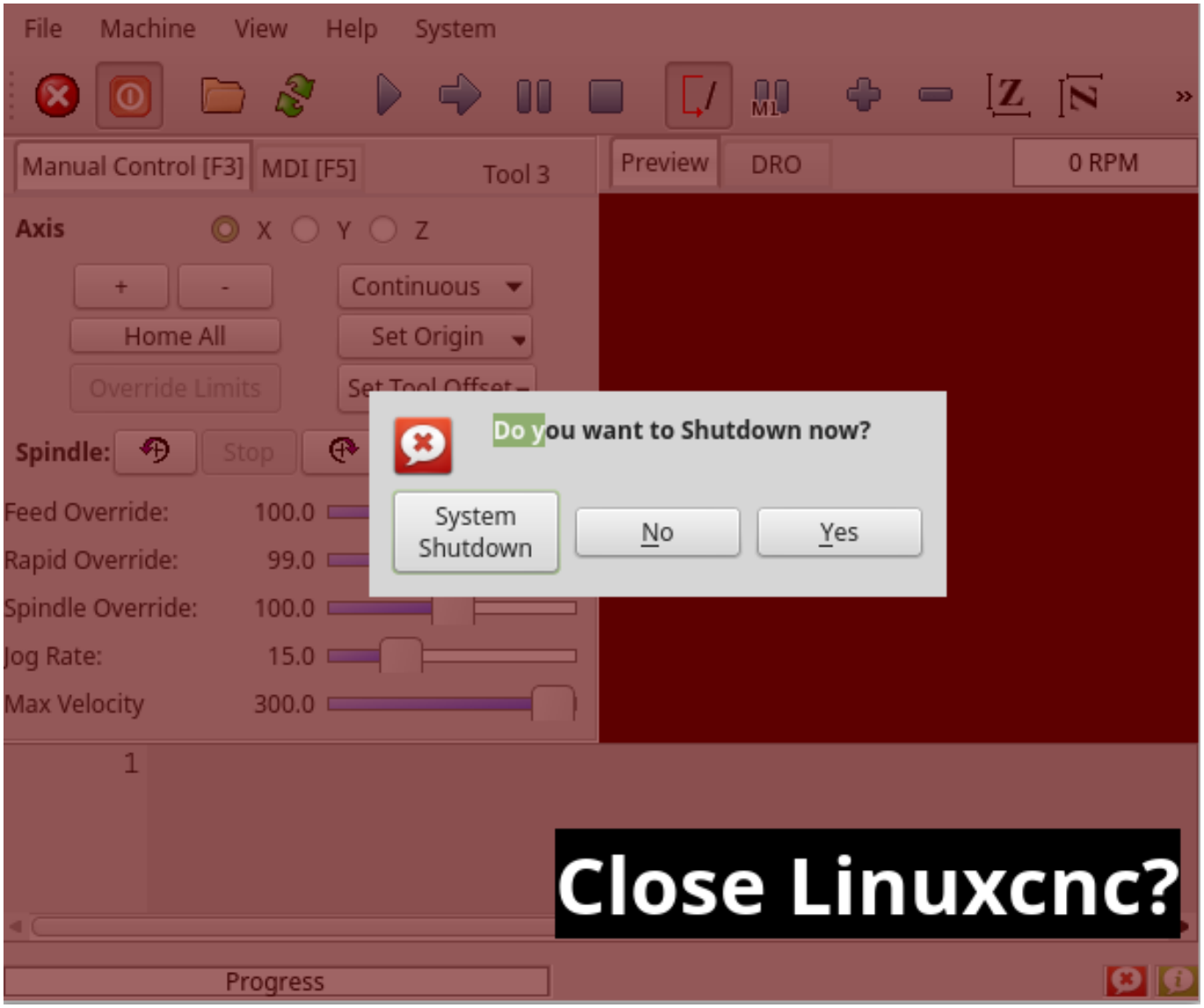


Figure 12.83: Приклад накладання фокуса для підтвердження закриття

Використовується для створення відчуття «зосередженості» та привернення уваги до важливої інформації.

Він також може відображати напівпрозоре зображення.
Він також може відображати текст повідомлення та кнопки.

Цим віджетом _можна керувати за допомогою повідомлень `STATUS`.

12.7.1.5 `Gauge` - Віджет круглого циферблатного індикатора

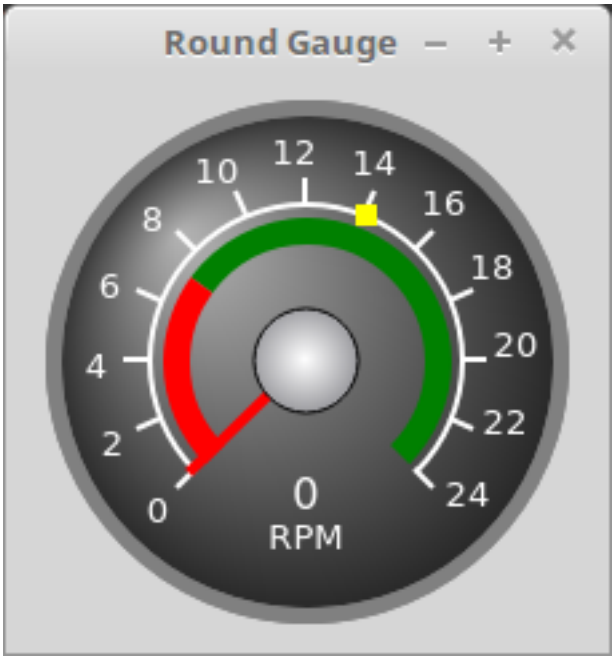


Figure 12.84: QtVCP `Gauge`: Віджет круглого циферблатного індикатора

Круглий калібр можна використовувати в графічному інтерфейсі LinuxCNC для **відображення вхідного параметра** на циферблаті.

Налаштовувані параметри Існує кілька властивостей, які користувач може налаштувати, щоб налаштувати *зовнішній вигляд датчика*.

Наступні параметри можна встановити або програмно, або через редактор властивостей Qt Designer.

`halpin_option`
Встановлення цього значення на `True` *створить 2 виводи HAL*:

- Один призначений для встановлення вхідного значення `value`
- Інший призначений для встановлення «заданої точки».

Якщо цей параметр не встановлено, то `value` та `setpoint` мають бути пов’язані програмно, тобто у файлі обробника.

`max_reading`
Це значення визначає *найбільше число, що відображається* на поверхні калібру.

`max_value`
Це *максимальне очікуване значення вхідного сигналу*.
Іншими словами, це повний діапазон вхідних значень.

`num_ticks`
Це *кількість позначок/показників індикатора* на циферблаті індикатора.
Слід встановити число, яке забезпечує читабельність текстових показників навколо циферблата індикатора.
Мінімальне дозволене значення – 2.

zone1_color
Зона 1 простягається *від максимального значення до порогової точки*.
Її можна встановити на будь-який колір RGB.

zone2_color
Зона2 простягається *від порогової точки до мінімального значення*, яке дорівнює 0.
Їй можна встановити будь-який колір RGB.

bezel_color
Це колір *зовнішнього кільця калібру*.

bezel_width
Це ширина *зовнішнього кільця калібру*.

поріг
Поріг є *точкою переходу між зонами*.
Його слід встановити на значення від 0 до максимального значення.
Максимально допустиме значення встановлюється на `max_value` датчика, а мінімальне значення дорівнює 0.

gauge_label
Це *текст під зчитуванням значення*, біля нижньої частини індикатора.
Тоді функція індикатора легко видима.

base_color
Колір калібру.

base_gradient_color
Колір підсвічування датчика.

center_color
Колір центру калібру.

center_gradient_color
Колір підсвічування центру датчика.

Неналаштовувані параметри Є 2 входи, які не можна налаштувати. Їх можна встановити через виводи HAL, програмно або за допомогою сигналів з інших віджетів:

значення
Це *фактичне вхідне значення*, яке буде відображатися разом зі стрілкою манометра та на цифровому індикаторі.
Його необхідно встановити на значення від 0 до максимального значення `max_value`.

задане значення
Це значення, яке визначає розташування маленького *маркера* на поверхні калібру. Його необхідно встановити на значення від 0 до максимального значення.

12.7.1.6 `GeneralHALInput` - Віджет підключення вхідних сигналів/слотів

Цей віджет використовується для **підключення довільного віджета Qt до HAL за допомогою сигналів/слотів**.

Використовується *для віджетів, які повинні **реагувати** на зміни пінів HAL*.

12.7.1.7 `GeneralHALOutput` - Загальні сигнали/слоти Віджет підключення виходу

Цей віджет використовується для **підключення довільного віджета Qt до HAL за допомогою сигналів/слотів**.

Він використовується *для віджетів, які повинні **керувати** пінами HAL*.

12.7.1.8 `GridLayout` - Віджет макета сітки

Цей віджет **керує тим, чи віджети всередині нього ввімкнені чи вимкнені**.

Вимкнені віджети зазвичай мають інший колір і не реагують на дії.

Він базується на `QGridLayout` від PyQt.

12.7.1.9 *HalBar* - Індикатор рівня HAL Bar

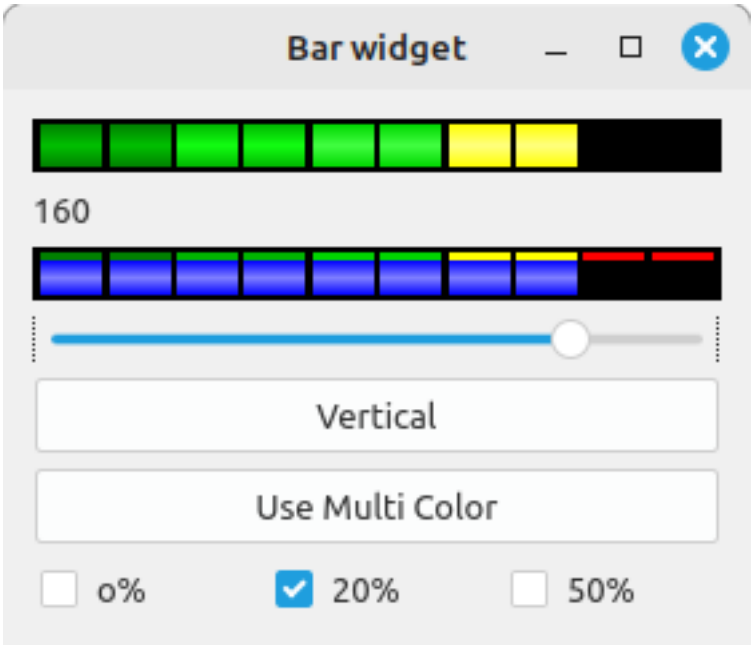


Figure 12.85: QtVCP `HalBar`: панель, що демонструє індикатор рівня HAL Bar

Цей віджет використовується для індикації рівня або значення, зазвичай для виводу HAL s32/float. Ви також можете вимкнути вивод HAL та використовувати сигнали Qt або команди Python для зміни рівня.

HalBar є підкласом віджета Bar, тому він успадковує такі властивості:

- *stepColorList*: список кольорових рядків, кількість кольорів визначає кількість стовпців.
- *backgroundColor*: визначення QColor кольору фону.
- *indicatorColor*: визначення QColor для додаткової одноколірної шкали поточного значення.
- *useMultiColorIndicator*: Булевий перемикач для вибору опції одноколірної або багатоколірної шкали значень.
- *split*: цілочисельний відсотковий поділ смужки максимального значення відносно смужки поточного значення (від 0 до 50%).
- *setVertical*: перемикач bool для вибору вертикального або горизонтального індикатора.
- *setInverted*: логічний перемикач для вибору інвертованого напрямку.

- *setMaximum*: ціле число, яке визначає максимальний рівень індикації.
- *setMinimum*: ціле число, яке визначає найнижчий рівень індикації.
- *pinType*: щоб вибрати **тип контактів HAL**:
  - `NONE` PIN-код HAL не буде додано
  - `S32` Буде додано цілочисельний пін S32
  - `FLOAT` Буде додано плаваючий штифт
- *pinName*: щоб змінити **назву виводу HAL**, інакше використовується базова назва віджета.

Вищезазначені властивості Bar можна встановити в *таблицях стилів*.
Властивості pinType та pinName не можна змінювати в таблицях стилів.

Note
У таблицях стилів stepColorList — це один рядок назв кольорів, розділених комами.

```
HalBar{
    qproperty-backgroundColor: #000;
    qproperty-stepColorList: 'green,green,#00b600,#00b600,#00d600,#00d600,yellow,yellow,red ↩
        ,red';
}
```

12.7.1.10 `HALPad` - Кнопки HAL Джойстик

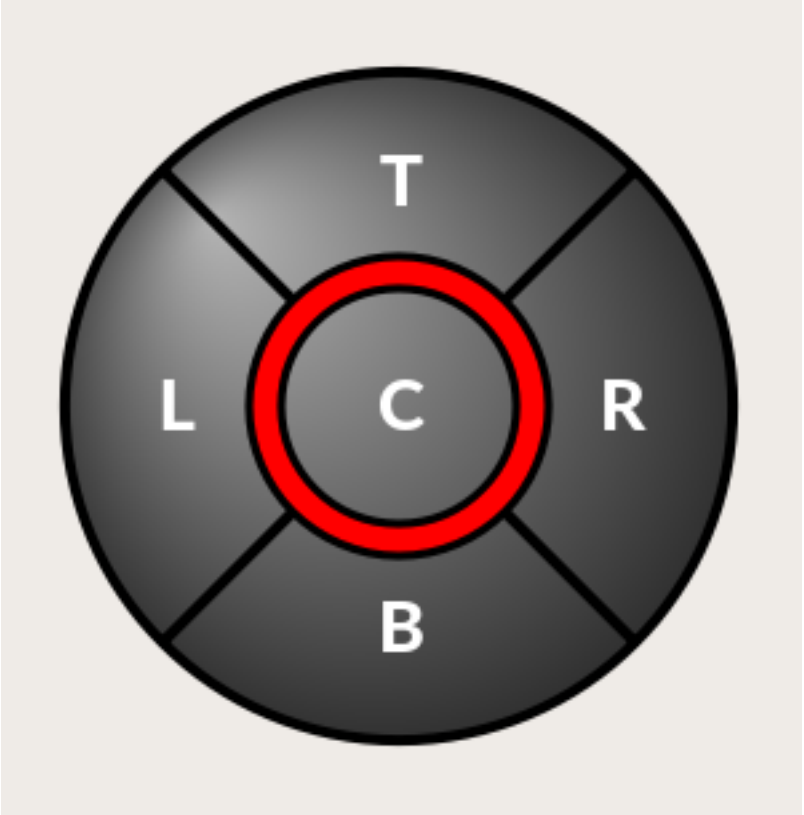


Figure 12.86: QtVCP `HALPad`: Кнопки HAL Джойстик

Цей віджет виглядає та працює як **5-кнопковий D-pad** зі світлодіодним кільцем.

Кожна кнопка має вивід HAL одного типу (біт, S32 або число з плаваючою комою), який можна вибрати.

Центральне кільце світлодіода має вибір кольорів для ввімкнення та вимкнення та керується бітовим виводом HAL.

HALPad ENUMS Використовуються *перелічені константи*:

- Щоб посилатися на **положення індикаторів**:
  - НІЧОГО
  - ЛІВОРУЧ
  - ПРАВОРУЧ
  - ЦЕНТР
  - ВЕРХ
  - ВНИЗ
  - ЛІВОРУЧПРАВОРУЧ
  - ЗВЕРХВНИЗ
- Для **типів контактів HAL**:
  - НІЧОГО
  - БІТ
  - S32
  - FLOAT

Ви використовуєте назву віджета в Qt Designer плюс константу посилання:

```
self.w.halpadname.set_highlight(self.w.halpadname.LEFTRIGHT)
```

HALPad Властивості

pin_name
Необов’язкова назва для *базової назви пінів HAL*. Якщо залишити поле порожнім, буде використано назву віджета Qt Designer.

pin_type
Виберіть *тип вихідного виводу HAL*. Ця властивість використовується лише під час запуску. Вибір можна встановити в Qt Designer:

- НІЧОГО
- БІТ
- S32
- FLOAT

left_image_path , right_image_path , center_image_path , top_image_path , bottom_image_path

Шлях до файлу або ресурсу зображення, яке потрібно відобразити в описаному місці розташуванн
кнопки.
Якщо натиснути кнопку скидання у властивості редактора Qt Designer, зображення не відображат
(дозволяючи додатковий текст).

left_text , right_text , center_text , top_text , bottom_text
Текстовий рядок, який буде відображено в описаному місці кнопки.
Якщо залишити поле порожнім, можна призначити зображення для відображення.

true_color , false_color
Вибір кольору для центрального світлодіодного кільця, яке відображатиметься, коли вивід HAL `<BASENAME>.light.center` має значення `True` або `False`.

text_color
Вибір кольору для тексту кнопки.

text_font
Вибір шрифту для тексту кнопки.

HALPad Стилі Вищезазначені властивості можна встановити в *стилях*.

```
HALPad{
    qproperty-on_color: #000;
    qproperty-off_color: #444;
}
```

12.7.1.11 `HALLabel` - Віджет міток HAL

Цей віджет **відображає значення, надіслані йому**.

Значення можна надсилати з:

- *HAL-піни*
  Вхідний пін можна вибрати як бітовий, S32, Float або взагалі не вибрати пін
- *Програмно*
- *A `QtSignal`*

Існує властивість `textTemplate` для встановлення форматованого тексту та/або форматування тексту.
Базове форматування може бути:

- `%r` для логічних значень
- `%d` для цілих чисел
- `%0.4f` для чисел з плаваючою комою.

Приклад форматованого тексту може бути таким:

```
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>
"""
)
```

Слот `setDisplay` може бути підключений до цілочисельного, числа з плаваючою комою або булевого сигналу.

Якщо властивість `pin_name` не встановлена, буде використано назву віджета.

Існують виклики функцій для відображення значень:

[HALLabelName].setDisplay(some_value)
Може використовуватися для налаштування дисплея, якщо не вибрано жодного контакту HAL.

[HALLabelName].setProperty(textTemplate,”%d”)
Встановлює шаблон відображення.

Він базується на *QLabel* PyQt.

12.7.1.12 `LCDNumber` - Віджет зчитування номерів у стилі РК-дисплея

Цей віджет *відображає значення HAL з плаваючою комою/s32/біт у вигляді РК-дисплея*.

Він може відображати числа у десятковому, шістнадцятковому, двійковому та вісімковому форматах, встановивши властивість **mode**.

Під час використання чисел з плаваючою комою ви можете встановити рядок форматування.

Ви повинні встановити властивість **digitCount** у відповідне значення, щоб відобразити найбільше число.

Властивості

pin_name
Рядок опцій, який буде використовуватися як назва виводу HAL.
Якщо встановлено значення порожнього рядка, буде використано назву віджета.

bit_pin_type
Вибирає вхідний контакт типу BIT.

s32_pin_type
Вибирає вхідний контакт типу S32.

float_pin_type
Виберіть тип вхідного контакту `FLOAT`.

floatTemplate
Строка, яка буде використовуватися як шаблон формату Python3 для налаштування LCD-дисплея.
Використовується тільки при виборі виводу `FLOAT`, наприклад, `{:.2f}` відображатиме число з плаваючою комою, округлене до 2 цифр після десяткової крапки.
Порожнє поле дозволить десятковій крапці переміщатися за необхідністю.

Він базується на *QLCDNumber* PyQt.

12.7.1.13 `LED` - Віджет індикатора



Figure 12.87: QtVCP `LED`: Віджет світлодіодного індикатора

Індикатор, подібний до світлодіода, який за бажанням відповідає логіці виводу HAL.

halpin_option
Вибирає, чи світлодіод відображає стан вхідного виводу HAL або програми.

diameter
Діаметр світлодіода (за замовчуванням 15).

color
Колір світлодіода, коли він увімкнений (за замовчуванням зелений).

off_color
Колір світлодіода у вимкненому стані (за замовчуванням чорний).

gradient
вмикає або вимикає градієнтне підсвічування (за замовчуванням увімкнено).

on_gradient_color
Колір підсвічування світлодіода, коли він увімкнений (за замовчуванням білий).

off_gradient_color
Колір підсвічування світлодіода, коли він вимкнений (за замовчуванням білий).

вирівнювання
Підказка щодо вирівнювання Qt.

стан
Поточний стан світлодіода

миготливий
Вмикає та вимикає опцію миготіння.

Швидкість спалаху
Встановлює частоту спалахів.

Властивості LED можна визначити в *stylesheet* за допомогою наступного коду, доданого до файлу `.qss`, де `name_of_led` – це назва віджета, визначена в редакторі Qt Designer:

```
LED #name_0f_led{
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

12.7.1.14 `PushButton` - Віджет перемикання закріплення HAL

Цей віджет дозволяє користувачеві **встановити значення HAL pin на true або false** одним натисканням кнопки.

Як варіант, це може бути *кнопка-перемикач*.

Щоб дізнатися більше про *опцію світлодіодного індикатора*, див Section 12.7.5.1[IndicatedPushButton] нижче.

Також має інші опції.

Він базується на *QPushButton* PyQt.

12.7.1.15 `RadioButton` Віджет

Цей віджет дозволяє користувачеві **встановлювати значення пінів HAL на значення true або false**. Лише один віджет `RadioButton` групи може мати значення true одночасно.

Він базується на *QRadioButton* від PyQt.

12.7.1.16 `Slider` - Віджет налаштування значення HAL Pin

Дозволяє **налаштувати значення виводу HAL за допомогою ковзного вказівника**.

12.7.1.17 `TabWidget` - Віджет вкладки

Цей віджет дозволяє налаштовувати висоту вкладок за допомогою таблиць стилів.

Властивості `TabWidget` можна визначити в *stylesheet*, додавши наступний код до файлу `.qss`.
`name_of_tab` — це ім'я віджета, визначене в редакторі Qt Designer.
Якщо ви пропустите текст «#name_of_tab», буде встановлено висоту всіх вкладок TabWidgets.
Це показує, як встановити висоту вкладки певного віджета:

```
TabWidget #name_of_tab{
  qproperty-tabsize: 1.5;
}
```

Він базується на QTabWidget від PyQt.

12.7.1.18 `WidgetSwitcher` - Віджет перемикача режимів перегляду макета кількох віджетів

Це використовується для перемикання вигляду макета з кількома віджетами на відображення лише одного віджета, тобто для **перемикання між великим виглядом віджета та меншим виглядом кількох віджетів**.

Він *відрізняється від стекового віджета* тим, що може витягувати віджет з будь-якої точки екрана та розміщувати його на своїй сторінці з іншим макетом, ніж той, що був спочатку.

Щоб перемикач міг повернути *оригінальний віджет, він має бути в макеті*.

У Qt Designer ви будете:

- Додайте на екран віджет `WidgetSwitcher`.
- Клацніть правою кнопкою миші на `WidgetSwitcher` та додайте сторінку.
- Заповніть його віджетами/макетами, які ви хочете бачити у формі за замовчуванням.
- Додайте стільки сторінок, скільки є подань для переходу.
- На кожній сторінці додайте віджет макета.
  Після додавання макета потрібно знову клацнути правою кнопкою миші на перемикачі віджетів і встановити параметр макета.
- Клацніть на віджеті `WidgetSwitcher`, а потім прокрутіть униз редактора властивостей.
- Знайдіть динамічну властивість `widget_list` та двічі клацніть праворуч від неї.
- З'явиться діалогове вікно, яке дозволить вам додати назви віджетів для переміщення на сторінки, додані до `WidgetSwitcher`.

Існують *виклики функцій* для відображення певних віджетів.
Викликаючи одну з цих функцій, ви контролюєте, який віджет відображається на даний момент:

`[_WidgetSwitcherName_].show_id_widget(_number_)`, `[_WidgetSwitcherName_].show_named_widget(`

Це показує макет `сторінка 0`, а всі інші віджети повертаються до початкового стану, коли вони були створені в Qt Designer.

`[_WidgetSwitcherName_].show_next()`
Показати наступний віджет.

Він базується на віджеті *QStack*.

12.7.1.19 `XEmbed` - Віджет вбудовування програм

Дозволяє **вбудувати програму у віджет**.

Працюватимуть лише програми, що використовують протокол `xembed`, такі як:

- Віртуальні панелі керування GladeVCP
- Вбудована віртуальна клавіатура
- Віртуальні панелі керування QtVCP
- відеоплеєр mplayer

12.7.2 Віджети контролера машини

Ці віджети **взаємодіють зі станом контролера машини**.

12.7.2.1 `ActionButton` - Віджет керування діями контролера машини

Ці кнопки використовуються для **керування діями на контролері машини**.

Вони побудовані на базі `IndicatedPushButton`, тому можуть мати світлодіоди, що накладаються один на одного.

Note
Якщо ви залишили подвійний клік на цьому віджеті, ви можете запустити діалогове вікно для налаштування будь-якої з цих дій. Діалогові вікна допоможуть налаштувати правильні дані, пов’язані з вибраною дією. Ви також можете змінити ці властивості безпосередньо в редакторі властивостей.

Дії Ви можете вибрати один із цих:

`Estop`, `Машина ввімкнена`, `Авто`, `mdi`, `ручний`, `run`, `run_from_line статус`
: Отримує номер рядка з повідомлення `STATUS gcode-line-selected`.

`run_from_line слот`
: Отримує номер рядка з Qt Designer int/str slot `setRunFromLine`.

`abort`, `pause`, `завантажити діалогове вікно`
: Потрібно наявність віджета діалогового вікна.

`Діалогове вікно перегляду камери`
: Потрібна наявність віджета діалогового вікна `camview`.

`діалогове вікно зміщення походження`
: Потрібно наявність віджета діалогового вікна зміщення походження.

`діалогове вікно макросу`
: Потрібна наявність віджета діалогового вікна макросів.

`Запуск Халметра`, `Стан запуску`, `Запуск Halshow`, `Головна сторінка`
: Встановіть номер суглоба на -1 для `all-home`.

`Ви бездомні`
: Встановіть номер суглоба на -1 для `all-unhome`.

Вибрано для дому
Встановлює опорну точку для з’єднання/вісі, вибраної за допомогою STATUS.

Вибрано
Знімає з початкового положення з’єднання/вісь, вибрану за допомогою STATUS.

нульова вісь , нуль G5X
Обнуляє поточні зміщення системи координат користувача.

нуль G92
Обнуляє додаткові зміщення G92.

нульовий поворотний момент Z
Обнуляє зміщення обертання.

позитивний суглоб підтюпцем
Встановіть номер суглоба.

поштовховий суглоб негативний
Встановіть номер суглоба.

пробіжка вибрана позитивною
Вибрано за допомогою іншого віджета або СТАТУСУ.

поштовх вибрано негативне
Вибрано за допомогою іншого віджета або СТАТУСУ.

приріст поштовху
Встановіть метричні/імперські/кутові числа.

швидкість штовхання
Встановіть число з плаваючою комою/альтернативне число з плаваючою комою.

перевизначення каналу
Встановіть число з плаваючою комою/альтернативне число з плаваючою комою.

швидке перевизначення
Встановіть число з плаваючою комою/альтернативне число з плаваючою комою.

корекція шпинделя
Встановіть число з плаваючою комою/альтернативне число з плаваючою комою.

шпиндель вперед , шпиндель назад , зупинка шпинделя , шпиндель вгору , шпиндель вниз , зміна пер

Набір view_type_string.

обмеження перевизначення , повінь , mist , видалення блоку , необов’язкова зупинка , команда mdi

Встановлює command_string, тобто викликає жорстко закодовану MDI-команду

ЦЕ номер MDI
Встановлює ini_mdi_number, тобто викликає MDI-команду на основі INI

дро абсолют , родич дро , dro dtg , екран виходу
Закриває LinuxCNC

Замінити обмеження
Тимчасово змінити жорсткі обмеження

діалогові вікна запуску
Відкриває діалогові вікна, якщо вони включені до файлу інтерфейсу користувача.

встановити DRO на відносний , встановити DRO на абсолютне значення , встановити DRO на відст

Атрибути Ці встановлюють *атрибути* вибраної дії (доступність залежить від віджета):

перемикання опції плаваючої точки
Дозволяє перемикатися між двома швидкостями за допомогою кнопок джогінгу та коригування.

спільний номер
Вибирає з'єднання/вісь, якою керує кнопка.

збільшення імперського числа
Встановлює приріст імперської шкали (встановіть від'ємне значення, щоб ігнорувати).

число мм
Встановлює приріст метричної похибки (встановіть від'ємне значення, щоб ігнорувати).

приріст кутового числа
Встановлює приріст кутового повороту (встановлює від'ємне значення для ігнорування).

число з комою
Використовується для `jograte` та перевизначень.

альтернативне число з плаваючою комою
Для `jograte` та перевизначень, які можуть перемикатися між двома числами з комою.

рядок типу перегляду
Може бути:

- `p`,
- `x`, `y`, `y2`, `z`, `z2`,
- `збільшення масштабу`, `зменшення масштабу`,
- `pan-up`, `pan-down`, `pan-left`, `pan-right`,
- `rotate-up`, `rotate-down`, `rotate-cw`, `rotate-ccw`
- `clear`.

рядок команди
Рядок команди MDI, який буде викликано, якщо вибрано дію команди MDI.

ini_mdi_number
(Старий спосіб)
Посилання на розділ *INI файлу* `[MDI_COMMAND_LIST]`.
Встановіть ціле число, вибравши один рядок під рядком INI `[MDI_COMMAND]`, починаючи з 0.
Потім у файлі INI під заголовком `[MDI_COMMAND_LIST]` додайте відповідні рядки.
Команди, розділені символом `;`, будуть виконуватися одна за одною.
Текст мітки кнопки можна встановити будь-яким текстом після коми, символ `\n` додає розрив рядка.

ini_mdi_key
(бажаний спосіб)
Посилання на розділ *INI файлу* `[MDI_COMMAND_LIST]`.
Цей рядок буде додано до *MDI_COMMAND_*, щоб сформувати запис для пошуку
у файлі INI під заголовком `[MDI_COMMAND_LIST]`.
Команди, розділені символом `;`, будуть виконуватися одна за одною
Текст мітки кнопки можна встановити будь-яким текстом після коми, символ `\n` додає розрив рядка.

```
[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
```

Кнопки дій є підкласами Section 12.7.5.1[IndicatedPushButton]. Див. наступні розділи для отримання додаткової інформації про:

- LED Indicator option
- Enabled on State
- Зміни тексту щодо стану
- Виклик команди Python у стані увімкнення

12.7.2.2 ActionToolButton - Віджет кнопки меню додаткових дій

Кнопки ActionToolButton за своєю концепцією схожі на кнопки дій, але вони використовують *QToolButtons*, щоб дозволити вибирати **опціональні дії** шляхом натискання і утримання кнопки до появи меню опцій. Наразі є лише один варіант: userView.

Він базується на *QToolButton* PyQt. **userView Віджет запису та налаштування перегляду користувача**

Кнопка інструмента «Вигляд користувача» дозволяє **записувати та повертатися до довільного графічного вигляду**.

Натисніть і утримуйте кнопку, щоб з'явилося меню, і натисніть *запис перегляду*, щоб записати поточний графічний вигляд.

Натисніть кнопку звичайним способом, щоб повернутися до останньої записаної позиції.

Записана позиція буде запам'ятана під час вимкнення, якщо налаштовано опцію файлу налаштувань.

Note
Через обмеження програмування записане положення може не відображатися точно так само. Особливо, якщо ви зменшили масштаб і знову збільшили його під час налаштування бажаного виду.
Найкращий спосіб — вибрати основний вид, змінити його за бажанням, записати, а потім одразу натиснути кнопку, щоб перейти до записаного положення. Якщо воно вам не подобається, змініть його поточне положення і запишіть знову.

12.7.2.3 AxisToolButton - Віджет «Вибрати та встановити осі»

Це дозволяє **вибрати та встановити вісь**.

Якщо кнопку встановлено як прапорець, вона вказуватиме на вибрану вісь.

Якщо натиснути та утримувати кнопку, з'явиться спливаюче меню, яке дозволить:

- Обнулення осі
- Поділіть вісь на 2
- Встановіть вісь довільно
- Скинути значення осі до останнього записаного числа

Ви повинні вибрати віджет діалогового вікна введення, який відповідає dialog_code_string, зазвичай це вибирається з віджета screenOptions.

halpin_option
: Встановить вивід HAL у значення "true" (істина), коли вісь вибрано.

joint_number
Слід встановити відповідний номер суглоба.

axis_letter
Слід встановити відповідну літеру осі.

Ось властивості меню, що активується за допомогою функції «клацання та утримання»:

showLast
Показати дію «Встановити на останній».

showDivide
Покажіть дію «Поділити на 2».

showGotoOrigin
Показати дію «Перейти до початку координат G53/G5x».

showZeroOrigin
Покажіть дію «Нульове походження».

showSetOrigin
Показати дію «Встановити походження».

dialog_code_string
Встановлює, яке діалогове вікно з’являтиметься для числового введення, наприклад, «ВВЕДЕННЯ
або «КАЛЬКУЛЯТОР», щоб викликати діалогове вікно введення лише за допомогою друку
або діалогове вікно введення за допомогою сенсорного/друкованого калькулятора.

Ось зразок запису в таблиці стилів:

```
AxisToolButton {
    /* b''3b''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''вb''b''cb''b''ib'' b''пb''b' ↩
        'ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''мb''b''eb''b''нb''b''юb''  ↩
        */
    qproperty-showLast: false;
    qproperty-showDivide : true;
    qproperty-showGotoOrigin: true;
    qproperty-showZeroOrigin: true;
    qproperty-showSetOrigin: false;
    qproperty-dialog_code_string: CALCULATOR;
}
```

Він базується на *QToolButton* PyQt.

12.7.2.4 `BasicProbe` - Простий віджет зондування фрезерування

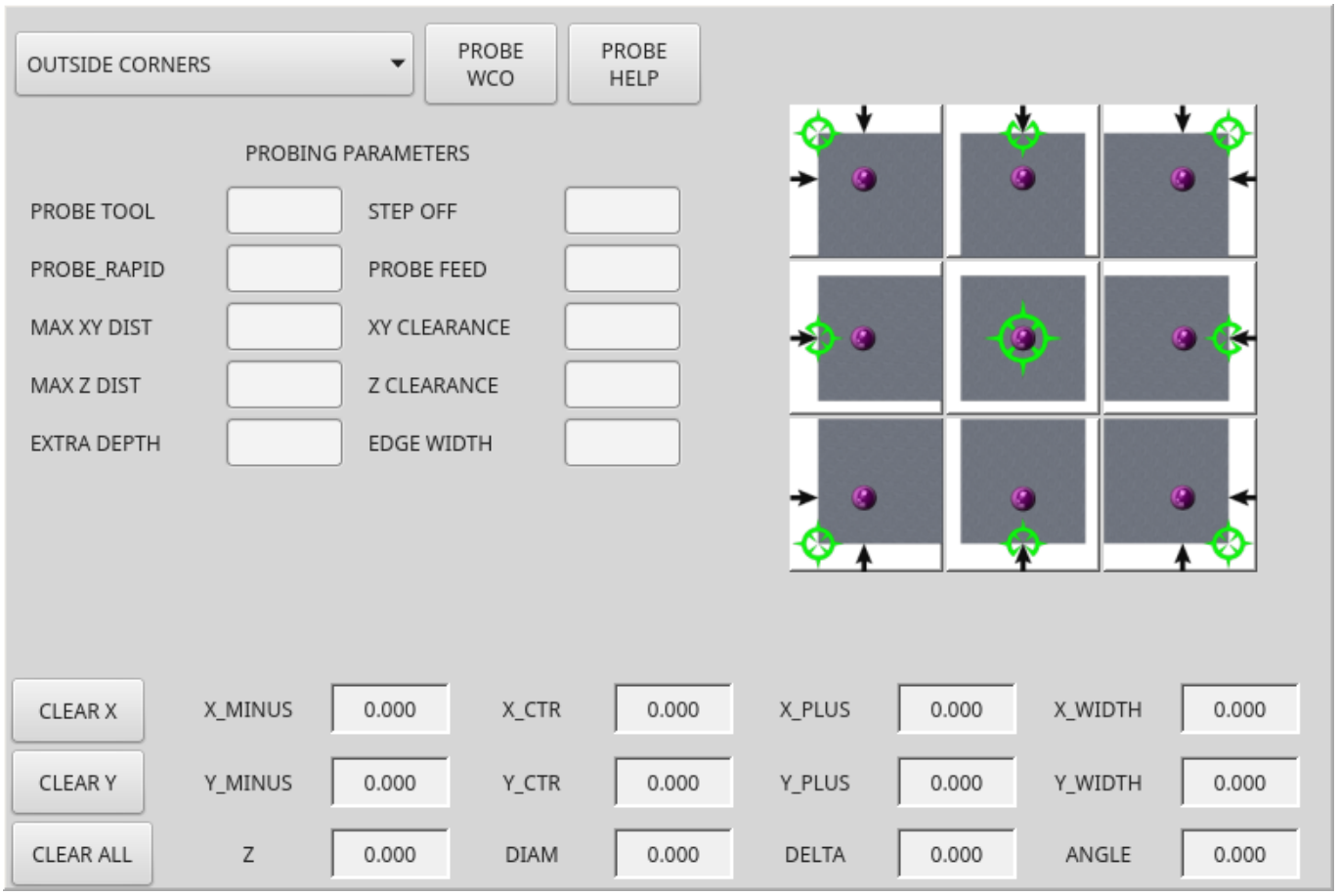


Figure 12.88: QtVCP `BasicProbe`: Простий віджет зондування фрезерування

Віджет для **зондування на фрезерному верстаті**. Використовується екраном *QtDragon*.

12.7.2.5 `CamView` - Віджет вирівнювання заготовки та налаштування початку координат

Цей віджет **відображає зображення з веб-камери**.

Він *накладає регульовану круглу мішень та перехрестя* на зображення.

CamView був створений з урахуванням точного візуального позиціонування.

Це використовується для **вирівнювання елементів заготовки або нульової деталі за допомогою веб-камери**.

Він використовує бібліотеку візуального мистецтва *OpenCV*.

12.7.2.6 `DROLabel` - Віджет відображення положення осі

Це **відобразить поточне положення осі**.
Ви також можете натиснути на підпис і переглянути список дій.

Qjoint_number
Номер індексу суглоба (X=0, Y=1) зміщення для відображення (10 вказуватиме обертальне зміщення).

Qreference_type
Фактична, відносна або відстань, що залишилася (0,1,2).

metric_template
Формат відображення, наприклад `%10.3f`.

imperial_template
формат відображення, наприклад `%9.4f`.

angular_template
Формат відображення, наприклад `%Rotational: 10.1f`.

always_display_diameter
Перемикає опцію відображення

always_display_radius
Перемикає опцію відображення

display_as_per_m7m8
Перемикає опцію відображення. Буде дотримуватися поточного режиму M7/8.

follow_reference_changes
Перемикає опцію відображення. Відповідатиме режиму довідки про повідомлення СТАН, тобто ви можете використовувати кнопки дій, щоб налаштувати, як воно наразі відображається.

Ось опції меню, що відкриваються при натисканні:

showLast
Показати дію «Встановити на останній».

showDivide
Покажіть дію «Поділити на 2».

showGotoOrigin
Показати дію «Перейти до початку координат G53/G5x».

showZeroOrigin
Покажіть дію «Нульове походження».

showSetOrigin
Показати дію «Встановити походження».

dialogName
Встановлює, яке діалогове вікно з’явиться для числового введення, тобто ВВЕДЕННЯ чи КАЛЬКУЛЯТОР.

Віджет `DROLabel` містить властивість **`isHomed`**, яку можна використовувати зі стилями для зміни кольору `DRO_Label` на основі стану відправлення номера з’єднання в LinuxCNC.

Ось зразок запису таблиці стилів, який:

- Встановлює шрифт усіх віджетів `DRO_Label`
- Встановлює текстовий шаблон (для встановлення роздільної здатності) цифрового індикатора (DRO)
- Потім встановлює колір тексту на основі властивості Qt `isHomed`.

- показати всі опції меню.

```
DROLabel {
    font: 25pt "Lato Heavy";
    qproperty-imperial_template: '%9.4f';
    qproperty-metric_template: '%10.3f';
    qproperty-angular_template: '%11.2f';

    /* b''3b''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''вb''b''cb''b''ib'' b''пb''b' ↩
        'ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''мb''b''eb''b''нb''b''юb''  ↩
        */
    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;
}
```

Ось як вказати певний віджет за його `objectName` у Qt Designer:

```
DROLabel #dr0_x_axis [isHomed=false] {
    color: yellow;
}
```

Він базується на *QLabel* PyQt.

12.7.2.7 `FileManager` - Віджет вибору завантаження файлів

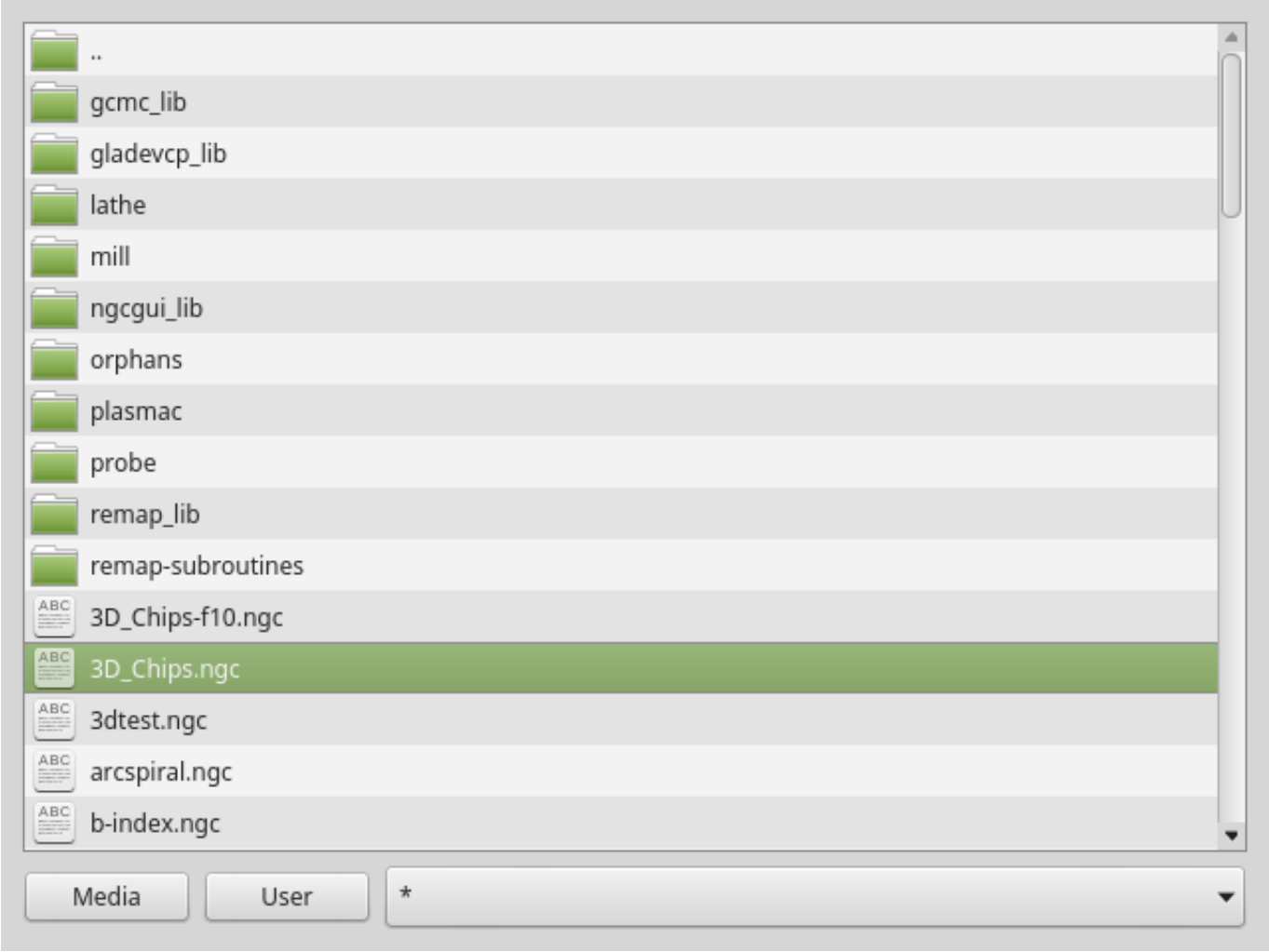


Figure 12.89: QtVCP `FileManager`: Віджет вибору завантаження файлів

Цей віджет використовується для **вибору файлів для завантаження**.

Він має можливість прокручувати назви за допомогою обладнання, такого як MPG.

Можна налаштувати завантаження файлів за допомогою класу-патчу функції `load(self,fname)`.

Функція `getCurrentSelected()` поверне кортеж Python, що містить шлях до файлу та інформацію про те, чи є це файлом.

```
temp = FILEMANAGER.getCurrentSelected()
print('filepath={}'.format(temp[0]))
if temp[1]:
    print('Is a file')
```

Властивості таблиць стилів

`doubleClickSelection` *(bool)*
Визначає, чи потрібно *двічі клацнути на папці*.
Одноразове клацання на папці (False) увімкнено за замовчуванням і призначено для користувачів

сенсорних екранів.
Нижче наведено приклад налаштування цієї властивості:

```
#filemanager {
    qproperty-doubleClickSelection: True;
}
```

showListView ***(bool)***
Визначає, чи слід *відображати структуру файлів/папок у формі списку*.
Табличний вигляд (False) увімкнено за замовчуванням.
Нижче наведено приклад того, як встановити цю властивість:

```
#filemanager {
    qproperty-showListView: True;
}
```

Він базується на FIXME від PyQt

12.7.2.8 `GcodeDisplay` - Віджет відображення тексту G-коду

Це **відображає G-код у текстовому вигляді**, виділяючи поточний рядок.

Це також може відображати:

- **Історія MDI**, коли LinuxCNC перебуває в режимі `MDI`.
- **Записи журналу**, коли LinuxCNC перебуває в режимі `MANUAL`.
- **Записи файлу налаштувань**, якщо ви введете `PREFERENCE` великими літерами у віджет `MDILine`.

Він має *signal* **`percentDone(int)`**, який можна підключити до слота (наприклад, `progressBar` для відображення відсотка виконання).

`auto_show_mdi_status`
Встановіть значення «true», щоб віджет перемикався на історію MDI, коли він у режимі MDI.

`auto_show_manual_status`
Встановіть значення «true», щоб віджет перемикався в режим машинного журналу в ручному режимі.

Властивості `GcodeDisplay` можна встановити в таблиці стилів за допомогою наступного коду, доданого до файлу .qss (наступні варіанти кольорів є випадковими).

```
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”;
}
```

Для *лексера G-коду за замовчуванням* віджета `GcodeDisplay`:

- **styleColor0 = За замовчуванням**: Все, що не входить до груп нижче
- **styleColor1 = Номер рядка та коментарі**: Nxxx та коментарі (символи всередині та включно з *()* або будь-що після *;* (якщо використовується поза дужками), за винятком примітки нижче)
- **styleColor2 = G-код**: G та цифри після нього
- **styleColor3 = M-код**: M та цифри після нього
- **styleColor4 = Axis**: XYZABCUVW
- **styleColor5 = Інше**: EFHIJKDQLRPST (подача, об/хв, радіус тощо)
- **styleColor6 = AxisValue**: Значення після XYZABCUVW
- **styleColor7 = OtherValue**: Значення, що відповідають EFHIJKDQLRPST$

Note
Для коментарів колір «OtherValue» (колір 5) можна використовувати для виділення «print», «debug», «msg», «logopen», «logappend», «logclose», «log», «pyrun», «pyreload», «abort», «probeopen», «probeclose» всередині коментаря в дужках у рядку G-коду. А також «py», якщо рядок починається з «;py,». Приклади: (print, text), (log, text), (msg, text) або (debug, text). Якщо в одному рядку є кілька прикладів, виділятиметься лише останній.

Визначення шрифтів:

```
”style name, size, -1, 0, bold setting (0-99), italics (0-1),
underline (0-1),0,0,0”
```

Він базується на *QsciScintilla* PyQt.

12.7.2.9 `GcodeEditor` - Віджет редактора G-кодів програм

Це розширення віджета `GcodeDisplay`, яке **додає зручності редагування**.

Він базується на *QWidget* PyQt, який містить віджет `GcodeDisplay`.

12.7.2.10 `GCodeGraphics` - Віджет графічного фону G-коду

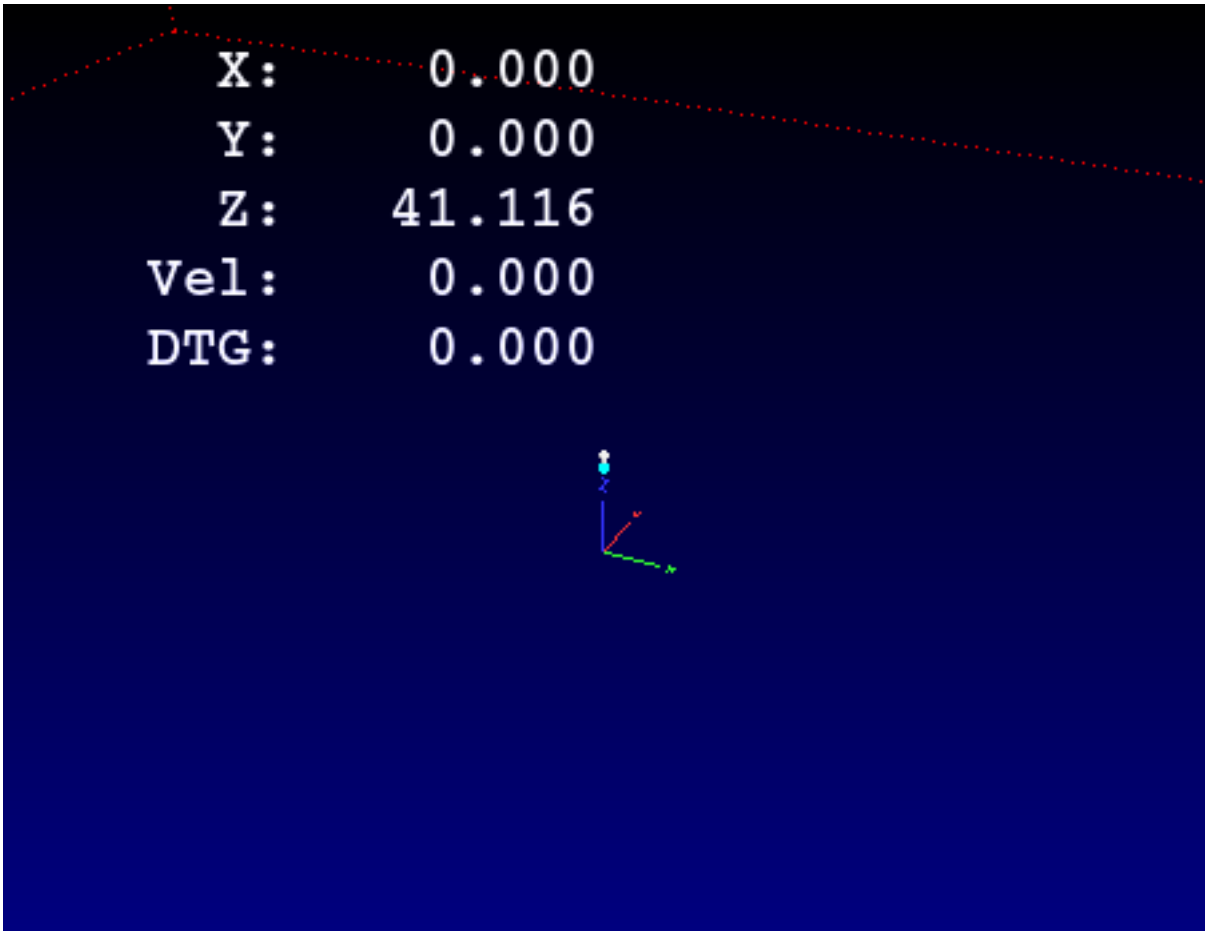


Figure 12.90: QtVCP GcodeGraphics: Віджет графічного фону G-коду

Це **відображає поточний G-код у графічному вигляді**.

Властивості таблиць стилів

`dro-font/dro-large-font` *(string)*
Встановлює властивості шрифту DRO для малого та великого форматів.
Тут ми посилаємося на базову назву віджета; GCodeGraphics

```
GCodeGraphics{
    qproperty-dro_font:”monospace  bold 12”;
}
GCodeGraphics{
    qproperty-dro_large_font:”Times 25”;
}
```

`_view` *(string)*
Встановлює *стандартну орієнтацію перегляду* під час завантаження графічного інтерфейсу користувача.
Для токарного верстата допустимими варіантами є p, y, y2. Для інших екранів допустимими

варіантами є p, x, y, z, z2.
Нижче наведено приклад того, як встановити цю властивість (з посиланням на ім'я, вибране користувачем віджета):

```
#gcodegraphics{
    qproperty-_view: z;
}
```

_dro *(bool)*
Визначає, чи *показувати* DRO.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_dro: False;
}
```

_dtg *(bool)*
Визначте, чи *показувати відстань, що залишилася*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_dtg: False;
}
```

_metric *(bool)*
Визначає, чи *відображати одиниці вимірювання в метричній системі за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_metric: False;
}
```

_overlay *(bool)*
Визначає, чи *показувати накладання за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_overlay: False;
}
```

_offsets *(bool)*
Визначає, чи *відображати зміщення за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_offsets: False;
}
```

_small_origin *(bool)*
Визначає, чи *показувати малий початок координат за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_small_origin: False;
}
```

overlay_color *(основний, вторинний або колір у форматі RGBA)*
Встановлює *колір накладання за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-overlay_color: blue;
}
```

overlay_alpha ***(float)***
Встановлює *значення альфа-відображення накладання за замовчуванням*. Це впливає на непрозорість накладання, якщо встановлено значення від 0.0 до 1.0.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-overlay_alpha: 0.15;
}
```

background_color ***(основний, вторинний або колір у форматі RGBA)***
Встановлює *колір фону за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-background_color: blue;
}
```

+_use_gradient_background+ ***(bool)***
Визначає, чи *використовувати градієнтний фон за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-_use_gradient_background: False;
}
```

jog_color ***(основний, вторинний або колір у форматі RGBA)***
Встановлює *колір штрихування за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-jog_color: red;
}
```

Feed_color ***(основний, вторинний або колір у форматі RGBA)***
Встановлює *колір стрічки за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-Feed_color: green;
}
```

Rapid_color ***(основний, вторинний або колір у форматі RGBA)***
Встановлює *колір швидкого переходу за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-Rapid_color: rgba(0, 0, 255, .5);
}
```

InhibitControls ***(bool)***
Визначає, чи *забороняти зовнішні елементи керування за замовчуванням*.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-InhibitControls:True;
}
```

`MouseButtonMode` *(int)*
Змінює *поведінку кнопки миші* для обертання, переміщення або масштабування в попередньому перегляді.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-MouseButtonMode: 1;
}
```

Існує 12 дійсних режимів:

Режим	Перемістити	Зум	Повернути
0	Ліворуч	Середній	Праворуч
1	Середній	Праворуч	Ліворуч
2	Середній	Ліворуч	Праворуч
3	Ліворуч	Праворуч	Середній
4	Праворуч	Ліворуч	Середній
5	Праворуч	Середній	Ліворуч

Режими 6-11 призначені для верстатів, яким потрібен лише 2D-попередній перегляд, таких як плазмові або деякі токарні верстати, і для яких не призначена кнопка повороту.

Режим	Перемістити	Зум
6	Ліворуч	Середній
7	Середній	Ліворуч
8	Праворуч	Ліворуч
9	Ліворуч	Праворуч
10	Середній	Праворуч
11	Праворуч	Середній

`MouseWheelInvertZoom` *(bool)*
Визначає, чи слід *інвертувати напрямок масштабування* під час масштабування за допомогою коліщатка миші.
Нижче наведено приклад того, як встановити цю властивість:

```
#gcodegraphics{
    qproperty-MouseWheelInvertZoom:True;
}
```

ACTION функції Бібліотека `ACTION` може керувати графічним віджетом G-коду.

`ACTION.RELOAD_DISPLAY()`
Перезавантажте поточну програму, яка перерахує початок координат/зміщення.

`ACTION.SET_GRAPHICS_VIEW(_view_)`
Можна надсилати такі команди `view`:

- `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_)
Безпосередньо встановіть відносну панораму огляду в напрямках x та y.

ACTION.ADJUST_ROTATE(_X,Y_)
Безпосередньо встановити відносне обертання огляду в напрямку x та y.

Він базується на віджеті *OpenGL* від PyQt.

12.7.2.11 JointEnableWidget - FIXME

Документація FIXME JointEnableWidget

12.7.2.12 `JogIncrements` - Віджет вибору значення кроку поштовху

Цей віджет дозволяє користувачеві **вибирати значення кроку підйому для підйому**.

Значення джоггу беруться з *INI-файлу* за адресою:

- `[DISPLAY]INCREMENTS`, або
- `[DISPLAY]ANGULAR_INCREMENTS`

Це буде *доступно для всіх віджетів* через `STATUS`.

Ви можете вибрати лінійний або кутовий приріст за допомогою властивості **`linear_option`** у редакторі властивостей Qt Designer.

Він базується на *ComboBox* PyQt.

12.7.2.13 `MacroTab` - Віджет спеціальних макросів

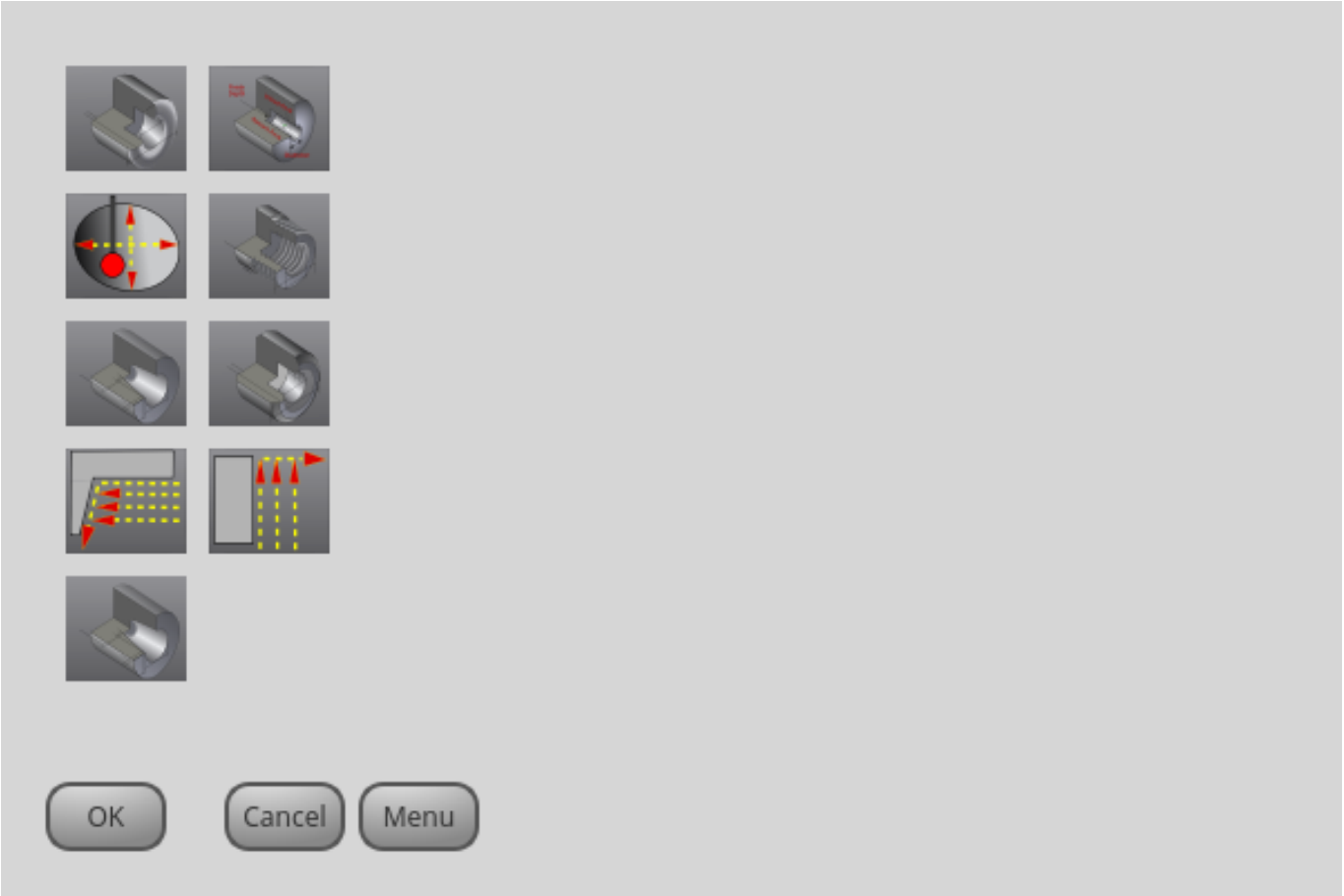


Figure 12.91: QtVCP `MacroTab`: спеціальний віджет макросів

Цей віджет дозволяє користувачеві **вибирати та налаштовувати спеціальні макропрограми** для виконання невеликих завдань.

Він використовує *зображення* для візуального представлення макросу та для значка.

Він шукає спеціальні макроси, використовуючи визначення *INI*:

```
[RS274NGC]
SUBROUTINE_PATH =
```

Макроси — це підпрограми **`O-word` зі спеціальними коментарями** для роботи з панеллю запуску. Перші три рядки *обов'язкові* містити ключові слова нижче, четвертий є необов'язковим.

Ось приклад перших чотирьох рядків у *O-словному файлі*:

```
; 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 Це *перший рядок* у файлі O-word.

Це **список тексту, розділеного комами, який відображається над записом**.
У функції O-word буде **по одному для кожної необхідної змінної**.

Якщо макрос не потребує змінних, залиште його порожнім:

```
; MACROCOMMAND=
```

MACRODEFAULTS Це має бути *другий рядок* у файлі O-word.

Це **список значень за замовчуванням для кожної змінної** у функції O-word, розділених комами.

Якщо ви використовуєте в списку слова «radiotrue», «radiofalse», «true» або «false», буде показано «**radiobutton**».
Якщо ви використовуєте слово «checktrue» або «checkfalse» у списку, буде показано «**checkbox**».
Якщо ви використовуєте слово «buttontrue» або «buttonfalse» у списку, буде показано «**Checkable Pushbutton**».
Якщо значення за замовчуванням має десяткову частину, macroTab припускає, що ви хочете отримати число з плаваючою комою, інакше — ціле число.

Note
Під час використання радіокнопок встановлюйте значення «true» лише для однієї з них. Радіокнопки використовуються для ексклюзивного вибору.

MACROIMAGE Це має бути *третій рядок* у файлі O-word.

- **Зображення SVG**
  Якщо використовуються файли зображень SVG, вони повинні закінчуватися розширенням `.svg`.

  Зображення необхідно додати до *шарів SVG*, які використовуються для визначення різних зображень для макросів та піктограм.

  Значення – це список із трьох упорядкованих полів, розділених комами:

```
; MACROIMAGE=filename.svg,macro_layer_name[,icon_layer_name]
```

  З:

  `_filename_.svg`
  Ім'я файлу зображення SVG як перше поле.
  Вважається, що він знаходиться в тій самій папці, що й файл O-word.

  `*macro_layer_name`
  Назва шару макрозображення як друге поле.

icon_layer_name
Назва шару зображення значка як необов’язкове третє поле. Якщо третій запис відсутній, те саме зображення буде використано для макросу та значка.

- **Зображення PNG/JPG**:
  Значення залишається списком, розділеним комами:

```
; MACROIMAGE=macro_image.(png|jpg)[,icon_image.(png|jpg)]
```

  З:

_macro_image_.(png|jpg)
Ім’я файлу зображення макросу як перше поле.
Передбачається, що файл зображення знаходиться в тій самій папці, що й макрос.

_icon_image_.(png|jpg)
Назва файлу зображення піктограми як необов’язкове друге поле.
Якщо другий запис відсутній, те саме зображення буде використано для макросу та зображення.

Якщо ключове слово присутнє, але записи відсутні, зображення не використовуватимуться.

MACROOPTIONS Цей *необов’язковий* рядок має бути четвертим рядком у файлі O-word.

Це список ключових слів та даних, розділених комами, всі з яких є необов’язковими:

LOAD:yes
Показує кнопку завантаження.

SAVE:yes
Показує кнопку збереження.

DEFAULT:ThisMacroData.txt
Встановлює попередньо вибрану назву файлу за замовчуванням під час завантаження/збереженн
даних для цього макросу.
Це може бути будь-яка дійсна назва файлу, але вона має закінчуватися на *.txt*.

PATH:~/linuxcnc/nc_files/mySavedMacrosData
Встановлює папку каталогу за замовчуванням для попереднього вибору під час завантаження/збе
даних для цього макросу.

Ось підказки щодо таблиць стилів для налаштування віджета MacroTab.

```
MacroTab CustomButton{
    width: 20px;
    height: 40px;
}

MacroTab QPushButton {
    width: 80px;
    height: 40px;
}

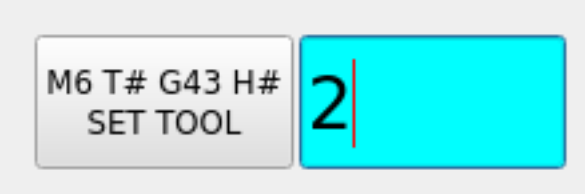
MacroTab QLabel {
    font: 24pt ”Lato Heavy”;
}

TouchSpinBox  LineEdit {
    font: 12pt ”Lato Heavy”;
}

TouchSpinBox  QPushButton {
    width: 60px;
    height: 100px;
}
```

12.7.2.14 `OperatorValueLine` - Віджет введення рядка значення оператора

Оператор вводить значення в цей віджет, які будуть застосовані до шаблону, а потім, за бажанням, відправлені в MDI негайно або застосовані пізніше. Віджет підтримує опціональний спливаючий калькулятор, клавіатуру або вибір інструментів для зручного введення даних на сенсорному екрані за допомогою налаштування `dialog_keyboard_option`. Щоб змінити тип діалогового вікна, відредагуйте `dialog_code_option`.



Віджет підтримує опцію форматування, яка передається до функції Python `format()` для створення кінцевого результату для команди MDI. Спеціальний маркер `{value}` можна вставити в будь-якому місці цього форматного рядка, де має з'явитися значення. Властивість форматування називається `mdi_command_format_option`, наприклад:

- `M3 S{value}` запустити шпиндель зі швидкістю, введеною оператором.
- `M6 T{value} G43 H{value}` щоб виконати зміну інструменту та зміну зміщення довжини інструменту на основі введеного номера інструменту

Віджет може бути налаштований на автоматичне виконання команди MDI після відправлення, якщо `issue_mdi_on_submit_option` встановлено на `True`. Якщо `False`, виконання команди може бути здійснено пізніше за допомогою сигналу або виклику функції з іншого віджета.

У випадках, коли `issue_mdi_on_submit_option` має значення `False`, виклик функції `issue_mdi()` призведе до виконання команди. Слоти, приєднані до віджетів, таких як PushButtons, можуть запускати команду MDI при натисканні, наприклад:

```
    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()
```

Віджет відстежує, чи введене значення є очікуючим і ще не було видане, за допомогою властивості `isPendingValue`. Це може бути використано для стилізації віджета за допомогою таблиці стилів. Це може бути використано для попередження оператора про те, що він ввів значення, але для його застосування необхідно виконати іншу дію.

У наступному уривку таблиці стилів віджет запису буде виділено блакитним фоном, коли значення очікують на розгляд і ще не застосовані.

```
#lineSpindleSpeed[isPendingValue=true],
#lineToolNumber[isPendingValue=true] {
    background: cyan;
}

#lineSpindleSpeed[isPendingValue=false],
#lineToolNumber[isPendingValue=false] {
    background: none;
}
```

12.7.2.15 `MDILine` - Віджет введення команд MDI в рядок

Тут можна **вводити команди MDI**.

Доступна спливаюча клавіатура.

Вбудовані команди Також у цьому віджеті доступні **вбудовані команди**.

Введіть будь-яку з цих команд, *чутливих до регістру*, щоб завантажити відповідну програму або отримати доступ до функції:

`HALMETER`
Запускає LinuxCNC `halmeter` utility.

`HALSHOW`
Запускає LinuxCNC `halshow` корисність.

`HALSCOPE`
Запускає утиліту LinuxCNC `halscope`.

`STATUS`
Запускає утиліту LinuxCNC `status`.

`CALIBRATION`
Запускає LinuxCNC Calibration

`CLASSICLADDER`
Запускає посилання:../ladder/classic-ladder.html[ClassicLadder GUI], якщо *компонент HAL реального часу ClassicLadder* був завантажений конфігураційними файлами машини.

`PREFERENCE`
Завантажує файл налаштувань у `GcodeEditor`.

`CLEAR HISTORY`
Очищає історію MDI.

`net`
Див. команди `halcmd net`.
Якщо команда виконана невдало, виникне помилка.

- *Синтаксис*: `net <назва сигналу> <назва виводу>`
- *Приклад*: `net plasmac:jog-inhibit motion.jog-stop`

`setp`
Встановлює значення виводу або параметра.
Допустимі значення залежать від типу об’єкта виводу або параметра.
Це призводить до помилки, якщо типи даних не збігаються або вивод підключено до сигналу.

- *Синтаксис*: `setp <pin/parameter-name> <value>`
- *Приклад*: `setp plasmac.resolution 100`

`unlinkp`
Відключає контакт від сигналу.
Якщо контакт не існує, виникне помилка.
Запуск LinuxCNC з терміналу може допомогти визначити основну причину, оскільки там будуть відображатися повідомлення про помилки з `hal_lib.c`.

- *Синтаксис*: `unlinkp <pin name>`
- *Приклад*: `unlinkp motion.jog-stop`

Note
Функція `MDILine` **`spindle_inhibit`** може використовуватися файлом-обробником графічного інтерфейсу для блокування команд шпинделя `M3`, `M4` та `M5`, якщо це необхідно.

Він базується на *QLineEdit* від PyQt.

12.7.2.16 `MDIHistory` - Віджет історії команд MDI

Відображає **список попередніх MDI-команд, який можна прокручувати**.

Для команд MDI вбудовано рядок редагування. З цього віджета можна отримати доступ до тих самих вбудованих команд MDILine.

Історія *записується у файл, визначений в INI* під заголовком `[DISPLAY]` (тут показано значення за замовчуванням):

```
MDI_HISTORY_FILE = '~/.axis_mdi_history'
```

12.7.2.17 `MDITouchy` - Віджет введення MDI на сенсорному екрані

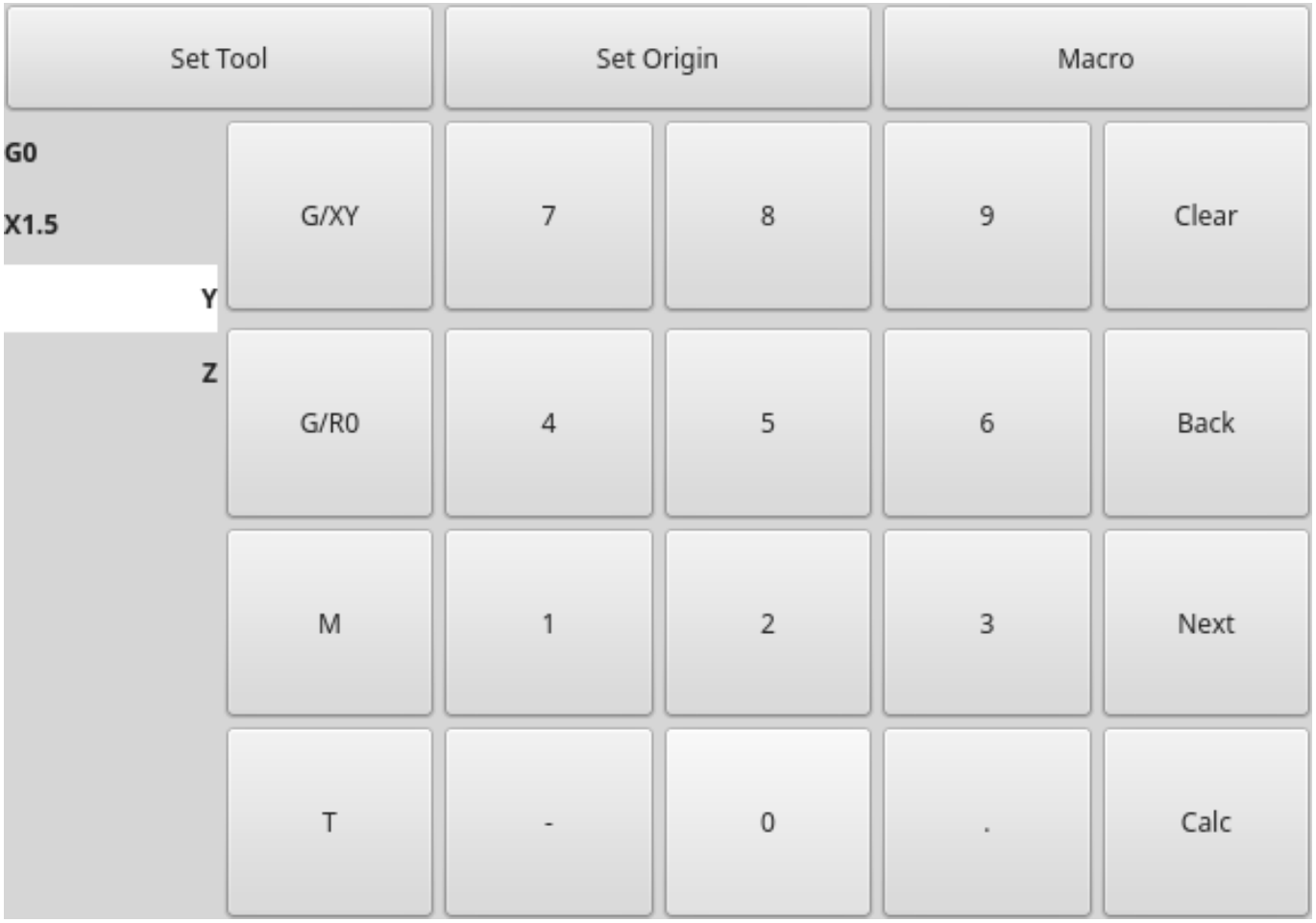


Figure 12.92: QtVCP `MDITouchy`: Віджет введення MDI на сенсорному екрані

Цей віджет відображає **кнопки та рядки введення для введення команд MDI**.

Виходячи з процесу введення MDI на сенсорному екрані LinuxCNC, його великі кнопки найбільш корисні для сенсорних екранів.

Використання `MDITouchy`:

- Спочатку натисніть одну з кнопок `G/XY`, `G/RO`, `M` або `T`. Ліворуч з’являться поля для введення, які можна заповнити.
- Потім натисніть «Далі» та «Назад», щоб переходити між полями.
- `Calc` з’явиться діалогове вікно калькулятора.
- `Clear` очищає поточний запис.
- Команда `Set Tool` викличе зміну інструменту.
- `Set Origin` дозволить встановити початок координат поточної системи G6x.
- `Macro` викличе будь-які доступні програми для роботи з макросами (ngc).

Віджет *потрібний явний виклик коду Python MDITouchy для фактичного виконання команди MDI*:

- **Для коду файлу обробника**
  Якщо віджет мав назву «mditouchy» в Qt Designer, команда нижче виконала б відображену команду MDI:

  ```
  self.w.mditouchy.run_command()
  ```

- **Для використання кнопки дії**
  Якщо віджет мав назву «mditouchy» в Qt Designer, скористайтеся опцією кнопки дії «Викликати команди Python» та введіть:

  ```
  INSTANCE.mditouchy.run_command()
  ```

Кнопка макросу *циклічно перемикає макроси, визначені в заголовку INI [DISPLAY].*

Додайте один або декілька рядків `MACRO` такого формату:

```
MACRO = macro_name [param1] [... paramN]
```

У наведеному нижче прикладі `increment` – це назва макросу, і він приймає два параметри з назвами `xinc` та `yinc`.

```
MACRO = incerment xinc yinc
```

Тепер помістіть макрос у файл з назвою `macro_name.ngc` у каталозі `PROGRAM_PREFIX` або в будь-який каталог у `SUBROUTINE_PATH`, зазначеному у файлі INI.

Продовжуючи наведений вище приклад, він матиме назву `increment.ngc`, а його вміст виглядатиме так:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Зверніть увагу, що _назва підфайлу точно збігається з назвою файлу та назвою макросу, включаючи регістр літер.

Коли ви викликаєте макрос, натиснувши кнопку «Макрос», ви можете ввести значення для параметрів (у нашому прикладі це `xinc` та `yinc`).
Вони передаються макросу як позиційні параметри: `#1`, `#2`... `#N` відповідно.
Параметри, які ви залишили порожніми, передаються як значення `0`.

Якщо є кілька різних макросів, натискайте кнопку Макрос кілька разів, щоб перемикатися між ними.

У цьому простому прикладі, якщо ввести -1 для xinc та викликати цикл MDI, буде викликано швидкий рух *G0*, що переміщує на одиницю ліворуч.

Ця макрофункція корисна для зондування країв/отворів та інших завдань налаштування, а також, можливо, для фрезерування отворів або інших простих операцій, які можна виконувати з панелі без необхідності використання спеціально написаних програм G-коду.

12.7.2.18 `OriginOffsetView` - Віджет перегляду та налаштувань походження

	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`: Віджет перегляду та налаштувань походження

Цей віджет дозволяє безпосередньо **візуалізувати та змінювати зміщення походження системи користувача**.

Він *оновить файл параметрів LinuxCNC* для внесених або знайдених змін.

Налаштування можна змінити в LinuxCNC лише після повернення до початкового положення та коли контролер руху не використовується.

Відображення та введення змінюватимуться між метричною та імперською системами вимірювання, залежно від поточних налаштувань `G20` / `G21` LinuxCNC.

Поточна система користувача, що використовується, буде виділена.

Додаткові дії можуть бути інтегровані для маніпулювання налаштуваннями.
Ці дії залежать від додаткового коду, доданого або до комбінованого віджета, такого як діалогове вікно `originoffsetview`, або до коду обробника екранів.
Типовими діями можуть бути «Очистити поточні зміщення користувача» або «Обнулити X».

Клацання по стовпцях і рядках дозволяє налаштувати параметри.

Для введення даних або тексту можна налаштувати спливаюче діалогове вікно.

Розділ коментарів буде записаний у файлі налаштувань.

Він базується на *QTableView*, *QAbstractTableModel* та *ItemEditorFactory* PyQt.
Властивості, функції та стилі базових об’єктів PyQt завжди доступні.

Властивості `OriginOffsetView` має такі властивості:

`dialog_code_string`
: Встановлює, яке діалогове вікно з’явиться для введення чисел.

`test_dialog_code_string`
: Встановлює, яке діалогове вікно з’явиться після введення тексту.

`metric_template`
: Формат числових даних у метричній системі.

`imperial_template`
: Імперський числовий формат даних.

`styleCodeHighlight`
: Колір підсвічування поточної використовуваної системи користувача.

Їх можна встановити в:

- Qt Designer, в
- Код обробника Python

```
self.w.originoffsetview.setProperty('dialog_code','CALCULATOR')
self.w.originoffsetview.setProperty('metric_template','%10.3f')
```

- Або (якщо доречно) у таблицях стилів

```
OriginOffsetView{
  qproperty-styleColorHighlist: lightblue;
}
```

12.7.2.19 `RadioAxisSelector` - FIXME

Документація FIXME RadioAxisSelector

12.7.2.20 `RoundButton` - Віджет `ActionButton` круглої форми

Круглі кнопки працюють так само, як і *ActionButtons*, за винятком того, що кнопка обрізана.

Вони призначені лише для візуальної відмінності.

Вони мають *дві властивості шляху* для відображення **зображень** на значеннях true та false.

12.7.2.21 `StateLabel` - Віджет відображення міток станів режимів контролера

Це **відобразить мітку на основі станів true/false режимів контролера машини**.

Ви можете вибирати між різними текстами на основі «правда» чи «хиба».

Властивості вибору штатів Штати можна вибрати за допомогою цих властивостей:

`css_mode_status`
: `True` коли верстат знаходиться в режимі `G96` *Постійна швидкість поверхні*.

`diameter_mode_status`
: `True` коли верстат знаходиться в режимі `G7` *Діаметр токарного верстата*.

`fpr_mode_status`
: `True` коли верстат знаходиться в режимі `G95` *Подача на оберт*.

`metric_mode_status`
: `True` коли машина знаходиться в метричному режимі `G21`.

Властивості текстових шаблонів

`true_textTemplate`
: Це буде текст, встановлений, коли опція має значення `True`.
Ви можете використовувати код *Qt rich text* для різних шрифтів/кольорів тощо.
Типовий шаблон для метричного режиму в стані true може бути: *Metric Mode*

`false_textTemplate`
: Це буде текст, встановлений, коли опція має значення `False`.
Ви можете використовувати код *Qt rich text* для різних шрифтів/кольорів тощо.
Типовий шаблон для метричного режиму в стані false може бути: *Imperial Mode*.

Він базується на *QLabel* PyQt.

12.7.2.22 `StatusLabel` - Віджет відображення міток стану змінних контролера

Це відобразить мітку на основі вибраного стану контролера машини.
Ви можете змінити спосіб відображення стану, замінивши код форматування Python у текстовому шаблоні. Ви також можете використовувати форматований текст для різних шрифтів/кольорів тощо.

Вибрані штати Ці стани можна вибрати:

`actual_spindle_speed_status`
: Використовується для відображення фактичної швидкості шпинделя, *повідомленої з виводу HAL* `spindle.0.speed-i`.
Вона перетворюється на *RPM*.
Зазвичай використовується `textTemplate` з `%d`.

actual_surface_speed_status
Використовується для відображення фактичної швидкості різання на токарному верстаті на основі осі X і швидкості шпинделя.
Вона перетворюється на відстань за хвилину.
Зазвичай використовується `textTemplate` з `%4.1f` (фути за хвилину) і `altTextTemplate` з `%d` (метри за хвилину).

blendcode_status
Показує поточне налаштування `G64`.

current_feedrate_status
Показує поточну фактичну швидкість подачі.

current_FPU_status
Показує поточну фактичну подачу на одиницю.

fcode_status
Показує поточне запрограмоване налаштування коду `F`.

feed_override_status
Показує поточне налаштування корекції подачі у відсотках.

filename_status
Показує назву останнього завантаженого файлу.

filepath_status
Показує повний шлях до останнього завантаженого файлу.

gcode_status
Показує всі активні G-коди.

gcode_selected_status
Показати поточний вибраний рядок G-коду.

halpin_status
Показує вихідний сигнал вибраного виводу HAL.

jograte_status
Показує поточну швидкість поштовхового переміщення на основі QtVCP.

jograte_angular_status
Показує поточну кутову швидкість поштовху на основі QtVCP.

jogincr_status
Показує поточний приріст Jog на основі QtVCP.

jogincr_angular_status
Показує поточний приріст кутового повороту на основі QtVCP.

machine_state_status
Показує поточний *стан інтерпретатора* машини, використовуючи текст, описаний у списку_ста
Станами інтерпретатора є:

- `Зупинено`
- `Запускаю`
- `Зупинено`
- `Призупинено`
- `Очікування`
- `Читання`

max_velocity_override_status
Показує поточне налаштування корекції максимальної швидкості осі.

mcode_status
Шоу *all active M-codes*.

motion_type_status
Показує поточний тип руху машини, використовуючи текст, описаний у списку типів руху (motion_type_list).

- *Жоден*
- *Швидкий*
- *Годувати*
- *Arc*
- *Зміна інструменту*
- *Зонд*
- *Ротарі Індекс*

requested_spindle_speed_status
Показує запитувану швидкість шпинделя — фактична може відрізнятися.

rapid_override_status
Показує поточне налаштування швидкого перемикання у відсотках (0-100).

spindle_override_status
Показує поточне налаштування корекції шпинделя у відсотках.

timestamp_status
Показує час на основі системних налаштувань.
Приклад корисного налаштування `textTemplate`: `%I:%M:%S %p`.
Див. модуль часу Python для отримання додаткової інформації.

tool_comment_status
Повертає текст коментаря з поточного завантаженого інструменту.

tool_diameter_status
Повертає діаметр з поточного завантаженого інструменту.

tool_number_status
Повертає номер поточного завантаженого інструменту.

tool_offset_status
Повертає зміщення поточного завантаженого інструменту, індексоване за допомогою `index_numb`
для вибору осі (0=x, 1=y тощо).

user_system_status
Показує *активну систему координат користувача* (налаштування `G5x`).

Інші об’єкти нерухомості

index_number
Ціле число, що визначає індекс стану інструмента для відображення.

state_label_list
Список позначок, що використовуються для опису різних станів машини.

motion_label_list
Список позначок, що використовуються для опису різних типів руху.

halpin_names
Ім’я halpin-a для моніторингу (повне ім’я, включаючи базову назву компонента HAL).

textTemplate
Зазвичай це використовується для **імперських (G20) або кутових числових налаштувань**, хоча не кожна опція має імперське/метричне перетворення.
Для налаштування текстового виводу використовуються правила форматування Python.
Можна використовувати %s для відсутності перетворення, %d для перетворення цілих чисел, %f для перетворення чисел з плаваючою комою тощо.
Також можна використовувати код Qt rich text.
Типовим шаблоном, що використовується для форматування імперських чисел з плаваючою комою в текст, є %9.4f або %9.4f inch.

alt_textTemplate
Зазвичай використовується для **метричних (G21) числових налаштувань**.
Використовує *правила форматування Python* для налаштування виводу тексту.
Типовим шаблоном, що використовується для форматування метричного числа з плаваючою комою в текст, є `%10.3f` або `%10.3f mm`.

Він базується на *QLabel* PyQt.

12.7.2.23 `StatusImageSwitcher` - Перемикач зображень стану контролера

Перемикач зображень статусу **перемикатиметься між зображеннями на основі станів LinuxCNC**.

***watch_spindle**
Перемикається між *3 images*: `stop`, `fwd`, `revs`.

***watch_axis_homed**
Перемикається між *2 зображеннями*: `вісь не встановлена в початкове положення`, `вісь встановлена в початкове положення`.

***watch_all_homed**
Перемикатиметься між *2 зображеннями*: `не всі розміщені`, `усі розміщені`.

***watch_hard_limits**
Перемикатиметься між *2 зображеннями або одним на кожне з’єднання*.

Ось приклад його використання для відображення піктограми стану самонаведення осі Z:

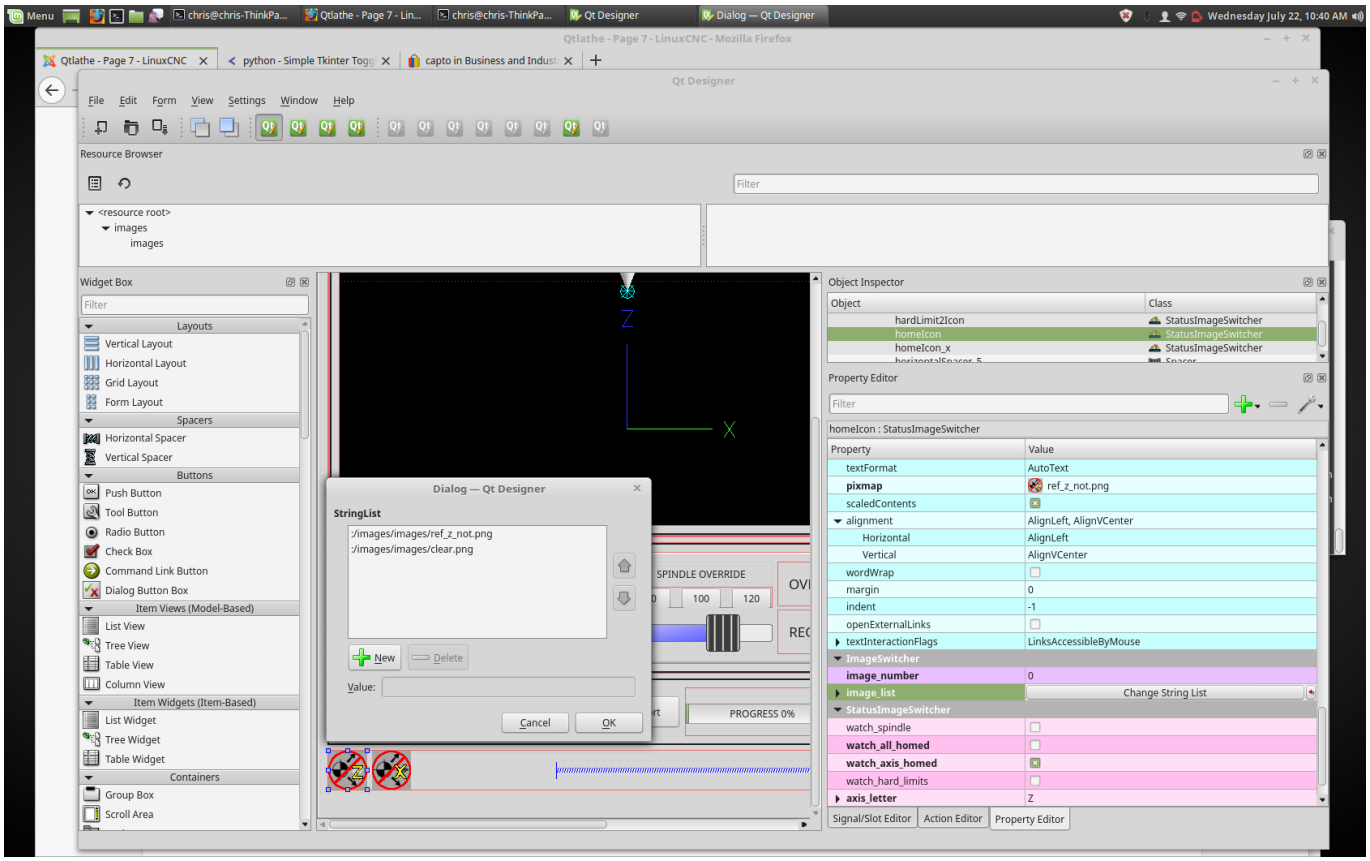


Figure 12.94: QtVCP `StatusImageSwitcher`: Перемикач зображень стану контролера

У розділі властивостей зверніть увагу на таке:

- `watch_axis_homed` позначено
- `axis_letter` встановлено на Z

Якщо двічі клацнути на `image_list`, з’явиться діалогове вікно, яке дозволить вам додати шляхи до зображень.

Якщо у вас є одне зображення як значок і одне *прозоре зображення*, то виглядатиме так, ніби воно показує та *приховує значок*.

Вибір шляхів до зображень можна здійснити, вибравши властивість `pixmap` та вибравши зображення.

Note
Налаштування `pixmap` призначене лише для тестового відображення та буде ігноруватися за межами Qt Designer.

- Клацніть правою кнопкою миші на назві зображення, і ви побачите «Копіювати шлях».
- Натисніть «Копіювати шлях».
- Тепер двічі клацніть властивість «список зображень», щоб відобразилося діалогове вікно.
- Натисніть кнопку «Новий».
- Вставте шлях до зображення в поле введення.

Зробіть це ще раз для наступного зображення.
Використайте чітке зображення для позначення прихованої значка.

Ви можете перевірити відображення зображень зі списку зображень, змінивши номер зображення.
У цьому випадку 0 не має базового положення, а 1 має базове положення.
Це призначено тільки для тестування відображення і буде ігноруватися поза Qt Designer.

12.7.2.24 `StatusStacked` - Віджет перемикання режимів відображення стану

Цей віджет **відображає одну з трьох панелей залежно від режиму LinuxCNC**.

Це дозволяє автоматично відображати різні віджети в режимах *Manual*, *MDI* та *Auto*. **TODO** Він базується на віджеті *QStacked* від PyQt.

12.7.2.25 `ScreenOption` - Віджет налаштувань загальних параметрів

Цей віджет не додає нічого візуального до екрана, але **налаштовує важливі опції**.

Це *бажаний спосіб використання цих опцій*.

Властивості Ці властивості можна встановити в Qt Designer, у коді обробника Python або (якщо доречно) у таблицях стилів.

До них належать:

`halCompBaseName`
Якщо залишити порожнім, QtVCP використовуватиме ім’я екрана як базове ім’я компонента HAL.
Якщо встановлено, QtVCP використовуватиме цей рядок як базове ім’я компонента HAL.
Якщо під час завантаження QtVCP використовується опція командного рядка `-c`, він використовув
ім’я, вказане в командному рядку — це замінить усі вищезазначені опції.
Якщо ви програмно встановите базове ім’я в `handlerfile`, воно замінить усі вищезазначені опції.
Цю властивість не можна встановити в таблицях стилів.

`notify_option`
Підключення до бульбашок сповіщень на робочому столі для отримання помилок та повідомлень.

`notify_max_messages`
Кількість повідомлень, що відображаються на екрані одночасно.

`catch_close_option`
Ловіть подію закриття, щоб з’явилося запитання _«Ви впевнені?».

`close_overlay_color`
Колір прозорого шару, що відображається після виходу.

`catch_error_option`
Моніторинг каналу помилок LinuxCNC.
Це також надсилає повідомлення через `STATUS` до будь-чого, що реєструється.

`play_sounds_option`
Відтворення звуків за допомогою `beep`, `espeak` та системного звуку.

`use_pref_file_option`
Налаштування шляху до файлу налаштувань.
Використання магічного слова `WORKINGFOLDER` у шляху до файлу налаштувань замінить його на шлях до запущеної конфігурації, наприклад `WORKINFOLDER/my_preferences`.

use_send_zmq_option
Використовується для ініціювання *вихідних повідомлень на основі ZMQ*.

use_receive_zmq_messages
Використовується для ініціювання *ZMQ на основі вхідних повідомлень*.
Ці повідомлення *можуть бути використані для виклику функцій у файлі обробника*, що дозволяє **зовнішнім програмам тісно інтегруватися з екранами на основі QtVCP**.

embedded_program_option
Вбудовувати програми, визначені у *INI*.

default_embed_tab
Це властивість для *розташування за замовчуванням для вбудовування зовнішніх програм*.
Її слід встановити як назву віджета сторінки вкладки в Qt Designer.

focusOverlay_option
Focus_overlay розмістить прозоре зображення або кольорову панель поверх головного екрана, щоб підкреслити фокус на зовнішній події – зазвичай діалоговому вікні.

messageDialog_option
Налаштовує діалогове вікно повідомлень – використовується для загальних повідомлень.

message_overlay_color
Колір прозорого шару, що відображається під час відображення діалогового вікна повідомлення.

closeDialog_option
Встановлює стандартне діалогове вікно закриття екрана.

entryDialog_option
Налаштовує діалогове вікно введення чисел.

entryDialogSoftKey_option
Встановлює плаваючу програмну клавіатуру, коли діалогове вікно введення перебуває у фокусі.

entry_overlay_color
Колір прозорого шару, що відображається під час відображення діалогового вікна введення.

toolDialog_option
Налаштовує діалогове вікно ручної зміни інструменту, включаючи контакт HAL.

tool_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно інструмента.

ToolUseDesktopNotify
Можливість використання діалогових вікон сповіщень на робочому столі для діалогового вікна ручної зміни інструменту.

ToolFrameless
Користувачі не можуть легко переміщувати діалогові вікна без рамки.

fileDialog_option
Встановлює діалогове вікно вибору файлу.

file_overlay_color
Колір прозорого шару, що відображається під час відображення діалогового вікна файлу.

keyboardDialog_option
Налаштовує віджет введення з клавіатури.

keyboard_overlay_color
Колір прозорого шару, що відображається під час відображення діалогового вікна клавіатури.

vesaProbe_option
Налаштовує діалогове вікно зонду у стилі Versa.

versaProbe_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `versaProbe`.

macroTabDialog_option
Налаштовує діалогове вікно вибору макросу.

macroTab_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `macroTab`.

camViewDialog_option
Налаштовує діалогове вікно вирівнювання камери.

camView_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `camView`.

toolOffset_option
Налаштовує діалогове вікно відображення/редактора зміщення інструменту.

toolOffset_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `toolOffset`.

originOffset_option
Налаштовує діалогове вікно відображення/редактора походження.

originOffset_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `originOffset`.

calculatorDialog_option
Налаштовує діалогове вікно введення даних калькулятора.

calculator_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно калькулятора.

machineLogDialog_option
Налаштовує діалогове вікно для відображення журналів з машини та QtVCP.

machineLog_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `machineLog`.

runFromLineDialog_option
Встановлює діалогове вікно для відображення параметрів запуску під час запуску виконання машини з довільного рядка.

runFromLine_overlay_color
Колір прозорого шару, що відображається, коли відображається діалогове вікно `runFromLine`.

user1Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user2Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user3Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user4Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user5Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user6Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user7Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user8Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user9Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

user10Color
Додатковий колір, який дизайнер екрану може використовувати у своєму дизайні.

Програмне налаштування властивостей Дизайнер екрану вибирає **налаштування за замовчува
для віджета `screenOptions`.

Після вибору більшість із них більше не потрібно буде змінювати. Але за потреби деякі можна змінити у файлі обробника або в таблицях стилів.

- **У файлі обробника**:
  Тут ми посилаємося на віджет за допомогою імені, визначеного користувачем у Qt Designer:

```
# b''чb''b''eb''b''pb''b''вb''b''ob''b''нb''b''иb''b''йb'', b''зb''b''eb''b''лb''b''eb''b ↩
    ''нb''b''иb''b''йb'', b''cb''b''иb''b''нb''b''ib''b''йb'', b''ab''b''лb''b''ьb''b' ↩
    'фb''b''ab'' 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)
```

- **У таблицях стилів**:
  Тут ми можемо посилатися на віджет за назвою, визначеною користувачем у Qt Designer, або за назвою класу віджета.

```
/* b''чb''b''eb''b''pb''b''вb''b''ob''b''нb''b''иb''b''йb'', b''зb''b''eb''b''лb''b''eb'' ↩
    b''нb''b''иb''b''йb'', b''cb''b''иb''b''нb''b''ib''b''йb'' 0-255, b''ab''b''лb''b' ↩
    'ьb''b''фb''b''ab'' 0-100% b''ab''b''бb''b''ob'' b''вb''b''ib''b''дb'' 0,0 b''дb''b' ↩
    'ob'' 1,0 */
/* b''Зb''b''нb''b''ab''b''кb'' # b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b' ↩
    'тb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''дb''b''лb''b''яb'' b' ↩
    'пb''b''ob''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''ib''b''мb''b' ↩
    'eb''b''нb''b''ib'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ab'', b''вb''b''иb''b ↩
    ''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''ob''b''гb''b''ob'' b''вb'' Qt Designer */
/* b''вb''b''ib''b''дb''b''пb''b''ob''b''вb''b''ib''b''дb''b''ab''b''єb''/b''зb''b''ab''b ↩
    ''cb''b''тb''b''ob''b''cb''b''ob''b''вb''b''уb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b' ↩
    'лb''b''иb''b''шb''b''eb'' b''дb''b''ob'' b''цb''b''ьb''b''ob''b''гb''b''ob'' b''нb''b ↩
    ''ab''b''зb''b''вb''b''ab''b''нb''b''ob''b''гb''b''ob'' b''вb''b''ib''b''дb''b''жb''b' ↩
    'eb''b''тb''b''ab'' */
#screen_options {
  qproperty-close_overlay_color: rgba(0, 255, 0, 0.75)
}
```

Деякі налаштування перевіряються лише під час запуску, тому не призведуть до змін після запуску. У цих випадках вам потрібно буде *внести зміни лише в Qt Designer*.

Записи файлу налаштувань Якщо вибрано опцію *preference file*, віджет `screenOption` створить **INI-файл налаштувань**.

Хоча *інші віджети QtVCP додадуть до цього списку*, віджет `screenOptions` додасть ці записи під такими заголовками:

[SCREEN_OPTIONS]

catch_errors ***(bool)*** **, desktop_notify** ***(bool)***
Чи відображати помилки/повідомлення в механізмі сповіщень системи.

notify_max_msgs ***(int)***
Кількість помилок, що відображаються одночасно.

shutdown_check ***(bool)***
Чи слід виводити діалогове вікно підтвердження.

sound_player_on ***(bool)***
Вмикає або вимикає всі звуки.

[MCH_MSG_OPTIONS]

mchnMsg_play_sound ***(bool)***
Відтворювати звуковий сигнал сповіщення, коли з’являється діалогове вікно.

mchnMsg_speak_errors ***(bool)***
Використання Espeak для озвучування повідомлень про помилки.

mchnMsg_speak_text ***(bool)***
Використовувати Espeak для озвучування всіх інших повідомлень.

mchnMsg_sound_type ***(str)***
Звук, який відтворюватиметься під час відображення повідомлень. Див. примітки нижче.

[USER_MSG_OPTIONS]

usermsg_play_sound ***(bool)***
Відтворювати звуковий сигнал сповіщення, коли з’являється діалогове вікно.

userMsg_sound_type ***(str)***
Звук, який відтворюватиметься під час відображення повідомлень користувача. Див. примітки нижче.

userMsg_use_focusOverlay ***(bool)***

[SHUTDOWN_OPTIONS]

shutdown_play_sound ***(bool)*** **, shutdown_alert_sound_type** ***(str)***
Звук, який відтворюватиметься під час відображення повідомлень. Див. примітки нижче.

shutdown_exit_sound_type ***(str)***
Звук, який відтворюватиметься під час відображення повідомлень. Див. примітки нижче.

shutdown_msg_title ***(str)***
Короткий заголовок для відображення в діалоговому вікні.

shutdown_msg_focus_text ***(str)***
Великий текстовий рядок для накладання на шар фокуса.

shutdown_msg_detail ***(str)***
Довший описовий рядок для відображення в діалоговому вікні.

NOTIFY_OPTIONS

notify_start_greeting ***(bool)***
Чи відображати діалогове вікно привітання під час запуску.

notify_start_title ***(str)***
Короткий рядок заголовка.
Якщо також вибрано опцію озвучування, заголовок буде озвучено мовою Espeak.

notify_start_detail ***(str)***
Довший рядок опису.

notify_start_timeout ***(int)***
Час у секундах, який відображатиметься перед закриттям.

*_sound_type записи

- **Системні звуки**
  В інсталяціях на базі Debian/Ubuntu/Mint ці *системні звуки* мають бути доступні як записи типу звуку вище:

  - ERROR
  - READY
  - DONE
  - ATTENTION
  - RING
  - LOGIN
  - LOGOUT
  - BELL

  Ці параметри звуку вимагають встановлення python3-gst1.0.

- **Аудіофайли**
  Ви також можете вказати *шлях до довільного аудіофайлу*.
  Ви можете використовувати ~ у шляху, щоб замінити шлях до домашнього файлу користувача.

- **Звукові сигнали ядра**
  Якщо модуль beep *kernel* встановлено і він не вимкнено, доступні такі записи звукових типів:

  - BEEP
  - BEEP_RING
  - BEEP_START

- **Перетворення тексту на мовлення**
  Якщо встановлено модуль *Espeak* (python3-espeak), ви можете використовувати запис SPEAK для вимовляння тексту:

- **SPEAK '_моє повідомлення_'**

12.7.2.26 StatusSlider - Віджет повзунка налаштування контролера

Цей віджет дозволяє користувачеві **налаштовувати налаштування LinuxCNC за допомогою повзунка**.

Віджет може налаштовувати:

- Швидкість штовхання
- Швидкість кутового поштовху
- Швидкість подачі
- Швидкість корекції шпинделя
- Швидка швидкість перевизначення

Властивості StatusSlider має такі властивості:

halpin_option
Встановлює опцію для створення HAL-виведення з плаваючою точкою, яке відображає поточне значення.

rapid_rate
Вибирає повзунок швидкості швидкого перевизначення.

feed_rate
Вибирає повзунок швидкості корекції подачі.

spindle_rate
Вибирає повзунок швидкості корекції шпинделя.

jograte_rate
Вибирає лінійний повзунок зі змінною довжиною.

jograte_angular_rate
Вибирає повзунок з кутовою візерунчастою решіткою.

max_velocity_rate
Вибирає повзунок максимальної швидкості.

alertState
Рядок для визначення зміни стилю: `read-only`, `under`, `over` та `normal`.

alertUnder
Встановлює значення числа з плаваючою комою, яке сигналізує таблиці стилів про попередження
«менше».

alertOver
Встановлює значення числа з плаваючою комою, яке сигналізує таблиці стилів про попередження
«перевищення».

Їх можна встановити в:

- Дизайнер Qt
- Код обробника Python,

```
self.w.status_slider.setProperty('spindle_rate',True)
self.w.status_slider.setProperty('alertUnder',35)
self.w.status_slider.setProperty('alertOver',100)
```

- Або (якщо доречно) у таблицях стилів.

```
/* b''кb''b''ob''b''лb''b''ьb''b''ob''b''pb''b''иb'' b''пb''b''ob''b''пb''b''eb''b''pb''b ↩
    ''eb''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b ↩
    ''pb''b''eb''b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''ьb'', b''яb''b ↩
    ''кb''b''щb''b''ob'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''вb''b ↩
    ''иb''b''хb''b''ob''b''дb''b''яb''b''тb''b''ьb'' b''зb''b''ab'' b''мb''b''eb''b''жb''b ↩
    ''ib'' b''нb''b''ob''b''pb''b''мb''b''ab''b''лb''b''ьb''b''нb''b''ob''b''гb''b''ob'' b ↩
    ''дb''b''ib''b''ab''b''пb''b''ab''b''зb''b''ob''b''нb''b''yb''*/
/* b''нb''b''ab''b''зb''b''вb''b''ab'' b''ob''b''бb''b''єb''b''кb''b''тb''b''ab'' b' ↩
    'вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ab'' b''—b'' slider_spindle_ovr */

#slider_spindle_ovr[alertState='over'] {
    background: red;
}
#slider_spindle_ovr[alertState='under'] {
    background: yellow;
}
```

Він базується на *QSlider* від PyQt.

12.7.2.27 StateLED - Віджет світлодіодного індикатора стану контролера

Цей віджет надає **статус вибраного стану LinuxCNC**.

Штати Варіанти штату такі:

is_paused_status , is_estopped_status , is_on_status , is_idle_status_ , is_homed_status , is_fl

Властивості Є властивості, які можна змінити:

halpin_option
: Додає вихідний контакт, який відображає вибраний стан.

invert_state_status
: Інвертувати стан світлодіода порівняно зі станом LinuxCNC.

diameter
: Діаметр світлодіода.

color
: Колір світлодіода у ввімкненому стані.

off_color
: Колір світлодіода у вимкненому стані.

вирівнювання
: Підказка щодо вирівнювання в Qt.

стан
: Поточний стан світлодіода (для тестування в Qt Designer).

миготливий
: Вмикає та вимикає опцію миготіння.

Швидкість спалаху
: Встановлює частоту спалахів.

Властивості світлодіода можна визначити в таблиці стилів за допомогою наступного коду, доданого до файлу `.qss`.

```
b''Cb''b''вb''b''ib''b''тb''b''лb''b''ob''b''дb''b''ib''b''ob''b''дb''_b''cb''b''тb''b' ↩
    'ab''b''нb''b''yb'' #name_of_led{       <t>\coref{1}{CO3-1}</t>
  qproperty-color: red;
  qproperty-diameter: 20;
  qproperty-flashRate: 150;
}
```

❶ `name_of_led` буде ім’ям, визначеним у редакторі Qt Designer.

Він базується на віджеті *LED*.

12.7.2.28 `StatusAdjustmentBar` - Віджет налаштування значень контролера

Цей віджет дозволяє **встановлювати значення за допомогою кнопок під час відображення панелі**.

Він також має *додаткову кнопку перемикання високого/низького рівня*, яку можна утримувати для встановлення **рівнів**.

Віджет може налаштовувати:

- Швидкість штовхання
- Швидкість кутового поштовху
- Швидкість подачі
- Швидкість корекції шпинделя
- Швидка швидкість перевизначення

Він базується на *QProgressBar* PyQt.

12.7.2.29 `SystemToolButton` - Віджет вибору системи користувача

Цей віджет дозволяє вам **вручну вибрати систему користувача G5x, натиснувши та утримуючи**.

Якщо ви не встановите текст кнопки, він автоматично оновиться до поточної версії системи.

Він базується на *QToolButton* PyQt.

12.7.2.30 `StateEnableGridlayout` - Віджет контейнера з увімкненим станом контролера

```
_b''вb''b''иb''b''мb''b''кb''b''нb''b''ib''b''тb''b''ьb'' b''вb''b''ib''b''дb''b''жb''b' ↩
    'eb''b''тb''b''иb'' b''вb''b''cb''b''eb''b''pb''b''eb''b''дb''b''иb''b''нb''b''ib'' b' ↩
    'нb''b''ьb''b''ob''b''гb''b''ob'' b''зb''b''ab''b''лb''b''eb''b''жb''b''нb''b''ob'' b' ↩
    'вb''b''ib''b''дb'' b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b''ob''b''гb''b''ob'' b' ↩
    'cb''b''тb''b''ab''b''нb''b''yb'' LinuxCNC_.
```

Це **контейнер, в який можна розміщувати інші віджети**.

Вбудовані віджети будуть сірими, коли `StateEnableGridlayout` вимкнено.

Він може вибірково реагувати на:

- Машина ввімкнена
- Інтерпретатор неактивний
- Аварійна зупинка вимкнена
- Усі домашні

Він базується на *QGridLayout* PyQt.

12.7.2.31 `StatusImageSwitcher` - Віджет перемикання зображень стану контролера

Цей віджет **відображатиме зображення на основі статусу LinuxCNC**.

Ви можете переглянути:

- стан шпинделя,
- стан усіх, хто має житло,
- стан певної осі, що знаходиться в головному положенні,
- стан жорстких обмежень.

Він базується на FIXME від PyQt

12.7.2.32 `ToolOffsetView` - Інструменти Зміщення Перегляд та редагування Віджет

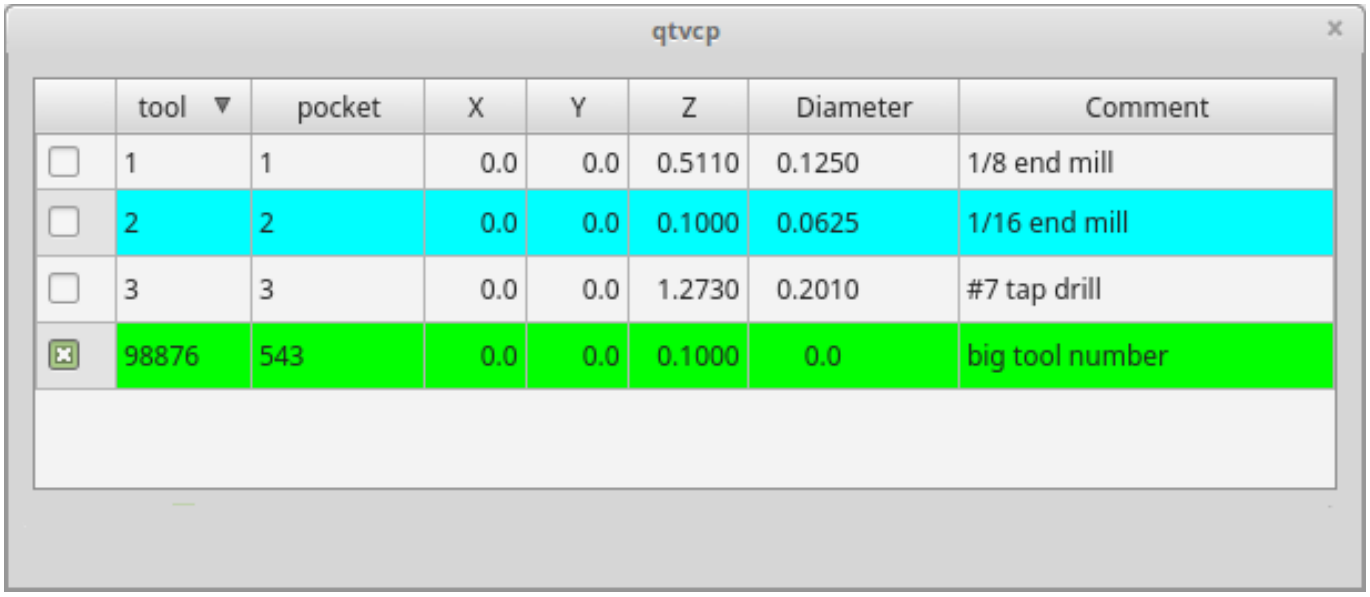


Figure 12.95: QtVCP `ToolOffsetView`: Інструменти Зміщення Перегляд та редагування Віджет

Цей віджет **відображає та дозволяє змінювати зміщення інструментів**.

Він *оновить таблицю інструментів LinuxCNC* для внесених або знайдених змін.

Налаштування інструмента можна змінити в LinuxCNC лише після повернення до початкового положення та коли контролер руху не використовується.

Відображення та введення змінюватимуться між метричною та імперською системами залежно від поточних налаштувань `G20`/`G21` LinuxCNC.

Поточний інструмент, що використовується, буде виділено, а поточний вибраний інструмент буде виділено іншим кольором.

Прапорець поруч із кожним інструментом можна використовувати для вибору дії, яка залежить від додаткового коду, доданого до комбінованого віджета, такого як діалогове вікно `toolOffsetView` або код обробника екранів.
Типовими діями є «завантажити вибраний інструмент», «видалити вибрані інструменти» тощо.

Клацання по стовпцях і рядках дозволяє налаштувати параметри.

Для введення даних або тексту можна налаштувати спливаюче діалогове вікно.

Розділ коментарів зазвичай відображається в діалоговому вікні ручної зміни інструменту.

Якщо використовується *конфігурація токарного верстата*, можуть бути стовпці для зносу по осях X та Z.

Щоб використовувати ці стовпці для налаштування *зносу інструменту*, потрібна переналаштована процедура зміни інструменту.

Він базується на *QTableView*, *QAbstractTableModel* та *ItemEditorFactory* PyQt.
Властивості, функції та стилі базових об'єктів PyQt завжди доступні.

Властивості `ToolOffsetView` має властивості, які можна встановити в Qt Designer, у коді обробника Python або (якщо це доречно) у таблицях стилів:

dialog_code_string
: Встановлює, яке діалогове вікно з'явиться для введення чисел.

text_dialog_code_string
: Встановлює, яке діалогове вікно з'явиться після введення тексту.

metric_template
: Формат числових даних у метричній системі.

imperial_template
: Імперський числовий формат даних.

styleCodeHighlight
: Поточний колір виділення інструменту, що використовується.

styleCodeSelected
: Вибраний колір виділення.

У файлі обробника:

```
self.w.tooloffsetview.setProperty('dialog_code','CALCULATOR')
self.w.tooloffsetview.setProperty('metric_template','%10.3f')
```

і в таблицях стилів:

```
ToolOffsetView{
  qproperty-styleColorHighlist: lightblue;
  qproperty-styleColorSelected: #444;
}
```

Функції `ToolOffsetView` має деякі функції, корисні для розробників екранів для додавання дій:

add_tool()
: Додає порожній інструмент-заповнювач (99), який користувач може редагувати відповідно до своїх потреб.

delete_tools()
: Видаляє інструменти, вибрані з поточного прапорця.

get_checked_list()
: Повертає список інструментів, вибраних за допомогою прапорців.

set_all_unchecked()
: Зніміть позначки з усіх вибраних інструментів.

Приклад файлу-обробника, що виконує вищезгадані функції.

```
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` - Віджет зондування фрезерування

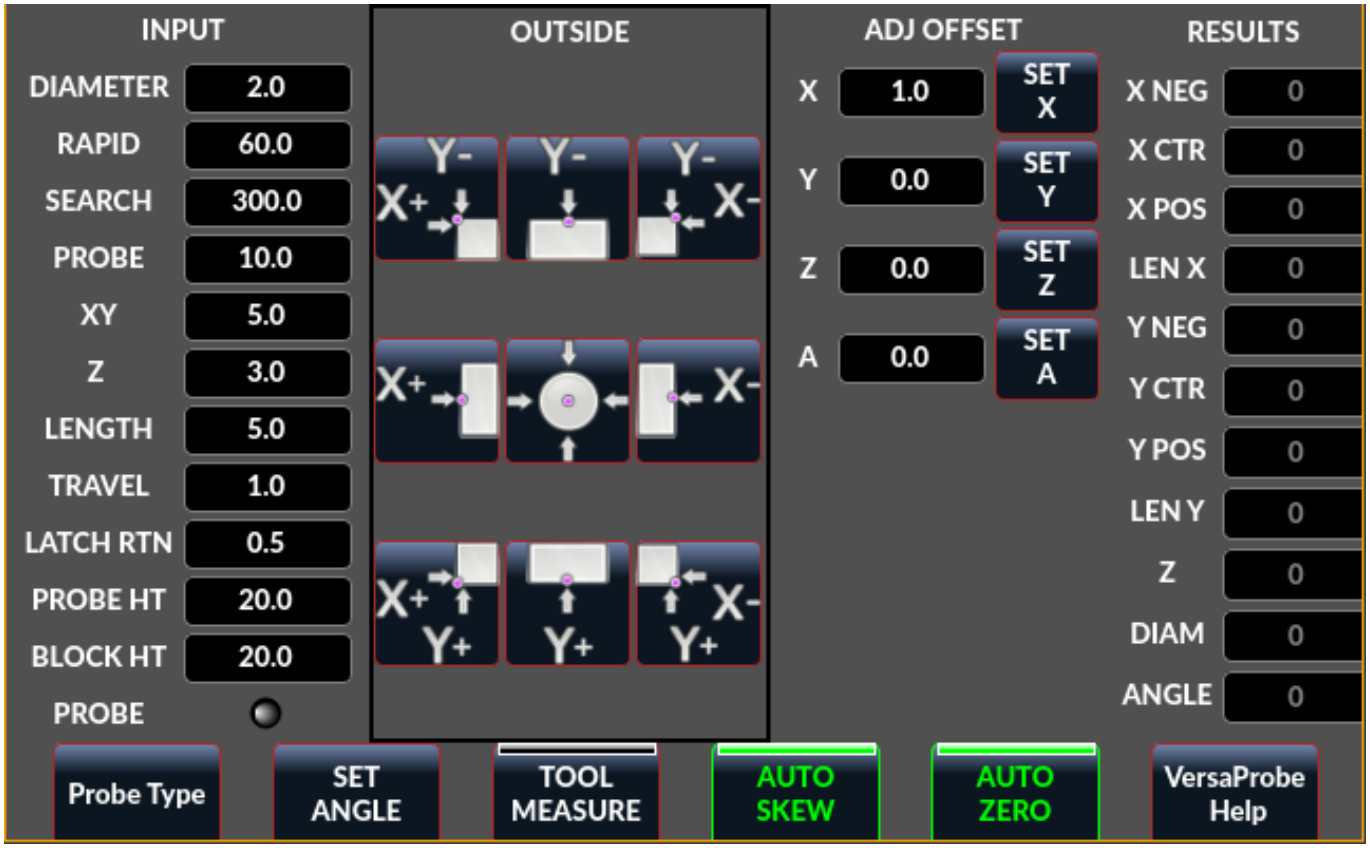


Figure 12.96: QtVCP `VersaProbe`: Віджет зондування фрезерування

Віджет для **зондування на фрезерному верстаті**. Використовується екраном *QtDragon*.

12.7.3 Віджети діалогових вікон

Діалогові вікна використовуються для **презентації або запиту негайно необхідної інформації** цілеспрямованим чином.

Типові діалогові вікна, що використовуються, можна завантажити за допомогою віджета `ScreenOption`

Ви також можете додати їх безпосередньо до *UI*, але кожне діалогове вікно повинно мати унікальну назву запуску, інакше ви побачите кілька діалогових вікон, що відображатимуться одне за одним.

Використовуйте діалоги з коду Python Ви можете відображати діалогові вікна безпосередньо за допомогою *коду Python*, але безпечніший спосіб — **використовувати повідомлення `STATUS`** для запиту запуску діалогового вікна та повернення зібраної інформації.

- **Зареєструйтесь на каналі `STATUS`**:
  Щоб налаштувати це, спочатку зареєструйтесь, щоб отримувати повідомлення `general` від `STATUS`:

```
STATUS.connect('general',self.return_value)
```

- **Додати функцію для виклику діалогового вікна**:
  Ця функція повинна *створити повідомлення `dict` для відправки в діалогове вікно*.
  Це повідомлення буде передано назад у загальному повідомленні з додаванням `*повернення змінної*.
  До повідомлення можна додати *додаткову інформацію про користувача*. Діалогове вікно проігнорує її і передасть назад.

  `NAME`
  Запускає кодову назву діалогового вікна для відображення.

  `ID`
  Унікальний ідентифікатор, щоб ми обробляли лише запитуваний нами діалог.

  `TITLE`
  Заголовок, який буде використано в діалоговому вікні.

```
def show_dialog(self):
    mess = {'NAME':'ENTRY','ID':'__test1__',
            'TITLE':'Test Entry'}
    ACTION.CALL_DIALOG, mess)
```

- **Додайте функцію зворотного виклику, яка обробляє загальне повідомлення**:
  Майте на увазі, що ця функція отримає всі загальні повідомлення, тому не гарантується, що ключові імена `dict` будуть там. Рекомендується використовувати функцію `.get()` та/або `try`/`except`.
  Ця функція повинна:

  - перевірте, чи ім'я та ідентифікатор такі ж, як ми надіслали,
  - потім витягніть повернене значення та будь-які змінні користувача.

```
# b''ob''b''6b''b''pb''b''ob''b''6b''b''ив''b''тb''b''ив'' b''пb''b''ob''b''вb''b''ib''b' ↩
    'дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яв'' b''пb''b''pb''b''ob'' b''пb''b' ↩
    'ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''яв'' STATUS
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` - Віджет загального діалогового вікна повідомлень

Це **загальний віджет діалогового вікна повідомлень**.

Якщо присутній віджет Focus Overlay, він може сигналізувати про його відображення.

Якщо бібліотеку звуків налаштовано, вона може *відтворювати звуки*.

Існують *параметри*, які можна встановити під час запиту діалогу, вони будуть додані до повідомлення `dict`.

`TITLE`
Заголовок діалогового вікна.

MESSAGE
Текст заголовка повідомлення виділено жирним шрифтом.

MORE
Стандартний текст під заголовком.

DETAILS
Початковий прихований текст.

TYPE *(OK|YESNO|OKCANCEL)* , ICON (QUESTION|INFO|CRITICAL|WARNING) , PINNAME
Ще не впроваджено.

FOCUSTEXT *(overlay text|None)*
Текст, який буде відображатися, якщо використовується накладання фокуса. Використовуйте «Немає», якщо тексту немає.

FOCUSCOLOR (QColor(_R, G, B, A_))
Колір, який буде використано, якщо використовується накладання фокуса.

PLAYALERT
Звук, який відтворюватиметься, якщо звук доступний, наприклад, SPEAK <*розмовлене_повідомлен*
.

Під час використання функції `request-dialog` класу `STATUS`, *ім'я запуску за замовчуванням* — **`MESSAGE`**.

Він базується на *QMessagebox* PyQt.

12.7.3.2 `ToolDialog` - Віджет діалогового вікна ручної зміни інструментів

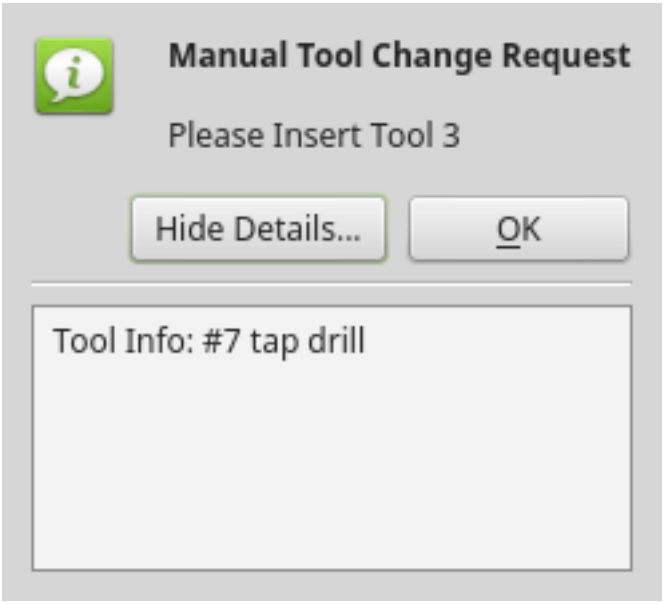


Figure 12.97: QtVCP `ToolDialog`: Діалогове вікно ручної зміни інструменту

Це використовується як **підказка для ручної зміни інструменту**.

Він має *виводи HAL для підключення до контролера верстата*. Виводи мають назви та ж самі, що й у оригінальному інструкційному посібнику AXIS, і працюють так само.

Діалогове вікно зміни інструменту *можна запустити лише за допомогою контактів HAL*.

Якщо присутній віджет Focus Overlay, він сповістить про його відображення.

Він базується на *QMessagebox* PyQt.

12.7.3.3 `FileDialog` - Віджет діалогу вибору файлів для завантаження та збереження

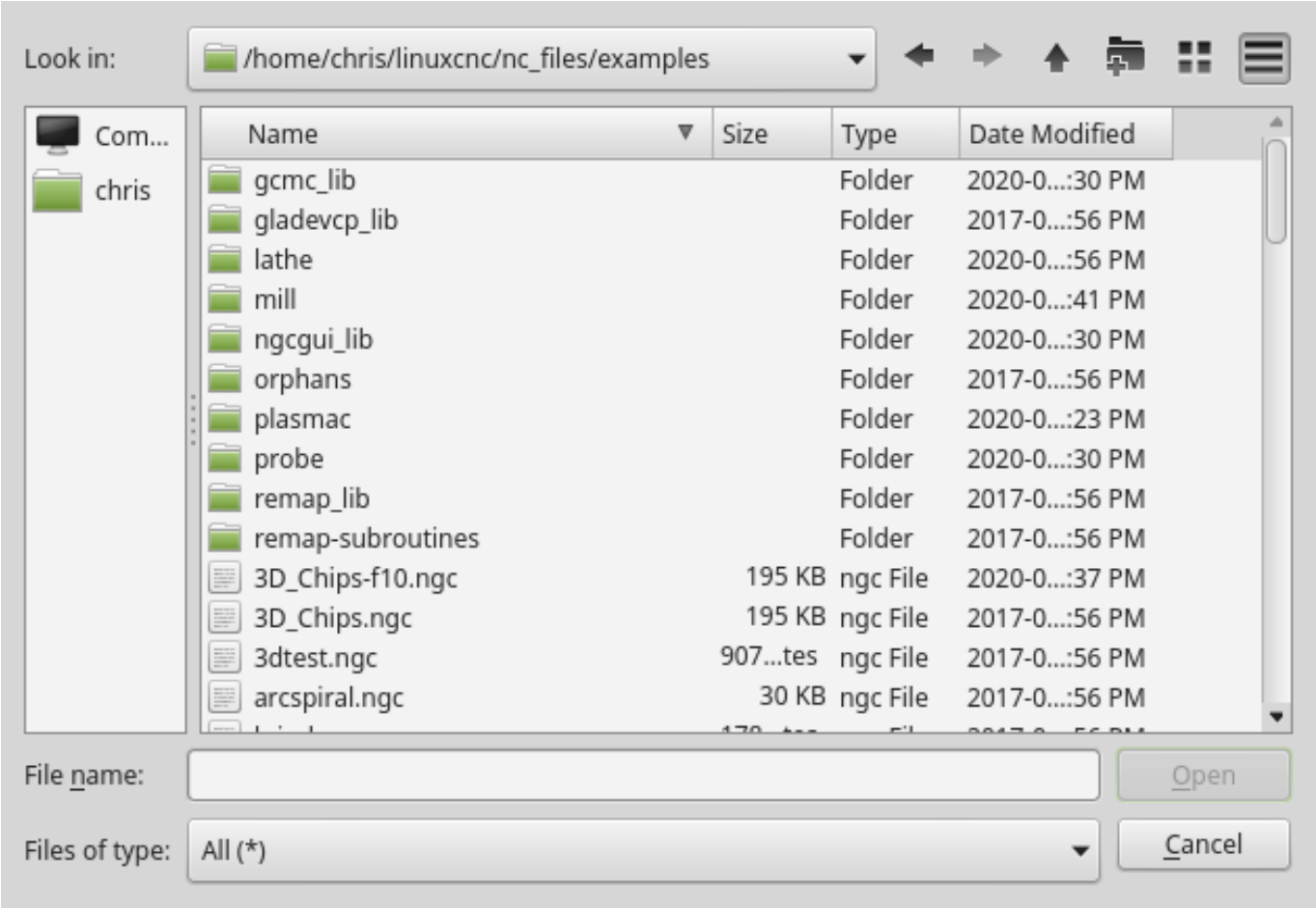


Figure 12.98: QtVCP `FileDialog`: Віджет вибору файлів для завантаження та збереження

Це використовується для **завантаження файлів G-коду**.

Якщо присутній віджет Focus Overlay, він сповістить про його відображення.

Під час використання функції `request-dialog` класу `STATUS`, назви запуску за замовчуванням - **`LOAD`** або **`SAVE`**.

Існують *параметри*, які можна встановити під час запиту діалогу, вони будуть додані до словника повідомлення:

`EXTENSIONS` , **`FILENAME`** , **`DIRECTORY`**

Приклад виклику Python для *завантаження діалогу*:

```
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)
```

А також для *діалогового вікна збереження*

```
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)
```

Він базується на *QMessagebox* PyQt.

12.7.3.4 `OriginOffsetDialog` - Віджет діалогового вікна налаштування зміщення походження

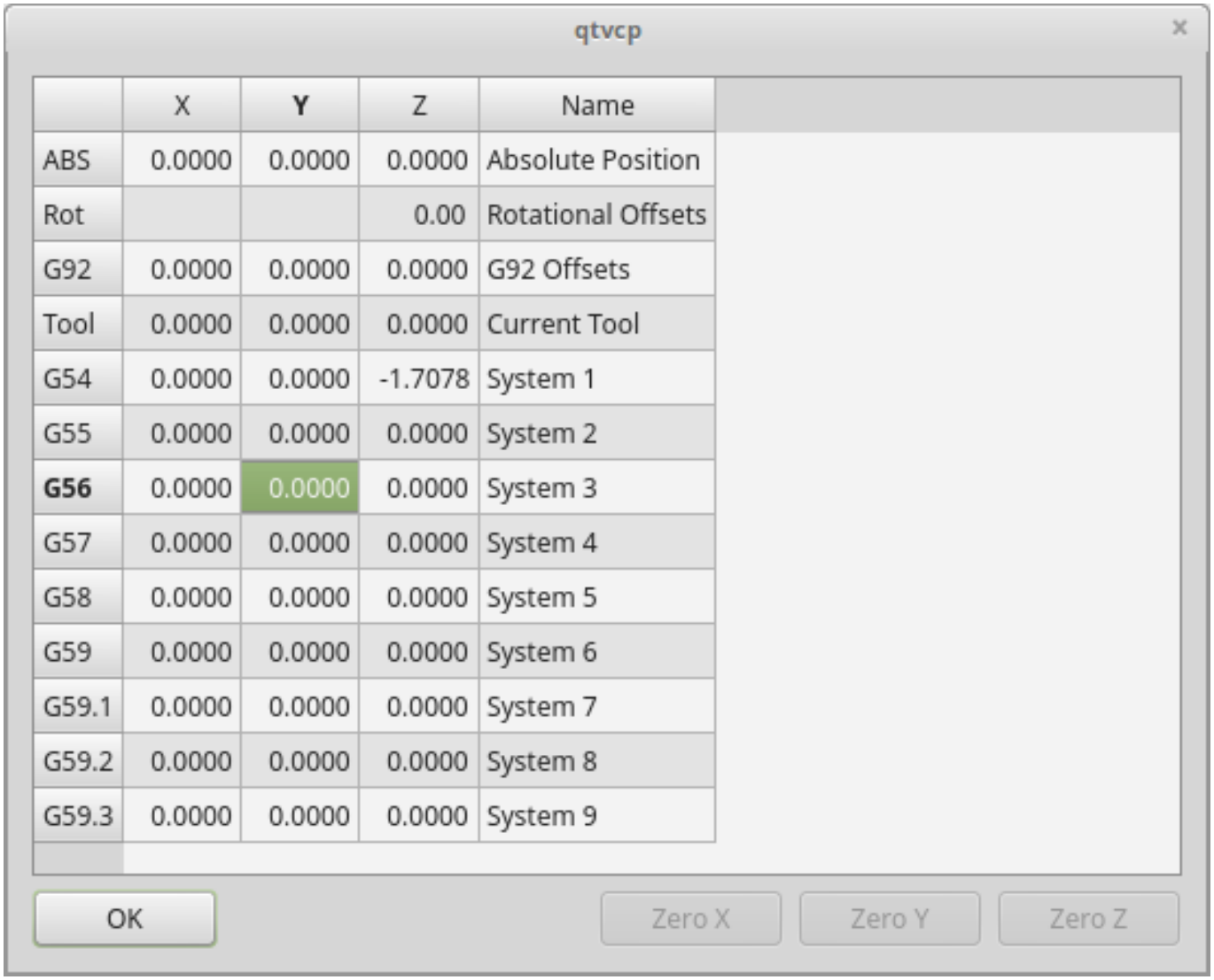


Figure 12.99: QtVCP `OriginOffsetDialog`: Віджет налаштування зміщення походження

Цей віджет дозволяє **безпосередньо змінювати зміщення походження системи користувача** у діалоговому вікні.

Якщо присутній віджет Focus Overlay, він сигналізує про його відображення.

Під час використання функції `request-dialog` класу `STATUS`, ім’я запуску за замовчуванням — **`ORIGINOFFSET`**.

Він базується на *QDialog* від PyQt.

12.7.3.5 `ToolOffsetDialog` - Віджет діалогового вікна налаштування зміщення інструмента

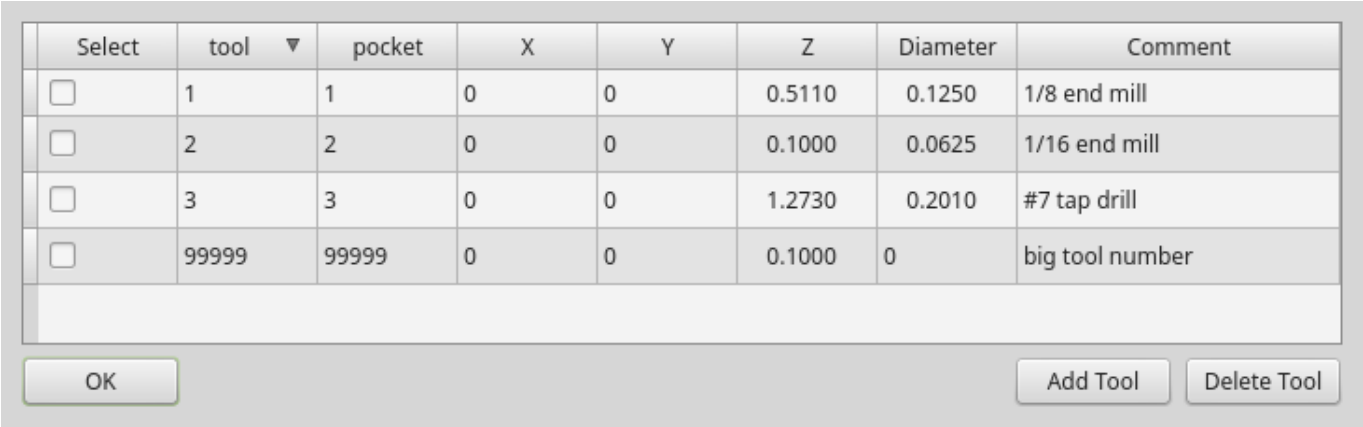


Figure 12.100: QtVCP `ToolOffsetDialog`: Віджет діалогового вікна налаштування зміщення інструмента

Цей віджет дозволяє **змінювати зміщення інструменту безпосередньо** у діалоговому вікні.

Якщо присутній віджет Focus Overlay, він сигналізує про його відображення.

Під час використання функції `request-dialog` класу `STATUS`, ім’я запуску за замовчуванням — **`TOOLOFFSET`**.

Він базується на *QDialog* від PyQt.

12.7.3.6 `ToolChooserDialog` - Віджет діалогового вікна вибору інструментів

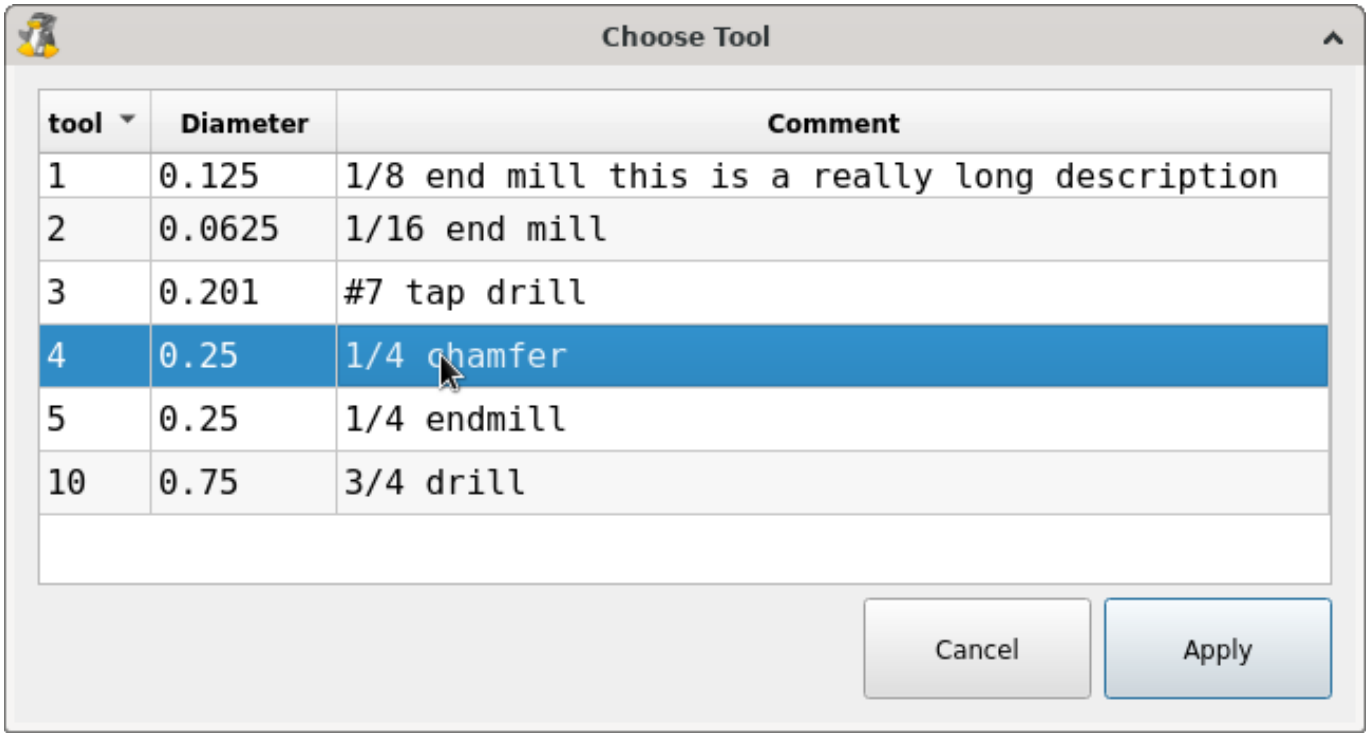


Figure 12.101: QtVCP `ToolChooserDialog`: Віджет діалогового вікна вибору інструментів

Цей віджет дозволяє оператору вибрати один з інструментів, визначених у таблиці інструментів. Якщо інструмент вибрано і натиснуто кнопку «Застосувати» або двічі клацнуто на інструменті, діалогове вікно поверне номер вибраного інструменту. Це можна використовувати разом з віджетом `OperatorValueLine`, наприклад, для створення віджета зміни інструменту.

Якщо присутній віджет Focus Overlay, він сигналізує про його відображення.

Під час використання функції `request-dialog` класу `STATUS`, ім'я запуску за замовчуванням — **`TOOLCHOOSER`**.

Він базується на *QDialog* від PyQt.

12.7.3.7 `MachineLog` - Віджет відображення журналу подій машини

Цей віджет відображає різні **повідомлення про події журналу**, які були виведені системою протягом поточного сеансу. Це включає інформаційні повідомлення, а також помилки.

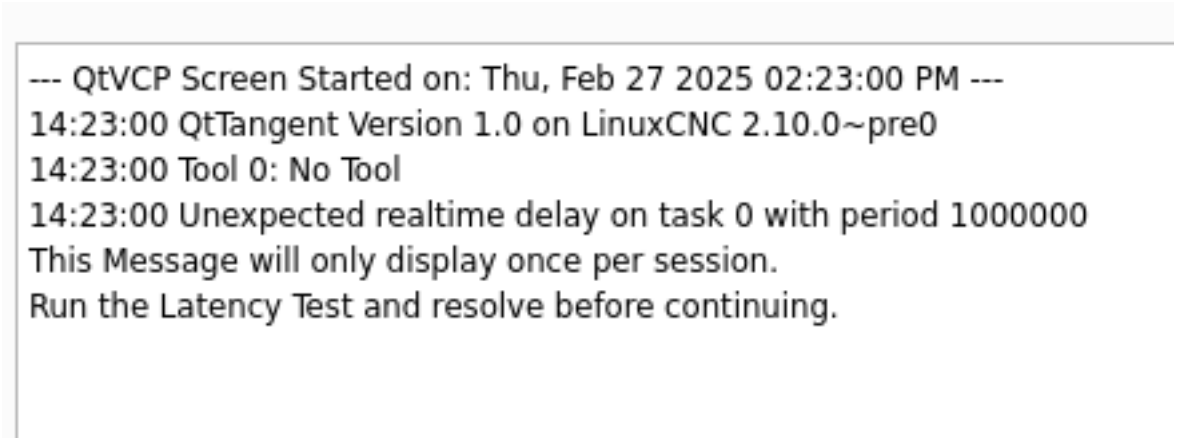


Figure 12.102: QtVCP `MachineLog`: Журнал машинних подій у режимі `machine_log` (звичайний)

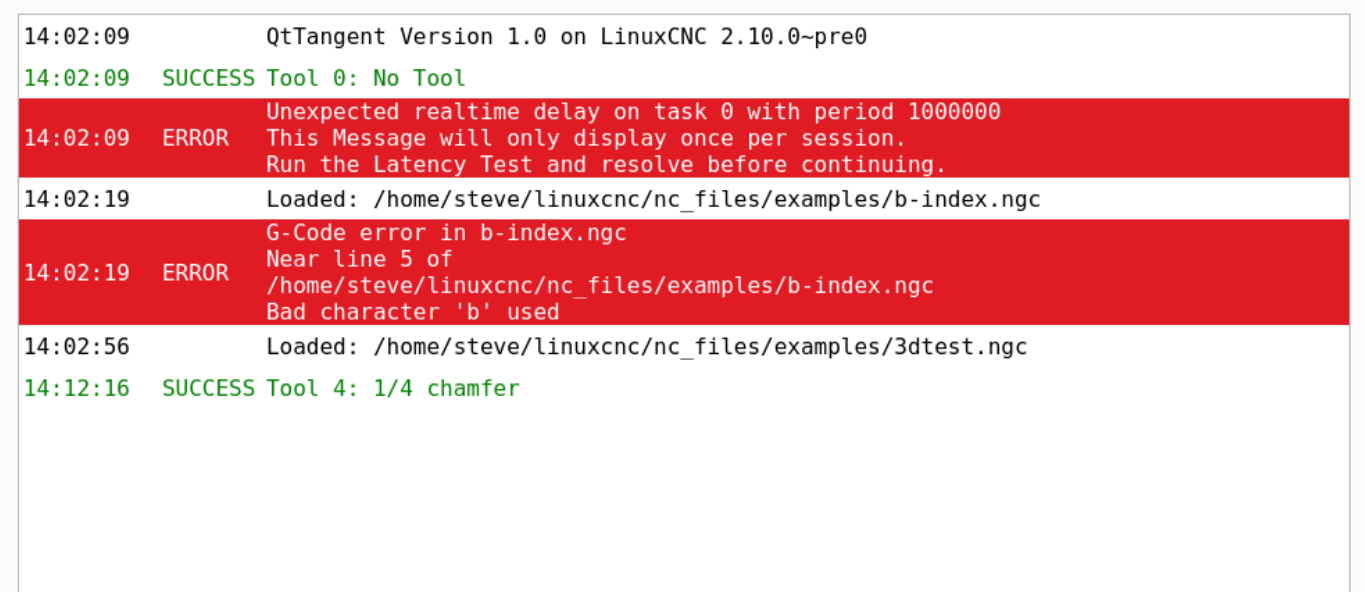


Figure 12.103: QtVCP `MachineLog`: Журнал машинних подій у режимі `machine_log_severity`

Можуть відображатися два різних типи журналів:

- журнал машини (звичайний текст або виділено рівень серйозності)
- журнал інтегратора (лише звичайний текст)

Тип журналу, що відображається віджетом, контролюється властивостями опції віджета. Вибравши `machine_log_option` або `integrator_log_option`, буде відображено відповідний журнал. Ці опції відображатимуть журнали у простому стилі у віджеті Qt `QTextEdit`.

Крім того, можна вибрати властивість `machine_log_severity_option`, яка відображатиме журнал машини різними кольорами залежно від важливості повідомлення, використовуючи `QTableWidget`. Кольори можна налаштувати за допомогою властивостей віджета.

Серйозність передається через значення `option`, яке надсилається разом із сигналом `STATUS` під назвою `update-machine-log`. Параметр `option` – це список, розділений комами, який зазвичай містить

``` text = *an error has occurred.* STATUS.emit(*update-machine-log*, text, *TIME,ERROR*) ```

Журнал можна очистити, викликавши метод `clear()` віджета.

12.7.3.8 `MacroTabDialog` - Віджет діалогового вікна запуску макросів

Це діалогове вікно для **відображення віджета макровкладки**.

`MacroTab` відображає *вибір макропрограм для запуску за допомогою піктограм*.

Якщо присутній віджет Focus Overlay, він сповістить про його відображення.

Під час використання функції `request-dialog` класу ``STATUS``, ім’я запуску за замовчуванням — **`MACROTAB`**.

Він базується на *QDialog* від PyQt.

12.7.3.9 `CamViewDialog` - Віджет діалогового вікна вирівнювання деталей веб-камери

Це діалогове вікно для **відображення віджета CamView для вирівнювання деталей веб-камери**.

Під час використання функції `request-dialog` класу ``STATUS``, ім’я запуску за замовчуванням — **`CAMVIEW`**.

Він базується на *QDialog* від PyQt.

12.7.3.10 `EntryDialog` - Віджет діалогового вікна редагування рядка

Це діалогове вікно для **відображення рядка редагування для введення інформації**, такої як зміщення початку координат.

Він повертає запис через повідомлення `STATUS`, використовуючи Python `DICT`.

`DICT` містить щонайменше назву запитуваного діалогу та ідентифікаційний код.

Під час використання функції `request-dialog` класу ``STATUS``, ім’я запуску за замовчуванням — **`ENTRY`**.

Він базується на *QDialog* від PyQt.

12.7.3.11 `CalculatorDialog` - Віджет діалогового вікна калькулятора

Це діалогове вікно для **відображення калькулятора для введення чисел**, таких як зміщення від початку координат, оберти шпинделя тощо. Воно призначене в першу чергу для використання на сенсорному екрані, але також підтримує введення з фізичної клавіатури.
```

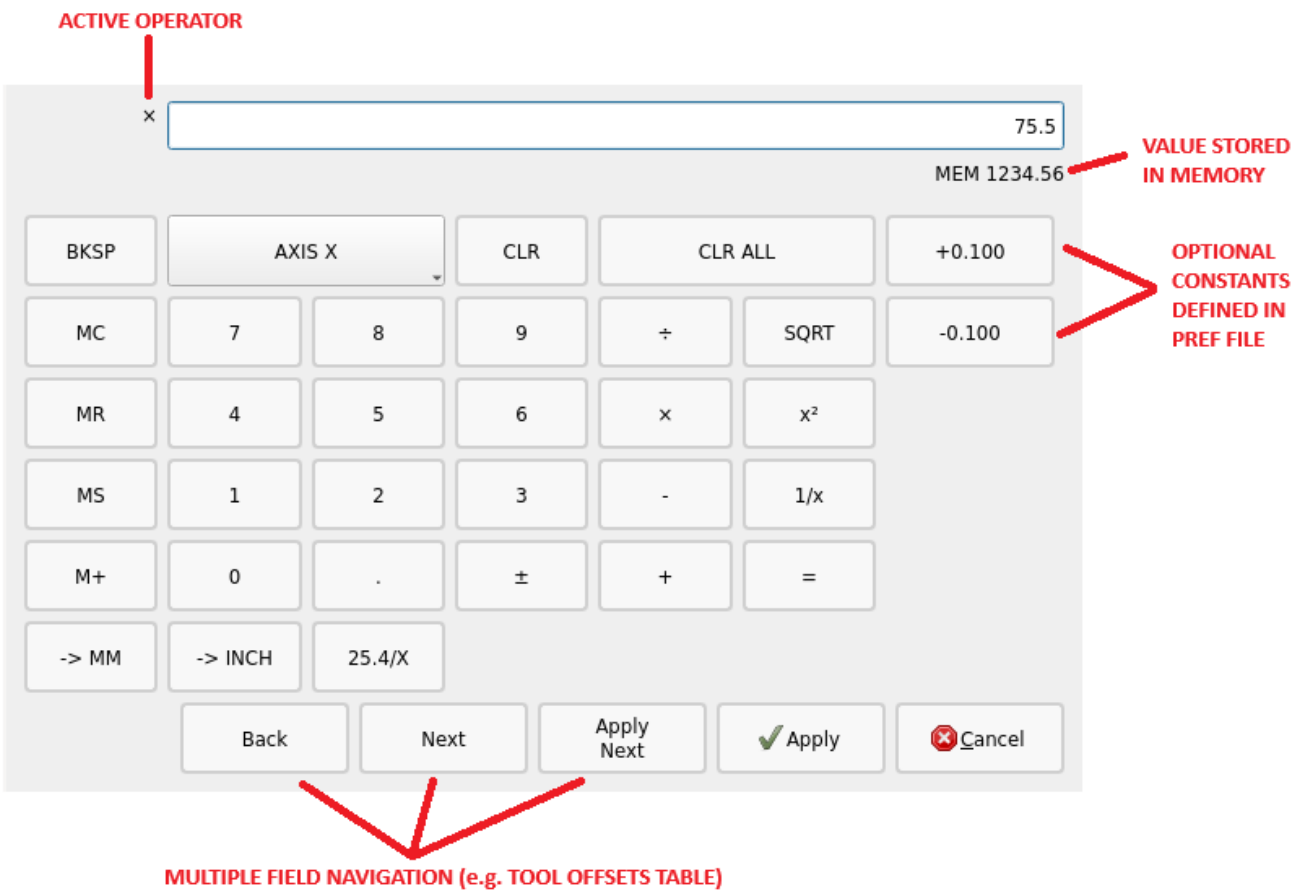


Figure 12.104: QtVCP `CalculatorDialog`: Віджет діалогового вікна калькулятора

Він повертає запис через повідомлення `STATUS`, використовуючи Python `DICT`.

`DICT` містить щонайменше назву запитуваного діалогу та ідентифікаційний код.

Під час використання функції `request-dialog` класу ``STATUS``, ім'я запуску за замовчуванням — **`CALCULATOR`**.

Він базується на *QDialog* від PyQt.

У розділі `CALCULATOR` файлу налаштувань можна встановити такі параметри:

- `constValuesList` - список загальних значень, розділених комами, які ви можете ввести і які з'являться на спеціальному рядку кнопок внизу калькулятора. Наприклад, при встановленні значення `0.100, -0.100` з'являться дві кнопки для +0.100 і -0.100, які часто використовуються при пошуку країв на дюймових фрезах. Можна ввести до шести (6) значень, після чого список буде обрізаний. Значення повинні бути дійсними числами з плаваючою комою або цілими числами.
- `onShowBehavior` - Список необов'язкових дій, які будуть активовані, коли відображається діалогове вікно калькулятора. Кожну опцію потрібно розділяти комою.
  - `CLEAR_ALL` видавати команду **Очистити все** щоразу, коли відображається калькулятор. Це очистить усі попередньо введені значення з моменту останнього використання калькулятора та відкриє його зі значенням відображення, встановленим на `0`

  - `FORCE_FOCUS` примусово фокусує поле введення калькулятора, коли відображається віджет. Це дозволить фізичній клавіатурі правильно вводити дані у віджет без додаткових кліків. Також це має побічний ефект вибору поточного значення, так що введення з фізичної клавіатури замінить існуюче значення, якщо вибір тексту не буде змінено.

- `acceptOnReturnKey` - Якщо встановлено значення `True`, калькулятор прийме поточне значення і закриє діалогове вікно при натисканні клавіші повернення/введення на клавіатурі. Якщо встановлено значення `False`, клавіша повернення буде проігнорована і необхідно натиснути кнопку **Застосувати**. У випадках, коли доступна кнопка **Застосувати наступне**, клавіша повернення виконає цю дію, а діалогове вікно залишиться відкритим.

Хоча метою цього віджета є забезпечення зручного інтерфейсу для сенсорного екрану, для введення значень можна використовувати фізичну клавіатуру, зазвичай цифрову. Клавіші працюють в основному так, як і слід очікувати, але є також кілька спеціальних функцій клавіш:

- Клавіша **Enter** або **Return** еквівалентна оператору **Equal** (=) у випадках, коли обчислення знаходиться в стані очікування. В іншому випадку вона виконуватиме функцію **Apply**, якщо вона ввімкнена через налаштування `acceptOnReturnKey`.
- Клавіша «мінус» (-) змінює знак поточного числа на дисплеї калькулятора, якщо натиснути її двічі поспіль. В іншому випадку, якщо натиснути її тільки один раз, вона виконає операцію віднімання, як і очікується.
- **Alt+стрілка вліво** виконає функцію **Назад**, переходячи до попереднього поля, якщо це підтримуєть
- **Alt+Стрілка вправо** виконає функцію **Далі**, переходячи до наступного поля, якщо це підтримується
- **Alt+Backspace** призведе до виходу з калькулятора та не поверне значення для викликаючого віджета.

12.7.3.12 `RunFromLine` - Віджет діалогового вікна «Запустити з рядка»



Figure 12.105: QtVCP `RunFromLine`: Віджет діалогового вікна «Запустити з рядка»

Діалогове вікно для **попереднього встановлення параметрів шпинделя перед запуском програми з певного рядка**.

Він базується на *QDialog* від PyQt.

12.7.3.13 `VersaProbeDialog` - Віджет діалогового вікна дотику до деталі

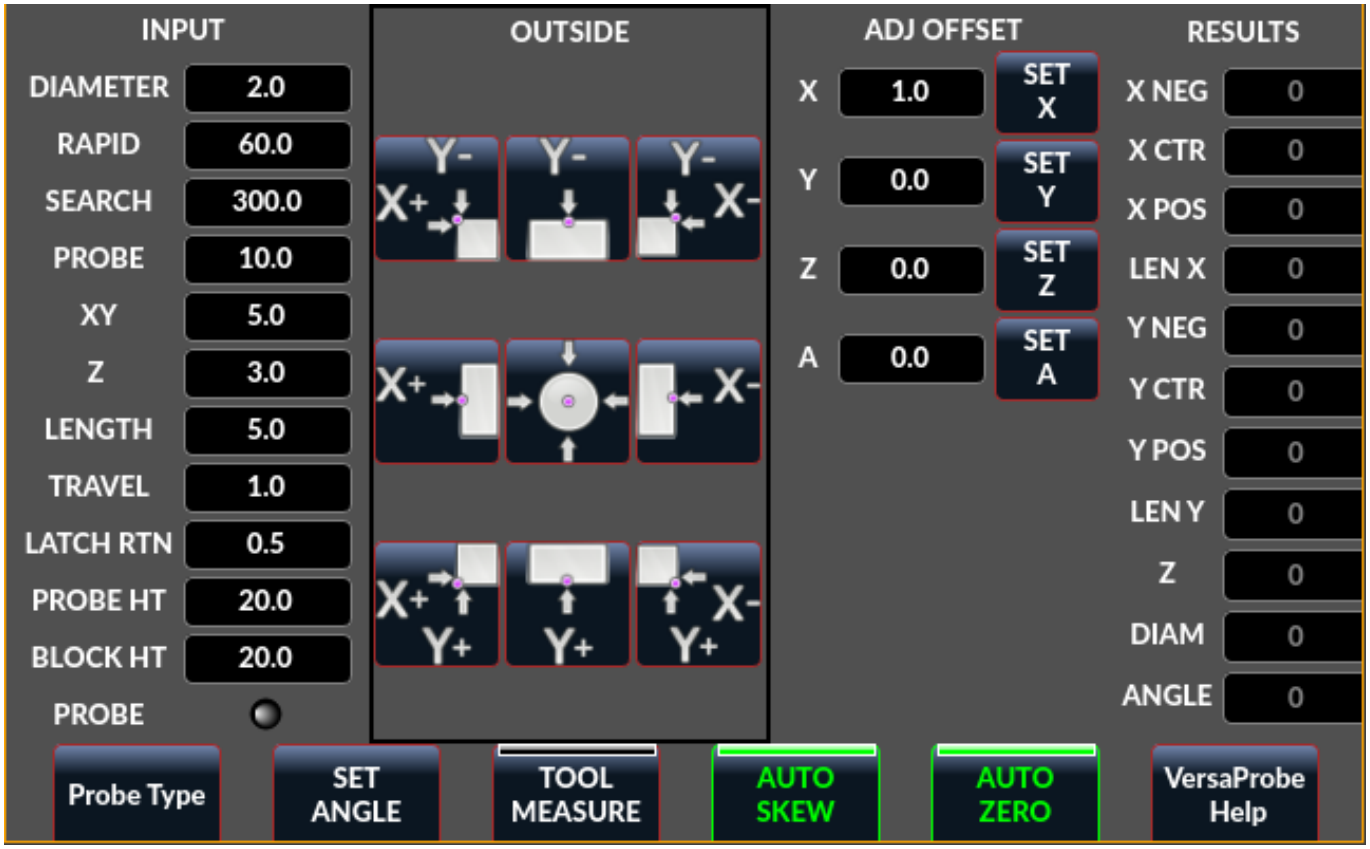


Figure 12.106: QtVCP `VersaProbeDialog`: Віджет діалогового вікна дотику до деталі

Це діалогове вікно для відображення **екрана зондування деталі на основі Verser Probe v2**. Він базується на *QDialog* від PyQt.

12.7.3.14 `MachineLogDialog` - Віджет діалогового вікна журналів машини та налагодження

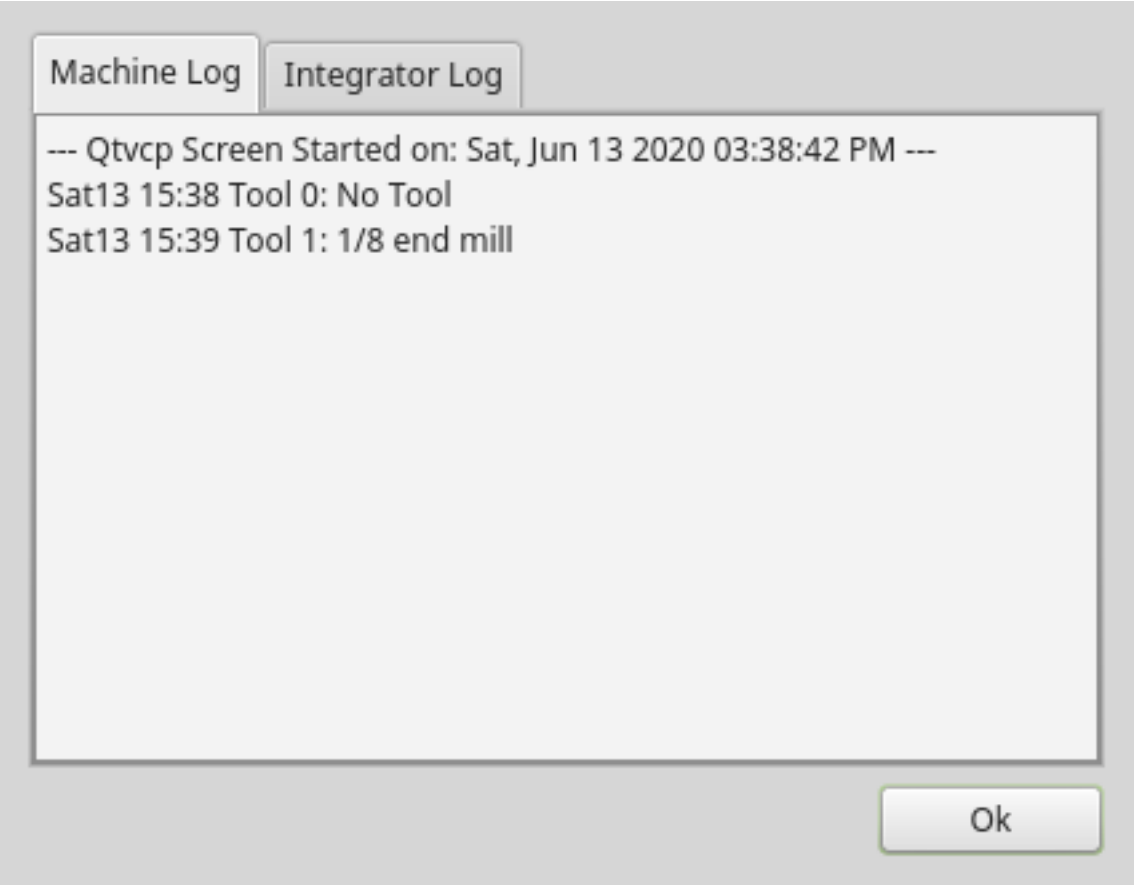


Figure 12.107: QtVCP `MachineLogDialog`: Віджет діалогового вікна журналів машини та налагодження

Це діалогове вікно для **відображення журналу машини та журналу налагодження QtVCP**.

Він базується на *QDialog* від PyQt.

12.7.4 Інші віджети

Інші доступні віджети:

12.7.4.1 `NurbsEditor` - Віджет редагування NURBS

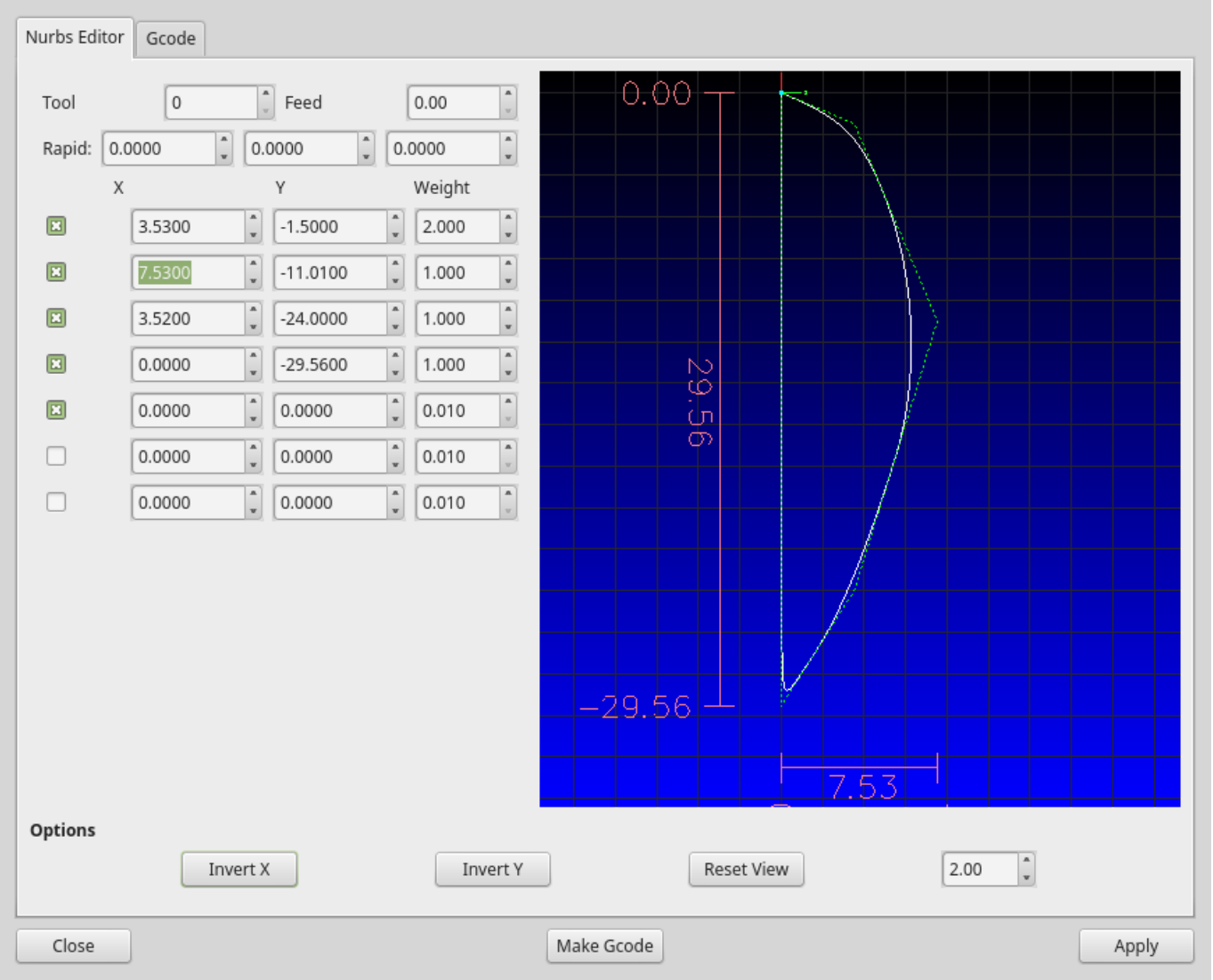


Figure 12.108: QtVCP `NurbsEditor`: Віджет редагування NURBS

Редактор NURBS дозволяє вам **маніпулювати геометрією на основі NURBS** на екрані, а потім **конвертувати NURBS у G-код**.

Ви можете редагувати G-код на екрані, а потім надіслати його до LinuxCNC.

Він базується на *QDialog* від PyQt.

12.7.4.2 `JoyPad` - 5-кнопковий віджет D-pad

Це базовий клас для віджета `HALPad`.

Цей віджет виглядає та працює як **5-кнопковий D-pad зі світлодіодними індикаторами в кільці**.

Ви можете розмістити текст або значки в кожній з позицій кнопок.

Ви можете *підключитися до вихідних сигналів*, натиснувши кнопки.

Також є *слоти введення* для зміни кольору індикатора(ів).

ENUMS Існують **перелічувані константи, що використовуються для посилання на позиції індикаторів**.
Вони використовуються в редакторі властивостей редактора Qt Designer або в коді Python.

NONE , LEFT, L , RIGHT, R , CENTER, C , TOP, T , BOTTOM, B , LEFTRIGHT, X , TOPBOTTOM, A
: Для коду обробника Python ви використовуєте назву віджета в Qt Designer плюс константу посилання:

```
self.w.joypadname.set_highlight(self.w.joypadname.LEFT)
```

Корисні функції, які можна перевизначити

```
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)
```

Згідно з кодом, ці функції *issue (випромінюють) сигнали PyQt5 (joy_btn_pressed та joy<*літера*>_presse*
для будь-якого натискання або відпускання кнопки_.

Сигнал *joy_btn_pressed* виводить рядковий код для кнопки.
Сигнал *joy_<літера>_pressed* виводить логічне значення.

Ви можете перевизначити функції, щоб вони виконували щось інше, якщо створюєте власний віджет:

Функції, що викликаються

reset_highlight()
: Очищає індикатор виділення.

set_highlight(_button_, state=_True_)
: Встановіть індикатор підсвічування в позиції `button` у стан `state`.
Ви можете використовувати *рядки_літер* (LRCTBXA) або *позицію* ENUMS як аргумент кнопки.

set_button_icon(_button_, _pixmap_)
: Встановлює розпізнавальну карту значка кнопки.

set_button_text(_button_, _text_)
: Встановлює текст значка кнопки.

set_tooltip(_button_, _text_)
: Встановлює описовий текст спливаючої підказки для кнопок.

setLight(_state_)
: Встановлює для індикатора підсвічування колір `True` або `False`.
Функцію `set_highlight()` необхідно використовувати перед встановленням індикатора.

Сигнали Ці сигнали будуть **надсилатися при натисканні кнопок**.

До них можна підключитися в редакторі Qt Designer або в коді Python.

Перші два виводять рядок, який вказує на натиснуту кнопку:

joy_btn_pressed *(string)* , **joy_btn_released *(string)*** , **joy_l_pressed *(bool)*** , **joy_l_released *(boo***

Вони базуються на *Signal* від PyQt (`QtCore.pyqtSignal()`)

Слоти До слотів можна підключитися в редакторі Qt Designer або коді Python:

set_colorStateTrue() , **set_colorStateFalse()** , **set_colorState(_bool_)** , **set_true_color(_str_)**

Властивості Їх можна встановити в таблицях стилів або коді Python:

highlightPosition
: Встановіть положення індикатора.

setColorState
: Виберіть колірний стан індикатора.

left_image_path , **right_image_path** , **center_image_path** , **top_image_path** , **bottom_image_path**

: Шлях до файлу або ресурсу зображення, яке буде відображатися в зазначеному місці кнопки. Якщо в редакторі Qt Designer натиснути кнопку «Скинути», зображення не буде відображатися (замість нього можна відобразити текст).

left_text , **right_text** , **center_text** , **top_text** , **bottom_text**
: Текстовий рядок, який буде відображено в описаному місці кнопки.
Якщо залишити поле порожнім, можна призначити зображення для відображення.

true_color , **false_color**
: Вибір кольору для центрального світлодіодного кільця, яке буде відображатися, коли контакт `BASENAME.light.center` *HAL* має значення `True` або `False`.

text_color
: Вибір кольору для тексту кнопки.

button_font
: Вибір шрифту для тексту кнопки.

Вищезазначені властивості можна встановити в:

- **Таблиці стилів**:
  Зазвичай для встановлення окремих властивостей віджетів використовується ім’я віджета Qt Designer з префіксом #, в іншому випадку для встановлення однакових властивостей для всіх віджетів `JoyPad` використовується ім’я класу `JoyPad`:

```
#joypadname{
  qproperty-true_color: #000;
  qproperty-false_color: #444;
}
```

- **У коді обробника Python**:

```
self.w.joypadename.setProperty('true_color','green')
self.w.joypadename.setProperty('false_color','red')
```

12.7.4.3 *WebWidget*

Цей віджет створить сторінку перегляду html/pdf за допомогою бібліотек QtWebKit або QtWebEngine.
Якщо в системі є обидві бібліотеки, краще використовувати нову QtWebEngine.
Якщо бібліотека QtWebEngine використовується з редактором Qt Designer, в Qesigner з'явиться замінник QWidget. Він буде замінений віджетом QtWebEngine під час виконання.

12.7.5 Віджети BaseClass/Mixin

Ці віджети використовуються для **поєднання різних властивостей та поведінки в інші віджети**.

Ви побачите їх як згортальний заголовок у стовпці властивостей Qt Designer.

12.7.5.1 `IndicatedPushButtons`

Цей клас **змінює поведінку `QPushButton`**.

`indicator_option` *розміщує «світлодіод» у верхній частині кнопки*.

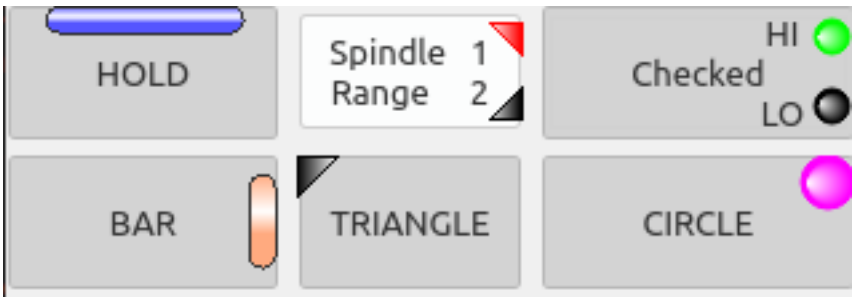


Figure 12.109: QtVCP `PushButton`: Кнопкова індикація, варіант світлодіодного індикатора

Це може бути *трикутник*, *коло*, *верхня смуга* або *бічна смуга*.
Світлодіоди *трикутник* і *коло* можуть відображати подвійні вертикальні світлодіоди, кожен з яких має власний контакт HAL. + Розмір і положення можна регулювати.

Це вказуватиме:

- **поточний стан кнопки**, або
- **стан виводу HAL**, або
- **Статус LinuxCNC**.

Властивості Ці властивості доступні для налаштування індикатора (не всі вони застосовні до кожної форми світлодіода):

`doubleIndicator`
: До трикутних або круглих світлодіодів додайте другий вертикальний світлодіод.

`on_color`, **`off_color`**, **`flashIndicator`**
: Коли індикатор спрацює, світлодіод блимає.

`Швидкість спалаху`
: Частота миготіння

indicator_size
Розмір трикутного світлодіода

circle_diameter
Діаметр круглого світлодіода

shape_option
Тип форми світлодіода 0-4

right_edge_offset
Відстань від правого краю

top_edge_offset
Простір від верхнього краю

height_fraction
Використовується для світлодіодних панелей

width_fraction
Використовується для світлодіодних панелей

corner_radius
Радіус кута індикатора.

Колір світлодіодного індикатора можна визначити в *stylesheet* за допомогою наступного коду, доданого до файлу `.qss`:

```
Indicated_PushButton{
  qproperty-on_color: #000;
  qproperty-off_color: #444;
}
```

Або для певної кнопки:

```
Indicated_PushButton #button_estop{
  qproperty-on_color: black;
  qproperty-off_color: yellow;
}
```

Опції `IndicatedPushButton` мають **ексклюзивні опції**:

indicator_HAL_pin_option
Додає `halpin` з назвою `<buttonname>-led`, який керує станом індикатора кнопки.

indicator_status_option
Змушує світлодіод відображати стан цих вибраних статусів LinuxCNC:

- *Зупинено*
- *Увімкнено*
- *Усі Homed*
- *Є спільним житлом*
- *Бездіяльність*
- *Призупинено*
- *Повінь*
- *Туман*
- *Видалення блоку*
- *Додаткова зупинка*

- *Керівництво*
- *MDI*
- *Авто*
- *Шпиндель зупинився*
- *Шпиндель вперед*
- *Зворотний рух шпинделя*
- *На межі*

Деякі `indicator_status_options` містять властивість, яку можна використовувати з *stylesheet* для зміни кольору кнопки залежно від стану властивості в LinuxCNC.
Наразі ці властивості стану можна використовувати для автоматичного стилювання кнопок:

- `is_estopped_status` перемикатиме властивість `isEstop`
- `is_on_status` перемикатиме властивість `isStateOn`
- `is_manual_status`, `is_mdi_status`, `is_auto_status` перемикатимуть властивості `isManual`, `isMDI`, `isAuto`.
- `is_homed_status` вмикає/вимикає властивість *isAllHomed*

Ось приклад запису таблиці стилів, який встановлює фон віджетів кнопок режиму, коли LinuxCNC перебуває в цьому режимі:

```
ActionButton[isManual=true] {
    background: red;
}
ActionButton[isMdi=true] {
    background: blue;
}
ActionButton[isAuto=true] {
    background: green;
}
```

Ось як вказати певний віджет за його objectName у Qt Designer:

```
ActionButton #estop button [isEstopped=false] {
    color: yellow;
}
```

Часто необхідно вмикати та виключати кнопку залежно від стану контролера руху LinuxCNC.

Є кілька властивостей, які можна вибрати для допомоги в цьому:

`isAllHomedSentive`, **`isOnSensitive`**, **`isIdleSensitive`**, **`isRunSensitive`**, **`isRunPausedSensitive`**, **`is`**

Ви можете вибрати кілька властивостей для об’єднаних вимог.

Вибір опції **`checked_state_text_option`** дозволяє кнопці з можливістю перевірки *змінювати текст на основі її позначеного стану*.

Він використовує такі властивості для визначення тексту для кожного стану:

`true_state_string`, **`false_state_string`**
: `\\n` буде перетворено на новий рядок.

Ви можете встановити/змінити їх у таблицях стилів:

```
ActionButton #action_aux{
  qproperty-true_state_string: ”Air\\nOn”;
  qproperty-false_state_string: ”Air\\nOff”;
}
```

Опція **python_command_option** дозволяє запускати невеликі фрагменти коду Python одним натисканням кнопки, без необхідності редагування файлу обробника. Однак, вона може викликати функції у файлі обробника.

Під час використання властивостей `command_string`.

true_python_cmd_string
: Команда Python, яка буде викликана, коли значення кнопки перемикається на `True`.

false_python_cmd_string
: Команда Python, яка буде викликана, коли значення кнопки перемикається на `False`.

Спеціальні слова з великої літери нададуть доступ до наступного:

ПРИМЕР
: Надасть доступ до екземплярів віджетів та функцій обробника.
Наприклад, `INSTANCE.my_handler_function_call(True)`

ДІЯ
: Надасть доступ до бібліотеки `ACTION` QtVCP.
Наприклад, `ACTION.TOGGLE_FLOOD()`

PROGRAM_LOADER
: Надасть доступ до бібліотеки `PROGRAM_LOADER` QtVCP.
Наприклад, `PROGRAM_LOADER.load_halshow()`

HAL
: Надасть доступ до модуля Python HAL.
Наприклад, `HAL.set_p('motion.probe-input','1')`

12.7.6 Віджети лише для імпорту

Ці віджети зазвичай є **базовим класом віджетів для інших віджетів QtVCP**.

Вони *недоступні безпосередньо з редактора Qt Designer*, але їх можна **імпортувати та вставити вручну**.

Їх також можна **розділити на підкласи**, щоб створити подібний віджет з новими функціями.

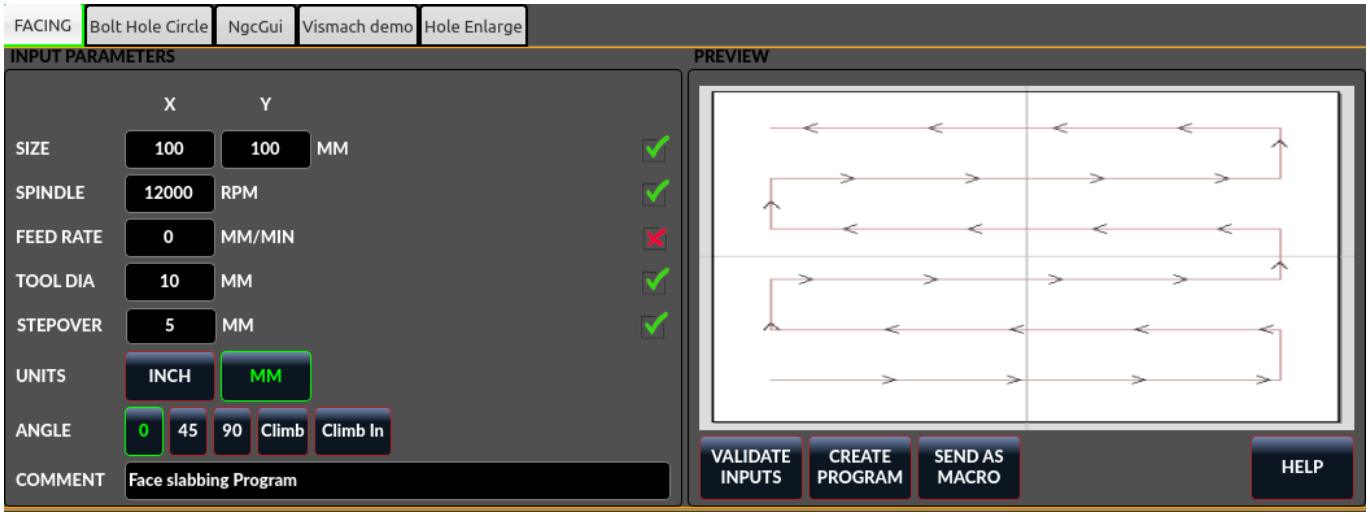
12.7.6.1 Автоматична висота

Віджет для вимірювання двох висот за допомогою зонда.
Для налаштування.

12.7.6.2 Утиліта G-коду

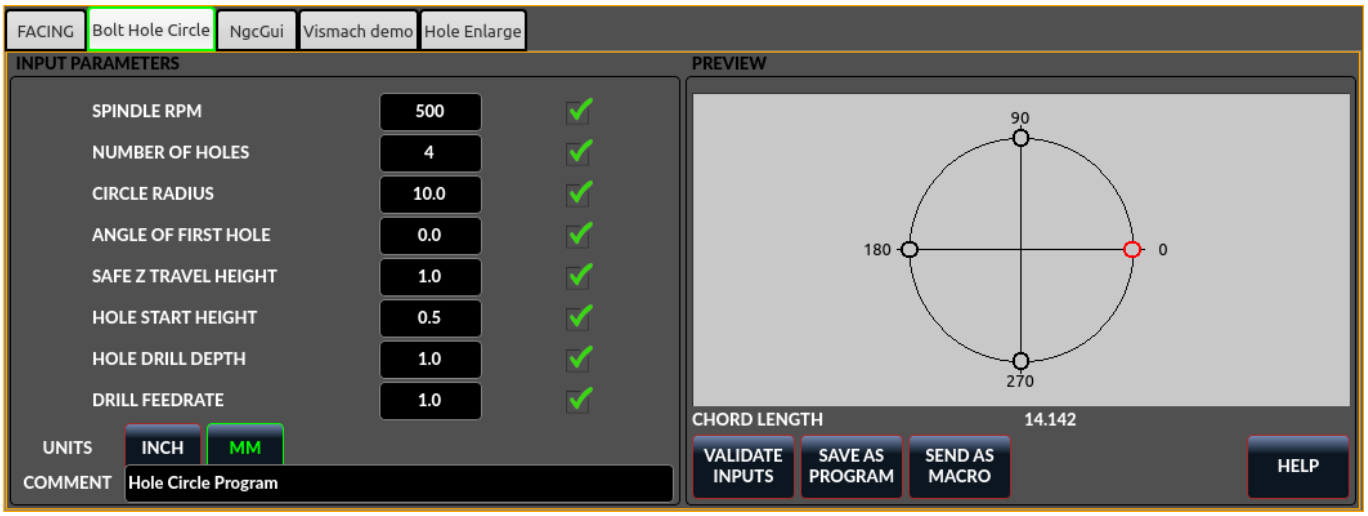
Віджети для виконання поширених процесів обробки.

12.7.6.3 Облицювання



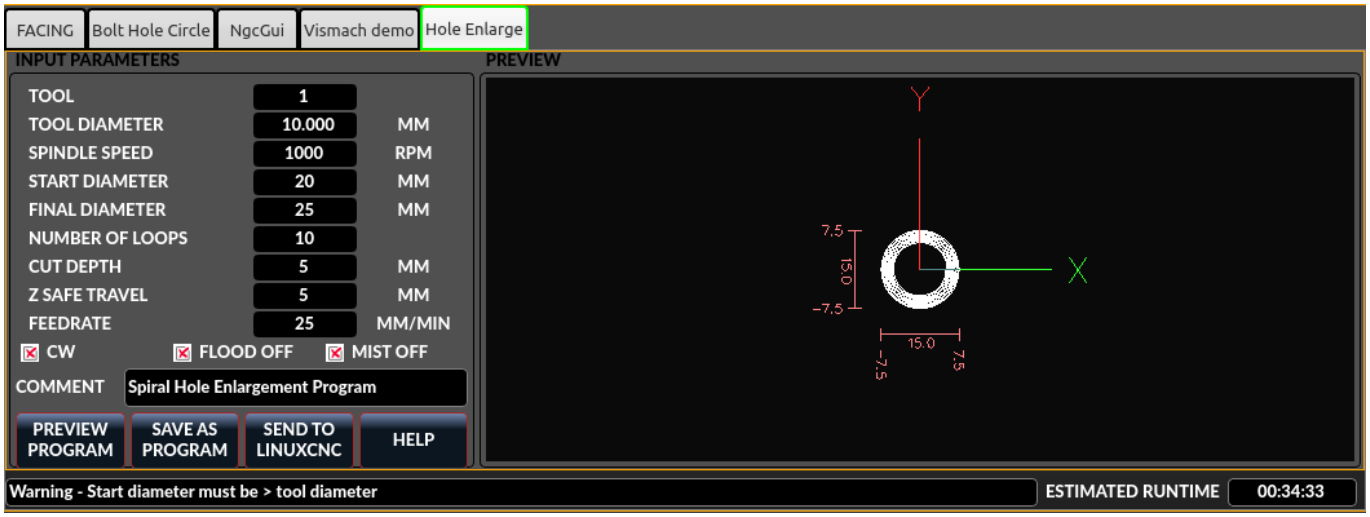
Облицювання плитою або облицюванням визначеної області за допомогою різних стратегій.

12.7.6.4 Коло отворів



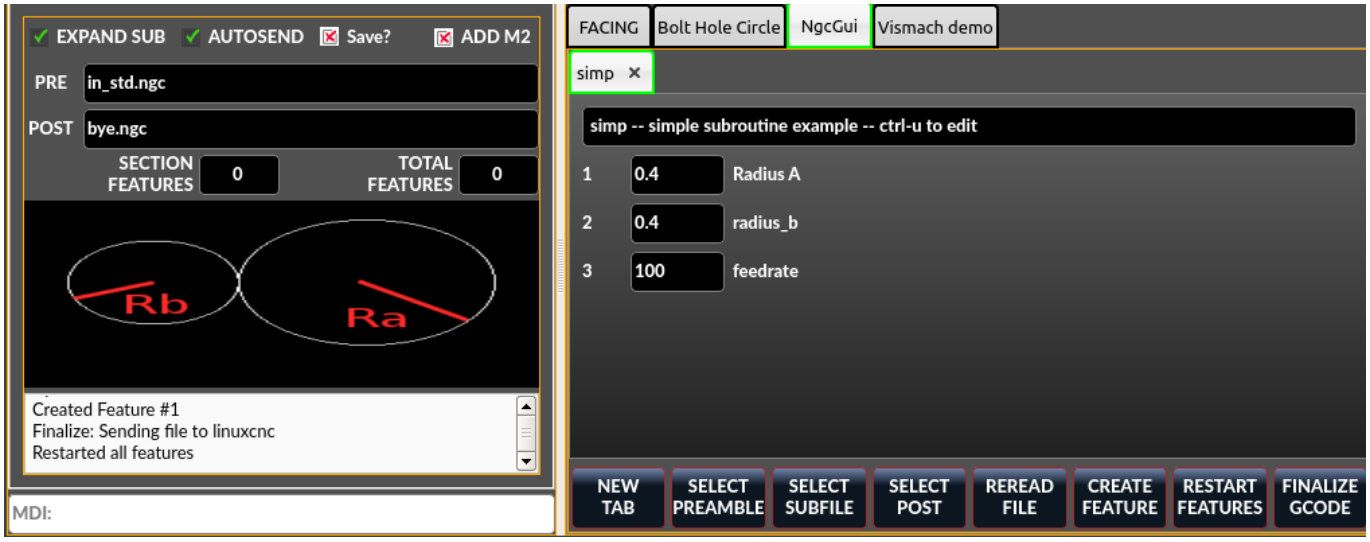
Просвердліть кілька отворів по колу для болтів.

12.7.6.5 Збільшити отвір



Використовуйте торцеву фрезу для розширення просвердленого отвору.

12.7.6.6 Qt NGCGUI



Версія селектора підпрограм NGC у QtVCP (показано, як використовується в QtDragon).

LinuxCNC потрібно знати, де шукати для запуску підпрограм.
Якщо підпрограма викликає інші підпрограми або власні M-коди, ці шляхи також необхідно додати.

```
[RS274NGC]
SUBROUTINE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib:~/linuxcnc/nc_files/examples/ ↩
    ngcgui_lib/utilitysubs
```

QtVCP має знати, звідки відкривати підпрограми.
Ви також можете вказати підпрограми, які потрібно попередньо відкрити у вкладках.

```
[DISPLAY]
# b''Шb''b''лb''b''яb''b''xb'' b''дb''b''ob'' b''пb''b''ib''b''дb''b''пb''b''pb''b''ob''b' ↩
    'гb''b''pb''b''ab''b''мb''b''иb'' NGCGUI.
```

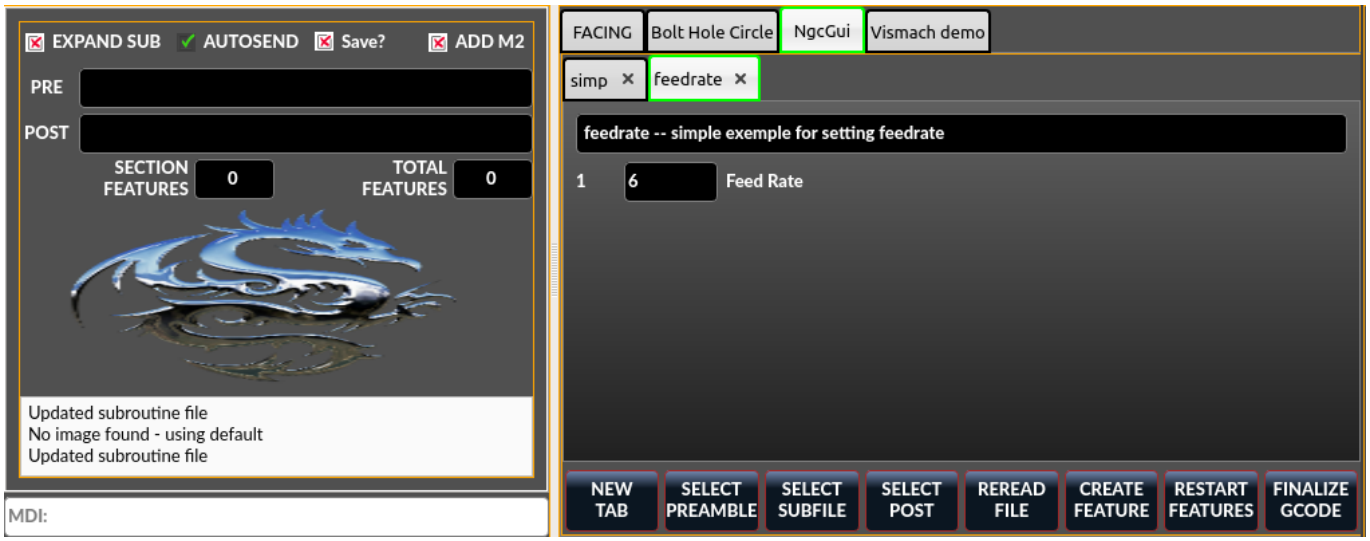
```
# b''Цb''b''eb''b''йb'' b''шb''b''лb''b''яb''b''хb'' b''тb''b''ab''b''кb''b''ob''b''жb'' b' ↩
    'пb''b''ob''b''вb''b''иb''b''нb''b''eb''b''нb'' b''бb''b''уb''b''тb''b''иb'' b''вb''b' ↩
    'кb''b''ab''b''зb''b''ab''b''нb''b''иb''b''йb'' b''вb'' [RS274NGC] SUBROUTINE_PATH
NGCGUI_SUBFILE_PATH = ~/linuxcnc/nc_files/examples/ngcgui_lib
# b''пb''b''ob''b''пb''b''eb''b''рb''b''eb''b''дb''b''нb''b''ьb''b''ob'' b''вb''b''иb''b' ↩
    'бb''b''рb''b''ab''b''нb''b''ib'' b''вb''b''кb''b''лb''b''ab''b''дb''b''кb''b''иb'' b' ↩
    'пb''b''рb''b''ob''b''гb''b''рb''b''ab''b''мb''
# b''вb''b''кb''b''ab''b''жb''b''ib''b''тb''b''ьb'' b''тb''b''ib''b''лb''b''ьb''b''кb''b' ↩
    'иb'' b''ib''b''мb''b''eb''b''нb''b''ab'' b''фb''b''ab''b''йb''b''лb''b''ib''b''вb'', b' ↩
    'фb''b''ab''b''йb''b''лb''b''иb'' b''пb''b''ob''b''вb''b''иb''b''нb''b''нb''b''ib'' b' ↩
    'бb''b''уb''b''тb''b''иb'' b''вb'' NGCGUI_SUBFILE_PATH
NGCGUI_SUBFILE = slot.ngc
NGCGUI_SUBFILE = qpocket.ngc
```

- *NEW TAB* - додати нову порожню вкладку до NGCGUI
- *SELECT PREAMBLE* - виберіть файл, який додає G-код преамбули
- *SELECT SUBFILE* - вибрати файл підпрограми NGCGUI
- *SELECT POST* - виберіть файл, який додає G-код посту
- *REREAD FILE* - перезавантажити файл підпрограми
- *CREATE FEATURE* - додати функцію до списку
- *RESTART FEATURE* - вилучає всі функції зі списку
- *FINALIZE GCODE* - створити повний G-код та надсилати його до файлу LinuxCNC/a

Ви можете створювати власні підпрограми для використання з NGCGUI. Вони повинні відповідати таким правилам:

- Для створення підпрограми для використання з NGCGUI ім’я файлу та ім’я підпрограми мають збігатися.
- Підпрограма має знаходитися в папці в межах шляху пошуку INI-файлу LinuxCNC.
- У першому рядку може бути коментар типу info:
- Підпрограма має бути оточена тегами sub та endsub.
- Використані змінні повинні бути пронумерованими змінними та не повинні пропускати номер.
- Можуть бути включені коментарі та пресети.
- Якщо в папці є файл зображення з такою ж назвою, він буде відображений.

```
(info: feedrate -- simple example for setting feedrate)
o<feedrate> sub
    #<feedrate>       = #1 (= 6 b''Шb''b''вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b' ↩
        'ьb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'') ; b''кb''b''ob''b''мb''b''eb''b' ↩
        'нb''b''тb''b''ab''b''pb''b''ib'' b''вb'' b''дb''b''уb''b''жb''b''кb''b''ab''b''xb'' ↩
         b''бb''b''уb''b''дb''b''уb''b''тb''b''ьb'' b''вb''b''ib''b''дb''b''ob''b''бb''b' ↩
        'pb''b''ab''b''жb''b''ab''b''тb''b''иb''b''cb''b''яb'' b''вb'' ngcui
    f#<feedrate>
o<feedrate> endsub
```



12.7.6.7 Qt PDF

Дозволяє додавати на екран PDF-файли, що можна завантажити.

12.7.6.8 Qt Vismach

Використовуйте це для створення/додавання машин, що імітують OpenGl.

12.7.6.9 Коробка вибору Hal

Цей віджет являє собою вибраний список, який дозволяє вибрати пін-код або сигнал у системі.

```
from qtvcp.widgets.hal_selectionbox import HALSelectionBox

    def buildComboBox(self):
        # b''кb''b''оb''b''мb''b''бb''b''ib''b''нb''b''оb''b''вb''b''аb''b''нb''b''иb''b' ↩
            'йb'' b''сb''b''пb''b''иb''b''сb''b''оb''b''кb'' b''дb''b''лb''b''яb'' b''вb''b' ↩
            'иb''b''бb''b''оb''b''рb''b''уb'' b''кb''b''оb''b''нb''b''тb''b''аb''b''кb''b' ↩
            'тb''b''ib''b''вb'' HAL
        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)
```

Є виклики функцій

```
# b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' b''сb''b''пb''b' ↩
    'иb''b''сb''b''оb''b''кb'' b''тb''b''иb''b''пb''b''ib''b''вb'' b''дb''b''лb''b''яb'' b' ↩
    'вb''b''ib''b''дb''b''оb''b''бb''b''рb''b''аb''b''жb''b''еb''b''нb''b''нb''b''яb'' b' ↩
    'зb'': PINS SIGNALS
combobox.setShowTypes([combobox.PINS])
```

```
# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''тb''b''иb''b' ↩
    'пb''b''иb'' b''вb''b''иb''b''вb''b''ob''b''дb''b''іb''b''вb'' b''дb''b''лb''b''яb'' b' ↩
    'вb''b''іb''b''дb''b''ob''b''бb''b''рb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'':  ↩
    HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# b''тb''b''ab'' b''cb''b''пb''b''иb''b''cb''b''ob''b''кb'' b''нb''b''ab''b''пb''b''рb''b' ↩
    'яb''b''мb''b''кb''b''іb''b''вb'': HAL_IN HAL_OUT
combobox.setPinTypes(types=[combobox.HAL_BIT], direction = [HAL_IN])

# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''тb''b''иb''b' ↩
    'пb''b''иb'' b''cb''b''иb''b''гb''b''нb''b''ab''b''лb''b''іb''b''вb'' b''дb''b''лb''b' ↩
    'яb'' b''вb''b''іb''b''дb''b''ob''b''бb''b''рb''b''ab''b''жb''b''eb''b''нb''b''нb''b' ↩
    'яb'': HAL_BIT,HAL_FLOAT,HAL_S32,HAL_U32
# b''тb''b''ab'' b''cb''b''пb''b''иb''b''cb''b''ob''b''кb'' b''кb''b''eb''b''рb''b''ob''b' ↩
    'вb''b''ab''b''нb''b''иb''b''хb''/b''нb''b''eb''b''кb''b''eb''b''рb''b''ob''b''вb''b' ↩
    'ab''b''нb''b''иb''b''хb'' (b''пb''b''іb''b''дb''b''кb''b''лb''b''юb''b''чb''b''eb''b' ↩
    'нb''b''иb''b''мb'' b''вb''b''иb''b''вb''b''ob''b''дb''b''ob''b''мb'') b''дb''b''лb''b' ↩
    'яb'' b''вb''b''іb''b''дb''b''ob''b''бb''b''рb''b''ab''b''жb''b''eb''b''нb''b''нb''b' ↩
    'яb''
combobox.setSignalTypes( types=[combobox.HAL_BIT], driven = [True,True])
```

12.8 Модулі бібліотек QtVCP

Бібліотеки — це **готові модулі Python, які надають додаткові функції QtVCP**. Таким чином, ви можете вибрати потрібні функції, але вам не доведеться самостійно створювати поширені.

12.8.1 Status

Status — це бібліотека, яка **надсилає повідомлення GObject на основі поточного стану LinuxCNC**. Це *розширення об'єкта GStat класу GladeVCP*.

Він також має деякі функції для звітування про стан таких речей, як внутрішня швидкість штовхання.

Ви *підключаєте виклик функції* до повідомлення STATUS, яке вас цікавить, і QtVCP викличе цю функцію, коли повідомлення буде надіслано зі STATUS.

12.8.1.1 Застосування

- **Імпорт модулів Status**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

b''зb'' b''іb''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' qtvcp.core b''Cb''b''тb''b' ↩
    'ab''b''нb''
```

- **Створення екземпляра модуля Status**
  Додайте цей код Python до розділу створення екземпляра:

```
STATUS = Status()
```

- **Підключення до повідомлень STATUS**
  Use *GObject syntax*.

12.8.1.2 Приклад

Наприклад, ви можете ловити повідомлення про ввімкнення та вимкнення машини.

Note
У наведеному нижче прикладі показано два поширені способи з'єднання сигналів, один з яких використовує лямбду. **`lambda`** використовується для видалення або маніпулювання аргументами зі статусного повідомлення перед викликом функції. Ви можете побачити різницю в сигнатурі викликаної функції: та, що використовує лямбду, не приймає об'єкт статусу — лямбда не передала його функції.

- Помістіть ці команди в розділ `[INITIALIZE]` файлу обробника Python:

```
STATUS.connect('state-on', self.on_state_on)
STATUS.connect('state-off', lambda: w, self.on_state_off())
```

У цьому прикладі коду, коли LinuxCNC перебуває у стані «машина увімкнена», буде викликано функцію `self.on_state_on`.
Коли LinuxCNC перебуває у стані «машина вимкнена», буде викликано функцію `self.on_state_off`.

- Вони викликали б функції, які виглядають так:

```
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` — це бібліотека для **збору та фільтрації даних з INI-файлу**.

12.8.2.1 Доступні дані та значення за замовчуванням

```
LINUXCNC_IS_RUNNING
LINUXCNC_VERSION
INIPATH
INI = linuxcnc.ini(INIPATH)
MDI_HISTORY_PATH = '~/.axis_mdi_history'
QTVCP_LOG_HISTORY_PATH = '~/qtvcp.log'
MACHINE_LOG_HISTORY_PATH = '~/.machine_log_history'
PREFERENCE_PATH = '~/.Preferences'
SUB_PATH = None
SUB_PATH_LIST = []
self.MACRO_PATH = None
MACRO_PATH_LIST = []
INI_MACROS = self.INI.findall("DISPLAY", "MACRO")

IMAGE_PATH = IMAGEDIR
LIB_PATH = os.path.join(HOME, "share","qtvcp")

PROGRAM_FILTERS = None
PARAMETER_FILE = None
MACHINE_IS_LATHE = False
MACHINE_IS_METRIC = False
```

```
MACHINE_UNIT_CONVERSION = 1
MACHINE_UNIT_CONVERSION_9 = [1]*9
TRAJ_COORDINATES =
JOINT_COUNT = int(self.INI.find(”KINS”,”JOINTS”)or 0)
AVAILABLE_AXES = [’X’,’Y’,’Z’]
AVAILABLE_JOINTS = [0,1,2]
GET_NAME_FROM_JOINT = {0:’X’,1:’Y’,2:’Z’}
GET_JOG_FROM_NAME = {’X’:0,’Y’:1,’Z’:2}
NO_HOME_REQUIRED = False
HOME_ALL_FLAG
JOINT_TYPE = self.INI.find(section, ”TYPE”) or ”LINEAR”
JOINT_SEQUENCE_LIST
JOINT_SYNC_LIST

JOG_INCREMENTS = None
ANGULAR_INCREMENTS = None
GRID_INCREMENTS

DEFAULT_LINEAR_JOG_VEL = 15 b’’ob’’b’’дb’’b’’иb’’b’’нb’’b’’иb’’b’’цb’’b’’ьb’’ b’’зb’’b’ ↩
    ’ab’’ b’’xb’’b’’вb’’b’’иb’’b’’лb’’b’’иb’’b’’нb’’b’’yb’’
MIN_LINEAR_JOG_VEL = 60 b’’ob’’b’’дb’’b’’иb’’b’’нb’’b’’иb’’b’’цb’’b’’ьb’’ b’’зb’’b’’ab’’ b’ ↩
    ’xb’’b’’вb’’b’’иb’’b’’лb’’b’’иb’’b’’нb’’b’’yb’’
MAX_LINEAR_JOG_VEL = 300 b’’ob’’b’’дb’’b’’иb’’b’’нb’’b’’иb’’b’’цb’’b’’ьb’’ b’’зb’’b’’ab’’ b ↩
    ’’xb’’b’’вb’’b’’иb’’b’’лb’’b’’иb’’b’’нb’’b’’yb’’

DEFAULT_ANGULAR_JOG_VEL =
MIN_ANGULAR_JOG_VEL =
MAX_ANGULAR_JOG_VEL =

MAX_FEED_OVERRIDE =
MAX_TRAJ_VELOCITY =

AVAILABLE_SPINDLES = int(self.INI.find(”TRAJ”, ”SPINDLES”) or 1)
DEFAULT_SPINDLE_0_SPEED = 200
MAX_SPINDLE_0_SPEED = 2500
MAX_SPINDLE_0_OVERRIDE = 100
MIN_SPINDLE_0_OVERRIDE = 50

MAX_FEED_OVERRIDE = 1.5
MAX_TRAJ_VELOCITY
```

12.8.2.2 Інформація про діалогове вікно повідомлення користувача

```
USRMESS_BOLDTEXT = self.INI.findall(”DISPLAY”, ”MESSAGE_BOLDTEXT”)
USRMESS_TEXT = self.INI.findall(”DISPLAY”, ”MESSAGE_TEXT”)
USRMESS_TYPE = self.INI.findall(”DISPLAY”, ”MESSAGE_TYPE”)
USRMESS_PINNAME = self.INI.findall(”DISPLAY”, ”MESSAGE_PINNAME”)
USRMESS_DETAILS = self.INI.findall(”DISPLAY”, ”MESSAGE_DETAILS”)
USRMESS_ICON = self.INI.findall(”DISPLAY”, ”MESSAGE_ICON”)
ZIPPED_USRMESS =

self.GLADEVCP = (self.INI.find(”DISPLAY”, ”GLADEVCP”)) or None
```

12.8.2.3 Інформація про вбудовану програму

```
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 Помічники

Є деякі *допоміжні функції,* які здебільшого використовуються для підтримки віджетів:

get_error_safe_setting(_self_, _heading_, _detail_, default=_None_), **convert_metric_to_mac**

Отримати розширення фільтрів у форматі Qt.

12.8.2.5 Застосування

- **Імпорт модуля `Info`**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b’’Pb’’b’’Ob’’b’’3b’’b’’Дb’’b’’Ib’’b’’Лb’’ b’’Ib’’b’’Mb’’b’’Пb’’b’’Ob’’b’’Pb’’b’ ↩
    ’Tb’’ **** #
############################

b’’зb’’ qtvcp.core b’’ib’’b’’мb’’b’’пb’’b’’ob’’b’’pb’’b’’тb’’b’’yb’’b’’вb’’b’’ab’’b’’тb’’ ↩
    b’’иb’’ b’’ib’’b’’нb’’b’’фb’’b’’ob’’b’’pb’’b’’мb’’b’’ab’’b’’цb’’b’’ib’’b’’юb’’
```

- **Створення екземпляра модуля `Info`.**
  Додайте цей код Python до розділу створення екземпляра:

```
########################################
# **** b’’Pb’’b’’Ob’’b’’3b’’b’’Дb’’b’’Ib’’b’’Лb’’ «b’’Mb’’b’’Ib’’b’’Cb’’b’’Цb’’b’’Eb’’b’ ↩
    ’Bb’’b’’Ib’’ b’’Бb’’b’’Ib’’b’’Бb’’b’’Лb’’b’’Ib’’b’’Ob’’b’’Tb’’b’’Eb’’b’’Kb’’b’’Иb’’» ↩
    **** #
########################################

INFO = Info()
```

- **Доступ до даних `INFO`** Використовуйте цей загальний синтаксис:

```
home_state = INFO.NO_HOME_REQUIRED
if INFO.MACHINE_IS_METRIC is True:
    print(’Metric based’)
```

12.8.3 `Action`

Бібліотека **`Action`** використовується для **керування контролером руху LinuxCNC**.

Він намагається приховати випадкові деталі та додати зручні методи для розробників.

12.8.3.1 Помічники

Є деякі **допоміжні функції**, які здебільшого використовуються для підтримки цієї бібліотеки:

get_jog_info (_num_) , jnum_check(_num_) , ensure_mode(_modes_) , open_filter_program(_filena

Відкрийте програму фільтрації G-коду.

12.8.3.2 Застосування

- **Імпорт модуля `Action`**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

b''Дb''b''ib''b''яb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' b''зb'' qtvcp. ↩
    core
```

- **Створення екземпляра модуля `Action`**
  Додайте цей код Python до розділу створення екземпляра:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''»  ↩
    **** #
########################################

ACTION = Action()
```

- **Доступ до команд `ACTION`**
  Використовуйте загальний синтаксис, наприклад:

```
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()          # b''Пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''ab''b''єb'' ↩
     b''пb''b''ab''b''yb''b''зb''b''yb''/b''вb''b''іb''b''дb''b''нb''b''ob''b''вb''b''лb'' ↩
    b''eb''b''нb''b''нb''b''яb''
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(b''cb''b''иb''b''cb''b''тb''b''eb''b''мb''b''ab'')

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(b''нb''b''ob''b''мb''b''eb''b''pb'' b''cb''b''yb''b''гb''b''лb'' ↩
    b''ob''b''бb''b''ab'')

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(b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''ab''):

ACTION.HIDE_POINTER(b''cb''b''тb''b''ab''b''нb''):

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 Калал

Бібліотека для взаємодії компонентів/систем HAL.

12.8.4.1 Атрибути

Ось функції, які можна викликати для об’єкта Qhal:

setUpdateRate(cyclerate)
: Встановити частоту циклу в мс

newPin(назва, константа типу виводу, константа напрямку виводу)
: повертає новий об’єкт QPin

getPinObject(name)
: повертає існуючий іменований об’єкт QPin

getValue(name)
: повертає значення іменованого виводу, сигналу або параметра, використовуйте повну назву component.pin.

setPin(name,value)
: встановлює значення іменованого виводу, використовуйте повну назву component.pin.

setSignal(name,value)
: встановлює значення іменованого сигналу, використовуйте повну назву component.pin.

makeUniqueName(name)
: повертає унікальний рядок назви виводу HAL, додаючи *-x* (число) до заданого рядка назви виводу

exit()
: знищує компонент

12.8.4.2 Константи

Ось доступні константи:

- **HAL_BIT**
- **HAL_FLOAT**
- **HAL_S32**
- **HAL_U32**
- **HAL_IN**
- **HAL_OUT**
- **HAL_IO**
- **HAL_RO**
- **HAL_RW**

12.8.4.3 Посилання

Доступні посилання на об’єкти:

- **comp** об’єкт-компонент
- **hal** об’єкт бібліотеки hal

12.8.5 QPin

Клас-обгортка навколо пінів HAL

12.8.5.1 Сигнали

Існує 3 сигнали Qt, до яких можна підключити вивід QPin:

- **value_changed** викличе іменовану функцію з аргументом поточного значення (амортизований)
- **pinValueChanged** викличе іменовану функцію з аргументами об’єкта pin та поточним значенням
- **isDrivenChanged** викличе іменовану функцію з аргументами об’єкта pin та поточного стану, коли штифт (не)з’єднаний з ведучим штифтом

12.8.5.2 Атрибути

Ось функції, які можна викликати для об’єкта QPin:

- **<Pin object>.get()** повертає поточне значення об’єкта pin
- **<Pin object>.set(X)** встановлює значення об’єкта pin на значення X
- **<Pin object>.text()** повертає рядок назви піна

12.8.5.3 Посилання

Доступні посилання на об’єкти:

- **hal** об’єкт бібліотеки hal

12.8.5.4 Приклад

Додати функцію, яка викликається при зміні стану виводу

```
b''зb'' qtvcp.core b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' Qhal
QHAL = Qhal()

########################################
# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
    'кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP
########################################

# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''pb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b' ↩
    'нb''b''ob''.
# b''пb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''pb''b''ab''b''нb''b''ob'', b' ↩
    'ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''
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

Ця бібліотека **обробляє зміни файлу зміщення інструменту**.



Warning
LinuxCNC погано обробляє маніпуляції файлу інструменту сторонніми особами.

12.8.6.1 Помічники

GET_TOOL_INFO(_toolnumber_)
Це поверне **список інформації про номер запитуваного інструменту** у форматі Python.

GET_TOOL_ARRAY()
Це повертає один **список Python зі списками інформації про інструменти**.

Це необроблений список, сформований *з файлу системних інструментів*.

ADD_TOOL(_newtool_ = [_-99, 0,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’,’0’, 0,’New Too

Це поверне **кортеж із двох списків Python зі списків інформації про інструменти Python**:

- **[0]** буде *інформацією про реальні інструменти*
- **[1]** буде *інформація про знос інструментів* (номери інструментів будуть понад 10000; знос інструментів у стилі Fanuc)

За замовчуванням додає порожній запис інструмента з номером інструмента -99.
Ви можете попередньо завантажити масив `newtool` інформацією про інструмент.

DELETE_TOOLS(_toolnumber_)
Видалити пронумерований інструмент.

SAVE_TOOLFILE(_toolarray_)
Це **розбере toolarray та збереже його у файлі інструменту**, зазначеному у *INI-файлі* як шлях інструменту.

Цей масив інструментів має містити всю доступну інформацію про інструменти..

Очікується, що цей масив використовуватиме *необроблений масив інструментів* LinuxCNC, тобто він не містить записів про знос інструменту.

Якщо сталася помилка, повернеться значення True.

CONVERT_TO_WEAR_TYPE(_toolarray_)
Ця функція **перетворює масив необроблених інструментів LinuxCNC на масив інструменті QtVCP**.

Масив інструментів QtVCP містить записи для зносу інструментів по осях X та Z.

LinuxCNC підтримує знос інструменту, додаючи **інформацію про знос інструменту до записів інструментів вище 10000**.

Note
Це також **потрібно написати код перепризначення, щоб додати зміщення зносу під час зміни інструменту**.

CONVERT_TO_STANDARD_TYPE(_toolarray_)
Ця функція **перетворює масив інструментів QtVCP на необроблений масив інструментів LinuxCNC**.

Масив *QtVCP містить записи для зносу інструментів по осях X та Z*.

LinuxCNC підтримує знос інструменту, додаючи **інформацію про знос інструменту до записів інструментів вище 10000**.

Note
Це також **потрібно перепризначити код для додавання зміщень зносу t під час зміни інструменту**.

12.8.7 Path

Path Модуль надає **посилання на шляхи до важливих файлів**.

12.8.7.1 Шляхи, на які посилаються

PATH.PREFS_FILENAME
Шлях до файлу налаштувань.

PATH.WORKINGDIR
Каталог, з якого було запущено QtVCP.

PATH.IS_SCREEN
Це екран чи VCP?

PATH.CONFIGPATH
Запущено папку конфігурації.

PATH.RIPCONFIGDIR
Папка конфігурації запуску на місці для екранів QtVCP.

PATH.BASEDIR
Базова папка для LinuxCNC.

PATH.BASENAME
Ім’я файлу Qt Designer (без закінчення).

PATH.IMAGEDIR
Папка образів QtVCP.

PATH.SCREENDIR
Вбудована папка Screen у QtVCP.

PATH.PANELDIR
Вбудована папка VCP QtVCP.

PATH.HANDLER
Шлях до файлу обробника.

PATH.HANDLERDIR
Директорія, де було знайдено файл обробника Python.

PATH.XML
Шлях до файлу інтерфейсу користувача QtVCP.

PATH.HANDLERDIR
Каталог, де було знайдено файл інтерфейсу користувача.

PATH.QSS
Шлях до файлу QtVCP QSS.

PATH.PYDIR
Бібліотека Python для LinuxCNC.

PATH.LIBDIR
Папка бібліотеки QtVCP.

PATH.WIDGET
Папка віджетів QtVCP.

PATH.PLUGIN
Папка плагіна віджета QtVCP.

PATH.VISMACHDIR
Директорія, де знаходяться попередньо зібрані файли Vismach.

Наразі не використовується:

PATH.LOCALEDIR
Папка перекладу локалізації.

PATH.DOMAIN
Домен перекладу.

12.8.7.2 Помічники

Є деякі допоміжні функції:

```
file_list = PATH.find_vismach_files()
directory_list = PATH.find_screen_dirs()
directory_list = PATH.find_panel_dirs()
```

12.8.7.3 Застосування

- **Імпорт модуля `Path`**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.core import Path
```

- **Створення екземпляра модуля `Path`**
  Додайте цей код Python до розділу створення екземпляра:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''» ↩
    **** #
########################################

PATH = Path()
```

12.8.8 VCPWindow

Модуль **`VCPWindow`** надає **посилання на `MainWindow` та віджети**.

Зазвичай це використовується для бібліотеки (наприклад, бібліотека панелей інструментів використов
її), оскільки віджети отримують посилання на `MainWindow` з функції `_hal_init()`.

12.8.8.1 Застосування

- **Імпорт модуля `VCPWindow`**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.qt_makegui import VCPWindow
```

- **Створення екземпляра модуля `VCPWindow`**
  Додайте цей код Python до розділу створення екземпляра:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''» ↩
    **** #
########################################

WIDGETS = VCPWindow()
```

12.8.9 `Aux_program_loader`

Модуль **`Aux_program_loader`** дозволяє легко **завантажувати допоміжні програми, які часто використовує LinuxCNC**.

12.8.9.1 Помічники

`load_halmeter()`
Halmeter використовується для **відображення даних одного виводу HAL**.
Завантажте `halmeter` за допомогою:

```
AUX_PRGM.load_halmeter()
```

`load_ladder()`
Завантаження програми PLC *ClassicLadder*:

```
AUX_PRGM.load_ladder()
```

`load_status()`
Завантаження програми `status` LinuxCNC:

```
AUX_PRGM.load_status()
```

`load_halshow()`
Завантажте *HALshow*, налаштуйте програму відображення:

```
AUX_PRGM.load_halshow()
```

`load_halscope()`
Завантаження програми *HALscope*:

```
AUX_PRGM.load_halscope()
```

`load_tooledit()`
Завантажте програму *Tooledit*:

```
AUX_PRGM.load_tooledit(<TOOLEFILE_PATH>)
```

load_calibration()
Завантажити програму *Калібрування*:

```
AUX_PRGM.load_calibration()
```

keyboard_onboard()
Завантажити *вбудовану/Matchbox клавіатуру*

```
AUX_PRGM.keyboard_onboard(<ARGS>)
```

12.8.9.2 Застосування

- **Імпорт модуля Aux_program_loader**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.lib.aux_program_loader import Aux_program_loader
```

- **Створення екземпляра модуля Aux_program_loader**
  Додайте цей код Python до розділу створення екземпляра:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''»  ↩
    **** #
########################################

AUX_PRGM = Aux_program_loader()
```

12.8.10 Keylookup

Модуль **Keylookup** використовується для **уможливлення керування поведінкою** за допомогою натискань клавіш, наприклад, біг підтюпцем.

Він використовується всередині файлу обробника для полегшення створення **прив'язок клавіш**, таких як переміщення клавіш по клавіатурі тощо.

12.8.10.1 Застосування

Імпорт модуля Keylookup Щоб імпортувати ці модулі, додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.lib.keybindings import Keylookup
```

Створення екземпляра модуля Keylookup Щоб створити екземпляр модуля Keylookup* для використання, додайте цей код Python до розділу створення екземплярів:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''» ↩
    **** #
########################################

KEYBIND = Keylookup()
```

Note
Для виклику функції Keylookup потрібен код у функції processed_key_event.
Більшість файлів обробників вже містять цей код.

У файлі обробника, у розділі *ініціалізованої функції,* використовуйте цей загальний синтаксис для **створення комбінацій клавіш**:

```
KEYBIND.add_call("DEFINED_KEY","FUNCTION TO CALL", USER DATA)
```

Тут ми додаємо комбінацію клавіш для F10, F11 та F12:

```
########################################
# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
    'кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP
########################################

# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''рb''b''eb''b' ↩
    'нb''b''ob''.
# b''пb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''рb''b''ab''b''нb''b''ob'', b' ↩
    'ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''
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)
```

А потім нам потрібно **додати функції, які викликаються**.
У файлі обробника, в розділі KEY BINDING CALLS, додайте це:

```
####################
# b''Пb''b''Pb''b''Иb''b''Bb''b'''b''b''Яb''b''3b''b''Kb''b''Ab'' b''Kb''b''Лb''b''Ab''b' ↩
    'Bb''b''Ib''b''Шb'' b''Bb''b''Иb''b''Kb''b''Лb''b''Иb''b''Kb''b''Ib''b''Bb'' #
####################

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 Ключові визначення

Ось список розпізнаних ключових слів. Використовуйте цитований текст.
Літерні ключі використовують «Key_» з додаванням верхньої або нижньої літери.
Наприклад, «Key_a» та «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

Модуль **Messages** використовується для **відображення спливаючих діалогових повідомлень на екрані**.

Ці повідомлення:

- *визначено у файлі INI під заголовком [DISPLAY]*, та
- *керується контактами HAL*.

Використовуйте цей стиль, якщо вам потрібні незалежні виводи HAL для кожного діалогового повідомлення.

12.8.11.1 Властивості

BOLDTEXT
Зазвичай це титул.

ТЕКСТ
Текст під заголовком, і зазвичай довший.

ДЕТАЛІ
Текст прихований, якщо на нього не натиснути.

PINNAME
Базова назва виводу(ів) HAL.

ТИП
Визначає, чи це (може мати опції діалогового вікна та стану разом): **статус** – відображається в _рядку стану* та діалоговому вікні сповіщень.
Не потребує втручання користувача.

- **nonedialog** – спеціально не показує діалогове вікно.
  okdialog - *вимагає від користувача натискання кнопки «OK» для закриття діалогового вікна*.
  Повідомлення «OK» мають *два контакти HAL*:
  - Один контакт HAL для запуску діалогового вікна та
  - один, щоб означити, що він очікує відповіді. **yesnodialog** - *вимагає від користувача натискання кнопок «так» або «ні» для закриття діалогового вікна*.
    Повідомлення «Так/Ні» мають *три контакти HAL*:
  - Один для відображення діалогу,
  - Один для очікування, і
  - один за відповідь. ***okcanceldialog*** *- вимагає від користувача вибору ”OK” або ”Скасувати” Повідомлення ”OK/Скасувати” мають _три контакти HAL*:
  - Один для відображення діалогу,
  - Один для очікування, і
  - один за відповідь.
- ***closepromptdialog*** *- вимагає від користувача вибору*

За замовчуванням повідомлення STATUS для `focus_overlay` та звук сповіщень надсилатимуться, коли з’явиться діалогове вікно.
Це дозволяє додавати затемнення/розмиття екрана для `focus` та звуки до сповіщень.

12.8.11.2 Піни HAL

Назви контактів HAL використовуватимуть такі шаблони:

<SCREEN BASENAME>.<PINNAME>
: виклик PIN-коду s32

<SCREEN BASENAME>.<PINNAME>-waiting
: Вихідний біт «Очікування відповіді користувача»

<SCREEN BASENAME>.<PINNAME>-response
: Вихідний біт «Відповідь користувача»

<SCREEN BASENAME>.<PINNAME>-response-s32
: Вивід «Відповідь користувача», контакт s32

12.8.11.3 Приклади

Ось приклади блоків коду визначення INI-повідомлення, які можна знайти під заголовком `[DISPLAY]`:

- Спливаюче повідомлення в рядку стану та на робочому столі:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a statusbar test
MESSAGE_DETAILS = STATUS DETAILS
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest
```

- Спливаюче діалогове вікно з питанням «Так/Ні»:

```
MESSAGE_BOLDTEXT = NONE
MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_DETAILS = Y/N DETAILS
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yndialogtest
```

- Спливаюче діалогове вікно із запитом на відповідь «OK» + рядок стану та сповіщення на робочому столі:

```
MESSAGE_BOLDTEXT = b''Цb''b''eb'' b''кb''b''ob''b''pb''b''ob''b''тb''b''кb''b''иb''b' ↩
    'йb'' b''тb''b''eb''b''кb''b''cb''b''тb''.
MESSAGE_TEXT = b''Цb''b''eb'' b''дb''b''ob''b''вb''b''гb''b''иb''b''йb'' b''тb''b''eb''b' ↩
    'кb''b''cb''b''тb'' b''ob''b''бb''b''ob''b''хb'' b''тb''b''иb''b''пb''b''іb''b''вb'' b ↩
    ''тb''b''eb''b''cb''b''тb''b''yb''. b''Вb''b''іb''b''нb'' b''мb''b''ob''b''жb''b''eb'' ↩
     b''бb''b''yb''b''тb''b''иb'' b''дb''b''ob''b''вb''b''шb''b''иb''b''мb'' b''зb''b' ↩
    'ab'' b''тb''b''eb''b''кb''b''cb''b''тb'' b''yb'' b''pb''b''яb''b''дb''b''кb''b''yb''  ↩
    b''cb''b''тb''b''ab''b''нb''b''yb''.
MESSAGE_DETAILS = b''Дb''b''Eb''b''Tb''b''Ab''b''Лb''b''Ьb''b''Нb''b''Іb''b''Шb''b''Eb''  ↩
    b''Пb''b''Pb''b''Ob'' b''Ob''b''Бb''b''Иb''b''Дb''b''Вb''b''Ab'' b''Tb''b''Иb''b''Пb'' ↩
    b''Иb''
MESSAGE_TYPE = b''cb''b''тb''b''ab''b''нb'' okdialog
MESSAGE_PINNAME = bothtest
```

Віджет `ScreenOptions` може автоматично налаштувати систему повідомлень.

12.8.12 багатоповідомлення

Модуль **Messages** використовується для **відображення спливаючих діалогових повідомлень на екрані**.

Ці повідомлення:

- *визначено у файлі INI під заголовком [DISPLAY]*, та
- *керується одним виводом HAL s32 на кожен визначений ідентифікатор.*
- *Кожне повідомлення викликається відповідним номером на виводі s32.*

Використовуйте цей стиль повідомлень користувача, наприклад, коли VFD надсилає повідомлення про помилки, закодовані у вигляді чисел.
Він використовує загальні виклики/відповіді/очікування HAL-контактів для всіх (за іменем ID) діалогових вікон з декількома повідомленнями. Імена HAL-контактів використовують такі шаблони:

<SCREEN BASENAME>.<ID NAME>
виклик PIN-коду s32

<SCREEN BASENAME>.<ID NAME>-waiting
Вихідний біт «Очікування відповіді користувача»

<SCREEN BASENAME>.<ID NAME>-response
Вихідний біт «Відповідь користувача»

<SCREEN BASENAME>.<ID NAME>-response-s32
Вивід «Відповідь користувача», контакт s32

12.8.12.1 Властивості

TITLE
Це заголовок, який відображається у діалоговому вікні.

ТЕКСТ
Текст під заголовком, і зазвичай довший.

ДЕТАЛІ
Текст прихований, якщо на нього не натиснути.

ТИП
Визначає тип повідомлення, яке бачить користувач (може мати опції діалогового вікна та стану разом): **статус** – відображається в _рядку стану* та діалоговому вікні сповіщень.
Не потребує втручання користувача.

- **nonedialog** – спеціально не показує діалогове вікно.
  okdialog - *вимагаючи від користувача натискання кнопки «OK» для закриття діалогового вікна*.
  Повідомлення «OK» використовують *два контакти HAL*:
  - Один контакт HAL для запуску діалогового вікна та
  - один, щоб означати, що він очікує відповіді. **yesnodialog** - *вимагає від користувача натискання кнопок «так» або «ні» для закриття діалогового вікна*.
    Повідомлення «так/ні» використовують *три контакти HAL*:
  - Один для відображення діалогу,
  - Один для очікування, і
  - один за відповідь.

За замовчуванням повідомлення STATUS для focus_overlay та звук сповіщень надсилатимуться, коли з’явиться діалогове вікно.
Це дозволяє додавати затемнення/розмиття екрана для focus та звуки до сповіщень.

12.8.12.2 Приклади

Ось приклади блоків коду визначення INI-повідомлення, які можна знайти під заголовком `[DISPLAY]`:

```
[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` Модуль використовується для **надсилання повідомлень, інтегрованих у робочий стіл**.

Він використовує бібліотеку `pynotify`.

Ubuntu/Mint не дотримується стандарту, тому ви не можете встановити, як довго повідомлення буде відображатися.
Я пропоную виправити це за допомогою пакета `notify-osd`, доступного за адресою цей PPA (ПРИПИНЕНО через перехід Ubuntu на Gnome).

Notify *зберігає список усіх повідомлень про тривогу з моменту запуску* у **`self.alarmpage`**.
Якщо ви натиснете *’Показати всі повідомлення’* у спливаючому вікні сповіщення, воно виведе їх на термінал.

Віджет `ScreenOptions` може автоматично налаштувати систему сповіщень.

Зазвичай для надсилання сповіщень використовуються *повідомлення* `STATUS`.

12.8.13.1 Властивості

Ви можете встановити:

`title`
: Текст заголовка сповіщення.

`message`
: Текст вмісту сповіщення.

`icon`
: Піктограма сповіщення.

`timeout`
: Як довго повідомлення залишається активним.

12.8.14 Preferences

Preferences Модуль дозволяє **завантажувати та постійно зберігати дані про налаштування на носії інформації**.

Віджет ScreenOptions може автоматично налаштувати систему налаштувань.

QtVCP спочатку шукає віджет ScreenOptions і, якщо його знаходить, викликає **_pref_init()**.
Це *створює об'єкт налаштувань* і повертає його до QtVCP, щоб передати всім віджетам і додати до атрибутів об'єкта вікна.
У цьому випадку об'єкт налаштувань буде доступний з методу initialized_ файлу обробника як **self.w.PREFS_**. Також усі віджети можуть мати доступ до певного файлу налаштувань під час ініціалізації. Віджет ScreenOptions може автоматично налаштувати файл налаштувань.

12.8.15 Player

Цей модуль **дозволяє відтворювати звуки за допомогою Gstreamer, beep та Espeak**.

Це може:

- **play sound/music files** using *Gstreamer* (неблокуючий),
- **відтворювати звуки** за допомогою бібліотеки beep (наразі блокується під час звукового сигналу),
- **вимовляти слова**, використовуючи бібліотеку espeak (без блокування під час мовлення).

Існують *звуки сповіщень за замовчуванням*, що використовують звуки за замовчуванням Mint або FreeDesktop.

Ви можете відтворювати довільні звуки або навіть пісні, вказавши шлях.

STATUS містить _повідомлення для керування модулем Player.

Віджет ScreenOptions може автоматично налаштувати аудіосистему.

12.8.15.1 Звуки

Сповіщення На вибір є **сповіщення** за замовчуванням:

- ERROR
- READY
- ATTENTION
- RING
- DONE
- LOGIN
- LOGOUT

Звукові сигнали Лунає три **звукові сигнали**:

- BEEP_RING
- BEEP_START
- BEEP

12.8.15.2 Застосування

- **Імпорт модуля `Player`**
  Додайте цей код Python до розділу імпорту:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.lib.audio_player import Player
```

- **Створення екземпляра модуля `Player`**
  Додайте цей код Python до вашого екземпляра розділу:

```
########################################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' «b''Mb''b''Ib''b''Cb''b''Цb''b''Eb''b' ↩
    'Bb''b''Ib'' b''Бb''b''Ib''b''Бb''b''Лb''b''Ib''b''Ob''b''Tb''b''Eb''b''Kb''b''Иb''» ↩
    **** #
########################################

SOUND = Player()
SOUND._register_messages()
```

  Функція **`_register_messages()`** підключає аудіоплеєр до бібліотеки `STATUS`, щоб звуки можна було відтворювати за допомогою системи повідомлень `STATUS`.

12.8.15.3 Приклад

Щоб відтворювати звуки після повідомлень `STATUS`, використовуйте такі загальні синтаксиси:

```
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 Віртуальна клавіатура

Ця бібліотека дозволяє вам **використовувати повідомлення `STATUS` для запуску віртуальної клавіатури**.

Він використовує бібліотеки `Onboard` або `Matchbox` для клавіатури.

12.8.17 Дії панелі інструментів

Ця бібліотека надає **готові підменю та дії для меню та кнопок панелі інструментів**.

Кнопки інструментів, меню та меню панелі інструментів:

- *побудовано в Qt Designer* та
- *призначені дії/підменю у файлі обробника*.

12.8.17.1 Дії

estop , power , load , reload , gcode_properties , run , pause , abort , block_delete , optional_stop

Вмикає/вимикає відображення розмірів.

12.8.17.2 Підменю

recent_submenu , home_submenu , unhome_submenu , zero_systems_submenu , grid_size_submenu

Меню для налаштування розміру графічної сітки

12.8.17.3 Застосування

Ось типовий код, який потрібно додати до відповідних розділів файлу *handler*:

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from qtvcp.lib.toolbar_actions import ToolBarActions

########################################
# **** b''pb''b''ob''b''зb''b''дb''b''ib''b''лb'' b''cb''b''тb''b''вb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
    'pb''b''ib''b''вb'' b''бb''b''ib''b''бb''b''лb''b''ib''b''ob''b''тb''b''eb''b''кb'' **** ↩
     #
########################################

TOOLBAR = ToolBarActions()
```

12.8.17.4 Приклади

- Призначення дій інструментів кнопкам панелі інструментів

```
########################################
# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b ↩
    ''кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP
########################################

# b''Hb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# * b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''pb''b''иb'' b''вb''b''ib'' ↩
    b''дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b ↩
    ''нb''b''ob'',
# * b''пb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''pb''b''ab''b''нb''b''ob'', b ↩
    ''ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''.
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)
```

- Додати функцію власної панелі інструментів:

```
####################
# b''3b''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Hb''b''Ib'' b''Фb''b''Уb''b''Hb''b''Kb''b' ↩
    'Цb''b''Ib''b''Їb'' #
####################

def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

12.8.18 Бібліотека машинної графіки Qt Vismach

`Qt_vismach` — це *набір функцій Python*, які можна **використовувати для створення та анімації моделей машин**.

Vismach:

- *відображає модель* у **3D-вікні перегляду**
- *анімує частини моделі*, коли значення пов'язаних з ними контактів HAL змінюються.

Це *версія бібліотеки на основі Qt*, також доступна версія для tkinter у LinuxCNC.

Версія Qt *дозволяє вбудовувати симуляцію в інші екрани*.

12.8.18.1 Вбудовані зразки

У QtVCP включено *зразки панелей* для:

- 3-осьовий фрезерний верстат XYZ,
- 5-осьовий портальний фрезерний верстат,
- 3-осьовий фрезерний верстат з віссю/шпинделем A, та
- масштабувати млин.

Більшість цих зразків, якщо їх завантажити після запущеної конфігурації LinuxCNC (включаючи екрани, що не базуються на QtVCP), реагуватимуть на рух верстата.
Деякі вимагають підключення контактів HAL для руху.

З терміналу (виберіть один):

```
qtvcp vismach_mill_xyz
qtvcp vismach_scara
qtvcp vismach_millturn
qtvcp vismach_5axis_gantry
```

12.8.18.2 Бібліотека примітивів

Надає **основні структурні елементи змодельованої машини**.

`Колекція`
: `Колекція` — це **об'єкт окремих деталей машини**.

  Тут міститься **ієрархічний список** примітивних фігур або *STL-об'єктів*, до яких можна застосовувати операції.

Перекласти
Цей об’єкт виконає обчислення **перетворення OpenGL** *для об’єкта Collection*.

Переклад стосується *переміщення об’єкта по прямій* в інше положення на екрані.

Масштаб
Цей об’єкт виконуватиме функцію масштабування **OpenGL** *для об’єкта колекції*.

HalTranslate
Цей об’єкт виконає обчислення **перетворення OpenGL** *для об’єкта Collection*, **зі зміщенням на значення виводу HAL**.

Переклад означає переміщення об’єкта по прямій лінії в інше положення на екрані.

Ви можете:

- *зчитати пін-код з компонента, що належить об’єкту Vismach*, або
- *безпосередньо зчитувати системний пін HAL*, якщо аргумент компонента встановлено на `None`.

Повернути
Цей об’єкт виконає обчислення обертання **OpenGL** *для об’єкта Collection*.

HalRotate
Цей об’єкт виконає розрахунок **обертання OpenGL** *для об’єкта Collection*, **зі зміщенням на значення виводу HAL**.

Ви можете:

- *зчитати пін-код з компонента, що належить об’єкту vismach*, або
- *безпосередньо зчитувати системний пін HAL*, якщо аргумент компонента встановлено на `None`.

HalToolCylinder
Цей об’єкт створить об’єкт *CylinderZ*, який **змінюватиме розмір та довжину на основі завантаженого виміру інструменту** (з таблиці інструментів)

Він зчитує *HAL-піни* `halui.tool.diameter` та `motion.tooloffset.z`.

Приклад із зразка mill_xyz:

```
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)
])
```

Трек
Переміщувати та обертати об’єкт, щоб він вказував з однієї системи координат `capture()` в іншу.

Базовий об’єкт для *зберігання координат примітивних фігур*.

CylinderX, CylinderY, CylinderZ
Побудуйте циліндр на осі X, Y або Z, вказавши координати *кінцевої точки* (X, Y або Z) та *радіуса*.

Сфера
Побудуйте сферу з координат *центру* та *радіуса*.

TriangleXY, TriangleXZ, TriangleYZ
Побудуйте трикутник у *задана площині*, вказавши *координати Z кутів* для кожної сторони.

ArcX
Побудуйте дугу, вказавши

Коробка
Побудуйте блок, заданий *6 координатами вершин*.

BoxCentered
Побудуйте прямокутник з центром у початку координат, вказавши *ширину* по X та Y_, а також *висоту* по Z_.

BoxCenteredXY
Побудуйте прямокутник з центром у X та Y, що починається від Z=0, вказавши *ширину в X та Y*, та простягаючись вгору або вниз до заданої *висоти в Z*.

Захоплення
Захопити поточну матрицю перетворення колекції.

Note
Це *трансформується з поточної системи координат до системи області перегляду*, а НЕ до світової системи.

Хаб
Heads Up Display малює *напівпрозоре текстове поле*.

Використання:

- `HUD.strs` для речей, які потрібно *постійно* оновлювати,
- `HUD.show("stuff")` для одноразових речей, таких як повідомлення про помилки.

Колір
Застосовує колір до *частин* колекції.

AsciiSTL, AsciiOBJ
Завантажує файл даних STL або OBJ як *частину Vismach*.

12.8.18.3 Застосування

Імпорт симуляції Ось як можна імпортувати симуляцію XYZ_mill у файл панелі або обробника екрана QtVCP.

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

import mill_xyz as MILL
```

Створення екземпляра та використання віджета симуляції Створіть екземпляр віджета симуляції та додайте його до основного макета екрана:

```
########################################
# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
    'кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP
########################################

# b''Hb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# * b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''рb''b''eb''b' ↩
    'нb''b''ob'',
# * b''пb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''рb''b''ab''b''нb''b''ob'', b' ↩
    'ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''.
def initialized__(self):
    machine = MILL.Window()
    self.w.mainLayout.addWidget(machine)
```

12.8.18.4 Більше інформації

Більше інформації про те, як створити власну симуляцію машини, див. у розділі Qt Vismach.

12.9 QtVismach

Vismach — це набір **функцій Python**, які можна використовувати для створення та анімації моделей машин*.

Цей розділ присвячений вбудованій версії Qt Vismach, див. також: https://sa-cnc.com/linuxcnc-vismach/ .

12.9.1 Вступ

Vismach відображає модель у **3D-вікні перегляду**, а **деталі моделі анімуються, коли значення пов'язаних з ними контактів HAL змінюються**.

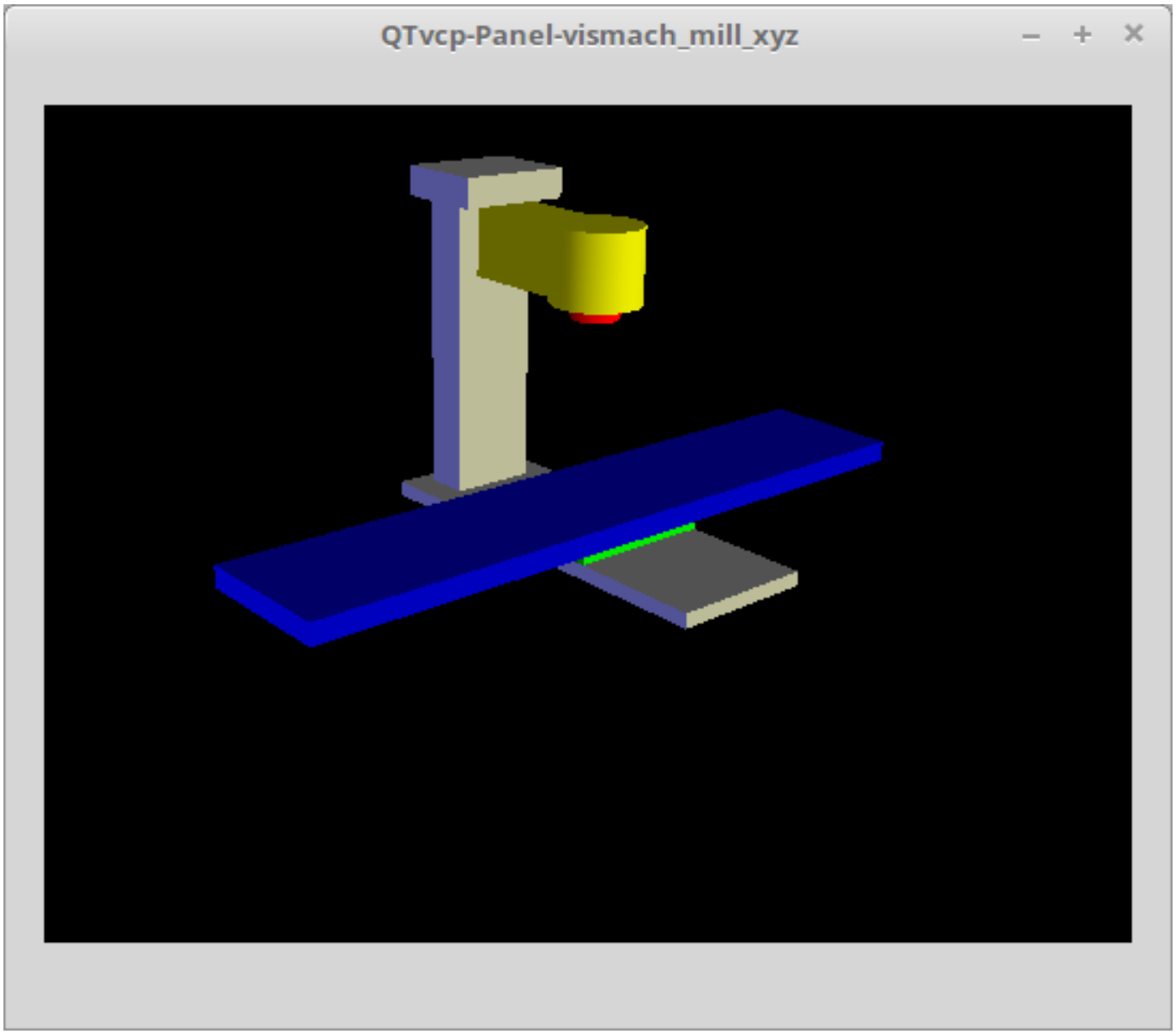


Figure 12.110: 3D-оглядова область QtVismach

Виглядом 3D-вікна Vismach можна маніпулювати наступним чином:

- **масштаб** за допомогою *колеса прокручування*
- **панорамування** перетягуванням середньою кнопкою миші
- **повернути** перетягуванням правою кнопкою миші
- **нахил** шляхом перетягування лівою кнопкою миші

Модель Vismach має форму *скрипта Python* та може використовувати стандартний синтаксис Python.

Це означає, що існує більше одного способу розміщення скрипта, але в наведених у цьому документі прикладах буде використано найпростіший та найбазовіший з них.

Основна послідовність створення моделі Vismach така:

1. Створіть деталі
2. Визначте, як вони рухаються
3. Об’єднайтеся в рухові групи

12.9.2 Ієрархія проектування машин

Модель дотримується **логічного деревоподібного дизайну**.

Уявіть собі дерево з корінням/стовбуром, гілками та меншими відгалуженнями. Якщо ви рухаєте більшу гілку, менші гілки рухатимуться разом з нею, але якщо ви рухаєте меншу гілку, більша не рухатиметься.

Проектування машин відповідає цьому концептуальному проєкту.

Візьмемо, наприклад, млин, показаний на зображенні 3D-огляду вище:

- Якщо перемістити X, він може рухатися самостійно,
- але якщо ви перемістите Y, це також перемістить збірку X, оскільки вона приєднана до збірки Y.

Отже, для цієї машини дерево виглядає так:

```
model
  |
  |---frame
  |     |
  |     |---base
  |     |
  |     |---column
  |     |
  |     |---top
  |
  |---yassembly
  |       |
  |       |---xassembly
  |       |       |
  |       |       |---xbase
  |       |       |
  |       |       |---work
  |       |
  |       |---ybase
  |
  |---zassembly
           |
           |---zframe
           |     |
           |     |---zbody
           |     |
           |     |---spindle
           |
           |---toolassembly
                     |
                     |---cat30
                     |
                     |---tool
                          |
                          |---tooltip
                          |
                          |---(tool cylinder function)
```

Як бачите, *найнижчі частини повинні існувати* першими, *перед тим, як їх можна буде згрупувати з іншими в збірку*. Отже, ви *будуєте вгору* від найнижчої точки дерева та збираєте їх разом.

Те саме стосується будь-якої конструкції машини: подивіться на приклад маніпулятора машини, і ви побачите, що він починається з кінчика, додається до більшої частини маніпулятора, а потім зрештою групується з основою.

12.9.3 Запустіть скрипт

Для тестування корисно включити рядок `#!/usr/bin/env python3`, щоб _дозволити виконання файлу безпосередньо з командного рядка.

Перше, що потрібно зробити, це *імпортувати необхідні бібліотеки*.

```
#!/usr/bin/env python3

import hal
import math
import sys

from qtvcp.lib.qt_vismach.qt_vismach import *
```

12.9.4 Штифти HAL.

Спочатку бібліотека vismach вимагала створення компонента та підключення виводів HAL для керування симуляцією.

`qt_vismach` можна зчитувати системні піни HAL безпосередньо або, якщо бажаєте, використовувати окремі піни HAL, які необхідно визначити в компоненті HAL:

```
c = hal.component("samplegui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

Ви можете вибрати один із двох варіантів у функціях, які приймають ці записи:

hal_comp
: Об'єкт компонента *HAL* або None.
У QtVCP, якщо ви безпосередньо зчитуєте *системні виводи*, тоді аргумент компонента встановлює на `None`.

hal_pin
: Назва виводу BIT HAL IN, який змінюватиме колір.
Якщо hal_comp має значення «None», то це має бути повна назва системного виводу, інакше це назва виводу без назви компонента.

12.9.5 Створення деталей

12.9.5.1 Імпорт файлів STL або OBJ

Найпростіше, мабуть, зробити так:

- *створення геометрії в пакеті CAD*

- *імпортувати в скрипт моделі за допомогою функцій `AsciiSTL()` або `AsciiOBJ()`.*

Обидві функції можуть приймати один із двох іменованих аргументів: *filename* або *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 ...")
```

- Деталі моделі STL додаються до простору Vismach в *тих самих місцях, де вони були створені в просторі STL або OBJ,* тобто в ідеалі з точкою обертання в їх початку координат.

Note
Набагато легше переміщувати модель під час побудови, якщо початок координат знаходиться в точці обертання.

12.9.5.2 Побудувати з геометричних примітивів

Або ж деталі можна *створити всередині скрипта моделі з діапазону примітивів форми*.

`assembly = collction([part1,part2,part3])`
Колекція — це загальний контейнер пов'язаних частин

Багато фігур *створюються у початку координат* і після створення їх потрібно *перемістити в потрібне місце*.

`cylinder = CylinderX(x1, r1, x2, r2)`, **`cylinder = CylinderY(y1, r1, y2, r2)`**, **`cylinder = Cyl`**

Створює *(необов'язково конічний) циліндр на заданій осі* із заданими радіусами у заданих точках на осі.

`sphere = Sphere(x, y, z, r)`
Створює *сферу радіуса r у точці (x,y,z).*

`triangle = TriangleXY(x1, y1, x2, y2, x3, y3, z1, z2)`, **`triangle = TriangleXZ(x1, z1, x2, z`**

Створює *трикутну пластину між площинами,* визначеними двома останніми значеннями, паралельними заданій площині, з вершинами, заданими трьома парами координат.

`arc = ArcX(x1, x2, r1, r2, a1, a2)`
Створіть *дугоподібну* фігуру.

`box = Box(x1, y1, z1, x2, y2, z2)`
Створює *прямокутну призму* з протилежними кутами у вказаних положеннях та ребрами, паралельними осям XYZ.

`box = BoxCentered(xw, yw, zw)`
Створює *бокс xw на yw на zw з центром у початку координат*.

`box = BoxCenteredXY(xw, yw, z)`
Створює *блок-основу на площині WY* шириною xw / yw та висотою z.

Складені деталі можуть бути створені шляхом складання цих примітивів або під час створення, або згодом:

```
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 Переміщення частин моделі

Для складання моделі деталі може знадобитися перемістити в просторі Vismach. Початок координат не рухається — функції Translate() та Rotate() переміщують колекцію під час додавання деталей відносно стаціонарного початку координат.

12.9.6.1 Переклад деталей моделі

part1 = Translate([part1], x, y, z)
Перемістіть деталь1 на задані відстані по осях x, y та z.

12.9.6.2 Обертові деталі моделі

part1 = Rotate([part1], theta, x, y, z)
Повернути деталь на кут тета [градуси] навколо осі між початком координат та координатами x, y, z.

12.9.7 Анімація деталей

Для **анімації моделі, що контролюється значеннями виводів HAL**, є чотири функції: `HalTranslat` `HalRotate`, `HalToolCylinder` та `HalToolTriangle`.

_Щоб деталі рухалися всередині збірки, їм потрібно визначити рухи HAL перед складанням за допомогою команди ”Колекція”.

Вісь обертання та вектор переміщення рухаються разом з деталлю:

- оскільки його переміщує скрипт Vismach під час складання моделі, або
- оскільки він рухається у відповідь на контакти HAL під час анімації моделі.

12.9.7.1 `HalTranslate`

part = HalTranslate([part], hal_comp, hal_pin, xs, ys, zs)

part
Колекція або частина.
Її можна створити заздалегідь у скрипті або, за бажанням, на цьому етапі, наприклад,

```
‘part1 = HalTranslate([Box(....)], ...)‘. +
```

hal_comp
Наступним аргументом є *HAL-компонент*.
У QtVCP, якщо ви безпосередньо зчитуєте *системні виводи*, то аргумент component встановлюється на `None`.

hal_pin
Назва HAL-піна, який анімуватиме рух.
Це має відповідати існуючому HAL-піну, який описує положення суглоба, наприклад:

```
"joint.2.pos-fb"
```

В іншому випадку буде вказано екземпляр компонента та назву виводу цього компонента.
`xs, ys, zs`;; Шкали X, Y, Z.
Для декартової машини, створеної в масштабі 1:1, це зазвичай буде «1,0,0» для руху в позитивному напрямку X.
Однак, якщо файл STL був у см, а машина — в дюймах, це можна виправити, використовуючи 0,3937 (= 1 см/1 дюйм = 1 см /2,54 см) як масштаб.

12.9.7.2 HalRotate

part = HalRotate([part], hal_comp, hal_pin, angle_scale, x, y, z)
Ця команда схожа за своєю роботою на `HalTranslate`, за винятком того, що зазвичай спочатку необхідно перемістити деталь до початку координат, щоб визначити вісь.

x, y, z
Визначає вісь обертання від початку координат точки (x, y, z).
Коли деталь переміщується назад від початку координат до свого правильного розташування, вісь обертання можна вважати «вбудованою» в деталь.

angle_scale
Кути повороту вказані в градусах, тому для поворотного з'єднання з масштабуванням 0-1 вам потрібно буде використовувати шкалу кутів 360.

12.9.7.3 *HalToolCylinder*

tool = HalToolCylinder()
Створіть циліндр для представлення циліндричного фрезерного інструменту на основі таблиці інструментів та поточного завантаженого інструменту.

```
tool = HalToolCylinder()
toolshape = Color([1, .5, .5, .5],[tool])

# b''ab''b''бb''b''ob'' b''бb''b''ib''b''лb''b''ьb''b''шb'' b''кb''b''ob''b''мb''b''пb''b' ↩
    'ab''b''кb''b''тb''b''нb''b''ob'':
toolshape = Color([1, .5, .5, .5], [HalToolCylinder()])
```

12.9.7.4 *HalToolTriangle*

tool = HalToolTriangle()
Створіть трикутник для позначення трикутного токарного інструменту, виходячи з таблиці інструментів та поточного завантаженого інструменту.

```
tool = HalToolTriangle()
toolshape = Color([1, 1, 0, 1],[tool])

# b''ab''b''бb''b''ob'' b''бb''b''ib''b''лb''b''ьb''b''шb'' b''кb''b''ob''b''мb''b''пb''b' ↩
    'ab''b''кb''b''тb''b''нb''b''ob'':
toolshape = Color([1, 1, 0, 1],[HalToolTriangle()])
```

12.9.7.5 Налаштовувані примітиви HAL

Усі примітивні фігури можуть мати імена виводів HAL, що замінюють координати.
Це можна зробити, додавши об’єкт компонента як першу змінну і замінивши рядок імені виводу на координату, або
замінивши повний рядок імені компонента/виводу на координату.

У цьому прикладі створюється *прямокутна призма з протилежними кутами* у вказаних положеннях та ребрами, паралельними осям XYZ.
Початкова координата Z буде контролюватися виводом HAL «Zstart».

```
box = Box(component, x1, y1, 'Zstart', x2, y2, z2)
box = Box(x1, y1, 'componentName.Zstart', x2, y2, z2)
```

12.9.8 Збірка моделі

Щоб деталі рухалися разом, їх потрібно зібрати за допомогою команди **`Collection()`**.

Важливо **зібрати деталі та визначити їхні рухи у правильній послідовності**.

Наприклад, щоб створити фрезерний верстат з рухомою головкою, обертовим шпинделем та анімованою тягою, вам потрібно:

- Створіть основну частину голови.
- Створіть шпиндель у початку координат.
- Дайте визначення обертання.
- Перемістіть головку до шпинделя або шпиндель до головки.
- Створіть планку для витягування.
- Визначте рух тяги.
- Зберіть три частини в головний вузол.
- Визначте рух головки.

У цьому прикладі обертання шпинделя позначається обертанням набору приводних собачок:

```
#b''Cb''b''ob''b''бb''b''ab''b''кb''b''иb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''гb''b''ob''b''нb''b''ib''b''вb''
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)

#b''Дb''b''иb''b''шb''b''лb''b''ob''
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)

# b''гb''b''ob''b''лb''b''ob''b''вb''b''кb''b''ab''/b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'eb''b''лb''b''ьb''
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)

# b''бb''b''ab''b''зb''b''ab''
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# b''пb''b''ob''b''кb''b''лb''b''ab''b''cb''b''тb''b''иb'' b''нb''b''ab'' b''нb''b''ьb''b' ↩
    'ob''b''гb''b''ob'' b''гb''b''ob''b''лb''b''ob''b''вb''b''yb''
base = Collection([head, base])
```

Нарешті, потрібно створити **єдину колекцію всіх деталей машини, підлоги та роботи** (якщо такі є).

Для *серійної машини* кожна нова частина буде додана до колекції попередньої частини.

Для *паралельної машини* може бути кілька "базових" частин.

Таким чином, наприклад, у `scaragui.py` link3 додається до link2, link2 до link1 та link1 до link0, тож остаточна модель створюється за допомогою:

```
model = Collection([link0, floor, table])
```

Тоді як модель VMC з окремими частинами, що рухаються на основі, може мати

```
model = Collection([base, saddle, head, carousel])
```

12.9.9 Інші функції

12.9.9.1 Колір

Встановлює *колір відображення* деталі.

part = Color([_colorspec_], [_part_])
Зверніть увагу, що на відміну від інших функцій, визначення деталі в цьому випадку йде другим.

colorspec
Три значення RGB (0-1.0) та непрозорість. [R, G, B, A]
Наприклад, [1.0,0,0,0.5] для червоного кольору з непрозорістю 50%.

12.9.9.2 HALColorFlip

Встановлює *колір відображення деталі на основі призначеного стану виводу біта HAL*.

part = HALColorFlip([_colorspec_], [_colorspec_], [_part_], hal_comp, hal_pin)
Зверніть увагу, що на відміну від інших функцій, визначення деталі в цьому випадку йде другим.

colorspec
Три значення RGB (0-1.0) та непрозорість.
Наприклад, [1.0,0,0,0.5] для червоного кольору з непрозорістю 50%.

hal_comp
Об’єкт компонента *HAL* або None.
У QtVCP, якщо ви безпосередньо зчитуєте *системні виводи*, тоді аргумент компонента встановлюється на None.

hal_pin
Назва виводу BIT HAL IN, який змінюватиме колір.
Якщо hal_comp має значення «None», то це має бути повна назва системного виводу, інакше це назва виводу без урахування назви компонента.

12.9.9.3 HALColorRGB

Встановлює колір відображення деталі на основі визначеного значення виводу HAL U32.
Колір декодується з значення U32. Кожен колір є десятковим значенням від 0 до 255 (тут показано в шістнадцятковому форматі)
червоний = 0xXXXXXXRR
зелений = 0xXXXXGGXX
синій = 0xXXBBXXXX
у поєднанні як 0xXXBBGGRR

part = HALColorRGB([_part_], hal_comp, hal_pin, alpha=1.0)

hal_comp
Об’єкт компонента *HAL* або None.
У QtVCP, якщо ви безпосередньо зчитуєте *системні виводи*, тоді аргумент компонента встановлюється на None.

hal_pin
Назва виводу U32 HAL IN, який змінюватиме колір.
Якщо hal_comp має значення «None», то це має бути повна назва системного виводу, інакше це назва виводу без урахування назви компонента.

alpha=
Встановлює непрозорість. (0-1.0)

12.9.9.4 Проекційний дисплей

Створює *підказку* у графічному інтерфейсі Vismach для відображення таких елементів, як положення осей, заголовки або повідомлення.

myhud = Hud()

```
myhud = Hud()
myhud.show(”Mill_XYZ”)‘
```

12.9.9.5 Проекційний дисплей HAL

Більш просунута версія Hud, яка дозволяє відображати піни HAL:

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("-------------")
```

Деякі з доступних функцій HalHUD:

- set_background_color(red, green, blue, alpha)
- add_pin(text, format, pinname)
- set_text_color(red, green, blue)

12.9.9.6 ПриховатиКолекцію

```
 HideCollection b''—b'' b''цb''b''еb'' b''кb''b''оb''b''нb''b''тb''b''еb''b''йb''b''нb''b' ↩
    'еb''b''рb'', b''яb''b''кb''b''иb''b''йb'' b''вb''b''иb''b''кb''b''оb''b''рb''b''иb''b' ↩
    'сb''b''тb''b''оb''b''вb''b''уb''b''єb'' b''вb''b''иb''b''вb''b''іb''b''дb'' HAL b' ↩
    'дb''b''лb''b''яb'' b''кb''b''еb''b''рb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb'' b' ↩
    'вb''b''іb''b''дb''b''оb''b''бb''b''рb''b''аb''b''жb''b''еb''b''нb''b''нb''b''яb''b' ↩
    'мb'' b''чb''b''аb''b''сb''b''тb''b''иb''b''нb'', b''щb''b''оb'' b''мb''b''іb''b''сb''b ↩
    ''тb''b''яb''b''тb''b''ьb''b''сb''b''яb'' b''вb'' b''нb''b''ьb''b''оb''b''мb''b''уb''.  ↩
    +
b''Лb''b''оb''b''гb''b''іb''b''кb''b''аb'' b''вb''b''иb''b''сb''b''оb''b''кb''b''оb''b' ↩
    'гb''b''оb'' b''рb''b''іb''b''вb''b''нb''b''яb'' b''нb''b''аb'' b''вb''b''иb''b''вb''b' ↩
    'оb''b''дb''b''іb'' HAL b''пb''b''рb''b''иb''b''хb''b''оb''b''вb''b''уb''b''вb''b''аb''b ↩
    ''тb''b''иb''b''мb''b''еb'' b''чb''b''аb''b''сb''b''тb''b''иb''b''нb''b''иb'', b''щb''b' ↩
    'оb'' b''мb''b''іb''b''сb''b''тb''b''яb''b''тb''b''ьb''b''сb''b''яb'' b''вb'' b''нb''b' ↩
    'ьb''b''оb''b''мb''b''уb''.
```

```
comp.newpin("hide-chuck", hal.HAL_BIT, hal.HAL_IN)
# <b''вb''b''иb''b''гb''b''оb''b''тb''b''оb''b''вb''b''иb''b''тb''b''иb'' b''вb''b''еb''b' ↩
    'рb''b''сb''b''тb''b''аb''b''тb'' b''зb'' b''пb''b''аb''b''тb''b''рb''b''оb''b''нb''b' ↩
    'оb''b''мb'' b''оb''b''сb''b''іb'' b''Ab''
chuckassembly = HideCollection([chuckassembly],comp,'hide-chuck')
# b''тb''b''аb''b''кb''b''оb''b''жb'' b''мb''b''оb''b''жb''b''нb''b''аb'' b''вb''b''иb''b' ↩
    'кb''b''оb''b''рb''b''иb''b''сb''b''тb''b''оb''b''вb''b''уb''b''вb''b''аb''b''тb''b' ↩
    'иb'' b''оb''b''сb''b''ьb'' b''тb''b''аb''b''кb''
chuckassembly = HideCollection([chuckassembly],None,'myvismach.hide-chuck')
```

12.9.9.7 Колір графіка залежно від типу руху

Якщо ви хочете відображати різні кольори для різних рухів, вам потрібно додати ще трохи коду на Python.

Додайте це на початку файлу:

```
b''зb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' qtvcp.core b''Cb''b''тb''b''ab''b ↩
    ''нb''
STATUS = Status()
```

і це для класу Window:

```
STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

# b''Pb''b''ob''b''зb''b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''yb''b''йb''b''тb''b' ↩
    'eb'', b''щb''b''ob''b''бb'' b''зb''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''кb''b' ↩
    'ob''b''лb''b''ib''b''pb'' b''пb''b''ob''b''дb''b''ab''b''чb''b''ib'' b''тb''b''ab'' b' ↩
    'пb''b''ob''b''бb''b''ab''b''чb''b''иb''b''тb''b''иb'' b''вb''b''cb''b''ib'' b''кb''b' ↩
    'ob''b''лb''b''ьb''b''ob''b''pb''b''иb'', b''щb''b''ob'' b''дb''b''pb''b''yb''b''кb''b' ↩
    'yb''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''нb''b''ab'' b''тb''b''eb''b''pb''b''мb''b' ↩
    'ib''b''нb''b''ab''b''лb''
#v.setColorsAttribute('FEED',(0,1,0))
#print(v.colors)
```

Ви можете встановити кольори DEFAULT, FEED, TRAVERSE, ARC, PROBE, ROTARYINDEX, TOOLCHANG
за допомогою setColorsAttribute().

12.9.9.8 Захоплення

- `tooltip = Capture()`
  Уявіть це як невидиму частину, яку потрібно прикріпити до підказки, щоб відстежувати положення та орієнтацію системи координат інструменту. Насправді це матриця перетворення, яка постійно оновлюється під час руху моделі.
- `work = Capture()`
  Те саме, що й вище, але прикріплено до робочого столу для відстеження системи координат заготовки.

12.9.9.9 головний

Це команда, яка все це реалізує, створює відображення тощо, якщо її викликати безпосередньо з Python.
Зазвичай цей файл імпортується QtVCP, а об'єкт `window()` інстанціюється та вбудовується в інший екран.

`main(model, tooltip, work, size=10, hud=myhud, rotation_vectors=None, lat=0, lon=0)`

model
: Має бути колекція, яка містить усі деталі машини.

tooltip і _work_
: Потрібно створити за допомогою `Capture()`, щоб намалювати задній план, який в основному є позицією підказки, намальованою в робочій системі координат. Дивіться `mill_xyz.py` для прикладу того, як підключити підказку до інструменту, а інструмент до моделі.

size
: Встановлює ступінь візуалізації об'єму в початковому вигляді.

hud
: стосується проекційного дисплея.

_rotation_vectors_ або _lat, lon_
: Можна використовувати для встановлення початкової точки огляду. Рекомендується це зробити, оскільки початкова точка огляду за замовчуванням досить некорисна безпосередньо згори.

12.9.10 Поради

Створіть маркер початку координат осей, щоб мати змогу бачити деталі відносно нього для цілей побудови. Ви можете видалити його після завершення.

```
# b''пb''b''оb''b''бb''b''уb''b''дb''b''уb''b''вb''b''аb''b''тb''b''иb'' b''мb''b''аb''b' ↩
    'рb''b''кb''b''еb''b''рb''b''иb'' b''пb''b''оb''b''чb''b''аb''b''тb''b''кb''b''уb'' b' ↩
    'оb''b''сb''b''ib''
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])
```

Додайте його до колекції класу Window, щоб він ніколи не переміщувався з початку.

```
v.model = Collection([origin, model, world])
```

Почніть з кінчика різання та рухайтеся назад. Додайте кожну колекцію до моделі в початку координат та запустіть скрипт для підтвердження розташування, потім поверніть/змістіть та запустіть скрипт для повторного підтвердження.

12.9.11 Базова структура скрипта `QtVismach`

```
# b''ib''b''мb''b''пb''b''оb''b''рb''b''тb''
import hal
from qtvcp.lib.qt_vismach.qt_vismach import *

# b''ib''b''мb''b''пb''b''оb''b''рb''b''тb''b''уb''b''вb''b''аb''b''тb''b''иb'' b''сb''b' ↩
    'тb''b''аb''b''тb''b''уb''b''сb'' b''дb''b''лb''b''яb'' b''пb''b''оb''b''вb''b''ib''b' ↩
    'дb''b''оb''b''мb''b''лb''b''еb''b''нb''b''ьb'' b''пb''b''рb''b''оb'' b''тb''b''иb''b' ↩
    'пb'' b''рb''b''уb''b''хb''b''уb''
from qtvcp.core import Status
STATUS = Status()

# b''сb''b''тb''b''вb''b''оb''b''рb''b''ib''b''тb''b''ьb'' b''тb''b''уb''b''тb'' HAL-b' ↩
    'пb''b''ib''b''нb''b''иb'', b''яb''b''кb''b''щb''b''оb'' b''пb''b''оb''b''тb''b''рb''b' ↩
    'ib''b''бb''b''нb''b''оb''
#c = hal.component("samplegui")
#c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)

# b''сb''b''тb''b''вb''b''оb''b''рb''b''иb''b''тb''b''иb'' b''пb''b''ib''b''дb''b''лb''b' ↩
    'оb''b''гb''b''уb'', b''ib''b''нb''b''сb''b''тb''b''рb''b''уb''b''мb''b''еb''b''нb''b' ↩
    'тb''b''иb'' b''тb''b''аb'' b''пb''b''рb''b''аb''b''цb''b''юb''b''вb''b''аb''b''тb''b' ↩
    'иb''
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()

# b''3b''b''бb''b''еb''b''рb''b''ib''b''тb''b''ьb'' b''тb''b''аb'' b''зb''b''бb''b''еb''b' ↩
    'рb''b''ib''b''тb''b''ьb'' b''мb''b''оb''b''дb''b''еb''b''лb''b''ьb''
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)
```

```
# b''cb''b''tb''b''vb''b''ob''b''pb''b''ib''b''tb''b''ib'' b''mb''b''ob''b''db''b''eb''b' ↩
    'lb''b''ьb'' b''vb''b''eb''b''pb''b''xb''b''nb''b''ьb''b''ob''b''гb''b''ob'' b''pb''b' ↩
    'ib''b''vb''b''nb''b''яb''
model = Collection([base, saddle, head, carousel])

# b''mb''b''ib'' b''xb''b''ob''b''чb''b''eb''b''mb''b''ob'' b''ab''b''бb''b''ob'' b''vb''b' ↩
    'бb''b''yb''b''db''b''yb''b''vb''b''ab''b''tb''b''ib'' b''vb'' qtvcp, b''ab''b''бb''b' ↩
    'ob'' b''vb''b''ib''b''db''b''ob''b''бb''b''pb''b''ab''b''зb''b''ib''b''tb''b''ib'' b' ↩
    'бb''b''eb''b''зb''b''пb''b''ob''b''cb''b''eb''b''pb''b''eb''b''db''b''nb''b''ьb''b' ↩
    'ob'' b''зb''b''ab'' b''db''b''ob''b''пb''b''ob''b''mb''b''ob''b''гb''b''ob''b''юb''  ↩
    PyQt5
# b''tb''b''ob''b''mb''b''yb'' b''cb''b''tb''b''vb''b''ob''b''pb''b''ib''b''mb''b''ob'' b' ↩
    'vb''b''ib''b''kb''b''nb''b''ob'' b''db''b''lb''b''яb'' b''vb''b''ib''b''db''b''ob''b' ↩
    'бb''b''pb''b''ab''b''жb''b''eb''b''nb''b''nb''b''яb'' b''mb''b''ob''b''db''b''eb''b' ↩
    'lb''b''ib''

class Window(QWidget):

    def __init__(self):
        super(Window, self).__init__()
        self.glWidget = GLWidget()
        v = self.glWidget
        v.set_latitudelimits(-180, 180)

        world = Capture()

        # b''pb''b''ob''b''зb''b''kb''b''ob''b''mb''b''eb''b''nb''b''tb''b''yb''b''йb''b' ↩
            'tb''b''eb'', b''яb''b''kb''b''щb''b''ob'' b''єb'' HUD
        # HUD b''mb''b''ab''b''єb'' b''зb''b''nb''b''ab''b''tb''b''ib'', b''db''b''eb'' b' ↩
            'kb''b''pb''b''eb''b''cb''b''lb''b''ib''b''tb''b''ib''
        #v.hud = myhud
        #v.hud.app = v

        # b''ob''b''nb''b''ob''b''vb''b''ib''b''tb''b''ib'' b''kb''b''ob''b''lb''b''ib''b' ↩
            'pb'' b''гb''b''pb''b''ab''b''фb''b''ib''b''kb''b''ab'' b''зb''b''ab''b''lb''b' ↩
            'eb''b''жb''b''nb''b''ob'' b''vb''b''ib''b''db'' b''tb''b''ib''b''пb''b''yb'' b' ↩
            'pb''b''yb''b''xb''b''yb''
        STATUS.connect('motion-type-changed', lambda w, data: v.choosePlotColor(data))

        # b''pb''b''ob''b''зb''b''kb''b''ob''b''mb''b''eb''b''nb''b''tb''b''yb''b''йb''b' ↩
            'tb''b''eb'', b''щb''b''ob''b''бb'' b''зb''b''mb''b''ib''b''nb''b''ib''b''tb''b' ↩
            'ib'' b''kb''b''ob''b''lb''b''ib''b''pb'' b''cb''b''tb''b''pb''b''ib''b''чb''b' ↩
            'kb''b''ib''
        #v.setColorsAttribute('FEED',(0,1,0))
        # b''ib'' b''пb''b''ob''b''бb''b''ab''b''чb''b''ib''b''tb''b''ib'' b''vb''b''cb''b' ↩
            'ib'' b''kb''b''ob''b''lb''b''ьb''b''ob''b''pb''b''ib'', b''nb''b''ab''b''db''b' ↩
            'pb''b''yb''b''kb''b''ob''b''vb''b''ab''b''nb''b''ib'' b''nb''b''ab'' b''tb''b' ↩
            'eb''b''pb''b''mb''b''ib''b''nb''b''ab''b''lb''b''ib''
        #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)
```

```
# b''яв''b''кв''b''щв''b''ов'' b''вв''b''ив''b''кв''b''лв''b''ив''b''кв''b''ав''b''тв''b' ↩
    'ив'' b''цв''b''ев''b''йв'' b''фв''b''ав''b''йв''b''лв'' b''бв''b''ев''b''зв''b''пв''b' ↩
    'ов''b''св''b''ев''b''рв''b''ев''b''дв''b''нв''b''ьв''b''ов'' b''зв'' python3, b''вв''b' ↩
    'ів''b''нв'' b''вв''b''ів''b''дв''b''ов''b''бв''b''рв''b''ав''b''зв''b''ив''b''тв''b' ↩
    'ьв'' b''вв''b''ів''b''кв''b''нв''b''ов'' PyQt5
# b''дв''b''ов''b''бв''b''рв''b''ев'' b''пв''b''ів''b''дв''b''хв''b''ов''b''дв''b''ив''b' ↩
    'тв''b''ьв'' b''дв''b''лв''b''яв'' b''пв''b''ів''b''дв''b''тв''b''вв''b''ев''b''рв''b' ↩
    'дв''b''жв''b''ев''b''нв''b''нв''b''яв'' b''чв''b''ав''b''св''b''тв''b''ив''b''нв'' b' ↩
    'зв''b''бв''b''ів''b''рв''b''кв''b''ив''.

if __name__ == '__main__':
    main(model, tooltip, work, size=600, hud=None, lat=-75, lon=215)
```

12.9.12 Вбудовані зразки панелей Vismach

Вбудовані панелі Vismach у QtVCP

12.10 QtVCP: Створення власних віджетів

12.10.1 Огляд

Створення власних віджетів дозволяє **використовувати редактор Qt Designer для розміщення власного віджета**, *замість того, щоб робити це вручну у файлі обробника*.

Корисні користувацькі віджети були б чудовим способом зробити свій внесок у LinuxCNC.

12.10.1.1 Віджети

Віджет — це *загальна назва для об'єктів інтерфейсу користувача*, таких як кнопки та мітки в PyQt.

Також існують **спеціальні віджети, створені для LinuxCNC**, які спрощують інтеграцію.

Всі ці віджети можна *розмістити за допомогою редактора Qt Designer*, що дозволяє *побачити результат* перед фактичним завантаженням панелі в LinuxCNC.

12.10.1.2 Дизайнер Qt

Qt Designer — це *WYSIWYG (що бачиш, те й отримуєш) редактор для розміщення віджетів PyQt*.

Спочатку його метою було створення графічних віджетів для програм.
Ми використовуємо його для **створення екранів та панелей для LinuxCNC**.

У Qt Designer, у лівій частині редактора, ви знайдете **три категорії віджетів LinuxCNC**:

- *Віджети лише для HAL*.
- *Віджети контролера LinuxCNC*.
- *діалогові віджети*.

Щоб Qt Designer міг *додавати власні віджети* до свого редактора, потрібно додати **плагін** до потрібної папки.

12.10.1.3 Процес ініціалізації

QtVCP виконує *додаткове налаштування* для **віджетів, підкласованих з `_HALWidgetBase`**, тобто віджетів, "підібраних HAL".

Це включає:

- Введення *важливих змінних*,
- Виклик *додаткової функції налаштування*
- Виклик *функції очищення* під час завершення роботи.

Ці функції не викликаються, коли редактор Qt Designer відображає віджети.

Коли QtVCP створює екран з файлу `.ui`:

1. Він шукає всі HAL-подібні віджети.
2. Він знаходить віджет `ScreenOptions` для збору інформації, необхідної для вставки в інші віджети
3. Він створює екземпляр кожного віджета, і якщо це HAL-подібний віджет, викликає функцію `hal_init()`.
   Функція **`hal_init()`** визначена в базовому класі та:
   a. Додає змінні, такі як файл налаштувань, до кожного HAL-подібного віджета.
   b. Викличте `+_hal_init()+` для віджета.
   `+_hal_init()+` дозволяє розробнику віджета виконувати налаштування, що потребують доступу до додаткових змінних.

Ось опис додаткових змінних, введених у віджети, "підібрані до HAL":

`self.HAL_GCOMP`
Екземпляр компонента *HAL*

`self.HAL_NAME`
Назва цього *віджета* у вигляді рядка

`self.QT_OBJECT_`
Цей *екземпляр об'єкта віджета*

`self.QTVCP_INSTANCE_`
Найвищий рівень батьківського елемента екрана

`self.PATHS_`
Екземпляр бібліотеки шляхів QtVCP

`self.PREFS_`
Екземпляр *додаткового файлу налаштувань*

`self.SETTINGS_`
Екземпляр об'єкта `Qsettings`

12.10.1.4 процес очищення

Коли QtVCP закривається, він викликає функцію **`+_hal_cleanup()+`** *на всіх HAL-подібних віджетах*.

Базовий клас створює порожню функцію `+_hal_cleanup()+`, яку можна перевизначити в підкласі користувацького віджета.

Це можна використовувати для таких дій, як налаштування записів тощо.

Ця функція не викликається, коли редактор Qt Designer відображає віджети.

12.10.2 Користувацькі віджети HAL

Найпростішим прикладом є віджети HAL.
Файл `qtvcp/widgets/simple_widgets.py` містить багато віджетів лише HAL.

Давайте розглянемо фрагмент `simple_widgets.py`:

У розділі «Імпорт» Тут ми імпортуємо бібліотеки, необхідні нашому класу віджетів.

```
#!/usr/bin/env python3

##############################
# b''Ib''b''мb''b''пb''b''ob''b''pb''b''тb''
##############################
from PyQt5 import QtWidgets                     # <t>\coref{1}{CO4-1}</t>
from qtvcp.widgets.widget_baseclass \
    import _HalWidgetBase, _HalSensitiveBase    # <t>\coref{2}{CO4-2}</t>
import hal                                      # <t>\coref{3}{CO4-3}</t>
```

У цьому випадку нам потрібен доступ до:

❶ Бібліотека QtWidgets від PyQt,

❷ Бібліотека HAL від LinuxCNC та

❸ Віджет QtVCP `baseclass` 's **`_HalSensitiveBase`** для *автоматичної настройки контактів HAL* та для *відключення/включення віджета* (також відомого як чутливість вводу).
У бібліотеці також доступні функції `_HalToggleBase` та `_HalScaleBase`.

У розділі «ВІДЖЕТ» Ось *користувацький віджет*, заснований на віджеті **`QGridLayout`** від PyQt.

`QGridLayout` дозволяє:

- *Розташуйте об'єкти у вигляді сітки*.
- *Увімкнути/вимкнути всі віджети всередині* на основі **стану виводу HAL**.

```
######################
# WIDGET
######################

class Lcnc_GridLayout(QtWidgets.QWidget, _HalSensitiveBase):    # ❶
    def __init__(self, parent = None):                          # ❷
        super(GridLayout, self).__init__(parent)                # ❸
```

Рядок за рядком:

❶ Це визначає *ім'я класу* та *бібліотеки, від яких він успадковує*.
Цей клас, названий `Lcnc_GridLayout`, успадковує функції `QWidget` та `+_HalSensitiveBase+`.
`+_HalSensitiveBase+` є «підкласом» **`+_HalWidgetBase+`**, *базовим класом більшості віджетів QtVCP*, що означає, що він має всі функції `+_HalWidgetBase+` плюс функції `+_HalSensitiveBase+`.
Він додає функцію, яка дозволяє вмикати або вимикати віджет на основі вхідного BIT-контакту HAL.

❷ Це функція, яка *викликається, коли віджет вперше створюється* (тобто створюється його екземпляр) – це досить стандартно.

❸ Ця функція ініціалізує **`Super` класи** нашого віджета.
`Super` означає просто *успадковані базові класи*, тобто `QWidget` і `_HalSensitiveBase`.
Досить стандартно, за винятком того, що змінюється назва віджета.

12.10.3 Віджети користувацького контролера з використанням STATUS

Віджети, що взаємодіють з контролером LinuxCNC, лише трохи складніші та потребують деяких *додаткових бібліотек*.

У цьому скороченому прикладі ми додамо властивості, які можна змінити в Qt Designer.

Цей віджет світлодіодного індикатора реагуватиме на вибрані стани контролера LinuxCNC.

```
#!/usr/bin/env python3

##############################
# b''Ib''b''мb''b''пb''b''ob''b''pb''b''тb''
##############################
from PyQt5.QtCore import pyqtProperty
from qtvcp.widgets.led_widget import LED
from qtvcp.core import Status

##########################################
# **** b''pb''b''ob''b''зb''b''дb''b''ib''b''лb'' b''cb''b''тb''b''вb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
    'pb''b''ib''b''вb'' b''бb''b''ib''b''бb''b''лb''b''ib''b''ob''b''тb''b''eb''b''кb'' **** ↩
     #
##########################################
STATUS = Status()

##########################################
# b''вb''b''иb''b''зb''b''нb''b''аb''b''чb''b''eb''b''нb''b''нb''b''яb'' b''кb''b''лb''b' ↩
    'аb''b''cb''b''уb'' b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b''аb''b' ↩
    'цb''b''ьb''b''кb''b''ob''b''гb''b''ob'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b' ↩
    'аb''
##########################################
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)

    ########################################################################
    # b''Bb''b''cb''b''тb''b''аb''b''нb''b''ob''b''вb''b''лb''b''юb''b''вb''b''аb''b''чb''b ↩
        ''ib''/b''гb''b''eb''b''тb''b''тb''b''eb''b''pb''b''иb''/b''cb''b''кb''b''иb''b' ↩
        'дb''b''аb''b''нb''b''нb''b''яb'' b''вb''b''лb''b''аb''b''cb''b''тb''b''иb''b''вb''b ↩
        ''ob''b''cb''b''тb''b''eb''b''йb'' Qt Designer
    ########################################################################

    # b''ib''b''нb''b''вb''b''eb''b''pb''b''тb''b''ob''b''вb''b''аb''b''нb''b''иb''b''йb''  ↩
        b''cb''b''тb''b''аb''b''тb''b''уb''b''cb''
```

```
    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

    # b''cb''b''тb''b''ab''b''нb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''иb'' b''зb''b' ↩
        'уb''b''пb''b''иb''b''нb''b''eb''b''нb''b''ob''
    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

    # b''cb''b''тb''b''ab''b''нb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''иb'' b''уb''b' ↩
        'вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob''
    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

    ######################################
    # b''Bb''b''лb''b''ab''b''cb''b''тb''b''иb''b''вb''b''ob''b''cb''b''тb''b''ib'' Qt  ↩
        Designer
    ######################################
    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 У розділі «Імпорт»

Тут ми імпортуємо бібліотеки, необхідні нашому класу віджетів.

```
#!/usr/bin/env python3

################################
# b''Ib''b''мb''b''пb''b''ob''b''pb''b''тb''
################################
from PyQt5.QtCore import pyqtProperty       # <t>\coref{1}{CO6-1}</t>
from qtvcp.widgets.led_widget import LED    # <t>\coref{2}{CO6-2}</t>
from qtvcp.core import Status               # <t>\coref{3}{CO6-3}</t>
```

Ми імпортуємо

1. `pyqtProperty` щоб ми могли взаємодіяти з редактором Qt Designer,
2. `LED` оскільки наш власний віджет базується на ньому,
3. `Status` тому що він надає нам повідомлення про стан від LinuxCNC.

12.10.3.2 У розділі «Створення миттєвих бібліотек»

Тут ми створюємо екземпляр бібліотеки `Status`:

```
##############################################
# **** b''pb''b''ob''b''зb''b''дb''b''ib''b''лb'' b''cb''b''тb''b''вb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
    'pb''b''ib''b''вb'' b''бb''b''ib''b''бb''b''лb''b''ib''b''ob''b''тb''b''eb''b''кb'' **** ↩
     #
##############################################
STATUS = Status()
```

Зазвичай ми створювали екземпляр бібліотеки *поза класом віджета*, щоб посилання на неї було **глобальним**, тобто вам не потрібно було використовувати `self` перед ним.

За домовленістю ми використовуємо *всі великі* літери в назві для глобальних посилань.

12.10.3.3 У розділі «Визначення класу користувацького віджета»

Це суть нашого власного віджета.

Визначення класу та функція ініціалізації екземпляра

```
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 Визначає **ім'я** *нашого користувацького віджета* та від якого іншого *класу* він успадковується. У цьому випадку ми успадковуємо `LED` - віджет QtVCP, який представляє індикатор стану.

❷ Типово для більшості віджетів — викликається, коли віджет вперше створюється.

❸ Типово для більшості віджетів — викликає код ініціалізації батьківського (супер) віджета.
Потім ми встановлюємо деякі атрибути:

❹ Успадковано від `Lcnc_Led` - ми встановлюємо його тут, щоб не створювати контакт HAL.

❺ Успадковано від `Lcnc_led` — ми встановлюємо його, щоб переконатися, що світлодіод вимкнено.

Інші атрибути стосуються опцій, які можна вибрати для нашого віджета.

Функція ініціалізації HAL віджета

```
    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))
```

Ця функція підключає STATUS (бібліотеку повідомлень про стан LinuxCNC) до нашого віджета, щоб світлодіод вмикався або вимикався залежно від вибраного стану контролера.

Ми можемо вибрати два стани: `is_estopped` або `is_on`.
Залежно від того, який із них активний, наш віджет підключається до відповідних повідомлень STATUS.

+_hal_init()+ викликається для кожного віджета, який успадковує +_HalWidgetBase+, коли QtVCP вперше будує екран.
Ви можете запитати, чому вона викликається для цього віджета, оскільки ми не мали +_HalWidgetBase+ у нашому визначенні класу (`class Lcnc_State_Led(Lcnc_Led):`) — вона викликається, оскільки `Lcnc_Led` успадковує +_HalWidgetBase+.

У цій функції ви маєте доступ до деякої додаткової інформації (хоча ми не використовуємо її в цьому прикладі):

self.HAL_GCOMP
: екземпляр компонента *HAL*

self.HAL_NAME
: Назва цього *віджета* у вигляді рядка

self.QT_OBJECT_
: Екземпляр об’єкта PyQt цього *віджета*

self.QTVCP_INSTANCE_
: *Найвищий рівень батьківського* елемента екрана

self.PATHS_
: Екземпляр бібліотеки path у QtVCP

self.PREFS_
: *екземпляр додаткового файлу налаштувань*

self.SETTINGS_
: об’єкт `Qsettings`

Ми могли б використовувати цю інформацію для створення HAL-пінів або пошуку шляхів до зображень тощо.

```
STATUS.connect('state-estop', lambda w:self._flip_state(True))
```

Давайте розглянемо цей рядок уважніше:

- STATUS — дуже поширена тема для створення віджетів.
  STATUS використовує систему повідомлень `GObject` для надсилання повідомлень віджетам, які реєструються в ньому.
  Цей рядок відповідає за процес реєстрації.
- «state-estop» – це повідомлення, яке ми хочемо прослухати та вжити заходів. Існує багато доступних повідомлень.
- `lambda w:self._flip_state(True)` — це те, що відбувається, коли повідомлення перехоплюється. Функція lambda приймає екземпляр віджета (`w`), який надсилає GObject, а потім викликає функцію `self._flip_state(True)`.
  Lambda використовувалася для видалення об’єкта (`w`) перед викликом функції `self._flip_state`. Вона також дозволяла надсилати `self._flip_state()` стан True.

```
    def _flip_state(self, data):
            if self.invert_state:
                data = not data
            self.change_state(data)
```

Це функція, яка фактично змінює стан світлодіода.
Вона викликається, коли прийнято відповідне повідомлення STATUS.

```
STATUS.connect('current-feed-rate', self._set_feedrate_text)
```

Викликана функція виглядає так:

```
    def _set_feedrate_text(self, widget, data):
```

в якому віджет та будь-які дані повинні бути прийняті функцією.

```
    ##########################################################################
    # b''Bb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b''юb''b''вb''b''ab''b''чb''b ↩
        ''ib''/b''гb''b''eb''b''тb''b''тb''b''eb''b''pb''b''иb''/b''cb''b''кb''b''иb''b' ↩
        'дb''b''ab''b''нb''b''нb''b''яb'' b''вb''b''лb''b''ab''b''cb''b''тb''b''иb''b''вb''b ↩
        ''ob''b''cb''b''тb''b''eb''b''йb'' Qt Designer
    ##########################################################################

    # b''ib''b''нb''b''вb''b''eb''b''pb''b''тb''b''ob''b''вb''b''ab''b''нb''b''иb''b''йb''  ↩
        b''cb''b''тb''b''ab''b''тb''b''yb''b''cb''
    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

    # b''cb''b''тb''b''ab''b''нb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''иb'' b''зb''b' ↩
        'yb''b''пb''b''иb''b''нb''b''eb''b''нb''b''ob''
    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

    # b''cb''b''тb''b''ab''b''нb'' b''мb''b''ab''b''шb''b''иb''b''нb''b''иb'' b''yb''b' ↩
        'вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob''
    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
```

Ось так *Qt Designer встановлює атрибути віджета*.
Це *також можна викликати безпосередньо* у віджеті.

```
    ########################################
    # b''Bb''b''лb''b''ab''b''cb''b''тb''b''иb''b''вb''b''ob''b''cb''b''тb''b''ib'' Qt  ↩
        Designer
    ########################################
    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)
```

Це **реєстрація властивостей у Qt Designer**.

Назва власності:

- це *текст, що використовується в Qt Designer,*
- *не може бути таким самим, як атрибути*, які вони представляють.

Ці властивості відображаються в Qt Designer у порядку, в якому вони тут з’являються.

12.10.4 Віджети налаштованого контролера з діями

Ось приклад віджета, який встановлює систему відліку користувача.

Це змінюється:

- стан контролера машини за допомогою бібліотеки ACTION,
- чи можна натиснути кнопку, використовуючи бібліотеку STATUS.

```
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

# b''Ib''b''нв''b''cb''b''тb''b''ab''b''нв''b''цb''b''ib''b''юb''b''вb''b''ab''b''нв''b' ↩
    'нв''b''яв'' b''бb''b''ib''b''бb''b''лв''b''ib''b''ob''b''тb''b''eb''b''кb'' b''ib''b' ↩
    'зb'' b''гb''b''лв''b''ob''b''бb''b''ab''b''лв''b''ьb''b''нв''b''ив''b''мb'' b''пв''b' ↩
    'ob''b''cb''b''ив''b''лв''b''ab''b''нв''b''нв''b''яв''b''мb''
# STATUS b''нв''b''ab''b''дв''b''ab''b''єb'' b''нв''b''ab''b''мb'' b''пв''b''ob''b''вb''b' ↩
    'ib''b''дв''b''ob''b''мb''b''лв''b''eb''b''нв''b''нв''b''яв'' b''пв''b''рb''b''ob'' b' ↩
    'cb''b''тb''b''ab''b''нв'' b''вb''b''ib''b''дв'' LinuxCNC
# INFO b''мb''b''ib''b''cb''b''тb''b''ив''b''тb''b''ьb'' b''дв''b''eb''b''тb''b''ab''b' ↩
    'лв''b''ib'' INI
# ACTION b''нв''b''ab''b''дв''b''ab''b''єb'' b''кb''b''ob''b''мb''b''ab''b''нв''b''дв''b' ↩
    'ив'' 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)

    ##############################
    # b''кb''b''ob''b''дb'' b''нb''b''eb''b''ob''b''бb''b''хb''b''ib''b''дb''b''нb''b''ob'' ↩
        b''гb''b''ob'' b''кb''b''лb''b''ab''b''cb''b''yb'' b''кb''b''ob''b''тb''b''лb''b' ↩
        'ab'' #
    ##############################

    def __getitem__(self, item):
```

```
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

12.10.5 Зміни властивостей таблиці стилів на основі подій

Можна **змінити стиль віджетів у разі зміни подій**. Ви повинні явно *"відполірувати" віджет*, щоб PyQt переробив стиль.
Це відносно дорога функція, тому її слід використовувати економно.

У цьому прикладі встановлюється властивість `isHomed` на основі стану homed у LinuxCNC та, у свою чергу, використовується для зміни властивостей таблиці стилів:

У цьому прикладі властивість isHomed буде встановлено на основі стану homed у LinuxCNC.

```
class HomeLabel(QLabel, _HalWidgetBase):
    def __init__(self, parent=None):
        super(HomeLabel, self).__init__(parent)
        self.joint_number = 0
        # b''дb''b''лb''b''яb'' b''чb''b''иb''b''тb''b''аb''b''нb''b''нb''b''яb'' b''тb''b' ↩
            'аb''b''бb''b''лb''b''иb''b''цb''b''ьb'' b''сb''b''тb''b''иb''b''лb''b''іb''b' ↩
            'вb''
        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))

    # b''оb''b''нb''b''оb''b''вb''b''иb''b''тb''b''иb'' b''вb''b''лb''b''аb''b''сb''b''тb'' ↩
        b''иb''b''вb''b''іb''b''сb''b''тb''b''ьb'' ishomed
    # b''пb''b''оb''b''лb''b''іb''b''рb''b''уb''b''вb''b''аb''b''тb''b''иb'' b''вb''b''іb'' ↩
        b''дb''b''жb''b''еb''b''тb'', b''щb''b''оb''b''бb'' b''тb''b''аb''b''бb''b''лb''b' ↩
        'иb''b''цb''b''яb'' b''сb''b''тb''b''иb''b''лb''b''іb''b''вb'' b''бb''b''аb''b''чb'' ↩
        b''иb''b''лb''b''аb'' b''зb''b''мb''b''іb''b''нb''b''уb'' b''вb''b''лb''b''аb''b' ↩
        'сb''b''тb''b''иb''b''вb''b''оb''b''сb''b''тb''b''іb''
    # b''дb''b''еb''b''яb''b''кb''b''іb'' b''тb''b''аb''b''бb''b''лb''b''иb''b''цb''b''іb'' ↩
         b''сb''b''тb''b''иb''b''лb''b''іb''b''вb'' b''зb''b''аb''b''бb''b''аb''b''рb''b' ↩
        'вb''b''лb''b''юb''b''юb''b''тb''b''ьb'' b''тb''b''еb''b''кb''b''сb''b''тb'' b''нb'' ↩
        b''аb'' 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)

    # b''Гb''b''еb''b''тb''b''тb''b''еb''b''рb'' b''тb''b''аb'' b''сb''b''еb''b''тb''b' ↩
        'тb''b''еb''b''рb'' Qproperty
    def getisHomed(self):
        return self._isHomed
    def setisHomed(self, data):
        self._isHomed = data

    # Qproperty
    isHomed = QtCore.pyqtProperty(bool, getisHomed, setisHomed)
```

Ось зразок таблиці стилів для зміни кольору тексту залежно від штату.

У цьому випадку будь-який віджет, заснований на віджеті HomeLabel вище, змінить колір тексту. Зазвичай ви вибираєте певні віджети за допомогою `HomeLabel #specific_widget_name[homed=true]`:

```
HomeLabel[homed=true] {
    color: green;
}
HomeLabel[homed=false] {
    color: red;
}
```

12.10.6 Використання таблиць стилів для зміни властивостей власного віджета

```
class Label(QLabel):
    def __init__(self, parent=None):
        super(Label, self).__init__(parent)
        alternateFont0 = self.font

    # b''Гb''b''eb''b''тb''b''тb''b''eb''b''pb'' b''тb''b''ab'' b''cb''b''eb''b''тb''b' ↩
        'тb''b''eb''b''pb'' Qproperty
    def getFont0(self):
        return self.aleternateFont0
    def setFont0(self, value):
        self.alternateFont0(value)
    # Qproperty
    styleFont0 = pyqtProperty(QFont, getFont0, setFont0)
```

Зразок таблиці стилів, яка встановлює властивість власного віджета.

```
Label{
    qproperty-styleFont0: ”Times,12,-1,0,90,0,0,0,0,0”;
}
```

12.10.7 Плагіни віджетів

Ми повинні *зареєструвати наш користувацький віджет*, щоб Qt Designer міг його використовувати.

Ось типові приклади.
Їх потрібно додати до `qtvcp/plugins/`.
Потім `qtvcp/plugins/qtvcp_plugin.py` потрібно буде налаштувати для їх *імпорту*.

12.10.7.1 Приклад сітки

```
#!/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 Приклад кнопки SystemTool

```
#!/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 Створення плагіна з діалоговим вікном MenuEntry

Можна додати запис до діалогового вікна, яке з’являється, коли ви клацнете правою кнопкою миші на віджеті в макеті.

Це може робити такі речі, як вибір опцій зручнішим способом.

Це плагін, який використовується для *кнопок дій*.

```
#!/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):

    # b’’Mb’’b’’eb’’b’’тb’’b’’ob’’b’’дb’’ __init__() b’’вb’’b’’иb’’b’’кb’’b’’ob’’b’’pb’’b’ ↩
        ’иb’’b’’cb’’b’’тb’’b’’ob’’b’’вb’’b’’уb’’b’’єb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’лb’’b ↩
        ’’иb’’b’’шb’’b’’eb’’ b’’дb’’b’’лb’’b’’яb’’ b’’нb’’b’’ab’’b’’лb’’b’’ab’’b’’шb’’b’ ↩
        ’тb’’b’’уb’’b’’вb’’b’’ab’’b’’нb’’b’’нb’’b’’яb’’ b’’пb’’b’’лb’’b’’ab’’b’’гb’’b’’іb’’b ↩
        ’’нb’’b’’ab’’ b’’тb’’b’’ab’’ b’’вb’’b’’иb’’b’’зb’’b’’нb’’b’’ab’’b’’чb’’b’’eb’’b’ ↩
        ’нb’’b’’нb’’b’’яb’’ b’’йb’’b’’ob’’b’’гb’’b’’ob’’
    # b’’іb’’b’’нb’’b’’іb’’b’’цb’’b’’іb’’b’’ab’’b’’лb’’b’’іb’’b’’зb’’b’’ob’’b’’вb’’b’’ab’’b ↩
        ’’нb’’b’’ab’’ b’’зb’’b’’мb’’b’’іb’’b’’нb’’b’’нb’’b’’ab’’.
    def __init__(self, parent=None):
        super(ActionButtonPlugin, self).__init__(parent)
        self.initialized = False

    # b’’Mb’’b’’eb’’b’’тb’’b’’ob’’b’’дb’’b’’иb’’ initialize() b’’тb’’b’’ab’’ isInitialized ↩
        () b’’дb’’b’’ob’’b’’зb’’b’’вb’’b’’ob’’b’’лb’’b’’яb’’b’’юb’’b’’тb’’b’’ьb’’ b’’пb’’b’ ↩
        ’лb’’b’’ab’’b’’гb’’b’’іb’’b’’нb’’b’’уb’’ b’’нb’’b’’ab’’b’’лb’’b’’ab’’b’’шb’’b’’тb’’b ↩
        ’’уb’’b’’вb’’b’’ab’’b’’тb’’b’’иb’’
    # b’’бb’’b’’уb’’b’’дb’’b’’ьb’’-b’’яb’’b’’кb’’b’’іb’’ b’’нb’’b’’eb’’b’’ob’’b’’бb’’b’ ↩
        ’xb’’b’’іb’’b’’дb’’b’’нb’’b’’іb’’ b’’pb’’b’’eb’’b’’cb’’b’’уb’’b’’pb’’b’’cb’’b’’иb’’, ↩
         b’’гb’’b’’ab’’b’’pb’’b’’ab’’b’’нb’’b’’тb’’b’’уb’’b’’юb’’b’’чb’’b’’иb’’, b’’щb’’b’ ↩
        ’ob’’ b’’цb’’b’’eb’’ b’’мb’’b’’ob’’b’’жb’’b’’eb’’ b’’cb’’b’’тb’’b’’ab’’b’’тb’’b’ ↩
        ’иb’’b’’cb’’b’’яb’’ b’’лb’’b’’иb’’b’’шb’’b’’eb’’ b’’ob’’b’’дb’’b’’иb’’b’’нb’’ b’ ↩
```

```
        ’pb’’b’’ab’’b’’zb’’ b’’db’’b’’lb’’b’’yb’’ b’’kb’’b’’ob’’b’’jb’’b’’nb’’b’’ob’’b’’gb’’ ↩
        b’’ob’’
    # b’’pb’’b’’lb’’b’’ab’’b’’gb’’b’’ib’’b’’nb’’b’’ab’’.
    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

    # b’’Cb’’b’’eb’’b’’yb’’ b’’fb’’b’’ab’’b’’6b’’b’’pb’’b’’ib’’b’’vb’’b’’nb’’b’’ib’’b’’yb’’ ↩
         b’’mb’’b’’eb’’b’’tb’’b’’ob’’b’’db’’ b’’cb’’b’’tb’’b’’vb’’b’’ob’’b’’pb’’b’’ub’’b’ ↩
        ’eb’’ b’’nb’’b’’ob’’b’’vb’’b’’ib’’ b’’eb’’b’’kb’’b’’zb’’b’’eb’’b’’mb’’b’’pb’’b’’lb’’ ↩
        b’’yb’’b’’pb’’b’’ib’’ b’’nb’’b’’ab’’b’’sb’’b’’ob’’b’’gb’’b’’ob’’ b’’vb’’b’’lb’’b’ ↩
        ’ab’’b’’cb’’b’’nb’’b’’ob’’b’’gb’’b’’ob’’ b’’vb’’b’’ib’’b’’db’’b’’jb’’b’’eb’’b’’tb’’b ↩
        ’’ab’’
    def createWidget(self, parent):
        return ActionButton(parent)

    # b’’Cb’’b’’eb’’b’’yb’’ b’’mb’’b’’eb’’b’’tb’’b’’ob’’b’’db’’ b’’pb’’b’’ob’’b’’vb’’b’ ↩
        ’eb’’b’’pb’’b’’tb’’b’’ab’’b’’eb’’ b’’nb’’b’’ab’’b’’zb’’b’’vb’’b’’yb’’ b’’kb’’b’’lb’’ ↩
        b’’ab’’b’’cb’’b’’yb’’ b’’kb’’b’’ob’’b’’pb’’b’’ib’’b’’cb’’b’’tb’’b’’yb’’b’’vb’’b’ ↩
        ’ab’’b’’cb’’b’’sb’’b’’kb’’b’’ob’’b’’gb’’b’’ob’’ b’’vb’’b’’ib’’b’’db’’b’’jb’’b’’eb’’b ↩
        ’’tb’’b’’ab’’
    def name(self):
        return ”ActionButton”

    # b’’Pb’’b’’ob’’b’’vb’’b’’eb’’b’’pb’’b’’tb’’b’’ab’’b’’eb’’ b’’nb’’b’’ab’’b’’zb’’b’’vb’’ ↩
        b’’yb’’ b’’gb’’b’’pb’’b’’yb’’b’’pb’’b’’ib’’ b’’yb’’ b’’vb’’b’’ib’’b’’kb’’b’’nb’’b’ ↩
        ’ib’’ b’’vb’’b’’ib’’b’’db’’b’’jb’’b’’eb’’b’’tb’’b’’ab’’ Qt Designer
    def group(self):
        return ”LinuxCNC - Controller”

    # b’’Pb’’b’’ob’’b’’vb’’b’’eb’’b’’pb’’b’’tb’’b’’ab’’b’’eb’’ b’’zb’’b’’nb’’b’’ab’’b’’cb’’ ↩
        b’’ob’’b’’kb’’
    def icon(self):
        return QtGui.QIcon(QtGui.QPixmap(ICON.get_path(’actionbutton’)))

    # b’’Pb’’b’’ob’’b’’vb’’b’’eb’’b’’pb’’b’’tb’’b’’ab’’b’’eb’’ b’’kb’’b’’ob’’b’’pb’’b’’ob’’ ↩
        b’’tb’’b’’kb’’b’’ib’’b’’yb’’ b’’ob’’b’’pb’’b’’ib’’b’’cb’’ b’’pb’’b’’ib’’b’’db’’b’ ↩
        ’kb’’b’’ab’’b’’zb’’b’’kb’’b’’ib’’
    def toolTip(self):
        return ”Action button widget”

    # b’’Pb’’b’’ob’’b’’vb’’b’’eb’’b’’pb’’b’’tb’’b’’ab’’b’’eb’’ b’’kb’’b’’ob’’b’’pb’’b’’ob’’ ↩
        b’’tb’’b’’kb’’b’’ib’’b’’yb’’ b’’ob’’b’’pb’’b’’ib’’b’’cb’’ b’’kb’’b’’ob’’b’’pb’’b’ ↩
        ’ib’’b’’cb’’b’’tb’’b’’yb’’b’’vb’’b’’ab’’b’’cb’’b’’sb’’b’’ob’’b’’gb’’b’’ob’’ b ↩
        ’’vb’’b’’ib’’b’’db’’b’’jb’’b’’eb’’b’’tb’’b’’ab’’ b’’db’’b’’lb’’b’’yb’’ b’’vb’’b’ ↩
        ’ib’’b’’kb’’b’’ob’’b’’pb’’b’’ib’’b’’cb’’b’’tb’’b’’ab’’b’’nb’’b’’nb’’b’’yb’’ b’’vb’’  ↩
        b’’pb’’b’’ob’’b’’zb’’b’’db’’b’’ib’’b’’lb’’b’’ib’’ «b’’Wb’’b’’ob’’
    # ”b’’Cb’’b’’eb’’?” b’’db’’b’’ob’’b’’vb’’b’’ib’’b’’db’’b’’kb’’b’’ob’’b’’vb’’b’’eb’’ b’ ↩
        ’pb’’b’’ob’’b’’vb’’b’’ib’’b’’db’’b’’ob’’b’’mb’’b’’lb’’b’’eb’’b’’nb’’b’’nb’’b’’yb’’ b ↩
        ’’db’’b’’lb’’b’’yb’’ b’’vb’’b’’ib’’b’’db’’b’’jb’’b’’eb’’b’’tb’’b’’ab’’.
    def whatsThis(self):
        return ””
```

```
    # b''Пb''b''ob''b''вb''b''eb''b''pb''b''тb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb'' ↩
        b''eb''b''нb''b''нb''b''яb'' True, b''яb''b''кb''b''щb''b''ob'' b''кb''b''ob''b' ↩
        'pb''b''иb''b''cb''b''тb''b''уb''b''вb''b''ab''b''цb''b''ьb''b''кb''b''иb''b''йb'' b ↩
        ''вb''b''іb''b''дb''b''жb''b''eb''b''тb'' b''дb''b''іb''b''єb'' b''яb''b''кb'' b' ↩
        'кb''b''ob''b''нb''b''тb''b''eb''b''йb''b''нb''b''eb''b''pb'' b''дb''b''лb''b''яb''  ↩
        b''іb''b''нb''b''шb''b''иb''b''хb'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''b' ↩
        'іb''b''вb'';
    def isContainer(self):
        return False

    # b''Пb''b''ob''b''вb''b''eb''b''pb''b''тb''b''ab''b''єb'' XML-b''ob''b''пb''b''иb''b' ↩
        'cb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''pb''b''ab'' b''кb'' ↩
        b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b''ab''b''цb''b''ьb''b''кb''b''ob'' ↩
        b''гb''b''ob'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''ab'', b''яb''b''кb''b' ↩
        'иb''b''йb'' b''ob''b''пb''b''иb''b''cb''b''уb''b''єb''
    # b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''ab'' b''зb''b' ↩
        'ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'' b ↩
        ''дb''b''лb''b''яb'' b''йb''b''ob''b''гb''b''ob'' b''вb''b''лb''b''ab''b''cb''b' ↩
        'тb''b''иb''b''вb''b''ob''b''cb''b''тb''b''eb''b''йb''.
    def domXml(self):
        return '<widget class="ActionButton" name="actionbutton" />\n'

    # b''Пb''b''ob''b''вb''b''eb''b''pb''b''тb''b''ab''b''єb'' b''мb''b''ob''b''дb''b''уb'' ↩
        b''лb''b''ьb'', b''щb''b''ob'' b''мb''b''іb''b''cb''b''тb''b''иb''b''тb''b''ьb'' b' ↩
        'кb''b''лb''b''ab''b''cb'' b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''уb''b''вb''b ↩
        ''ab''b''цb''b''ьb''b''кb''b''ob''b''гb''b''ob'' b''вb''b''іb''b''дb''b''жb''b''eb'' ↩
        b''тb''b''ab''. b''Bb''b''іb''b''нb'' b''мb''b''ob''b''жb''b''eb'' b''мb''b''іb''b' ↩
        'cb''b''тb''b''иb''b''тb''b''иb''
    # b''шb''b''лb''b''яb''b''хb'' b''дb''b''ob'' b''мb''b''ob''b''дb''b''уb''b''лb''b' ↩
        'яb''.
    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("b''Hb''b''ab''b''лb''b''ab''b''шb''b''тb''b''yb''b''вb'' ↩
          b''ab''b''нb''b''нb''b''яb'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b''тb''b' ↩
```

```
          'pb''b''ib''b''вb''"))

   def updateWidget(self):

      formWindow = QDesignerFormWindowInterface.findFormWindow(self.widget)
      if formWindow:
          formWindow.cursor().setProperty("estop_action",
              QtCore.QVariant(self.c_estop.isChecked()))
      self.accept()

b''кb''b''лb''b''ab''b''cb'' 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 Фрагменти коду файлу обробника QtVCP

12.11.1 Завантаження/збереження файлу налаштувань

Ось як **завантажити та зберегти налаштування під час запуску та закриття**.

Передумови

- *Параметр файлу налаштувань має бути встановлений* у віджеті `ScreenOptions`.
- Шлях до файлу налаштувань має бути встановлений у конфігурації INI.

Параметри читання під час запуску У функції **`def initialized__(self):`** додайте:

```
if self.w.PREFS_:
    # b''ib''b''мb'''b''яb'' b''зb''b''мb''b''ib''b''нb''b''нb''b''ob''b''їb'' (b''ib''b' ↩
        'мb'''b''яb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''yb'', b''зb''b''нb''b''ab''b' ↩
        'чb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb'' ↩
       b''чb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'', b''тb''b''иb''b''пb'', b' ↩
        'ib''b''мb'''b''яb'' b''pb''b''ob''b''зb''b''дb''b''ib''b''лb''b''yb'')
    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')
```

Уподобання щодо письма наприкінці терміну У функції **`closing_cleanup__()`** додайте:

```
if self.w.PREFS_:
    # b''ib''b''мb'''b''яb'' b''зb''b''мb''b''ib''b''нb''b''нb''b''ob''b''їb'' (b''ib''b' ↩
        'мb'''b''яb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''yb'', b''ib''b''мb'''b''яb'' b' ↩
        'зb''b''мb''b''ib''b''нb''b''нb''b''ob''b''їb'', b''тb''b''иb''b''пb'', b''ib''b' ↩
        'мb'''b''яb'' b''pb''b''ob''b''зb''b''дb''b''ib''b''лb''b''yb'')
    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 Використання `QSettings` для читання/збереження змінних

Ось як **завантажувати та зберігати змінні за допомогою функцій `QSettings`** у PyQt:

Гарні практики

- *Використовуйте Група для упорядкування та унікальності імен.*
- _Враховувати значення none, яке повертається під час читання параметра, який не має запису.
- *Встановіть значення за замовчуванням для першого запуску* за допомогою синтаксису `або _<значення_за_замовчуванням>_`.

Note
Файл фактично зберігається в `~/.config/QtVcp`

Приклад У цьому прикладі:

- Ми додаємо `or 20` та `or 2.5` як значення за замовчуванням.
- Назви `MyGroupName`, `int_value`, `float_value`, `myInteger` та `myFloat` визначаються користувачем.
- У функції **`def initialized__(self):`** додайте:

```
# b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b''ab''b' ↩
    'pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''cb''b''ob''b''pb''b''тb''b''yb''b' ↩
    'вb''b''ab''b''нb''b''нb''b''яb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''ab''b''нb''b' ↩
    'иb''b''xb'' b''cb''b''тb''b''ob''b''вb''b''пb''b''цb''b''ib''b''вb''
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()
```

- У функції **`def closing_cleanup__(self):`** додайте:

```
# b''зb''b''бb''b''eb''b''pb''b''eb''b''жb''b''eb''b''нb''b''нb''b''яb'' b''зb''b''нb''b' ↩
    'ab''b''чb''b''eb''b''нb''b''ьb'' b''зb''b''ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b' ↩
    'ob''b''гb''b''ob''b''юb'' 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 Додати базовий редактор стилів

Можливість **редагувати стиль на екрані, що працює**, зручна.

Імпортуйте модуль StyleSheetEditor у РОЗДІЛ ІМПОРТ:

```
b''зb'' qtvcp.widgets.stylesheeteditor b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb''b' ↩
    'вb''b''ab''b''тb''b''иb'' StyleSheetEditor b''яb''b''кb'' SSE
```

Створіть екземпляр модуля StyleSheetEditor у розділі INSTANTIATE SECTION:

```
STYLEEDITOR = SSE()
```

Створіть прив'язку клавіш у СЕКЦІЇ ІНІЦІАЛІЗАЦІЇ: У функції `+__init__.(self, halcomp, widgets, paths):+` додано:

```
KEYBIND.add_call('Key_F12','on_keycall_F12')
```

Створіть функцію прив'язки клавіш у розділі KEYBINDING SECTION:

```
def on_keycall_F12(self,event,state,shift,cntrl):
    if state:
        STYLEEDITOR.load_dialog()
```

12.11.4 Запит на запис у діалоговому вікні

QtVCP використовує повідомлення `STATUS` для **спливаючого відображення та повернення інформації з діалогових вікон**.

Попередньо створені діалогові вікна відстежують свою останню позицію та включають опції затінення фокуса та звуку.

Щоб *отримати інформацію з діалогового вікна*, потрібно використовувати повідомлення `STATUS` **`general`**.

Імпортуйте та створіть екземпляр модуля Status у IMPORT SECTION

```
b''зb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' qtvcp.core b''Cb''b''тb''b''ab''b ↩
    ''нb''
STATUS = Status()
```

Це завантажує та ініціалізує бібліотеку `Status`.

Реєстрація функції для повідомлень STATUS general у розділі INITIALIZE SECTION У функції `+__init__.(self, halcomp, widgets, paths)+`:

```
STATUS.connect('general',self.return_value)
```

Це реєструє `STATUS` для виклику функції `self.return_value`, коли надсилається загальне повідомлення

Додати функцію запиту діалогового вікна введення в розділі ЗАГАЛЬНІ ФУНКЦІЇ

```
def request_number(self):
    mess = {'NAME':'ENTRY','ID':'FORM__NUMBER', 'TITLE':'Set Tool Offset'}
    STATUS.emit('dialog-request', mess)
```

Функція

- створює словник Python за допомогою:
  - **NAME** – потрібно встановити унікальну назву запуску *dialogs*.
    NAME встановлює, який діалог запитувати.
    ENTRY або CALCULATOR дозволяє вводити числа.
  - **ID** - має бути встановлено на *унікальне ім'я, яке надає функція*. ID має бути унікальним ключем.
  - **TITLE** встановлює заголовок діалогового вікна.
  - **До dict можна додавати довільні дані**. Діалогове вікно проігнорує їх, але поверне їх до коду повернення.
- Надсилає dict як повідомлення **dialog-request** STATUS

Додати функцію обробки даних повідомлень у розділі CALLBACKS FROM STATUS.

```
# b''Ob''b''6b''b''pb''b''ob''b''6b''b''ив''b''тb''b''ив'' b''пb''b''ob''b''вb''b''ib''b' ↩
    'дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' STATUS b''вb''b''ib''b''дb'' 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))
```

Це захоплює всі загальні повідомлення, тому необхідно перевірити тип діалогового вікна та ідентифікаційний код, щоб підтвердити, що це наше діалогове вікно. У цьому випадку ми запросили діалогове вікно ENTRY, а наш унікальний ідентифікатор був FORM_NUMBER, тому тепер ми знаємо, що це повідомлення для нас. Діалогові вікна ENTRY або CALCULATOR повертають число з плаваючою комою.

12.11.5 Промовте привітання стартапу

Для цього потрібна бібліотека espeak, встановлена в системі.

Імпортуйте та створіть екземпляр Status у розділі IMPORT

```
b''зb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' qtvcp.core b''Cb''b''тb''b''ab''b ↩
    ''нb''
STATUS = Status()
```

Передавати голосове повідомлення в розділі «ІНІЦІАЛІЗАЦІЯ» У функції init*(self, halcomp, widgets, paths)*:

```
STATUS.emit('play-alert','b''Гb''b''Ob''b''Bb''b''Ob''b''Pb''b''Иb''b''Tb''b''Иb'' b''Бb''b ↩
    ''yb''b''дb''b''ьb'' b''лb''b''ab''b''cb''b''кb''b''ab'', b''нb''b''eb'' b''зb''b''ab''b ↩
    ''6b''b''yb''b''дb''b''ьb'' b''зb''b''мb''b''ab''b''cb''b''тb''b''иb''b''тb''b''иb'' b' ↩
    'шb''b''лb''b''яb''b''xb''b''иb''.')
```

SPEAK – це ключове слово: *все після нього буде вимовлятися*.

12.11.6 Функції панелі інструментів

Кнопки панелі інструментів та підменю додано в Qt Designer, але код, який змушує їх виконувати певні дії, додано до файлу обробника. Щоб **додати підменю** в Qt Designer:

- Додайте `Qaction`, ввівши його в стовпці панелі інструментів, а потім натиснувши значок + праворуч.
- Це додасть підстовпець, у який потрібно ввести ім’я.
- Тепер оригінальна `Qaction` буде `Qmenu`.
- Тепер видаліть `Qaction`, який ви додали до `Qmenu`, меню залишиться як меню.

У цьому прикладі ми припускаємо, що ви додали панель інструментів з одним підменю та трьома діями. Ці дії будуть налаштовані для створення:

- меню вибору нещодавніх файлів,
- дія спливаючого діалогового вікна «Про нас»,
- дія виходу з програми та
- дія функції, визначеної користувачем.

`objectName` кнопки панелі інструментів використовується для ідентифікації кнопки під час її налаштування - *допомога з описовими назвами*.

Використовуючи меню редактора дій, клацніть правою кнопкою миші та виберіть «Редагувати». Відредагуйте назву об’єкта, текст і тип кнопки для відповідної дії.

У цьому прикладі:

- назва підменю має бути `menuRecent`,
- назви дій мають бути `actionAbout`, `actionQuit`, `actionMyFunction`

Завантажує бібліотеку toolbar_actions у РОЗДІЛ ІМПОРТУ

```
from qtvcp.lib.toolbar_actions import ToolBarActions
```

Створити екземпляр модуля ToolBarActions у INSTANTIATE LIBRARY SECTION

```
TOOLBAR = ToolBarActions()
```

Налаштуйте підменю та дії в РОЗДІЛІ СПЕЦІАЛЬНИХ ФУНКЦІЙ У функції `def initialized__(s` додайте:

```
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)
```

Визначте функцію користувача в ЗАГАЛЬНОМУ РОЗДІЛІ ФУНКЦІЙ

```
def my_function(self, widget, state):
    print('My function State = ()'.format(state))
```

Функція, яка викликається, якщо натиснуто кнопку дії «Моя функція».

12.11.7 Додати HAL-виводи, що викликають функції

Таким чином, вам *не потрібно опитувати стан вхідних контактів*.

Завантажує бібліотеку Qhal у РОЗДІЛ ІМПОРТУ

```
from qtvcp.core import Qhal
```

Це дозволяє доступ до **HAL-компонента QtVCP**. Додаткова інформація: Qhal
Функція newPin Qhal повертає об’єкт QPin. Додаткова інформація: QPin

Створити екземпляр Qhal у РОЗДІЛІ `INSTANTIATE LIBRARY

```
QHAL = Qhal()
```

Додати функцію, яка викликається при зміні стану виводу У функції `initialized__` переконайте
що є запис, подібний до цього:

```
########################################
# b''Cb''b''пв''b''eb''b''цb''b''ib''b''ab''b''лв''b''ьb''b''нв''b''ib'' b''фb''b''yb''b' ↩
    'нв''b''кв''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вв''b''ив''b''кb''b''лв''b''ив''b' ↩
    'кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яв'' b''зв'' QtVCP
########################################

# b''нв''b''ab'' b''цb''b''ьb''b''ob''b''мв''b''yb'' b''eb''b''тb''b''ab''b''пв''b''ib'':
# b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''ив'' b''ib''b''нв''b''cb''b''тb''b''ab''b' ↩
    'нв''b''цb''b''ib''b''юb''b''юb''b''тb''b''ьb''b''cb''b''яв''.
# b''кb''b''ob''b''нв''b''тb''b''ab''b''кb''b''тb''b''ив'' HAL b''пв''b''ob''b''бb''b''yb'' ↩
    b''дb''b''ob''b''вb''b''ab''b''нв''b''ib'', b''ab''b''лв''b''eb'' HAL b''нв''b''eb'' b' ↩
    'гb''b''ob''b''тb''b''ob''b''вb''b''ив''b''йв'' b''дb''b''ob'' b''pb''b''ob''b''бb''b' ↩
    'ob''b''тb''b''ив''
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))
```

Визначте функцію, яка викликається зміною стану виводу, у ЗАГАЛЬНОМУ РОЗДІЛІ ФУНКЦІЙ

```
####################
# b''зb''b''ab''b''гb''b''ab''b''лв''b''ьb''b''нв''b''ib'' b''фb''b''yb''b''нв''b''кb''b' ↩
    'цb''b''ib''b''їb'' #
####################

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()
```

Ця функція припускає, що існує віджет вкладок з назвою `mainTab`, який містить вкладки з назвами `tab_auto`, `tab_graphics`, `tab_filemanager` та `tab_mdi`.

Таким чином, кнопка запуску циклу працює по-різному залежно від того, яка вкладка відображається.

Це спрощено — *перевірка стану та перехоплення помилок можуть бути корисними*.

12.11.8 Безпосереднє читання/запис системних HAL-виводів

Іноді потрібно зчитати системний пін, а створення HAL-піна та підключення до нього вимагає більше роботи, ніж потрібно. Ви можете зчитати його безпосередньо, не підключаючись до нього.

Ось як зчитати вивід, параметр або сигнал:

```
self.h.hal.get_value('spindle.0.at-speed')

# b''ab''b''6b''b''ob'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b' ↩
    'вb''b''уb''b''йb''b''тb''b''eb'' b''6b''b''ib''b''6b''b''лb''b''ib''b''ob''b''тb''b' ↩
    'eb''b''кb''b''уb'' Qhal b''ob''b''cb''b''ьb'' b''тb''b''ab''b''кb'':
QHAL.getValue('spindle.0.at-speed')
```

Ось як записати на непідключений контакт або некерований сигнал:

```
self.h.hal.set_p('componentName.pinName','10')
self.h.hal.set_s('componentName.signalName','10')

# b''ab''b''6b''b''ob'' b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b' ↩
    'вb''b''уb''b''йb''b''тb''b''eb'' b''6b''b''ib''b''6b''b''лb''b''ib''b''ob''b''тb''b' ↩
    'eb''b''кb''b''уb'' Qhal b''ob''b''cb''b''ьb'' b''тb''b''ab''b''кb'':
QHAL.setPin('componentName.pinName', '10')
QHAL.setSignal('componentName.signalName', '10')
```

Використання *self.h.hal* або *QHAL.hal* дозволяє отримати доступ до функцій модуля HAL Python: Інтерфейс Python HAL

12.11.9 Додати спеціальний повзунок максимальної швидкості на основі відсотків

Іноді потрібно **створити віджет для виконання чогось, чого немає вбудованого**. Вбудований повзунок максимальної швидкості діє на одиниці за хвилину, тут ми покажемо, як це зробити на відсотках.

Команда **STATUS** гарантує, що повзунок змінюватиметься, якщо LinuxCNC змінює поточну максимальну швидкість.

valueChanged.connect() *викликає функцію, коли повзунок переміщується*.

У Qt Designer додайте віджет **QSlider** під назвою mvPercent, а потім додайте наступний код до файлу обробника:

```
############################
# b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Cb''b''Пb''b''Eb''b''Цb''b''Ib''b''Ab''b' ↩
    'Лb''b''Ьb''b''Hb''b''Иb''b''Xb'' b''Фb''b''Уb''b''Hb''b''Kb''b''Цb''b''Ib''b''Йb'' #
############################

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)

####################
# b''3b''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Hb''b''Ib'' b''Фb''b''Уb''b''Hb''b''Kb''b' ↩
    'Цb''b''Ib''b''Їb'' #
```

```
####################

def setMVPercentValue(self, value):
    ACTION.SET_MAX_VELOCITY_RATE(INFO.MAX_TRAJ_VELOCITY * (value/100.0))
```

12.11.10 Увімкнення та вимкнення безперервного поштовху

Зазвичай вибір безперервного поштовху – це тимчасова кнопка, яка вимагає від вас вибору попереднього кроку поштовху.

Ми створимо кнопку, яка перемикатиметься між безперервним поштовхом та будь-яким уже вибраним приростом.

У дизайнері Qt:

- Додати `ActionButton` без жодної дії
- Назвіть це `btn_toggle_continuous`.
- Встановіть властивість `checkable` `AbstractButton` у значення `True`.
- Встановіть властивості `incr_imperial_number` та `incr_mm_number` об’єкта `ActionButton` на `0`.
- Використовуйте редактор слотів Qt Designer, щоб за допомогою сигналу кнопки `clicked(bool)` викликати функцію обробки форми `toggle_continuous_clicked()`.
  Докладнішу інформацію див. у розділі Використання Qt Designer для додавання слотів.

Потім додайте цей фрагмент коду до файлу обробника у функції `initialized__`:

```
# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''ib''b''нb''b''cb''b''тb''b''ab''b' ↩
    'нb''b''цb''b''ib''b''юb''b''юb''b''тb''b''ьb''b''cb''b''яb''.
# b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb''b''иb'' HAL b''пb''b''ob''b''бb''b''уb'' ↩
    b''дb''b''ob''b''вb''b''ab''b''нb''b''ib'', b''ab''b''лb''b''eb'' HAL b''нb''b''eb'' b' ↩
    'гb''b''ob''b''тb''b''ob''b''вb''b''иb''b''йb'' b''дb''b''ob'' b''pb''b''ob''b''бb''b' ↩
    'ob''b''тb''b''иb''.
def initialized__(self):
    STATUS.connect('jogincrement-changed', \
        lambda w, d, t: self.record_jog_incr(d,t) \
        )
    # b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''пb''b''pb'' ↩
        b''иb''b''pb''b''ib''b''cb''b''тb'' b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b' ↩
        'вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b''мb'' b''дb''b''лb''b''яb''  ↩
        b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''eb''b''нb''b''нb''b''яb'' b''нb''b' ↩
        'ab''b''зb''b''ab''b''дb''
    self.L_incr = 0.01
    self.L_text = "0.01in"
```

У розділі «ЗАГАЛЬНІ ФУНКЦІЇ» додайте:

```
####################
# b''Зb''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Нb''b''Ib'' b''Фb''b''Уb''b''Нb''b''Кb''b' ↩
    'Цb''b''Ib''b''Їb'' #
####################

# b''яb''b''кb''b''щb''b''ob'' b''цb''b''eb'' b''нb''b''eb'' b''бb''b''eb''b''зb''b''пb''b' ↩
    'eb''b''pb''b''eb''b''pb''b''вb''b''нb''b''иb''b''йb'' b''pb''b''уb''b''хb'', b''зb''b' ↩
    'ab''b''пb''b''иb''b''шb''b''ib''b''тb''b''ьb'' b''ob''b''cb''b''тb''b''ab''b''нb''b' ↩
    'нb''b''ib''b''йb'' b''кb''b''pb''b''ob''b''кb'' b''пb''b''ob''b''шb''b''тb''b''ob''b' ↩
    'вb''b''хb''b''уb''
```

```
# b''тb''b''ab'' b''вb''b''иb''b''мb''b''кb''b''нb''b''іb''b''тb''b''ьb'' b''кb''b''нb''b' ↩
    'ob''b''пb''b''кb''b''уb'' b''бb''b''eb''b''зb''b''пb''b''eb''b''pb''b''eb''b''pb''b' ↩
    'вb''b''нb''b''ob''b''гb''b''ob'' b''pb''b''уb''b''хb''b''уb''
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)
```

У розділі `ЗВОРОТНІ ВИКЛИКИ З ФОРМИ` додайте:

```
######################
# b''Зb''b''Bb''b''Ob''b''Pb''b''Ob''b''Tb''b''Hb''b''Ib'' b''Bb''b''Иb''b''Kb''b''Лb''b' ↩
    'Иb''b''Kb''b''Иb'' b''Зb'' b''Фb''b''Ob''b''Pb''b''Mb''b''Иb'' #
######################

def toggle_continuous_clicked(self, state):
    if state:
        # b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''бb''b' ↩
            'eb''b''зb''b''пb''b''eb''b''pb''b''eb''b''pb''b''вb''b''нb''b''іb''b''cb''b' ↩
            'тb''b''ьb'' (b''вb''b''иb''b''кb''b''лb''b''иb''b''кb'' b''фb''b''уb''b''нb''b' ↩
            'кb''b''цb''b''іb''b''їb'' b''кb''b''нb''b''ob''b''пb''b''кb''b''иb'' b''дb''b' ↩
            'іb''b''їb'')
        self.w.btn_toggle_continuous.incr_action()
    else:
        # b''cb''b''кb''b''иb''b''нb''b''уb''b''тb''b''иb'' b''pb''b''ab''b''нb''b''іb''b' ↩
            'шb''b''eb'' b''зb''b''ab''b''пb''b''иb''b''cb''b''ab''b''нb''b''иb''b''йb'' b' ↩
            'пb''b''pb''b''иb''b''pb''b''іb''b''cb''b''тb''
        ACTION.SET_JOG_INCR(self.L_incr, self.L_text)
```

12.11.11 Патч класу Віджет файлового менеджера

Note
Класове патчування (monkey patching) трохи нагадує «чорну магію», тому використовуйте його тільки в разі потреби. Основна проблема полягає в тому, що якщо функції бібліотеки віджетів змінюються під час розробки, вони можуть перестати працювати.
Віджет «Файловий менеджер» призначений для завантаження вибраної програми в LinuxCNC. Але, можливо, ви хочете спочатку надрукувати ім’я файлу.

Ми можемо «класовий патч» бібліотеки, щоб перенаправити виклик функції.
Ви можете зробити цей класовий патч всередині або зовні екземпляра HandlerClass.
Це змінить значення, яке представляє «self» у функції.
За межами HanderClass «self» буде виправленим екземпляром класу.
Всередині HanderClass «self» буде екземпляром HandlerClass.
Це змінить функції/змінні, до яких ви можете отримати доступ у функції.
Тут ми показуємо приклад всередині HandlerClass:
У розділі `IMPORT SECTION` додайте:

```
b''зb'' qtvcp.widgets.file_manager b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb''b''вb'' ↩
    b''ab''b''тb''b''иb'' FileManager b''яb''b''кb'' FM
```

Ось що ми збираємося зробити:

1. *Зберегти посилання на оригінальну функцію* (1), щоб ми все ще могли її викликати
2. *Перенаправте клас на виклик нашої користувацької функції* (2) у файлі обробника.

```
########################################
# b''Cb''b''пb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''yb'' ↩
    b''нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b' ↩
    'иb''b''кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP    #
########################################

# b''Дb''b''лb''b''яb'' b''зb''b''мb''b''ib''b''нb''b''иb'' b''фb''b''yb''b''нb''b' ↩
    'кb''b''цb''b''ib''b''йb'' b''yb'' b''вb''b''ib''b''дb''b''жb''b''eb''b''тb''b' ↩
    'ab''b''хb'' b''мb''b''иb'' b''мb''b''ob''b''жb''b''eb''b''мb''b''ob'' b''вb''b' ↩
    'иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b''yb''b''вb''b''ab''b ↩
    ''тb''b''иb'' «b''пb''b''ab''b''тb''b''чb'' b''кb''b''лb''b''ab''b''cb''b''yb''».
# b''Пb''b''ab''b''тb''b''чb'' b''кb''b''лb''b''ab''b''cb''b''yb'' b''мb''b''ab''b' ↩
    'єb'' b''бb''b''yb''b''тb''b''иb'' b''вb''b''иb''b''кb''b''ob''b''нb''b''ab''b' ↩
    'нb''b''иb''b''йb'' b''дb''b''ob'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b' ↩
    'нb''b''нb''b''яb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''pb'' ↩
    b''ab'' b''кb''b''лb''b''ab''b''cb''b''yb''.
def class_patch__(self):
    self.old_load = FM.load # b''зb''b''бb''b''eb''b''pb''b''eb''b''гb''b''тb''b''иb''  ↩
        b''пb''b''ob''b''cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab''  ↩
        b''cb''b''тb''b''ab''b''pb''b''yb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b ↩
        ''юb'' <t>\coref{1}{CO8-1}</t>
    FM.load = self.our_load # b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''яb'' b''пb'' ↩
        b''eb''b''pb''b''eb''b''нb''b''ab''b''пb''b''pb''b''ab''b''вb''b''лb''b''eb''b' ↩
        'нb''b''нb''b''яb'' b''дb''b''ob'' b''нb''b''ab''b''шb''b''ob''b''їb'' b''фb''b ↩
        ''yb''b''нb''b''кb''b''цb''b''ib''b''їb'' b''фb''b''ab''b''йb''b''лb''b''yb'' b ↩
        ''дb''b''eb''b''cb''b''кb''b''pb''b''иb''b''пb''b''тb''b''ob''b''pb''b''ib''b' ↩
        'вb'' <t>\coref{2}{CO8-2}</t>
```

3. *Напишіть власну функцію, щоб замінити оригінальну*:
   Ця функція повинна мати **таку саму сигнатуру, як і оригінальна функція**.
   self є екземпляром HandlerClass, а *не* екземпляром виправленого класу.
   У цьому прикладі ми все ще будемо викликати оригінальну функцію, використовуючи посилання на неї, яке ми записали раніше.
   Вона *вимагає, щоб першим аргументом був екземпляр віджета*, який у цьому випадку є `self.w.filemanager` (ім'я, надане в редакторі Qt Designer).

```
####################
# b''Зb''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Hb''b''Ib'' b''Фb''b''Уb''b''Hb''b''Kb'' ↩
    b''Цb''b''Ib''b''Їb'' #
####################

def our_load(self,fname):
    print(fname)
    self.old_load(self.w.filemanager,fname)
```

Тепер наша користувацька функція виведе шлях до файлу в термінал перед завантаженням файлу. Звісно, нудно, але принцип показано.

Note
Є ще один, дещо інший спосіб, який може мати свої переваги: ви можете *зберегти посилання на оригінальну функцію в оригінальному класі*.
Тут головне — переконатися, що ім'я функції, яке ви використовуєте для збереження, ще не використовується в класі.
Хорошим вибором буде додавання super__ до імені функції.
Ми не будемо використовувати це у вбудованих віджетах QtVCP.

```
########################################
# b''Cb''b''пb''b''eb''b''цb''b''іb''b''ab''b''лb''b''ьb''b''нb''b''іb'' b''фb''b''уb''b' ↩
    'нb''b''кb''b''цb''b''іb''b''їb'', b''щb''b''оb'' b''вb''b''иb''b''кb''b''лb''b''иb'' ↩
    b''кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtVCP
########################################

# b''Дb''b''лb''b''яb'' b''зb''b''мb''b''іb''b''нb''b''иb'' b''фb''b''уb''b''нb''b''кb''b ↩
    ''цb''b''іb''b''йb'' b''уb'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''ab''b''хb'' ↩
     b''мb''b''иb'' b''мb''b''оb''b''жb''b''eb''b''мb''b''оb'' b''вb''b''иb''b''кb''b' ↩
    'оb''b''рb''b''иb''b''cb''b''тb''b''оb''b''вb''b''уb''b''вb''b''ab''b''тb''b''иb'' «b ↩
    ''кb''b''лb''b''ab''b''cb''b''оb''b''вb''b''иb''b''йb'' b''пb''b''ab''b''тb''b''чb''» ↩
    .
# b''Кb''b''лb''b''ab''b''cb''b''оb''b''вb''b''иb''b''йb'' b''пb''b''ab''b''тb''b''чb'' b ↩
    ''пb''b''оb''b''вb''b''иb''b''нb''b''eb''b''нb'' b''бb''b''уb''b''тb''b''иb'' b''вb'' ↩
    b''иb''b''кb''b''оb''b''нb''b''ab''b''нb''b''иb''b''йb'' b''дb''b''оb'' b''іb''b' ↩
    'нb''b''cb''b''тb''b''ab''b''нb''b''цb''b''іb''b''юb''b''вb''b''ab''b''нb''b''нb''b' ↩
    'яb'' b''кb''b''лb''b''ab''b''cb''b''уb''.
def class_patch__(self):
    FM.super__load = FM.load # b''зb''b''бb''b''eb''b''pb''b''eb''b''гb''b''тb''b''иb'' b ↩
        ''пb''b''оb''b''cb''b''иb''b''лb''b''ab''b''нb''b''нb''b''яb'' b''нb''b''ab'' b' ↩
        'cb''b''тb''b''ab''b''pb''b''уb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b' ↩
        'юb'' b''вb'' b''оb''b''pb''b''иb''b''гb''b''іb''b''нb''b''ab''b''лb''b''ьb''b' ↩
        'нb''b''оb''b''мb''b''уb'' b''кb''b''лb''b''ab''b''cb''b''іb''
    FM.load = self.our_load # b''пb''b''eb''b''pb''b''eb''b''нb''b''ab''b''пb''b''pb''b' ↩
        'ab''b''вb''b''иb''b''тb''b''иb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b' ↩
        'юb'' b''нb''b''ab'' b''нb''b''ab''b''шb''b''уb'' b''фb''b''уb''b''нb''b''кb''b' ↩
        'цb''b''іb''b''юb'' b''оb''b''бb''b''pb''b''оb''b''бb''b''кb''b''иb'' b''фb''b' ↩
        'ab''b''йb''b''лb''b''іb''b''вb''

####################
# b''Зb''b''Ab''b''Гb''b''Ab''b''Лb''b''Ьb''b''Нb''b''Іb'' b''Фb''b''Уb''b''Нb''b''Кb''b' ↩
    'Цb''b''Іb''b''Їb'' #
####################

def our_load(self,fname):
    print(fname)
    self.w.filemanager.super__load(fname)
```

12.11.12 Додавання віджетів програмним способом

У деяких ситуаціях можливо **додавати віджети лише за допомогою коду Python**, а не використовув редактор Qt Designer.

Під час програмного додавання віджетів QtVCP іноді потрібно виконати *додаткові кроки*.

Тут ми додамо індикатор швидкості шпинделя та світлодіодний індикатор швидкості до кута віджета вкладки. Qt Designer не підтримує додавання кутових віджетів до вкладок, але PyQt підтримує.

Це скорочений приклад з файлу обробника екрана QtAxis.

Імпортувати необхідні бібліотеки Спочатку нам потрібно імпортувати потрібні бібліотеки, якщо вони ще не імпортовані у файлі обробника:

- `QtWidgets` надає нам доступ до `QProgressBar`,
- `QColor` для *LED color*,
- `StateLED` бібліотека QtVCP, яка використовується для *створення світлодіода шпинделя на швидкості*,
- `Status` використовується для *отримання інформації про стан LinuxCNC*,
- `Info` надає нам *інформацію про конфігурацію машини*.

```
############################
# **** b''Pb''b''Ob''b''3b''b''Дb''b''Ib''b''Лb'' b''Ib''b''Mb''b''Пb''b''Ob''b''Pb''b' ↩
    'Tb'' **** #
############################

from PyQt5 import QtWidgets
from PyQt5.QtGui import QColor
from qtvcp.widgets.state_led import StateLED as LED
from qtvcp.core import Status, Info
```

Створення екземплярів каналів Status та Info `STATUS` та `INFO` ініціалізуються поза класом обробника, щоб бути *глобальними посиланнями* (без self. попереду):

```
########################################
# **** b''pb''b''ob''b''зb''b''дb''b''ib''b''лb'' b''cb''b''тb''b''вb''b''ob''b''pb''b' ↩
    'eb''b''нb''b''нb''b''яb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
    'pb''b''ib''b''вb'' b''бb''b''ib''b''бb''b''лb''b''ib''b''ob''b''тb''b''eb''b''кb'' **** ↩
     #
########################################

STATUS = Status()
INFO = Info()
```

Зареєструйте функцію моніторингу STATUS Для індикатора швидкості обертання шпинделя нам потрібно знати поточну швидкість шпинделя. Для цього ми *реєструємося* за допомогою `STATUS`, щоб:

- *Зловити сигнал* `фактичної-зміни-швидкості-шпинделя`
- *Виклик* функції `self.update_spindle()`

```
########################
# **** b''Ib''b''Hb''b''Ib''b''Цb''b''Ib''b''Ab''b''Лb''b''Ib''b''3b''b''Ab''b''Цb''b''Ib'' ↩
    b''Яb'' **** #
########################
# b''Bb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''дb''b''ob''b''зb''b''вb''b''ob''b' ↩
    'лb''b''яb''b''юb''b''тb''b''ьb'' b''ob''b''тb''b''pb''b''иb''b''мb''b''ab''b''тb''b' ↩
    'иb'' b''дb''b''ob''b''cb''b''тb''b''уb''b''пb'' b''дb''b''ob'' b''вb''b''ib''b''дb''b' ↩
    'жb''b''eb''b''тb''b''ib''b''вb'' b''ib''b''зb'' b''фb''b''ab''b''йb''b''лb''b''ib''b' ↩
    'вb'' QtVCP.
# b''Hb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'' b' ↩
    'вb''b''ib''b''дb''b''жb''b''eb''b''тb''b''иb'' b''тb''b''ab'' b''кb''b''ob''b''нb''b' ↩
    'тb''b''ab''b''кb''b''тb''b''иb'' HAL b''нb''b''eb'' b''ib''b''нb''b''cb''b''тb''b''ab'' ↩
    b''нb''b''цb''b''ib''b''юb''b''юb''b''тb''b''ьb''b''cb''b''яb''.
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))
```

Додайте віджети до вкладки Перед тим, як додавати до них нові елементи, потрібно переконатися, що віджети Qt Designer вже створені. Для цього додаємо виклик функції `self.make_corner_widgets()`, щоб створити додаткові віджети в потрібний момент, тобто в функції `initialized__()`:

```
########################################
# b''Cb''b''nb''b''eb''b''цb''b''ib''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''yb''b' ↩
    'нb''b''кb''b''цb''b''ib''b''їb'', b''щb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
    'кb''b''ab''b''юb''b''тb''b''ьb''b''cb''b''яb'' b''зb'' QtScreen #
########################################

# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''yb'' b''eb''b''тb''b''ab''b''nb''b''ib'':
# b''eb''b''кb''b''зb''b''eb''b''мb''b''nb''b''лb''b''яb''b''pb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b' ↩
    'нb''b''ob''.
# b''nb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''pb''b''ab''b''нb''b''ob'', b' ↩
    'ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''
def initialized__(self):
    self.make_corner_widgets()
```

Створення функцій побудови віджетів Добре, давайте напишемо функцію для створення віджетів та додамо їх до віджета вкладок. Ми припускаємо, що існує віджет вкладок, створений за допомогою Designer, під назвою *rightTab*.

Ми припускаємо, що існує віджет вкладок, створений за допомогою Qt Designer, під назвою `rightTab`.

```
####################
# b''зb''b''ab''b''гb''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''yb''b''нb''b''кb''b' ↩
    'цb''b''ib''b''їb'' #
####################

def make_corner_widgets(self):
    # b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''зb''b''eb''b''лb''b''eb''b''нb'' ↩
        b''иb''b''йb'' b''cb''b''вb''b''ib''b''тb''b''лb''b''ob''b''дb''b''ib''b''ob''b' ↩
        'дb'', b''щb''b''ob'' b''пb''b''ob''b''кb''b''ab''b''зb''b''yb''b''єb'' b''шb''b' ↩
        'вb''b''иb''b''дb''b''кb''b''ib''b''cb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b ↩
        ''дb''b''eb''b''лb''b''яb''
    self.w.led = LED()                                          # <t>\coref{1}{CO8-3}</t>
    self.w.led.setProperty('is_spindle_at_speed_status',True) # <t>\coref{2}{CO8-4}</t>
    self.w.led.setProperty('color',QColor(0,255,0,255))       # <t>\coref{3}{CO8-5}</t>
    self.w.led.hal_init(HAL_NAME = 'spindle_is_at_speed')     # <t>\coref{4}{CO8-6}</t>

    # b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''пb''b''лb''b''ab''b''нb''b''кb'' ↩
        b''yb'' b''шb''b''вb''b''иb''b''дb''b''кb''b''ob''b''cb''b''тb''b''ib'' b''шb''b' ↩
        'пb''b''иb''b''нb''b''дb''b''eb''b''лb''b''яb''
    self.w.rpm_bar = QtWidgets.QProgressBar()                   # <t>\coref{5}{CO8-7}</t>
    self.w.rpm_bar.setRange(0, INFO.MAX_SPINDLE_SPEED)          # <t>\coref{6}{CO8-8}</t>

    # b''кb''b''ob''b''нb''b''тb''b''eb''b''йb''b''нb''b''eb''b''pb''
    w = QtWidgets.QWidget()                                     # <t>\coref{7}{CO8-9}</t>
    w.setContentsMargins(0,0,0,6)
    w.setMinimumHeight(40)

    # b''мb''b''ab''b''кb''b''eb''b''тb''
    hbox = QtWidgets.QHBoxLayout()                              # <t>\coref{8}{CO8-10}</t>
```

```
    hbox.addWidget(self.w.rpm_bar)                              # <t>\coref{9}{CO8-11}</t>
    hbox.addWidget(self.w.led)                                  # <t>\coref{10}{CO8-12}</t>
    w.setLayout(hbox)

    # b''дb''b''оb''b''дb''b''аb''b''тb''b''иb'' b''кb''b''оb''b''нb''b''тb''b''еb''b''йb'' ↩
        b''нb''b''еb''b''рb'' b''уb'' b''кb''b''уb''b''тb'' b''вb''b''іb''b''дb''b''жb''b' ↩
        'еb''b''тb''b''аb'' b''пb''b''рb''b''аb''b''вb''b''оb''b''їb'' b''вb''b''кb''b''лb'' ↩
        b''аb''b''дb''b''кb''b''иb''
    self.w.rightTab.setCornerWidget(w)                          # <t>\coref{11}{CO8-13}</t>
```

❶, ❶ Це ініціалізує базовий віджет StateLed та використовує `self.w.led` як посилання з цього моменту.

❷, ❷ Оскільки світлодіод стану може використовуватися для багатьох індикацій, ми повинні встановити властивість, яка позначає його як світлодіод, що вказує на швидкість шпинделя.

❸ Це робить його зеленим, коли він увімкнений.

❹ Це додатковий виклик функції, необхідний для деяких віджетів QtVCP.
Якщо `HAL_NAME` пропущено, буде використано `objectName` віджета, якщо такий є.
Це надає спеціальним віджетам посилання на:

`self.HAL_GCOMP`
екземпляр компонента *HAL*

`self.HAL_NAME`
Назва цього *віджета* у вигляді рядка

`self.QT_OBJECT_`
Екземпляр об'єкта PyQt цього *віджета*

`self.QTVCP_INSTANCE_`
Найвищий рівень батьківського елемента екрана

`self.PATHS_`
Екземпляр бібліотеки path у QtVCP

`self.PREFS_`
екземпляр додаткового файлу налаштувань

`self.SETTINGS_`
об'єкт `Qsettings`

❺ Ініціалізує PyQt5 `QProgressBar`.

❻ Встановлює максимальний діапазон індикатора прогресу на значення, вказане в `INI`.

❼ Ми створюємо QWidget
Оскільки ви можете додати лише один віджет у кут вкладки, а нам потрібно два, ми повинні додати обидва в **контейнер**.

❽ додати QHBoxLayout до QWidget.

❾, ❿ Потім ми додаємо наш індикатор QProgress та світлодіод до макета.

⓫ Нарешті, ми додаємо QWidget (з нашою індикатором QProgress та світлодіодом у ньому) до кута віджета вкладки.

Створіть функцію моніторингу STATUS Тепер ми створюємо функцію, яка фактично оновлюватиме `QProgressBar`, коли `STATUS` оновлює швидкість шпинделя:

```
########################
# b''зb''b''вb''b''оb''b''рb''b''оb''b''тb''b''нb''b''іb'' b''вb''b''иb''b''кb''b''лb''b' ↩
    'иb''b''кb''b''иb'' b''вb''b''іb''b''дb'' STATUS #
########################
def update_spindle(self, data):
    self.w.rpm_bar.setInvertedAppearance(bool(data<0))       # <1t>\coref{1}{CO9-1}</t>
    self.w.rpm_bar.setFormat('{0:d} RPM'.format(int(data)))  # <1t>\coref{2}{CO9-2}</t>
    self.w.rpm_bar.setValue(abs(data))                       # <1t>\coref{3}{CO9-3}</t>
```

❶ У цьому випадку ми вибрали відображення зліва направо або справа наліво, залежно від того, чи повертаємося ми за годинниковою стрілкою чи проти неї.

❷ Це форматує текст у смузі.

❸ Це встановлює довжину кольорової смуги.

12.11.13 Періодично оновлювати/читати об’єкти

Іноді потрібно **оновлювати віджет або регулярно зчитувати значення**, яке не охоплюється звичайними бібліотеками.

Тут ми оновлюємо світлодіод на основі спостережуваного виводу HAL кожні 100 мс.

Ми припускаємо, що у файлі інтерфейсу Qt Designer є світлодіод з назвою `led`.

Завантажте бібліотеку Qhal для доступу до компонента HAL QtVCP У розділі «IMPORT SECTION» додайте:

```
from qtvcp.core import Qhal
```

Створення екземпляра Qhal У розділі «INSTANTIATE LIBRARY» додайте:

```
QHAL = Qhal()
```

Тепер додайте/змініть ці розділи, щоб включити код, подібний до цього:

Зареєструвати функцію, яка буде викликатися в період CYCLE_TIME Зазвичай це кожен 100 мс.

```
########################
# **** b''Ib''b''Hb''b''Ib''b''Цb''b''Ib''b''Ab''b''Лb''b''Ib''b''Зb''b''Ab''b''Цb''b''Ib'' ↩
    b''Яb'' **** #
########################
# widgets b''дb''b''оb''b''зb''b''вb''b''оb''b''лb''b''яb''b''єb'' b''дb''b''оb''b''сb''b' ↩
    'тb''b''уb''b''пb'' b''дb''b''оb'' b''вb''b''іb''b''дb''b''жb''b''еb''b''тb''b''іb''b' ↩
    'вb'' b''зb'' b''фb''b''аb''b''йb''b''лb''b''іb''b''вb'' QtVCP
# b''нb''b''аb'' b''цb''b''ьb''b''оb''b''мb''b''уb'' b''еb''b''тb''b''аb''b''пb''b''іb'' b' ↩
    'вb''b''іb''b''дb''b''жb''b''еb''b''тb''b''иb'' b''тb''b''аb'' b''пb''b''іb''b''нb''b' ↩
    'иb'' hal b''нb''b''еb'' b''сb''b''тb''b''вb''b''оb''b''рb''b''юb''b''юb''b''тb''b''ьb'' ↩
    b''сb''b''яb''d
def __init__(self,halcomp,widgets,paths):
    self.hal = halcomp
    self.w = widgets
    self.PATHS = paths

    # b''зb''b''аb''b''рb''b''еb''b''єb''b''сb''b''тb''b''рb''b''уb''b''вb''b''аb''b''тb''b ↩
        ''иb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''юb'', b''яb''b''кb''b''аb'' b' ↩
        'бb''b''уb''b''дb''b''еb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''тb''b ↩
        ''иb''b''сb''b''яb'' b''чb''b''еb''b''рb''b''еb''b''зb'' b''пb''b''еb''b''рb''b' ↩
        'іb''b''оb''b''дb'' CYCLE_TIME (b''зb''b''аb''b''зb''b''вb''b''иb''b''чb''b''аb''b' ↩
        'йb'' b''кb''b''оb''b''жb''b''нb''b''іb'' 100 b''мb''b''сb'')
    STATUS.connect('periodic', lambda w: self.update_periodic())
```

Створіть власну функцію, яка буде викликатися періодично

```
####################
# b''зb''b''ab''b''гb''b''ab''b''лb''b''ьb''b''нb''b''іb'' b''фb''b''уb''b''нb''b''кb''b' ↩
    'цb''b''іb''b''їb'' #
####################
def update_periodic(self):
    data = QHAL.getvalue('spindle.0.is-oriented')
    self.w.led.setState(data)
```

12.11.14 Зовнішнє керування за допомогою ZMQ

QtVCP може автоматично налаштувати **обмін повідомленнями ZMQ** *для надсилання та/або отримання віддалених повідомлень від зовнішніх програм.*

Він використовує шаблон обміну повідомленнями **публікація/підписка** від ZMQ.

Як завжди, подумайте про **безпеку**, перш ніж дозволяти програмам взаємодіяти через повідомлення.

12.11.14.1 Читання повідомлень ZMQ

Іноді хочеться **керувати екраном за допомогою окремої програми**.

Увімкнути отримання повідомлень ZMQ У віджеті `ScreenOptions` ви можете вибрати властивість **`use_receive_zmq_option`**.
Ви також можете встановити цю властивість безпосередньо *у файлі обробника*, як у цьому прикладі.

Ми припускаємо, що віджет `ScreenOptions` називається `screen_options` у Qt Designer:

```
########################
# **** b''Ib''b''Hb''b''Ib''b''Цb''b''Ib''b''Ab''b''Лb''b''Ib''b''Зb''b''Ab''b''Цb''b''Ib'' ↩
   b''Яb'' **** #
########################
# widgets b''дb''b''ob''b''зb''b''вb''b''ob''b''лb''b''яb''b''єb'' b''дb''b''ob''b''cb''b' ↩
    'тb''b''уb''b''пb'' b''дb''b''ob'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''іb''b' ↩
    'вb'' b''зb'' b''фb''b''ab''b''йb''b''лb''b''іb''b''вb'' QtVCP
# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''іb'' b' ↩
    'вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''иb'' b''тb''b''ab'' b''пb''b''іb''b''нb''b' ↩
    'иb'' hal b''нb''b''eb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b''нb''b''іb''
def __init__(self,halcomp,widgets,paths):
    # b''бb''b''eb''b''зb''b''пb''b''ob''b''cb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b ↩
        ''ob'' b''вb''b''иb''b''бb''b''pb''b''ab''b''тb''b''иb'' b''ob''b''тb''b''pb''b' ↩
        'иb''b''мb''b''ab''b''нb''b''нb''b''яb'' b''пb''b''ob''b''вb''b''іb''b''дb''b''ob''b ↩
        ''мb''b''лb''b''eb''b''нb''b''ьb'' ZMQ
    self.w.screen_options.setProperty('use_receive_zmq_option',True)
```

Це **дозволяє зовнішній програмі викликати функції у файлі обробника**.

Додано функцію, яка буде викликатися під час отримання повідомлення ZMQ Давайте додамо спеціальну функцію для тестування. Вам потрібно буде запустити LinuxCNC з терміналу, щоб побачити надрукований текст.

```
####################
# b''зb''b''ab''b''гb''b''ab''b''лb''b''ьb''b''нb''b''іb'' b''фb''b''уb''b''нb''b''кb''b' ↩
    'цb''b''іb''b''їb'' #
####################
def test_zmq_function(self, arg1, arg2):
    print('zmq_test_function called: ', arg1, arg2)
```

Створіть зовнішню програму, яка надсилатиме ZMQ-повідомлення, що ініціюватимуть виклик функцій Ось приклад зовнішньої програми для виклику функції. Вона щосекунди чергує два набори даних. Запустіть її в окремому терміналі від LinuxCNC, щоб переглянути надіслані повідомлення.

```
#!/usr/bin/env python3
b''зb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' b''чb''b''ab''b''cb''b''yb'' b' ↩
    'cb''b''нb''b''yb''

import zmq
import json

context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://127.0.0.1:5690")
topic = b'QtVCP'

# b''пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob'' b''cb''b''тb''b' ↩
    'вb''b''ob''b''pb''b''eb''b''нb''b''eb'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b' ↩
    'мb''b''лb''b''eb''b''нb''b''нb''b''яb'' 1
# b''cb''b''тb''b''вb''b''ob''b''pb''b''юb''b''єb'' b''cb''b''лb''b''ob''b''вb''b''нb''b' ↩
    'иb''b''кb'' b''фb''b''yb''b''нb''b''кb''b''цb''b''ib''b''їb'' b''дb''b''лb''b''яb'' b' ↩
    'вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''yb'' b''пb''b''лb''b''юb''b''cb'' b''бb''b' ↩
    'yb''b''дb''b''ьb''-b''яb''b''кb''b''ib'' b''ab''b''pb''b''гb''b''yb''b''мb''b''eb''b' ↩
    'нb''b''тb''b''иb''
x = {                               # <t>\coref{1}{CO10-1}</t>
  "FUNCTION": "test_zmq_function",
  "ARGS": [True,200]
}
# b''кb''b''ob''b''нb''b''вb''b''eb''b''pb''b''тb''b''yb''b''вb''b''ab''b''тb''b''иb'' b' ↩
    'вb'' b''ob''b''бb'''b''єb''b''кb''b''тb'' JSON
m1 = json.dumps(x)

# b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b' ↩
    'пb''b''ob''b''пb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob''b''їb'' b''зb''b' ↩
    'бb''b''ib''b''pb''b''кb''b''иb'' 2
x = {                               # <t>\coref{2}{CO10-2}</t>
  "FUNCTION": "test_zmq_function",
  "ARGS": [False,0],
}
# b''кb''b''ob''b''нb''b''вb''b''eb''b''pb''b''тb''b''yb''b''вb''b''ab''b''тb''b''иb'' b' ↩
    'вb'' b''ob''b''бb'''b''єb''b''кb''b''тb'' JSON
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 Встановіть **функцію для виклику** та **аргументи для надсилання** цій функції.

Вам потрібно знати *сигнатуру* функції, яку ви хочете викликати. Також зверніть увагу, що *повідомлення перетворюється на об'єкт JSON*. Це пов'язано з тим, що ZMQ надсилає байтові повідомлення, а не об'єкти Python. `json` перетворює об'єкти Python на байти, які будуть перетворені назад після отримання.

12.11.14.2 Написання повідомлень ZMQ

Ви також можете **спілкуватися із зовнішньою програмою з екрана**.

У віджеті `ScreenOptions` ви можете вибрати властивість **`use_send_zmq_message`**. Ви також можете встановити цю властивість безпосередньо *у файлі обробника*, як у цьому прикладі.

Ми припускаємо, що віджет `ScreenOptions` називається `screen_options` у Qt Designer:

Увімкнути надсилання повідомлень ZMQ

```
########################
# **** b''Ib''b''Hb''b''Ib''b''Цb''b''Ib''b''Ab''b''Лb''b''Ib''b''Зb''b''Ab''b''Цb''b''Ib'' ↩
    b''Яb'' **** #
########################
# «widgets» b''дb''b''ob''b''зb''b''вb''b''ob''b''лb''b''яb''b''єb'' b''дb''b''ob''b''cb''b ↩
    ''тb''b''уb''b''пb'' b''дb''b''ob'' b''вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''іb''b' ↩
    'вb'' b''зb'' b''фb''b''аb''b''йb''b''лb''b''іb''b''вb'' QtVCP
# b''нb''b''аb'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''аb''b''пb''b''іb'' b' ↩
    'вb''b''іb''b''дb''b''жb''b''eb''b''тb''b''иb'' b''тb''b''аb'' b''пb''b''іb''b''нb''b' ↩
    'иb'' hal b''щb''b''eb'' b''нb''b''eb'' b''cb''b''тb''b''вb''b''ob''b''pb''b''eb''b' ↩
    'нb''b''іb''
def __init__(self, halcomp,widgets,paths):
    # b''бb''b''eb''b''зb''b''пb''b''ob''b''cb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b ↩
        ''ob'' b''вb''b''иb''b''бb''b''pb''b''аb''b''тb''b''иb'' b''нb''b''аb''b''дb''b' ↩
        'cb''b''иb''b''лb''b''аb''b''нb''b''нb''b''яb'' b''пb''b''ob''b''вb''b''іb''b''дb''b ↩
        ''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' ZMQ
    self.w.screen_options.setProperty('use_send_zmq_option',True)
```

Це дозволяє надсилати повідомлення до окремої програми.
Надіслане повідомлення залежатиме від того, чого очікує зовнішня програма.

Створіть функцію для надсилання ZMQ-повідомлень Додамо спеціальну функцію для тестування
Щоб побачити надрукований текст, вам потрібно буде запустити LinuxCNC з терміналу.
Також потрібно додати щось для виклику цієї функції, наприклад, клацання кнопки.

```
####################
# b''зb''b''аb''b''гb''b''аb''b''лb''b''ьb''b''нb''b''іb'' b''фb''b''уb''b''нb''b''кb''b' ↩
    'цb''b''іb''b''їb'' #
####################
def send_zmq_message(self):
    # b''Цb''b''eb'' b''мb''b''ob''b''жb''b''eb'' b''бb''b''уb''b''тb''b''иb'' b''бb''b' ↩
        'уb''b''дb''b''ьb''-b''яb''b''кb''b''иb''b''йb'' b''ob''b''бb''b''єb''b''кb''b' ↩
        'тb'' Python, b''яb''b''кb''b''иb''b''йb'' JSON b''мb''b''ob''b''жb''b''eb'' b''кb'' ↩
       b''ob''b''нb''b''вb''b''eb''b''pb''b''тb''b''уb''b''вb''b''аb''b''тb''b''иb''
    message = {”name”: ”John”, ”age”: 30}
    self.w.screen_options.send_zmq_message(message)
```

Використайте або створіть програму, яка отримуватиме повідомлення ZMQ Ось приклад програми, яка отримає повідомлення та виведе його на термінал:

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# b''Bb''b''иb''b''зb''b''нb''b''аb''b''чb''b''тb''b''eb'' b''cb''b''ob''b''кb''b''eb''b' ↩
    'тb'' b''зb''b''аb'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' ”b' ↩
    'Kb''b''ob''b''нb''b''тb''b''eb''b''кb''b''cb''b''тb''b''уb''”
sock = context.socket(zmq.SUB)
```

```
# b''Bb''b''ив''b''зb''b''нb''b''аb''b''чb''b''тb''b''eb'' b''пb''b''іb''b''дb''b''пb''b' ↩
    'ив''b''сb''b''кb''b''уb'' b''тb''b''аb'' b''пb''b''оb''b''вb''b''іb''b''дb''b''оb''b' ↩
    'мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''зb'' b''тb''b''eb''b''мb''b''оb''b''юb'' b' ↩
    'дb''b''лb''b''яb'' b''пb''b''рb''b''иb''b''йb''b''нb''b''яb''b''тb''b''тb''b''яb''.
topic = "" # b''вb''b''сb''b''іb'' b''тb''b''eb''b''мb''b''иb''
sock.setsockopt_string(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

b''пb''b''оb''b''кb''b''иb'' True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

12.11.15 Надсилання повідомлень у рядок стану або діалогові вікна сповіщен на робочому столі

Існує кілька способів **повідомити інформацію користувачеві**.

Рядок стану використовується для *короткої інформації,* яка відображається користувачеві.

Note
Не всі екрани мають рядок стану.

Приклад використання рядка стану

```
self.w.statusbar.showMessage(message, timeout * 1000)
```

`timeout` вказується в секундах, і ми припускаємо, що `statusbar` — це ім'я віджета в наборі Qt Designer.

Ви також можете використовувати бібліотеку `Status` для надсилання повідомлення до бібліотеки `notify`, якщо вона увімкнена (зазвичай це встановлюється у віджеті `ScreenOptions`): Це надішле повідомлення до панелі стану та **діалогового вікна сповіщень на робочому столі**.

Повідомлення також записуються, доки користувач не видалить їх за допомогою елементів керування. Користувачі можуть відновити будь-які записані повідомлення.

Є кілька варіантів:

`STATUS.TEMPORARY_MESSAGE`
: Показувати повідомлення лише короткий час.

`STATUS.OPERATOR_ERROR`, `STATUS.OPERATOR_TEXT`, `STATUS.NML_ERROR`, `STATUS.NML_TEXT`

Приклад надсилання повідомлення оператору:

```
STATUS.emit('error', STATUS.OPERATOR_ERROR, 'message')
```

Ви можете надсилати повідомлення через функції операторських повідомлень LinuxCNC. Зазвичай вони перехоплюються системою сповіщень, тому еквівалентні вищезазначеному. Вони також будуть виведені на термінал.

```
ACTION.SET_DISPLAY_MESSAGE('MESSAGE')
ACTION.SET_ERROR_MESSAGE('MESSAGE')
```

12.11.16 Зміни фокусу

Фокус використовується для **спрямування дій користувача**, таких як введення з клавіатури, до відповідного віджета.

Отримати віджет, що відображає поточний фокус

```
fwidget = QtWidgets.QApplication.focusWidget()
if fwidget is not None:
    print("focus widget class: {} name: {} ".format(fwidget, fwidget.objectName()))
```

Отримувати віджет з фокусом, коли фокус змінюється

```
# b''нb''b''ab'' b''цb''b''ьb''b''ob''b''мb''b''уb'' b''eb''b''тb''b''ab''b''пb''b''ib'':
# b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b''рb''b''иb'' b''вb''b''ib''b' ↩
    'дb''b''жb''b''eb''b''тb''b''ib''b''вb'' b''cb''b''тb''b''вb''b''ob''b''рb''b''eb''b' ↩
    'нb''b''ob''.
# b''пb''b''ib''b''нb''b''иb'' HAL b''зb''b''ib''b''бb''b''рb''b''ab''b''нb''b''ob'', b' ↩
    'ab''b''лb''b''eb'' HAL b''щb''b''eb'' b''нb''b''eb'' b''гb''b''ob''b''тb''b''ob''b' ↩
    'вb''b''иb''b''йb''
def initialized__(self):
    QtWidgets.QApplication.instance().event_filter.focusIn.connect(self.focusInChanged)

#####################
# b''зb''b''ab''b''гb''b''ab''b''лb''b''ьb''b''нb''b''ib'' b''фb''b''уb''b''нb''b''кb''b' ↩
    'цb''b''ib''b''їb'' #
#####################

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')
```

Зверніть увагу, що ми іноді порівнюємо з `widget`, іноді з `widget.parent()`.

Це тому, що *деякі віджети QtVCP побудовані з кількох* ***підвіджетів***, і останні фактично отримують фокус; тому нам потрібно **перевірити батьківський** елемент цих підвіджетів.

В інших випадках фокус отримує головний віджет, наприклад, віджет відображення G-коду можна налаштувати так, щоб він приймав фокус. У цьому випадку в ньому немає підвіджетів, тому порівняння з `widget.parent()` дасть вам контейнер, що містить віджет G-коду.

12.11.17 Параметри часу завантаження командного рядка читання

Деякі панелі потребують інформації під час завантаження для налаштування/опцій. QtVCP задовольняє цю вимогу за допомогою опцій «-o».
Аргумент «-o» підходить для декількох відносно коротких опцій, які можна додати до командного рядка завантаження.
Для більш детальної інформації, ймовірно, краще прочитати файл INI або файл налаштувань.

У командному рядку можна використовувати кілька опцій «-o», тому їх необхідно декодувати. «self.w.USEROPTIONS_» зберігатиме всі знайдені опції «-o» у вигляді списку рядків. Ви повинні проаналізувати та визначити, що приймається і що з цим робити.

Приклад коду для отримання опцій -o для номера камери та розміру вікна

```
    def initialized__(self):

        # b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''нb''b' ↩
            'ob''b''мb''b''eb''b''pb'' b''кb''b''ab''b''мb''b''eb''b''pb''b''иb'' b''зb''b' ↩
            'ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''yb''b''вb''b''ab''b''нb''b' ↩
            'нb''b''яb''b''мb''
        number = 0

        # b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b''иb''b''тb''b''иb'', b''чb''b' ↩
            'иb'' b''єb'' b''вb''b''зb''b''ab''b''гb''b''ab''b''лb''b''ib'' b''яb''b''кb''b' ↩
            'ib''b''cb''b''ьb'' b''ob''b''пb''b''цb''b''ib''b''їb'' -o
        if self.w.USEROPTIONS_ is not None:

            # b''яb''b''кb''b''щb''b''ob'' b''вb'' b''pb''b''eb''b''жb''b''иb''b''мb''b' ↩
                'ib'' b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb'' ↩
                b''нb''b''яb'' b''вb''b''иb''b''вb''b''eb''b''cb''b''тb''b''иb'' b''пb''b' ↩
                'ab''b''pb''b''ab''b''мb''b''eb''b''тb''b''pb''b''иb'' b''вb'' b''тb''b' ↩
                'eb''b''pb''b''мb''b''ib''b''нb''b''ab''b''лb''
            LOG.debug('cam_align user options: {}'.format(self.w.USEROPTIONS_))

            # go b''чb''b''eb''b''pb''b''eb''b''зb'' b''зb''b''нb''b''ab''b''йb''b''дb''b' ↩
                'eb''b''нb''b''ib'' b''вb''b''ab''b''pb''b''ib''b''ab''b''нb''b''тb''b''иb'' ↩
                 b''ob''b''дb''b''иb''b''нb'' b''зb''b''ab'' b''ob''b''дb''b''нb''b''иb''b' ↩
                'мb''
            for num, i in enumerate(self.w.USEROPTIONS_):

                # b''яb''b''кb''b''щb''b''ob'' b''пb''b''ab''b''pb''b''ab''b''мb''b''eb''b' ↩
                    'тb''b''pb'' -o b''мb''b''ib''b''cb''b''тb''b''иb''b''тb''b''ьb'' 'size ↩
                    =', b''пb''b''pb''b''иb''b''пb''b''yb''b''cb''b''кb''b''ab''b''тb''b' ↩
                    'иb'', b''щb''b''ob'' b''цb''b''eb'' b''шb''b''иb''b''pb''b''иb''b''нb'' ↩
                    b''ab'' b''тb''b''ab'' b''вb''b''иb''b''cb''b''ob''b''тb''b''ab'' b' ↩
                    'вb''b''ib''b''кb''b''нb''b''ab''
                # b''зb''b''мb''b''ib''b''нb''b''иb''b''тb''b''иb'' b''шb''b''иb''b''pb''b' ↩
                    'иb''b''нb''b''yb'' b''тb''b''ab'' b''вb''b''иb''b''cb''b''ob''b''тb''b' ↩
                    'yb'' b''вb''b''ib''b''кb''b''нb''b''ab'' b''зb''b''ab'' b''зb''b''ab''b ↩
                    ''мb''b''ob''b''вb''b''чb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb''b' ↩
                    'мb''
                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 ↩
                            ])

                #  # b''яb''b''кb''b''щb''b''ob'' b''пb''b''ab''b''pb''b''ab''b''мb''b' ↩
                    'eb''b''тb''b''pb'' -o b''мb''b''ib''b''cb''b''тb''b''иb''b''тb''b''ьb'' ↩
                     'camnumber=', b''пb''b''pb''b''иb''b''пb''b''yb''b''cb''b''кb''b''ab''b ↩
                    ''тb''b''иb'', b''щb''b''ob'' b''цb''b''eb'' b''нb''b''ob''b''мb''b' ↩
                    'eb''b''pb'' b''кb''b''ab''b''мb''b''eb''b''pb''b''иb'', b''яb''b''кb''b ↩
                    ''иb''b''йb'' b''пb''b''ob''b''тb''b''pb''b''ib''b''бb''b''нb''b''ob'' b ↩
                    ''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ob''b''вb''b' ↩
                    'yb''b''вb''b''ab''b''тb''b''иb''
                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')
```

```
        # b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''иb''b''тb''b''иb'' b''нb''b' ↩
            'ob''b''мb''b''eb''b''pb'' b''кb''b''ab''b''мb''b''eb''b''pb''b''иb'' b''ab''b' ↩
            'бb''b''ob'' b''зb''b''ab'' b''зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b' ↩
            'вb''b''ab''b''нb''b''нb''b''яb''b''мb'', b''ab''b''бb''b''ob'', b''яb''b''кb''b ↩
            ''щb''b''ob'' b''ob''b''пb''b''цb''b''ib''b''яb'' -o b''зb''b''мb''b''ib''b' ↩
            'нb''b''иb''b''лb''b''ab'' b''зb''b''мb''b''ib''b''нb''b''нb''b''уb'' 'number',  ↩
            b''нb''b''ab'' b''цb''b''eb''b''йb'' b''нb''b''ob''b''мb''b''eb''b''pb''.
        self.w.camview._camNum = number
```

12.11.18 G-код для зчитування налаштувань Qt

Ось як створити програму O-Word, щоб прочитати запис файлу переваг QtDragon і додавати його як параметр G-коду.
Зкликання цього O-слову оновлює параметр *toolToLoad*
Це використовує "горячий коментар Python" для спілкування з вбудованим прикладом python.
Перегляньте розділ переробки документів для опису.

```
(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 Розробка QtVCP

12.12.1 Огляд

Метою QtVCP є **надання інфраструктури для підтримки створення екранів та панелей VCP для LinuxCNC**.

Надаючи *різноманітний набір віджетів* та підтримуючи *власне кодування*, QtVCP сподівається, що енергія розробки буде витрачена на *один інструментарій*, а не на постійне перевинаходження.

Використовуючи один і той самий набір інструментів на багатьох екранах/панелях, користувачам має бути легше налаштовувати/створювати їх, а розробникам – легше допомагати у вирішенні проблем з меншими зусиллями.

QtVCP використовує **файл .ui, створений у Qt Designer**, та **файл обробника Python**

- **завантажувати та керувати екраном/панеллю, яка відображає віджети Qt** та
- **керувати контролером руху LinuxCNC або контактами HAL**.

Є *вбудовані екрани та панелі*, які користувач легко завантажує, або ж користувачі можуть створювати/змінювати свої власні.

QtVCP використовує **бібліотеки та користувацькі віджети**, щоб приховати деякі складнощі взаємодії з LinuxCNC. Використовуючи бібліотеку QtVCP замість LinuxCNC, ми можемо зменшити незначні зміни коду LinuxCNC.

12.12.2 Вбудовані місця розташування

Вбудовані екрани та панелі зберігаються в окремих папках:

- *Screens* в `share/qtvcp/screens`
- *Panels* в `share/qtvcp/panels`
- *Стокові зображення* у `share/qtvcp/images`

Екрани та панелі сортуються за назвою папки, яка також є назвою, що використовується для їх завантаження.

Усередині папки буде:

- the `.ui` *file*,
- файл *обробника*, та
- можливо, файл теми `.qss`.

12.12.3 Запуск QtVCP для вимкнення

Вихідний код QtVCP знаходиться в папці +src/emc/usr_intf/qtvcp+ дерева вихідних кодів LinuxCNC.

12.12.3.1 Запуск QtVCP

Під час першого запуску QtVCP:

1. Він повинен вирішити, чи є цей об’єкт екраном чи панеллю.
2. Він шукає та збирає інформацію про шляхи до потрібних файлів та корисних папок.
3. Тоді:
    a. Збирає компонент HAL,
    b. Завантажує екземпляр вікна,
    c. Додає розширення обробників,
    d. Встановлює фільтр подій.

Тепер вікна/віджети створено, піни HAL зібрано. Це також ініціює функцію +_init_hal()+ віджетів. . Викликається функція-обробник +initialized__()+ . Бібліотеку STATUS примусово оновлено. . На цьому етапі компонент HAL готовий. . Встановлюється різноманітні додаткові аргументи switch, включаючи виклик HAL-файлу POSTGUI (якщо це екран). . Сигнали завершення перехоплюються і QtVCP тепер опитує події.

12.12.3.2 Вимкнення QtVCP

Зрештою, коли QtVCP запитується про вимкнення:

1. Він викликає функції вимкнення у файлі обробника,
2. STATUS моніторинг вимкнено
3. Компонент HAL вимикається

12.12.4 Інформація про шлях

Коли QtVCP завантажується, він збирає інформацію про шляхи.

Це доступно у функції +__init__()+ файлу обробника як **path**:

IMAGEDIR
: Шлях вбудованих зображень

SCREENDIR
: Шлях вбудованих екранів контролера руху

PANELDIR
: Шлях вбудованих допоміжних панелей

WORKINGDIR
: Шлях, з якого було запущено QtVCP

CONFIGPATH
: Шлях запущеної конфігурації

BASEDIR
: Загальний шлях, що використовується для виведення всіх шляхів

BASENAME
: Загальна назва, що використовується для виведення всіх шляхів

LIBDIR
: Шлях до бібліотеки Python QtVCP

HANDLER
: Шлях до файлу обробника

XML
: Шлях до файлу .ui

DOMAIN
: Шлях перекладу

IS_SCREEN
: Перемикач екрана/панелі

12.12.5 Ідіосинкразії

Вони намагаються охопити неочевидні ситуації.

12.12.5.1 Збір кодів помилок

Коди помилок LinuxCNC можна зчитувати лише з одного місця.

Під час читання воно ***споживається***, тобто *жоден інший об'єкт не може його прочитати*.

На екранах QtVCP рекомендується *використовувати віджет* `ScreenOptions` *для налаштування читання помилок*.

Потім помилки ***надсилаються іншим об'єктам*** *через* **`STATUS`** *сигнали*.

12.12.5.2 Швидкість поштовху

LinuxCNC не має внутрішнього запису швидкості поштовхового переміщення: *ви повинні вказати його під час поштовхового переміщення*.

QtVCP використовує бібліотеку `STATUS` для *відстеження останніх лінійних та кутових швидкостей поштовхового переміщення*.

Він **завжди вказується в одиницях машини за хвилину** і *повинен бути перетворений, якщо ви перебуваєте в режимі не машинні одиниці*.
Отже, якщо ваша машина працює в імперській системі, але ви перебуваєте в метричному режимі, зміни швидкості поштовху, що надсилаються до функцій `ACTION`, повинні бути перетворені в імперські одиниці.
Аналогічно, якщо машина працює в метричній системі, а ви перебуваєте в імперському режимі, зміни швидкості переміщення повинні надсилатися до функцій `ACTION` у метричних одиницях.
Для кутових швидкостей переміщення одиниці виміру не змінюються в метричному/імперському режимі, тому ви можете надсилати їх до функцій `ACTION` без перетворення.

Хоча ви можете ігнорувати цей запис про швидкість прокрутки під час створення екранів, будь-хто, хто модифікує ваш екран і використовує вбудовані віджети швидкості прокрутки, не отримає бажаних результатів, оскільки функція **`DO_JOG`** бібліотеки `ACTION` отримує швидкість прокрутки з бібліотеки `STATUS`.

12.12.5.3 Прив'язка клавіш



Warning
Прив'язка клавіш завжди є справою, яку *важко-підібрати-правильно-у-всіх-випадках*.

Функції користувацького призначення клавіш мають бути *визначені у файлі обробника*.

Найголовніше, що віджети, які потребують регулярного введення клавіш, а не пробіжок, слід перевіряти у функції `processed_key_event__`.

12.12.5.4 Файл налаштувань

Деякі віджети QtVCP використовують файл налаштувань для запису важливої інформації.

Це *вимагає налаштування файлу налаштувань на ранній стадії* процесу ініціалізації віджета.
Найпростіший спосіб зробити це – **використати віджет `ScreenOptions`**.

12.12.5.5 Функції спеціального налаштування віджета

QtVCP шукає та викликає функцію +_hal_init()+ *коли віджет вперше завантажується*.

Він не викликається під час використання редактора Qt Designer.

Після виклику цієї функції віджет має доступ до деяких спеціальних змінних:

self.HAL_GCOMP
: Екземпляр компонента *HAL*

self.HAL_NAME
: Назва цього *віджета* у вигляді рядка

self.QT_OBJECT_
: Екземпляр об’єкта PyQt цього *віджета*

self.QTVCP_INSTANCE_
: *Найвищий рівень батьківського* елемента екрана

self.PATHS_
: Екземпляр бібліотеки path у QtVCP

self.PREFS_
: Екземпляр *додаткового файлу налаштувань*

self.SETTINGS_
: Об’єкт `Qsettings`

Під час створення власного віджета, _import та підклас _the **+_HalWidgetBase+** для цієї поведінки.

12.12.5.6 Діалоги

Діалогові вікна (також відомі як «спливаючі вікна») *найкраще завантажувати за допомогою* віджета `ScreenOptions`, але їх можна розмістити на екрані в Qt Designer.

Неважливо, де саме на макеті вони розташовані, але *щоб зробити їх прихованими*, потрібно циклічно змінювати властивість `state` на true, а потім на false.

За замовчуванням, якщо є файл налаштувань, діалогові вікна запам’ятають свій останній розмір/розта
Це можна змінити, щоб вони щоразу відкривалися в одному й тому ж місці.

12.12.5.7 Стилі (теми)

Хоча стилі можна встановлювати в Qt Designer, зручніше змінювати їх пізніше, якщо вони всі встановлені в окремому файлі .qss. Файл слід розмістити в тому ж місці, що й файл обробника.

Chapter 13

Програмування інтерфейсу користува

13.1 Панель

13.1.1 Вступ

Panelui — це компонент, що не працює в режимі реального часу, для інтерфейсу кнопок до LinuxCNC або HAL:

- Він декодує коди сканування ключів у стилі MESA 7I73 та викликає відповідну процедуру.
- Він отримує вхідні дані від компонента реального часу — семплера. Семплер отримує вхідні дані або від компонента MESA 7I73, або від компонента sim_matrix_kb.
- Panelui можна налаштувати за допомогою текстового файлу у стилі INI для визначення типів кнопок, типів контактів HAL та/або команд.
- Його можна розширити за допомогою файлу-обробника на основі Python для додавання функцій.

Хоча фактичні кнопки введення повинні бути миттєвими, Panelui використовуватиме цей вхід для створення виводу перемикання, радіо або миттєвої кнопки.

13.1.2 Команди завантаження

Команда, що використовується для завантаження panelui (з необов’язковим параметром debug -d):

```
loadusr -W panelui -d
```

Це ініціалізує panelui, який шукатиме INI-файл panelui.ini в папці config або папці користувача.

Перевірити INI-файл можна за допомогою цієї команди:

```
loadusr pyui
```

Ця програма зчитає, спробує виправити, а потім збереже файл panelui.ini. Якщо будуть знайдені помилки, програма виведе їх у термінал.

У типовому HAL-файлі додаються такі команди:

```
# b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''иb'', b''нb''b''eb''b''ob''b''бb''b''xb''b' ↩
    'ib''b''дb''b''нb''b''ib'' b''дb''b''лb''b''яb'' b''зb''b''ab''b''вb''b''ab''b''нb''b' ↩
    'тb''b''ab''b''жb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ab''b''нb''b''eb''b''лb''b' ↩
    'ib''
#
# b''cb''b''eb''b''мb''b''пb''b''лb''b''eb''b''pb'' b''пb''b''ob''b''тb''b''pb''b''ib''b' ↩
    'бb''b''eb''b''нb'' b''дb''b''лb''b''яb'' panelui
# cfg= b''зb''b''ab''b''вb''b''жb''b''дb''b''иb'' b''мb''b''ab''b''єb'' b''бb''b''yb''b' ↩
    'тb''b''иb'' u b''дb''b''лb''b''яb'' panelui. depth b''вb''b''cb''b''тb''b''ab''b''нb''b ↩
    ''ob''b''вb''b''лb''b''юb''b''єb'' b''дb''b''ob''b''cb''b''тb''b''yb''b''пb''b''нb''b' ↩
    'иb''b''йb'' b''бb''b''yb''b''фb''b''eb''b''pb''
loadrt sampler cfg=u depth=1025

#b''pb''b''ob''b''зb''b''кb''b''ob''b''мb''b''eb''b''нb''b''тb''b''yb''b''вb''b''ab''b' ↩
    'тb''b''иb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b''вb''b''ib''b''pb''b' ↩
    'кb''b''иb'' INI-b''фb''b''ab''b''йb''b''лb''b''yb'' panelui
#loadusr pyui

# -d = b''нb''b''ab''b''лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b' ↩
    'яb'', -v = b''дb''b''eb''b''тb''b''ab''b''лb''b''ьb''b''нb''b''eb'' b''нb''b''ab''b' ↩
    'лb''b''ab''b''гb''b''ob''b''дb''b''жb''b''eb''b''нb''b''нb''b''яb''
# -d b''пb''b''ob''b''кb''b''ab''b''жb''b''eb'' b''вb''b''ab''b''мb'' b''ib''b''дb''b''eb'' ↩
   b''нb''b''тb''b''иb''b''фb''b''ib''b''кb''b''ab''b''цb''b''ib''b''юb'' b''нb''b''ab''b' ↩
    'тb''b''иb''b''cb''b''кb''b''ab''b''нb''b''нb''b''яb'' b''кb''b''лb''b''ab''b''вb''b' ↩
    'ib''b''шb'' b''тb''b''ab'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''нb''b' ↩
    'ib'' b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''иb''
# -v b''пb''b''pb''b''иb''b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''ob'' b''дb''b''лb''b ↩
    ''яb'' b''ib''b''нb''b''фb''b''ob''b''pb''b''мb''b''ab''b''цb''b''ib''b''їb'' b''pb''b' ↩
    'ob''b''зb''b''pb''b''ob''b''бb''b''нb''b''иb''b''кb''b''ab''
loadusr -W panelui -d

# b''вb''b''иb''b''кb''b''ob''b''pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'' b' ↩
    'ib''b''мb''b''ib''b''тb''b''ab''b''цb''b''ib''b''їb'' b''кb''b''нb''b''ob''b''пb''b' ↩
    'ob''b''кb'' b''зb''b''ab''b''мb''b''ib''b''cb''b''тb''b''ьb'' b''пb''b''лb''b''ab''b' ↩
    'тb''b''иb'' MESA 7I73
# b''тb''b''ob''b''мb''b''yb'' b''мb''b''иb'' b''зb''b''ab''b''вb''b''ab''b''нb''b''тb''b' ↩
    'ab''b''жb''b''yb''b''єb''b''мb''b''ob'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb'' sim_matrix_kb b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b''eb''b' ↩
    'тb''b''вb''b''ob''b''pb''b''eb''b''нb''b''нb''b''яb'' b''кb''b''ob''b''нb''b''тb''b' ↩
    'ab''b''кb''b''тb''b''ib''b''вb'' HAL b''yb'' b''кb''b''ob''b''дb''b''иb'' b''cb''b' ↩
    'кb''b''ab''b''нb''b''yb''b''вb''b''ab''b''нb''b''нb''b''яb'' b''кb''b''лb''b''ab''b' ↩
    'вb''b''ib''b''шb''
loadrt sim_matrix_kb

# b''зb''b''єb''b''дb''b''нb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b' ↩
    'нb''b''eb''b''нb''b''тb''b''иb'' b''pb''b''ab''b''зb''b''ob''b''мb''.
# b''cb''b''eb''b''мb''b''пb''b''лb''b''eb''b''pb'' b''вb''b''зb''b''ab''b''єb''b''мb''b' ↩
    'ob''b''дb''b''ib''b''єb'' b''зb'' panelui b''вb''b''нb''b''yb''b''тb''b''pb''b''ib''b' ↩
    'шb''b''нb''b''ьb''b''ob''
net key-scan    sim-matrix-kb.0.out
net key-scan    sampler.0.pin.0

# b''дb''b''ob''b''дb''b''ab''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b''ob''b''нb''b' ↩
    'eb''b''нb''b''тb''b''иb'' panelui b''дb''b''ob'' b''пb''b''ob''b''тb''b''ob''b''кb''b' ↩
    'yb''

addf sim-matrix-kb.0    servo-thread
addf sampler.0          servo-thread
```

13.1.3 Посилання на файл panelui.ini

Ключові слова

- KEY= Використовується для позначення клавіші, на яку реагує кнопка. Це може бути NONE або номер ROW та номер стовпця, наприклад, R1C2. Рядок та стовпець можна використовувати лише один раз.
- ВИХІД= Це встановлює тип виводу кнопки, наприклад, S32, U32, FLOAT, BIT, NONE, COMMAND, ZMQ.
- ЗА ЗАМОВЧАННЯМ= Це встановлює початковий вихід групи або кнопки.
- GROUP= У радіокнопках позначає групу, з якою взаємодіє кнопка.
- GROUP_OUTPUT= встановлює вихідний сигнал групового контакту, якщо ця кнопка активна.
- STATUS_PIN= Якщо значення TRUE (ІСТИНА), буде додано HAL-пін, який відображає поточний стан кнопки.
- TRUE_STATE= встановлює вихідний сигнал на виводі HAL, якщо button має значення TRUE.
- FALSE_STATE= встановлює ВИХІД, який матиме вивід HAL, якщо кнопка має значення FALSE.
- TRUE_COMMAND= встановлює команду та аргументи, які будуть викликані, коли кнопка має значення TRUE.
- FALSE_COMMAND= встановлює команду та аргументи, які будуть викликані, коли кнопка має значення FALSE.
- TRUE_FUNCTION= встановлює функцію повідомлення ZMQ та аргументи, які будуть викликані, коли кнопка має значення TRUE.
- FALSE_FUNCTION= встановлює функцію повідомлення ZMQ та аргументи, які будуть викликані, коли кнопка має значення FALSE.

Префікс HAL

```
[HAL_PREFIX]
    NAME= Yourname
```

Це дозволяє змінити префікс виводів HAL з «panelui» на довільну назву.

Налаштування повідомлень ZMQ

```
[ZMQ_SETUP]
  TOPIC = 'QTVCP'
  SOCKET = 'tcp://127.0.0.1:5690'
  ENABLE = True
```

Це налаштовує та активує обмін повідомленнями на основі ZMQ. TOPIC та SOCKET повинні відповідати програмі прослуховування.

Радіокнопки Радіокнопки дозволяють активувати одночасно тільки одну кнопку в групі. Кожна група має власний вихідний контакт, окремий від кожної кнопки в групі. Визначення радіокнопок починаються з тексту «RADIO_BUTTON» в одинарних дужках.

```
[RADIO_BUTTONS]
  # b''Pb''b''ob''b''зb''b''дb''b''ib''b''лb''b''иb'' b''зb'' b''пb''b''ob''b''дb''b''вb''b ↩
      ''ib''b''йb''b''нb''b''иb''b''мb''b''иb'' b''дb''b''уb''b''жb''b''кb''b''аb''b''мb''b' ↩
      'иb'' b''вb''b''иb''b''зb''b''нb''b''аb''b''чb''b''аb''b''юb''b''тb''b''ьb'' b''гb''b' ↩
      'pb''b''уb''b''пb''b''иb'' b''пb''b''eb''b''pb''b''eb''b''мb''b''иb''b''кb''b''аb''b' ↩
      'чb''b''ib''b''вb''.
```

```
# b''Ib''b''мb'''b''яb'' b''гb''b''pb''b''yb''b''пb''b''иb'' b''мb''b''ab''b''єb'' b' ↩
    'бb''b''yb''b''тb''b''иb'' b''yb''b''нb''b''ib''b''кb''b''ab''b''лb''b''ьb''b''нb''b' ↩
    'иb''b''мb'' b''ib'' b''чb''b''yb''b''тb''b''лb''b''иb''b''вb''b''иb''b'мb'' b''дb''b ↩
    ''ob'' b''pb''b''eb''b''гb''b''ib''b''cb''b''тb''b''pb''b''yb''.
# b''Bb''b''иb''b''вb''b''ib''b''дb'' b''гb''b''pb''b''yb''b''пb'' b''кb''b''ob''b''нb''b ↩
    ''тb''b''pb''b''ob''b''лb''b''юb''b''єb''b''тb''b''ьb''b''cb''b''яb'' b''тb''b''иb''b' ↩
    'мb'', b''яb''b''кb''b''ab'' b''кb''b''нb''b''ob''b''пb''b''кb''b''ab'' b''ab''b''кb'' ↩
    b''тb''b''иb''b''вb''b''нb''b''ab'', b''ab'' b''нb''b''eb'' b''бb''b''eb''b''зb''b' ↩
    'пb''b''ob''b''cb''b''eb''b''pb''b''eb''b''дb''b''нb''b''ьb''b''ob'' b''кb''b''ob''b' ↩
    'дb''b''ob''b''мb'' b''кb''b''лb''b''ab''b''вb''b''ib''b''шb''b''ib''.
# DEFAULT b''пb''b''ob''b''cb''b''иb''b''лb''b''ab''b''єb''b''тb''b''ьb''b''cb''b''яb'' b ↩
    ''нb''b''ab'' b''кb''b''нb''b''ob''b''пb''b''кb''b''yb'' b''вb'' b''гb''b''pb''b''yb'' ↩
    b''пb''b''ib'' b''зb''b''ab'' b''ib''b''мb''b''eb''b''нb''b''eb''b''мb'' b''ib'' b' ↩
    'чb''b''yb''b''тb''b''лb''b''иb''b''вb''b''иb''b''йb'' b''дb''b''ob'' b''pb''b''eb''b' ↩
    'гb''b''ib''b''cb''b''тb''b''pb''b''yb''.
[[group1_name]]
  KEY = NONE
  OUTPUT = FLOAT
  DEFAULT = small
  # b''Pb''b''ob''b''зb''b''дb''b''ib''b''лb''b''иb'' b''зb'' b''пb''b''ob''b''тb''b' ↩
      'pb''b''ib''b''йb''b''нb''b''иb''b''мb''b''иb'' b''дb''b''yb''b''жb''b''кb''b''ab''b ↩
      ''мb''b''иb'' b''вb''b''иb''b''зb''b''нb''b''ab''b''чb''b''ab''b''юb''b''тb''b''ьb'' ↩
       b''кb''b''нb''b''ob''b''пb''b''кb''b''иb'' b''вb'' b''цb''b''ib''b''йb'' b''гb''b' ↩
      'pb''b''yb''b''пb''b''ib''.
  # b''Ib''b''мb''b''eb''b''нb''b''ab'' b''кb''b''нb''b''ob''b''пb''b''ob''b''кb'' b' ↩
      'пb''b''ob''b''вb''b''иb''b''нb''b''нb''b''ib'' b''бb''b''yb''b''тb''b''иb'' b''yb'' ↩
      b''нb''b''ib''b''кb''b''ab''b''лb''b''ьb''b''нb''b''иb''b''мb''b''иb'' b''ib'' b' ↩
      'чb''b''yb''b''тb''b''лb''b''иb''b''вb''b''иb''b''мb''b''иb'' b''дb''b''ob'' b''pb'' ↩
      b''eb''b''гb''b''ib''b''cb''b''тb''b''pb''b''yb''.
  # b''Уb'' b''гb''b''pb''b''yb''b''пb''b''ib'' b''пb''b''ob''b''вb''b''иb''b''нb''b' ↩
      'нb''b''ob'' b''бb''b''yb''b''тb''b''иb'' b''пb''b''pb''b''иb''b''нb''b''ab''b''йb'' ↩
      b''мb''b''нb''b''ib'' b''дb''b''вb''b''ib'' b''кb''b''нb''b''ob''b''пb''b''кb''b' ↩
      'иb''.
  #
  # b''Цb''b''яb'' b''кb''b''нb''b''ob''b''пb''b''кb''b''ab'', b''нb''b''ab''b''зb''b' ↩
      'вb''b''ab''b''нb''b''ab'' «small», b''кb''b''eb''b''pb''b''yb''b''єb''b''тb''b' ↩
      'ьb''b''cb''b''яb'' b''кb''b''лb''b''ab''b''вb''b''ib''b''шb''b''eb''b''юb'' b''pb'' ↩
      b''яb''b''дb''b''кb''b''ab'' 0 b''cb''b''тb''b''ob''b''вb''b''пb''b''цb''b''яb'' 1.
  # b''Пb''b''pb''b''иb'' b''нb''b''ab''b''тb''b''иb''b''cb''b''кb''b''ab''b''нb''b''нb'' ↩
      b''ib'' b''вb''b''ob''b''нb''b''ab'' b''пb''b''pb''b''иb''b''зb''b''вb''b''eb''b' ↩
      'дb''b''eb'' b''дb''b''ob'' b''вb''b''иb''b''вb''b''eb''b''дb''b''eb''b''нb''b''нb'' ↩
      b''яb'' b''гb''b''pb''b''yb''b''пb''b''иb'' .0001.
  # b''Bb''b''ob''b''нb''b''ab'' b''нb''b''eb'' b''мb''b''ab''b''єb'' b''вb''b''лb''b' ↩
      'ab''b''cb''b''нb''b''ob''b''гb''b''ob'' b''вb''b''иb''b''xb''b''ob''b''дb''b''yb'', ↩
       b''ab''b''лb''b''eb'' b''мb''b''ab''b''єb'' b''cb''b''тb''b''ab''b''нb''
  # b''кb''b''ob''b''нb''b''тb''b''ab''b''кb''b''тb'', b''яb''b''кb''b''иb''b''йb'' b' ↩
      'бb''b''yb''b''дb''b''eb'' b''cb''b''лb''b''ib''b''дb''b''yb''b''вb''b''ab''b''тb''b ↩
      ''иb'' b''зb''b''ab'' b''ïb''b''ïb'' b''пb''b''ob''b''тb''b''ob''b''чb''b''нb''b' ↩
      'иb''b''мb'' b''cb''b''тb''b''ab''b''нb''b''ob''b''мb''.
  # b''Ob''b''cb''b''кb''b''ib''b''лb''b''ьb''b''кb''b''иb'' b''цb''b''яb'' b''кb''b' ↩
      'нb''b''ob''b''пb''b''кb''b''ab'' b''зb''b''нb''b''ab''b''xb''b''ob''b''дb''b''иb''b ↩
      ''тb''b''ьb''b''cb''b''яb'' b''вb'' b''гb''b''pb''b''yb''b''пb''b''ib'', DEFAULT b' ↩
      'нb''b''eb'' b''мb''b''ab''b''єb'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb'' ↩
      b''яb''.
  # b''Ob''b''cb''b''кb''b''ib''b''лb''b''ьb''b''кb''b''иb'' OUTPUT b''нb''b''eb'' b' ↩
      'єb'' «COMMAND», b''зb''b''ab''b''пb''b''иb''b''cb''b''иb'' _COMMAND b''ib''b''гb''b ↩
      ''нb''b''ob''b''pb''b''yb''b''юb''b''тb''b''ьb''b''cb''b''яb''.
  [[[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
  # b''Цb''b''яb'' b''кb''b''нb''b''оb''b''пb''b''кb''b''аb'', b''нb''b''аb''b''зb''b' ↩
      'вb''b''аb''b''нb''b''аb'' «large», b''кb''b''еb''b''рb''b''уb''b''єb''b''тb''b' ↩
      'ьb''b''сb''b''яb'' b''кb''b''лb''b''аb''b''вb''b''іb''b''шb''b''еb''b''юb'' b''рb'' ↩
      b''яb''b''дb''b''кb''b''аb'' 0 b''сb''b''тb''b''оb''b''вb''b''пb''b''цb''b''яb'' 2.
  # b''Пb''b''рb''b''иb'' b''нb''b''аb''b''тb''b''иb''b''сb''b''кb''b''аb''b''нb''b''нb'' ↩
      b''іb'' b''вb''b''оb''b''нb''b''аb'' b''пb''b''рb''b''иb''b''зb''b''вb''b''еb''b' ↩
      'дb''b''еb'' b''дb''b''оb'' b''тb''b''оb''b''гb''b''оb'', b''щb''b''оb'' b''вb''b' ↩
      'иb''b''хb''b''іb''b''дb'' b''гb''b''рb''b''уb''b''пb''b''иb'' b''бb''b''уb''b''дb'' ↩
      b''еb'' 1000.
  # b''Bb''b''оb''b''нb''b''аb'' b''мb''b''аb''b''єb'' b''вb''b''лb''b''аb''b''сb''b' ↩
      'нb''b''иb''b''йb'' b''вb''b''иb''b''хb''b''іb''b''дb'' S32, b''яb''b''кb''b''иb''b' ↩
      'йb'' b''бb''b''уb''b''дb''b''еb'' 20 b''пb''b''рb''b''иb'' true b''іb'' 0 b''пb''b' ↩
      'рb''b''иb'' false.
  # b''Bb''b''оb''b''нb''b''аb'' b''тb''b''аb''b''кb''b''оb''b''жb'' b''мb''b''аb''b' ↩
      'єb'' b''кb''b''оb''b''нb''b''тb''b''аb''b''кb''b''тb'' b''сb''b''тb''b''аb''b''нb'' ↩
      b''уb'', b''яb''b''кb''b''иb''b''йb'' b''бb''b''уb''b''дb''b''еb'' b''сb''b''лb''b' ↩
      'іb''b''дb''b''уb''b''вb''b''аb''b''тb''b''иb'' b''зb''b''аb'' b''їb''b''їb'' b' ↩
      'пb''b''оb''b''тb''b''оb''b''чb''b''нb''b''иb''b''мb'' b''сb''b''тb''b''аb''b''нb''b ↩
      ''оb''b''мb''.
  # b''Ob''b''сb''b''кb''b''іb''b''лb''b''ьb''b''кb''b''иb'' b''цb''b''яb'' b''кb''b' ↩
      'нb''b''оb''b''пb''b''кb''b''аb'' b''зb''b''нb''b''аb''b''хb''b''оb''b''дb''b''иb''b ↩
      ''тb''b''ьb''b''сb''b''яb'' b''вb'' b''гb''b''рb''b''уb''b''пb''b''іb'', DEFAULT b' ↩
      'нb''b''еb'' b''мb''b''аb''b''єb'' b''зb''b''нb''b''аb''b''чb''b''еb''b''нb''b''нb'' ↩
      b''яb''.
  # b''Ob''b''сb''b''кb''b''іb''b''лb''b''ьb''b''кb''b''иb'' OUTPUT b''нb''b''еb'' b' ↩
      'єb'' «COMMAND», b''зb''b''аb''b''пb''b''иb''b''сb''b''иb'' _COMMAND b''іb''b''гb''b ↩
      ''нb''b''оb''b''рb''b''уb''b''юb''b''тb''b''ьb''b''сb''b''яb''.
  [[[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
```

Кнопки перемикання Кнопки-перемикачі змінюють стан лише при кожному натисканні кнопки. Визначення кнопок-перемикачів починаються з тексту «TOGGLE_BUTTON» в одинарних дужках.

```
[TOGGLE_BUTTONS]
  # b''Kb''b''оb''b''жb''b''нb''b''аb'' b''нb''b''аb''b''зb''b''вb''b''аb'' b''кb''b''нb''b ↩
      ''оb''b''пb''b''кb''b''иb'' b''вb'' b''пb''b''оb''b''дb''b''вb''b''іb''b''йb''b''нb''b ↩
      ''иb''b''хb'' b''дb''b''уb''b''жb''b''кb''b''аb''b''хb'' b''пb''b''оb''b''вb''b''иb''b ↩
      ''нb''b''нb''b''аb'' b''бb''b''уb''b''тb''b''иb'' b''уb''b''нb''b''іb''b''кb''b''аb''b ↩
      ''лb''b''ьb''b''нb''b''оb''b''юb'' b''іb'' b''чb''b''уb''b''тb''b''лb''b''иb''b''вb''b ↩
      ''оb''b''юb'' b''дb''b''оb'' b''рb''b''еb''b''гb''b''іb''b''сb''b''тb''b''рb''b''уb''.
  # b''Цb''b''яb'' b''кb''b''нb''b''оb''b''пb''b''кb''b''аb'', b''нb''b''аb''b''зb''b''вb'' ↩
      b''аb''b''нb''b''аb'' «tool_change», b''кb''b''еb''b''рb''b''уb''b''єb''b''тb''b''ьb'' ↩
      b''сb''b''яb'' b''кb''b''лb''b''аb''b''вb''b''іb''b''шb''b''еb''b''юb'' b''рb''b''яb'' ↩
      b''дb''b''кb''b''аb'' 2 b''сb''b''тb''b''оb''b''вb''b''пb''b''цb''b''яb'' 5.
  # b''Bb''b''оb''b''нb''b''аb'' b''мb''b''аb''b''єb'' b''вb''b''иb''b''хb''b''іb''b''дb''  ↩
      BIT, b''яb''b''кb''b''иb''b''йb'' b''вb''b''иb''b''дb''b''аb''b''єb'' 1 b''уb'' b' ↩
      'сb''b''пb''b''рb''b''аb''b''вb''b''жb''b''нb''b''ьb''b''оb''b''мb''b''уb'' b''сb''b' ↩
```

```
    ’тb’’b’’ab’’b’’нb’’b’’ib’’ b’’тb’’b’’ab’’ 0 b’’yb’’ b’’xb’’b’’иb’’b’’бb’’b’’нb’’b’ ↩
    ’ob’’b’’мb’’b’’yb’’ b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’ib’’.
  # b’’Bb’’b’’ob’’b’’нb’’b’’ab’’ b’’тb’’b’’ab’’b’’кb’’b’’ob’’b’’жb’’ b’’мb’’b’’ab’’b’’єb’’  ↩
    b’’кb’’b’’ob’’b’’нb’’b’’тb’’b’’ab’’b’’кb’’b’’тb’’ b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’yb’’, ↩
     b’’яb’’b’’кb’’b’’иb’’b’’йb’’ b’’cb’’b’’лb’’b’’ib’’b’’дb’’b’’yb’’b’’єb’’ b’’зb’’b’ ↩
    ’ab’’ b’’ïb’’b’’ïb’’ b’’пb’’b’’ob’’b’’тb’’b’’ob’’b’’чb’’b’’нb’’b’’иb’’b’’мb’’ b’’cb’’b ↩
    ’’тb’’b’’ab’’b’’нb’’b’’ob’’b’’мb’’.
  # DEFAULT b’’вb’’b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’ob’’b’’вb’’b’’лb’’b’’юb’’b’’єb’’ b’’цb’’b ↩
    ’’єb’’ b’’зb’’b’’нb’’b’’ab’’b’’чb’’b’’єb’’b’’нb’’b’’нb’’b’’яb’’ b’’яb’’b’’кb’’ b’’ib’’ ↩
    b’’cb’’b’’тb’’b’’иb’’b’’нb’’b’’нb’’b’’єb’’ b’’пb’’b’’pb’’b’’иb’’ b’’пb’’b’’єb’’b’’pb’’ ↩
    b’’шb’’b’’ib’’b’’йb’’ b’’ib’’b’’нb’’b’’ib’’b’’цb’’b’’ib’’b’’ab’’b’’лb’’b’’ib’’b’’зb’’b ↩
    ’’ab’’b’’цb’’b’’ib’’b’’ïb’’.
  # _COMMAND b’’нb’’b’’єb’’ b’’вb’’b’’иb’’b’’кb’’b’’ob’’b’’pb’’b’’иb’’b’’cb’’b’’тb’’b’’ob’’ ↩
    b’’вb’’b’’yb’’b’’єb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’, b’’ob’’b’’cb’’b’’кb’’b’’ib’’b’’лb’’ ↩
    b’’ьb’’b’’кb’’b’’иb’’ OUTPUT b’’нb’’b’’єb’’ b’’вb’’b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’ob’’ ↩
    b’’вb’’b’’лb’’b’’єb’’b’’нb’’b’’ob’’ b’’нb’’b’’ab’’ COMMAND, b’’ab’’b’’лb’’b’’єb’’ b’ ↩
    ’пb’’b’’єb’’b’’pb’’b’’єb’’b’’вb’’b’’ib’’b’’pb’’b’’кb’’b’’ab’’
  # b’’дb’’b’’ob’’b’’дb’’b’’ab’’b’’cb’’b’’тb’’b’’ьb’’ b’’pb’’b’’яb’’b’’дb’’b’’кb’’b’’иb’’ b ↩
    ’’нb’’b’’єb’’b’’зb’’b’’ab’’b’’лb’’b’’єb’’b’’жb’’b’’нb’’b’’ob’’ b’’вb’’b’’ib’’b’’дb’’ b ↩
    ’’цb’’b’’ьb’’b’’ob’’b’’гb’’b’’ob’’.
  [[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
```

Кнопки миттєвого натискання Миттєві кнопки мають значення true при натисканні та false при відпусканні. Визначення миттєвих кнопок починаються з тексту *MOMENTARY_BUTTON* в одинарних дужках.

```
[MOMENTARY_BUTTONS]
  # b’’Kb’’b’’ob’’b’’жb’’b’’нb’’b’’ab’’ b’’нb’’b’’ab’’b’’зb’’b’’вb’’b’’ab’’ b’’кb’’b’’нb’’b ↩
    ’’ob’’b’’пb’’b’’кb’’b’’иb’’ b’’вb’’ b’’пb’’b’’ob’’b’’дb’’b’’вb’’b’’ib’’b’’йb’’b’’нb’’b ↩
    ’’иb’’b’’xb’’ b’’дb’’b’’yb’’b’’жb’’b’’кb’’b’’ab’’b’’xb’’ b’’пb’’b’’ob’’b’’вb’’b’’иb’’b ↩
    ’’нb’’b’’нb’’b’’ab’’ b’’бb’’b’’yb’’b’’тb’’b’’иb’’ b’’yb’’b’’нb’’b’’ib’’b’’кb’’b’’ab’’b ↩
    ’’лb’’b’’ьb’’b’’нb’’b’’ob’’b’’юb’’ b’’ib’’ b’’чb’’b’’yb’’b’’тb’’b’’лb’’b’’иb’’b’’вb’’b ↩
    ’’ob’’b’’юb’’ b’’дb’’b’’ob’’ b’’pb’’b’’єb’’b’’гb’’b’’ib’’b’’cb’’b’’тb’’b’’pb’’b’’yb’’.
  # b’’Цb’’b’’яb’’ b’’кb’’b’’нb’’b’’ob’’b’’пb’’b’’кb’’b’’ab’’, b’’нb’’b’’ab’’b’’зb’’b’’вb’’ ↩
    b’’ab’’b’’нb’’b’’ab’’ «spindle_rev», b’’кb’’b’’єb’’b’’pb’’b’’yb’’b’’єb’’b’’тb’’b’’ьb’’ ↩
    b’’cb’’b’’яb’’ b’’кb’’b’’лb’’b’’ab’’b’’вb’’b’’ib’’b’’шb’’b’’єb’’b’’юb’’ b’’pb’’b’’яb’’ ↩
    b’’дb’’b’’кb’’b’’ab’’ 2 b’’cb’’b’’тb’’b’’ob’’b’’вb’’b’’пb’’b’’цb’’b’’яb’’ 3.
  # b’’Bb’’b’’ob’’b’’нb’’b’’ab’’ b’’мb’’b’’ab’’b’’єb’’ b’’вb’’b’’иb’’b’’xb’’b’’ib’’b’’дb’’  ↩
    COMMAND, b’’тb’’b’’ob’’b’’мb’’b’’yb’’ b’’бb’’b’’yb’’b’’дb’’b’’єb’’ b’’вb’’b’’иb’’b’ ↩
    ’кb’’b’’ob’’b’’pb’’b’’иb’’b’’cb’’b’’тb’’b’’ob’’b’’вb’’b’’yb’’b’’вb’’b’’ab’’b’’тb’’b’ ↩
    ’иb’’ TRUE_COMMAND b’’ib’’ FALSE_COMMAND.
  # b’’Bb’’b’’ob’’b’’нb’’b’’ab’’ b’’тb’’b’’ab’’b’’кb’’b’’ob’’b’’жb’’ b’’мb’’b’’ab’’b’’єb’’  ↩
    b’’кb’’b’’ob’’b’’нb’’b’’тb’’b’’ab’’b’’кb’’b’’тb’’ b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’yb’’, ↩
     b’’яb’’b’’кb’’b’’иb’’b’’йb’’ b’’бb’’b’’yb’’b’’дb’’b’’єb’’ b’’cb’’b’’лb’’b’’ib’’b’ ↩
    ’дb’’b’’yb’’b’’вb’’b’’ab’’b’’тb’’b’’иb’’ b’’зb’’b’’ab’’ b’’їb’’b’’їb’’ b’’пb’’b’’ob’’b ↩
    ’’тb’’b’’ob’’b’’чb’’b’’нb’’b’’иb’’b’’мb’’ b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’ob’’b’’мb’’.
  # COMMANDs b’’мb’’b’’ab’’b’’тb’’b’’иb’’b’’мb’’b’’єb’’ b’’ib’’b’’мb’’’b’’яb’’ b’’кb’’b’ ↩
    ’ob’’b’’мb’’b’’ab’’b’’нb’’b’’дb’’b’’иb’’, b’’ab’’ b’’пb’’b’’ob’’b’’тb’’b’’ib’’b’’мb’’  ↩
    b’’бb’’b’’yb’’b’’дb’’b’’ьb’’-b’’яb’’b’’кb’’b’’ib’’ b’’нb’’b’’єb’’b’’ob’’b’’бb’’b’’xb’’ ↩
    b’’ib’’b’’дb’’b’’нb’’b’’ib’’ b’’ab’’b’’pb’’b’’гb’’b’’yb’’b’’мb’’b’’єb’’b’’нb’’b’’тb’’b ↩
    ’’иb’’.
  # b’’Цb’’b’’яb’’ TRUE_COMMAND b’’вb’’b’’иb’’b’’кb’’b’’лb’’b’’иb’’b’’кb’’b’’ab’’b’’єb’’ b’ ↩
    ’вb’’b’’нb’’b’’yb’’b’’тb’’b’’pb’’b’’ib’’b’’шb’’b’’нb’’b’’юb’’ b’’кb’’b’’ob’’b’’мb’’b’ ↩
    ’ab’’b’’нb’’b’’дb’’b’’yb’’ b’’дb’’b’’лb’’b’’яb’’ b’’зb’’b’’ab’’b’’пb’’b’’yb’’b’’cb’’b’ ↩
```

```
    ’кв’’в’’уь’’ в’’шь’’в’’пь’’в’’иь’’в’’нь’’в’’дь’’в’’еь’’в’’ль’’в’’яь’’ в’’вь’’ в’’зь’’в ↩
    ’’вь’’в’’оь’’в’’рь’’в’’оь’’в’’ть’’в’’нь’’в’’оь’’в’’мь’’в’’уь’’ в’’нь’’в’’аь’’в’’пь’’в’ ↩
    ’рь’’в’’яь’’в’’мь’’в’’кь’’в’’уь’’ в’’зь’’в’’іь’’ в’’шь’’в’’вь’’в’’иь’’в’’дь’’в’’кь’’в’ ↩
    ’іь’’в’’сь’’в’’ть’’в’’юь’’ 200 в’’оь’’в’’бь’’/в’’хь’’в’’вь’’.
# в’’Яь’’в’’кь’’в’’щь’’в’’оь’’ в’’шь’’в’’пь’’в’’иь’’в’’нь’’в’’дь’’в’’еь’’в’’ль’’в’’ьь’’ в ↩
    ’’вь’’в’’жь’’в’’еь’’ в’’зь’’в’’аь’’в’’пь’’в’’уь’’в’’щь’’в’’еь’’в’’нь’’в’’иь’’в’’йь’’,  ↩
    в’’шь’’в’’вь’’в’’иь’’в’’дь’’в’’кь’’в’’іь’’в’’сь’’в’’ть’’в’’ьь’’ в’’оь’’в’’бь’’в’’еь’’в ↩
    ’’рь’’в’’ть’’в’’аь’’в’’нь’’в’’нь’’в’’яь’’ в’’зь’’в’’бь’’в’’іь’’в’’ль’’в’’ьь’’в’’шь’’в’ ↩
    ’иь’’в’’ть’’в’’ьь’’в’’сь’’в’’яь’’.
# DEFAULT в’’нь’’в’’еь’’ в’’вь’’в’’иь’’в’’кь’’в’’оь’’в’’рь’’в’’иь’’в’’сь’’в’’ть’’в’’оь’’в ↩
    ’’вь’’в’’уь’’в’’єь’’в’’ть’’в’’ьь’’в’’сь’’в’’яь’’ в’’зь’’ в’’мь’’в’’иь’’в’’ть’’в’’ть’’в ↩
    ’’єь’’в’’вь’’в’’иь’’в’’мь’’в’’иь’’ в’’кь’’в’’нь’’в’’оь’’в’’пь’’в’’кь’’в’’аь’’в’’мь’’в’ ↩
    ’иь’’.
# _STATE в’’нь’’в’’еь’’ в’’вь’’в’’иь’’в’’кь’’в’’оь’’в’’рь’’в’’иь’’в’’сь’’в’’ть’’в’’оь’’в’ ↩
    ’вь’’в’’уь’’в’’юь’’в’’ть’’в’’ьь’’в’’сь’’в’’яь’’, в’’оь’’в’’сь’’в’’кь’’в’’іь’’в’’ль’’в’ ↩
    ’ьь’’в’’кь’’в’’иь’’ OUTPUT в’’вь’’в’’сь’’в’’ть’’в’’аь’’в’’нь’’в’’оь’’в’’вь’’в’’ль’’в’ ↩
    ’еь’’в’’нь’’в’’оь’’ в’’нь’’в’’аь’’ COMMAND, в’’аь’’в’’ль’’в’’еь’’ в’’пь’’в’’еь’’в’ ↩
    ’рь’’в’’еь’’в’’вь’’в’’іь’’в’’рь’’в’’кь’’в’’аь’’
# в’’дь’’в’’оь’’в’’дь’’в’’аь’’в’’сь’’в’’ть’’в’’ьь’’ в’’рь’’в’’яь’’в’’дь’’в’’кь’’в’’иь’’ в ↩
    ’’нь’’в’’еь’’в’’зь’’в’’аь’’в’’ль’’в’’еь’’в’’жь’’в’’нь’’в’’оь’’ в’’вь’’в’’іь’’в’’дь’’ в ↩
    ’’ць’’в’’ьь’’в’’оь’’в’’гь’’в’’оь’’.
[[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 Довідник внутрішніх команд

Існує кілька внутрішніх команд, які ви можете використовувати.

home_selected

- обов’язковий аргумент: номер осі (ціле число)

unhome_selected

- обов’язковий аргумент: номер осі (ціле число)

spindle_forward_adjust

- необов’язковий аргумент: початкове число RPM (ціле число) - за замовчуванням 100
- Опис: Якщо шпиндель зупинено, він почне обертатися вперед. Якщо він вже працює, швидкість обертання збільшиться або зменшиться залежно від напрямку обертання шпинделя.

spindle_forward

- необов’язковий аргумент: початкове число RPM (ціле число) - за замовчуванням 100

spindle_reverse

- необов’язковий аргумент: початкове число RPM (ціле число) - за замовчуванням 100

spindle_reverse_adjust

- необов’язковий аргумент: початкове число RPM (ціле число) - за замовчуванням 100
- Опис: Якщо шпиндель зупинено, він почне обертатися у зворотному напрямку. Якщо він вже працює, швидкість обертання збільшиться або зменшиться залежно від напрямку обертання шпинделя.

spindle_faster

- Опис: збільшує швидкість шпинделя на 100 об/хв

spindle_slower

- Опис: зменшує швидкість шпинделя на 100 об/хв, доки об/хв не досягне 100

set_linear_jog_velocity

- обов’язковий аргумент: швидкість у дюймах за хвилину (число з плаваючою комою)
- опис: встановлює швидкість штовхання по осях 0,1,2,6,7,8 (X,Y,Z,U,V,W)

set_angular_jog_velocity

- обов’язковий аргумент: швидкість у градусах за хвилину (число з плаваючою комою)
- опис: встановлює швидкість штовхання на осях 3, 4, 5 (A.B.C)

continuous_jog

- обов’язкові аргументи: номер осі (ціле число), напрямок (ціле число)

incremental_jog

- обов’язкові аргументи: номер осі (ціле число), напрямок (ціле число), відстань (число з плаваючою комою)

quill_up

- додаткові аргументи: абсолютне положення осі Z машини (число з плаваючою комою)
- Опис: Перемістити вісь Z у задану позицію верстата

feed_hold

- обов’язковий аргумент: стан (bool 0 або 1)

feed_override

- обов’язковий аргумент: ставка (число з плаваючою комою)

rapid_override

- обов’язковий аргумент: rate (float 0-1)

spindle_override

- обов’язковий аргумент: ставка (число з плаваючою комою)

max_velocity

- обов’язковий аргумент: ставка (число з плаваючою комою)

optional_stop

- обов’язковий аргумент: стан (bool 0 або 1)

block_delete

- обов’язковий аргумент: стан (bool 0 або 1)

single_block

- обов’язковий аргумент: стан (bool 0 або 1)

smart_cycle_start

- Опис: У режимі очікування запускає програму G-коду, якщо призупинено, виконується один рядок.

re_start line

- обов’язковий аргумент: номер рядка (ціле число)

mdi_and_return

- обов’язковий аргумент: команда(и) G-коду
- Опис: записує поточний режим, викликає команди та повертається до цього режиму.

mdi

- обов’язковий аргумент: команда(и) G-коду
- Опис: встановлює режим MDI, викликає команди.

13.1.5 Повідомлення ZMQ

Panelui може надсилати повідомлення на основі ZMQ при натисканні кнопок.
Таким чином, panelui може взаємодіяти з іншими програмами, такими як екрани QtVCP.

```
[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
```

Ось приклад програми, яка отримає повідомлення та виведе його на термінал.

```
import zmq
import json

# ZeroMQ Context
context = zmq.Context()

# b''Bb''b''ив''b''зb''b''нb''b''ab''b''чb''b''тb''b''eb'' b''cb''b''ob''b''кb''b''eb''b' ↩
    'тb'' b''зb''b''ab'' b''дb''b''ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' "b' ↩
    'Kb''b''ob''b''нb''b''тb''b''eb''b''кb''b''cb''b''тb''b''yb''"
sock = context.socket(zmq.SUB)

# b''Bb''b''ив''b''зb''b''нb''b''ab''b''чb''b''тb''b''eb'' b''пb''b''ib''b''дb''b''пb''b' ↩
    'ив''b''cb''b''кb''b''yb'' b''тb''b''ab'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b' ↩
    'мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''зb'' b''тb''b''eb''b''мb''b''ob''b''юb'' b' ↩
    'дb''b''лb''b''яb'' b''пb''b''pb''b''ив''b''йb''b''нb''b''яb''b''тb''b''тb''b''яb''.
topic = "" # b''вb''b''cb''b''ib'' b''тb''b''eb''b''мb''b''ив''
sock.setsockopt(zmq.SUBSCRIBE, topic)
sock.connect("tcp://127.0.0.1:5690")

b''пb''b''ob''b''кb''b''ив'' True:
    topic, message = sock.recv_multipart()
    print('{} sent message:{}'.format(topic,json.loads(message)))
```

13.1.6 Розширення файлу обробника

Спеціальний файл можна використовувати для додавання власного коду Python, який буде доступний як команди. panelui_handler.py повинен бути написаний на Python і розміщений у папці конфігурації. Якщо panelui знайде там файл, він додасть його виклики функцій до доступних команд. Ось приклад файлу обробника, який додає дві функції — hello_world і cycle_mode:

```
# b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b''нb''b''ив''b''йb'' b''вb''b' ↩
    'ив''b''кb''b''лb''b''ив''b''кb'' b''ob''b''бb''b''pb''b''ob''b''бb''b''нb''b''ив''b' ↩
    'кb''b''ab'' - b''цb''b''eb'' b''зb''b''ab''b''вb''b''жb''b''дb''b''ив'' b''бb''b''yb''b ↩
    ''дb''b''eb'' b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb''b''ob''b''вb''b''ив''b' ↩
    'мb''
def get_handlers(linuxcnc_stat, linucnc_cmd, commands, master):
     return [HandlerClass(linuxcnc_stat, linucnc_cmd, commands, master)]

# b''Tb''b''ab''b''кb''b''ob''b''жb'' b''ob''b''бb''b''ob''b''вb'''b''яb''b''зb''b''кb''b' ↩
    'ob''b''вb''b''ив''b''йb'' - b''кb''b''лb''b''ab''b''cb'' b''ob''b''бb''b''pb''b''ob''b' ↩
    'бb''b''нb''b''ив''b''кb''b''ab'' b''кb''b''лb''b''ab''b''cb''b''yb'' HandlerClass:

    # b''Цb''b''eb'' b''бb''b''yb''b''дb''b''eb'' b''дb''b''ob''b''cb''b''ив''b''тb''b' ↩
        'ьb'' b''cb''b''тb''b''ab''b''нb''b''дb''b''ab''b''pb''b''тb''b''нb''b''ив''b''мb'' ↩
        b''дb''b''лb''b''яb'' b''ob''b''тb''b''pb''b''ив''b''мb''b''ab''b''нb''b''нb''b' ↩
        'яb'' b''дb''b''ob''b''cb''b''тb''b''yb''b''пb''b''yb'' b''дb''b''ob'' b''вb''b' ↩
        'cb''b''ьb''b''ob''b''гb''b''ob''.
    # linuxcnc_stat: b''цb''b''eb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b' ↩
        'яb''b''pb'' b''cb''b''тb''b''ab''b''нb''b''yb'' Python b''дb''b''лb''b''яb'' ↩
        LinuxCNC.
    # linuxcnc_cmd: b''цb''b''eb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb'' ↩
        b''pb'' b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''ив'' Python b''дb''b''лb''b' ↩
        'яb'' LinuxCNC.
    # commands: b''цb''b''eb'' b''eb''b''кb''b''зb''b''eb''b''мb''b''пb''b''лb''b''яb''b' ↩
        'pb'' b''кb''b''ob''b''мb''b''ab''b''нb''b''дb''b''ив'', b''зb''b''ab'' b''дb''b' ↩
        'ob''b''пb''b''ob''b''мb''b''ob''b''гb''b''ob''b''юb'' b''яb''b''кb''b''ob''b''гb''b ↩
        ''ob'' b''мb''b''ob''b''жb''b''нb''b''ab'' b''вb''b''ив''b''кb''b''лb''b''ив''b' ↩
```

```
        'кb''b''ab''b''тb''b''иb'' b''вb''b''нb''b''уb''b''тb''b''рb''b''іb''b''шb''b''нb''b ↩
        ''ib'' b''пb''b''рb''b''оb''b''цb''b''еb''b''дb''b''уb''b''рb''b''иb''.
    # master: b''нb''b''ab''b''дb''b''ab''b''єb'' b''дb''b''оb''b''сb''b''тb''b''уb''b' ↩
        'пb'' b''дb''b''оb'' b''оb''b''сb''b''нb''b''оb''b''вb''b''нb''b''иb''b''хb'' b' ↩
        'фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''йb''/b''дb''b''ab''b''нb''b''иb''b''хb''.

    def __init__(self, linuxcnc_stat, linuxcnc_cmd, commands, master):
        self.parent = commands
        self.current_mode = 0

    # b''Кb''b''оb''b''мb''b''аb''b''нb''b''дb''b''иb'' b''пb''b''оb''b''вb''b''иb''b''нb'' ↩
        b''нb''b''іb'' b''мb''b''аb''b''тb''b''иb'' b''тb''b''аb''b''кb''b''иb''b''йb'' b' ↩
        'вb''b''иb''b''гb''b''лb''b''яb''b''дb'':
    # def some_name(self, widget_instance, arguments from widget):
    # widget_instance b''нb''b''аb''b''дb''b''аb''b''єb'' b''дb''b''оb''b''сb''b''тb''b' ↩
        'уb''b''пb'' b''дb''b''оb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''їb''/b' ↩
        'дb''b''аb''b''нb''b''иb''b''хb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''уb''b ↩
        ''юb''b''чb''b''иb''b''мb'' b''вb''b''іb''b''дb''b''жb''b''еb''b''тb''b''оb''b' ↩
        'мb''.
    # arguments b''мb''b''оb''b''жb''b''еb'' b''бb''b''уb''b''тb''b''иb'' b''сb''b''пb''b' ↩
        'иb''b''сb''b''кb''b''оb''b''мb'' b''аb''b''рb''b''гb''b''уb''b''мb''b''еb''b''нb''b ↩
        ''тb''b''іb''b''вb'', b''оb''b''дb''b''нb''b''иb''b''мb'' b''аb''b''рb''b''гb''b' ↩
        'уb''b''мb''b''еb''b''нb''b''тb''b''оb''b''мb'' b''аb''b''бb''b''оb'' None,
    # b''зb''b''аb''b''лb''b''еb''b''жb''b''нb''b''оb'' b''вb''b''іb''b''дb'' b''тb''b' ↩
        'оb''b''гb''b''оb'', b''щb''b''оb'' b''бb''b''уb''b''лb''b''оb'' b''вb''b''кb''b' ↩
        'аb''b''зb''b''аb''b''нb''b''оb'' b''вb'' INI-b''фb''b''аb''b''йb''b''лb''b''іb'' ↩
        panelui.
    def hello_world(self, wname, m):
        # b''вb''b''иb''b''вb''b''еb''b''сb''b''тb''b''иb'' b''нb''b''аb'' b''тb''b''еb''b' ↩
            'рb''b''мb''b''іb''b''нb''b''аb''b''лb'', b''щb''b''оb''b''бb'' b''мb''b''иb'' b ↩
            ''зb''b''нb''b''аb''b''лb''b''иb'', b''щb''b''оb'' b''вb''b''сb''b''еb'' b''пb'' ↩
            b''рb''b''аb''b''цb''b''юb''b''єb''
        print('\nHello world\n')
        print(m)     # b''вb''b''иb''b''вb''b''еb''b''сb''b''тb''b''иb'' b''аb''b''рb''b' ↩
            'гb''b''уb''b''мb''b''еb''b''нb''b''тb''(b''иb'')
        print(wname.metadata)    # b''вb''b''иb''b''вb''b''еb''b''сb''b''тb''b''иb'' b' ↩
            'вb''b''нb''b''уb''b''тb''b''рb''b''іb''b''шb''b''нb''b''іb'' b''мb''b''еb''b' ↩
            'тb''b''аb''b''дb''b''аb''b''нb''b''іb'' b''вb''b''иb''b''кb''b''лb''b''иb''b' ↩
            'кb''b''уb''b''юb''b''чb''b''иb''b''хb'' b''вb''b''іb''b''дb''b''жb''b''еb''b' ↩
            'тb''b''іb''b''вb'' (b''зb'' b''фb''b''аb''b''йb''b''лb''b''уb'' b''кb''b''оb''b ↩
            ''нb''b''фb''b''іb''b''гb''b''уb''b''рb''b''аb''b''цb''b''іb''b''їb'')

        # b''Вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''аb''b''йb''b''тb''b''еb'' b''кb''b' ↩
            'оb''b''мb''b''аb''b''нb''b''дb''b''уb'' mdi b''дb''b''лb''b''яb'' b''дb''b' ↩
            'рb''b''уb''b''кb''b''уb'' b''пb''b''оb''b''вb''b''іb''b''дb''b''оb''b''мb''b' ↩
            'лb''b''еb''b''нb''b''нb''b''яb'' b''вb'' LinuxCNC.
        # b''Дb''b''лb''b''яb'' b''цb''b''ьb''b''оb''b''гb''b''оb'' b''пb''b''оb''b''тb''b' ↩
            'рb''b''іb''b''бb''b''нb''b''оb'', b''щb''b''оb''b''бb'' LinuxCNC b''бb''b''уb'' ↩
            b''вb'' b''вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''лb''b''еb''b''нb''b' ↩
            'иb''b''йb'', b''аb''b''лb''b''еb'' b''цb''b''еb'' b''нb''b''еb'' b''пb''b''еb'' ↩
            b''рb''b''еb''b''вb''b''іb''b''рb''b''яb''b''єb''b''тb''b''ьb''b''сb''b''яb''.
        # b''Бb''b''аb''b''тb''b''ьb''b''кb''b''іb''b''вb''b''сb''b''ьb''b''кb''b''іb'' b' ↩
            'кb''b''оb''b''мb''b''аb''b''нb''b''дb''b''иb'' b''оb''b''чb''b''іb''b''кb''b' ↩
            'уb''b''юb''b''тb''b''ьb'' widget_instance - b''зb''b''аb''b''мb''b''іb''b''нb'' ↩
            b''юb''b''єb''b''тb''b''ьb''b''сb''b''яb'' None.
        self.parent.mdi(None,'(MSG, Hello Linuxcnc World!)')

    # b''Кb''b''оb''b''жb''b''еb''b''нb'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb'' b' ↩
        'цb''b''іb''b''єb''b''їb'' b''фb''b''уb''b''нb''b''кb''b''цb''b''іb''b''їb'' b''цb'' ↩
        b''иb''b''кb''b''лb''b''іb''b''чb''b''нb''b''оb'' b''зb''b''мb''b''іb''b''нb''b' ↩
        'юb''b''вb''b''аb''b''тb''b''иb''b''мb''b''еb'' b''рb''b''еb''b''жb''b''иb''b''мb'' ↩
        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)

    # b''Kb''b''ob''b''дb'' b''кb''b''ob''b''тb''b''лb''b''ab'', b''чb''b''ab''b''cb''b' ↩
        'тb''b''ob'' b''пb''b''ob''b''тb''b''рb''b''ib''b''бb''b''eb''b''нb''
    def __getitem__(self, item):
        return getattr(self, item)
    def __setitem__(self, item, value):
        return setattr(self, item, value)
```

13.2 Модуль LinuxCNC на Python

У цій документації описано модуль python `linuxcnc`, який надає API Python для взаємодії з LinuxCNC.

13.2.1 Вступ

Користувацькі інтерфейси керують діяльністю LinuxCNC, надсилаючи повідомлення NML до контролер завдань LinuxCNC, та контролюють результати, спостерігаючи за структурою стану LinuxCNC, а також каналом повідомлення про помилки.

Програмний доступ до NML здійснюється через C++ API; однак найважливіші частини інтерфейсу NML до LinuxCNC також доступні для програм на Python через модуль `linuxcnc`.

Окрім інтерфейсу NML для каналів команд, стану та помилок, модуль `linuxcnc` також містить:

- підтримка читання значень з INI-файлів

13.2.2 Шаблони використання інтерфейсу LinuxCNC NML

Загальна схема використання `linuxcnc` приблизно така:

- Імпортуйте модуль `linuxcnc`.
- За потреби встановіть з’єднання з каналами NML команд, стану та помилок.
- Опитуйте канал стану періодично або за потреби.
- Перед відправкою команди визначте за статусом, чи дійсно можна це зробити (наприклад, немає сенсу відправляти команду «Виконати», якщо завдання перебуває в стані ESTOP або інтерпретатор не перебуває в режимі очікування).
- Надішліть команду за допомогою одного з методів каналу команди `linuxcnc`.

Щоб отримати повідомлення з каналу помилок, періодично опитуйте канал помилок та обробляйте будь-які отримані повідомлення.

- Опитуйте канал стану періодично або за потреби.
- Виведіть будь-яке повідомлення про помилку та дослідіть код винятку.

Модуль `linuxcnc` також визначає тип винятку Python `error` для підтримки звітності про помилки.

13.2.3 Зчитування стану LinuxCNC за допомогою модуля linuxcnc Python

Ось фрагмент коду Python для дослідження вмісту об'єкта `linuxcnc.stat`, який містить понад 80 значень (типові значення запускаються під час роботи LinuxCNC):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
import linuxcnc
try:
    s = linuxcnc.stat() # b''cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''пb''b ↩
        ''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''eb''b''нb''b''нb''b''яb'' b''дb''b''ob'' ↩
         b''кb''b''ab''b''нb''b''ab''b''лb''b''yb'' b''cb''b''тb''b''ab''b''нb''b''yb''
    s.poll() # b''ob''b''тb''b''pb''b''иb''b''мb''b''ab''b''тb''b''иb'' b''пb''b''ob''b' ↩
        'тb''b''ob''b''чb''b''нb''b''ib'' b''зb''b''нb''b''ab''b''чb''b''eb''b''нb''b''нb''b ↩
        ''яb''
except linuxcnc.error, detail:
    print("error", detail)
    sys.exit(1)
for x in dir(s):
    if not x.startswith("_"):
        print(x, getattr(s,x))
```

Модуль linuxcnc використовує стандартний скомпільований шлях до файлу конфігурації NML, якщо його не перевизначено, див. приклад ReadingINI file values.

13.2.3.1 Атрибути linuxcnc.stat

прискорення
(повертає число з комою) – прискорення за замовчуванням, відображає запис INI [TRAJ]DEFAULT

active_queue
(повертає ціле число) – кількість рухів, що змішуються.

actual_position
(повертає кортеж чисел з плаваючою комою) - поточна позиція траєкторії, (x y z a b c u v w) в одиницях вимірювання.

adaptive_feed_enabled
(повертає логічне значення) – стан адаптивного коригування швидкості подачі (0/1).

ain
(повертає кортеж чисел з плаваючою комою) - поточне значення аналогових вхідних контактів.

angular_units
(повертає число з комою) – одиниці кутової довжини на градус, відображає значення INI [TRAJ]ANGULAR_UNITS.

aout
(повертає кортеж чисел з плаваючою комою) - поточне значення аналогового виходу.

axes (Вилучено з версії 2.9_)
замість цього використовуйте `axis_mask.bit_count()`, щоб отримати кількість налаштованих осей.

axis
(повертає кортеж словників) - відображає поточні значення осі. Див. Словник осей.

axis_mask
(повертає ціле число) – маска доступної осі, визначена параметром `[TRAJ]COORDINATES` у INI-файлі. Повертає суму значень осей X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256.

block_delete
(повертає логічне значення) – поточний стан видалення блоку.

call_level
(повертає ціле число)` - поточна глибина підпрограми. - 0, якщо не в підпрограмі, глибина, якщо не вказано інше.

команда
(повертає рядок) – команда, що виконується на даний момент.

current_line
(повертає ціле число) – рядок, що виконується в даний момент.

current_vel
(повертає число з комою) - поточна швидкість в одиницях виміру користувача за секунду.

cycle_time
(повертає число з плаваючою комою) - період потоку

налагодження
(повертає ціле число) - прапорець налагодження з INI-файлу.

delay_left
(повертає число з комою) – час, що залишився на команді затримки (G4), секунди.

din
(повертає кортеж цілих чисел) - поточне значення цифрових вхідних контактів.

distance_to_go
(повертає число з плаваючою комою) - відстань, що залишилася до кінця поточного руху, згідно з планувальником траєкторії.

dout
(повертає кортеж цілих чисел) - поточне значення цифрових вихідних контактів.

dtg
(повертає кортеж чисел з плаваючою комою) - відстань, що залишилася від поточного переміщення для кожної осі, як повідомляється планувальником траєкторії.

echo_serial_number
(повертає ціле число) - Серійний номер останньої виконаної команди, надісланої інтерфейсом користувача до завдання. Усі команди мають серійний номер. Після виконання команди її серійний номер відображається в `echo_serial_number`.

увімкнено
(повертає логічне значення) – прапорець увімкненого планувальника траєкторії.

estop
(повертає ціле число) - Повертає або STATE_ESTOP, або ні.

exec_state
(повертає ціле число) – стан виконання завдання. Один із EXEC_ERROR, EXEC_DONE, EXEC_WAITING_FOR_MOTION, EXEC_WAITING_FOR_MOTION_QUEUE, EXEC_WAITING_FOR_IO, EXEC_WAITING_FOR_MOTION_AND_IO, EXEC_WAITING_FOR_DELAY, EXEC_WAITING_FOR_SYSTE EXEC_WAITING_FOR_SPINDLE_ORIENTED.

feed_hold_enabled
(повертає логічне значення) – увімкнути прапорець для затримки подачі.

feed_override_enabled
(повертає логічне значення) – прапорець увімкнення для перевизначення каналу.

feedrate
(повертає число з плаваючою комою) – поточне коригування швидкості подачі, 1.0 = 100%.

file
(повертає рядок) – ім’я поточного завантаженого файлу G-коду зі шляхом.

flood
(повертає ціле число) – Стан повені, FLOOD_OFF або FLOOD_ON.

g5x_index
(повертає ціле число) - поточна активна система координат, G54=1, G55=2 тощо.

g5x_offset
(повертає кортеж чисел з плаваючою комою) - зміщення поточної активної системи координат.

g92_offset
(повертає кортеж чисел з плаваючою комою) - положення поточного зміщення g92.

gcodes
(повертає кортеж цілих чисел) – Активні G-коди для кожної модальної групи.
Цілочисельні значення відображають номінальні числа G-кодів, помножені на 10. (Приклади: 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
(повертає кортеж цілих чисел) – поточні розміщені з’єднання, де 0 = не розміщено, 1 = розміщено.

id
(повертає ціле число) – ідентифікатор руху, що виконується на даний момент.

ini_filename
(повертає рядок) – шлях до INI-файлу, переданого до linuxcnc.

inpos
(повертає логічне значення) – прапорець «машина в позиції».

input_timeout
(повертає логічне значення) – прапорець для таймера M66, що працює.

interp_state
(повертає ціле число) – поточний стан інтерпретатора RS274NGC. Один з INTERP_IDLE, INTERP_READING, INTERP_PAUSED, INTERP_WAITING.

interpreter_errcode
(повертає ціле число) - поточний код повернення інтерпретатора RS274NGC. Один із IN-TERP_OK, INTERP_EXIT, INTERP_EXECUTE_FINISH, INTERP_ENDFILE, INTERP_FILE_NOT_OPEN, INTERP_ERROR. Див src/emc/nml_intf/interp_return.hh

joint
(повертає кортеж словників) - відображає поточні спільні значення. Див. Спільний словник.

joint_actual_position
(повертає кортеж чисел з плаваючою комою) – фактичні позиції суглобів.

joint_position
(повертає кортеж чисел з плаваючою комою) – Бажані позиції суглобів.

joints
(повертає ціле число) – кількість суглобів. Відображає значення INI [KINS]JOINTS.

kinematics_type
(повертає ціле число) – Тип кінематики. Один з:

- KINEMATICS_IDENTITY
- KINEMATICS_FORWARD_ONLY
- KINEMATICS_INVERSE_ONLY
- KINEMATICS_BOTH

limit
(повертає кортеж цілих чисел) – маски граничних значень осей. minHardLimit=1, maxHardLimit=2, minSoftLimit=4, maxSoftLimit=8.

linear_units
(повертає число з комою) – одиниці лінійного обчислення на мм, відображає значення INI [TRAJ]LINEAR_UNITS.

max_acceleration
(повертає число з комою) – максимальне прискорення. Відображає [TRAJ]MAX_ACCELERATION.

max_velocity
(повертає число з комою) – максимальна швидкість. Відображає поточну максимальну швидкість. Якщо не змінено за допомогою halui.max-velocity або подібного, воно має відображати [TRAJ]MAX_VELOCITY.

mcodes
(повертає кортеж з 10 цілих чисел) – активні на даний момент M-коди.

mist
(повертає ціле число) - Стан туману, MIST_OFF або MIST_ON

motion_line
(повертає ціле число) – виконується рух номера вихідного рядка. Зв'язок з `id` незрозумілий.

motion_mode
(повертає ціле число) – Це режим контролера руху. Один з TRAJ_MODE_COORD, TRAJ_MODE_FRE
TRAJ_MODE_TELEOP.

motion_type
(повертає ціле число) – Тип поточного виконуваного руху. Один з:

- MOTION_TYPE_TRAVERSE
- MOTION_TYPE_FEED
- MOTION_TYPE_ARC
- MOTION_TYPE_TOOLCHANGE
- MOTION_TYPE_PROBING
- MOTION_TYPE_INDEXROTARY
- Або 0, якщо наразі рух не відбувається.

optional_stop
(повертає ціле число) – прапорець зупинки опції.

paused
(повертає логічне значення) - прапорець `рух призупинено`.

single_stepping
(returns boolean) - `motion single_stepping` flag.

pocket_prepped
(повертає ціле число) – Команда Tx завершена, і ця кишеня підготовлена. -1, якщо кишеня не підготовлена.

poll()
Метод -*(вбудована функція)* для оновлення атрибутів поточного стану.

position
(повертає кортеж чисел з плаваючою комою) - позиція на траєкторії.

probe_tripped
(повертає логічне значення) - прапорець, True, якщо зонд спрацював (фіксатор).

probe_val
(повертає ціле число) - відображає значення виводу `motion.probe-input`.

probed_position
(повертає кортеж чисел з плаваючою комою) – позиція спрацьовування зонда.

probing
(повертає логічне значення) – прапорець, True, якщо виконується операція зондування.

program_units
(повертає ціле число) - одне з CANON_UNITS_INCHES=1, CANON_UNITS_MM=2, CANON_UNITS_

queue
(повертає ціле число) – поточний розмір черги планувальника траєкторії.

queue_full
(повертає логічне значення) – черга планувальника траєкторії заповнена.

rapidrate
(повертає число з плаваючою комою) – швидке перевизначення масштабу.

read_line
(повертає ціле число) – рядок, який зараз читає інтерпретатор RS274NGC.

rotation_xy
(повертає число з плаваючою комою) - поточний кут повороту XY навколо осі Z.

settings
(повертає кортеж чисел з плаваючою комою) - поточні налаштування інтерпретатора:
settings[0] = порядковий номер,
settings[1] = швидкість подачі,
settings[2] = швидкість,
settings[3] = толерантність змішування `G64 P`,
settings[4] = толерантність наївного CAM `G64 Q`.

spindle
’ (повертає кортеж словників) ’ - повертає поточний стан шпинделя, див. Словник шпинделя.

spindles
(повертає ціле число) – кількість шпинделів. Відображає INI-значення `[TRAJ]SPINDLES`.

state
(повертає ціле число) – поточний стан виконання команди. Один з RCS_DONE, RCS_EXEC, RCS_ERROR.

task_mode
(повертає ціле число) – поточний режим завдання. Один з MODE_MDI, MODE_AUTO, MODE_MANUAL.

task_paused
(повертає ціле число) – прапорець призупинення завдання.

task_state
(повертає ціле число) – поточний стан завдання. Один з STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON, STATE_OFF.

tool_in_spindle
(повертає ціле число) – поточний номер інструменту.

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
(повертає ціле число) – номер гнізда для поточного завантаженого інструмента (0, якщо інструмент не завантажено).

tool_offset
(повертає кортеж чисел з плаваючою комою) – значення зміщення поточного інструменту.

tool_table
(повертає кортеж tool_results) - список записів інструменту. Кожен запис є послідовністю наступних полів: id, xoffset, yoffset, zoffset, aoffset, boffset, coffset, uoffset, voffset, woffset, diameter, frontangle, backangle, orientation. id та orientation є цілими числами, а решта - числами з плаваючою комою.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
# b''щb''b''оb''b''бb'' b''зb''b''нb''b''аb''b''йb''b''тb''b''иb'' b''іb''b''нb''b''фb''b' ↩
    'оb''b''рb''b''мb''b''аb''b''цb''b''іb''b''юb'' b''пb''b''рb''b''оb'' b''зb''b''аb''b' ↩
    'вb''b''аb''b''нb''b''тb''b''аb''b''жb''b''еb''b''нb''b''иb''b''йb'' b''іb''b''нb''b' ↩
    'сb''b''тb''b''рb''b''уb''b''мb''b''еb''b''нb''b''тb'', b''вb''b''оb''b''нb''b''аb'' b' ↩
    'зb''b''нb''b''аb''b''хb''b''оb''b''дb''b''иb''b''тb''b''ьb''b''сb''b''яb'' b''вb'' b' ↩
    'тb''b''аb''b''бb''b''лb''b''иb''b''цb''b''іb'' b''іb''b''нb''b''сb''b''тb''b''рb''b' ↩
    'уb''b''мb''b''еb''b''нb''b''тb''b''іb''b''вb'' b''зb'' b''іb''b''нb''b''дb''b''еb''b' ↩
    'кb''b''сb''b''оb''b''мb'' 0
b''яb''b''кb''b''щb''b''оb'' s.tool_table[0].id != 0: # b''іb''b''нb''b''сb''b''тb''b''рb'' ↩
    b''уb''b''мb''b''еb''b''нb''b''тb'' b''зb''b''аb''b''вb''b''аb''b''нb''b''тb''b''аb''b' ↩
    'жb''b''еb''b''нb''b''оb''
    print(s.tool_table[0].zoffset)
else:
    print("No tool loaded.")
```

toolinfo(*toolno*)
«(повертає словник tooldata для toolno)» — для ініціалізації необхідний початковий stat.poll(). *toolno* повинен бути більшим за нуль і меншим або рівним найвищому номеру інструменту, що використовується. Елементи словника включають всі елементи tooldata: *toolno*, *pocketno*, *diameter*,*frontangle*,*backangle*,*orientation*, *xoffset*,*yoffset*, … *woffset*, *comment*.
Як приклад, наступний скрипт

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
toolno = 1
print(s.toolinfo(toolno))
```

```
b''вb''b''иb''b''дb''b''аb''b''єb'' b''рb''b''еb''b''зb''b''уb''b''лb''b''ьb''b''тb''b' ↩
    'аb''b''тb'':
```

```
': 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
(повертає число з плаваючою комою) – Ця властивість визначена, але не має корисної інтерпретації.

13.2.3.2 Словник «axis»

Значення конфігурації та стану осі доступні через список словників для кожної осі. Ось приклад того, як отримати доступ до атрибуту певної осі: Зверніть увагу, що багато властивостей, які раніше були в словнику `axis`, тепер знаходяться в словнику `joint`, оскільки на нетривіальних кінематичних машинах ці елементи (такі як люфт) не є властивостями осі.

max_position_limit
(повертає число з плаваючою комою) – максимальне обмеження (м'яке обмеження) для руху осі, в параметрі конфігурації машини units. відображає `[JOINT___n__]MAX_LIMIT`.

min_position_limit
(повертає число з плаваючою комою) – мінімальне обмеження (м'яке обмеження) для руху осі, в параметрі конфігурації машини units. відображає `[JOINT___n__]MIN_LIMIT`.

velocity
(повертає число з плаваючою комою) - поточна швидкість.

13.2.3.3 Словник «joint»

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
s.poll()
print("b''Cb''b''тb''b''иb''b''кb'' 1 b''зb'' b''дb''b''оb''b''мb''b''аb''b''шb''b''нb''b' ↩
    'іb''b''мb'' b''рb''b''оb''b''зb''b''тb''b''аb''b''шb''b''уb''b''вb''b''аb''b''нb''b' ↩
    'нb''b''яb''b''мb'': ", s.joint[1]["homed"])
```

Для кожного суглоба доступні такі ключі словника:

backlash
(повертає число з комою) – Люфт у машинних одиницях. Параметр конфігурації, відображає `[JOINT___n__]BACKLASH`.

увімкнено
(повертає ціле число) – ненульове значення означає увімкнено.

fault
(повертає ціле число) - ненульове значення означає несправність підсилювача осі.

ferror_current
(повертає число з плаваючою комою) – поточна наступна помилка.

ferror_highmark
(повертає число з плаваючою комою) – величина максимальної помилки відстеження.

homed
(повертає ціле число) – ненульове значення означає, що було додано до основного поля.

homing
(повертає ціле число) – ненульове значення означає, що виконується перенаправлення.

inpos
(повертає ціле число) – ненульове значення означає «на місці».

input
(повертає число з плаваючою комою) – поточна позиція вводу.

jointType
(повертає ціле число) – тип параметра конфігурації осі, відображає `[JOINT___n__]TYPE` з `LINEAR=1`, `ANGULAR=2`. Див. Конфігурація Joint INI для отримання детальної інформації.

max_ferror
(повертає число з комою) – максимальна наступна помилка. параметр конфігурації, відображає `[JOINT___n__]FERROR`.

max_hard_limit
(повертає ціле число) – ненульове значення означає перевищення максимального жорсткого ліміту.

max_position_limit
(повертає число з плаваючою комою) – максимальне обмеження (м'яке обмеження) для руху суглоба, в одиницях виміру. параметр конфігурації, відображає `[JOINT___n__]MAX_LIMIT`.

max_soft_limit
ненульове значення означає, що `max_position_limit` було перевищено, int

min_ferror
(повертає число з комою) – параметр конфігурації, відображає `[JOINT___n__]MIN_FERROR`.

min_hard_limit
(повертає ціле число) – ненульове значення означає перевищення мінімального жорсткого ліміту.

min_position_limit
(повертає число з плаваючою комою) – мінімальне обмеження (м'яке обмеження) для руху суглоба, в одиницях виміру. параметр конфігурації, відображає `[JOINT___n__]MIN_LIMIT`.

min_soft_limit
(повертає ціле число) – ненульове значення означає, що перевищено `min_position_limit`.

output
(повертає число з плаваючою комою) – позиція виводу, що була задана.

override_limits
(повертає ціле число) – ненульове значення означає, що обмеження перевизначено.

units
(повертає число з плаваючою комою) – одиниці вимірювання з'єднання на мм або на градус для кутових з'єднань.
(одиниці вимірювання з'єднання такі ж, як і одиниці машинного з'єднання, якщо не встановлено інше параметром конфігурації `[JOINT___n__]UNITS`)

velocity
(повертає число з плаваючою комою) - поточна швидкість.

13.2.3.4 Словник «веретено»

brake
(повертає ціле число) – значення прапорця гальма шпинделя.

direction
(повертає ціле число) - напрямок обертання шпинделя, де вперед=1, назад=-1.

увімкнено
(повертає ціле число) – значення прапорця ввімкнення шпинделя.

homed
(наразі не реалізовано)

increasing
(повертає ціле число) – незрозуміло.

orient_fault
(повертає ціле число)

orient_state
(повертає ціле число)

override
(повертає число з плаваючою комою) – шкала корекції швидкості шпинделя.

override_enabled
(повертає логічне значення) – значення прапорця ввімкнення перевизначення шпинделя.

speed
(повертає float) - значення швидкості обертання шпинделя, об/хв, > 0: за годинниковою стрілкою, < 0: проти годинникової стрілки.
При активному G96 це відображає максимальну швидкість, встановлену опціональним словом G96 *D*, або, якщо слово *D* відсутнє, значення за замовчуванням +/-1e30.

13.2.4 Підготовка до надсилання команд

Деякі команди завжди можна надсилати, незалежно від режиму та стану; наприклад, метод `linuxcnc.command.abort()` можна викликати завжди.

Інші команди можуть бути надіслані лише у відповідному стані, і ці перевірки можуть бути дещо складними. Наприклад, команду MDI можна надіслати лише якщо:

- ESTOP не було запущено, і
- машина ввімкнена та
- осі розташовані в самому центрі та
- інтерпретатор не працює, і

• режим встановлено на режим MDI

Відповідна перевірка перед надсиланням команди MDI через `linuxcnc.command.mdi()` може бути такою:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
s = linuxcnc.stat()
c = linuxcnc.command()

def ok_for_mdi():
    s.poll()
    b''пb''b''ob''b''вb''b''eb''b''pb''b''нb''b''уb''b''тb''b''иb'' b''нb''b''eb'' s.estop  ↩
        b''тb''b''ab'' s.enabled b''тb''b''ab'' (s.homed.count(1) == s.joints) b''ib'' (s. ↩
        interp_state == linuxcnc.INTERP_IDLE)

if ok_for_mdi():
    c.mode(linuxcnc.MODE_MDI)
    c.wait_complete() # b''зb''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'', b' ↩
        'пb''b''ob''b''кb''b''иb'' b''нb''b''eb'' b''бb''b''уb''b''дb''b''eb'' b''вb''b' ↩
        'иb''b''кb''b''ob''b''нb''b''ab''b''нb''b''ob'' b''пb''b''eb''b''pb''b''eb''b''мb''b ↩
        ''иb''b''кb''b''ab''b''нb''b''нb''b''яb'' b''pb''b''eb''b''жb''b''иb''b''мb''b''ib'' ↩
        b''вb'' (b''ab''b''бb''b''ob'' b''нb''b''eb'' b''нb''b''ab''b''cb''b''тb''b''ab''b' ↩
        'нb''b''eb'' b''тb''b''ab''b''йb''b''мb''-b''ab''b''уb''b''тb'' b''зb''b''ab'' b' ↩
        'зb''b''ab''b''мb''b''ob''b''вb''b''чb''b''уb''b''вb''b''ab''b''нb''b''нb''b''яb''b' ↩
        'мb'' 5 b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''!).
    c.mdi("G0 X10 Y20 Z30")
```



Warning
Важливу інформацію про *wait_complete()* див. у розділі «Методи linuxcnc.command» нижче.

13.2.5 Надсилання команд через `linuxcnc.command`

Перед надсиланням команди ініціалізуйте канал команди таким чином:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import linuxcnc
c = linuxcnc.command()

# b''Пb''b''pb''b''иb''b''кb''b''лb''b''ab''b''дb''b''иb'' b''вb''b''иb''b''кb''b''ob''b' ↩
    'pb''b''иb''b''cb''b''тb''b''ab''b''нb''b''нb''b''яb'' b''дb''b''eb''b''яb''b''кb''b' ↩
    'иb''b''xb'' b''кb''b''ob''b''мb''b''ab''b''нb''b''дb'', b''пb''b''eb''b''pb''b''eb''b' ↩
    'лb''b''ib''b''чb''b''eb''b''нb''b''иb''b''xb'' b''нb''b''иb''b''жb''b''чb''b''eb'':
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` атрибути

`serial`
поточний серійний номер команди

13.2.5.2 `linuxcnc.command` методи:

`abort()`
надіслати повідомлення EMC_TASK_ABORT.

`auto(int[, int])`
запустити, покроково виконати, призупинити або відновити програму.

`brake(int)`
увімкнути або відпустити гальмо шпинделя.

`debug(int)`
встановити рівень налагодження за допомогою повідомлення EMC_SET_DEBUG.

`display_msg(string)`
надсилає повідомлення оператора на екран. (максимум 254 символи)

`error_msg(string)`
надсилає на екран повідомлення про помилку оператора. (максимум 254 символи)

`feedrate(float)`
встановити корекцію швидкості подачі, 1,0 = 100%.

flood(int)
увімкнути/вимкнути затоплення.

Синтаксис
LinuxCNCflood(command)
flood(linuxcnc.FLOOD_ON)
flood(linuxcnc.FLOOD_OFF)

Константи
FLOOD_ON
FLOOD_OFF

home(int)
додому даний суглоб.

jog(command-constant, bool, int[, float[, float]])

Синтаксис
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)

Константи команд
linuxcnc.JOG_STOP
linuxcnc.JOG_CONTINUOUS
linuxcnc.JOG_INCREMENT

jjogmode

True
запит на індивідуальний спільний поштовх (потрібне teleop_enable(0))

False
запит на переміщення по осі за декартовими координатами (потрібне teleop_enable(1))

joint_num_or_axis_index

Для спільного поштовху (jjogmode=1)
joint_number

Для осі декартових координат штовхання (jjogmode=0)
нульовий індекс координати осі відносно відомих літер координат XYZABCUVW (x=>0,y=>

load_tool_table()
перезавантажте таблицю інструментів.

maxvel(float)
встановити максимальну швидкість

mdi(string)
надіслати команду MDI. Максимум 254 символи.

mist(int)
увімкнути/вимкнути туман.

Синтаксис
туман(команда)
туман(linuxcnc.MIST_ON)
туман(linuxcnc.MIST_OFF)

Константи
MIST_ON
MIST_OFF

mode(int)
встановити режим (MODE_MDI, MODE_MANUAL, MODE_AUTO).

override_limits()
встановити прапорець перевизначення меж осі.

program_open(string)
відкрити файл NGC.

rapidrate()
встановити коефіцієнт швидкого перевизначення

reset_interpreter()
скинути налаштування інтерпретатора RS274NGC

set_adaptive_feed(int)
встановити прапорець адаптивної подачі

set_analog_output(int, float)
встановити значення аналогового вихідного контакту

set_block_delete(int)
встановити прапорець видалення блоку

set_digital_output(int, int)
встановити значення цифрового виходу

set_feed_hold(int)
увімкнення/вимкнення блокування подачі

set_feed_override(int)
увімкнення/вимкнення перевизначення подачі

set_max_limit(int, float)
встановити максимальне обмеження положення для заданої осі

set_min_limit()
встановити мінімальне обмеження положення для заданої осі

set_optional_stop(int)
встановити додаткову зупинку ввімкнено/вимкнено

set_spindle_override(int [, int])
увімкнено корекцію шпинделя. За замовчуванням встановлено на шпиндель 0.

spindle(direction: int, speed: float=0, spindle: int=0, wait_for_speed: int=0)

- Напрямок: [SPINDLE_FORWARD, SPINDLE_REVERSE, SPINDLE_OFF, SPINDLE_INCREASE, SPINDLE_DECRE or SPINDLE_CONSTANT]
- Швидкість: Швидкість в обертах за хвилину, за замовчуванням 0.
- Шпиндель: Номер шпинделя для команди за замовчуванням дорівнює 0.
- Wait_for_speed: якщо 1 рух очікуватиме на швидкість перед продовженням, за замовчуванням встановлено значення «ні».



Warning
Команди MDI проігнорують це. ”S1000” після цього вимкне шпиндель.

text_msg(string)
надсилає оператору текстове повідомлення на екран (максимум 254 символи).

```
#!/usr/bin/env python3
import linuxcnc
c = linuxcnc.command()

# b''Зb''b''бb''b''іb''b''лb''b''ьb''b''шb''b''тb''b''еb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''іb''b''сb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''еb''b''лb''b' ↩
    'яb'' 0 b''нb''b''аb'' 100 b''оb''b''бb''/b''хb''b''вb''. b''Шb''b''пb''b''иb''b''нb''b' ↩
    'дb''b''еb''b''лb''b''ьb'' b''мb''b''аb''b''єb'' b''бb''b''уb''b''тb''b''иb'' b''уb''b' ↩
    'вb''b''іb''b''мb''b''кb''b''нb''b''еb''b''нb''b''иb''b''йb'' b''пb''b''еb''b''рb''b' ↩
    'шb''b''иb''b''мb''.
c.spindle(linuxcnc.INCREASE)

# b''Зb''b''бb''b''іb''b''лb''b''ьb''b''шb''b''тb''b''еb'' b''шb''b''вb''b''иb''b''дb''b' ↩
    'кb''b''іb''b''сb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b''еb''b''лb''b' ↩
    'яb'' 2 b''нb''b''аb'' 100 b''оb''b''бb''/b''хb''b''вb''. b''Шb''b''пb''b''иb''b''нb''b' ↩
    'дb''b''еb''b''лb''b''ьb'' b''мb''b''аb''b''єb'' b''бb''b''уb''b''тb''b''иb'' b''уb''b' ↩
    'вb''b''іb''b''мb''b''кb''b''нb''b''еb''b''нb''b''иb''b''йb'' b''пb''b''еb''b''рb''b' ↩
    'шb''b''иb''b''мb''.
c.spindle(linuxcnc.SPINDLE_INCREASE, 2)

# b''Bb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''іb''b''тb''b''ьb'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''іb''b''сb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'еb''b''лb''b''яb'' b''вb''b''іb''b''дb'' 0 b''дb''b''оb'' 1024 b''оb''b''бb''/b''хb''b' ↩
    'вb''.
c.spindle.(linuxcnc.SPINDLE_FORWARD, 1024)

# b''Bb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''іb''b''тb''b''ьb'' b''шb''b''вb''b' ↩
    'иb''b''дb''b''кb''b''іb''b''сb''b''тb''b''ьb'' b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'еb''b''лb''b''яb'' 1 b''нb''b''аb'' -666 b''оb''b''бb''/b''хb''b''вb''.
c.spindle.(linuxcnc.SPINDLE_REVERSE, 666, 1)

# b''Зb''b''уb''b''пb''b''иb''b''нb''b''иb''b''тb''b''иb'' b''шb''b''пb''b''иb''b''нb''b' ↩
    'дb''b''еb''b''лb''b''ьb'' 0.
c.spindle.(linuxcnc.SPINDLE_OFF)

# b''Яb''b''вb''b''нb''b''оb'' b''зb''b''уb''b''пb''b''иb''b''нb''b''иb''b''тb''b''иb'' b' ↩
    'шb''b''пb''b''иb''b''нb''b''дb''b''еb''b''лb''b''ьb'' 0.
c.spindle.(linuxcnc.SPINDLE_OFF, 0)
```

spindleoverride(float [, int])
Встановити коефіцієнт корекції шпинделя. За замовчуванням шпиндель 0.

state(int)
Встановіть стан машини. Стан машини має бути STATE_ESTOP, STATE_ESTOP_RESET, STATE_ON, or STATE_OFF.

task_plan_sync()
Після завершення цього виклику VAR-файл на диску оновлюється значеннями з інтерпретатора в реальному часі.

teleop_enable(int)
Увімкнути/вимкнути режим телеоптичної операції (вимкнути для спільного бігання трусцем).

tool_offset(int, float, float, float, float, float, int)
Встановіть зміщення інструменту. Див. приклад використання вище.

traj_mode(int)
Встановити режим траєкторії. Режим є одним із MODE_FREE, MODE_COORD, or MODE_TELEOP.

unhome(int)
Зняти з дому заданий джоін.

wait_complete([float])
Очікує завершення останньої команди. Приймає необов'язкове значення часу очікування в секундах.
Якщо значення часу очікування пропущено, воно становить 5 секунд.
Повертає -1, якщо час очікування вичерпано.
Повертає RCS_DONE або RCS_ERROR залежно від статусу виконання команди.
Зверніть увагу, що виконання python буде заблоковано, доки ця функція не поверне значення.

13.2.6 Зчитування каналу помилок

Для обробки повідомлень про помилки підключіться до каналу помилок та періодично запитуйте його за допомогою функції poll().

Зверніть увагу, що канал NML для повідомлень про помилки має чергу (на відміну від каналів команд і статусів), що означає, що перший споживач повідомлення про помилку видаляє це повідомлення з черги; чи «побачить» це повідомлення інший споживач повідомлень про помилки (наприклад, AXIS), залежить від часу. Рекомендується мати лише одне завдання зчитування каналу помилок у налаштуваннях.

```
#!/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 Читання значень INI-файлу

Ось приклад зчитування значень з INI-файлу через об'єкт `linuxcnc.ini`:

```
#!/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() b''пb''b''оb''b''вb''b''еb''b''рb''b''тb''b''аb''b''єb'' None, b''яb''b' ↩
    'кb''b''щb''b''оb'' b''кb''b''лb''b''юb''b''чb'' b''нb''b''еb'' b''зb''b''нb''b''аb''b' ↩
    'йb''b''дb''b''еb''b''нb''b''оb'' -
# b''нb''b''аb''b''сb''b''тb''b''уb''b''пb''b''нb''b''аb'' b''іb''b''дb''b''іb''b''оb''b' ↩
    'мb''b''аb'' b''кb''b''оb''b''рb''b''иb''b''сb''b''нb''b''аb'' b''дb''b''лb''b''яb'' b' ↩
    'вb''b''сb''b''тb''b''аb''b''нb''b''оb''b''вb''b''лb''b''еb''b''нb''b''нb''b''яb'' b' ↩
    'зb''b''нb''b''аb''b''чb''b''еb''b''нb''b''нb''b''яb'' b''зb''b''аb'' b''зb''b''аb''b' ↩
    'мb''b''оb''b''вb''b''чb''b''уb''b''вb''b''аb''b''нb''b''нb''b''яb''b''мb'':
```

```
machine_name = inifile.find(”EMC”, ”MACHINE”) b’’ab’’b’’бb’’b’’ob’’ ”unknown”
print(”machine name: ”, machine_name)

# inifile.findall() b’’пb’’b’’ob’’b’’вb’’b’’eb’’b’’pb’’b’’тb’’b’’ab’’b’’єb’’ b’’cb’’b’’пb’’ ↩
    b’’иb’’b’’cb’’b’’ob’’b’’кb’’ b’’зb’’b’’бb’’b’’іb’’b’’гb’’b’’іb’’b’’вb’’ b’’ab’’b’’бb’’b’ ↩
    ’ob’’ b’’пb’’b’’ob’’b’’pb’’b’’ob’’b’’жb’’b’’нb’’b’’іb’’b’’йb’’ b’’cb’’b’’пb’’b’’иb’’b’ ↩
    ’cb’’b’’ob’’b’’кb’’
# b’’яb’’b’’кb’’b’’щb’’b’’ob’’ b’’кb’’b’’лb’’b’’юb’’b’’чb’’ b’’нb’’b’’eb’’ b’’зb’’b’’нb’’b’ ↩
    ’ab’’b’’йb’’b’’дb’’b’’eb’’b’’нb’’b’’ob’’:

extensions = inifile.findall(”FILTER”, ”PROGRAM_EXTENSION”)
print(”extensions: ”, extensions)

# b’’пb’’b’’eb’’b’’pb’’b’’eb’’b’’вb’’b’’иb’’b’’зb’’b’’нb’’b’’ab’’b’’чb’’b’’иb’’b’’тb’’b’ ↩
    ’иb’’ b’’cb’’b’’тb’’b’’ab’’b’’нb’’b’’дb’’b’’ab’’b’’pb’’b’’тb’’b’’нb’’b’’иb’’b’’йb’’ NML- ↩
    b’’фb’’b’’ab’’b’’йb’’b’’лb’’ b’’пb’’b’’ab’’b’’pb’’b’’ab’’b’’мb’’b’’eb’’b’’тb’’b’’pb’’b’ ↩
    ’ob’’b’’мb’’ INI, b’’яb’’b’’кb’’b’’щb’’b’’ob’’ b’’йb’’b’’ob’’b’’гb’’b’’ob’’ b’’зb’’b’ ↩
    ’ab’’b’’дb’’b’’ab’’b’’нb’’b’’ob’’
nmlfile = inifile.find(”EMC”, ”NML_FILE”)
if nmlfile:
    linuxcnc.nmlfile = os.path.join(os.path.dirname(sys.argv[1]), nmlfile)
```

Або для того ж INI-файлу, що й 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)

# b’’Дb’’b’’иb’’b’’вb’’. b’’пb’’b’’рb’’b’’иb’’b’’кb’’b’’лb’’b’’аb’’b’’дb’’ b’’вb’’b’’иb’’b’ ↩
    ’щb’’b’’eb’’ b’’щb’’b’’ob’’b’’дb’’b’’ob’’ b’’вb’’b’’иb’’b’’кb’’b’’ob’’b’’pb’’b’’иb’’b’ ↩
    ’cb’’b’’тb’’b’’ab’’b’’нb’’b’’нb’’b’’яb’’ b’’ob’’b’’бb’’’b’’єb’’b’’кb’’b’’тb’’b’’ab’’ ’ ↩
    inifile’
```

13.2.8 Тип `linuxcnc.positionlogger`

Деякі поради щодо використання можна почерпнути з `src/emc/usr_intf/gremlin/gremlin.py`.

13.2.8.1 члени

npts
кількість балів.

13.2.8.2 методи

start(float)
запустити реєстратор позицій та виконувати його кожні ARG секунд

clear()
очистити реєстратор позицій

stop()
зупинити реєстратор позицій

call()
Намалюйте задній план зараз.

last([int])
Повернути найновішу точку на графіку або Немає

13.3 Модуль HAL для Python

У цій документації описано модуль `hal` на Python, який надає API Python для створення та доступу до контактів і сигналів HAL.

13.3.1 Базове використання

```
#!/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 Функції

компонент
+
Сам компонент створюється за допомогою виклику конструктора `hal.component`. Аргументами є ім'я компонента HAL і (опціонально) префікс, що використовується для імен контактів і параметрів. Якщо префікс не вказано, використовується ім'я компонента.
.Приклад

```
h = hal.component("passthrough")
```

newpin
+
Створити новий контакт.
Аргументи: суфікс імені контакту, тип контакту та напрямок контакту. Для параметрів аргументами є: суфікс імені параметра, тип параметра та напрямок параметра.
Приклад:

```
h.newpin("in", hal.HAL_FLOAT, hal.HAL_IN)
```

готовий
Повідомляє системі HAL про ініціалізацію компонента. Блокує додавання контактів.

неготовий
Дозволяє компоненту додавати контакти після виклику *ready()*. Слід викликати *ready()* на компоненті після цього.

component_exists
Чи існує зазначений компонент на даний момент.

Приклад

```
hal.component_exists(”testpanel”)
```

component_is_ready
Чи готовий зазначений компонент на цей час.

Приклад

```
hal.component_is_ready(”testpanel”)
```

get_msg_level
Отримати поточний рівень повідомлень у реальному часі.

set_msg_level
Встановлює поточний рівень повідомлень у реальному часі, який використовується для налагоджу
інформації.

connect
Підключіть контакт до сигналу.

Приклад

```
hal.connect(”pinname”,”signal_name”)
```

відключитися
Від’єднайте контакт від сигналу.

Приклад

```
hal.disconnect(”pinname”)
```

get_value
Зчитувати пін, параметр або сигнал безпосередньо.

Приклад

```
value = hal.get_value(”iocontrol.0.emc-enable-in”)
```

get_info_pins()
Повертає список словників усіх системних пінів.

```
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()
Повертає список словників усіх системних сигналів.

```
listOfDicts = hal.get_info_signals()
signalName1 = listOfDicts[0].get('NAME')
signalValue1 = listOfDicts[0].get('VALUE')
driverPin1 = listOfDicts[0].get('DRIVER')
```

get_info_params()
Повертає список словників усіх системних параметрів.

```
listOfDicts = hal.get_info_params()
paramName1 = listOfDicts[0].get('NAME')
paramValue1 = listOfDicts[0].get('VALUE')
paramDirection1 = listOfDicts[0].get('DIRECTION')
```

new_sig
Створіть новий сигнал заданого типу.

Приклад

```
hal.new_sig("signalname",hal.HAL_BIT)
```

pin_has_writer
Чи підключений до вказаного контакту керуючий контакт?
Повертає значення True або False.

```
h.in.pin_has_writer()
```

get_name
Отримати назву об'єкта HAL.
Повернути рядок.

```
h.in.get_name()
```

get_type
Отримати тип об'єкта HAL.
Повертає ціле число.

```
h.in.get_type()
```

get_dir
Отримати тип напрямку об'єкта HAL.
Повертає ціле число.

```
h.in.get_dir()
```

get
Отримати значення об’єкта HAL.

```
h.in.get()
```

set
Встановіть значення об’єкта HAL.

```
h.out.set(10)
```

is_pin
Об’єкт є PIN-кодом чи параметром?
Повертає True або False.

```
h.in.is_pin()
```

sampler_base
TODO

stream_base
TODO

stream
TODO

set_p
Встановіть значення будь-якого виводу в системі HAL.

Приклад

```
hal.set_p(”pinname”,”10”)
```

set_s
Встановіть значення будь-якого непідключеного сигналу в системі HAL.

Приклад

```
hal.set_s(”signalname”,”10”)
```

13.4 Модуль Python GStat

13.4.1 Вступ

GStat — це клас Python, який використовується для надсилання повідомлень з LinuxCNC до інших програм Python. Він використовує GObject для доставки повідомлень, що полегшує прослуховування конкретної інформації. Це називається подієвим програмуванням, яке є більш ефективним, ніж одночасне опитування LinuxCNC кожною програмою. GladeVCP, Gscreen, Gmoccapy та QtVCP широко використовують GStat. GStat знаходиться в модулі `hal_glib`.

Огляд

- Спочатку програма імпортує модуль `hal_glib` та створює екземпляр GStat.
- Потім він «підключається» до повідомлень, які хоче відстежувати.
- GStat перевіряє стан LinuxCNC кожні 100 мс, і якщо є відмінності від останньої перевірки, він надсилає повідомлення зворотного виклику всім підключеним програмам з поточним станом.
- Коли GStat викликає зареєстровану функцію, вона надсилає об'єкт GStat разом із будь-якими кодами повернення з повідомлення.

Типові підписи коду:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', FUNCTION_TO_CALL)

def FUNCTION_TO_CALL(gstat_object, return_codes):
```

Часто LAMBDA використовується для видалення об'єкта GSTAT та маніпулювання кодами повернення:

```
GSTAT.connect('MESSGAE-TO-LISTEN-FOR', lambda o, return: FUNCTION_TO_CALL(not return))

def FUNCTION_TO_CALL(return_codes):
```

13.4.2 Зразок коду GStat

Існує кілька основних схем використання GStat, залежно від того, в якій бібліотеці ви їх використовуєте
Якщо ви використовуєте GStat з GladeVCP, Gscreen або QtVCP, бібліотека GObject не потрібна, оскільки ці набори інструментів вже налаштовані для роботи з GObject.

13.4.2.1 Зразок шаблону коду компонента HAL

Ця програма створює два виводи HAL, які виводять стан 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()

# b''зb''b''вb''b''оb''b''рb''b''оb''b''тb''b''нb''b''иb''b''йb'' b''вb''b''иb''b''кb''b' ↩
    'лb''b''иb''b''кb'' b''дb''b''лb''b''яb'' b''зb''b''мb''b''іb''b''нb''b''иb'' b''сb''b' ↩
    'тb''b''аb''b''нb''b''уb'' b''вb''b''иb''b''вb''b''оb''b''дb''b''уb'' HAL
def mode_changed(obj, data):
        h['g20'] = not data
        h['g21'] = data

# b''3b''b''pb''b''ob''b''6b''b''ib''b''tb''b''ьb'' b''кb''b''оb''b''мb''b''пb''b''оb''b' ↩
    'нb''b''еb''b''нb''b''тb'' b''іb'' b''вb''b''иb''b''вb''b''оb''b''дb''b''иb''
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()
```

```
# b''пb''b''ib''b''дb''b''кb''b''лb''b''юb''b''чb''b''иb''b''тb''b''иb'' b''пb''b''ob''b' ↩
    'вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' GSTAT b''дb''b' ↩
    'ob'' b''фb''b''уb''b''нb''b''кb''b''цb''b''ib''b''їb'' b''зb''b''вb''b''ob''b''рb''b' ↩
    'ob''b''тb''b''нb''b''ob''b''гb''b''ob'' b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b' ↩
    'уb''
GSTAT.connect("metric-mode-changed",mode_changed)

# b''пb''b''рb''b''иb''b''мb''b''уb''b''сb''b''иb''b''тb''b''иb'' GSTAT b''ib''b''нb''b' ↩
    'ib''b''цb''b''ib''b''аb''b''лb''b''ib''b''зb''b''уb''b''вb''b''аb''b''тb''b''иb'' b' ↩
    'cb''b''тb''b''аb''b''нb''b''иb''
GSTAT.forced_update()

# b''пb''b''eb''b''тb''b''лb''b''яb'' b''дb''b''ob'' b''вb''b''иb''b''хb''b''ob''b''дb''b' ↩
    'уb''
try:
    GLib.MainLoop().run()
except KeyboardInterrupt:
    raise SystemExit
```

Це можна завантажити за допомогою команди `loadusr python PATH-TO-FILE/FILENAME.py` або, якщо потрібно дочекатися створення контактів, перш ніж продовжувати:
`loadusr python -Wn metric_status PATH-TO-FILE/FILENAME.py`
Контакти будуть такими: `metric_status.g20` та `metric_status.g21`.

13.4.2.2 Шаблон коду розширення Python GladeVCP

У цьому файлі припускається, що є три мітки GTK з такими назвами:

- *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 Шаблон коду розширення Python QtVCP

QtVCP розширює GStat, тому його потрібно завантажувати по-різному, але всі повідомлення доступні в QtVCP.
Цей файл обробника припускає, що є три QLabel з такими назвами:

- *state_label*
- *e_state_label*
- *interp_state_label*

```
#!/usr/bin/env python3

b''зb'' b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''yb'' qtvcp.core b''Cb''b''тb''b''ab''b ↩
    ''нb''
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 Повідомлення

periodic
«(нічого не повертає)» – надсилається кожні 100 мс.

state-estop
(returns nothing) - Надсилається, коли LinuxCNC переходить у режим estop.

state-estop-reset
(returns nothing) - Надсилається, коли LinuxCNC виходить з режиму estop.

state-on
(returns nothing) - Надсилається, коли LinuxCNC перебуває у стані «машина ввімкнена».

state-off
(returns nothing) - Надсилається, коли LinuxCNC вимкнено.

homed
(returns string) - Надсилається, коли кожен джоін переводиться в хоум-код.

all-homed
(returns nothing) - Надсилається, коли всі визначені з’єднання переведені в вихідне положення.

not-all-homed
(returns string) - Надсилає список суглобів, які наразі не є хоум-ранами.

override_limits_changed
(returns string) - Надсилається, якщо LinuxCNC було наказано перевизначити свої обмеження.

hard-limits-tripped
(returns bool, Python List) - Надсилається, коли спрацьовує будь-яке жорстке обмеження. bool вказує, що якщо спрацьовує будь-яке обмеження, список показує поточні граничні значення всіх доступних з’єднань.

mode-manual
(returns nothing) - Надсилається, коли LinuxCNC перемикається в ручний режим.

mode-mdi
(returns nothing) - Надсилається, коли LinuxCNC перемикається в режим MDI.

mode-auto
(returns nothing) - Надсилається, коли LinuxCNC перемикається в автоматичний режим.

command-running
(returns nothing) - Надсилається під час запуску програми або MDI

command-stopped
(returns nothing) - Надсилається, коли програма або MDI зупинені

command-error
(returns nothing) - Надсилається, коли виникає помилка команди

interp-run
(returns nothing) - Надсилається, коли інтерпретатор LinuxCNC виконує MDI або програму.

interp-idle
(returns nothing) - Надсилається, коли інтерпретатор LinuxCNC неактивний.

interp-paused
(returns nothing) - Надсилається, коли інтерпретатор LinuxCNC призупинено.

interp-reading
(returns nothing) - Надсилається, коли інтерпретатор LinuxCNC читає.

interp-waiting
(returns nothing) - Надсилається, коли інтерпретатор LinuxCNC очікує.

jograte-changed
(returns float) - Відправляється, коли швидкість ручного переміщення змінилася.
LinuxCNC не має внутрішньої швидкості ручного переміщення.
Це внутрішня швидкість ручного переміщення GStat.
Очікується, що вона буде вказана в одиницях виміру, властивих для даної машини, незалежно від поточного режиму одиниць виміру.

jograte-angular-changed
(returns float) - Відправляється, коли змінюється кутова швидкість переміщення.
LinuxCNC не має внутрішньої кутової швидкості переміщення.
Це внутрішня швидкість переміщення GStat.
Очікується, що вона буде вказана в одиницях виміру, властивих для даної машини, незалежно від поточного режиму одиниць виміру.

jogincrement-changed
(returns float, text) - Відправляється, коли змінюється приріст кроку.
LinuxCNC не має внутрішнього приросту кроку.
Це внутрішній приріст кроку GStat.
Очікується, що він буде в нативних одиницях виміру машини, незалежно від поточного режиму одиниць виміру.

jogincrement-angular-changed
(returns float, text) - Відправляється, коли змінюється кутовий крок переміщення.
LinuxCNC не має внутрішнього кутового кроку переміщення.
Це внутрішній кутовий крок переміщення GStat.
Очікується, що він буде в нативних одиницях виміру машини, незалежно від поточного режиму одиниць виміру.

program-pause-changed
(returns bool) - Надсилається, коли програма призупинена/відновлена.

optional-stop-changed
(returns bool) - Надсилається, коли встановлено/знято необов’язкову зупинку

block-delete-changed
(returns bool) - надсилається, коли встановлено/скасовано видалення блоку.

file-loaded
(returns string) - Надсилається, коли LinuxCNC завантажує файл

reload-display
(returns nothing) - Надсилається, коли є запит на перезавантаження дисплея

line-changed
(returns integer) - Надсилається, коли LinuxCNC зчитує новий рядок.
LinuxCNC не оновлює це для кожного типу рядка.

tool-in-spindle-changed
(returns integer) - Надсилається, коли інструмент змінюється.

tool-info-changed
(returns Python object) - Надсилається, коли змінюється інформація про поточний інструмент.

current-tool-offset
(returns Python object) - Надсилається, коли змінюються поточні зміщення інструменту.

motion-mode-changed
(returns integer) - Надсилається, коли режим руху змінюється

spindle-control-changed
(returns integer, bool, integer, bool) - (номер шпинделя, стан шпинделя ввімкнено, запитуваний напрямок і швидкість шпинделя, стан на швидкості)
Надсилається, коли змінюється напрямок або стан роботи шпинделя, або змінюється швидкість.

current-feed-rate
(returns float) - Надсилається, коли змінюється поточна швидкість подачі.

current-x-rel-position
(повертає число з плаваючою комою) – Надсилається кожні 100 мс.

current-position
(returns pyobject, pyobject, pyobject, pyobject) - Надсилається кожні 100 мс.
Повертає кортежі позиції, відносної позиції, відстані до переміщення та фактичної позиції суглоба. Перед поверненням до початкової точки, на осях з кількома суглобами, дійсним є лише положення суглоба.

current-z-rotation
(returns float) - Надсилається, коли змінюється поточний кут повороту навколо осі Z

requested-spindle-speed-changed
(returns float) - Надсилається, коли змінюється поточний запитуваний RPM

actual-spindle-speed-changed
(returns float) - Надсилається, коли фактична швидкість обертання змінюється на основі даних виводу HAL `spindle.0.speed-in`.

spindle-override-changed
(returns float) - Надсилається, коли змінюється значення корекції шпинделя у відсотках

feed-override-changed
(returns float) - Надсилається, коли значення перевизначення каналу змінюється у відсотках

rapid-override-changed
(returns float) - Надсилається, коли змінюється значення швидкого перевизначення у відсотках (0-100)

max-velocity-override-changed
(returns float) - Надсилається, коли змінюється значення максимальної швидкості в одиницях за хвилину

feed-hold-enabled-changed
(returns bool) - Надсилається, коли змінюється стан затримки каналу

itime-mode
(returns bool) - Надсилається, коли змінюється стан G93 (режим зворотного часу)

fpm-mode
(returns bool) - Надсилається, коли змінюється стан G94 (режим подачі за хвилину)

fpr-mode
(returns bool) - Надсилається, коли змінюється стан G95 (режим подачі на оберт)

css-mode
(returns bool) - Надсилається, коли змінюється стан G96 (режим постійної подачі на поверхню)

rpm-mode
(returns bool) - Надсилається, коли змінюється стан G97 (режим постійних обертів)

radius-mode
(returns bool) - Надсилається, коли змінюється статус G8
відображати X у режимі радіуса

diameter-mode
(returns bool) - Надсилається, коли змінюється стан G7
відображати X у режимі діаметра

flood-changed
(returns bool) - Надсилається, коли змінюється стан охолоджувальної рідини.

mist-changed
(returns bool) - Надсилається, коли змінюється стан охолоджувальної рідини туманом.

m-code-changed
(returns string) - Надсилається, коли змінюються активні M-коди

g-code-changed
(returns string) - Надсилається під час активної зміни G-коду

metric-mode-changed
(returns bool) - Надсилається, коли змінюється статус G21

user-system-changed
(returns string) - Надсилається, коли змінюється система опорних координат (G5x)

mdi-line-selected
(returns string, string) - призначено для надсилання, коли користувач вибирає рядок MDI.
Це залежить від використовуваних віджетів/бібліотек.

gcode-line-selected
(returns integer) - призначено для надсилання, коли користувач вибирає рядок G-коду.
Це залежить від використаних віджетів/бібліотек.

graphics-line-selected
(returns integer) - призначено для надсилання, коли користувач вибирає графічну лінію.
Це залежить від використаних віджетів/бібліотек.

graphics-loading-progress
(returns integer) - призначений для повернення відсотка виконання завантаження або запуску програми.
Це залежить від використаних віджетів/бібліотек.

graphics-gcode-error
(returns string) - призначено для надсилання, коли під час завантаження виявлено помилку G-коду.
Це залежить від використаних віджетів/бібліотек.

graphics-gcode-properties
(returns Python dict) - Надсилається під час завантаження G-коду.
Словник містить такі ключі:

- `name` (*string*): Ім’я завантаженого файлу
- `size` (*string*): Розмір у байтах та рядках
- `g0` (*string*): Загальна швидкісна дистанція
- `g1` (*string*): Загальна відстань подачі
- `run` (*string*): Орієнтовний час виконання програми
- `toollist` (*list*): Список використаних інструментів
- `x` (*string*): X екстентів (меж) [1]

- `x_zero_rxy` (*string*): X-екстенції без обертання навколо z (межі) [1]
- `y` (*string*): Y екстентів (границь) [1]
- `y_zero_rxy` (*string*): Екстенти Y без обертання навколо z (межі) [1]
- `z` (*string*): Z-екстенти (межі) [1]
- `z_zero_rxy` (*string*): Z-екстенції без обертання навколо z (межі) [1]
- `machine_unit_sys` (*string*): Одиниці вимірювання (*Метричні* або *Імперські*)
- `gcode_units` (*string*): Одиниці вимірювання у файлі G-коду (*мм* або *дюйми*)

Note

1. Дивіться зображення extends non-rotated and extends rotated 30 degrees для кращого розуміння.

---

graphics-view-changed
(returns string, Python dict or None) - призначено для надсилання під час зміни графічного вигляду.
Це залежить від використаних віджетів/бібліотек.

mdi-history-changed
(returns None) - призначене для надсилання, коли потрібно перезавантажити історію MDI.
Це залежить від використаних віджетів/бібліотек.

machine-log-changed
(returns None) - призначено для надсилання, коли журнал машини змінився.
Це залежить від використаних віджетів/бібліотек.

update-machine-log
(returns string, string) - призначений для надсилання під час оновлення комп'ютера.
Це залежить від використовуваних віджетів/бібліотек.

move-text-lineup
(returns None) - призначене для надсилання під час переміщення курсора на один рядок вгору на дисплеї G-коду.
Це залежить від використовуваних віджетів/бібліотек.

move-text-linedown
(returns None) - призначене для надсилання під час переміщення курсора на один рядок вниз у відображенні G-коду.
Це залежить від використовуваного віджета/бібліотек.

dialog-request
(returns Python dict) - призначено для надсилання під час запиту діалогу графічного інтерфейсу.
Для зв'язку використовується словник Python. Словник повинен містити таку пару ключів:

- NAME: *requested dialog name*
  Словник зазвичай має кілька пар імен ключів — це залежить від діалогу.
  Діалоги повертають інформацію за допомогою загального повідомлення.
  Це залежить від використаного віджета/бібліотек.

focus-overlay-changed
(returns bool, string, Python object) - призначений для надсилання під час запиту на розміщення накладання поверх дисплея.
Це залежить від використаних віджетів/бібліотек.

play-sound
(returns string) - призначений для надсилання під час запиту відтворення певного звукового файлу.
Це залежить від використаного віджета/бібліотек.

virtual-keyboard
(returns string) - призначене для надсилання під час запиту екранної клавіатури.
Це залежить від використаного віджета/бібліотек.

dro-reference-change-request
(returns integer) - призначене для надсилання під час запиту віджета DRO на зміну його посилання.
0 = машина, 1 = відносна, 3 = залишкова відстань
Це залежить від використовуваного віджета/бібліотек.

show-preferences
(returns None) - призначене для надсилання під час запиту на відображення налаштувань екрана.
Це залежить від використовуваних віджетів/бібліотек.

shutdown
(returns None) - призначений для надсилання під час запиту на завершення роботи LinuxCNC.
Це залежить від використовуваних віджетів/бібліотек.

status-message
returns python dict (message), python dict (options) Призначено для екрана/панелі для отримання повідомлень про стан/журнал з віджетів, але може використовуватися загалом.
Очікується, що об’єкт прослуховування шукатиме та оброблятиме щонайменше такі записи:

Словник повідомлення міститиме:

- TITLE: (рядок)
- SHORTTEXT: (рядок)
- DETAILS: (рядок)

Список опцій включатиме:

- LEVEL: (ціле число)
- LOG: (логічна змінна)

Об’єкт прослуховування може використовувати це для відображення інформації в текстовому рядку або діалоговому вікні повідомлення.
РІВЕНЬ вказує на терміновість 0 = ЗАЗВИЧАЙ 1 = ПОПЕРЕДЖЕННЯ 2 = КРИТИЧНИЙ
LOG вказує, чи слід записувати повідомлення у файл/на сторінку, якщо це можливо.
Передбачається, що повідомлення LOG використовують запис DETAILS.

Приклад того, як надіслати повідомлення:

```
mess = {'SHORTTEXT':'File Copy Failed',
        'TITLE':'FileManager',
        'DETAILS':'b''Hb''b''ab'' b''дb''b''иb''b''cb''b''кb''b''yb'' b''нb''b''eb''b''дb'' ↩
           b''ob''b''cb''b''тb''b''ab''b''тb''b''нb''b''ьb''b''ob'' b''мb''b''ib''b''cb''b' ↩
           'цb''b''яb'' b''дb''b''лb''b''яb'' b''кb''b''ob''b''пb''b''ib''b''юb''b''вb''b' ↩
           'ab''b''нb''b''нb''b''яb'' b''фb''b''ab''b''йb''b''лb''b''yb'''}
opt = {'LOG':True,'LEVEL':2}
STATUS.emit('status-message',mess,opt)
```

Приклад об'єкта прослуховування:

```
# b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''иb''b''тb''b''иb'' STATUS, b''щb''b' ↩
    'ob'' b''мb''b''иb'' b''xb''b''ob''b''чb''b''eb''b''мb''b''ob'' b''вb''b''ib''b''дb''b' ↩
    'пb''b''ob''b''вb''b''ib''b''дb''b''ab''b''тb''b''иb'' b''нb''b''ab'' b''бb''b''yb''b' ↩
    'дb''b''ьb''-b''яb''b''кb''b''ib'' b''нb''b''ab''b''дb''b''ib''b''cb''b''лb''b''ab''b' ↩
    'нb''b''ib'' b''пb''b''ob''b''вb''b''ib''b''дb''b''ob''b''мb''b''лb''b''eb''b''нb''b' ↩
    'нb''b''яb'' b''тb''b''иb''b''пb''b''yb'' «status-message».
STATUS.connect('status-message', lambda w, d, o: self.add_external_status(m,o))

    def add_external_status(self, message, option):

        # b''вb''b''иb''b''лb''b''yb''b''чb''b''eb''b''нb''b''нb''b''яb'' b''тb''b''ab'' b' ↩
            'пb''b''eb''b''pb''b''eb''b''xb''b''ob''b''пb''b''лb''b''eb''b''нb''b''нb''b' ↩
            'яb'' b''пb''b''ob''b''мb''b''иb''b''лb''b''ob''b''кb'' b''дb''b''лb''b''яb'' b' ↩
            'ob''b''чb''b''ib''b''кb''b''yb''b''вb''b''ab''b''нb''b''иb''b''xb'' b''зb''b' ↩
            'ab''b''пb''b''иb''b''cb''b''ib''b''вb''
        level = option.get('LEVEL', STATUS.DEFAULT)
        log = option.get("LOG", True)
        title = message.get('TITLE', '')
        mess = message.get('SHORTTEXT', '')
        logtext = message.get('DETAILS', '')

        # b''вb''b''иb''b''кb''b''лb''b''иb''b''кb''b''ab''b''тb''b''иb'' b''фb''b''yb''b' ↩
            'нb''b''кb''b''цb''b''ib''b''юb'' b''дb''b''лb''b''яb'' b''вb''b''иb''b''вb''b' ↩
            'eb''b''дb''b''eb''b''нb''b''нb''b''яb'' b''пb''b''ob''b''вb''b''ib''b''дb''b' ↩
            'ob''b''мb''b''лb''b''eb''b''нb''b''нb''b''яb'' b''вb'' b''pb''b''яb''b''дb''b' ↩
            'кb''b''yb'' b''cb''b''тb''b''ab''b''нb''b''yb'':
        self.add_status(mess, level, noLog=True)

        # b''зb''b''ab''b''пb''b''иb''b''тb'' b''нb''b''ab'' b''ob''b''нb''b''ob''b''вb''b' ↩
            'лb''b''eb''b''нb''b''нb''b''яb'' b''фb''b''ab''b''йb''b''лb''b''yb'' b''жb''b' ↩
            'yb''b''pb''b''нb''b''ab''b''лb''b''yb''
        if log:
            STATUS.emit('update-machine-log', "{}\n{}".format(title, logtext), 'TIME')
```

error

(returns integer, string) - призначено для надсилання, коли повідомляється про помилку.
ціле число представляє тип помилки. ERROR, TEXT або DISPLAY
рядок – це фактичне повідомлення про помилку.
Це залежить від використаних віджетів/бібліотек.

general

(returns Python dict) - призначений для відправлення, коли необхідно надіслати повідомлення, яке не охоплюється більш конкретним повідомленням.
Загальне повідомлення слід використовувати якомога рідше, оскільки всі об'єкти, пов'язані з ним, повинні будуть його розібрати.
Для комунікації використовується словник Python.
Словник повинен містити та перевірятися на наявність унікальної пари ідентифікатора та імені ключа:

- ID: *UNIQUE_ID_CODE*
  У словнику зазвичай більше пар ключових слів — це залежить від реалізації.

forced-update
(returns None) - призначений для надсилання, коли потрібно ініціалізувати або довільно оновити об’єкт.
Це залежить від використовуваних віджетів/бібліотек.

progress
(returns integer, Python object) - призначений для надсилання для індикації прогресу програми фільтрації.
Це залежить від використаних віджетів/бібліотек.

following-error
(returns Python list) - повертає список усіх поточних суглобів після помилки.

13.4.4 Функції

Це зручні функції, які зазвичай використовуються в програмуванні.

set_jograte
(float) - LinuxCNC не має внутрішньої концепції швидкості переміщення - кожен графічний інтерфейс має свою власну. Це не завжди зручно.
Ця функція дозволяє встановити швидкість переміщення для всіх об’єктів, підключених до сигналу `jograte-changed`.
За замовчуванням вона становить 15.
GSTAT.set_jog_rate(10) встановить швидкість переміщення на 10 одиниць машини на хвилину і видасть сигнал `jograte-changed`.

get_jograte()
(Nothing) - x = GSTAT.get_jograte() поверне поточне внутрішнє значення jograte (число з плаваючою комою) GSTAT.

set_jograte_angular
(float) -

get_jograte_angular
(Жоден) -

set_jog_increment_angular
(float, string) -

get_jog_increment_angular
(Жоден) -

set_jog_increments
(float, string) -

get_jog_increments
(Жоден) -

is_all_homed
(nothing) - Це поверне поточний стан all_homed (BOOL).

machine_is_on
(nothing) - Це поверне поточний стан машини (BOOL).

estop_is_clear
(nothing) - Це поверне стан Estop (BOOL)

set_tool_touchoff
(tool,axis,value) - Ця команда буде

1. запис поточного режиму,
2. переключитися в режим MDI,
3. викликати команду MDI: G10 L10 P[TOOL] [AXIS] [VALUE],
4. зачекайте, поки це завершиться,
5. викликати G43,
6. зачекайте, поки це завершиться,
7. повернутися до початкового режиму.

set_axis_origin
(axis,value) - Ця команда буде

1. запис поточного режиму,
2. переключитися в режим MDI,
3. викликати команду MDI: G10 L20 P0 [AXIS] [VALUE],
4. зачекайте, поки це завершиться,
5. повернутися до початкового режиму,
6. видавати сигнал «перезавантаження дисплея».

do_jog
(axis_number,direction, distance) - Це дозволить безперервно переміщувати вісь або переміщувати її на заданій відстані.
Для переміщення потрібно бути у відповідному режимі.

check_for_modes
(mode) - Ця функція перевіряє необхідний режим LinuxCNC.
Вона повертає кортеж Python (стан, режим)
режим буде встановлено на режим, у якому знаходиться система
стан буде встановлено на:

- хибність, якщо режим дорівнює 0
- false, якщо машина зайнята
- true, якщо LinuxCNC перебуває в запитуваному режимі
- Немає, якщо можливо змінити, але не в запитуваному режимі

get_current_mode
(nothing) - повертає ціле число: поточний режим LinuxCNC.

set_selected_joint
(integer) - записує номер вибраного з’єднання внутрішньо.
запитує вибір з’єднання, надсилаючи повідомлення
joint-selection-changed.

get_selected_joint
(None) - повертає ціле число, що представляє внутрішній вибраний номер суглоба.

set_selected_axis
(string) - записує вибрану літеру осі внутрішньо.
Запитує вибір осі, надсилаючи повідомлення `axis-selection-changed`.

get_selected_axis
(None) - повертає рядок, що представляє внутрішню вибрану літеру осі.

is_man_mode
(Жоден) -

is_mdi_mode
(Жоден) -

is_auto_mode
(Жоден) -

is_on_and_idle
(Жоден) -

is_auto_running
(Жоден) -

is_auto_paused
(Жоден) -

is_file_loaded
(Жоден) -

is_metric_mode
(Жоден) -

is_spindle_on
(Жоден) -

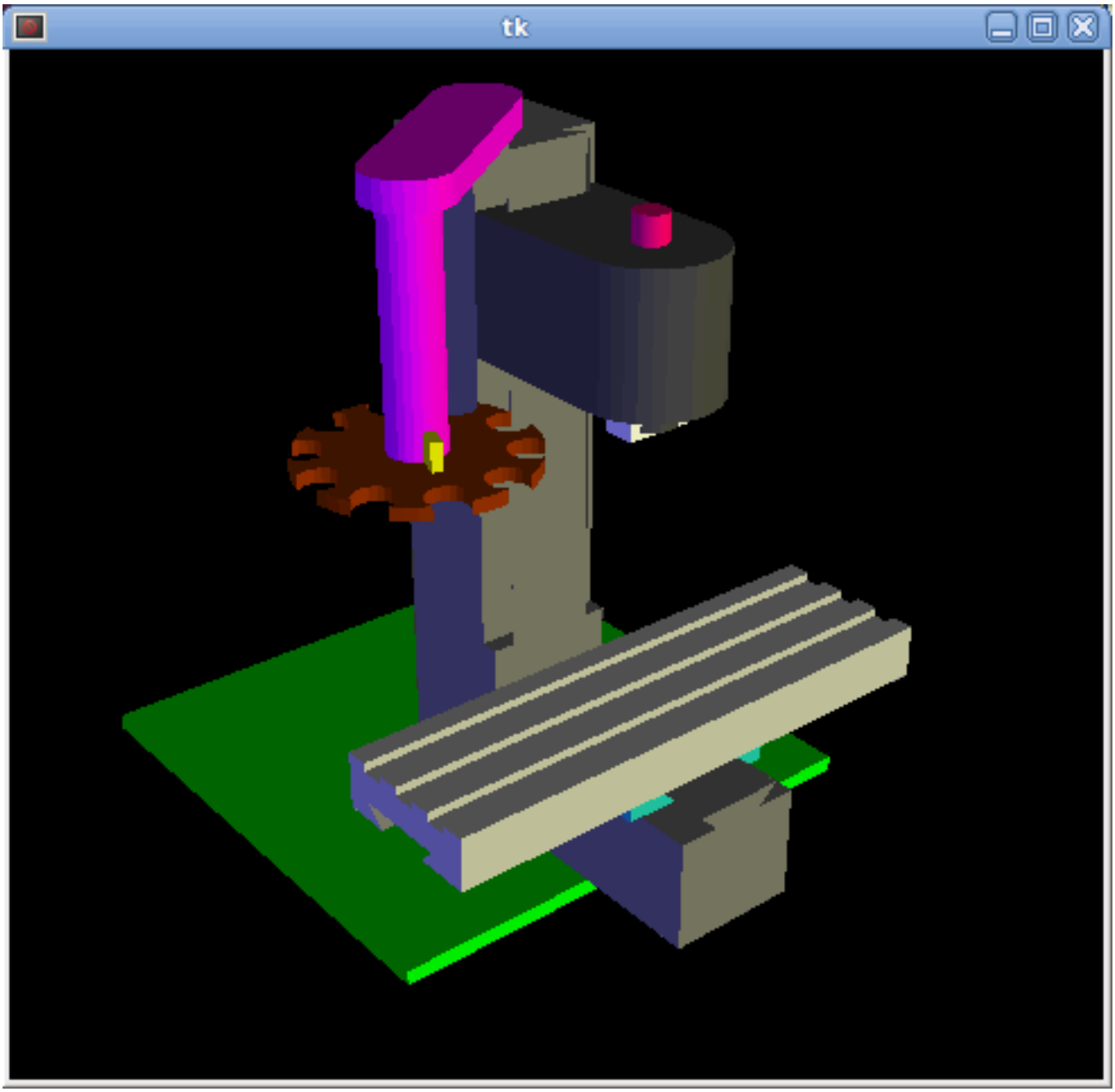
shutdown
(Жоден) -

13.4.5 Відомі проблеми

Деякі статусні точки повідомляються неправильно під час виконання програми, оскільки інтерпретато працює попереду поточної позиції програми, що виконується. Сподіваємося, що ця проблема буде вирішена після об’єднання гілки state-tags.

13.5 Vismach

Vismach — це набір функцій Python, які можна використовувати для створення та анімації моделей машин. Vismach відображає модель у 3D-вікні, а частини моделі анімуються у міру зміни значень відповідних контактів HAL.



Виглядом області перегляду Vismach можна маніпулювати наступним чином:

- **масштабування** за допомогою колеса прокручування або перетягування правою кнопкою миші,
- **pan** перетягуванням лівою кнопкою миші,
- **повернути** перетягуванням середньою кнопкою миші або перетягуванням із утримуваною клавішею Shift.

Модель Vismach має вигляд скрипта Python і може використовувати стандартний синтаксис Python. Це означає, що існує кілька способів написання скрипта, але в прикладах, наведених у цьому документі, я буду використовувати найпростіший і найосновніший з них.

Основна послідовність створення моделі Vismach така

- Створіть контакти HAL, які керують рухом.

- Створіть деталі.
- Визначте, як вони рухаються.
- Об’єднайтеся в рухові групи.

13.5.1 Запустіть скрипт

Для тестування корисно включити *#!/usr/bin/env python3*, щоб файл можна було запустити як скрипт. Перше, що потрібно зробити, це імпортувати необхідні бібліотеки.

```
#!/usr/bin/env python3

from vismach import *
import hal
import math
import sys
```

13.5.2 Створіть контакти HAL.

Контакти HAL створюються за допомогою звичайної бібліотеки Python «hal» і не є специфічними для Vismach. Більш детальну інформацію можна знайти в розділі Створення нереального часу компонентів в Python. Компонент слід створювати з іменем, яке відповідає імені файлу скрипта, а потім до цього компонента додаються контакти HAL. Вони будуть посилатися на їх компонентну ручку і коротке ім’я при використанні для анімації моделі Vismach.

```
c = hal.component("samplegui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

Створить HAL-піни *samplegui.joint0* та *samplegui.joint1*. Під час завантаження моделі Vismach за допомогою *loadusr -W samplegui* функція *c.ready()* повідомляє loadusr про готовність.

13.5.3 Створення деталей

Найпростіше створити геометрію в пакеті CAD і **імпортувати** її в скрипт моделі за допомогою функцій `AsciiSTL()` або `AsciiOBJ()`. Обидві функції можуть приймати один з двох іменованих аргументів: ім’я файлу або необроблені дані:

- `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 ...")`
  Деталі будуть створені в просторі Vismach у тих самих місцях, що й у просторі STL або OBJ. Це означає, що можливо зібрати модель у пакеті CAD.

Або ж деталі можна створювати всередині скрипта моделі з ряду **примітивів фігур**. Багато фігур створюються у початку координат і після створення їх потрібно перемістити в потрібне місце:

- `cylinder = CylinderX(x1, r1, x2, r2) + cylinder = CylinderY(y1, r1, y2, r2) + cylinder = CylinderZ(z1, r1, z2, r2)`
  Створює (за бажанням конічний) циліндр на заданій осі із заданими радіусами у заданих точках на осі.

- `sphere = Sphere(x, y, z, r)`
  Створює сферу радіуса r у точках (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)`
  Створює трикутну пластину між площинами, визначеними двома останніми значеннями, паралельни заданій площині, з вершинами, заданими трьома парами координат.
- `arc = ArcX(x1, x2, r1, r2, a1, a2)`
  Створіть дугоподібну форму.
- `box = Box(x1, y1, z1, x2, y2, z2)`
  Створює прямокутну призму з протилежними кутами у вказаних положеннях та ребрами, паралельними осям XYZ.
- `box = BoxCentered(xw, yw, zw)`
  Створює прямокутник xw на yw на zw з центром у початку координат.
- `box = BoxCenteredXY(xw, yw, z)`
  Створює блок шириною xw / yw та висотою z.

Складені частини можна створювати шляхом **асемблювання** цих примітивів або під час створення, або згодом за допомогою `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 Рухомі частини

Для складання моделі може знадобитися переміщення деталей у просторі Vismach. Також може знадобитися їх переміщення для створення анімації, оскільки вісь обертання анімації створюється в точці початку координат (але переміщується разом із деталлю):

- `part1 = Translate([part1], x, y, z)`
  Перемістіть деталь1 на задані відстані по осях x, y та z.
- `part1 = Rotate([part1], theta, x, y, z)`
  Повернути деталь на кут тета навколо осі між початком координат та координатами x, y, z.

13.5.5 Анімація деталей

Для анімації моделі (керованої значеннями контактів HAL) існують дві функції: «HalTranslate» і «HalRotate». Щоб деталі могли рухатися всередині збірки, перед складанням за допомогою команди «Collection» необхідно визначити їхні рухи HAL. Вісь обертання і вектор перенесення рухаються разом з деталлю, коли вона переміщується скриптом vismach під час складання моделі або коли вона рухається у відповідь на контакти HAL під час анімації моделі:

- `part = HalTranslate([part], comp, ”hal_pin”, xs, ys, zs)`
  Аргументи функції:
  - спочатку *колекцію/частину*, яку можна попередньо створити раніше в скрипті, або ж, якщо бажаєте, створити на цьому етапі, наприклад, `part1 = HalTranslate([Box(....)], ...)`.
  - Компонент *HAL* є наступним аргументом, тобто об’єктом, що повертається командою `comp = hal.component(...)`. Далі йде назва HAL, який буде анімувати рух. Вона повинна відповідати існуючому виводу HAL, що є частиною компонента HAL, створеного раніше в скрипті.

  - Потім слідуйте за шкалами *X, Y, Z*.
    Для декартової машини, створеної в масштабі 1:1, це зазвичай буде 1,0,0 для руху в позитивному напрямку X.
    Однак, якщо файл STL виявився в см, а машина була в дюймах, це можна виправити на цьому етапі, використовуючи 0,3937 (1 см / 2,54 дюйма) як масштаб.

- `part = HalRotate([part], comp, "hal_pin", angle_scale, x, y, z)`
  Ця команда подібна за своєю роботою до HalTranslate, за винятком того, що зазвичай спочатку необхідно перемістити деталь до початку координат, щоб визначити вісь.

  - Ось обертання проходить від точки початку координат до точки, визначеної координатами (x, y, z).
    Коли деталь переміщується назад від початку координат до свого правильного положення, ось обертання можна вважати «вбудованою» в деталь.
  - *Кути повороту* вказані в градусах, тому для поворотного з'єднання з масштабуванням 0-1 вам потрібно буде використовувати шкалу кутів 360.

13.5.6 Збірка моделі.

Щоб деталі рухалися разом, їх потрібно зібрати за допомогою команди Collection(). Важливо зібрати деталі та визначити їхні рухи у правильній послідовності. Наприклад, щоб створити фрезерний верстат із рухомою головкою, обертовим шпинделем та анімованою тяговою штангою, потрібно:

- Створіть основну частину голови.
- Створіть шпиндель у початку координат.
- Дайте визначення обертання.
- Перемістіть головку до шпинделя або шпиндель до головки.
- Створіть планку для витягування.
- Визначте рух тяги.
- Зберіть три частини в головний вузол.
- Визначте рух головки.

У цьому прикладі обертання шпинделя позначається обертанням набору приводних собачок:

```
#b''Cb''b''ob''b''бb''b''ab''b''кb''b''иb'' b''дb''b''лb''b''яb'' b''пb''b''eb''b''pb''b' ↩
    'eb''b''гb''b''ob''b''нb''b''ib''b''вb''
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)

#b''Дb''b''иb''b''шb''b''лb''b''ob''
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)

# b''гb''b''ob''b''лb''b''ob''b''вb''b''кb''b''ab''/b''шb''b''пb''b''иb''b''нb''b''дb''b' ↩
    'eb''b''лb''b''ьb''
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)

# b''бb''b''ab''b''зb''b''ab''
base = AsciiSTL(filename="./base.stl")
base = Color([0.5,0.5,0.5,1],[base])
# b''пb''b''ob''b''кb''b''лb''b''ab''b''cb''b''тb''b''иb'' b''нb''b''ab'' b''нb''b''ьb''b' ↩
    'ob''b''гb''b''ob'' b''гb''b''ob''b''лb''b''ob''b''вb''b''yb''
base = Collection([head, base])
```

Нарешті, потрібно створити єдину колекцію всіх деталей машини, підлоги та робіт (якщо такі є):

- Для *серійної машини* кожна нова частина буде додана до колекції попередньої частини.
- Для *паралельної машини* може бути кілька "базових" частин.

Таким чином, наприклад, у scaragui.py link3 додається до link2, link2 до link1 та link1 до link0, тож остаточна модель створюється за допомогою:

```
model = Collection([link0, floor, table])
```

Тоді як модель VMC з окремими частинами, що рухаються на основі, може мати:

```
model = Collection([base, saddle, head, carousel])
```

13.5.7 Інші функції

- `part = Color([colorspec], [part])`
  Встановлює колір відображення деталі. Зверніть увагу, що на відміну від інших функцій, в цьому випадку визначення деталі йде другим.
  Колір складається з трьох значень RGB і непрозорості. Наприклад, [1,0,0,0.5] для червоного кольору з непрозорістю 50%.
- `myhud = Hud()`
  Створює віконце в графічному інтерфейсі Vismach для відображення таких елементів, як положення осей.
- `tooltip = Capture()`
  Уявіть це як невидиму частину, яку потрібно прикріпити до підказки, щоб відстежувати положення та орієнтацію системи координат інструменту. Насправді це матриця перетворення, яка постійно оновлюється під час руху моделі.
- `work = Capture()`
  Те саме, що й вище, але прикріплено до робочого столу для відстеження системи координат заготовки.
- `main(model, tooltip, work, size=10, hud=0, rotation_vectors=None, lat=0, lon=0)`
  Це команда, яка все це робить, створює дисплей тощо.
  - *model* має бути колекцією, яка містить усі деталі машини.
  - *tooltip* та *work* повинні бути створені за допомогою `Capture()`. Vismach потребує цю інформацію для малювання заднього плану, який є, по суті, позицією підказки, намальованою в системі координат роботи.
    Дивіться `scaragui.py` для прикладу того, як підключити підказку до інструменту, а інструмент до моделі.
  - Для встановлення початкової точки огляду можна використовувати або *rotation_vectors*, або *latitude/longitude*, і це доцільно зробити, оскільки початкова точка огляду за замовчуванням досить незручна безпосередньо згори.
  - *size* встановлює розмір об'єму, що візуалізується у початковому вигляді.
  - *hud* стосується відображення положення осей на лобовому склі.

13.5.8 Базова структура скрипту Vismach.

```
#b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''
b''зb'' vismach b''ib''b''мb''b''пb''b''ob''b''pb''b''тb''b''уb'' *
b''ib''b''мb''b''пb''b''ob''b''pb''b''тb'' hal
#b''cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''кb''b''ob''b''мb''b''пb''b' ↩
    'ob''b''нb''b''eb''b''нb''b''тb'' HAL b''ib'' b''кb''b''ob''b''нb''b''тb''b''ab''b''кb'' ↩
    b''тb''b''иb''
comp = hal.component("compname")
comp.newpin("pin_name", hal.HAL_FLOAT, hal.HAL_IN)
...
#b''cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''пb''b''ib''b''дb''b''лb''b' ↩
    'ob''b''гb''b''уb'', b''ib''b''нb''b''cb''b''тb''b''pb''b''уb''b''мb''b''eb''b''нb''b' ↩
    'тb'' b''ib'' b''pb''b''ob''b''бb''b''ob''b''тb''b''уb''
floor = Box(-50, -50, -3, 50, 50, 0)
work = Capture()
tooltip = Capture()...

#b''Пb''b''ob''b''бb''b''уb''b''дb''b''уb''b''вb''b''ab''b''тb''b''иb'' b''ib'' b''зb''b' ↩
    'ib''b''бb''b''pb''b''ab''b''тb''b''иb'' b''мb''b''ob''b''дb''b''eb''b''лb''b''ьb''
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)
...
#b''Cb''b''тb''b''вb''b''ob''b''pb''b''иb''b''тb''b''иb'' b''мb''b''ob''b''дb''b''eb''b' ↩
    'лb''b''ьb'' b''вb''b''eb''b''pb''b''хb''b''нb''b''ьb''b''ob''b''гb''b''ob'' b''pb''b' ↩
    'ib''b''вb''b''нb''b''яb''
model = Collection([base, saddle, head, carousel])
#b''3b''b''ab''b''пb''b''уb''b''cb''b''тb''b''иb''b''тb''b''иb'' b''вb''b''ib''b''зb''b' ↩
    'уb''b''ab''b''лb''b''ib''b''зb''b''ab''b''цb''b''ib''b''юb''
main(model, tooltip, work, 100, lat=-75, lon=215)
```

Part III

Глосарій, авторське право та історія

Chapter 14

Зворотній бік

Цей посібник знаходиться в процесі розробки. Якщо ви можете допомогти з написанням, редагуванням або графічною підготовкою, будь ласка, зв'яжіться з будь-яким членом команди авторів або приєднайтеся та надішліть електронного листа на адресу emc-users@lists.sourceforge.net.

Авторське право © 2000-2025 LinuxCNC.org

Дозволяється копіювати, розповсюджувати та/або модифікувати цей документ відповідно до умов Ліцензії на вільну документацію GNU, версія 1.1 або будь-яка пізніша версія, опублікована Фундацією вільного програмного забезпечення; без незмінних розділів, текстів на передній обкладинці та текстів на задній обкладинці. Копія ліцензії міститься в розділі під назвою «Ліцензія на вільну документацію GNU».

Якщо ви не знайдете ліцензію, ви можете замовити її копію за адресою:

```
Free Software Foundation, Inc.
51 b''Фb''b''pb''b''ab''b''нb''b''кb''b''лb''b''ib''b''нb''-b''cb''b''тb''b''pb''b''ib''b' ↩
    'тb''
b''Пb'''b''яb''b''тb''b''иb''b''йb'' b''пb''b''ob''b''вb''b''eb''b''pb''b''xb''
b''Бb''b''ob''b''cb''b''тb''b''ob''b''нb'', MA 02110-1301 USA.
```

(Англомовна версія є авторитетною)

LINUX® є зареєстрованою торговою маркою Лінуса Торвальдса в США та інших країнах. Зареєстрована торгова марка Linux® використовується відповідно до субліцензії від LMI, ексклюзивного ліцензіата Лінуса Торвальдса, власника торгової марки на світовому рівні.

Проект LinuxCNC не пов'язаний з Debian®. *Debian* є зареєстрованою торговою маркою, що належить Software in the Public Interest, Inc.

Проект LinuxCNC не пов'язаний з UBUNTU®. *UBUNTU* є зареєстрованою торговою маркою, що належить Canonical Limited.

Chapter 15

Глосарій

Список термінів та їх значення. Деякі терміни мають загальне значення та кілька додаткових значень для користувачів, інсталяторів та розробників.

Гвинт Acme
Тип ходового гвинта, що використовує різьбу Acme. Різьба Acme має дещо нижче тертя і знос, ніж проста трикутна різьба, але кулькові гвинти мають ще нижчі показники. Більшість ручних верстатів використовують ходові гвинти Acme.

Вісь
Одна з рухомих частин верстата, що керується комп'ютером. У типовому вертикальному фрезерному верстаті стіл є віссю X, супорт — віссю Y, а піноль або коліно — віссю Z. Кутові осі, такі як поворотні столи, позначаються літерами A, B і C. Додаткові лінійні осі відносно інструменту позначаються відповідно літерами U, V і W.

ВІСЬ (графічний інтерфейс користувача)
Один із графічних інтерфейсів користувача, доступних для користувачів LinuxCNC. Він відрізняєт
сучасним використанням меню та кнопок миші, а також автоматизацією та приховуванням деяких традиційних елементів керування LinuxCNC. Це єдиний інтерфейс з відкритим кодом, який відображає весь шлях інструменту відразу після відкриття файлу.

GMOCCAPY (графічний інтерфейс користувача)
Графічний інтерфейс користувача, доступний для користувачів LinuxCNC. Він має вигляд і функціональність промислового контролера і може використовуватися з сенсорним екраном, мишею і клавіатурою. Він підтримує вбудовані вкладки і повідомлення користувача, що керуються HAL, і пропонує безліч HAL-бітів, які можна контролювати за допомогою апаратного забезпечення. GMOCCAPY має широкі можливості налаштування.

Зворотна реакція
Кількість «люфту» або втраченого руху, що виникає при зміні напрямку руху в ходовому гвинті або іншій механічній системі приводу. Це може бути наслідком ослаблення гайок на ходових гвинтах, прослизання ременів, провисання кабелів, «накручування» в обертових муфтах та інших місцях, де механічна система не є «натягнутою». Люфт призведе до неточного руху, а в разі руху, спричиненого зовнішніми силами (наприклад, ріжучий інструмент тягне за деталь), результатом може бути поломка ріжучих інструментів. Це може статися через раптове збільшення навантаження на різак, коли деталь тягнеться на відстань люфту ріжучим інструментом.

Компенсація люфту
Будь-яка техніка, яка намагається зменшити вплив люфту, не видаляючи його з механічної системи. Зазвичай це робиться в програмному забезпеченні контролера. Це може виправити кінцеве положення деталі, що рухається, але не вирішує проблеми, пов'язані зі зміною

напрямку руху (наприклад, кругова інтерполяція) та рухом, що викликаний зовнішніми силами (наприклад, ріжучий інструмент, що тягне за деталь).

Кульковий гвинт
Тип ходового гвинта, у якому між гайкою та гвинтом використовуються маленькі загартовані сталеві кульки для зменшення тертя. Кулькові гвинти мають дуже низьке тертя та люфт, але зазвичай досить дорогі.

Кулькова гайка
Спеціальна гайка, призначена для використання з кульковим гвинтом. Вона містить внутрішній канал для рециркуляції кульок від одного кінця гвинта до іншого.

CNC
Комп'ютерне числове управління. Загальний термін, що використовується для позначення комп'ютерного управління верстатами. Замість того, щоб оператор вручну обертав рукоятки для переміщення різального інструменту, CNC використовує комп'ютер і двигуни для переміщенн інструменту на основі програми обробки деталі.

Халкомпіл
Інструмент, що використовується для збирання, компіляції та встановлення компонентів HAL для LinuxCNC.

Конфігурація (n)
Каталог, що містить набір файлів конфігурації. Користувацькі конфігурації зазвичай зберігаються в каталозі home/linuxcnc/configs. Ці файли включають традиційні файли INI та HAL Linux-CNC. Конфігурація також може містити кілька загальних файлів, що описують інструменти, параметри та з'єднання NML.

Конфігурація(v)
Завдання налаштування LinuxCNC таким чином, щоб він відповідав апаратному забезпеченню верстата.

Координатно-вимірювальна машина
Координатно-вимірювальна машина використовується для виконання багатьох точних вимірюван деталей. Ці машини можуть використовуватися для створення CAD-даних для деталей, для яких немає креслень, коли потрібно оцифрувати ручний прототип для виготовлення форми або перевірити точність оброблених або відлитих деталей.

Блоки відображення
Лінійні та кутові одиниці вимірювання, що використовуються для відображення на екрані.

DRO
Цифровий індикатор (DRO) — це система пристроїв для вимірювання положення, прикріплених до супортів верстата, які підключені до цифрового дисплея, що показує поточне положення інструменту відносно деякої базової позиції. DRO дуже популярні на ручних верстатах, оскільки вони вимірюють справжнє положення інструменту без люфту, навіть якщо верстат має дуже вільні гвинти Acme. Деякі цифрові індикатори використовують лінійні квадратурні енкодери для зчитування інформації про положення з верстата, а деякі використовують методи, подібні до резольвера, який постійно перевертається.

EDM
EDM — це метод видалення металу з твердих або важко оброблюваних металів, або в тих випадках, коли обертові інструменти не можуть створити бажану форму економічно вигідним способом. Відмінним прикладом є прямокутні штампи, де потрібні гострі внутрішні кути. Фрезерування не дозволяє отримати гострі внутрішні кути за допомогою інструментів з обмеженим діаметром. Електроіскровий верстат з дротом може створювати внутрішні кути з радіусом, лише трохи більшим за радіус дроту. Електроіскровий верстат з занурювальним електродом може створювати внутрішні кути з радіусом, лише трохи більшим за радіус кута занурювального електрода.

EMC
Удосконалений машинний контролер. Спочатку проект NIST. Перейменовано на LinuxCNC у 2012 році.

EMCIO
Модуль у LinuxCNC, який обробляє загальноприйняті операції введення-виведення, не пов’язані з фактичним рухом осей.

EMCMOT
Модуль у LinuxCNC, який керує фактичним рухом різального інструменту. Він працює як програма реального часу та безпосередньо керує двигунами.

Енкодер
Пристрій для вимірювання положення. Зазвичай це механіко-оптичний пристрій, який видає квадратурний сигнал. Сигнал може бути підрахований спеціальним обладнанням або безпосередньо портом з LinuxCNC.

Годувати
Відносно повільний, контрольований рух інструменту, що використовується під час різання.

Швидкість подачі
Швидкість, з якою відбувається рух різання. В автоматичному режимі або режимі MDI швидкість подачі задається за допомогою слова F. F10 означатиме десять машинних одиниць за хвилину.

Зворотній зв’язок
Метод (наприклад, сигнали квадратурного енкодера), за допомогою якого LinuxCNC отримує інформацію про положення двигунів.

Коригування швидкості подачі
Ручна зміна швидкості руху інструменту під час різання, що контролюється оператором. Часто використовується, щоб оператор міг налаштувати інструменти, які трохи затупилися, або будь-що інше, що вимагає «підкоригування» швидкості подачі.

Число з плаваючою комою
Число, що має десяткову кому. (12.300) У HAL воно відоме як число з комою.

G-код
Загальний термін, що використовується для позначення найпоширенішої мови програмування деталей. Існує кілька діалектів G-коду, LinuxCNC використовує RS274/NGC.

Графічний інтерфейс користувача
Графічний інтерфейс користувача.

Загальне
Тип інтерфейсу, що забезпечує взаємодію між комп’ютером і людиною (у більшості випадків) за допомогою маніпуляцій з піктограмами та іншими елементами (віджетами) на екрані комп’ютера.

LinuxCNC
Програма, яка відображає графічний екран для оператора машини, дозволяючи йому керувати машиною та відповідною програмою управління.

HAL
Рівень абстракції апаратного забезпечення. На найвищому рівні це просто спосіб, що дозволяє завантажувати та з’єднувати між собою низку будівельних блоків для складання складної системи. Багато з цих будівельних блоків є драйверами для апаратних пристроїв. Однак HAL може робити більше, ніж просто налаштовувати драйвери апаратного забезпечення.

Головна сторінка
Певне місце в робочій області верстата, яке використовується для забезпечення узгодження положення інструменту між комп’ютером та фактичним верстатом.

INI-файл
Текстовий файл, що містить більшу частину інформації, яка налаштовує LinuxCNC для конкретної машини.

Екземпляр
Можна мати екземпляр класу або певного об’єкта. Екземпляр - це фактичний об’єкт, створений під час виконання. На жаргоні програмістів об’єкт ”Lassie” - це екземпляр класу ”Dog”.

Спільні координати
Вони визначають кути між окремими шарнірами машини. Див. також Кінематика

Так
Ручне переміщення осі верстата. У режимі покрокового переміщення вісь переміщується на фіксовану величину при кожному натисканні клавіші або переміщується з постійною швидкістю, поки ви утримуєте клавішу. У ручному режимі швидкість покрокового переміщення можна встановити за допомогою графічного інтерфейсу.

простір ядра
Код, що виконується всередині ядра, на відміну від коду, що виконується в просторі користувача. Деякі системи реального часу (наприклад, RTAI) виконують код реального часу в ядрі, а код нереального часу — в просторі користувача, тоді як інші системи реального часу (наприклад, Preempt-RT) виконують як код реального часу, так і код нереального часу в просторі користувача.

Кінематика
Відношення між світовими координатами та координатами суглобів машини. Існує два типи кінематики. Пряма кінематика використовується для обчислення світових координат на основі координат суглобів. Зворотна кінематика використовується для прямо протилежної мети. Зверніть увагу, що кінематика не враховує сили, моменти тощо, що діють на машину. Вона призначена виключно для позиціонування.

Ходовий гвинт
Гвинт, який обертається за допомогою двигуна для переміщення столу або іншої частини машини. Гвинти з різьбою зазвичай бувають кульковими або трапецієподібними, хоча звичайні гвинти з трикутною різьбою можуть використовуватися там, де точність і тривалий термін служби не так важливі, як низька вартість.

Машинні агрегати
Лінійні та кутові одиниці вимірювання, що використовуються для конфігурації машини. Ці одиниці вимірювання вказані та використовуються у файлі INI. Виводи та параметри HAL також зазвичай вказані в одиницях вимірювання машини.

MDI
Ручне введення даних. Це режим роботи, в якому контролер виконує окремі рядки G-коду, які вводить оператор.

NIST
Національний інститут стандартів і технологій. Агентство Міністерства торгівлі США.

NML
Neutral Message Language забезпечує механізм обробки декількох типів повідомлень в одному буфері, а також спрощує інтерфейс для кодування та декодування буферів у нейтральному форматі та механізм конфігурації.

Зміщення
Довільна величина, що додається до значення чогось, щоб воно дорівнювало бажаному значенню. Наприклад, програми G-коду часто пишуться навколо якоїсь зручної точки, такої як X0, Y0. Зсуви кріплення можуть використовуватися для зміщення фактичної точки виконання цієї програми G-коду, щоб вона відповідала фактичному розташуванню лещат і губок. Зсуви інструменту можуть використовуватися для зміщення «невиправленої» довжини інструменту, щоб вона дорівнювала фактичній довжині цього інструменту.

Програма частини
Опис деталі мовою, зрозумілою для контролера. Для LinuxCNC цією мовою є RS-274/NGC, зазвичай відома як G-код.

Програмні одиниці
Лінійні та кутові одиниці, що використовуються в програмі обробки деталі. Лінійні одиниці програми не обов'язково повинні збігатися з лінійними одиницями верстата. Докладнішу інформацію див. у розділах G20 та G21. Кутові одиниці програми завжди вимірюються в градусах.

Python
Загальнофункціональна мова програмування дуже високого рівня. Використовується в LinuxCNC для графічного інтерфейсу Axis, інструменту конфігурації StepConf та кількох скриптів програмування G-кодом.

Швидкий
Швидкий, можливо, менш точний рух інструменту, який зазвичай використовується для переміщення між розрізами. Якщо інструмент торкається заготовки або кріплення під час швидкого переміщення, це, ймовірно, погано!

Швидкий темп
Швидкість, з якою відбувається швидкий рух. В автоматичному або MDI-режимі швидкість швидкого переміщення зазвичай дорівнює максимальній швидкості верстата. Часто бажано обмежувати швидкість швидкого переміщення під час першого тестування програми G-коду.

У режимі реального часу
Програмне забезпечення, призначене для дотримання дуже суворих термінів. У Linux, щоб відповідати цим вимогам, необхідно встановити ядро реального часу, таке як RTAI або Preempt-RT, і скомпілювати програмне забезпечення LinuxCNC для роботи в спеціальному середовищі реального часу. Програмне забезпечення реального часу може працювати в ядрі або в просторі користувача, залежно від можливостей, що надаються системою.

RTAI
Інтерфейс застосунків реального часу, див. https://www.rtai.org/, розширення реального часу для Linux, які LinuxCNC може використовувати для досягнення продуктивності в реальному часі.

RTLINUX
Див. https://en.wikipedia.org/wiki/RTLinux, старіше розширення для Linux, яке LinuxCNC використовувало для досягнення продуктивності в режимі реального часу. Застаріле, замінене RTAI.

RTAPI
Портативний інтерфейс для операційних систем реального часу, включаючи RTAI та POSIX pthreads з розширеннями реального часу.

RS-274/NGC
Офіційна назва мови, що використовується програмами обробки деталей LinuxCNC.

Серводвигун
Зазвичай, це будь-який двигун, що використовується зі зворотним зв'язком на основі вимірювання помилок для корекції положення виконавчого механізму. Також це двигун, спеціально розроблений для забезпечення покращеної продуктивності в таких застосуваннях.

Сервоцикл
Контур керування, що використовується для керування положенням або швидкістю двигуна, оснащеного пристроєм зворотного зв'язку.

Знакове ціле число
Ціле число, яке може мати додатний або від'ємний знак. У HAL це зазвичай s32, але може бути також s64.

Шпиндель
Частина верстата, яка обертається для різання. На фрезерному або свердлильному верстаті шпиндель утримує ріжучий інструмент. На токарному верстаті шпиндель утримує заготовку.

Коригування швидкості шпинделя
Ручне, кероване оператором змінення швидкості обертання інструменту під час різання. Часто використовується, щоб оператор міг регулювати вібрацію, спричинену зубцями різака. Функція Spindle Speed Override передбачає, що програмне забезпечення LinuxCNC налаштовано для керування швидкістю шпинделя.

StepConf
Майстер налаштування LinuxCNC. Він здатний обробляти багато машин, що працюють на основі команд руху «крок і напрямок». Він записує повну конфігурацію після того, як користувач відповість на кілька запитань про комп'ютер і машину, на яких буде працювати LinuxCNC.

Крековий двигун
Тип двигуна, який обертається фіксованими кроками. Підраховуючи кроки, можна визначити, наскільки обернувся двигун. Якщо навантаження перевищує крутний момент двигуна, він пропускає один або кілька кроків, що призводить до помилок позиціонування.

TASK
Модуль у LinuxCNC, який координує загальне виконання та інтерпретує програму обробки деталей.

Tcl/Tk
Мова сценаріїв та набір інструментів графічних віджетів, за допомогою яких було написано кілька графічних інтерфейсів та майстрів вибору LinuxCNC.

Траверсний рух
Рух по прямій лінії від початкової точки до кінцевої точки.

Одиниці
Див. розділ «Машинні одиниці», «Одиниці відображення» або «Одиниці програмування».

Беззнакове ціле число
Ціле число без знака. У HAL це зазвичай u32, але може бути також u64.

Світові координати
Це абсолютна система відліку. Вона задає координати у вигляді фіксованої системи відліку, прикріпленої до певної точки (зазвичай до основи) верстата.

Chapter 16

Авторське право

16.1 Юридичний відділ

Переклади цього файлу, надані у дереві вихідних кодів, не мають юридичної сили.

16.1.1 Умови авторського права

Авторське право (c) 2000-2022 LinuxCNC.org

Дозволяється копіювати, розповсюджувати та/або модифікувати цей документ відповідно до умов Ліцензії на вільну документацію GNU, версія 1.1 або будь-яка пізніша версія, опублікована Фундацією вільного програмного забезпечення; без незмінних розділів, текстів на передній обкладинці та текстів на задній обкладинці. Копія ліцензії міститься в розділі під назвою «Ліцензія на вільну документацію GNU».

16.1.2 Ліцензія GNU Free Documentation

Ліцензія GNU Free Documentation, версія 1.1, березень 2000 р.

Авторські права © 2000 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Кожному дозволено копіювати та розповсюджувати дослівні копії цього ліцензійного документа, але змінювати його не дозволяється.

0. ПРЕАМБУЛА

Метою цієї Ліцензії є створення «вільного» в сенсі свободи посібника, підручника або іншого письмового документа: забезпечення кожному ефективної свободи копіювати та поширювати його, з модифікаціями або без них, як у комерційних, так і в некомерційних цілях. По-друге, ця Ліцензія зберігає для автора та видавця можливість отримати визнання за свою роботу, не несучи при цьому відповідальності за модифікації, внесені іншими особами.

Ця ліцензія є різновидом «копілефту», що означає, що похідні роботи від цього документа також повинні бути вільними в тому ж сенсі. Вона доповнює Загальну публічну ліцензію GNU, яка є ліцензією копілефту, розробленою для вільного програмного забезпечення.

Ми розробили цю Ліцензію для використання в посібниках до вільного програмного забезпечення, оскільки вільне програмне забезпечення потребує вільної документації: вільна програма повинна супроводжуватися посібниками, що надають ті самі свободи, що й програмне забезпечення. Але ця Ліцензія не обмежується посібниками до програмного забезпечення; її можна використовувати для будь-яких текстових творів, незалежно від теми чи того, чи вони опубліковані у вигляді

друкованої книги. Ми рекомендуємо цю Ліцензію переважно для творів, метою яких є навчання чи довідкова інформація.

1. ЗАСТОСУВАННЯ ТА ВИЗНАЧЕННЯ

Ця Ліцензія застосовується до будь-якого посібника або іншої роботи, що містить повідомлення від власника авторських прав про те, що вона може розповсюджуватися на умовах цієї Ліцензії. Термін «Документ», що використовується нижче, означає будь-який такий посібник або роботу. Будь-який член громадськості є ліцензіатом і позначається як «ви».

«Змінена версія» Документа означає будь-який твір, що містить Документ або його частину, скопійований дослівно або зі змінами та/або перекладений іншою мовою.

«Додаткова секція» — це названий додаток або передмова до Документа, яка стосується виключно відносин видавців або авторів Документа до загальної теми Документа (або до пов’язаних з нею питань) і не містить нічого, що могло б безпосередньо відноситися до цієї загальної теми. (Наприклад, якщо Документ є частково підручником з математики, Додаткова секція не може містити пояснень з математики.) Зв’язок може бути історичним зв’язком з темою або пов’язаними питаннями, або юридичною, комерційною, філософською, етичною чи політичною позицією щодо них.

«Незмінні розділи» – це певні Додаткові розділи, назви яких позначені як назви Незмінних розділів у повідомленні про те, що Документ випущено за цією Ліцензією.

«Тексти обкладинки» – це певні короткі уривки тексту, що позначені як Тексти передньої або задньої обкладинки у повідомленні про те, що Документ випущено за цією Ліцензією.

«Прозора» копія документа означає машиночитану копію, представлену у форматі, специфікація якого є загальнодоступною, вміст якої можна переглядати та редагувати безпосередньо та просто за допомогою загальних текстових редакторів або (для зображень, що складаються з пікселів) загальних програм для малювання або (для малюнків) деяких широко доступних редакторів малюнків, і яка підходить для введення в текстові форматувальники або для автоматичного перекладу в різні формати, придатні для введення в текстові форматувальники. Копія, створена в іншому прозорому форматі файлу, розмітка якого була розроблена для запобігання або перешкоджанн подальшим змінам з боку читачів, не є прозорою. Копія, яка не є «прозорою», називається «непрозорою».

Прикладами відповідних форматів для прозорих копій є простий ASCII без розмітки, формат введення Texinfo, формат введення LaTeX, SGML або XML з використанням загальнодоступного DTD, а також простий HTML, що відповідає стандартам і призначений для модифікації людиною. До непрозорих форматів належать PostScript, PDF, власницькі формати, які можна читати та редагувати лише за допомогою власницьких текстових процесорів, SGML або XML, для яких DTD та/або інструменти обробки зазвичай недоступні, а також HTML, згенерований машиною, який створюється деякими текстовими процесорами виключно для виведення на екран.

«Титульна сторінка» означає, для друкованої книги, саму титульну сторінку, а також наступні сторінки, необхідні для розміщення у читабельному вигляді інформації, яка повинна бути вказана на титульній сторінці відповідно до вимог цієї Ліцензії. Для творів у форматах, які не мають титульної сторінки як такої, «Титульна сторінка» означає текст, розташований поруч із найпомітніши зображенням назви твору, що передує початку основного тексту.

2. ДОСЛОВНЕ КОПІЮВАННЯ

Ви можете копіювати та розповсюджувати Документ на будь-якому носії, як у комерційних, так і в некомерційних цілях, за умови, що ця Ліцензія, повідомлення про авторські права та повідомлення про ліцензію, яке вказує, що ця Ліцензія застосовується до Документа, відтворюються у всіх копіях, і що ви не додаєте жодних інших умов до умов цієї Ліцензії. Ви не можете використовувати технічні засоби для перешкоджання або контролю читання або подальшого копіювання копій, які ви створюєте або розповсюджуєте. Однак ви можете приймати компенсацію в обмін на копії. Якщо ви розповсюджуєте достатньо велику кількість копій, ви також повинні дотримуватися умов, викладених у розділі 3.

Ви також можете позичати копії за тих самих умов, що зазначені вище, і можете публічно демонструвати копії.

3. КОПІЮВАННЯ У КІЛЬКОСТІ

Якщо ви публікуєте друковані копії Документа, кількість яких перевищує 100, і ліцензійна угода Документа вимагає наявності текстів на обкладинці, ви повинні додати до копій обкладинки, на яких чітко і розбірливо надруковані всі ці тексти: тексти на передній обкладинці та тексти на задній обкладинці. Обидві обкладинки також повинні чітко і розбірливо ідентифікувати вас як видавця цих копій. На передній обкладинці має бути вказано повну назву, всі слова якої мають бути однаково помітними та видимими. Ви можете додати на обкладинки додаткові матеріали. Копіювання зі змінами, що обмежуються обкладинками, за умови, що вони зберігають назву Документа та відповідають цим умовам, може розглядатися як дослівне копіювання в інших аспектах.

Якщо необхідні тексти для будь-якої з обкладинок занадто об'ємні для розбірливості, вам слід розмістити перші зі списку (стільки, скільки достатньо місця) на фактичній обкладинці, а решту продовжити на сусідній сторінці.

Якщо ви публікуєте або розповсюджуєте непрозорі копії Документа, кількість яких перевищує 100, ви повинні або додати до кожної непрозорої копії машиночитану прозору копію, або вказати в кожній непрозорій копії або разом з нею загальнодоступне місце в комп'ютерній мережі, де міститься повна прозора копія Документа без додаткових матеріалів, яку загальна мережева громадськість може анонімно завантажити безкоштовно, використовуючи загальноприйняті мережеві протоколи. Якщо ви використовуєте останній варіант, ви повинні вжити розумних обережних заходів, коли починаєте розповсюдження непрозорих копій у великій кількості, щоб забезпечити, що ця прозора копія залишатиметься доступною у вказаному місці принаймні протягом одного року після останнього розповсюдження вами непрозорої копії (безпосередньо або через ваших агентів чи роздрібних продавців) цього видання серед громадськості.

Бажано, але не обов'язково, зв'язатися з авторами Документа задовго до розповсюдження великої кількості копій, щоб дати їм можливість надати вам оновлену версію Документа.

4. МОДИФІКАЦІЇ

Ви можете копіювати та розповсюджувати Модифіковану версію Документа на умовах, викладених у розділах 2 та 3 вище, за умови, що ви випускаєте Модифіковану версію саме за цією Ліцензією, причому Модифікована версія виконує роль Документа, таким чином ліцензуючи розповсюдження та модифікацію Модифікованої версії будь-кому, хто володіє її копією. Крім того, ви повинні виконати наступні дії щодо Модифікованої версії:

A. Використовуйте на титульній сторінці (та на обкладинках, якщо такі є) назву, відмінну від назви Документа та від назв попередніх версій (які, якщо такі були, повинні бути перелічені в розділі «Історія» Документа). Ви можете використовувати той самий заголовок, що і попередня версія, якщо оригінальний видавець цієї версії надає дозвіл. B. Вкажіть на титульній сторінці як авторів одну або декількох осіб або організації, відповідальних за авторство модифікацій у Модифікованій версії, разом із щонайменше п'ятьма основними авторами Документа (всіма його основними авторами, якщо їх менше п'яти). C. Вкажіть на титульній сторінці ім'я видавця Модифікованої версії як видавця. D. Збережіть усі повідомлення про авторські права на Документ. E. Додайте відповідне повідомлення про авторські права на ваші модифікації поруч з іншими повідомленнями про авторські права. F. Відразу після повідомлень про авторські права додайте повідомлення про ліцензію, яке надає громадськості дозвіл на використання Модифікованої версії на умовах цієї Ліцензії, у формі, наведеній у Додатку нижче. G. Збережіть у цьому повідомленні про ліцензію повний перелік незмінних розділів та обов'язкових супровідних текстів, наведених у повідомленні про ліцензію Документа. H. Додайте незмінену копію цієї Ліцензії. I. Збережіть розділ під назвою «Історія» та його заголовок і додайте до нього пункт, що містить принаймні назву, рік, нових авторів та видавця Модифікованої версії, як зазначено на титульній сторінці. Якщо в Документі немає розділу під назвою «Історія», створіть його, вказавши назву, рік, авторів та видавця Документа, як зазначено на його титульній сторінці, а потім додайте пункт, що описує Модифіковану версію, як зазначено в попередньому реченні. J. Збережіть мережеве розташування, якщо таке є, вказане в Документі для публічного доступу до прозорої копії Документа, а також мережеві розташування, вказані в Документі для попередніх версій, на яких він базується.

Їх можна розмістити в розділі «Історія». Ви можете опустити мережеве розташування для роботи, яка була опублікована принаймні за чотири роки до самого Документа, або якщо оригінальний видавець версії, на яку вона посилається, дає дозвіл. K. У будь-якому розділі під назвою «Подяки» або «Присвяти» збережіть назву розділу та збережіть у розділі всю суть і тон кожної з подяк та/або присвяти, наведених у ньому. L. Збережіть усі незмінні розділи Документа без змін у їхньому тексті та назвах. Номери розділів або їх еквіваленти не вважаються частиною назв розділів. M. Видалити будь-який розділ під назвою «Рекомендації». Такий розділ не може бути включений до Модифікованої версії. N. Не перейменовувати будь-який існуючий розділ на «Рекомендації» або на назву, що суперечить назві будь-якого Незмінного розділу.

Якщо модифікована версія містить нові розділи передмови або додатки, які кваліфікуються як вторинні розділи і не містять матеріалів, скопійованих з документа, ви можете за власним бажанням позначити деякі або всі ці розділи як незмінні. Для цього додайте їх назви до списку незмінних розділів у ліцензійній записці модифікованої версії. Ці назви повинні відрізнятися від назв будь-яких інших розділів.

Ви можете додати розділ під назвою «Рекомендації», за умови, що він містить виключно рекомендації щодо вашої Модифікованої версії від різних сторін, наприклад, заяви про рецензування колегами або про те, що текст був затверджений організацією як авторитетне визначення стандарту.

Ви можете додати фрагмент тексту довжиною до п’яти слів як текст на передній обкладинці та фрагмент тексту довжиною до 25 слів як текст на задній обкладинці в кінці списку текстів обкладинки в модифікованій версії. Кожна організація може додати (або домовитися про додавання) лише один фрагмент тексту для передньої обкладинки та один фрагмент тексту для задньої обкладинки. Якщо документ вже містить текст для тієї самої обкладинки, раніше доданий вами або за домовленістю з організацією, яку ви представляєте, ви не можете додавати інший, але можете замінити старий за умови отримання явного дозволу від попереднього видавця, який додав старий текст.

Автор(и) та видавець(и) Документа цією Ліцензією не дають дозволу використовувати їхні імена для реклами або для ствердження чи натяку на схвалення будь-якої Зміненої Версії.

5. ОБ’ЄДНАННЯ ДОКУМЕНТІВ

Ви можете поєднувати Документ з іншими документами, опублікованими за цією Ліцензією, на умовах, визначених у розділі 4 вище для модифікованих версій, за умови, що ви включите до поєднання всі Незмінні розділи всіх оригінальних документів, без змін, і перелічите їх усі як Незмінні розділи вашої поєднаної роботи в її ліцензійному повідомленні.

Об’єднана робота повинна містити лише одну копію цієї Ліцензії, а кілька однакових Незмінних розділів можуть бути замінені однією копією. Якщо є кілька Незмінних розділів з однаковою назвою, але різним змістом, зробіть назву кожного такого розділу унікальною, додавши в кінці, в дужках, ім’я оригінального автора або видавця цього розділу, якщо воно відоме, або ж унікальний номер. Зробіть такі самі зміни до назв розділів у списку незмінних розділів у повідомленні про ліцензію об’єднаної роботи.

У комбінації ви повинні об’єднати всі розділи під назвою «Історія» з різних оригінальних документів, утворивши один розділ під назвою «Історія»; аналогічно об’єднайте всі розділи під назвою «Подяки» та всі розділи під назвою «Присвяти». Ви повинні видалити всі розділи під назвою «Рекомендації.»

6. ЗБІРКИ ДОКУМЕНТІВ

Ви можете створити збірку, що складається з Документа та інших документів, опублікованих за цією Ліцензією, і замінити окремі копії цієї Ліцензії в різних документах єдиною копією, що входить до збірки, за умови, що ви дотримуєтеся правил цієї Ліцензії щодо дослівного копіювання кожного з документів у всіх інших аспектах.

Ви можете витягти окремий документ з такої колекції та розповсюджувати його окремо відповідно до цієї Ліцензії, за умови, що ви вставите копію цієї Ліцензії до витягнутого документа та дотримуватим цієї Ліцензії в усіх інших аспектах, що стосуються дослівного копіювання цього документа.

7. АГРЕГАЦІЯ З НЕЗАЛЕЖНИМИ РОБОТАМИ

Компіляція Документа або його похідних з іншими окремими та незалежними документами або творами, у або на носії інформації або розподільному носії, в цілому не вважається Модифікованою версією Документа, за умови, що на компіляцію не заявлено авторських прав. Така компіляція називається «агрегатом», і ця Ліцензія не поширюється на інші самостійні твори, скомпільовані разом з Документом, з огляду на те, що вони скомпільовані таким чином, якщо вони самі по собі не є похідними творами Документа.

Якщо вимога щодо тексту на обкладинці, викладена в розділі 3, застосовується до цих копій Документа, то якщо Документ становить менше чверті від усього сукупного обсягу, тексти на обкладинці Документа можуть бути розміщені на обкладинках, що оточують лише Документ у сукупному обсязі. В іншому випадку вони повинні бути розміщені на обкладинках, що оточують весь сукупний обсяг.

8. ПЕРЕКЛАД

Переклад вважається різновидом модифікації, тому ви можете розповсюджувати переклади Документа на умовах, викладених у розділі 4. Заміна незмінних розділів перекладами вимагає спеціального дозволу від їхніх власників авторських прав, але ви можете додавати переклади деяких або всіх незмінних розділів на додаток до оригінальних версій цих незмінних розділів. Ви можете включати переклад цієї Ліцензії за умови, що ви також включаєте оригінальну англійську версію цієї Ліцензії. У разі розбіжностей між перекладом та оригінальною англійською версією цієї Ліцензії, перевагу має оригінальна англійська версія.

9. ПРИПИНЕННЯ

Ви не маєте права копіювати, модифікувати, субліцензувати або розповсюджувати Документ, за винятком випадків, прямо передбачених цією Ліцензією. Будь-яка інша спроба копіювати, модифікувати, субліцензувати або розповсюджувати Документ є недійсною і автоматично припиняє ваші права за цією Ліцензією. Однак сторони, які отримали від вас копії або права за цією Ліцензією, не втрачають своїх ліцензій, якщо вони повністю дотримуються її умов.

10. МАЙБУТНІ ПЕРЕГЛЯДИ ЦІЄЇ ЛІЦЕНЗІЇ

Фонд вільного програмного забезпечення може час від часу публікувати нові, переглянуті версії Ліцензії вільної документації GNU. Такі нові версії будуть схожими за духом до поточної версії, але можуть відрізнятися в деталях для вирішення нових проблем або питань. Дивіться https://www.gnu.org/copyleft/.

Кожній версії Ліцензії присвоюється ідентифікаційний номер версії. Якщо в Документі зазначено, що до нього застосовується певна пронумерована версія цієї Ліцензії «або будь-яка пізніша версія», ви маєте можливість дотримуватися умов і положень або зазначеної версії, або будь-якої пізнішої версії, опублікованої (не як проект) Фондом вільного програмного забезпечення. Якщо в Документі не вказано номер версії цієї Ліцензії, ви можете вибрати будь-яку версію, коли-небудь опубліковану (не як проект) Фондом вільного програмного забезпечення.

ДОДАТОК: Як використовувати цю Ліцензію для ваших документів

Щоб використовувати цю Ліцензію в документі, який ви написали, додайте копію Ліцензії до документа та розмістіть наступні повідомлення про авторські права та ліцензію одразу після титульної сторінки:

Авторські права (c) РІК ВАШЕ ІМ’Я. Дозволяється копіювати, розповсюджувати та/або модифікувати цей документ відповідно до умов Ліцензії на вільну документацію GNU, версія 1.1 або будь-яка пізніша версія, опублікована Фундацією вільного програмного забезпечення; з незмінними розділами, переліком яких є ПЕРЕЛІК ЇХНІХ НАЗВ, з текстами на передній обкладинці, переліком яких є ПЕРЕЛІК, та з текстами на задній обкладинці, переліком яких є ПЕРЕЛІК. Копія ліцензії міститься в розділі під назвою «Ліцензія GNU на вільну документацію».

Якщо у вас немає незмінних розділів, напишіть «без незмінних розділів» замість того, щоб вказувати, які саме розділи є незмінними. Якщо у вас немає текстів на передній обкладинці, напишіть «без текстів на передній обкладинці» замість «тексти на передній обкладинці: СПИСОК»; те саме стосується текстів на задній обкладинці..

Якщо ваш документ містить нетривіальні приклади програмного коду, ми рекомендуємо опублікувати ці приклади паралельно під обраною вами ліцензією на вільне програмне забезпечення, такою як Загальна публічна ліцензія GNU, щоб дозволити їх використання у вільному програмному забезпеченні.

Chapter 17

Історія LinuxCNC

17.1 Походження

EMC (Enhanced Machine Controller, розширений контролер обладнання) був створений Національним інститутом стандартів і технологій (https://www.nist.gov/index.html, NIST), який є агентством Міністерс торгівлі уряду США.

NIST вперше зацікавився створенням пакету програмного забезпечення для управління рухом як тестової платформи для концепцій і стандартів. Раннє спонсорство від General Motors призвело до адаптації початкової версії EMC з використанням інтелектуальних плат управління PMAC, що працювали під «реальною» версією Windows NT і керували великим фрезерним верстатом.

Як і вимагається від усіх «робочих продуктів» співробітників федерального уряду США, отримане програмне забезпечення та звіт про нього повинні бути у відкритому доступі, і звіт про нього був належним чином опублікований, в тому числі в Інтернеті. Саме там Метт Шейвер відкрив EMC. Він зв'язався з NIST і вступив у переговори з Фредом Проктором про адаптацію коду для використання в управлінні менш дорогим обладнанням, яке буде використовуватися для модернізації та заміни застарілих або просто непрацюючих систем CNC. NIST зацікавився, оскільки вони теж хотіли щось менш дороге. Для започаткування спільної роботи було укладено офіційну угоду, яка гарантувала, що отриманий код і дизайн залишаться у відкритому доступі.

Спочатку розглядалося питання заміни дорогої і примхливої системи Windows NT, що працювала в режимі реального часу. Було запропоновано випробувати відносно нове (на той час) розширення операційної системи Linux для роботи в режимі реального часу. Ця ідея була успішно реалізована. Наступним питанням було вирішення проблеми дорогих інтелектуальних плат управління рухом. На той час обчислювальна потужність ПК вважалася достатньою для безпосереднього управління руховими процедурами. Швидкий пошук доступного обладнання призвів до вибору інтерфейсної плати «Servo-To-Go» як першої платформи, що дозволяла ПК безпосередньо керувати двигунами. Програмне забезпечення для планування траєкторії та управління PID-контуром було додано до існуючого користувацького інтерфейсу та інтерпретатора RS274. Мет успішно використав цю версію для модернізації декількох машин з несправними системами управління, і це стало системою EMC, яка вперше привернула увагу зовнішнього світу. Згадка про EMC у новинній групі rec.crafts.metalworking USENET призвела до того, що перші користувачі, такі як Джон Елсон, почали створювати системи, щоб скористатися перевагами EMC.

NIST створив список розсилки для людей, зацікавлених в EMC. Згодом інші люди, не пов'язані з NIST, також зацікавилися вдосконаленням EMC. Багато хто просив або кодував невеликі вдосконаленн коду. Рей Генрі хотів удосконалити користувальницький інтерфейс. Оскільки Рей не хотів втручатися в код C, на якому був написаний користувальницький інтерфейс, було шукано простіший метод. Фред Проктор з NIST запропонував скриптову мову і написав код для інтерфейсу скриптової мови Tcl/Tk до внутрішніх комунікацій NML EMC. За допомогою цього інструменту Рей написав програму Tcl/Tk, яка на той час стала основним користувацьким інтерфейсом для EMC.

Щодо точки зору NIST, див. статтю «https://web.archive.org/web/20120417094958/https://www.isd.mel.ni
[документ], написану Вільямом Шаклфордом і Фредеріком Проктором, в якій описується історія EMC та її перехід до відкритого коду.

На цей час інтерес до EMC почав суттєво зростати. У міру того, як все більше людей намагалися встановити EMC, труднощі з патчуванням ядра Linux з розширеннями реального часу та компіляцією коду EMC стали очевидними. Було зроблено багато спроб задокументувати процес і написати скрипти, деякі з яких мали помірний успіх. Проблема підбору правильної версії патчів і компіляторів до обраної версії Linux продовжувала виникати. На допомогу прийшов Пол Корнер з BDI (brain dead install) — компакт-диском, з якого можна було встановити повністю робочу систему (Linux, патчі та EMC). Підхід BDI відкрив світ EMC для набагато більшої спільноти користувачів. У міру зростання цієї спільноти список розсилки EMC та архіви коду були перенесені на SourceForge, а також був створений веб-сайт LinuxCNC.

Зі збільшенням кількості користувачів, EMC стала головним об'єктом інтересу на виставках CNC, що проходили в NAMES, а NAMES стала щорічним місцем зустрічі для EMC. Протягом перших кількох років зустрічі відбувалися просто тому, що зацікавлені сторони були присутні на NAMES. У 2003 році спільнота користувачів EMC провела свою першу публічну зустріч. Вона відбулася в понеділок після NAMES у фойє арени, де проходила виставка NAMES. Організація була нечіткою, але народилася ідея апаратного рівня абстракції (HAL) і було запропоновано рух за реструктуризацію коду для полегшення розробки (EMC2).

17.1.1 Зміна імені

Навесні 2011 року до ради директорів LinuxCNC звернулася юридична фірма, що представляє корпорацію EMC (www.emc.com), з приводу використання назв «EMC» та «EMC2» для позначення програмного забезпечення, що пропонується на сайті linuxcnc.org. Корпорація EMC зареєструвала різні торговельні марки, пов'язані з EMC та EMC² (EMC з надрядковою цифрою два).

Після низки переговорів з представником EMC Corporation, остаточним результатом стало те, що, починаючи з наступної великої версії програмного забезпечення, linuxcnc.org припинить ідентифікувати програмне забезпечення за допомогою «emc» або «EMC», або цих термінів, за якими йдуть цифри. В тій мірі, в якій рада директорів LinuxCNC контролює назви, що використовуютьс
для ідентифікації програмного забезпечення, пропонованого на linuxcnc.org, рада погодилася з цим.

В результаті довелося вибрати нову назву для програмного забезпечення. З варіантів, розглянутих правлінням, було досягнуто консенсусу, що «LinuxCNC» є найкращим варіантом, оскільки це була назва нашого веб-сайту протягом багатьох років.

У рамках підготовки до введення нової назви ми отримали субліцензію на торговельну марку LINUX® від Linux Foundation (www.linuxfoundation.org), яка захищає наше право на використання назви LinuxCNC. (LINUX® є зареєстрованою торговельною маркою Лінуса Торвальдса в США та інших країнах.)

Ребрендинг включав веб-сайт linuxcnc.org, канали IRC, а також версії програмного забезпечення та документації, починаючи з версії 2.5.0.

17.1.2 Додаткова інформація

NIST опублікував статтю, в якій описано мову «https://www.nist.gov/node/704046» [RS274NGC] та абстрактний обробний центр, який вона контролює, а також ранню реалізацію EMC. Стаття також доступна за адресою https://linuxcnc.org/files/RS274NGCv3.pdf .

NIST також опублікував статтю про історію EMC та її перехід до відкритого програмного забезпечення (https://www.nist.gov/node/702276) [відкрите програмне забезпечення]. Стаття також доступна за адресою https://linuxcnc.org/files/Use-of-Open-Source-Distribution-for-a-Machine-Tool-Controller.pdf