# Посібник розробника
# V2.10.0-pre0-5837-g4025bf9a5b

# Contents

# Chapter 1

# Вступ


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.

# Chapter 2

# Загальний довідник HAL

## 2.1 Імена сутностей HAL

Всі об’єкти HAL доступні та можуть бути змінені за їхніми іменами, тому дуже важливо документувати імена контактів, сигналів, параметрів тощо. Імена в HAL мають максимальну довжину 41 символ (як визначено в HAL_NAME_LEN в hal.h). Багато імен будуть представлені в загальній формі, з форматованим текстом *<like-this>*, що представляє поля різних значень.

Коли контакти, сигнали або параметри описуються вперше, перед їхньою назвою в дужках вказується їхній тип (*float*), а після — короткий опис. Типові визначення контактів виглядають так, як у цих прикладах:

**(bit) parport.<portnum>.pin-<pinnum>-in**
Вивід HAL, пов’язаний з фізичним вхідним виводом *<номер виводу>* роз’єму db25.

**(float) pid.<loopnum>.output**
Вихід контуру PID

Іноді може використовуватися скорочена версія назви, наприклад, другий контакт вище можна просто викликати за допомогою *.output*, коли це можна зробити без плутанини.

## 2.2 Загальні правила іменування HAL

Послідовні правила іменування значно полегшили б використання HAL. Наприклад, якби кожен драйвер енкодера мав однаковий набір контактів і називався однаково, то було б легко перейти з одного типу драйвера енкодера на інший. На жаль, як і багато інших проектів з відкритим кодом, HAL є поєднанням елементів, які були розроблені, та елементів, які просто еволюціонували. В результаті існує багато невідповідностей. У цьому розділі ми спробуємо вирішити цю проблему, визначивши деякі правила, але, ймовірно, знадобиться деякий час, перш ніж усі модулі будуть перетворені відповідно до них.

Halcmd та інші низькорівневі утиліти HAL розглядають імена HAL як окремі об’єкти без внутрішньої структури. Однак більшість модулів мають певну неявну структуру. Наприклад, плата має кілька функціональних блоків, кожен блок може мати кілька каналів, а кожен канал має один або кілька виводів. Це створює структуру, схожу на дерево каталогів. Хоча halcmd не розпізнає структуру дерева, правильний вибір правил іменування дозволить йому групувати пов’язані елементи разом (оскільки він сортує імена). Крім того, інструменти вищого рівня можуть бути розроблені для розпізнавання такої структури, якщо імена надають необхідну інформацію. Для цього всі компоненти HAL повинні дотримуватися таких правил:

- Крапки (“.”) розділяють рівні ієрархії. Це аналогічно скісну риску (“/”) в імені файлу.
- Дефіси («-») розділяють слова або поля на одному рівні ієрархії.
- Компоненти HAL не повинні використовувати підкреслення або «MixedCase». Примітка: [Підкреслен символи були видалені, але все ще є кілька випадків неправильного поєднання, наприклад *pid.0.Pgain* замість *pid.0.p-gain*.]
- Використовуйте в іменах лише малі літери та цифри.

## 2.3 Правила іменування драйверів обладнання

**Note**
Більшість драйверів у версії 2.0 не дотримуються цих умовних позначень. Цей розділ насправді є керівництвом для майбутніх розробок.

### 2.3.1 Назви контактів/параметрів

Драйвери обладнання повинні використовувати п’ять полів (на трьох рівнях) для формування назви виводу або параметра, як показано нижче:

```
<device-name>.<device-num>.<io-type>.<chan-num>.<specific-name>
```

Окремі поля:

***<device-name>***
Пристрій, з яким призначений драйвер. Найчастіше це інтерфейсна плата певного типу, але є й інші можливості.

***<device-num>***
У комп’ютері можна встановити більше однієї плати сервоприводу, паралельного порту або іншого апаратного пристрою. Номер пристрою ідентифікує конкретний пристрій. Номери пристроїв починаються з 0 і збільшуються на нуль.

***<io-type>***
Більшість пристроїв забезпечують більше одного типу вводу-виводу. Навіть простий паралельний порт має як цифрові входи, так і цифрові виходи. Більш складні плати можуть мати цифрові входи і виходи, лічильники енкодера, генератори імпульсів pwm або step, аналого-цифрові перетворювачі, цифро-аналогові перетворювачі або інші унікальні можливості. Тип вводу-виводу використовується для ідентифікації виду вводу-виводу, з яким пов’язаний вивід або параметр. В ідеалі драйвери, що реалізують один і той самий тип вводу-виводу, навіть для дуже різних пристроїв, повинні забезпечувати однаковий набір контактів і параметрів та ідентичну поведінку. Наприклад, усі цифрові входи повинні поводитися однаково, якщо дивитися зсередини HAL, незалежно від пристрою.

***<chan-num>***
Практично кожен пристрій вводу-виводу має кілька каналів, і номер каналу ідентифікує один з них. Як і номери пристроїв, номери каналів починаються з нуля і збільшуються. Примітка: [Єдиним винятком із правила «номери каналів починаються з нуля» є паралельний порт. Його контакти HAL пронумеровані відповідно до номерів контактів на роз’ємі DB-25. Це зручно для підключення, але несумісно з іншими драйверами. Існують суперечки щодо того, чи є це помилкою, чи особливістю.] Якщо встановлено більше одного пристрою,

номери каналів на додаткових пристроях починаються з нуля. Якщо можливий номер каналу, більший за 9, то номери каналів повинні бути двозначними, з нулем на початку для чисел менше 10, щоб зберегти порядок сортування. Деякі модулі мають контакти та/або параметри, які впливають на більше ніж один канал. Наприклад, генератор PWM може мати чотири канали з чотирма незалежними входами «робочого циклу», але один параметр «частоти», який контролює всі чотири канали (через обмеження апаратного забезпечення). Параметр частоти повинен використовувати «0-3» як номер каналу.

***<specific-name>***
Окремий канал вводу/виводу може мати тільки один контакт HAL, але більшість з них мають більше одного. Наприклад, цифровий вхід має два контакти: один відповідає стану фізичного контакту, а інший — його інвертованому стану. Це дозволяє конфігуратору вибирати між активними високими та активними низькими входами. Для більшості типів вводу-виводу існує стандартний набір контактів і параметрів (який називається «канонічним інтерфейсом»), який повинен реалізовувати драйвер. Канонічні інтерфейси описані в розділі Канонічні інтерфейси пристроїв.

Приклади

**motenc.0.encoder.2.position**
Вихід положення третього каналу енкодера на першій платі Motenc.

**stg.0.din.03.in**
Стан четвертого цифрового входу на першій платі Servo-to-Go.

**ppmc.0.pwm.00-03.frequency**
Несуча частота, що використовується для каналів PWM з 0 по 3 на першій платі ppmc від Pico Systems.

### 2.3.2 Назви функцій

Драйвери апаратного забезпечення зазвичай мають лише два типи функцій HAL: функції, що зчитують апаратне забезпечення та оновлюють контакти HAL, та функції, що записують дані в апаратне забезпечення, використовуючи дані з контактів HAL. Вони повинні мати такі назви:

```
<device-name>-<device-num>.<io-type>-<chan-num-range>.read|write
```

**<device-name>**
Те саме, що використовується для контактів та параметрів.

**<device-num>**
Конкретний пристрій, до якого функція отримуватиме доступ.

**<io-type>**
Необов’язково. Функція може мати доступ до всіх входів/виходів на платі або тільки до певного типу. Наприклад, можуть існувати незалежні функції для зчитування лічильників енкодера та зчитування цифрових входів/виходів. Якщо такі незалежні функції існують, поле <io-type> визначає тип входів/виходів, до яких вони мають доступ. Якщо одна функція зчитує всі входи/виходи, що надаються платою, <io-type> не використовується. Примітка: [Примітка для програмістів драйверів: НЕ реалізуйте окремі функції для різних типів входів/виход якщо вони не є перериваними і не можуть працювати в незалежних потоках. Якщо переривання зчитування енкодера, зчитування цифрових входів, а потім відновлення зчитування енкодера спричинить проблеми, реалізуйте одну функцію, яка виконує всі ці дії.]

**<chan-num-range>**
Необов’язково. Використовується лише якщо введення-виведення <io-type> розділене на групи та доступ до нього здійснюється різними функціями.

**читати|писати**
Вказує, чи функція зчитує дані з обладнання, чи записує на нього.

Приклади

**motenc.0.encoder.read**
Зчитує всі енкодери на першій платі Motenc.

**generic8255.0.din.09-15.read**
Зчитує другий 8-бітний порт на першій універсальній платі цифрового вводу/виводу на базі 8255.

**ppmc.0.write**
Записує всі виходи (генератори кроків, PWM, DAC та цифрові) на першу плату ppmc Pico Systems.

# Chapter 3

# Примітки до коду

## 3.1 Цільова аудиторія

Цей документ є збіркою нотаток про внутрішню структуру LinuxCNC. Він в першу чергу цікавий розробникам, проте значна частина інформації, що міститься в ньому, може бути цікавою також системним інтеграторам та іншим особам, які просто цікавляться тим, як працює LinuxCNC. Значна частина цієї інформації на сьогодні є застарілою і ніколи не перевірялася на точність.

## 3.2 Організація

Буде розділ для кожного з основних компонентів LinuxCNC, а також розділ (розділи), що описують, як вони працюють разом. Цей документ знаходиться в стадії розробки, і його структура може змінитися в майбутньому.

## 3.3 Терміни та визначення

- «AXIS» (ОС) — Ось є одним із дев'яти ступенів свободи, що визначають положення інструменту в тривимірному декартовому просторі. Ці дев'ять осей позначаються як X, Y, Z, A, B, C, U, V і W. Лінійні ортогональні координати X, Y і Z визначають положення кінчика інструменту. Кутові координати A, B і C визначають орієнтацію інструменту. Другий набір лінійних ортогональних координат U, V і W дозволяє інструменту рухатися (зазвичай для різання) відносно попередньо зміщених і обернутих осей. На жаль, термін «вісь» іноді також використовується для позначення ступеня свободи самої машини, наприклад, сідла, столу або пінолі фрезерного верстата типу Bridgeport. На верстаті Bridgeport це не викликає плутанини, оскільки рух столу безпосередньо відповідає руху вздовж осі X. Однак плечові та ліктьові суглоби руки робота та лінійні приводи гексапода не відповідають руху вздовж будь-якої декартової осі, і загалом важливо розрізняти декартові осі та ступені свободи машини. У цьому документі останні будуть називатися «суглобами», а не осями. Графічні інтерфейси користувача та деякі інші частини коду не завжди дотримуються цього розрізнення, але внутрішні компоненти контролера руху дотримуються його.
- «JOINT» — склін є однією з рухомих частин машини. Скліни відрізняються від осей, хоча ці два терміни іноді (помилково) використовуються для позначення одного й того самого поняття. У LinuxCNC склін є фізичним об'єктом, який можна переміщати, а не координатою в просторі. Наприклад, піноль, коліно, сідло та стіл фрезерного верстата Bridgeport є склінами. Плече, лікоть і зап'ястя руки робота є суглобами, як і лінійні приводи гексапода. Кожен суглоб має пов'язаний з ним двигун або привід певного типу. Суглоби не обов'язково відповідають осям

X, Y і Z, хоча для машин з тривіальною кінематикою це може бути так. Навіть на цих машинах положення суглоба і положення осі є принципово різними речами. У цьому документі терміни «суглоб» і «вісь» використовуються обережно, щоб відобразити їх чітке значення. На жаль, це не завжди так. Зокрема, графічні інтерфейси для машин з тривіальною кінематикою можуть приховувати або повністю ігнорувати різницю між суглобами і осями. Крім того, у файлі INI термін «вісь» використовується для даних, які точніше було б описати як дані суглоба, такі як масштабування вхідних і вихідних даних тощо.

---

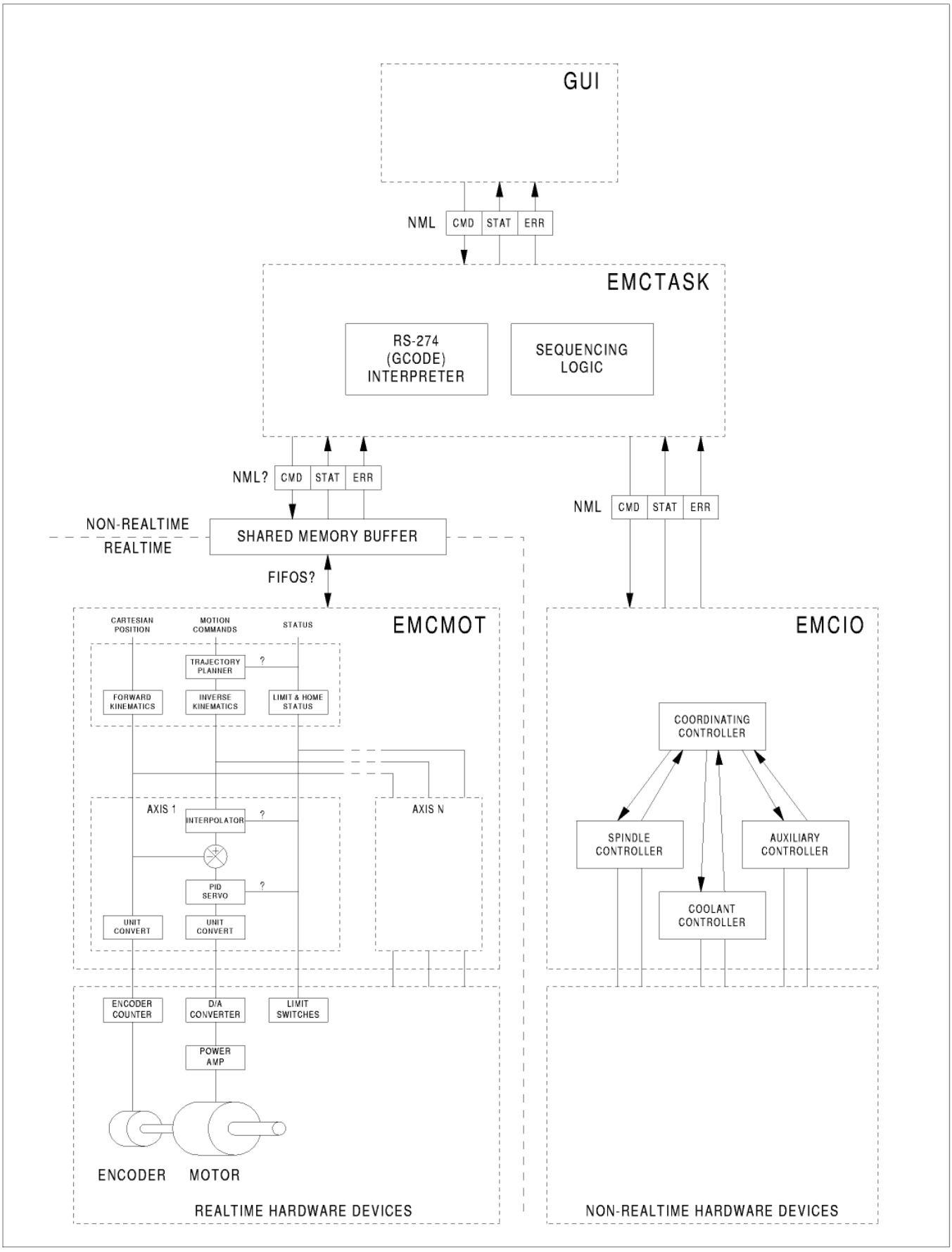
**Note**
Це розрізнення було введено у версії 2.8 LinuxCNC. Файл INI отримав новий розділ [JOINT_<num>]. Багато параметрів, які раніше належали до розділу [AXIS_<letter>], тепер знаходяться в новому розділі. Інші розділи, такі як [KINS], також отримали нові параметри, щоб відповідати цьому. Було надано скрипт оновлення для перетворення старих INI-файлів у нову конфігурацію осей/з'єднань.

---

- «POSE» — Поза — це повністю визначене положення в 3D-декартовому просторі. У контролері руху LinuxCNC під позою ми маємо на увазі структуру EmcPose, що містить шість лінійних координат (X, Y, Z, U, V і W) та три кутові (A, B і C).
- «coord», або скоординований режим, означає, що всі шарніри синхронізовані і рухаються разом відповідно до вказівок коду вищого рівня. Це звичайний режим під час обробки. У скоординованому режимі команди, як правило, подаються в декартовій системі координат, і якщо верстат не є декартовим, команди перетворюються кінематикою для переміщення кожного шарніра в необхідне положення.
- «Вільний» означає, що команди інтерпретуються в просторі суглобів. Він використовується для ручного переміщення (поштовхування) окремих суглобів, хоча це не заважає переміщати кілька суглобів одночасно (я так думаю). Повернення в вихідне положення також здійснюється у вільному режимі; фактично, машини з нетривіальною кінематикою повинні бути повернуті в вихідне положення, перш ніж вони зможуть перейти в режим координації або телеоперації.
- «teleop» — це режим, який вам, ймовірно, знадобиться, якщо ви бігаєте з гексаподом. Команди бігу, реалізовані контролером руху, є спільними бігами, які працюють у вільному режимі. Але якщо ви хочете перемістити гексапод або подібну машину, зокрема, вздовж декартової осі, ви повинні керувати більш ніж одним суглобом. Саме для цього і призначений «teleop».

## 3.4 Огляд архітектури

Архітектура LinuxCNC складається з чотирьох компонентів: контролер руху (EMCMOT), контролер дискретних входів-виходів (EMCIO), координатор завдань (EMCTASK) та кілька текстових і графічних інтерфейсів користувача. Кожен з них буде описаний у цьому документі як з точки зору проектування, так і з точки зору розробників (де знайти необхідні дані, як легко розширити/змінити щось тощо).


GUI
NML
CMD
STAT
ERR
EMCTASK
RS-274
(GCODE)
INTERPRETER
SEQUENCING
LOGIC
NML?
CMD
STAT
ERR
NML
CMD
STAT
ERR
NON-REALTIME
REALTIME
SHARED MEMORY BUFFER
FIFOS?
EMCMOT
CARTESIAN
POSITION
MOTION
COMMANDS
STATUS
TRAJECTORY
PLANNER
FORWARD
KINEMATICS
INVERSE
KINEMATICS
LIMIT & HOME
STATUS
AXIS 1
INTERPOLATOR
AXIS N
PID
SERVO
UNIT
CONVERT
UNIT
CONVERT
ENCODER
COUNTER
D/A
CONVERTER
LIMIT
SWITCHES
POWER
AMP
ENCODER
MOTOR
REALTIME HARDWARE DEVICES
EMCIO
COORDINATING
CONTROLLER
SPINDLE
CONTROLLER
AUXILIARY
CONTROLLER
COOLANT
CONTROLLER
NON-REALTIME HARDWARE DEVICES

### 3.4.1 Архітектура програмного забезпечення LinuxCNC

На найзагальнішому рівні LinuxCNC є ієрархією трьох контролерів: обробника команд рівня завдань та інтерпретатора програм, контролера руху та дискретного контролера вводу-виводу. Дискретний контролер вводу-виводу реалізований як ієрархія контролерів, в даному випадку для підсистем шпинделя, охолодження та допоміжних (наприклад, estop). Контролер завдань координує дії контролерів руху та дискретного вводу-виводу. Їхні дії програмуються у звичайних програмах числового управління «G та M код», які інтерпретуються контролером завдань у повідомлення NML та надсилаються до руху.

## 3.5 Вступ до контролера руху

Контролер руху є компонентом, що працює в режимі реального часу. Він отримує команди управління рухом від компонентів LinuxCNC, що не працюють в режимі реального часу (тобто інтерпретатора G-коду/завдання, графічних інтерфейсів користувача тощо) і виконує ці команди в режимі реального часу. Комунікація з контексту, що не працює в режимі реального часу, до контексту, що працює в режимі реального часу, відбувається за допомогою механізму передачі повідомлень IPC із використанням спільної пам’яті та за допомогою шару абстракції апаратного забезпечення (HAL).

Стан контролера руху стає доступним для решти LinuxCNC через той самий IPC спільної пам’яті для передачі повідомлень та через HAL.

Контролер руху взаємодіє з контролерами двигунів та іншим обладнанням реального та нереального часу за допомогою HAL.

Цей документ передбачає, що читач має базові знання про HAL, і використовує такі терміни, як «контакти HAL», «сигнали HAL» тощо, не пояснюючи їх. Більш детальну інформацію про HAL можна знайти в посібнику HAL. В іншому розділі цього документа ми розглянемо внутрішню структуру HAL, але в цьому розділі ми використовуємо лише API HAL, визначений у src/hal/hal.h.

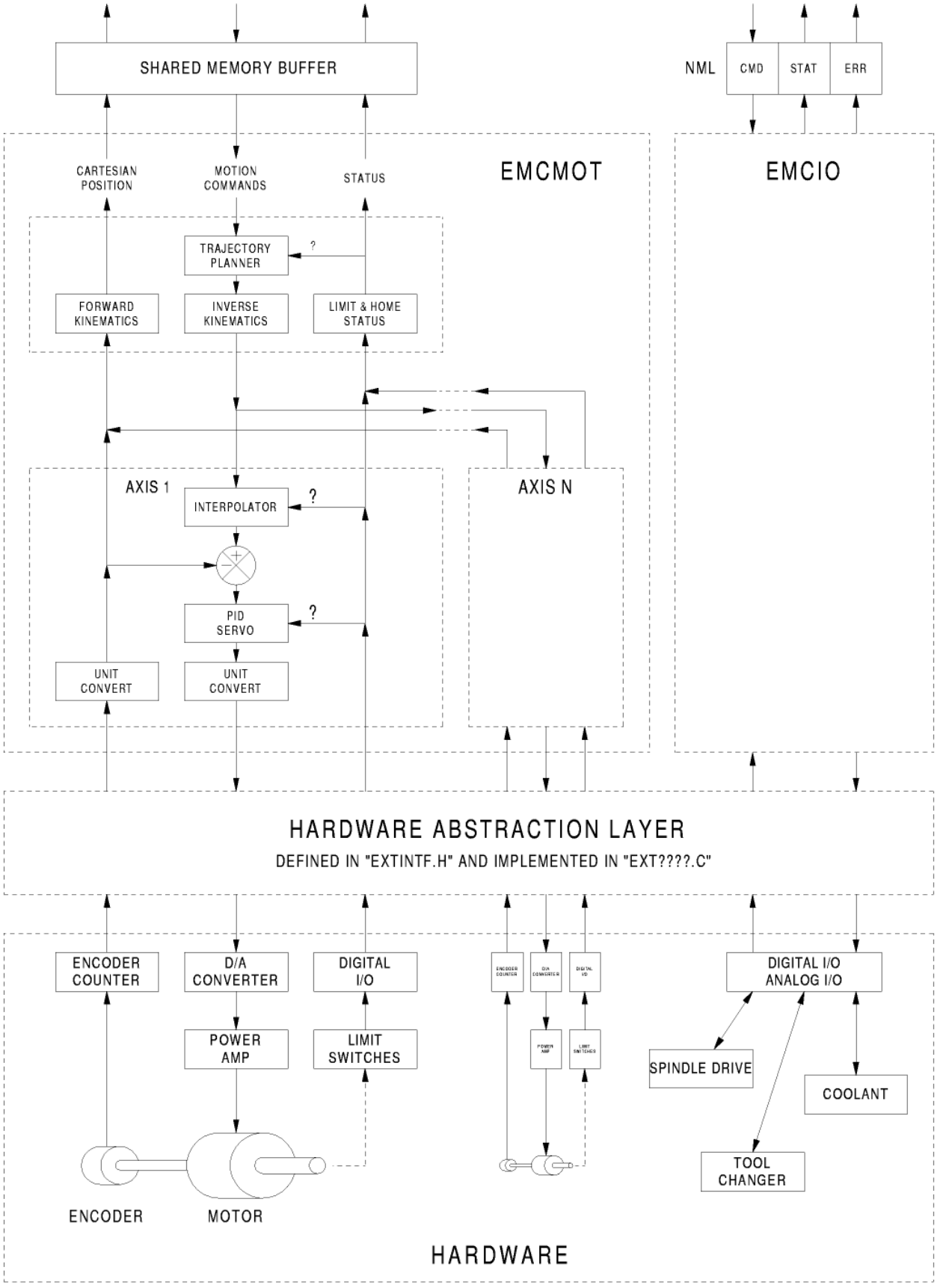
### 3.5.1 Модулі контролера руху

Функції контролера руху в режимі реального часу реалізовані за допомогою модулів реального часу — спільних об’єктів простору користувача для систем Preempt-RT або модулів ядра для деяких реалізацій режиму реального часу в режимі ядра, таких як RTAI:

- *tpmod* - планування траєкторії
- *homemod* - функції самонаведення
- *motmod* - обробляє команди NML та керує обладнанням через HAL
- «кінематичний модуль» — виконує прямі (суглоби-->координати) та зворотні (координати->суглоби) кінематичні розрахунки

LinuxCNC запускається за допомогою скрипта **linuxcnc**, який зчитує файл конфігурації INI і запускає всі необхідні процеси. Для управління рухом в реальному часі скрипт спочатку завантажує модулі tpmod і homemod за замовчуванням, а потім завантажує модулі кінематики і руху відповідно до налаштувань у файлах halfiles, зазначених у файлі INI.

Замість стандартних модулів можна використовувати власні (створені користувачем) модулі повернень до вихідної позиції або планування траєкторії за допомогою налаштувань файлу INI або опцій командного рядка. Власні модулі повинні реалізовувати всі функції, що використовуються стандартними модулями. Для створення власного модуля можна використовувати утиліту halcompile.


SHARED MEMORY BUFFER
NML
CMD
STAT
ERR
CARTESIAN POSITION
MOTION COMMANDS
STATUS
EMCMOT
EMCIO
TRAJECTORY PLANNER
?
FORWARD KINEMATICS
INVERSE KINEMATICS
LIMIT & HOME STATUS
AXIS 1
INTERPOLATOR
?
AXIS N
PID SERVO
?
UNIT CONVERT
UNIT CONVERT
HARDWARE ABSTRACTION LAYER
DEFINED IN "EXTINTF.H" AND IMPLEMENTED IN "EXT????.C"
ENCODER COUNTER
D/A CONVERTER
DIGITAL I/O
POWER AMP
LIMIT SWITCHES
DIGITAL I/O ANALOG I/O
SPINDLE DRIVE
COOLANT
TOOL CHANGER
ENCODER
MOTOR
HARDWARE

## 3.6 Блок-схеми та потік даних

Наступний рисунок є блок-схемою спільного контролера. На кожне з'єднання припадає один спільний контролер. Спільні контролери працюють на рівні, нижчому за кінематику, на якому всі з'єднання є повністю незалежними. Усі дані для з'єднання містяться в єдиній структурі з'єднання. Деякі елементи цієї структури видно на блок-схемі, наприклад coarse_pos, pos_cmd та motor_pos_fb.

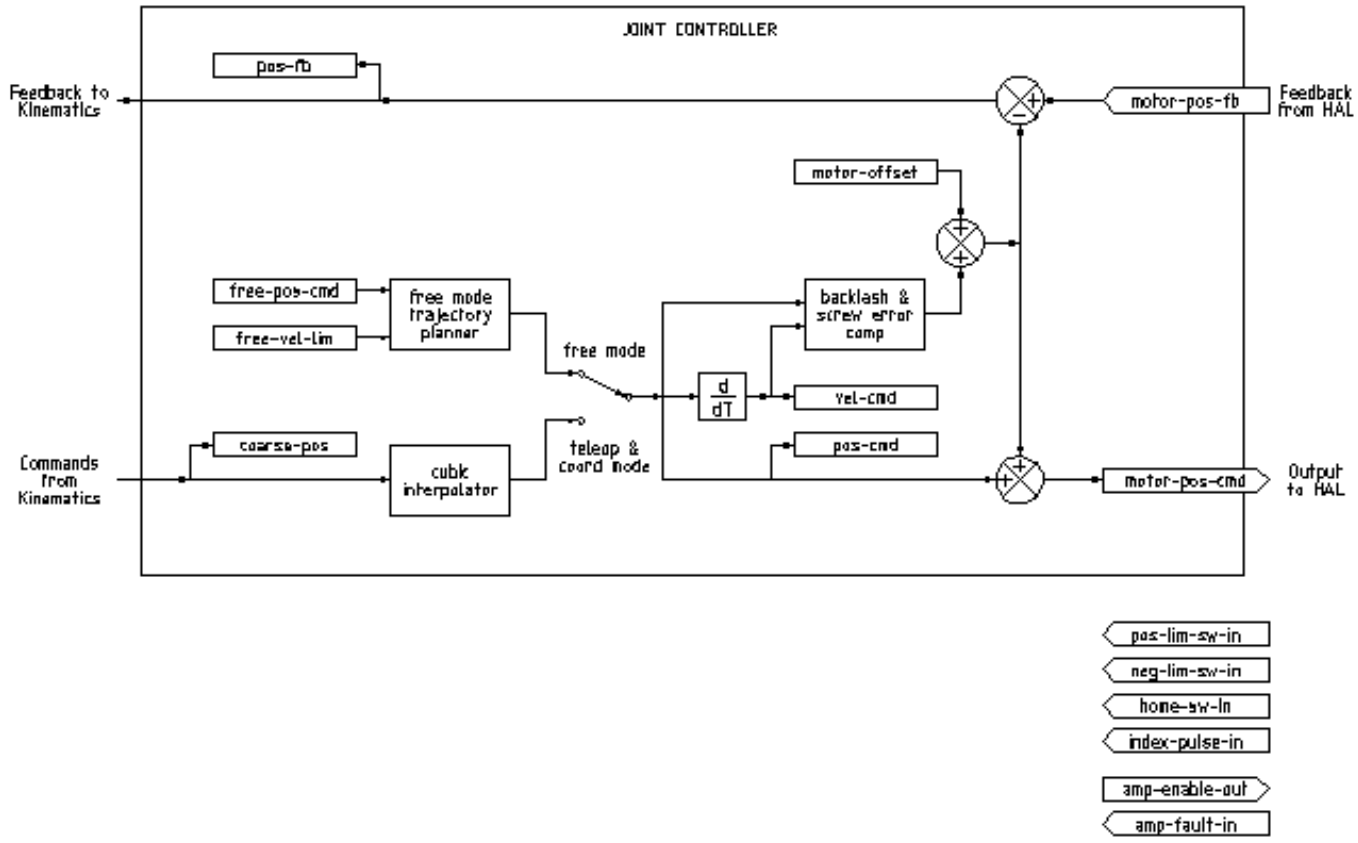



Figure 3.1: Блок-схема спільного контролера

На рисунку вище показано п'ять із семи наборів інформації про положення, які формують основний потік даних через контролер руху. Сім форм даних про положення є такими:

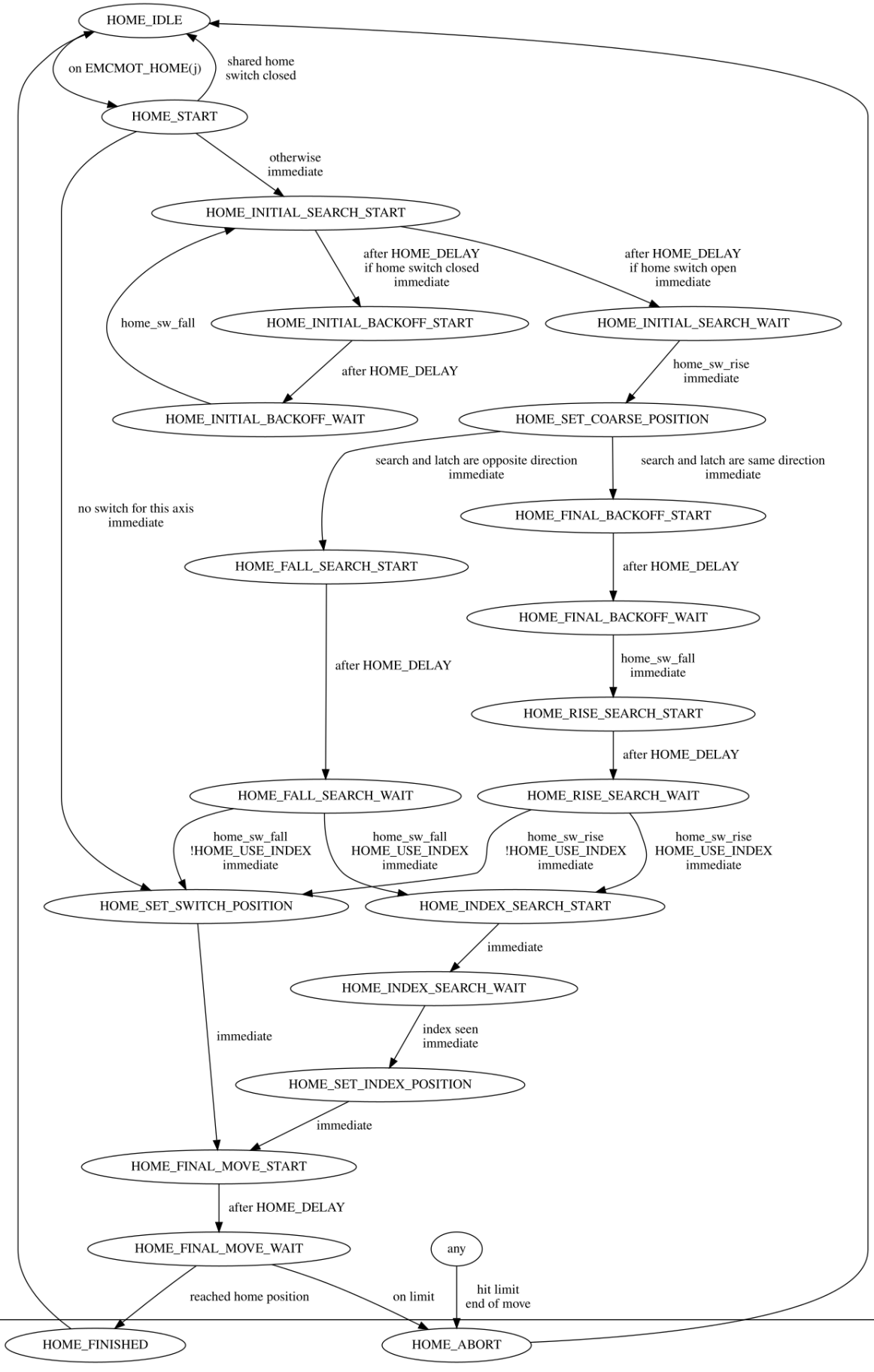
- *emcmotStatus->carte_pos_cmd* - Це бажане положення в декартових координатах. Воно оновлюється зі швидкістю траєкторії, а не сервоприводу. У режимі координат воно визначається планувальником траєкторії. У режимі teleop вона визначається планувальником траєкторії? У вільному режимі вона або копіюється з actualPos, або генерується шляхом застосування прямих кінематичних рівнянь до (2) або (3).
- *emcmotStatus->joints[n].coarse_pos* - Це бажане положення в координатах з'єднання, але до інтерполяції. Воно оновлюється зі швидкістю траєкторії, а не сервоприводу. У режимі координат вона генерується шляхом застосування обернених кінів до (1) У режимі телеоператора вона

генерується шляхом застосування обернених кінів до (1) У вільному режимі вона копіюється з (3), як я думаю.

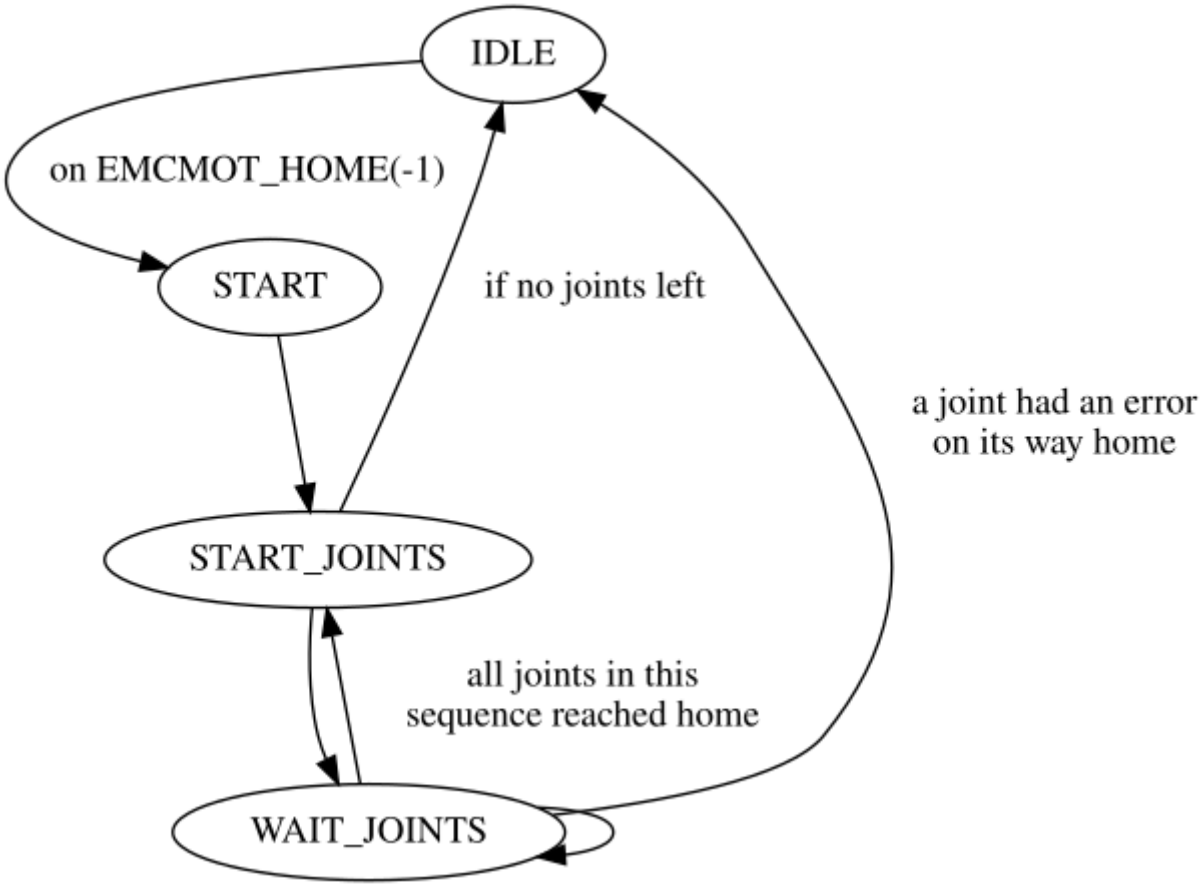
- ’emcmotStatus->joints[n].pos_cmd - Це бажане положення, в координатах суглоба, після інтерполяції. Новий набір цих координат генерується кожний сервоперіод. У режимі координат він генерується з (2) інтерполятором. У режимі телеоператора він генерується з (2) інтерполятором. У вільному режимі він генерується планувальником траєкторії вільного режиму.
- *emcmotStatus->joints[n].motor_pos_cmd* - Це бажане положення в координатах двигуна. Координати двигуна генеруються шляхом додавання компенсації люфту, компенсації похибки ходового гвинта та зміщення (для повернення в початкове положення) до (3). Воно генерується однаково незалежно від режиму і є виходом до контуру PID або іншого контуру положення.
- *emcmotStatus->joints[n].motor_pos_fb* - Це фактичне положення в координатах двигуна. Це вхідні дані від енкодерів або інших пристроїв зворотного зв’язку (або від віртуальних енкодерів на машинах з відкритим контуром). Воно «генерується» шляхом зчитування даних з пристрою зворотного зв’язку.
- *emcmotStatus->joints[n].pos_fb* - Це фактичне положення в координатах з’єднання. Воно генерується шляхом віднімання зміщення, компенсації похибки ходового гвинта та компенсації люфту від (5). Воно генерується однаково незалежно від режиму роботи.
- *emcmotStatus->carte_pos_fb* - Це фактичне положення в декартових координатах. Воно оновлюється зі швидкістю траєкторії, а не сервоприводу. В ідеалі, фактичне положення завжди обчислюється шляхом застосування прямої кінематики до (6). Однак пряма кінематика може бути недоступною або непридатною для використання, оскільки одна або кілька осей не повернуті в початкове положення. У цьому випадку є такі варіанти: А) підробити, скопіювавши (1), або Б) визнати, що ми насправді не знаємо декартових координат, і просто не оновлювати actualPos. Який би підхід не використовувався, я не бачу причин не робити це однаково, незалежно від режиму роботи. Я б запропонував наступне: якщо є прямі кінематичні функції, використовуйте їх, якщо вони не працюють через неверні осі або інші проблеми, в такому випадку виконайте (В). Якщо прямих кінематичних функцій немає, виконайте (А), оскільки в іншому випадку actualPos ніколи не буде оновлено.

# 3.7 Самонаведення

## 3.7.1 Діаграма стану самонаведення



### 3.7.2 Ще одна схема самонаведення




## 3.8 Команди

Команди реалізуються за допомогою великого оператора switch у функції emcmotCommandHandler(), яка викликається на швидкості сервоприводу. Докладніше про цю функцію пізніше.

Існує приблизно 44 команди — цей список все ще розробляється.

---

**Note**
Перелік cmd_code_t у файлі motion.h містить 73 команди, але оператор switch у файлі command.c враховує лише 70 команд (станом на 5 червня 2020 року). Команди ENABLE_WATCHDOG / DISABLE_WATCHDOG знаходяться в motion-logger.c. Можливо, вони застаріли. Команда SET_TELEOP_VECTOR з'являється тільки в motion-logger.c і не має ніякого ефекту, крім власного журналу.

---

### 3.8.1 ABORT

Команда ABORT просто зупиняє всі рухи. Вона може бути видана в будь-який час і завжди буде прийнята. Вона не вимикає контролер руху і не змінює інформацію про стан, а просто скасовує будь-який рух, який виконується в даний момент. Примітка: [Здається, що код вищого рівня (TASK і вище) також використовує ABORT для усунення несправностей. Кожного разу, коли виникає постійна помилка (наприклад, вихід за межі апаратних кінцевих вимикачів), код вищого

рівня надсилає постійний потік ABORT до контролера руху, намагаючись усунути помилку. Тисячі таких повідомлень... Це означає, що контролер руху повинен уникати постійних помилок. Це потрібно перевірити.]

#### 3.8.1.1 Вимоги

Жодного. Команда завжди приймається та виконується негайно.

#### 3.8.1.2 Результати

У вільному режимі планувальники траєкторії вільного режиму відключені. Це призводить до того, що кожен шарнір зупиняється так швидко, як дозволяє його обмеження прискорення (сповільнення). Зупинка не координується. У режимі телеоператора задана декартова швидкість встановлюється на нуль. Я не знаю, до якого саме зупинення це призведе (скоординоване, нескоординоване тощо), але зрештою з'ясую це. У режимі координації планувальник траєкторії в режимі координації отримує команду припинити поточний рух. Знову ж таки, я не знаю, до чого саме це призведе, але задокументую це, коли з'ясую.

### 3.8.2 FREE

Команда FREE переводить контролер руху у вільний режим. Вільний режим означає, що кожен шарнір є незалежним від усіх інших шарнірів. У вільному режимі ігноруються декартові координати, позиції та кінематика. По суті, кожен шарнір має власний простий планувальник траєкторії, і кожен шарнір повністю ігнорує інші шарніри. Деякі команди (наприклад, Joint JOG і HOME) працюють тільки у вільному режимі. Інші команди, включаючи всі, що мають відношення до декартових координат, у вільному режимі не працюють взагалі.

#### 3.8.2.1 Вимоги

Обробник команд не застосовує жодних вимог до команди FREE, вона завжди буде прийнята. Однак, якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_FLAG() == FALSE), то команда буде проігнорована. Ця поведінка контролюється кодом, який зараз знаходиться у функції *set_operating_mode()* в control.c, цей код потрібно очистити. Я вважаю, що команда не повинна ігноруватися без повідомлення, натомість обробник команд повинен визначати, чи може вона бути виконана, і повертати помилку, якщо це неможливо.

#### 3.8.2.2 Результати

Якщо машина вже перебуває у вільному режимі, нічого не відбувається. В іншому випадку машина переводиться у вільний режим. Планувальник траєкторії вільного режиму кожного суглоба ініціалізується до поточного положення суглоба, але планувальники не вмикаються, а суглоби залишаються нерухомими.

### 3.8.3 TELEOP

Команда TELEOP переводить машину в режим телеуправління. У режимі телеуправління рух машини базується на декартових координатах з використанням кінематики, а не на окремих суглобах, як у вільному режимі. Однак сам по собі планувальник траєкторії не використовується, натомість рух контролюється вектором швидкості. Рух у режимі телеуправління дуже схожий на біг підтюпцем, за винятком того, що він здійснюється в декартовому просторі, а не в просторі

суглобів. На машині з тривіальною кінематикою різниця між режимом телеоперації та вільним режимом незначна, і графічні інтерфейси для таких машин можуть навіть ніколи не видавати цю команду. Однак для нетривіальних машин, таких як роботи та гексаподи, режим телеоперації використовується для більшості рухів типу бігу, що виконуються за командою користувача.

#### 3.8.3.1 Вимоги

Командний обробник відхилить команду TELEOP з повідомленням про помилку, якщо кінематика не може бути активована через те, що один або декілька шарнірів не були повернені в початкове положення. Крім того, якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_FLAG() == FALSE), команда буде проігнорована (без повідомлення про помилку). Ця поведінка контролюється кодом, який зараз знаходиться у функції *set_operating_mode()* в control.c. Я вважаю, що команда не повинна бути проігнорована без повідомлення, натомість обробник команд повинен визначити, чи може вона бути виконана, і повернути помилку, якщо це неможливо.

#### 3.8.3.2 Результати

Якщо машина вже перебуває в режимі телеоператора, нічого не відбувається. В іншому випадку машина переводиться в режим телеоператора. Активується кінематичний код, інтерполятори спорожнюються і промиваються, а декартові команди швидкості встановлюються на нуль.

### 3.8.4 COORD

Команда COORD переводить верстат у координаційний режим. У координаційному режимі рух верстата базується на декартових координатах з використанням кінематики, а не на окремих з'єднаннях, як у вільному режимі. Крім того, для генерації руху використовується основний планувальник траєкторії на основі команд LINE, CIRCLE та/або PROBE, що знаходяться в черзі. Координаційний режим використовується під час виконання програми G-коду.

#### 3.8.4.1 Вимоги

Командний обробник відхилить команду COORD з повідомленням про помилку, якщо кінематику неможливо активувати через те, що один або декілька шарнірів не повернулися в початкове положення. Крім того, якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_FLAG() == FALSE), команда буде проігнорована (без повідомлення про помилку). Ця поведінка контролюється кодом, який зараз знаходиться у функції *set_operating_mode()* в control.c. Я вважаю, що команда не повинна бути проігнорована без повідомлення, натомість обробник команд повинен визначити, чи може вона бути виконана, і повернути помилку, якщо це неможливо.

#### 3.8.4.2 Результати

Якщо верстат вже перебуває в режимі координат, нічого не відбувається. В іншому випадку верстат переводиться в режим координат. Активується кінематичний код, інтерполятори спорожнюют і очищаються, а черги планувальника траєкторій спорожнюються. Планувальник траєкторій активний і очікує команди LINE, CIRCLE або PROBE.

### 3.8.5 ENABLE

Команда ENABLE вмикає контролер руху.

#### 3.8.5.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.5.2 Результати

Якщо контролер вже увімкнено, нічого не відбувається. Якщо ні, контролер увімкнено. Черги та інтерполятори очищуються. Будь-які операції переміщення або повернення в початкове положення припиняються. Виходи увімкнення підсилювача, пов’язані з активними з’єднаннями, увімкнено. Якщо пряма кінематика недоступна, машина переходить у вільний режим.

### 3.8.6 DISABLE

Команда DISABLE вимикає контролер руху.

#### 3.8.6.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.6.2 Результати

Якщо контролер вже вимкнений, нічого не відбувається. Якщо ні, контролер вимикається. Черги та інтерполятори очищаються. Будь-які операції переміщення або повернення в вихідне положення припиняються. Виходи підсилення, пов’язані з активними з’єднаннями, вимикаються. Якщо пряма кінематика недоступна, машина переходить у вільний режим.

### 3.8.7 ENABLE_AMPLIFIER

Команда ENABLE_AMPLIFIER вмикає вихід увімкнення підсилювача для підсилювача з одним виходом, не змінюючи нічого іншого. Може використовуватися для ввімкнення контролера швидкості шпинделя.

#### 3.8.7.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.7.2 Результати

Наразі нічого. (Виклик старої функції extAmpEnable наразі закоментовано.) Зрештою, це встановить пін HAL увімкнення підсилювача в значення true.

### 3.8.8 DISABLE_AMPLIFIER

Команда DISABLE_AMPLIFIER вимикає вихід увімкнення підсилювача для одного підсилювача, не змінюючи нічого іншого. Знову ж таки, корисно для контролерів швидкості шпинделя.

#### 3.8.8.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.8.2 Результати

Наразі нічого. (Виклик старої функції extAmpEnable наразі закоментовано.) Зрештою, це встановить пін HAL увімкнення підсилювача у значення false.

### 3.8.9 ACTIVATE_JOINT

Команда ACTIVATE_JOINT вмикає всі обчислення, пов’язані з одним з’єднанням, але не змінює вихідний контакт увімкнення підсилювача цього з’єднання.

#### 3.8.9.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.9.2 Результати

Розрахунки для зазначеного з’єднання ввімкнено. Вивід увімкнення підсилювача не змінюється, проте будь-які наступні команди ENABLE або DISABLE змінять вивід увімкнення підсилювача з’єднання.

### 3.8.10 DEACTIVATE_JOINT

Команда DEACTIVATE_JOINT вимикає всі обчислення, пов’язані з одним з’єднанням, але не змінює вихідний контакт увімкнення підсилювача цього з’єднання.

#### 3.8.10.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.10.2 Результати

Розрахунки для вказаного з’єднання ввімкнено. Вивід увімкнення підсилювача не змінюється, а наступні команди ENABLE або DISABLE не змінять вивід увімкнення підсилювача з’єднання.

### 3.8.11 ENABLE_WATCHDOG

Команда ENABLE_WATCHDOG вмикає апаратний сторожовий таймер (якщо він присутній).

#### 3.8.11.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.11.2 Результати

Наразі нічого. Старий сторожовий таймер був дивною річчю, яка використовувала певну звукову карту. Новий інтерфейс сторожового таймера може бути розроблений у майбутньому.

### 3.8.12 DISABLE_WATCHDOG

Команда DISABLE_WATCHDOG вимикає апаратний сторожовий таймер (якщо він присутній).

#### 3.8.12.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.12.2 Результати

Наразі нічого. Старий сторожовий таймер був дивною річчю, яка використовувала певну звукову карту. Новий інтерфейс сторожового таймера може бути розроблений у майбутньому.

### 3.8.13 PAUSE

Команда PAUSE зупиняє планувальник траєкторії. Вона не діє в режимі вільного або телеоперативного керування. На даний момент я не знаю, чи вона негайно зупиняє всі рухи, чи завершує поточний рух, а потім зупиняється, перш ніж витягнути інший рух із черги.

#### 3.8.13.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.13.2 Результати

Планувальник траєкторії зупиняється.

### 3.8.14 RESUME

Команда RESUME перезапускає планувальник траєкторії, якщо він призупинений. Вона не має жодного ефекту у вільному режимі або режимі телеоп, або якщо планувальник не призупинений.

#### 3.8.14.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.14.2 Результати

Планувальник траєкторії відновлює роботу.

### 3.8.15 STEP

Команда STEP перезапускає планувальник траєкторії, якщо він призупинений, і дає йому команду зупинитися знову, коли він досягне певної точки. Вона не діє в режимі вільного або телеоперативного керування. На даний момент я не знаю, як саме це працює. Я додам більше інформації, коли глибше вивчу планувальник траєкторії.

#### 3.8.15.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.15.2 Результати

Планувальник траєкторії відновлює роботу, а потім зупиняється, коли досягає певної точки.

### 3.8.16 SCALE

Команда SCALE масштабує всі обмеження швидкості та команди на задану величину. Вона використовується для реалізації перевизначення швидкості подачі та інших подібних функцій. Масштабування працює в режимах free, teleop та coord і впливає на все, включаючи швидкості повернення в вихідне положення тощо. Однак індивідуальні обмеження швидкості суглобів не змінюються.

#### 3.8.16.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята.

#### 3.8.16.2 Результати

Усі команди швидкості масштабуються на задану константу.

### 3.8.17 OVERRIDE_LIMITS

Команда OVERRIDE_LIMITS запобігає спрацьовуванню обмежень до кінця наступної команди JOG. Зазвичай вона використовується для того, щоб машина могла відійти від кінцевого вимикача після спрацьовування. (Команда може бути використана для перевизначення обмежень або для скасування попереднього перевизначення.)

#### 3.8.17.1 Вимоги

Жодної. Команду можна видати будь-коли, і вона завжди буде прийнята. (Я думаю, що вона має працювати лише у вільному режимі.)

#### 3.8.17.2 Результати

Обмеження на всі з’єднання ігноруються до кінця наступної команди JOG. (Наразі ця функція не працює… після отримання команди OVERRIDE_LIMITS обмеження ігноруються, доки інша команда OVERRIDE_LIMITS не увімкне їх знову.)

### 3.8.18 HOME

Команда HOME ініціює послідовність повернення в початкове положення на вказаному з'єднанні. Фактична послідовність повернення в початкове положення визначається низкою параметрів конфігурації і може варіюватися від простого встановлення поточного положення на нуль до багатоетапного пошуку перемикача початкового положення та імпульсу індексу, після чого відбувається переміщення до довільного початкового положення. Докладнішу інформацію про послідовність повернення в початкове положення див. у розділі «Повернення в початкове положення» посібника з інтегратора.

#### 3.8.18.1 Вимоги

Команда буде проігнорована без попереднього налаштування, якщо машина не перебуває у вільному режимі.

#### 3.8.18.2 Результати

Будь-який поштовх або інший рух суглоба переривається, і починається послідовність повернення до початкового положення.

### 3.8.19 JOG_CONT

Команда JOG_CONT ініціює безперервний поступальний рух на одному суглобі. Безперервний поступальний рух генерується шляхом встановлення цільової позиції планувальника траєкторії у вільному режимі на точку, що знаходиться за межами діапазону переміщення суглоба. Це гарантує, що планувальник буде рухатися постійно, доки його не зупинять межі суглоба або команда ABORT. Зазвичай графічний інтерфейс користувача надсилає команду JOG_CONT, коли користувач натискає кнопку поступального руху, і команду ABORT, коли кнопка відпускається.

#### 3.8.19.1 Вимоги

Командний обробник відхилить команду JOG_CONT з повідомленням про помилку, якщо машина не перебуває у вільному режимі, або якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_F == FALSE), або якщо рух не ввімкнено. Він також мовчки проігнорує команду, якщо шарнір вже знаходиться на межі або за межею свого діапазону, і заданий рух погіршить ситуацію.

#### 3.8.19.2 Результати

Активується планувальник траєкторії вільного режиму для суглоба, ідентифікованого emcmotCommand->axis, з цільовою позицією за межами кінця ходу суглоба та обмеженням швидкості emcmotCommand->vel. Це запускає рух суглоба, і рух триватиме, доки не буде зупинений командою ABORT або досягненням межі. Планувальник вільного режиму прискорюється до межі прискорення суглоба на початку руху і сповільнюється до межі прискорення суглоба, коли зупиняється.

### 3.8.20 JOG_INCR

Команда JOG_INCR ініціює інкрементний поштовх на одному шарнірі. Інкрементні поштовхи є кумулятивними, іншими словами, видача двох команд JOG_INCR, кожна з яких вимагає переміщення на 0,100 дюйма, призведе до переміщення на 0,200 дюйма, навіть якщо друга команда видається до завершення першої. Зазвичай інкрементальні переміщення зупиняються, коли вони проходять бажану відстань, однак вони також зупиняються, коли досягають межі або при виконанні команди ABORT.

#### 3.8.20.1 Вимоги

Команда обробки буде мовчки відхиляти команду JOG_INCR, якщо машина не перебуває у вільному режимі, або якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_FLAG() == FALSE), або якщо рух не ввімкнено. Вона також мовчки ігноруватиме команду, якщо шарнір вже знаходиться на межі або за межею, і заданий рух погіршить ситуацію.

#### 3.8.20.2 Результати

Активується планувальник траєкторії вільного режиму для з'єднання, ідентифікованого emcmotComma >axis, цільове положення збільшується/зменшується на emcmotCommand->offset, а обмеження швидкості встановлюється на emcmotCommand->vel. Планувальник траєкторії вільного режиму генерує плавний трапецієподібний рух від поточного положення до цільового положення. Планувальн може правильно обробляти зміни в цільовому положенні, що відбуваються під час руху, тому можна швидко послідовно видавати кілька команд JOG_INCR. Планувальник вільного режиму прискорюється до межі прискорення суглоба на початку руху і сповільнюється до межі прискорення суглоба, щоб зупинитися в цільовому положенні.

### 3.8.21 JOG_ABS

Команда JOG_ABS ініціює абсолютне переміщення на одному суглобі. Абсолютне переміщення — це просте переміщення до певного місця в координатах суглоба. Зазвичай абсолютні переміщення зупиняються, коли досягають бажаного місця, однак вони також зупиняються, коли досягають межі або за командою ABORT.

#### 3.8.21.1 Вимоги

Команда обробки буде мовчки відхиляти команду JOG_ABS, якщо машина не перебуває у вільному режимі, або якщо будь-який шарнір знаходиться в русі (GET_MOTION_INPOS_FLAG() == FALSE), або якщо рух не ввімкнено. Вона також мовчки ігноруватиме команду, якщо шарнір вже знаходиться на межі або за межею, і заданий рух погіршить ситуацію.

#### 3.8.21.2 Результати

Активується планувальник траєкторії вільного режиму для з'єднання, ідентифікованого emcmotComma >axis, цільове положення встановлюється на emcmotCommand->offset, а обмеження швидкості встановлюється на emcmotCommand->vel. Планувальник траєкторії вільного режиму генерує плавний трапецієподібний рух від поточного положення до цільового положення. Планувальник може правильно обробляти зміни в цільовому положенні, що відбуваються під час руху. Якщо кілька команд JOG_ABS видаються швидко поспіль, кожна нова команда змінює цільове положення, і машина переходить до кінцевого заданого положення. Планувальник вільного режиму прискорюєтьс до межі прискорення суглоба на початку руху і сповільнюється до межі прискорення суглоба, щоб зупинитися в цільовому положенні.

### 3.8.22 SET_LINE

Команда SET_LINE додає пряму лінію до черги планувальника траєкторії.

(Більше пізніше)

### 3.8.23 SET_CIRCLE

Команда SET_CIRCLE додає круговий рух до черги планувальника траєкторії.

(Більше пізніше)

### 3.8.24 SET_TELEOP_VECTOR

Команда SET_TELEOP_VECTOR наказує контролеру руху рухатися вздовж певного вектора в декартовому просторі.

(Більше пізніше)

### 3.8.25 PROBE

Команда PROBE дає інструкцію контролеру руху рухатися до певної точки в декартовому просторі, зупиняючись та записуючи своє положення, якщо спрацьовує вхід зонда.

(Більше пізніше)

### 3.8.26 CLEAR_PROBE_FLAG

Команда CLEAR_PROBE_FLAG використовується для скидання вхідного сигналу зонда під час підготовки до команди PROBE. (Запитання: чому команда PROBE не повинна автоматично скидати вхідний сигнал?)

(Більше пізніше)

### 3.8.27 SET_xix

Існує приблизно 15 команд SET_xxx, де xxx — це назва певного параметра конфігурації. Очікується, що з додаванням нових параметрів з'явиться ще кілька команд SET. Я хотів би знайти більш чіткий спосіб налаштування та зчитування параметрів конфігурації. Існуючі методи вимагають додавання багатьох рядків коду до декількох файлів кожного разу, коли додається новий параметр. Більша частина цього коду є ідентичною або майже ідентичною для кожного параметра.

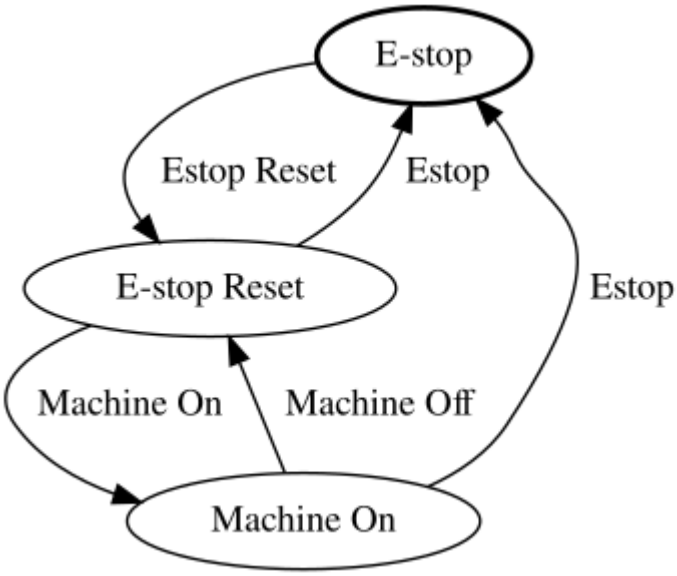
## 3.9 Компенсація люфту та похибки гвинта

Компенсація люфту та помилки гвинта FIXME

## 3.10 Контролер завдань (EMCTASK)

### 3.10.1 Штат

Завдання має три можливі внутрішні стани: **Аварійна зупинка**, **Скидання аварійної зупинки** та **Встановлення ввімкнено**.




## 3.11 Контролер вводу-виводу (EMCIO)

Контролер вводу/виводу є частиною TASK. Він взаємодіє із зовнішніми пристроями вводу/виводу за допомогою контактів HAL.

Наразі ESTOP/Enable, подача охолоджувальної рідини та зміна інструменту обробляються iocontrol. Це події з відносно низькою швидкістю, тоді як високошвидкісний координований ввід/вивід обробляється під час руху.

emctaskmain.cc надсилає команди вводу/виводу через taskclass.cc.

Процес основного циклу iocontrol:

- перевіряє, чи змінилися входи HAL
- перевіряє, чи read_tool_inputs() вказує на завершення зміни інструменту, та встановлює emcioStatus.status

## 3.12 Інтерфейс користувача

Інтерфейси користувача FIXME

## 3.13 Вступ до libnml

libnml походить від NIST rcslib без підтримки всіх платформ. Багато обгортків навколо коду, специфічного для платформи, було видалено разом із значною частиною коду, який не потрібен LinuxCNC. Сподіваємося, що сумісність із rcslib залишиться достатньою, щоб програми можна було реалізувати на платформах, відмінних від Linux, і вони все одно могли б спілкуватися з LinuxCNC.

Цей розділ не є вичерпним посібником з використання libnml (або rcslib), а надає загальний огляд кожного класу C++ та їхніх членів-функцій. Спочатку більшість цих приміток будуть випадковими коментарями, доданими під час ретельного вивчення та модифікації коду.

## 3.14 Зв’язаний список

Базовий клас для підтримки зв’язаного списку. Це один з основних структурних блоків, що використовуються для передачі NML-повідомлень та різноманітних внутрішніх структур даних.

## 3.15 Вузол зв’язаного списку

Базовий клас для створення зв’язаного списку - Призначення: зберігати вказівники на попередній та наступний вузли, вказівник на дані та розмір даних.

Пам’ять для зберігання даних не виділяється.

## 3.16 Спільна пам’ять

Надає блок спільної пам’яті разом із семафором (успадкованим від класу Semaphore). Створення та знищення семафора обробляється конструктором та деструктором SharedMemory.

## 3.17 ShmBuffer

Клас для передачі NML-повідомлень між локальними процесами з використанням спільного буфера пам’яті. Значна частина внутрішніх механізмів успадкована від класу CMS.

## 3.18 Таймер

Клас Timer забезпечує періодичний таймер, обмежений лише роздільною здатністю системного годинника. Якщо, наприклад, процес потрібно запускати кожні 5 секунд незалежно від часу, необхідного для його виконання, наступний фрагмент коду демонструє, як це зробити:

```
main()
{
    timer = new Timer(5.0);    /* b''Ib''b''нb''b''ib''b''цb''b''ib''b''ab''b''лb''b''ib''b ↩
        ''зb''b''уb''b''вb''b''ab''b''тb''b''иb'' b''тb''b''ab''b''йb''b''мb''b''eb''b''pb'' ↩
         b''нb''b''ab'' 5 b''cb''b''eb''b''кb''b''уb''b''нb''b''дb'' loop */
    while(0) {
        /* b''Bb''b''иb''b''кb''b''оb''b''нb''b''ab''b''йb''b''тb''b''eb'' b''пb''b''eb''b' ↩
            'вb''b''нb''b''иb''b''йb'' b''пb''b''pb''b''оb''b''цb''b''eb''b''cb'' */
        timer.wait();    /* b''3b''b''ab''b''чb''b''eb''b''кb''b''ab''b''йb''b''тb''b''eb'' ↩
             b''дb''b''оb'' b''нb''b''ab''b''cb''b''тb''b''уb''b''пb''b''нb''b''оb''b''гb''b ↩
            ''оb'' 5-b''cb''b''eb''b''кb''b''уb''b''нb''b''дb''b''нb''b''оb''b''гb''b''оb''  ↩
            b''ib''b''нb''b''тb''b''eb''b''pb''b''вb''b''ab''b''лb''b''уb''*/
    }
    delete timer;
}
```

## 3.19 Семафор

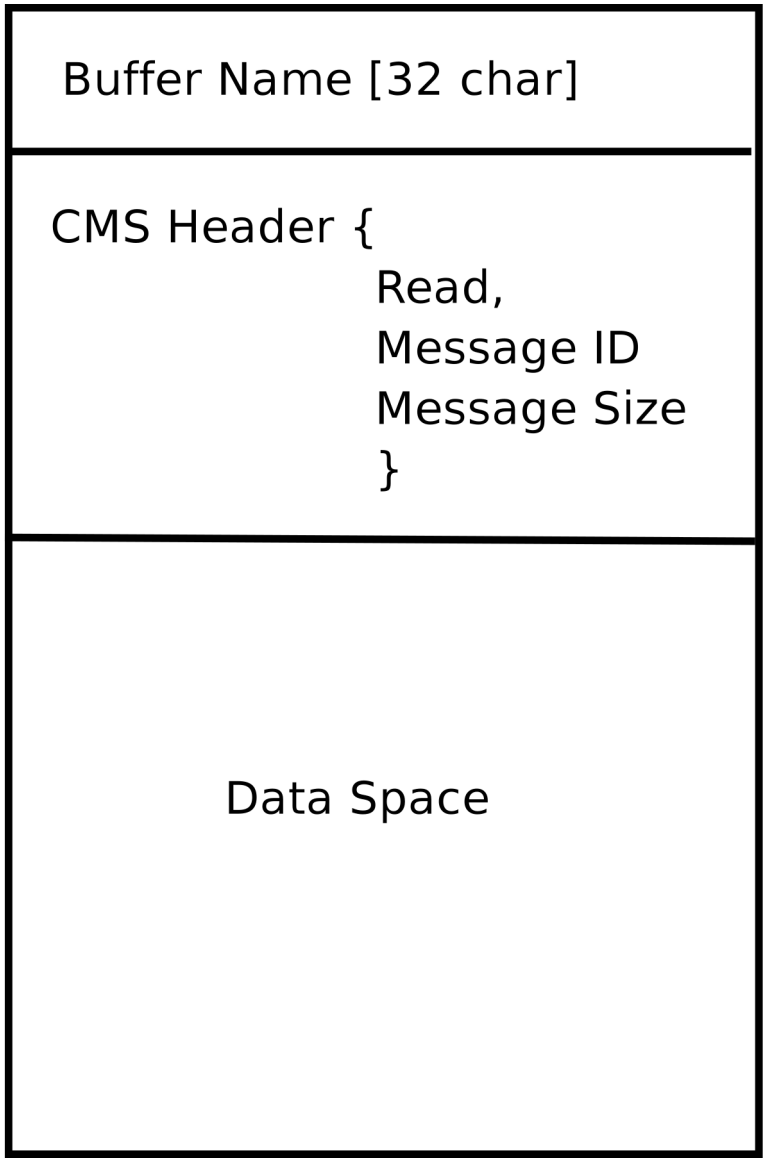
Клас Semaphore надає метод взаємного виключення для доступу до спільного ресурсу. Функція отримання семафора може або блокувати доступ доти, доки він не стане доступним, або повертатися після закінчення часу очікування, або повертатися негайно з отриманням семафора чи без нього. Конструктор створить семафор або приєднається до існуючого, якщо ідентифікатор вже використовується.

Semaphore::destroy() має викликатися лише останнім процесом.

## 3.20 CMS

В основі libnml лежить клас CMS, який містить більшість функцій, що використовуються libnml і, зрештою, NML. Багато внутрішніх функцій перевантажені, щоб забезпечити специфічні, залежні від апаратного забезпечення методи передачі даних. Зрештою, все обертається навколо центрального блоку пам’яті (який називається «буфером повідомлень» або просто «буфером»). Цей буфер може існувати як блок спільної пам’яті, до якого мають доступ інші процеси CMS/NML, або як локальний і приватний буфер для даних, що передаються через мережу або послідовні інтерфейси.

Буфер динамічно розподіляється під час виконання, щоб забезпечити більшу гнучкість підсистеми CMS/NML. Розмір буфера повинен бути достатньо великим, щоб вмістити найбільше повідомлення, невелику кількість для внутрішнього використання та дозволити кодування повідомлення, якщо обрано цю опцію (кодовані дані будуть розглянуті пізніше). Наступний малюнок є внутрішнім видом простору буфера.




**Буфер CMS** Базовий клас CMS в першу чергу відповідає за створення комунікаційних шляхів та взаємодію з операційною системою.

## 3.21 Формат файлу конфігурації

Конфігурація NML складається з двох типів форматів рядків. Один для буферів, а другий для процесів, які підключаються до буферів.

### 3.21.1 Буферна лінія

Оригінальний формат буферного рядка NIST такий:

- *B ім'я тип розмір хоста neutr RPC# buffer# max_procs ключ [конфігурації, специфічні для типу]*
- «B» – ідентифікує цю лінію як конфігурацію буфера.

- *name* – ідентифікатор буфера.
- *type* – описує тип буфера – SHMEM, LOCMEM, FILEMEM, PHANTOM або GLOBMEM.
- *host* – це або IP-адреса, або ім’я хоста для NML-сервера
- *size* – це розмір буфера
- *neut* – логічне значення, яке вказує, чи дані в буфері закодовані в машинно-незалежному форматі, чи в необробленому вигляді.
- *RPC#* – Застарілий – Заповнювач збережено лише для зворотної сумісності.
- *buffer#* – унікальний ідентифікаційний номер, який використовується, якщо сервер керує кількома буферами.
- *max_procs* – максимальна кількість процесів, яким дозволено підключатися до цього буфера.
- *ключ* – це числовий ідентифікатор буфера спільної пам’яті

### 3.21.2 Конфігурації, що відповідають певному типу

Тип буфера передбачає додаткові параметри конфігурації, тоді як операційна система хоста виключає певні комбінації. З метою узагальнення опублікованої документації в єдиному форматі, буде розглянуто лише тип буфера **SHMEM**.

- *mutex=os_sem* – режим за замовчуванням для забезпечення блокування буферної пам’яті семафорам
- *mutex=none* – Не використовується
- *mutex=no_interrupts* – не застосовується в системі Linux
- *mutex=no_switching* – не застосовується в системі Linux
- *mutex=mao split* – Розділяє буфер навпіл (або більше) та дозволяє одному процесу отримувати доступ до частини буфера, поки інший процес записує дані в іншу частину.
- *TCP=(номер порту)* – Вказує, який мережевий порт використовувати.
- *UDP=(номер порту)* - те саме
- *STCP=(номер порту)* - те саме
- *serialPortDevName=(послідовний порт)* – Недокументовано.
- *passwd=ім’я_файлу.pwd* – Додає рівень безпеки до буфера, вимагаючи від кожного процесу введення пароля.
- *bsem* – документація NIST передбачає ключ для блокувального семафора, і якщо bsem=-1, блокування читання запобігається.
- *queue* – Вмикає передачу повідомлень у чергу.
- *ascii* - Кодування повідомлень у форматі звичайного тексту
- *disp* - Кодувати повідомлення у форматі, придатному для відображення (???)
- *xdr* – Кодувати повідомлення у зовнішньому представленні даних. (докладніше див. rpc/xdr.h).
- *diag* - Вмикає збереження діагностичних даних у буфері (таймінг та кількість байтів?)

### 3.21.3 Технологічна лінія

Оригінальний формат NIST для технологічної лінії такий:

**P ім’я буфер тип хост-операції сервер тайм-аут master c_num [конфігурації, специфічні для типу]**

- «P» – ідентифікує цей рядок як конфігурацію процесу.
- *name* – це ідентифікатор процесу.
- *buffer* – один із буферів, визначених в іншому місці конфігураційного файлу.
- *type* – визначає, чи є цей процес локальним чи віддаленим відносно буфера.
- *host* – вказує, де в мережі виконується цей процес.
- *ops* – надає процесу доступ до буфера лише для читання, лише для запису або для читання/запису.
- *server* – вказує, чи цей процес запускатиме сервер для цього буфера.
- *timeout* - встановлює характеристики тайм-ауту для звернень до буфера.
- *master* - вказує, чи відповідає цей процес за створення та знищення буфера.
- *c_num* - ціле число від нуля до (max_procs -1)

### 3.21.4 Коментарі до конфігурації

Деякі комбінації конфігурацій є недійсними, тоді як інші передбачають певні обмеження. У системі Linux GLOBMEM є застарілим, тоді як PHANTOM є дійсно корисним лише на етапі тестування програми, так само як і FILEMEM. LOCMEM є малокорисним для багатопроцесорних програм і пропонує лише обмежені переваги в продуктивності порівняно з SHMEM. Таким чином, SHMEM залишається єдиним типом буфера, який можна використовувати з LinuxCNC.

Опція neut використовується тільки в багатопроцесорних системах, де різні (і несумісні) архітектури спільно використовують блок пам’яті. Імовірність побачити систему такого типу поза музеєм або науково-дослідною установою є дуже низькою і стосується тільки буферів GLOBMEM.

Номер RPC задокументовано як застарілий і зберігається лише з міркувань сумісності.

З унікальним іменем буфера наявність числового ідентифікатора здається безглуздою. Необхідно переглянути код, щоб визначити логіку. Аналогічно, поле ключа спочатку здається зайвим, і його можна отримати з імені буфера.

Мета обмеження кількості процесів, яким дозволено підключатися до будь-якого одного буфера, неясна з існуючої документації та з оригінального вихідного коду. Дозволити невизначеній кількості процесів підключатися до буфера не складніше реалізувати.

Типи мьютексів зводяться до одного з двох: стандартного «os_sem» або «mao split». Більшість повідомлень NML є відносно короткими і можуть бути скопійовані в буфер або з нього з мінімальною затримкою, тому роздільне читання не є необхідним.

Кодування даних є актуальним лише при передачі до віддаленого процесу - використання TCP або UDP передбачає кодування XDR. Кодування ASCII може бути корисним для діагностики або передачі даних до вбудованої системи, яка не підтримує NML.

Протоколи UDP мають менше перевірок даних і дозволяють відкидати певний відсоток пакетів. TCP є надійнішим, але трохи повільнішим.

Якщо LinuxCNC підключається до мережі, бажано, щоб це була локальна мережа за брандмауером. Єдина причина, яка може виправдати доступ до LinuxCNC через Інтернет, — це дистанційна діагностика. Однак це можна зробити набагато безпечніше іншими способами, наприклад, за допомогою веб-інтерфейсу.

Точна поведінка, коли час очікування встановлений на нуль або від'ємне значення, не зрозуміла з документів NIST. Згадуються тільки INF і додатні значення. Однак, занурившись у вихідний код rcslib, стає очевидним, що застосовується наступне:

**тайм-аут > 0**
Блокування доступу до закінчення інтервалу тайм-ауту або доки доступ до буфера не стане доступним.

**тайм-аут = 0**
Доступ до буфера можливий лише тоді, коли в цей час жоден інший процес не виконує читання або запис.

**тайм-аут < 0 або INF**
Доступ блокується, доки буфер не стане доступним.

## 3.22 Базовий клас NML

Розгорніть списки та зв'язок між NML, NMLmsg та класами CMS нижчого рівня.

Не слід плутати з NMLmsg, RCS_STAT_MSG або RCS_CMD_MSG.

NML відповідає за розбір конфігураційного файлу, налаштування буферів cms та є механізмом маршрутизації повідомлень до правильного(их) буфера(ів). Для цього NML створює кілька списків для:

- створені або підключені буфери cms.
- процеси та буфери, до яких вони підключаються
- довгий список функцій форматування для кожного типу повідомлення

Останній пункт, ймовірно, є основною причиною багатьох нарікань на libnml/rcslib та NML загалом. Кожне повідомлення, що передається через NML, вимагає додавання певної кількості інформації, окрім власне даних. Для цього послідовно викликаються кілька функцій форматування, щоб зібрати фрагменти загального повідомлення. Функції форматування включатимуть NML_TYPE, MSG_TYPE, крім даних, оголошених у похідних класах NMLmsg. Зміни в порядку виклику функцій форматування, а також переданих змінних порушать сумісність з rcslib, якщо з ними погратися. Є причини для збереження сумісності з rcslib, і є вагомі причини для втручання в код. Питання в тому, які з цих причин є більш вагомими?

### 3.22.1 Внутрішні механізми NML

#### 3.22.1.1 Конструктор NML

NML::NML() аналізує файл конфігурації та зберігає його у зв'язаному списку, який передається конструкторам cms у вигляді окремих рядків. Функція конструктора NML полягає у виклику відповідного конструктора cms для кожного буфера та підтримці списку об'єктів cms і процесів, пов'язаних з кожним буфером.

Саме за допомогою вказівників, що зберігаються у списках, NML може взаємодіяти з CMS, і саме тому Doxygen не показує реальні взаємозв'язки.

---

**Note**
Конфігурація зберігається в пам'яті перед передачею покажчика на конкретний рядок до конструктора cms. Потім конструктор cms знову аналізує рядок, щоб витягти кілька змінних... Було б доцільніше виконати ВЕСЬ аналіз і зберегти змінні в структурі, яка передається до конструктора cms. Це дозволило б уникнути обробки рядків і зменшити кількість дублювання коду в cms...

---

#### 3.22.1.2 Читання/запис NML

Виклики NML::read і NML::write виконують подібні завдання в частині обробки повідомлення. Єдина реальна відмінність полягає в напрямку потоку даних.

Виклик функції читання спочатку отримує дані з буфера, а потім викликає format_output(), тоді як функція запису викликає format_input() перед передачею даних до буфера. Саме в format_xxx() відбувається побудова або деконструкція повідомлення. Список різних функцій викликається по черзі, щоб розмістити різні частини заголовка NML (не плутати з заголовком cms) у правильному порядку. Остання функція, що викликається, — emcFormat() у emc.cc.

#### 3.22.1.3 NMLmsg та зв'язки NML

NMLmsg — це базовий клас, від якого походять усі класи повідомлень. Кожен клас повідомлень повинен мати унікальний ідентифікатор (який передається конструктору) та функцію update(*cms). Функція update() викликається функціями читання/запису NML під час виклику форматувальника NML — покажчик на форматувальник буде оголошений у конструкторі NML у певний момент. Завдяки зв'язаним спискам, які створює NML, він може вибрати покажчик cms, який передається до форматувальника, і, отже, який буфер буде використовуватися.

## 3.23 Додавання власних команд NML

LinuxCNC - це чудова програма, але деякі її частини потребують доопрацювання. Як відомо, комунікація здійснюється через канали NML, а дані, що надсилаються через такий канал, є одним із класів, визначених у emc.hh (реалізовано в emc.cc). Якщо комусь потрібен тип повідомлення, якого не існує, він повинен виконати наступні кроки, щоб додати новий. (Повідомлення, яке я додав у прикладі, називається EMC_IO_GENERIC (успадковує EMC_IO_CMD_MSG (успадковує RCS_CMD_MSG)))

1. додати визначення класу EMC_IO_GENERIC до emc2/src/emc/nml_intf/emc.hh
2. додати визначення типу: #define EMC_IO_GENERIC_TYPE \((NMLTYPE) 1605)

    a. (Я обрав 1605, бо він був доступний) до emc2/src/emc/nml_intf/emc.hh
3. додати регістр EMC_IO_GENERIC_TYPE до emcFormat у emc2/src/emc/nml_intf/emc.cc
4. додати регістр EMC_IO_GENERIC_TYPE до emc_symbol_lookup в emc2/src/emc/nml_intf/emc.cc
5. додати функцію EMC_IO_GENERIC::update до emc2/src/emc/nml_intf/emc.cc

Перекомпілюйте, і нове повідомлення має бути там. Наступний етап — надсилати такі повідомлення звідкись, отримувати їх в іншому місці та робити з ними якісь дії.

## 3.24 Стіл інструментів та змінник інструментів

LinuxCNC взаємодіє з апаратним забезпеченням змінника інструментів та має внутрішню абстракцію змінника інструментів. LinuxCNC керує інформацією про інструменти у файлі таблиці інструментів.

### 3.24.1 Абстракція Toolchanger у LinuxCNC

LinuxCNC підтримує два типи апаратного забезпечення для зміни інструментів, які називаються *nonrandom* та *random*. Налаштування INI [EMCIO]RANDOM_TOOLCHANGER визначає, до якого з цих типів апаратного забезпечення підключено LinuxCNC.

#### 3.24.1.1 Невипадкові змінники інструментів

Апаратне забезпечення невипадкового змінника інструментів повертає кожен інструмент у гніздо, з якого його було спочатку завантажено.

Прикладами обладнання для невипадкової зміни інструментів є ”ручний” змінник інструментів, токарні інструментальні револьвери та рейкові змінники інструментів.

При налаштуванні для невипадкового змінювача інструментів LinuxCNC не змінює номер кишені у файлі таблиці інструментів під час завантаження та вивантаження інструментів. Всередині LinuxCNC під час зміни інструменту інформація про інструмент **копіюється** з вихідної кишені таблиці інструментів до кишені 0 (яка представляє шпиндель), замінюючи будь-яку інформацію про інструмент, яка була там раніше.

---

**Note**
У LinuxCNC, налаштованому для невипадкового змінювача інструментів, інструмент 0 (T0) має особливе значення: «відсутність інструменту». T0 може не з’являтися у файлі таблиці інструментів, і перехід на T0 призведе до того, що LinuxCNC вважатиме, що шпиндель порожній.

---

#### 3.24.1.2 Випадкові змінники інструментів

Апаратне забезпечення для випадкової зміни інструментів замінює інструмент у шпинделі (якщо такий є) на запитуваний інструмент під час зміни інструменту. Таким чином, гніздо, в якому знаходиться інструмент, змінюється під час його вставки та виставки зі шпинделя.

Прикладом обладнання для випадкової зміни інструментів є карусельний змінник інструментів.

При налаштуванні для випадкового змінювача інструментів LinuxCNC обмінює номерами кишені старого та нового інструменту у файлі таблиці інструментів під час завантаження інструментів. Всередині LinuxCNC під час зміни інструменту інформація про інструмент **обмінюється** між вихідною кишенею таблиці інструментів та кишенею 0 (яка представляє шпиндель). Отже, після зміни інструменту кишеня 0 в таблиці інструментів містить інформацію про новий інструмент, а кишеня, з якої походить новий інструмент, містить інформацію про старий інструмент (інструмент, який був у шпинделі до зміни інструменту), якщо така є.

---

**Note**
Якщо LinuxCNC налаштований для випадкового перемикання інструментів, інструмент 0 (T0) не має **жодного** особливого значення. Він розглядається так само, як і будь-який інший інструмент у таблиці інструментів. Зазвичай T0 використовується для позначення «відсутності інструменту» (тобто інструменту з нульовим TLO), щоб у разі потреби можна було зручно звільнити шпиндель.

---

### 3.24.2 Стіл інструментів

LinuxCNC відстежує інструменти у файлі під назвою tool table. У таблиці інструментів записується така інформація для кожного інструменту:

**номер інструменту**
Ціле число, яке однозначно ідентифікує цей інструмент. Номери інструментів обробляються LinuxCNC по-різному, коли вони налаштовані для випадкових та невипадкових змінників інструментів:

- Коли LinuxCNC налаштовано для невипадкового змінника інструментів, це число має бути додатним. T0 отримує спеціальну обробку та не може з'являтися в таблиці інструментів.
- Коли LinuxCNC налаштовано для випадкового пристрою зміни інструментів, це число має бути невід'ємним. T0 дозволено в таблиці інструментів і зазвичай використовується для позначення "відсутності інструменту", тобто порожнього гнізда.

**номер кишені**
Ціле число, яке ідентифікує кишеню або слот в апаратному забезпеченні змінювача інструментів, де знаходиться інструмент. Номери кишень обробляються LinuxCNC по-різному, якщо він налаштований для випадкових і невипадкових змінювачів інструментів:

- Коли LinuxCNC налаштований для невипадкового змінювача інструментів, номер кишені в файлі інструментів може бути будь-яким додатним цілим числом (кишеня 0 не допускається). LinuxCNC без попередження ущільнює номери кишень під час завантаження файлу інструменті тому можуть бути відмінності між номерами кишень у файлі інструментів та внутрішніми номерами кишень, що використовуються LinuxCNC з невипадковим змінювачем інструментів.
- Коли LinuxCNC налаштовано для випадкового змінника інструментів, номери гнізд у файлі інструментів повинні бути від 0 до 1000 включно. Гнізда 1-1000 знаходяться в змінніку інструментів, гніздо 0 – це шпиндель.

**діаметр**
Діаметр інструменту, в одиницях виміру.

**зміщення довжини інструмента**
Зміщення довжини інструменту (також відоме як TLO) до 9 осей, в одиницях машинного вимірювання. Осі, які не мають заданого TLO, отримують значення 0.

### 3.24.3 G-коди, що впливають на інструменти

G-коди, які використовують або впливають на інформацію про інструмент:

#### 3.24.3.1 Txxx

Наказує апаратному засобу зміни інструментів підготуватися до перемикання на вказаний інструмент xxx.

Обробляється `Interp::convert_tool_select()`.

1. Верстату пропонується підготуватися до перемикання на вибраний інструмент шляхом виклику функції Canon `SELECT_TOOL()` з номером запитуваного інструменту.
   a. (сайканон) Ніякої операції.
   b. (emccanon) Створює повідомлення `EMC_TOOL_PREPARE` із запитуваним номером кишені та надсилає його до Task, який надсилає його до IO. IO отримує повідомлення та просить HAL підготувати кишеню, встановивши `iocontrol.0.tool-prep-pocket`, `iocontrol.0.tool-p` та `iocontrol.0.tool-prepare`. IO потім неодноразово викликає `read_tool_inputs()`, щоб опитувати контакт HAL `iocontrol.0.tool-prepared`, який через HAL сигналізує IO про завершення підготовки інструменту, що запитується, від апаратного забезпечення змінювача інструменту. Коли цей контакт стає True, IO встановлює `emcioStatus.tool.pocket` на номер кишені запитуваного інструменту.

2. Повернувшись до інтерфейсу, `settings->selected_pocket` отримує індекс tooldata запитуваного інструменту *xxx*.

---

**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).

---

#### 3.24.3.2 M6

Наказує змінникy інструментів переключитися на поточний вибраний інструмент (вибраний попередньою командою Txxx).

Обробляється за допомогою `Interp::convert_tool_change()`.

1. Верстату надсилається запит на перемикання на вибраний інструмент шляхом виклику функції Canon `CHANGE_TOOL()` з `settings->selected_pocket` (індексом даних інструменту).
   a. (saicanon) Встановлює `_active_slot` sai на переданий номер кишені. Інформація про інструмент копіюється з вибраної кишені таблиці інструментів (тобто з `_tools[_active_slot]` sai) на шпиндель (тобто `_tools[0]` sai).
   b. (emccanon) Відправляє повідомлення `EMC_TOOL_LOAD` до Task, який відправляє його до IO. IO встановлює `emcioStatus.tool.toolInSpindle` на номер інструменту в кишені, ідентифікованій `emcioStatus.tool.pocketPrepped` (встановлюється Txxx або `SELECT_TOOL()`). Потім він запитує, щоб апаратне забезпечення змінювача інструментів виконало зміну інструменту, встановивши HAL-контакт `iocontrol.0.tool-change` на True. Пізніше `read_tool` IO виявить, що контакт HAL `iocontrol.0.tool_changed` встановлено в значення True, що вказує на те, що змінювач інструменту завершив зміну інструменту. Коли це відбувається, він викликає `load_tool()`, щоб оновити стан верстата.
      i. `load_tool()` з невипадковою конфігурацією змінника інструментів копіює інформацію про інструмент з вибраного гнізда до шпинделя (гніздо 0).
      ii. `load_tool()` з випадковою конфігурацією змінювача інструментів обмінює інформацію про інструмент між гніздом 0 (шпиндель) та вибраним гніздом, а потім зберігає таблицю інструментів.

2. У інтерпретаторі `settings->current_pocket` присвоюється новий індекс tooldata з `settings->sele` (встановлений Txxx). Відповідні пронумеровані параметри (#5400-#5413) оновлюються новою інформацією про інструмент з кишені 0 (шпиндель).

#### 3.24.3.3 G43/G43.1/G49

Застосувати зміщення довжини інструменту. G43 використовує TLO поточного завантаженого інструменту або вказаного інструменту, якщо в блоці вказано H-слово. G43.1 отримує TLO з осьових слів у блоці. G49 скасовує TLO (використовує 0 для зміщення для всіх осей).

Обробляється за допомогою `Interp::convert_tool_length_offset()`.

1. Спочатку створюється `EmcPose`, що містить 9-осьові зміщення, які будуть використовуватися. Для `G43.1` ці зміщення інструменту походять від слів осі в поточному блоці. Для `G43` ці зміщення походять від поточного (інструменту в кишені 0) або від інструменту, зазначеного словом H в блоці. Для G49 всі зміщення дорівнюють 0.

2. Зміщення передаються до функції `USE_TOOL_LENGTH_OFFSET()` програми Canon.

a. (saicanon) Записує TLO в `_tool_offset`.
b. (emccanon) Створює повідомлення `EMC_TRAJ_SET_OFFSET`, що містить зміщення, і надсилає його до Task. Task копіює зміщення до `emcStatus->task.toolOffset` і надсилає їх до Motion за допомогою команди `EMCMOT_SET_OFFSET`. Motion копіює зміщення в `emcmotStatus->tool` де вони використовуються для зміщення майбутніх рухів.

3. Повернувшись до інтерпретації, зміщення записуються в `settings->tool_offset`. Ефективна кишеня записується в `settings->tool_offset_index`, хоча це значення ніколи не використовується

#### 3.24.3.4 G10 L1/L10/L11

Змінює таблицю інструментів.

Обробляється за допомогою `Interp::convert_setup_tool()`.

1. Вибирає номер інструменту з P-слова в блоці та знаходить гніздо для цього інструменту:
   a. У конфігурації невипадкового змінника інструментів це завжди номер гнізда в змінніку інструментів (навіть коли інструмент знаходиться в шпинделі).
   b. При випадковій конфігурації змінювача інструментів, якщо інструмент завантажений, він використовує кишеню 0 (кишеня 0 означає «шпиндель»), а якщо інструмент не завантажений, він використовує номер кишені в змінювачі інструментів. (Ця різниця є важливою.)
2. Визначає, якими мають бути нові зміщення.
3. Нова інформація про інструмент (діаметр, зміщення, кути та орієнтація), разом з номером інструмента та номером гнізда, передається до виклику Canon SET_TOOL_TABLE_ENTRY().
   a. (saicanon) Скопіювати інформацію про новий інструмент у вказане гніздо (у внутрішній таблиці інструментів sai, `_tools`).
   b. (emccanon) Створіть повідомлення `EMC_TOOL_SET_OFFSET` з новою інформацією про інструмент і надішліть його до Task, який передасть його до IO. IO оновлює вказану кишеню у своїй внутрішній копії таблиці інструментів (`emcioStatus.tool.toolTable`), і якщо вказаний інструмент наразі завантажений (його порівнюють з `emcioStatus.tool.toolInSpindle`), то нова інформація про інструмент також копіюється до кишені 0 (шпиндель). (FIXME: це невелика помилка, копіювання повинно відбуватися тільки на не випадкових машинах.) Нарешті, IO зберігає нову таблицю інструментів.
4. Повертаючись до interp, якщо модифікований інструмент наразі завантажений у шпиндель, а верстат не має випадкового змінювача інструментів, то нова інформація про інструмент копіюється з вихідної кишені інструменту до кишені 0 (шпиндель) у копії таблиці інструментів interp, `settings->tool_table`. (Ця копія не потрібна на верстатах з випадковим зміною інструменту, оскільки там інструменти не мають вихідної кишені, і замість цього ми просто оновили інструмент у кишені 0 безпосередньо). Відповідні пронумеровані параметри (#5400-#5413) оновлюються на основі інформації про інструмент у шпинделі (шляхом копіювання інформації з `settings->tool_table` інтерпретатора до `settings->parameters`). (FIXME: це невелика помилка, параметри повинні оновлюватися тільки в тому випадку, якщо було змінено поточний інструмент).
5. Якщо модифікований інструмент наразі завантажений у шпиндель, а конфігурація призначена для невипадкового змінювача інструментів, то інформація про новий інструмент також записується в кишеню 0 таблиці інструментів за допомогою другого виклику SET_TOOL_TABLE_ENTRY(). (Це друге оновлення таблиці інструментів не потрібне на машинах з випадковим змінювачем інструментів, оскільки там інструменти не мають базової комірки, і замість цього ми просто оновили інструмент безпосередньо в комірці 0.)

#### 3.24.3.5 M61

Встановити поточний номер інструменту. Це перемикає внутрішнє представлення LinuxCNC того, який інструмент знаходиться в шпинделі, без фактичного переміщення змінника інструментів або заміни будь-яких інструментів.

Обробляється за допомогою `Interp::convert_tool_change()`.

Канон: `CHANGE_TOOL_NUMBER()`

settings->current_pocket отримує індекс tooldata, який наразі містить інструмент, вказаний аргументом Q-word.

#### 3.24.3.6 G41/G41.1/G42/G42.1

Увімкнути компенсацію радіуса різця (зазвичай називається *компенсація різця*).

Обробляється `Interp::convert_cutter_compensation_on()`.

У інтерпретаторі не відбувається виклик Canon, cutter comp. Використовує таблицю інструментів очікуваним чином: якщо вказано номер інструменту D-word, він шукає номер кишені вказаного інструменту в таблиці, а якщо D-word не вказано, використовує кишеню 0 (шпиндель).

#### 3.24.3.7 G40

Скасувати компенсацію радіуса різця.

Обробляється `Interp::convert_cutter_compensation_off()`.

Немає виклику Canon, корекція різця відбувається в інтерпретаторі. Таблиця інструментів не використовується.

### 3.24.4 Внутрішні змінні стану

Це не вичерпний список! Інформація про інструменти поширюється по всьому LinuxCNC.

#### 3.24.4.1 IO

`emcioStatus` типу `EMC_IO_STAT`

**emcioStatus.tool.pocketPrepped**
: Коли IO отримує сигнал від HAL про завершення підготовки змінювача інструменту (після команди `Txxx`), ця змінна встановлюється на кишеню запитуваного інструменту. Коли IO отримує сигнал від HAL про завершення самої заміни інструменту (після команди `M6`), ця змінна скидається до -1.

**emcioStatus.tool.toolInSpindle**
: Номер інструменту, встановленого на даний момент у шпинделі. Експортовано на вивід HAL `iocontrol.0.tool-number` (s32).

**emcioStatus.tool.toolTable[]**
: Масив структур `CANON_TOOL_TABLE`, довжиною `CANON_POCKETS_MAX`. Завантажується з файлу таблиці інструментів під час запуску і зберігається там після цього. Індекс 0 — це шпиндель, індекси 1-(CANON_POCKETS_MAX-1) — це кишені в змінювачі інструментів. Це повна копія інформації про інструменти, яка зберігається окремо від `settings.tool_table` Interp.

#### 3.24.4.2 між

settings має тип settings, визначений як struct setup_struct у src/emc/rs274ngc/interp_internal

**settings.selected_pocket**
Індекс Tooldata інструмента, обраного востаннє за допомогою Txxx.

**settings.current_pocket**
Вихідний індекс tooldata інструмента, що наразі знаходиться в шпинделі. Іншими словами: з якого індексу tooldata було завантажено інструмент, що наразі знаходиться в шпинделі.

**settings.tool_table[]**
Масив інформації про інструменти. Індекс масиву — це «номер кишені» (або «номер слота»). Кишеня 0 — це шпиндель, кишені з 1 по (CANON_POCKETS_MAX-1) — це кишені змінювача інструментів.

**settings.tool_offset_index**
Невикористано. FIXME: Ймовірно, слід вилучити.

**settings.toolchange_flag**
Interp встановлює це значення як true при виклику функції CHANGE_TOOL() від Canon. Воно перевіряється в Interp::convert_tool_length_offset(), щоб вирішити, який індекс tooldata використовувати для G43 (без H-слова): settings->current_pocket, якщо заміна інструменту ще триває, індекс tooldata 0 (шпиндель), якщо заміна інструменту завершена.

**settings.random_toolchanger**
Встановлюється з змінної INI [EMCIO]RANDOM_TOOLCHANGER під час запуску. Керує різними логіками обробки таблиці інструментів. (IO також зчитує цю змінну INI і змінює свою поведінку на основі неї. Наприклад, під час збереження таблиці інструментів, випадковий змінювач інструментів зберігає інструмент у шпинделі (кишеня 0), але невипадковий змінювач інструментів зберігає кожен інструмент у його «домашній кишені».)

**settings.tool_offset**
Це змінна EmcPose.

- Використовується для обчислення положення в різних місцях.
- Надіслано до Motion через повідомлення EMCMOT_SET_OFFSET. Все, що робить Motion із зміщеннями, — це експортує їх до контактів HAL motion.0.tooloffset.[xyzabcuvw]. FIXME: експортуйте їх з місця, ближчого до таблиці інструментів (ймовірно, io або interp), і видаліть повідомлення EMCMOT_SET_OFFSET.

**settings.pockets_max**
Використовується взаємозамінно з CANON_POCKETS_MAX (#визначена константа, встановлена на 1000 станом на квітень 2020 року). ВИПРАВЛЕННЯ: Ця змінна налаштувань наразі не є корисною та, ймовірно, її слід вилучити.

**settings.tool_table**
Це масив структур CANON_TOOL_TABLE (визначених у src/emc/nml_intf/emctool.h), з CANON_POCKE
записами. Індексується за «номером кишені», також відомим як «номером слота». Індекс 0 — це шпиндель, індекси від 1 до (CANON_POCKETS_MAX-1) — це кишені в змінювачі інструментів. У випадковому змінювачі інструментів номери кишень мають значення. У невипадковому змінювачі інструментів кишені не мають значення; номери кишень у файлі таблиці інструментів ігноруються, а інструменти послідовно призначаються слотам tool_table.

**settings.tool_change_at_g30 , settings.tool_change_quill_up , settings.tool_change_with_spindle_**

Вони встановлюються з INI-змінних у розділі [EMCIO] та визначають, як виконуються зміни інструментів.

## 3.25 Розрахунок суглобів та осей

### 3.25.1 У буфері стану

Буфер стану використовується завданням та інтерфейсами користувача.

ВИПРАВЛЕННЯ: `axis_mask` та `axes` надмірно вказують кількість осей

**`status.motion.traj.axis_mask`**
: Бітова маска з «1» для осей, які присутні, і «0» для осей, які відсутні. X — це біт 0 із значенням $2^0 = 1$, якщо встановлено, Y — це біт 1 із значенням $2^1 = 2$, Z — це біт 2 із значенням 4 тощо. Наприклад, машина з осями X і Z матиме `axis_mask` 0x5, машина XYZ матиме 0x7, а машина XYZB матиме `axis_mask` 0x17.

**`status.motion.traj.axes` (вилучено)**
: Це значення було вилучено у версії LinuxCNC 2.9. Замість нього використовуйте `axis_mask`.

**`status.motion.traj.joints`**
: Кількість шарнірів, які має верстат. Звичайний токарний верстат має 2 шарніри: один приводить в рух вісь X, а другий — вісь Z. Фрезерний верстат XYYZ має 4 шарніри: один приводить в рух вісь X, один — одну сторону осі Y, другий — іншу сторону осі Y, а третій — вісь Z. Фрезерний верстат XYZA також має 4 шарніри.

**`status.motion.axis[EMCMOT_MAX_AXIS]`**
: Масив структур осей `EMCMOT_MAX_AXIS`. `axis[n]` є дійсним, якщо `(axis_mask & (1 << n))` є істинним. Якщо `(axis_mask & (1 << n))` є хибним, то `axis[n]` не існує на цій машині і його слід ігнорувати.

**`status.motion.joint[EMCMOT_MAX_JOINTS]`**
: Масив структур з'єднань `EMCMOT_MAX_JOINTS`. `joint[0]` до `joint[joints-1]` є дійсними, інші не існують на цій машині та мають бути ігноровані.

Наразі в гілці joints-axes все не так, але відхилення від цього дизайну вважаються помилками. Приклад такої помилки можна побачити в обробці осей в src/emc/ini/initraj.cc:loadTraj(). Безсумнівно, їх є більше, і мені потрібна ваша допомога, щоб їх знайти та виправити.

### 3.25.2 У русі

Компонент реального часу контролера руху спочатку отримує кількість з'єднань з параметра часу завантаження `num_joints`. Це визначає, скільки з'єднань HAL-пінів створюється під час запуску.

Кількість суглобів руху можна змінити під час виконання за допомогою команди `EMCMOT_SET_NUM_JOINT` з Task.

Контролер руху завжди працює з осями `EMCMOT_MAX_AXIS`. Він завжди створює дев'ять наборів контактів `axis.*.*`.

# Chapter 4

# Повідомлення NML

Список повідомлень NML.
Детальніше див. `src/emc/nml_intf/emc.hh`.

## 4.1 ОПЕРАТОР

```
EMC_OPERATOR_ERROR_TYPE
EMC_OPERATOR_TEXT_TYPE
EMC_OPERATOR_DISPLAY_TYPE
```

## 4.2 СУГЛОБ

```
EMC_JOINT_SET_JOINT_TYPE
EMC_JOINT_SET_UNITS_TYPE
EMC_JOINT_SET_MIN_POSITION_LIMIT_TYPE
EMC_JOINT_SET_MAX_POSITION_LIMIT_TYPE
EMC_JOINT_SET_FERROR_TYPE
EMC_JOINT_SET_HOMING_PARAMS_TYPE
EMC_JOINT_SET_MIN_FERROR_TYPE
EMC_JOINT_SET_MAX_VELOCITY_TYPE
EMC_JOINT_INIT_TYPE
EMC_JOINT_HALT_TYPE
EMC_JOINT_ABORT_TYPE
EMC_JOINT_ENABLE_TYPE
EMC_JOINT_DISABLE_TYPE
EMC_JOINT_HOME_TYPE
EMC_JOINT_ACTIVATE_TYPE
EMC_JOINT_DEACTIVATE_TYPE
EMC_JOINT_OVERRIDE_LIMITS_TYPE
EMC_JOINT_LOAD_COMP_TYPE
EMC_JOINT_SET_BACKLASH_TYPE
EMC_JOINT_UNHOME_TYPE
EMC_JOINT_STAT_TYPE
```

## 4.3 AXIS

```
EMC_AXIS_STAT_TYPE
```

## 4.4 JOG

```
EMC_JOG_CONT_TYPE
EMC_JOG_INCR_TYPE
EMC_JOG_ABS_TYPE
EMC_JOG_STOP_TYPE
```

## 4.5 TRAJ

```
EMC_TRAJ_SET_AXES_TYPE
EMC_TRAJ_SET_UNITS_TYPE
EMC_TRAJ_SET_CYCLE_TIME_TYPE
EMC_TRAJ_SET_MODE_TYPE
EMC_TRAJ_SET_VELOCITY_TYPE
EMC_TRAJ_SET_ACCELERATION_TYPE
EMC_TRAJ_SET_MAX_VELOCITY_TYPE
EMC_TRAJ_SET_MAX_ACCELERATION_TYPE
EMC_TRAJ_SET_SCALE_TYPE
EMC_TRAJ_SET_RAPID_SCALE_TYPE
EMC_TRAJ_SET_MOTION_ID_TYPE
EMC_TRAJ_INIT_TYPE
EMC_TRAJ_HALT_TYPE
EMC_TRAJ_ENABLE_TYPE
EMC_TRAJ_DISABLE_TYPE
EMC_TRAJ_ABORT_TYPE
EMC_TRAJ_PAUSE_TYPE
EMC_TRAJ_STEP_TYPE
EMC_TRAJ_RESUME_TYPE
EMC_TRAJ_DELAY_TYPE
EMC_TRAJ_LINEAR_MOVE_TYPE
EMC_TRAJ_CIRCULAR_MOVE_TYPE
EMC_TRAJ_SET_TERM_COND_TYPE
EMC_TRAJ_SET_OFFSET_TYPE
EMC_TRAJ_SET_G5X_TYPE
EMC_TRAJ_SET_HOME_TYPE
EMC_TRAJ_SET_ROTATION_TYPE
EMC_TRAJ_SET_G92_TYPE
EMC_TRAJ_CLEAR_PROBE_TRIPPED_FLAG_TYPE
EMC_TRAJ_PROBE_TYPE
EMC_TRAJ_SET_TELEOP_ENABLE_TYPE
EMC_TRAJ_SET_SPINDLESYNC_TYPE
EMC_TRAJ_SET_SPINDLE_SCALE_TYPE
EMC_TRAJ_SET_FO_ENABLE_TYPE
EMC_TRAJ_SET_SO_ENABLE_TYPE
EMC_TRAJ_SET_FH_ENABLE_TYPE
EMC_TRAJ_RIGID_TAP_TYPE
EMC_TRAJ_STAT_TYPE
```

## 4.6 MOTION

```
EMC_MOTION_INIT_TYPE
EMC_MOTION_HALT_TYPE
EMC_MOTION_ABORT_TYPE
EMC_MOTION_SET_AOUT_TYPE
EMC_MOTION_SET_DOUT_TYPE
EMC_MOTION_ADAPTIVE_TYPE
EMC_MOTION_STAT_TYPE
```

## 4.7 TASK

```
EMC_TASK_INIT_TYPE
EMC_TASK_HALT_TYPE
EMC_TASK_ABORT_TYPE
EMC_TASK_SET_MODE_TYPE
EMC_TASK_SET_STATE_TYPE
EMC_TASK_PLAN_OPEN_TYPE
EMC_TASK_PLAN_RUN_TYPE
EMC_TASK_PLAN_READ_TYPE
EMC_TASK_PLAN_EXECUTE_TYPE
EMC_TASK_PLAN_PAUSE_TYPE
EMC_TASK_PLAN_STEP_TYPE
EMC_TASK_PLAN_RESUME_TYPE
EMC_TASK_PLAN_END_TYPE
EMC_TASK_PLAN_CLOSE_TYPE
EMC_TASK_PLAN_INIT_TYPE
EMC_TASK_PLAN_SYNCH_TYPE
EMC_TASK_PLAN_SET_OPTIONAL_STOP_TYPE
EMC_TASK_PLAN_SET_BLOCK_DELETE_TYPE
EMC_TASK_PLAN_OPTIONAL_STOP_TYPE
EMC_TASK_STAT_TYPE
```

## 4.8 TOOL

```
EMC_TOOL_INIT_TYPE
EMC_TOOL_HALT_TYPE
EMC_TOOL_ABORT_TYPE
EMC_TOOL_PREPARE_TYPE
EMC_TOOL_LOAD_TYPE
EMC_TOOL_UNLOAD_TYPE
EMC_TOOL_LOAD_TOOL_TABLE_TYPE
EMC_TOOL_SET_OFFSET_TYPE
EMC_TOOL_SET_NUMBER_TYPE
EMC_TOOL_START_CHANGE_TYPE
EMC_TOOL_STAT_TYPE
```

## 4.9 AUX

```
EMC_AUX_ESTOP_ON_TYPE
EMC_AUX_ESTOP_OFF_TYPE
EMC_AUX_ESTOP_RESET_TYPE
EMC_AUX_INPUT_WAIT_TYPE
EMC_AUX_STAT_TYPE
```

## 4.10 SPINDLE

```
EMC_SPINDLE_ON_TYPE
EMC_SPINDLE_OFF_TYPE
EMC_SPINDLE_INCREASE_TYPE
EMC_SPINDLE_DECREASE_TYPE
EMC_SPINDLE_CONSTANT_TYPE
EMC_SPINDLE_BRAKE_RELEASE_TYPE
EMC_SPINDLE_BRAKE_ENGAGE_TYPE
EMC_SPINDLE_SPEED_TYPE
EMC_SPINDLE_ORIENT_TYPE
EMC_SPINDLE_WAIT_ORIENT_COMPLETE_TYPE
EMC_SPINDLE_STAT_TYPE
```

## 4.11 COOLANT

```
EMC_COOLANT_MIST_ON_TYPE
EMC_COOLANT_MIST_OFF_TYPE
EMC_COOLANT_FLOOD_ON_TYPE
EMC_COOLANT_FLOOD_OFF_TYPE
EMC_COOLANT_STAT_TYPE
```

## 4.12 LUBE

```
EMC_LUBE_ON_TYPE
EMC_LUBE_OFF_TYPE
EMC_LUBE_STAT_TYPE
```

## 4.13 IO (Вхід/Вихід)

```
EMC_IO_INIT_TYPE
EMC_IO_HALT_TYPE
EMC_IO_ABORT_TYPE
EMC_IO_SET_CYCLE_TIME_TYPE
EMC_IO_STAT_TYPE
EMC_IO_PLUGIN_CALL_TYPE
```

## 4.14 Інші

```
EMC_NULL_TYPE
EMC_SET_DEBUG_TYPE
EMC_SYSTEM_CMD_TYPE
EMC_INIT_TYPE
EMC_HALT_TYPE
EMC_ABORT_TYPE
EMC_STAT_TYPE
EMC_EXEC_PLUGIN_CALL_TYPE
```

# Chapter 5

# Стиль кодування

У цьому розділі описано стиль вихідного коду, якому надає перевагу команда LinuxCNC.

## 5.1 Не нашкодь

Під час внесення невеликих змін до коду в стилі, відмінному від описаного нижче, враховуйте локальний стиль кодування. Швидкі зміни від одного стилю кодування до іншого знижують читабельність коду.

Ніколи не реєструйте код після виконання команди ”indent”. Зміни пробілів, внесені командою indent, ускладнюють відстеження історії редагувань файлу.

Не використовуйте редактор, який вносить непотрібні зміни до пробілів (наприклад, замінює 8 пробілів на табуляцію в рядку, який не змінено, або переносить слова в рядках, які не змінено).

## 5.2 Позиції табуляції

Позиція табуляції завжди відповідає 8 пробілам. Не пишіть код, який коректно відображається лише з іншим значенням позиції табуляції.

## 5.3 Відступ

Використовуйте 4 пробіли на рівень відступу. Об’єднання 8 пробілів в одну табуляцію прийнятне, але не обов’язкове.

## 5.4 Встановлення брекетів

Поставте відкриваючу фігурну дужку останньою на рядку, а закриваючу — першою:

```
if (x) {
    // b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''щb''b''ob''b''cb''b''ьb'' b' ↩
        'дb''b''ob''b''pb''b''eb''b''чb''b''нb''b''eb''
}
```

Закриваюча фігурна дужка стоїть на окремому рядку, за винятком випадків, коли за нею йде продовження того ж оператора, тобто *while* в операторі do або *else* в операторі if, як у цьому випадку:

```
do {
    // b''щb''b''ob''b''cb''b''ьb'' b''вb''b''ab''b''жb''b''лb''b''иb''b''вb''b''eb''
} while (x > 0);
```

і

```
if (x == y) {
    // b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''ob''b''дb''b''нb''b''уb'' b' ↩
        'pb''b''ib''b''чb''
} else if (x < y) {
    // b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''щb''b''ob''b''cb''b''ьb'' b' ↩
        'ib''b''нb''b''шb''b''eb''
} else {
    // b''зb''b''pb''b''ob''b''бb''b''иb''b''тb''b''иb'' b''тb''b''pb''b''eb''b''тb''b' ↩
        'юb'' b''pb''b''ib''b''чb''
}
```

Таке розміщення фігурних дужок також мінімізує кількість порожніх (або майже порожніх) рядків, що дозволяє одночасно відображати більшу кількість коду або коментарів у терміналі фіксованого розміру.

## 5.5 Ми

C є спартанською мовою, і так само має бути і ваша система іменування. На відміну від програмістів Modula-2 та Pascal, програмісти C не використовують милі імена на кшталт ThisVariableIsATemporaryCounter. Програміст C назвав би цю змінну «tmp», що набагато простіше писати і не менш зрозуміло.

Однак, описові назви для глобальних змінних є обов'язковими. Виклик глобальної функції «foo» — це кидок у ворота.

Глобальні змінні (які слід використовувати тільки в разі **реальної** необхідності) повинні мати описові імена, як і глобальні функції. Якщо у вас є функція, яка підраховує кількість активних користувачів, ви повинні назвати її «count_active_users()» або подібним чином, але **не** називати її «cntusr()».

Кодування типу функції в імені (так звана угорська нотація) є безглуздим — компілятор і так знає типи і може їх перевірити, а це тільки заплутує програміста. Не дивно, що Microsoft створює програми з помилками.

Імена локальних змінних повинні бути короткими і точними. Якщо у вас є випадковий цілочисельний лічильник циклу, його, ймовірно, слід назвати «i». Називати його «loop_counter» непродуктивно, якщо немає ймовірності його неправильного розуміння. Аналогічно, «tmp» може бути будь-яким типом змінної, яка використовується для зберігання тимчасового значення.

Якщо ви боїтеся переплутати назви локальних змінних, у вас є ще одна проблема, яка називається синдромом дисбалансу функції-гормону росту. Див. наступний розділ.

## 5.6 Функції

Функції повинні бути короткими, зрозумілими та виконувати лише одну дію. Вони повинні поміщатися на одному або двох екранах тексту (розмір екрана ISO/ANSI становить 80x24, як ми всі знаємо), виконувати одну дію та робити її добре.

Максимальна довжина функції обернено пропорційна складності та рівню втягування цієї функції. Отже, якщо у вас є концептуально проста функція, яка складається лише з одного довгого (але простого) оператора case, де вам доводиться виконувати багато дрібних операцій для багатьох різних випадків, то довша функція є цілком прийнятною.

Однак, якщо у вас є складна функція і ви підозрюєте, що не надто обдарований першокурсник середньої школи може навіть не зрозуміти, про що йдеться у цій функції, вам слід ще суворіше дотримуватися максимальних обмежень. Використовуйте допоміжні функції з описовими назвами (ви можете попросити компілятор вбудувати їх, якщо вважаєте, що це критично важливо для продуктивності, і він, ймовірно, зробить це краще, ніж ви).

Іншим показником функції є кількість локальних змінних. Їх не повинно бути більше 5-10, інакше ви робите щось не так. Перегляньте функцію і розділіть її на менші частини. Людський мозок зазвичай може легко відстежувати близько 7 різних речей, а якщо їх більше, він заплутується. Ви знаєте, що ви геніальний, але, можливо, ви хотіли б зрозуміти, що ви зробили через 2 тижні.

## 5.7 Коментування

Коментарі – це добре, але існує також небезпека надмірного коментування. НІКОЛИ не намагайтеся пояснити, ЯК працює ваш код у коментарі: набагато краще написати код так, щоб його **робота** була очевидною, а пояснювати погано написаний код – це марна трата часу.

Як правило, коментарі повинні пояснювати, ЩО робить ваш код, а не ЯК. Коментар у вигляді блоку, що описує функцію, значення, яке вона повертає, та хто її викликає, розміщений над тілом функції, є хорошим варіантом. Також намагайтеся уникати розміщення коментарів всередині тіла функції: якщо функція настільки складна, що вам потрібно окремо коментувати її частини, вам, ймовірно, слід перечитати розділ «Функції» ще раз. Ви можете робити невеликі коментарі, щоб зазначити або попередити про щось особливо розумне (або потворне), але намагайтеся уникати надмірності. Натомість розміщуйте коментарі на початку функції, розповідаючи людям, що вона робить, і, можливо, ЧОМУ вона це робить.

Якщо використовуються коментарі на кшталт /* Виправити мене */ , будь ласка, вкажіть, чому щось потрібно виправити. Після внесення змін до відповідної частини коду видаліть коментар або доповніть його, вказавши, що зміни внесено і їх потрібно протестувати.

## 5.8 Скрипти оболонки та Makefile

Не всі мають однакові інструменти та пакети. Деякі використовують vi, інші — emacs, а дехто взагалі уникає встановлення будь-якого з цих пакетів, віддаючи перевагу легким текстовим редакторам, таким як nano або вбудований у Midnight Commander.

gawk проти mawk - Знову ж таки, не у всіх буде встановлено gawk, mawk займає майже вдесятеро менше місця і при цьому відповідає стандарту POSIX AWK. Якщо знадобиться якась маловідома команда, специфічна для gawk, якої немає в mawk, то скрипт не працюватиме для деяких користувачів. Те саме стосується і mawk. Коротше кажучи, краще використовувати загальне виклик awk, а не gawk або mawk.

## 5.9 Умовні позначення C++

Стилі кодування C++ завжди можуть стати предметом гарячих суперечок (трохи як суперечки між прихильниками emacs і vi). Однак одне можна сказати напевно: єдиний стиль, який використовую усі учасники проекту, забезпечує однорідність і читабельність коду.

Правила іменування: Константи з #defines або переліків повинні бути написані великими літерами. Обґрунтування: Це полегшує виявлення констант часу компіляції в вихідному коді, наприклад, EMC_MESSAGE_TYPE.

Класи та простори імен повинні починатися з великої літери кожного слова та уникати символів підкреслення. Обґрунтування: Ідентифікує класи, конструктори та деструктори, наприклад, GtkWidget.

Методи (або назви функцій) повинні відповідати наведеним вище рекомендаціям щодо C та не повинні містити назви класу. Обґрунтування: Зберігає спільний стиль для вихідних кодів C та C++, наприклад, get_foo_bar().

Однак булеві методи легше читати, якщо в них не використовуються підкреслення і застосовується префікс «is» (не плутати з методами, що маніпулюють булевим значенням). Обґрунтування: ідентифікує значення, що повертається, як TRUE або FALSE і ніщо інше, наприклад, isOpen, isHomed.

НЕ використовуйте *Not* у логічних назвах, це призводить лише до плутанини під час виконання логічних перевірок, наприклад, isNotOnLimit або is_not_on_limit є ПОГАНИМИ.

У назвах змінних слід уникати використання великих літер та підкреслень, за винятком локальних або приватних імен. Слід максимально уникати використання глобальних змінних. Обґрунтування: роз’яснює, які змінні є змінними, а які — методами. Публічні: наприклад, axislimit Приватні: наприклад, maxvelocity_ .

### 5.9.1 Спеціальні правила іменування методів

Терміни get та set слід використовувати там, де доступ до атрибута здійснюється безпосередньо. Обґрунтування: Вказує на призначення функції або методу, наприклад, get_foo set_bar.

Для методів, що використовують логічні атрибути, перевага надається методам set та reset. Обґрунтування: Як і вище. Наприклад, set_amp_enable reset_amp_fault

Математично інтенсивні методи повинні використовувати префікс compute. Обґрунтування: Показує, що це вимагає інтенсивних обчислень і перевантажуватиме процесор. Наприклад, compute_PID

Скорочень в іменах слід уникати, де це можливо. Виняток становлять імена локальних змінних. Обґрунтування: чіткість коду. Наприклад, покажчик є кращим варіантом, ніж ptr, обчислення є кращим варіантом, ніж cmp, порівняння знову є кращим варіантом, ніж cmp.

Перелічення та інші константи можуть мати префікс із загальною назвою типу, наприклад, `enum COLOR { COLOR_RED, COLOR_BLUE };`.

Слід уникати надмірного використання макросів та визначень – перевага надається використанню простих методів або функцій. Обґрунтування: Покращує процес налагодження.

Включення файлів заголовків Файли заголовків повинні бути включені у верхній частині вихідного файлу, а не розкидані по всьому тексту. Вони повинні бути відсортовані та згруповані за їх ієрархічним положенням у системі, причому файли нижчого рівня повинні бути включені першими. Шляхи до файлів заголовків НІКОЛИ не повинні бути абсолютними — замість цього використовуйте прапор -I компілятора для розширення шляху пошуку. Обґрунтування: заголовки можуть бути розташовані не в одному і тому ж місці на всіх системах.

Вказівники та посилання повинні мати символ посилання поруч з іменем змінної, а не з іменем типу. Обґрунтування: Зменшує плутанину, наприклад, `float *x` або `int &i`.

Неявні перевірки на нуль не слід використовувати, окрім булевих змінних, наприклад, `if (spindle_sp != 0)` НЕ `if (spindle_speed)`.

Тільки оператори керування циклом повинні бути включені до конструкції for(), наприклад, `sum = 0; for (i=0; i<10; i++) { sum += value[i]; }`
НЕ: `for (i=0, sum=0; i<10; i++) sum += value[i];`.

Так само слід уникати виконуваних операторів в умовних операторах, наприклад, `if (fd = open(file_` погано.

Слід уникати складних умовних операторів - натомість вводьте тимчасові логічні змінні.

Дужки слід використовувати в математичних виразах часто – не покладайтеся на пріоритет операторів, коли додаткові дужки можуть щось уточнити.

Імена файлів: Джерела та заголовки C++ використовують розширення .cc та .hh. Використання .c та .h зарезервовано для простого C. Заголовки призначені для оголошень класів, методів та структур, а не для коду (якщо функції не оголошені вбудованими).

## 5.10 Стандарти кодування Python

Використовуйте стиль PEP 8 для коду Python.

## 5.11 Стандарти комп’ютерного кодування

У частині оголошення файлу .comp починайте кожне оголошення з першого стовпця. Вставляйте додаткові порожні рядки, якщо вони допомагають групувати пов’язані елементи.

У кодовій частині .comp-файлу дотримуйтесь звичайного стилю кодування на C.

# Chapter 6

# Довідник з розробки графічного інтер

Цей документ є довідником «найкращих практик» для загального використання при розробці екранів.
Хоча з LinuxCNC можна програмувати практично все, використання загальної структури, мови та вимог до конфігурації полегшує перехід між екранами та дозволяє більшій кількості розробників підтримувати їх.
Тим не менш, ніщо в цьому документі не є остаточним.

## 6.1 Мова

Python є на даний момент переважною мовою коду екрану LinuxCNC.
Python має низький поріг входу для нових користувачів, які можуть модифікувати екрани відповідно до своїх потреб.
Python має багатий набір документації, підручників та бібліотек, з яких можна черпати інформацію.
Він вже використовується та інтегрований у системні вимоги LinuxCNC.
Хоча можна використовувати C або C, це суттєво обмежує коло осіб, які можуть їх підтримувати та розвивати.
Краще було б розширити Python модулями C/C для будь-яких функцій, які цього потребують.

## 6.2 Локалізація чисел з плаваючою комою в графічних інтерфей

Різні локалі використовують різні роздільники десяткових знаків і тисяч. Слід уникати використання функцій перетворення рядка в число з плаваючою комою, специфічних для локалі, оскільки вони можуть давати несподівані результати. (Наприклад, текстовий рядок «1.58» в de_DE буде перетворено в 158 за допомогою atof()). Наступні рекомендації (засновані на уникненні двозначності, а не на «правильності» в будь-якій конкретній локалі) рекомендуються при перетворенні числа з плаваючою комою в рядок і навпаки:

- У разі введення даних як десятковий роздільник допускається використання коми (,) або крапки (.), але відхиляються всі введені дані, що містять більше одного з цих символів. Пробіл повинен прийматися, але не є обов’язковим як роздільник тисяч.
- У випадку відображення використовуйте або крапку (.) послідовно, або послідовно використовуйте поточний формат локалізації. Акцент тут робиться на слові «послідовно».

## 6.3 Базова конфігурація

В даний час більшість екранів використовують комбінацію записів INI-файлу та файлу налаштувань для конфігурації своїх функцій.
Текстові INI-файли зазвичай використовуються для загальних налаштувань контролера машини, тоді як текстові файли налаштувань використовуються для властивостей, пов’язаних з графічним інтерфейсом користувача (таких як звуки, розмір, кольори).
Можуть бути й інші файли, що використовуються для перекладів, стилізації та налаштування функцій. Вони значною мірою залежать від базового набору інструментів віджетів.

### 6.3.1 INI [DISPLAY]

Розділ **[DISPLAY]** INI-файлу призначений для визначення налаштувань, пов’язаних з екраном.

#### 6.3.1.1 Дисплей

Найважливішим є вказання імені екрану, який скрипт LinuxCNC буде використовувати для завантажен
Програма екрану зазвичай розпізнає перемикачі, такі як для встановлення повноекранного режиму.
Title призначений для назви вікна, а icon використовується для іконок вікна.

```
[DISPLAY]
DISPLAY = axis
TITLE = XYZA Rotational Axis’
ICON = silver_dragon.png
```

#### 6.3.1.2 Час циклу

Якщо можна налаштувати, ось як встановити час циклу графічного інтерфейсу дисплея.
Часто це швидкість оновлення, а не час очікування між оновленнями.
Значення 100 мс (0,1 с) є типовим налаштуванням, хоча діапазон 50–200 мс також є прийнятним.

```
[DISPLAY]
CYCLE_TIME = 100
```

#### 6.3.1.3 Шляхи до файлів

Якщо ці функції доступні на екрані, ось як вказати шлях, який потрібно використовувати.
Вони повинні посилатися на поточний файл INI, або дозволяти використання символу «~» для домашньої папки, або дозволяти використання абсолютних шляхів.

```
MDI_HISTORY_FILE = mdi_history.txt
PREFERENCE_FILE_PATH = gui.pref
LOG_FILE = gui-log.txt
```

#### 6.3.1.4 Приріст поштовху

Для вибору кроку зазвичай використовуються перемикачі або комбіновані поля.
Лінійний крок може бути виражений в дюймах або міліметрах.
Крок кута вказується в градусах.
Слово «continuous» використовується для позначення безперервного переміщення і, ймовірно, слід додавати навіть якщо воно відсутнє в рядку INI.

```
INCREMENTS = continuous, 10 mm, 1.0 mm, 0.10 mm, 0.01 mm, 1.0 inch, 0.1 inch, 0.01 inch
ANGULAR_INCREMENTS = continuous, .5, 1, 45, 90, 360
```

#### 6.3.1.5 Підказка щодо типу машини

Екран часто потрібно налаштовувати відповідно до типу верстата. Токарні верстати мають різні елементи керування та по-різному відображають DRO. Пінопластові верстати відображають графік по-іншому.
Раніше для цього додавали перемикачі LATHE = 1, FOAM = 1 тощо.

```
MACHINE_TYPE_HINT = LATHE
```

#### 6.3.1.6 Перевизначення

Функція «Перевірка» дозволяє користувачеві регулювати швидкість подачі або швидкість шпинделя на льоту. Зазвичай використовується повзунок або циферблат.
Ці налаштування вказані у відсотках.

```
MAX_FEED_OVERRIDE       = 120
MIN_SPINDLE_0_OVERRIDE    = 50
MAX_SPINDLE_0_OVERRIDE    = 120
```

#### 6.3.1.7 Швидкість поштовху

Більшість екранів мають повзунки для регулювання лінійної та кутової швидкості переміщення.
Ці налаштування слід вказувати в одиницях машини на хвилину для лінійної швидкості та в градусах на хвилину для кутової швидкості.
«За замовчуванням» означає початкову швидкість при першому завантаженні екрана.

```
DEFAULT_LINEAR_VELOCITY =
MIN_LINEAR_VELOCITY =
MAX_LINEAR_VELOCITY =

DEFAULT_ANGULAR_VELOCITY =
MIN_ANGULAR_VELOCITY =
MAX_ANGULAR_VELOCITY =
```

#### 6.3.1.8 Ручне керування шпинделем

Ручне управління шпинделем може здійснюватися за допомогою кнопок, повзунків або регуляторів (або їх комбінацій).
Ви можете встановити обмеження, які є меншими за ті, що може використовувати контролер верстата, шляхом налаштування цих параметрів.
Якщо ваш екран підтримує роботу декількох шпинделів, то він повинен приймати значення, більші за показане «*0*».

```
SPINDLE_INCREMENT = 100
DEFAULT_SPINDLE_0_SPEED = 500
MIN_SPINDLE_0_SPEED = 50
MAX_SPINDLE_0_SPEED = 1000
```

### 6.3.2 INI [MDI_COMMAND]

Деякі екрани використовують кнопки для запуску команд «Macro» NGC.
Їх можна вказати, як показано в цих компактних прикладах.
Команди NGC, розділені двокрапками, виконуються до кінця перед наступною.
Опціональна кома відокремлює текст для кнопки від коду NGC.

```
[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’
```

### 6.3.3 INI [FILTER]

Цей розділ дозволяє налаштувати, які файли відображаються у вікні вибору файлів та які програми-фільтри будуть попередньо обробляти їх вихідні дані перед надсиланням до LinuxCNC.
Розширення мають такий вигляд:
PROGRAM_EXTENSION = .extension,.extension2[пробіл]Опис розширень
Визначення програм-фільтрів мають такий вигляд:
filter extension = програма для запуску

```
[FILTER]
# b’’Kb’’b’’eb’’b’’pb’’b’’yb’’b’’єb’’ b’’тb’’b’’иb’’b’’мb’’, b’’яb’’b’’кb’’b’’ib’’ b’’пb’’b ↩
    ’’pb’’b’’ob’’b’’гb’’b’’pb’’b’’ab’’b’’мb’’b’’иb’’ b’’вb’’b’’ib’’b’’дb’’b’’ob’’b’’бb’’b’ ↩
    ’pb’’b’’ab’’b’’жb’’b’’ab’’b’’юb’’b’’тb’’b’’ьb’’b’’cb’’b’’яb’’ b’’yb’’ b’’фb’’b’’ab’’b’ ↩
    ’йb’’b’’лb’’b’’ob’’b’’вb’’b’’ob’’b’’мb’’b’’yb’’ b’’мb’’b’’eb’’b’’нb’’b’’eb’’b’’дb’’b’ ↩
    ’жb’’b’’eb’’b’’pb’’b’’ib’’:
PROGRAM_EXTENSION = .ngc,.nc,.tap G-Code File (*.ngc,*.nc,*.tap)
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
PROGRAM_EXTENSION = .py Python Script

# b’’3b’’b’’ib’’b’’cb’’b’’тb’’b’’ab’’b’’вb’’b’’лb’’b’’яb’’b’’єb’’ b’’pb’’b’’ob’’b’’зb’’b’ ↩
    ’шb’’b’’иb’’b’’pb’’b’’eb’’b’’нb’’b’’нb’’b’’яb’’ b’’фb’’b’’ab’’b’’йb’’b’’лb’’b’’ib’’b’ ↩
    ’вb’’ b’’дb’’b’’ab’’b’’нb’’b’’иb’’b’’хb’’/b’’вb’’b’’иb’’b’’хb’’b’’ib’’b’’дb’’b’’нb’’b’ ↩
    ’ob’’b’’гb’’b’’ob’’ b’’кb’’b’’ob’’b’’дb’’b’’yb’’ b’’зb’’b’’ib’’ b’’cb’’b’’пb’’b’’eb’’b’ ↩
    ’цb’’b’’ib’’b’’ab’’b’’лb’’b’’ьb’’b’’нb’’b’’ob’’b’’юb’’ b’’пb’’b’’pb’’b’’ob’’b’’гb’’b’ ↩
    ’pb’’b’’ab’’b’’мb’’b’’ob’’b’’юb’’-«b’’фb’’b’’ib’’b’’лb’’b’’ьb’’b’’тb’’b’’pb’’b’’ob’’b’ ↩
    ’мb’’» b’’дb’’b’’лb’’b’’яb’’ b’’вb’’b’’ib’’b’’дb’’b’’ob’’b’’бb’’b’’pb’’b’’ab’’b’’жb’’b’ ↩
    ’eb’’b’’нb’’b’’нb’’b’’яb’’/b’’вb’’b’’иb’’b’’кb’’b’’ob’’b’’нb’’b’’ab’’b’’нb’’b’’нb’’b’ ↩
    ’яb’’:
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python3
```

### 6.3.4 INI [HAL]

Більшість екранів потребують кількох HAL-пінів. Їх потрібно підключити після того, як екран їх створить.

#### 6.3.4.1 Постгі Хаффіл

Ці файли слід запускати один за одним по порядку, після того, як усі виводи графічного інтерфейсу HAL будуть створені.

```
[HAL]
POSTGUI_HALFILE = keypad_postgui.hal
POSTGUI_HALFILE = vfd_postgui.hal
```

#### 6.3.4.2 Постгуй Халкомд

Ці файли слід запускати один за одним по порядку, після того, як будуть запущені всі файли POSTGUI.

```
[HAL]
POSTGUI_HALCMD = show pin qt
POSTGUI_HALCMD = loadusr halmeter
```

## 6.4 Розширена конфігурація

### 6.4.1 Вбудовування елементів графічного інтерфейсу

Дозвіл користувачам самостійно створювати невеликі панелі, які можна вбудувати в головний екран, є поширеною і дуже корисною функцією налаштування. Деякі екрани дозволяють вбудовувати сторонні програми, а інші — тільки панелі на основі набору власних віджетів.
Зазвичай вони вбудовуються у вкладки або віджети бічної панелі.
Ось як описати опціональний заголовок, команду завантаження та назву віджета розташування:

```
EMBED_TAB_NAME=Vismach demo
EMBED_TAB_COMMAND=qtvcp vismach_mill_xyz
EMBED_TAB_LOCATION=tabWidget_utilities
```

### 6.4.2 Діалогові вікна повідомлень користувача

Діалогові вікна користувача використовуються для відображення інформації про імпорт (зазвичай про помилки), яку користувач вважає важливою.
Деякі залишаються відкритими до вирішення проблеми, деякі вимагають підтвердження, інші — вибору «так/ні».
Контакт HAL I/O викликає діалогове вікно, яке скидає контакт I/O і встановлює будь-які контакти виходу відповіді.

```
[DISPLAY]
MESSAGE_BOLDTEXT = This is an information message
MESSAGE_TEXT = This is low priority
MESSAGE_DETAILS = press ok to clear
MESSAGE_TYPE = okdialog status
MESSAGE_PINNAME = bothtest
MESSAGE_ICON = INFO’
```

Цей стиль видає кілька повідомлень, визначених числом.
У цьому прикладі показано 3 можливі повідомлення на основі номера помилки частотного перетворюва

```
[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’

MULTIMESSAGE_VFD_NUMBER = 3
MULTIMESSAGE_VFD_TYPE = status
MULTIMESSAGE_VFD_TITLE = VFD Error: 3
MULTIMESSAGE_VFD_TEXT = This is the longer text FOR Error MESSAGE NUMBER 3.
MULTIMESSAGE_VFD_DETAILS = We should do something about this message.
MULTIMESSAGE_VFD_ICON = WARNING’
```

# Chapter 7

# Збірка LinuxCNC

## 7.1 Вступ

Цей документ описує, як зібрати програмне забезпечення LinuxCNC з вихідного коду. Це в першу чергу корисно, якщо ви є розробником, який модифікує LinuxCNC. Він також може бути корисним, якщо ви є користувачем, який тестує гілки розробників, хоча в цьому випадку у вас також є можливість просто встановити пакети Debian з buildbot (http://buildbot.linuxcnc.org) або як звичайний пакет з вашого дистрибутива Linux (https://tracker.debian.org/pkg/linuxcnc). Звісно, цей розділ також існує, оскільки LinuxCNC є результатом зусиль спільноти. Ми заохочуємо вас долучатися до розробки LinuxCNC. Зазвичай ви хочете самостійно скомпілювати LinuxCNC для негайного функціонального доступу

- до нової розробки LinuxCNC або
- нова розробка, яку ви, можливо, хочете зробити для LinuxCNC або допомогти іншим її завершити.

Наприклад, ви можете переносити LinuxCNC на якийсь новий дистрибутив Linux або, що є досить поширеним, розробник реагує на ваше повідомлення про проблему, виправлення якої ви хочете протестувати. У будь-якому з цих випадків buildbot не зможе допомогти, або допомога буде затримана, оскільки потрібно чекати на перевірку кимось іншим, а ви не хочете чекати, або ви є єдиною іншою особою, яка має конкретне обладнання для тестування коду.

Окрім програм, що керують вашою машиною та зібрані з дерева вихідного коду, ви також можете створювати ті ж PDF- та/або HTML-файли, з якими ви, ймовірно, зустрічалися в Інтернеті. https://linuxc documents/.

Якщо ви хочете зробити свій внесок у LinuxCNC, але не знаєте, з чого почати, серйозно розгляньте можливість участі в розробці документації. Кожен завжди знайде щось, що можна поліпшити — і якщо ви залишите в тексті лише «FIXME: з коментарем» як посилання для себе та інших, щоб повернутися до цього розділу пізніше. Також переклади на інші мови, крім англійської, дуже ймовірно, виграють від вашої уважності на https://hosted.weblate.org/projects/linuxcnc/ .

## 7.2 Завантаження дерева вихідних кодів

Репозиторій git проекту LinuxCNC знаходиться за адресою https://github.com/LinuxCNC/linuxcnc. GitHub — це популярний сервіс хостингу git та веб-сайт для обміну кодом.

Щоб отримати дерево вихідних кодів, у вас є два варіанти:

**Завантажити тар-архів**
На сторінці проекту LinuxCNC в GitHub знайдіть посилання на «релізи» або «теги», натисніть на гіперпосилання на сторінку архіву та завантажте останній файл .tar. Ви побачите, що цей файл стиснутий у форматі .tar.xz або .tar.gz. Цей файл, який зазвичай називають «tarball», є архівом, дуже схожим на .zip. Ваш робочий стіл Linux знатиме, як обробляти цей файл, коли ви двічі клацнете по ньому.

**Підготуйте локальну копію репозиторію LinuxCNC**
Спочатку вам слід встановити інструмент ”git” на ваш комп’ютер, якщо він ще недоступний (`sudo apt install git`). Потім підготуйте локальний екземпляр дерева вихідних кодів наступним чином: .

```
$ git clone https://github.com/LinuxCNC/linuxcnc.git linuxcnc-source-dir
```

. Перший аргумент команди git видає це: Це називається «клоном» репозиторію LinuxCNC. Перевага полягає в тому, що цей локальний клон підтримує обмін інформацією про зміни, які ви можете вирішити виконати в дереві джерел.

GitHub є самостійною інфраструктурою, про яку докладно розповідається в інших джерелах. Якщо ви ще не знаєте про це, то для мотивації скажімо, що GitHub пропонує виконати для вас клонування і зробити цей екземпляр загальнодоступним. GitHub називає такий додатковий екземпляр іншого репозиторію «форком». Ви можете легко (і безкоштовно) створити форк репозиторію LinuxCNC git на GitHub і використовувати його для відстеження та публікації своїх змін. Після створення власного форку LinuxCNC на GitHub клонуйте його на свою машину для розробки і продовжуйте хакерську діяльність, як зазвичай.

Ми, учасники проекту LinuxCNC, сподіваємося, що ви поділитеся з нами своїми змінами, щоб спільнота могла скористатися результатами вашої роботи. GitHub значно спрощує цей процес: після того, як ви доопрацюєте свої зміни та завантажите їх у свій форк на GitHub, надішліть нам запит на витяг (Pull Request).

### 7.2.1 Швидкий старт

Для нетерплячих спробуйте ось це:

```
$ git clone https://github.com/LinuxCNC/linuxcnc.git linuxcnc-source-dir
$ cd linuxcnc-source-dir/src
$ ./autogen.sh
$ ./configure --with-realtime=uspace
$ make
```

Це, ймовірно, не вдасться! Це не робить вас поганою людиною, це просто означає, що вам слід прочитати весь цей документ, щоб дізнатися, як вирішити свої проблеми. Особливо розділ «Задоволення залежностей збірки, задоволення залежностей збірки».

Якщо ви працюєте на системі з підтримкою реального часу (наприклад, інсталяція з образу LinuxCNC Live/Install, див. розділ Realtime нижче), на даний момент потрібен один додатковий крок збірки:

```
$ sudo make setuid
```

Після успішної збірки LinuxCNC настав час запустити тести:

```
$ source ../scripts/rip-environment
$ runtests
```

Це також може не вийти! Прочитайте весь цей документ, особливо розділ Налаштування тестового середовища.

## 7.3 Підтримувані платформи

Проект LinuxCNC орієнтований на сучасні дистрибутиви на базі Debian, включаючи Debian, Ubuntu та Mint. Ми постійно тестуємо на платформах, перелічених за посиланням http://buildbot.linuxc

LinuxCNC базується на більшості інших дистрибутивів Linux, хоча управління залежностями буде більш ручним і менш автоматичним. Патчі для покращення переносимості на нові платформи завжди вітаються.

### 7.3.1 У режимі реального часу

LinuxCNC — це контролер верстатів, який для роботи потребує платформи реального часу. Ця версія LinuxCNC підтримує наступні платформи. Перші три з переліку — це операційні системи реального часу:

**RTAI**
: З https://www.rtai.org. Ядро Linux з патчем RTAI доступне в архіві Debian за адресою https://linuxcn
Інструкції з інсталяції див. у Getting LinuxCNC.

**Xenomai**
: З https://xenomai.org. Вам доведеться самостійно скомпілювати або отримати ядро Xenomai.

**'Preempt-RT**
: З https://rt.wiki.kernel.org. Ядро Linux з патчем Preempt-RT іноді доступне з архіву Debian за адресою https://www.debian.org та з машини Wayback за адресою https://snapshot.debian.org.

**Не в реальному часі**
: LinuxCNC також можна створити та запустити на платформах, що не працюють у режимі реального часу, таких як звичайна інсталяція Debian або Ubuntu без спеціального ядра реального часу.
У цьому режимі LinuxCNC не підходить для керування верстатами, але є корисним для моделювання виконання G-коду та тестування частин системи, що не працюють у режимі реального часу (таких як інтерфейси користувача та деякі типи компонентів і драйверів пристроїв).
Щоб скористатися можливостями LinuxCNC в режимі реального часу, певні частини LinuxCNC повинні працювати з правами суперкористувача. Щоб увімкнути права суперкористувача для цих частин, виконайте цю додаткову команду після команди `make`, яка збирає LinuxCNC:

```
$ sudo make setuid
```

## 7.4 Режими збірки

Існує два способи зібрати LinuxCNC: зручний для розробників режим "запуску на місці" та зручний для користувача режим пакування Debian.

### 7.4.1 Будівництво для запуску на місці

У збірці Run-In-Place програми LinuxCNC компілюються з вихідного коду, а потім запускаються безпосередньо з каталогу збірки. Нічого не встановлюється за межами каталогу збірки. Це швидко і просто, а також підходить для швидкої ітерації змін. Набір тестів LinuxCNC працює тільки в збірці Run-In-Place. Більшість розробників LinuxCNC в основному використовують цей режим для збірки.

Збірка для Run-In-Place виконує кроки, описані в розділі Швидкий старт на початку цього документа, можливо, з різними аргументами для `src/configure` та `make`.

#### 7.4.1.1 Аргументи `src/configure`

Скрипт `src/configure` налаштовує спосіб компіляції вихідного коду. Він приймає багато необов'язкови аргументів. Перерахуйте всі аргументи `src/configure`, виконавши наступне:

```
$ cd linuxcnc-source-dir/src
$ ./configure --help
```

Найчастіше використовуються такі аргументи:

**'--with-realtime=uspace**
: Створюйте для будь-якої платформи реального часу або для нереального часу. Отримані виконувані файли LinuxCNC будуть працювати як на ядрі Linux з патчами Preempt-RT (забезпечую управління машиною в реальному часі), так і на звичайному (без патчів) ядрі Linux (забезпечуючи симуляцію G-коду, але без управління машиною в реальному часі).

```
b''Яb''b''кb''b''щb''b''ob'' b''вb''b''cb''b''тb''b''ab''b''нb''b''ob''b''вb''b''лb''b' ↩
    'eb''b''нb''b''ob'' b''фb''b''ab''b''йb''b''лb''b''иb'' b''pb''b''ob''b''зb''b' ↩
    'pb''b''ob''b''бb''b''кb''b''иb'' b''дb''b''лb''b''яb'' Xenomai (b''зb''b''ab''b' ↩
    'зb''b''вb''b''иb''b''чb''b''ab''b''йb'' b''зb'' b''пb''b''ab''b''кb''b''eb''b' ↩
    'тb''b''ab'' libxenomai-dev) b''ab''b''бb''b''ob'' RTAI (b''зb''b''ab''b''зb''b' ↩
    'вb''b''иb''b''чb''b''ab''b''йb'' b''зb'' b''пb''b''ab''b''кb''b''eb''b''тb''b' ↩
    'ab'' b''зb'' b''нb''b''ab''b''зb''b''вb''b''ob''b''юb'', 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'' «rtai- ↩
    modules»), b''тb''b''ab''b''кb''b''ob''b''жb'' b''бb''b''yb''b''дb''b''eb'' b''вb'' ↩
    b''вb''b''ib''b''мb''b''кb''b''нb''b''eb''b''нb''b''ob'' b''пb''b''ib''b''дb''b' ↩
    'тb''b''pb''b''иb''b''мb''b''кb''b''yb'' b''цb''b''иb''b''xb'' b''яb''b''дb''b' ↩
    'pb''b''ab'' 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''.
```

**'--with-realtime=/usr/realtime-$VERSION**
: Створіть для платформи RTAI реального часу, використовуючи стару модель «ядра реального часу». Для цього необхідно, щоб ядро RTAI та модулі RTAI були встановлені в `/usr/realtime-$VERS` Отримані виконувані файли LinuxCNC будуть працювати тільки на вказаному ядрі RTAI. Починаючи з LinuxCNC 2.7, це забезпечує найкращу продуктивність у реальному часі.

**'--enable-build-documentation**
: Створіть документацію, крім виконуваних файлів. Цей параметр значно збільшує час, необхідний для компіляції, оскільки створення документації є досить трудомістким процесом. Якщо ви не працюєте над документацією, ви можете пропустити цей параметр.

**'--disable-build-documentation-translation**
: Вимкніть створення перекладеної документації для всіх доступних мов. Створення перекладеної документації займає дуже багато часу, тому рекомендується пропустити цей крок, якщо він не є дійсно необхідним.

#### 7.4.1.2 `make` аргументи

Команда `make` приймає два корисних необов'язкових аргументи.

**Паралельна компіляція**
: `make` приймає опціональний аргумент `-j` *N* (де *N* — число). Це дозволяє виконувати паралельну компіляцію з *N* одночасними процесами, що може значно прискорити процес побудови.

  Корисним значенням для *N* є кількість процесорів у вашій системі збірки.

Ви можете дізнатися кількість процесорів, запустивши `nproc`.

**Побудова лише конкретної цілі**

Якщо ви хочете створити лише певну частину LinuxCNC, ви можете вказати назву того, що ви хочете створити, у командному рядку `make`. Наприклад, якщо ви працюєте над компонентом під назвою `froboz`, ви можете створити його виконуваний файл, виконавши:

```
$ cd linuxcnc-source-dir/src
$ make ../bin/froboz
```

### 7.4.2 Збірка пакетів Debian

При створенні пакетів Debian програми LinuxCNC компілюються з вихідного коду, а потім зберігаються в пакеті Debian разом з інформацією про залежності. Цей процес за замовчуванням також включає створення документації, що займає багато часу через великий обсяг вводу-виводу для багатьох мов, але це можна пропустити. Потім LinuxCNC встановлюється як частина цих пакетів на тих самих машинах або на будь-якій машині з тією ж архітектурою, на яку копіюються файли .deb. LinuxCNC не можна запустити, поки пакети Debian не будуть встановлені на цільовій машині, а потім виконувані файли не стануть доступними в /usr/bin і /usr/lib, як і інше звичайне програмне забезпечення системи.

Цей режим компіляції в першу чергу корисний при пакуванні програмного забезпечення для доставки кінцевим користувачам, а також при компіляції програмного забезпечення для машини, на якій не встановлено середовище компіляції або яка не має доступу до Інтернету.

Для нетерплячих спробуйте ось це:

```
$ sudo apt-get install build-essential
$ git clone https://github.com/LinuxCNC/linuxcnc.git linuxcnc-source-dir
$ cd linuxcnc-source-dir/src
$ ./debian/configure
$ sudo apt-get build-dep .
$ DEB_BUILD_OPTIONS=nocheck dpkg-buildpackage -uc -B
```

Збірка пакетів Debian виконується за допомогою інструменту `dpkg-buildpackage`, що постачається з пакетом `dpkg-dev`. Його виконання має ряд передумов, детально описаних нижче: * має бути встановлена загальна інфраструктура збірки, тобто компілятори тощо. * Потрібно встановити залежності часу збірки, тобто файли заголовків для зовнішніх бібліотек коду, що використовуються, як описано в розділі Задоволення залежностей збірки. * Файл у папці debian має бути повним і описувати пакет

Інструменти для збірки зібрані у віртуальний пакет під назвою `build-essential`. Щоб його встановити, виконайте:

```
$ sudo apt-get install build-essential
```

Після виконання цих передумов збирання пакетів Debian складається з двох кроків.

Першим кроком є генерація скриптів та метаданих пакета Debian з репозиторію git, виконавши наступну команду:

```
$ cd linuxcnc-dev
$ ./debian/configure
```

---

**Note**

Скрипт `debian/configure` відрізняється від скрипта `src/configure`!
`debian/configure` приймає аргументи залежно від платформи, на якій ви будуєте/для якої ви будуєте, див. розділ аргументи debian/configure. За замовчуванням LinuxCNC працює в просторі користувача (”uspace”), очікуючи, що ядро preempt_rt мінімізує затримки.

---

Після налаштування скриптів та метаданих пакета Debian, зберіть пакет, виконавши команду `dpkg-buildpackage`:

```
$ dpkg-buildpackage -b -uc'
```

---

**Note**
`dpkg-buildpackage` потрібно запускати з кореневого каталогу дерева джерел, який ви, можливо, назвали `linuxcnc-source-dir`, **а не** з каталогу `linuxcnc-source-dir/debian`. `dpkg-buildpackage` приймає опціональний аргумент ``-j`N` (де *N* — число). Це дозволяє виконувати кілька завдань одночасно.

---

#### 7.4.2.1 Аргументи `debian/configure` в LinuxCNC

Дерево джерел LinuxCNC має каталог debian з усією інформацією про те, як слід будувати пакет Debian, але деякі ключові файли в ньому поширюються лише як шаблони. Скрипт `debian/configure` готує ці інструкції з побудови для звичайних утиліт пакування Debian і тому повинен бути запущений перед `dpkg-checkbuilddeps` або `dpkg-buildpackage`.

Скрипт `debian/configure` приймає один аргумент, який вказує на базову платформу реального часу або нереального часу для збірки. Звичайні значення для цього аргументу:

**`no-docs`**
: Пропустити будівельну документацію.

**`uspace`**
: Налаштуйте пакет Debian для роботи в режимі реального часу Preempt-RT або для роботи не в режимі реального часу (ці два варіанти сумісні).

**`noauto` , `rtai` , '`xenomai`'**
: Зазвичай списки RTOS для підтримки uspace realtime виявляються автоматично. Однак, якщо ви бажаєте, ви можете вказати один або декілька з них після `uspace`, щоб увімкнути підтримку цих RTOS. Або, щоб вимкнути автоматичне виявлення, вкажіть `noauto`.

  Якщо вам потрібен лише традиційний RTAI "модуль ядра" в режимі реального часу, використовуйт
  замість цього `-r` або `$KERNEL_VERSION`.

**`rtai=<package name>`**
: Якщо пакет розробки для RTAI, lxrt, не починається з "rtai-modules", або якщо перший такий пакет, що відображається в списку apt-cache, не є потрібним, тоді явно вкажіть назву пакета.

**`-r`**
: Налаштуйте пакет Debian для поточного запущеного ядра RTAI. Щоб це працювало, на вашому комп'ютері для збірки має бути запущено ядро RTAI!

**`$KERNEL_VERSION`**
: Налаштуйте пакет Debian для вказаної версії ядра RTAI (наприклад, «3.4.9-rtai-686-pae»). На вашій машині для компіляції повинен бути встановлений відповідний пакет Debian із заголовками ядра, наприклад «linux-headers-3.4.9-rtai-686-pae». Зверніть увагу, що ви можете *скомпілювати* LinuxCNC у цій конфігурації, але якщо ви не використовуєте відповідне ядро RTAI, ви не зможете *запустити* LinuxCNC, включаючи набір тестів.

#### 7.4.2.2 Задоволення залежностей збірки

На платформах на базі Debian ми надаємо метадані пакетів, які знають, які зовнішні програмні пакети потрібно встановити для побудови LinuxCNC. Вони називаються *залежностями побудови* LinuxCNC, тобто тими пакетами, які повинні бути доступними, щоб

- збірка успішна, і
- конструкцію можна зібрати відтворювано.

Ви можете використовувати ці метадані, щоб легко скласти список необхідних пакетів, яких бракує у вашій системі збірки. Спочатку перейдіть до дерева джерел LinuxCNC і запустіть його стандартну самоконфігурацію, якщо це ще не зроблено:

```
$ cd linuxcnc-dev
$ ./debian/configure
```

Це підготує файл debian/control, який містить списки пакетів Debian, що будуть створені, з залежностями виконання для цих пакетів, а також, для наших цілей, залежностями побудови для цих пакетів, що будуть створені.

Найпростіший спосіб встановити всі залежності збірки – це просто виконати (з того ж каталогу):

```
sudo apt-get build-dep .
```

що встановить усі необхідні залежності, які ще не встановлені, але доступні. Символ «.» є частиною командного рядка, тобто інструкцією для отримання залежностей для даного дерева джерел, а не для залежностей іншого пакета. На цьому встановлення залежностей для побудови завершено.

У решті цього розділу описано напівручний підхід. Список залежностей у debian/control є довгим, і порівнювати поточний стан вже встановлених пакетів з ним є нудним заняттям. Системи Debian надають програму під назвою `dpkg-checkbuilddeps`, яка аналізує метадані пакета і порівнює пакети, вказані як залежності для побудови, зі списком встановлених пакетів, і повідомляє вам, чого бракує.

Спочатку встановіть програму `dpkg-checkbuilddeps`, виконавши команду:

```
$ sudo apt-get install dpkg-dev
```

Це створює файл `debian/control` у форматі yaml, який можна читати користувачем і який містить список залежностей збірки у верхній частині. Ви можете використовувати ці метадані, щоб легко перелічити необхідні пакети, яких бракує у вашій системі збірки. Ви можете вирішити перевірити ці файли вручну, якщо добре розумієте, що вже встановлено.

Крім того, системи Debian надають програму під назвою `dpkg-checkbuilddeps`, яка аналізує метадані пакета і порівнює пакети, вказані як залежності для побудови, зі списком встановлених пакетів, і повідомляє вам, чого не вистачає. Також `dpkg-buildpackage` повідомить вас про те, чого не вистачає, і це повинно бути достатньо. Однак вона повідомляє про відсутні залежності для побудови лише після автоматичного застосування патчів у каталозі `debian/patches` (якщо такі є). Якщо ви новачок у Linux та управлінні версіями git, для уникнення ускладнень краще почати з чистого аркуша.

Програму `dpkg-checkbuilddeps` (також з пакета dpkg-dev, який встановлюється як частина залежностей build-essential) можна попросити виконати свою роботу (зверніть увагу, що її потрібно запускати з каталогу `linuxcnc-source-dir`, **а не** з `linuxcnc-source-dir/debian`):

```
$ dpkg-checkbuilddeps
```

Він видасть список пакетів, необхідних для збірки LinuxCNC на вашій системі, але ще не встановлених. Тепер ви можете встановити відсутні залежності збірки

**вручну**
: Встановіть їх усі за допомогою команди `sudo apt-get install`, а потім вкажіть назви пакетів. Ви можете запустити `dpkg-checkbuilddeps` у будь-який час, щоб переглянути список відсутніх пакетів, що не вплине на дерево вихідного коду.

**автоматизований**
Виконайте команду `sudo apt build-dep`.

Якщо ви сумніваєтеся щодо того, що може надавати певний пакет build-dep, перегляньте опис пакета за допомогою ``apt-cache show`` *packagename*.

#### 7.4.2.3 Параметри для `dpkg-buildpackage`

Для створення типового пакета Debian потрібно запустити dpkg-buildpackage без будь-яких аргументів Як зазначено вище, до команди передаються дві додаткові опції. Як і для всіх хороших інструментів Linux, на сторінці man містяться всі деталі за допомогою команди `man dpkg-buildpackage`.

**`-uc`**
Не підписуйте цифровим підписом отримані бінарні файли. Ви можете підписати свої пакети своїм ключем GPG, тільки якщо хочете розповсюджувати їх серед інших. Якщо ця опція не встановлена і ви не підписуєте пакет, це не вплине на файл .deb.

**`-b`**
Компілює тільки пакети, що залежать від архітектури (такі як бінарні файли `linuxcnc` та графічні інтерфейси). Це дуже корисно, щоб уникнути компіляції того, що не залежить від апаратного забезпечення. Для LinuxCNC це документація, яка в будь-якому випадку доступна в Інтернеті.

Якщо у вас виникнуть труднощі під час компіляції, перегляньте онлайн-форум LinuxCNC.

Наразі з’являється підтримка змінної середовища DEB_BUILD_OPTIONS. Встановіть її на

**`nodocs`**
Щоб пропустити збірку документації, бажано використовувати прапорець `-B` для dpkg-buildpackag

**`nocheck`**
щоб пропустити самотестування процесу побудови LinuxCNC. Це економить час і зменшує потребу в декількох програмних пакетах, які можуть бути недоступними для вашої системи, зокрема xvfb. Не слід встановлювати цю опцію, щоб отримати додаткову впевненість у тому, що ваша побудова працюватиме як очікується, якщо ви не стикаєтеся з суто технічними труднощами, пов’язаними із залежностями програмного забезпечення, що використовується для тестування.

Змінну середовища можна встановити разом із виконанням команди, наприклад.

```
DEB_BUILD_OPTIONS=nocheck dpkg-buildpackage -uc -B
```

поєднав би всі варіанти, представлені в цьому розділі.

#### 7.4.2.4 Встановлення самостійно зібраних пакетів Debian

Пакет Debian можна впізнати за розширенням .deb. Інструмент для його встановлення, `dpkg`, є частиною кожної інсталяції Debian. Файли .deb, створені за допомогою `dpkg-buildpackage`, знаходяться в каталозі над linuxcnc-source-dir, тобто в `...` Щоб побачити, які файли містяться в пакеті, запустіть

```
dpkg -c ../linuxcnc-uspace*.deb
```

Версія LinuxCNC буде частиною імені файлу, яке повинно відповідати зірочці. Можливо, на екрані не вміститься весь список файлів. Якщо ви не можете прокрутити екран термінала вгору, додайте до команди `| more`, щоб її вихідні дані були передані через так званий «пейджер». Вийдіть з програми за допомогою «q».

Щоб встановити пакети, запустіть

```
sudo dpkg -i ../linuxcnc*.deb
```

## 7.5 Налаштування середовища

У цьому розділі описано спеціальні кроки, необхідні для налаштування машини для запуску програм LinuxCNC, включаючи тести.

### 7.5.1 Збільште ліміт заблокованої пам’яті

LinuxCNC намагається поліпшити свою затримку в реальному часі, блокуючи пам’ять, яку він використовує, в оперативній пам’яті. Це робиться для того, щоб запобігти переміщенню LinuxCNC на диск операційною системою, що негативно вплинуло б на затримку. Зазвичай блокування пам’яті в оперативній пам’яті не схвалюється, і операційна система встановлює суворі обмеження на обсяг пам’яті, який користувач може заблокувати.

При використанні платформи реального часу Preempt-RT LinuxCNC працює з достатніми правами, щоб самостійно підвищити обмеження блокування пам’яті. При використанні платформи реального часу RTAI він не має достатніх прав, і користувач повинен підвищити обмеження блокування пам’яті.

Якщо LinuxCNC відображає таке повідомлення під час запуску, проблема полягає в налаштованому системою ліміті заблокованої пам’яті:

```
RTAPI: b’’Пb’’b’’Ob’’b’’Mb’’b’’Иb’’b’’Лb’’b’’Kb’’b’’Ab’’: b’’нb’’b’’eb’’ b’’вb’’b’’дb’’b’ ↩
    ’ab’’b’’лb’’b’’ob’’b’’cb’’b’’яb’’ b’’зb’’b’’іb’’b’’cb’’b’’тb’’b’’ab’’b’’вb’’b’’иb’’b’ ↩
    ’тb’’b’’иb’’ shmem
RTAPI: b’’Лb’’b’’іb’’b’’мb’’b’’іb’’b’’тb’’ b’’зb’’b’’ab’’b’’бb’’b’’лb’’b’’ob’’b’’кb’’b’ ↩
    ’ob’’b’’вb’’b’’ab’’b’’нb’’b’’ob’’b’’їb’’ b’’пb’’b’’ab’’b’’мb’’’b’’яb’’b’’тb’’b’’іb’’ b’ ↩
    ’cb’’b’’тb’’b’’ab’’b’’нb’’b’’ob’’b’’вb’’b’’иb’’b’’тb’’b’’ьb’’ 32 b’’Kb’’b’’Бb’’, b’’pb’’ ↩
    b’’eb’’b’’кb’’b’’ob’’b’’мb’’b’’eb’’b’’нb’’b’’дb’’b’’ob’’b’’вb’’b’’ab’’b’’нb’’b’’ob’’ b’ ↩
    ’щb’’b’’ob’’b’’нb’’b’’ab’’b’’йb’’b’’мb’’b’’eb’’b’’нb’’b’’шb’’b’’eb’’ 20480 b’’Kb’’b’ ↩
    ’Бb’’.
```

Щоб вирішити цю проблему, додайте файл з назвою `/etc/security/limits.d/linuxcnc.conf` (як root) за допомогою улюбленого текстового редактора (наприклад, `sudo gedit /etc/security/limits.`
Файл повинен містити такий рядок:

```
* - memlock 20480’
```

Вийдіть із системи та знову ввійдіть, щоб зміни набули чинності. Перевірте, чи збільшено ліміт блокування пам’яті, за допомогою такої команди:

```
$ ulimit -l
```

## 7.6 Розробка на Gentoo

Створення на базі Gentoo можливе, але не підтримується. Переконайтеся, що ви використовуєте профіль для настільних комп’ютерів. Цей проект використовує набір віджетів Tk, asciidoc та має деякі інші залежності. Вони повинні бути встановлені як root:

```
~ # euse -E tk imagequant
~ # emerge -uDNa world
~ # emerge -a dev-libs/libmodbus dev-lang/tk dev-tcltk/bwidget dev-tcltk/tclx
~ # emerge -a dev-python/pygobject dev-python/pyopengl dev-python/numpy
~ # emerge -a app-text/asciidoc app-shells/bash-completion
```

Ви можете повернутися до звичайного користувача для більшої частини решти встановлення. Як цей користувач, створіть віртуальне середовище для pip, а потім встановіть пакети pip:

```
~/src $ python -m venv   --system-site-packages ~/src/venv
~/src $ . ~/src/venv/bin/activate
(venv) ~/src $ pip install yapps2
(venv) ~/src $
```

Тоді ви можете продовжувати як завжди:

```
(venv) ~/src $ git clone https://github.com/LinuxCNC/linuxcnc.git
(venv) ~/src $ cd linuxcnc
(venv) ~/src $ cd src
(venv) ~/src $ ./autogen.sh
(venv) ~/src $ ./configure --enable-non-distributable=yes
(venv) ~/src $ make
```

Немає потреби запускати ”make suid”, просто переконайтеся, що ваш користувач знаходиться в групі ”dialout”. Щоб запустити linuxcnc, ви повинні бути у віртуальному середовищі Python та налаштувати середовище linuxcnc:

```
~ $ . ~/src/venv/bin/activate
(venv) ~ $ . ~/src/linuxcnc/scripts/rip-environment
(venv) ~ $ ~/src/linuxcnc $ scripts/linuxcnc
```

## 7.7 Варіанти перегляду репозиторію git

Інструкції «Швидкий старт» у верхній частині цього документа клонують наше git-репозиторій за адресою https://github.com/LinuxCNC/linuxcnc.git. Це найшвидший і найпростіший спосіб розпочати роботу. Однак є й інші варіанти, які варто врахувати.

### 7.7.1 Зробіть форк на GitHub

Репозиторій проекту LinuxCNC git знаходиться за адресою https://github.com/LinuxCNC/linuxcnc. GitHub — це популярний сервіс хостингу git та веб-сайт для обміну кодом. Ви можете легко (і безкоштовно) створити форк (другий екземпляр, що містить копію, яку ви контролюєте) репозиторію LinuxCNC git на GitHub. Потім ви можете використовувати цей форк для відстеження та публікації своїх змін, отримання коментарів до своїх змін та прийняття патчів від спільноти. .

Після створення власного форку LinuxCNC на GitHub, клонуйте його на свій комп’ютер розробника та продовжуйте хакінг як завжди.

Ми, учасники проекту LinuxCNC, сподіваємося, що ви поділитеся з нами своїми змінами, щоб спільнота могла скористатися результатами вашої роботи. GitHub значно спрощує цей процес: після того, як ви доопрацюєте свої зміни та завантажите їх у свій форк GitHub, надішліть нам запит на витяг (Pull Request).

# Chapter 8

# Додавання елементів вибору конфігур

Приклади конфігурацій можна додати до засобу вибору конфігурацій двома способами:

- Допоміжні програми — Програми, встановлені незалежно за допомогою пакета deb, можуть розміщувати підкаталоги конфігурації у вказаному системному каталозі. Ім’я каталогу вказується за допомогою скрипта оболонки linuxcnc_var:

```
$ linuxcnc_var LINUXCNC_AUX_EXAMPLES
/usr/share/linuxcnc/aux_examples
```

- Налаштування часу виконання — селектор конфігурації також може пропонувати підкаталоги конфігурації, вказані під час виконання за допомогою експортованої змінної середовища (LINUXCNC_AUX_CONFIGS). 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 LINUXCNC_AUX_CONFIGS=~/myconfigs:/opt/otherconfigs
```

# Chapter 9

# Внесок у LinuxCNC

## 9.1 Вступ

Цей документ містить інформацію для розробників про інфраструктуру LinuxCNC та описує найкращі практики щодо внесення оновлень коду та документації до проєкту LinuxCNC.

У цьому документі термін «джерело» означає як вихідний код програм і бібліотек, так і вихідний текст документації.

## 9.2 Спілкування між розробниками LinuxCNC

Два основні способи спілкування розробників проектів один з одним:

- Через IRC, за адресою #linuxcnc-devel або на Libera.chat.
- Електронною поштою, на список розсилки розробників

## 9.3 Проєкт LinuxCNC Source Forge

Ми використовуємо Source Forge для списки розсилки.

## 9.4 Система контролю версій Git

Весь вихідний код LinuxCNC зберігається в система контролю версій Git.

### 9.4.1 Офіційний Git-репозиторій LinuxCNC

Офіційний git-репозиторій LinuxCNC знаходиться за адресою https://github.com/linuxcnc/linuxcnc/-

Будь-хто може отримати копію вихідного коду LinuxCNC лише для читання через git:

```
git clone https://github.com/linuxcnc/linuxcnc linuxcnc-dev’
```

Якщо ви розробник із доступом до push-розсилок, дотримуйтесь інструкцій github щодо налаштування репозиторію, з якого ви можете надсилати дані.

Зверніть увагу, що команда clone розміщує локальне сховище LinuxCNC у каталозі з назвою `linuxcnc-dev`, а не у стандартному каталозі `linuxcnc`. Це пов'язано з тим, що програмне забезпечення LinuxCNC за замовчуванням очікує, що конфігурації та програми G-коду будуть розміщені у каталозі з назвою `$HOME/linuxcnc`, і наявність там також сховища git може викликати плутанину.

Проблеми та запити на впровадження (скорочені PR) вітаються на GitHub: . https://github.com/-LinuxCNC/linuxcnc/issues . https://github.com/LinuxCNC/linuxcnc/pulls

### 9.4.2 Використання Git у проєкті LinuxCNC

Ми використовуємо робочі процеси git "злиття вгору" та "тематичні гілки", описані тут:

https://www.kernel.org/pub/software/scm/git/docs/gitworkflows.html

У нас є гілка розробки під назвою `master`, та одна або декілька стабільних гілок з назвами типу `2.6` та `2.7`, що вказують на номер версії релізів, які ми з неї створюємо.

Виправлення помилок вносяться в найстарішу відповідну стабільну гілку, яка потім об'єднується з наступною новою стабільною гілкою і так далі аж до `master`. Автор виправлення помилки може самостійно виконати об'єднання або доручити це іншому користувачеві.

Нові функції зазвичай додаються до гілки «master», але деякі типи функцій (зокрема, добре ізольовані драйвери пристроїв та документація) можуть (на розсуд менеджерів стабільної гілки) додаватися до стабільної гілки та об'єднуватися так само, як і виправлення помилок.

### 9.4.3 навчальні посібники з git

В інтернеті є багато чудових безкоштовних навчальних посібників з git.

Перше місце, куди варто звернутися, — це, мабуть, сторінка довідки «gittutorial». Цю сторінку довідки можна відкрити, виконавши команду «man gittutorial» у терміналі (якщо у вас встановлені сторінки довідки git). Gittutorial та супутня документація також доступні в Інтернеті за адресою:

- навчальний посібник з git: https://www.kernel.org/pub/software/scm/git/docs/gittutorial.html
- Підручник з git 2: https://www.kernel.org/pub/software/scm/git/docs/gittutorial-2.html
- Щоденний git з приблизно 20 командами: https://www.kernel.org/pub/software/scm/git/docs/-giteveryday.html
- Посібник користувача Git: https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

Для більш детальної документації git дивіться книгу "Pro Git": https://git-scm.com/book

Ще один рекомендований онлайн-посібник – «Git для ледачих»: https://wiki.spheredev.org/index.php/-Git_for_the_lazy

## 9.5 Огляд процесу

Загальний огляд того, як вносити зміни до вихідного коду, виглядає так:

- Зв'яжіться з розробниками проєкту та повідомте нам, що ви хакуєте. Поясніть, що ви робите та чому.
- Клонуйте репозиторій git.

- Внесіть зміни у місцевому відділенні.
- Додавання документації та writing tests є важливою частиною додавання нової функції. Інакше інші не знатимуть, як користуватися вашою функцією, а якщо інші зміни порушать її роботу, це може залишитися непоміченим без тестування.
- Поділіться своїми змінами з іншими розробниками проекту одним із таких способів:
  - Надішліть свою гілку на github та створіть пул-реквест на github за адресою https://github.com/-linuxcnc/linuxcnc (для цього потрібен обліковий запис github), або
  - Розмістіть свою гілку на публічно видимому git-репозиторії (наприклад, github, на вашому власному публічно доступному сервері тощо) та поділіться цим розташуванням у списку розсилки emc-developers, або
  - Надсилайте ваші коміти електронною поштою до списку розсилки LinuxCNC-developers (<emc-developers@lists.sourceforge.net>) (використовуйте `git format-patch` для створення патчів).
- Захисник вашого патчу:
  - Поясніть, яку проблему він вирішує та чому його слід включити до LinuxCNC.
  - Будьте відкриті до запитань та відгуків від спільноти розробників.
  - Нерідко трапляється, що патч проходить кілька редакцій, перш ніж його приймають.

## 9.6 конфігурація git

Щоб бути включеними до вихідного коду LinuxCNC, коміти повинні мати правильні поля Author, що ідентифікують автора коміту. Хороший спосіб забезпечити це — встановити глобальну конфігурацію git:

```
git config --global user.name ”b’’Bb’’b’’ab’’b’’шb’’b’’eb’’ b’’пb’’b’’ob’’b’’вb’’b’’нb’’b’ ↩
    ’eb’’ b’’ib’’b’’мb’’’b’’яb’’”
git config --global user.email ”you@example.com”
```

Використовуйте своє справжнє ім’я (не псевдонім) та адресу електронної пошти без розкриття інформації.

## 9.7 Ефективне використання git

### 9.7.1 Зміст комітів

Зробіть ваші коміти невеликими та лаконічними. Кожен коміт має виконувати одну логічну зміну в репозиторії.

### 9.7.2 Пишіть гарні повідомлення про коміти

Тримайте повідомлення комітів завширшки приблизно 72 стовпці (щоб у вікні терміналу розміру за замовчуванням вони не переносилися при відображенні `git log`).

Використовуйте перший рядок як короткий виклад наміру зміни (майже як тему електронного листа). Після нього додайте порожній рядок, а потім довше повідомлення з поясненням зміни. Приклад:

### 9.7.3 Перейдіть до відповідної гілки

Виправлення помилок слід розміщувати в найстарішій відповідній гілці. Нові функції слід розміщувати в гілці master. Якщо ви не впевнені, де знаходиться зміна, запитайте в irc або у списку розсилки.

### 9.7.4 Використовуйте кілька комітів для впорядкування змін

За необхідності, організуйте свої зміни у гілку (серію комітів), де кожен коміт є логічним кроком до вашої кінцевої мети. Наприклад, спочатку винесіть частину складного коду в нову функцію. Потім, у другому коміті, виправте базову помилку. Далі, у третьому коміті, додайте нову функцію, яка стала простішою завдяки рефакторингу і яка не працювала б без виправлення цієї помилки.

Це корисно для рецензентів, оскільки легше побачити, що крок «винести код у нову функцію» був правильним, коли немає інших редагувань; легше побачити, що помилка виправлена, коли зміна, яка її виправляє, відокремлена від нової функції; і так далі.

### 9.7.5 Дотримуйтесь стилю навколишнього коду

Намагайтеся дотримуватися стилю відступів, що переважає в навколишньому коді. Зокрема, зміни в пробілах ускладнюють іншим розробникам відстеження змін з часом. Якщо необхідно переформатувати код, робіть це окремим комітом, незалежним від будь-яких семантичних змін.

### 9.7.6 Позбудьтеся RTAPI_SUCCESS, використовуйте замість нього 0

Тест ”retval < 0” має бути знайомим; це такий самий тип тесту, який використовується в просторі користувача (повертає -1 у разі помилки) та в просторі ядра (повертає -ERRNO у разі помилки).

### 9.7.7 Спростіть складну історію, перш ніж ділитися нею з іншими розробника

За допомогою git можна записувати кожне редагування та помилковий старт як окремий коміт. Це дуже зручно для створення контрольних точок під час розробки, але часто ви не хочете ділитися цими помилковими стартами з іншими.

Git пропонує два основні способи очищення історії, обидва з яких можна зробити безкоштовно, перш ніж поділитися змінами:

`git commit --amend` дозволяє внести додаткові зміни до останнього коміту, а також, за бажанням, змінити повідомлення про коміт. Використовуйте цю команду, якщо ви відразу зрозуміли, що щось пропустили в коміті, або якщо ви зробили помилку в повідомленні про коміт.

`git rebase --interactive` upstream-branch дозволяє повернутися до кожного коміту, зробленого з моменту відгалуження вашої гілки функцій від гілки upstream, з можливістю редагування комітів, видалення комітів або об’єднання (squashing) комітів з іншими. Rebase також можна використовувати для розділення окремих комітів на кілька нових комітів.

### 9.7.8 Переконайтеся, що кожен коміт збирається

Якщо ваші зміни складаються з декількох патчів, можна використати `git rebase -i`, щоб перевпорядк ці патчі в послідовність комітів, яка чіткіше відображає етапи вашої роботи. Потенційним наслідком перевпорядкування патчів може бути неправильне визначення залежностей — наприклад, введення змінної, а оголошення цієї змінної з’являється лише в наступному патчі.

Хоча гілка HEAD буде компілюватися, не кожна коміт може компілюватися в такому випадку. Це порушує роботу `git bisect` — інструменту, який хтось інший може використовувати пізніше для

пошуку коміту, що спричинив появу помилки. Тому, крім того, що ви повинні переконатися, що ваша гілка компілюється, важливо також переконатися, що кожна окрема коміт також компілюється.

Існує автоматичний спосіб перевірки гілки на можливість побудови кожного коміту — див. https://dusti 2010/03/28/git-test-sequence.html та код за адресою https://github.com/dustin/bindir/blob/master/-git-test-sequence. Використовуйте наступним чином (у цьому випадку тестується кожна коміт від origin/master до HEAD, включаючи виконання регресійних тестів):

```
cd linuxcnc-dev
git-test-sequence origin/master..  '(cd src && make && ../scripts/runtests)'
```

Це або повідомить «Все гаразд», або «Зламано <commit>»

### 9.7.9 Перейменування файлів

Будь ласка, використовуйте можливість перейменувати файли дуже обережно. Як і виконання втягування в окремих файлах, перейменування ускладнює відстеження змін з часом. Як мінімум, ви повинні досягти консенсусу в irc або списку розсилки, що перейменування є поліпшенням.

### 9.7.10 Віддати перевагу "rebase"

Використовуйте `git pull --rebase` замість простої `git pull` для збереження гарної лінійної історії. Коли ви переосновуєте свою роботу, ви завжди зберігаєте її як ревізії, які попереду origin/master, тому можете робити такі речі, як `git-patch` їх поділитися з іншими без натиску до центрального сховища.

## 9.8 Переклади

Проект LinuxCNC використовує `gettext` для перекладу програмного забезпечення на багато мов. Ми раді будь-якій допомозі та внеску в цій сфері! Покращувати та розширювати переклади дуже просто: для цього не потрібно знати програмування, а також не потрібно встановлювати спеціальні програми для перекладу чи інше програмне забезпечення.

Найпростіший спосіб допомогти з перекладами – це скористатися Weblate, веб-сервісом з відкритим кодом. Наш проєкт перекладу знаходиться тут:

https://hosted.weblate.org/projects/linuxcnc/

Документація щодо використання Weblate знаходиться тут: https://docs.weblate.org/en/latest/-user/basic.html

## 9.9 Інші способи зробити внесок

Існує багато способів зробити внесок у LinuxCNC, які не розглядаються в цьому документі. Ці способи включають:

- Відповідь на запитання на форумі, у списках розсилки та в IRC
- Повідомлення про помилки в системі відстеження помилок, на форумі, у списках розсилки або в IRC
- Допомога в тестуванні експериментальних функцій

# Chapter 10

# 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 11

# Юридичний відділ

Переклади цього файлу, надані у дереві вихідних кодів, не мають юридичної сили.

## 11.1 Умови авторського права



## 11.2 Ліцензія GNU Free Documentation

**Ліцензія GNU Free Documentation, версія 1.1, березень 2000 р.**



**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.