Руководство разработчика
V2.10.0-pre0-3664-g8c68df8c0c, 09 Oct 2023

Contents

Chapter 1

Введение

Работа над этим справочником продолжается. Если вы можете помочь с написанием, редактированием или подготовкой графики, пожалуйста, свяжитесь с любым членом команды авторов или присоединитесь и отправьте электронное письмо по адресу emc-users@lists.sourceforge.net.

Chapter 2

HAL General Reference

2.1 HAL Entity Names

Все объекты HAL доступны и ими можно манипулировать по их именам, поэтому очень важно документировать имена контактов, сигналов, параметров и т. д. Имена в HAL имеют максимальную длину 41 символ (как определено HAL_NAME_LEN в hal.h). Многие имена будут представлены в общем виде с форматированным текстом <*mak*>, представляющим поля с различными значениями.

Когда контакты, сигналы или параметры описываются впервые, перед их именем в круглых скобках указывается их тип (*float*), а за ним следует краткое описание. Типичные определения выводов выглядят так:

(bit) parport.<portnum>.pin-<pinnum>-in
Контакт HAL, связанный с физическим входным контактом <*pinnum*> разъема db25.

(float) pid.<loopnum>.output
Выходной сигнал ПИД-регулятора

Иногда можно использовать сокращенную версию имени, например, второй контакт выше можно просто вызвать с помощью *.output*, когда это можно сделать, не вызывая путаницы.

2.2 HAL General Naming Conventions

Согласованные соглашения об именах значительно упростят использование HAL. Например, если бы каждый драйвер энкодерапредоставлял один и тот же набор контактов и называл их одинаково, то было бы легко перейти от одного типа драйвера энкодера к другому. К сожалению, как и многие проекты с открытым исходным кодом, HAL представляет собой комбинацию вещей, которые были разработаны, и вещей, которые просто развивались. В результате возникает множество несоответствий. В этом разделе делается попытка решить эту проблему путем определения некоторых соглашений, но, вероятно, пройдет некоторое время, прежде чем все модули будут преобразованы в соответствии с ними.

Halcmd и другие утилиты HAL низкого уровня рассматривают имена HAL как отдельные объекты без внутренней структуры. Однако большинство модулей имеют некоторую неявную структуру. Например, плата содержит несколько функциональных блоков, каждый блок может иметь несколько каналов, и каждый канал имеет один или несколько контактов. В результате получается структура, напоминающая дерево каталогов. Несмотря на то, что halcmd не распознает древовидную структуру, правильный выбор соглашений об именах позволит ему группировать связанные элементы вместе (поскольку он сортирует имена). Кроме того, инструменты более высокого уровня могут быть

разработаны для распознавания такой структуры, если имена предоставляют необходимую информац
Для этого все компоненты HAL должны следовать следующим правилам:

- Точки (“.”) разделяют уровни иерархии. Это аналог косой черты (“/”) в имени файла.
- Дефисы (“-”) разделяют слова или поля на одном уровне иерархии.
- Компоненты HAL не должны использовать символы подчеркивания или “MixedCase”. footnote: [Подчеркнутые символы были удалены, но все еще есть несколько случаев неправильного сочетания, например *pid.0.Pgain* вместо *pid.0.p-gain*.]
- Используйте в именах только строчные буквы и цифры.

2.3 Hardware Driver Naming Conventions

Note
Большинство драйверов не следуют этим соглашениям в версии 2.0. Эта глава на самом деле является руководством для будущих разработок.

2.3.1 Pins/Parameters names

Драйверы оборудования должны использовать пять полей (на трех уровнях) для создания имени контакта или параметра, а именно:

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

Отдельные поля:

<device-name>
Устройство, для которого предназначен драйвер. Чаще всего это интерфейсная плата какого-либо типа, но есть и другие возможности.

<device-num>
В компьютер можно установить более одной сервоплаты, параллельного порта или другого аппаратного устройства. Номер устройства идентифицирует конкретное устройство. Номера устройств начинаются с 0 и увеличиваются.

<io-type>
Большинство устройств предоставляют более одного типа ввода-вывода. Даже простой параллельный порт имеет как цифровые входы, так и цифровые выходы. Более сложные платы могут иметь цифровые входы и выходы, счетчики энкодеров, генераторы ШИМ или шаговых импульсов, аналого-цифровые преобразователи, цифро-аналоговые преобразователи или другие уникальные возможности. Тип ввода-вывода используется для идентификации типа ввода-вывода, с которым связан вывод или параметр. В идеале драйверы, реализующие один и тот же тип ввода-вывода, даже если для очень разных устройств, должны обеспечивать согласованный набор выводов и параметров и идентичное поведение. Например, все цифровые входы должны вести себя одинаково, если смотреть изнутри HAL, независимо от устройства.

<chan-num>
Практически каждое устройство ввода-вывода имеет несколько каналов, и номер канала идентифицирует один из них. Как и номера устройств, номера каналов начинаются с нуля и увеличиваются. footnote: [Одним исключением из правила ”номера каналов начинаются

с нуля” является параллельный порт. Его контакты HAL пронумерованы соответствующим номером контакта на разъеме DB-25. Это удобно для разводки, но несовместимо с другими драйверами. Есть некоторые споры о том, является ли это ошибкой или особенностью.] Если установлено более одного устройства, номера каналов на дополнительных устройствах начинаются с нуля. Если возможно иметь номер канала больше 9, то номера каналов должны состоять из двух цифр, с ведущим нулем в числах меньше 10, чтобы сохранить порядок сортировки. Некоторые модули имеют контакты и/или параметры, которые влияют более чем на один канал. Например, генератор ШИМ может иметь четыре канала с четырьмя независимыми входами ”скважность”, но одним параметром ”частота”, который управляет всеми четырьмя каналами (из-за аппаратных ограничений). Параметр частота должен использов ”0-3” в качестве номера канала.

<specific-name>
Отдельный канал ввода-вывода может иметь только один связанный с ним контакт HAL, но большинство из них имеют более одного. Например, цифровой вход имеет два контакта: один — состояние физического контакта, другой — то же самое, но в инвертированном виде. Это позволяет конфигуратору выбирать между входами с активным высоким и активным низким уровнем. Для большинства типов io существует стандартный набор контактов и параметров (называемый ”стандартным интерфейсом”), который должен реализовать драйвер. Стандартные интерфейсы описаны в главе Интерфейсы стандартных устройств.

Examples

motenc.0.encoder.2.position
Выход положения третьего канала энкодера на первой плате Motenc.

stg.0.din.03.in
Состояние четвертого цифрового входа на первой плате Servo-to-Go.

ppmc.0.pwm.00-03.frequency
Несущая частота, используемая для каналов ШИМ с 0 по 3 на первой плате ppmc Pico Systems.

2.3.2 Function Names

Драйверы оборудования обычно имеют только два типа функций HAL: одни считывают оборудование и обновляют контакты HAL, а другие записывают в оборудование, используя данные с контактов HAL. Их следует назвать следующим образом:

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

<device-name>
То же, что используется для контактов и параметров.

<device-num>
Конкретное устройство, к которому будет обращаться функция.

<io-type>
Опционально. Функция может иметь доступ ко всем вводам/выводам на плате или только к определенному типу. Например, могут существовать независимые функции для чтения счетчиков энкодера и чтения цифровых входов/выходов. Если такие независимые функции существуют, поле <io-type> идентифицирует тип ввода-вывода, к которому они обращаются. Если одна функция считывает все входы/выходы, предоставляемые платой, <io-type> не используется. footnote: [Примечание для программистов драйверов: НЕ реализуйте отдельные функции для разных типов ввода-вывода, если только они не являются прерываемыми и не могут работать в независимых потоках. Если прерывание чтения энкодера, чтение цифровых входов и последующее возобновление чтения энкодера вызовут проблемы, реализуйте одну функцию, которая сделает все.]

<chan-num-range>
Опционально. Используется только в том случае, если ввод-вывод <io-type> разбит на группы и доступен различным функциям.

read|write
Указывает, считывает ли функция с оборудования или записывает в него.

Examples

motenc.0.encoder.read
Считывает все энкодеры на первой плате motenc.

generic8255.0.din.09-15.read
Считывает второй 8-битный порт на первой универсальной плате цифрового ввода-вывода на базе 8255.

ppmc.0.write
Записывает все выходные данные (шаговые генераторы, ШИМ, ЦАП и цифровые данные) на первую плату ppmc Pico Systems.

Chapter 3

Примечания к коду

3.1 Целевая аудитория

Этот документ представляет собой сборник заметок о внутреннем устройстве LinuxCNC. Он в первую очередь представляет интерес для разработчиков, однако большая часть информации здесь может быть также интересна системным интеграторам и другим, кому просто интересно узнать, как работает LinuxCNC. Большая часть этой информации уже устарела и никогда не проверялась на точность.

3.2 Организация

В документации будет глава для каждого из основных компонентов LinuxCNC, а также глава(ы), описывающие, как они работают вместе. Этот документ в значительной степени находится в стадии разработки, и его макет может измениться в будущем.

3.3 Термины и определения

- *AXIS* - An axis is one of the nine degrees of freedom that define a tool position in three-dimensional Cartesian space. Those nine axes are referred to as X, Y, Z, A, B, C, U, V, and W. The linear orthogonal coordinates X, Y, and Z determine where the tip of the tool is positioned. The angular coordinates A, B, and C determine the tool orientation. A second set of linear orthogonal coordinates U, V, and W allows tool motion (typically for cutting actions) relative to the previously offset and rotated axes. Unfortunately ”axis” is also sometimes used to mean a degree of freedom of the machine itself, such as the saddle, table, or quill of a Bridgeport type milling machine. On a Bridgeport this causes no confusion, since movement of the table directly corresponds to movement along the X axis. However, the shoulder and elbow joints of a robot arm and the linear actuators of a hexapod do not correspond to movement along any Cartesian axis, and in general it is important to make the distinction between the Cartesian axes and the machine degrees of freedom. In this document, the latter will be called *joints*, not axes. The GUIs and some other parts of the code may not always follow this distinction, but the internals of the motion controller do.
- *JOINT* — Сочленение — это одна из подвижных частей машины. Сочленения отличаются от осей, хотя эти два термина иногда (неправильно) используются для обозначения одного и того же. В LinuxCNC сочленение — это физический объект, который можно перемещать, а не координата в пространстве. Например, пиноль, колено, седло и стол фрезера Bridgeport представляют собой сочленения. Плечо, локоть и запястье руки робота являются сочленениями,

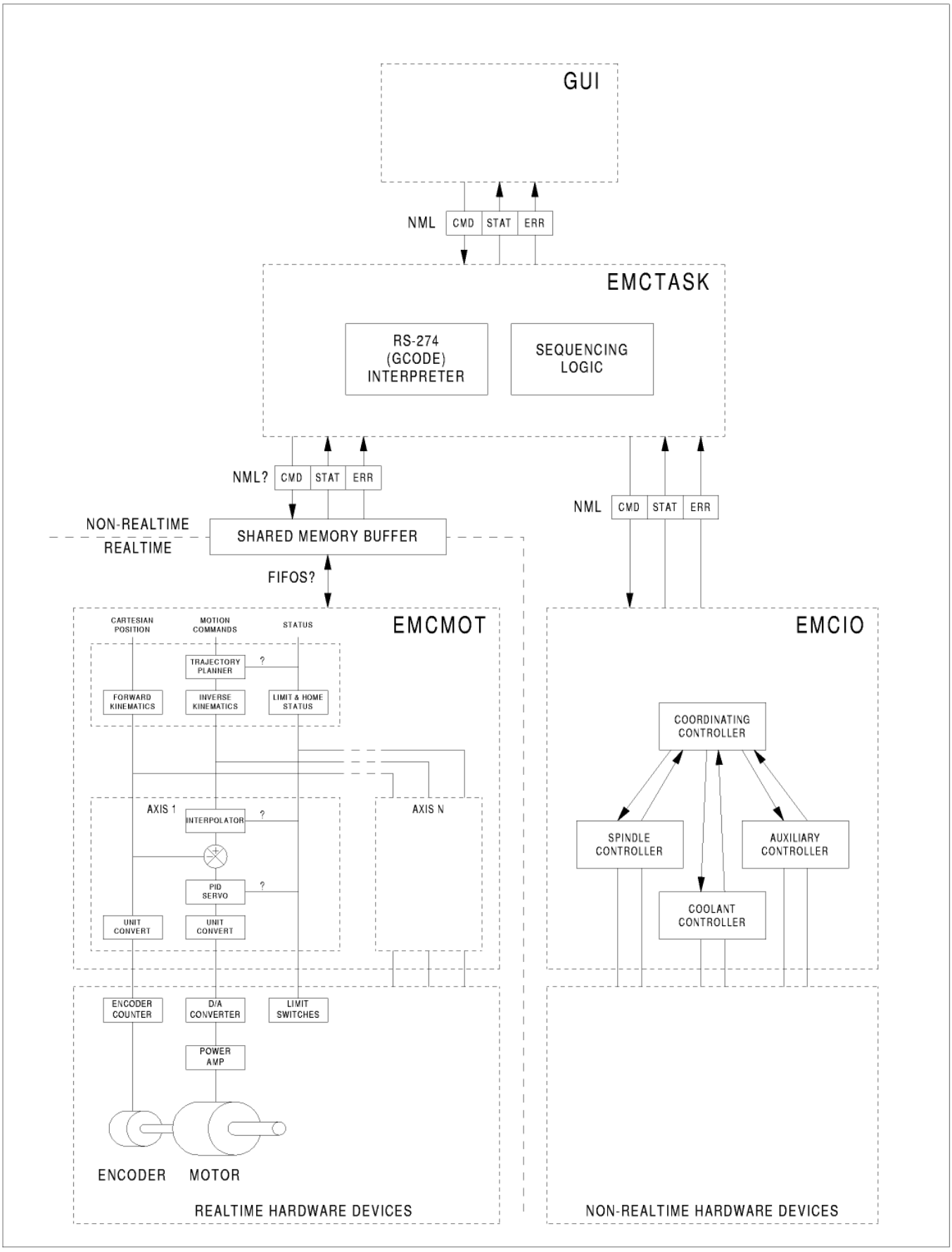
как и линейные приводы гексапода. С каждым сочленением связан двигатель или привод определенного типа. Сочленения не обязательно соответствуют осям X, Y и Z, хотя для машин с тривиальной кинематикой это может иметь место. Даже на этих машинах положение сочленения и положение оси — это принципиально разные вещи. В этом документе термины *сочленение* и *ось* используются осторожно, с учетом их различных значений. К сожалению, это не обязательно так везде. В частности, ГИП'ы станков с тривиальной кинематикой могут затушевывать или полностью скрывать различие между сочленениями и осями. Кроме того, в файле INI используется термин *ось* для данных, которые точнее было бы описать как данные сочленения, например масштабирование входных и выходных данных и т. д.

Note
Это различие было сделано в версии 2.8 LinuxCNC. В INI-файле появился новый раздел [JOINT_<num>]. Многие параметры, которые раньше относились к разделу [AXIS_<letter>], теперь находятся в новом разделе. Другие разделы, такие как [KINS], также принимают новые параметры, соответствующие этому. Был предоставлен скрипт обновления для преобразования старых файлов INI в новую конфигурацию осей/сочленений.

- *POSE* - A pose is a fully specified position in 3D Cartesian space. In the LinuxCNC motion controller, when we refer to a pose we mean an EmcPose structure, containing six linear coordinates (X, Y, Z, U, V, and W) and three angular ones (A, B, and C).
- *coord*, or coordinated mode, means that all articulations are synchronized and they move together as directed by the higher-level code. It is the normal mode when machining. In coordinated mode, commands are assumed to be given in the Cartesian reference frame, and if the machine is not Cartesian, the commands are translated by the kinematics to drive each joint into the joint space as needed.
- *free* means that commands are interpreted in joint space. It is used to manually move (jog) individual joints, although it does not prevent them from moving multiple joints at once (I think). Homing is also done in free mode; in fact, machines with non-trivial kinematics must be homed before they can go into coord or teleop mode.
- *teleop* is the mode you probably need if you are jogging with a hexapod. The jog commands implemented by the motion controller are joint jogs, which work in free mode. But if you want to move a hexapod or similar machine along a cartesian axis in particular, you must operate more than one joint. That's what *teleop* is for.

3.4 Architecture overview

There are four components contained in the LinuxCNC Architecture: a motion controller (EMCMOT), a discrete IO controller (EMCIO), a task executor which coordinates them (EMCTASK) and several text-mode and graphical User Interfaces. Each of them will be described in the current document, both from the design point of view and from the developers point of view (where to find needed data, how to easily extend/modify things, etc.).


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 LinuxCNC software architecture

At the coarsest level, LinuxCNC is a hierarchy of three controllers: the task level command handler and program interpreter, the motion controller, and the discrete I/O controller. The discrete I/O controller is implemented as a hierarchy of controllers, in this case for spindle, coolant, and auxiliary (e.g., estop) subsystems. The task controller coordinates the actions of the motion and discrete I/O controllers. Their actions are programmed in conventional numerical control ”G and M code” programs, which are interpreted by the task controller into NML messages and sent to the motion.

3.5 Motion Controller Introduction

The motion controller is a realtime component. It receives motion control commands from the non-realtime parts of LinuxCNC (i.e. the G-code interpreter/Task, GUIs, etc) and executes those commands within its realtime context. The communication from non-realtime context to realtime context happens via a message-passing IPC mechanism using shared memory, and via the Hardware Abstraction Layer (HAL).

The status of the motion controller is made available to the rest of LinuxCNC through the same message-passing shared memory IPC, and through HAL.

The motion controller interacts with the motor controllers and other realtime and non-realtime hardware using HAL.

Этот документ предполагает, что читатель имеет базовое представление о HAL, и используются такие термины, как контакты HAL, сигналы HAL и т. д., без их объяснения. Дополнительную информацию о HAL см. в HAL Руководство . Другая глава этого документа в конечном итоге будет посвящена внутреннему устройству самого HAL, но в этой главе мы используем только HAL API, как определено в src/hal/hal.h.

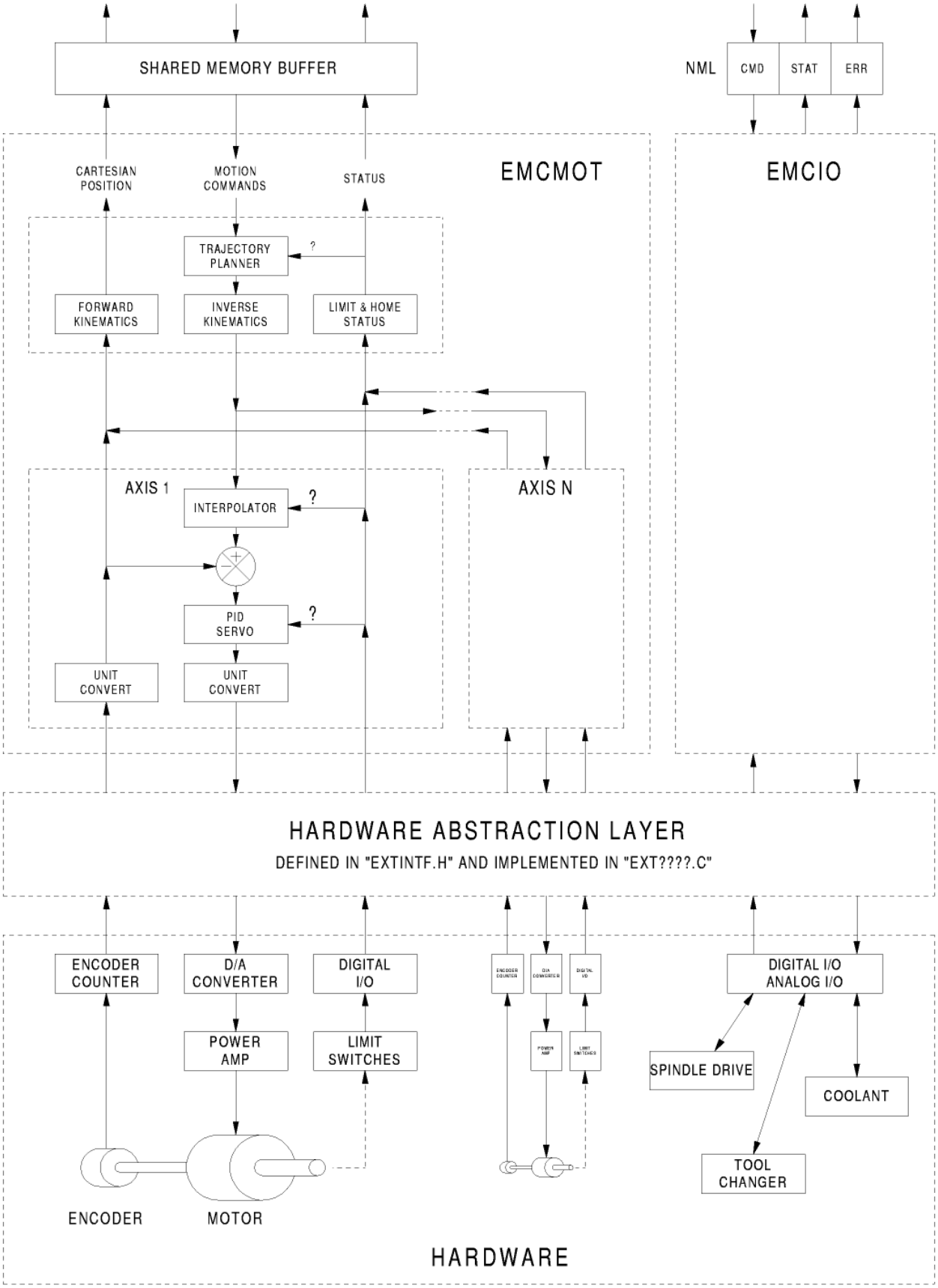
3.5.1 Motion Controller Modules

The realtime functions of the motion controller are implemented with realtime modules — userspace shared objects for Preempt-RT systems or kernel modules for some kernel-mode realtime implementations such as RTAI:

- *tpmod* - trajectory planning
- *homemod* - homing functions
- *motmod* - processes NML commands and controls hardware via HAL
- *kinematics module* - performs forward (joints-->coordinates) and inverse (coordinates->joints) kinematics calculations

LinuxCNC is started by a **linuxcnc** script which reads a configuration INI file and starts all needed processes. For realtime motion control, the script first loads the default tpmod and homemod modules and then loads the kinematics and motion modules according to settings in halfiles specified by the INI file.

Custom (user-built) homing or trajectory-planning modules can be used in place of the default modules via INI file settings or command line options. Custom modules must implement all functions used by the default modules. The halcompile utility can be used to create a custom module.


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 Block diagrams and Data Flow

The following figure is a block diagram of a joint controller. There is one joint controller per joint. The joint controllers work at a lower level than the kinematics, a level where all joints are completely independent. All the data for a joint is in a single joint structure. Some members of that structure are visible in the block diagram, such as coarse_pos, pos_cmd, and motor_pos_fb.

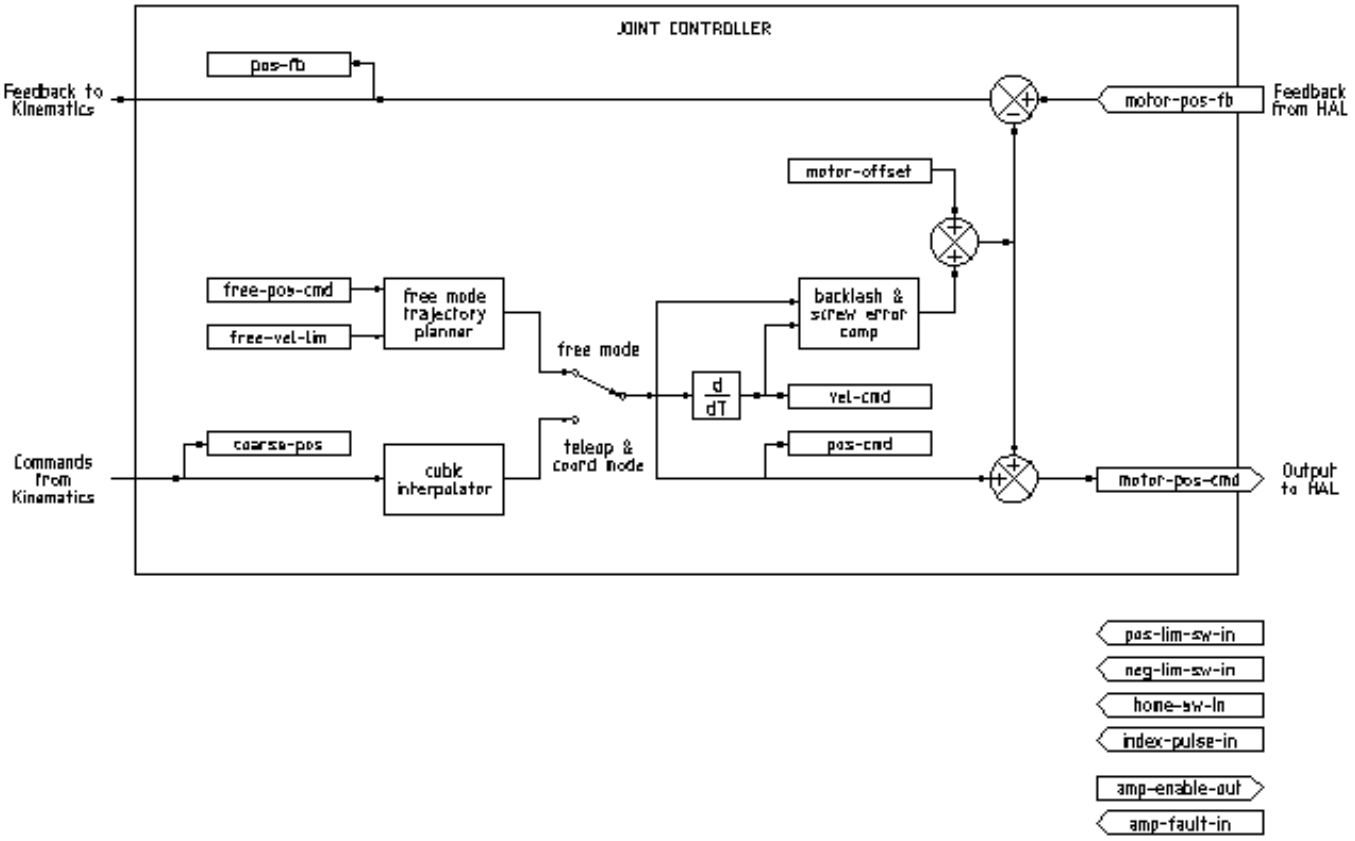


Figure 3.1: Joint Controller Block Diagram

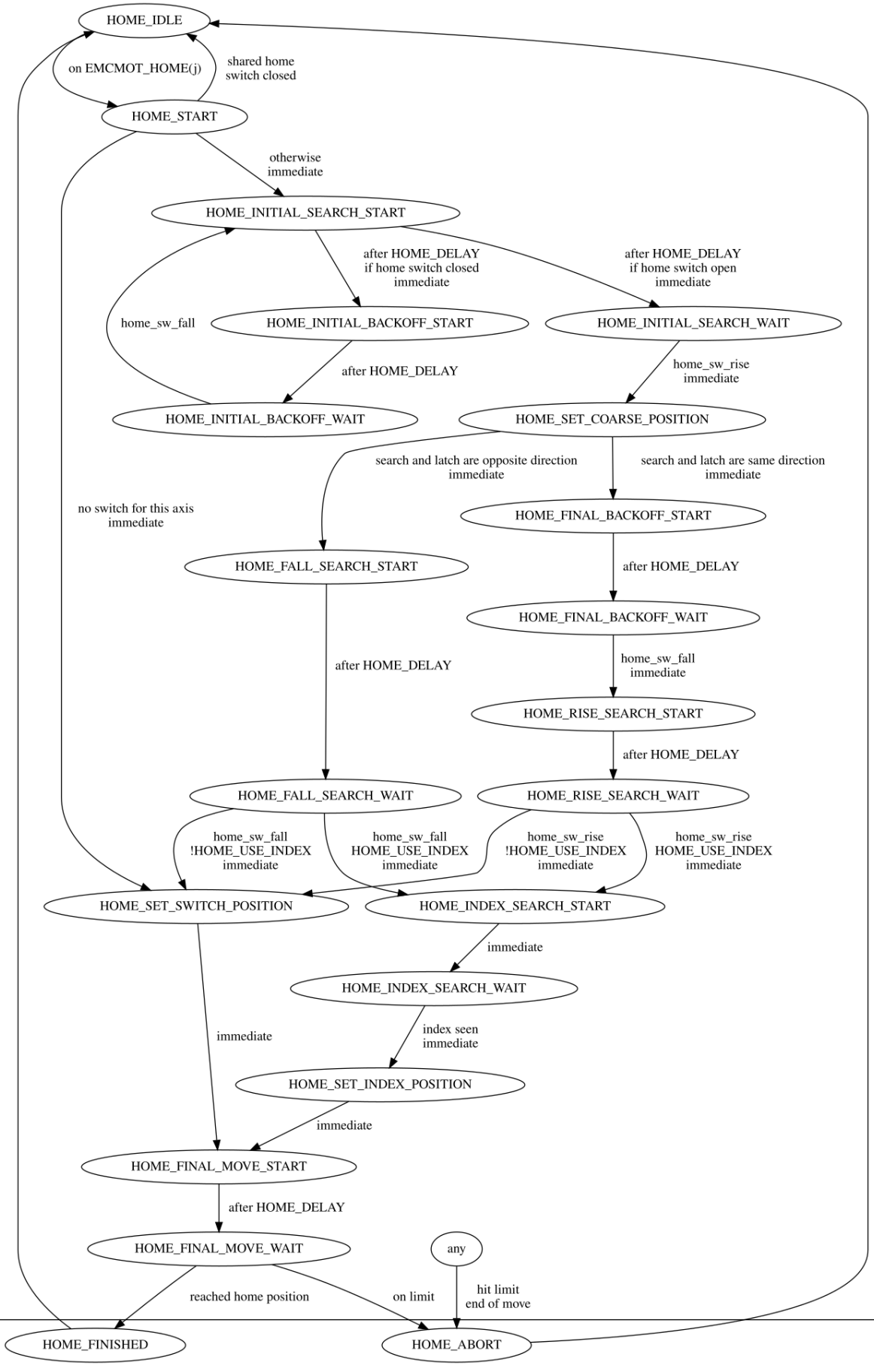
The above figure shows five of the seven sets of position information that form the main data flow through the motion controller. The seven forms of position data are as follows:

- *emcmotStatus->carte_pos_cmd* - This is the desired position, in Cartesian coordinates. It is updated at the traj rate, not the servo rate. In coord mode, it is determined by the traj planner. In teleop mode, it is determined by the traj planner? In free mode, it is either copied from actualPos, or generated by applying forward kins to (2) or (3).
- *emcmotStatus->joints[n].coarse_pos* - This is the desired position, in joint coordinates, but before interpolation. It is updated at the traj rate, not the servo rate. In coord mode, it is generated by applying inverse kins to (1) In teleop mode, it is generated by applying inverse kins to (1) In free mode, it is copied from (3), I think.

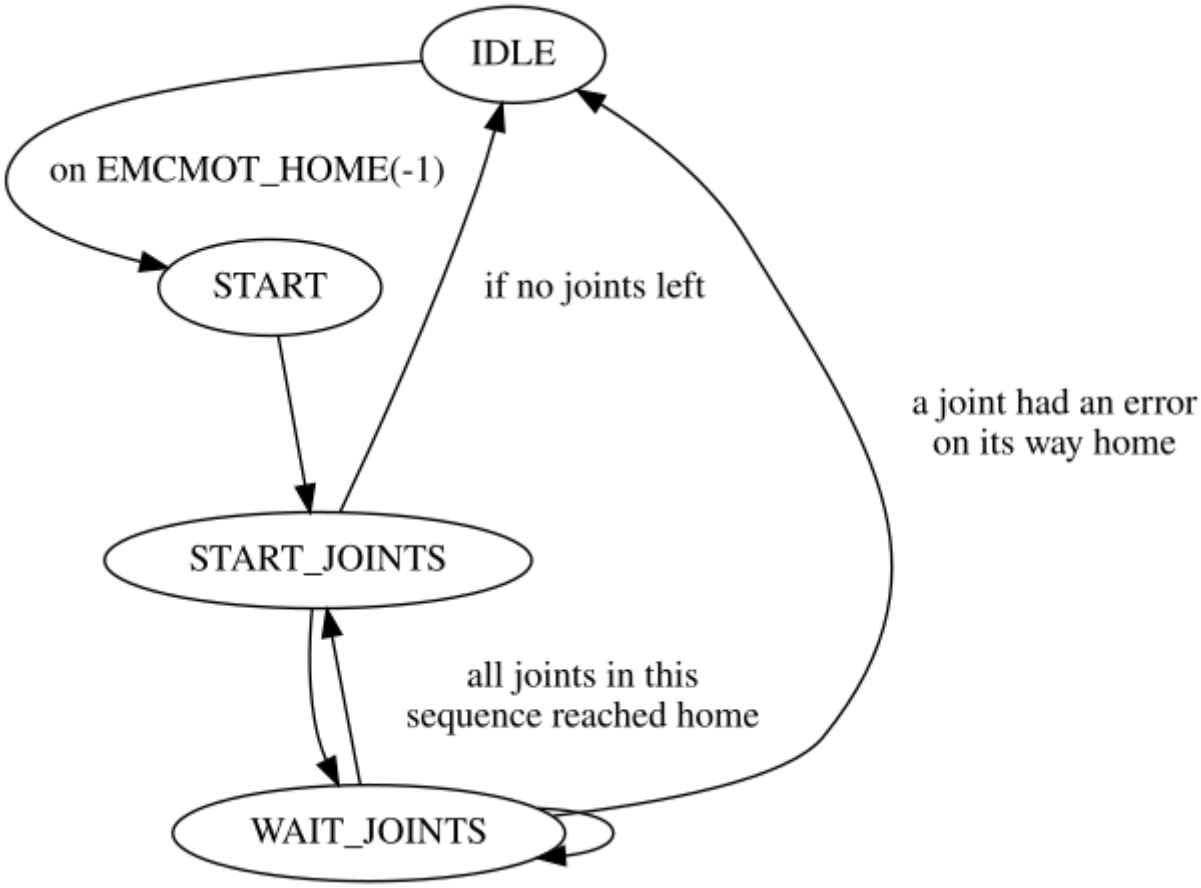
- ’emcmotStatus->joints[n].pos_cmd - This is the desired position, in joint coords, after interpolation. A new set of these coords is generated every servo period. In coord mode, it is generated from (2) by the interpolator. In teleop mode, it is generated from (2) by the interpolator. In free mode, it is generated by the free mode traj planner.
- *emcmotStatus->joints[n].motor_pos_cmd* - This is the desired position, in motor coords. Motor coords are generated by adding backlash compensation, lead screw error compensation, and offset (for homing) to (3). It is generated the same way regardless of the mode, and is the output to the PID loop or other position loop.
- *emcmotStatus->joints[n].motor_pos_fb* - This is the actual position, in motor coords. It is the input from encoders or other feedback device (or from virtual encoders on open loop machines). It is ”generated” by reading the feedback device.
- *emcmotStatus->joints[n].pos_fb* - This is the actual position, in joint coordinates. It is generated by subtracting offset, lead screw error compensation, and backlash compensation from (5). It is generated the same way regardless of the operating mode.
- *emcmotStatus->carte_pos_fb* - This is the actual position, in Cartesian coordinates. It is updated at the traj rate, not the servo rate. Ideally, actualPos would always be calculated by applying forward kinematics to (6). However, forward kinematics may not be available, or they may be unusable because one or more axes aren’t homed. In that case, the options are: A) fake it by copying (1), or B) admit that we don’t really know the Cartesian coordinates, and simply don’t update actualPos. Whatever approach is used, I can see no reason not to do it the same way regardless of the operating mode. I would propose the following: If there are forward kins, use them, unless they don’t work because of unhomed axes or other problems, in which case do (B). If no forward kins, do (A), since otherwise actualPos would *never* get updated.

3.7 Homing

3.7.1 Homing state diagram



3.7.2 Another homing diagram



3.8 Commands

The commands are implemented by a large switch statement in the function emcmotCommandHandler(), which is called at the servo rate. More on that function later.

There are approximately 44 commands - this list is still under construction.

Note
The cmd_code_t enumeration, in motion.h, contains 73 commands, but the switch statement in command.c contemplates only 70 commands (as of 6/5/2020). ENABLE_WATCHDOG / DISABLE_WATCHDOG commands are in motion-logger.c. Maybe they are obsolete. The SET_TELEOP_VECTOR command only appears in motion-logger.c, with no effect other than its own log.

3.8.1 ABORT

The ABORT command simply stops all motion. It can be issued at any time, and will always be accepted. It does not disable the motion controller or change any state information, it simply cancels any motion that is currently in progress.[1]

[1] It seems that the higher level code (TASK and above) also use ABORT to clear faults. Whenever there is a persistent fault (such as being outside the hardware limit switches), the higher level code sends a constant stream of ABORTs to the motion controller trying to make the fault go away. Thousands of them.... That means that the motion controller should avoid persistent faults. This needs to be looked into.

3.8.1.1 Requirements

None. The command is always accepted and acted on immediately.

3.8.1.2 Results

In free mode, the free mode trajectory planners are disabled. That results in each joint stopping as fast as its accel (decel) limit allows. The stop is not coordinated. In teleop mode, the commanded Cartesian velocity is set to zero. I don't know exactly what kind of stop results (coordinated, uncoordinated, etc), but will figure it out eventually. In coord mode, the coord mode trajectory planner is told to abort the current move. Again, I don't know the exact result of this, but will document it when I figure it out.

3.8.2 FREE

The FREE command puts the motion controller in free mode. Free mode means that each joint is independent of all the other joints. Cartesian coordinates, poses, and kinematics are ignored when in free mode. In essence, each joint has its own simple trajectory planner, and each joint completely ignores the other joints. Some commands (like Joint JOG and HOME) only work in free mode. Other commands, including anything that deals with Cartesian coordinates, do not work at all in free mode.

3.8.2.1 Requirements

The command handler applies no requirements to the FREE command, it will always be accepted. However, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored. This behavior is controlled by code that is now located in the function *set_operating_mode()* in control.c, that code needs to be cleaned up. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

3.8.2.2 Results

If the machine is already in free mode, nothing. Otherwise, the machine is placed in free mode. Each joint's free mode trajectory planner is initialized to the current location of the joint, but the planners are not enabled and the joints are stationary.

3.8.3 TELEOP

The TELEOP command places the machine in teleoperating mode. In teleop mode, movement of the machine is based on Cartesian coordinates using kinematics, rather than on individual joints as in free mode. However the trajectory planner per se is not used, instead movement is controlled by a velocity vector. Movement in teleop mode is much like jogging, except that it is done in Cartesian space instead of joint space. On a machine with trivial kinematics, there is little difference between teleop mode and free mode, and GUIs for those machines might never even issue this command. However for non-trivial machines like robots and hexapods, teleop mode is used for most user commanded jog type movements.

3.8.3.1 Requirements

The command handler will reject the TELEOP command with an error message if the kinematics cannot be activated because the one or more joints have not been homed. In addition, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored (with no error message). This behavior is controlled by code that is now located in the function *set_operating_mode()* in control.c. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

3.8.3.2 Results

If the machine is already in teleop mode, nothing. Otherwise the machine is placed in teleop mode. The kinematics code is activated, interpolators are drained and flushed, and the Cartesian velocity commands are set to zero.

3.8.4 COORD

The COORD command places the machine in coordinated mode. In coord mode, movement of the machine is based on Cartesian coordinates using kinematics, rather than on individual joints as in free mode. In addition, the main trajectory planner is used to generate motion, based on queued LINE, CIRCLE, and/or PROBE commands. Coord mode is the mode that is used when executing a G-code program.

3.8.4.1 Requirements

The command handler will reject the COORD command with an error message if the kinematics cannot be activated because the one or more joints have not been homed. In addition, if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), then the command will be ignored (with no error message). This behavior is controlled by code that is now located in the function *set_operating_mode()* in control.c. I believe the command should not be silently ignored, instead the command handler should determine whether it can be executed and return an error if it cannot.

3.8.4.2 Results

If the machine is already in coord mode, nothing. Otherwise, the machine is placed in coord mode. The kinematics code is activated, interpolators are drained and flushed, and the trajectory planner queues are empty. The trajectory planner is active and awaiting a LINE, CIRCLE, or PROBE command.

3.8.5 ENABLE

The ENABLE command enables the motion controller.

3.8.5.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.5.2 Results

If the controller is already enabled, nothing. If not, the controller is enabled. Queues and interpolators are flushed. Any movement or homing operations are terminated. The amp-enable outputs associated with active joints are turned on. If forward kinematics are not available, the machine is switched to free mode.

3.8.6 DISABLE

The DISABLE command disables the motion controller.

3.8.6.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.6.2 Results

If the controller is already disabled, nothing. If not, the controller is disabled. Queues and interpolators are flushed. Any movement or homing operations are terminated. The amp-enable outputs associated with active joints are turned off. If forward kinematics are not available, the machine is switched to free mode.

3.8.7 ENABLE_AMPLIFIER

The ENABLE_AMPLIFIER command turns on the amp enable output for a single output amplifier, without changing anything else. Can be used to enable a spindle speed controller.

3.8.7.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.7.2 Results

Currently, nothing. (A call to the old extAmpEnable function is currently commented out.) Eventually it will set the amp enable HAL pin true.

3.8.8 DISABLE_AMPLIFIER

The DISABLE_AMPLIFIER command turns off the amp enable output for a single amplifier, without changing anything else. Again, useful for spindle speed controllers.

3.8.8.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.8.2 Results

Currently, nothing. (A call to the old extAmpEnable function is currently commented out.) Eventually it will set the amp enable HAL pin false.

3.8.9 ACTIVATE_JOINT

The ACTIVATE_JOINT command turns on all the calculations associated with a single joint, but does not change the joint’s amp enable output pin.

3.8.9.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.9.2 Results

Calculations for the specified joint are enabled. The amp enable pin is not changed, however, any subsequent ENABLE or DISABLE commands will modify the joint's amp enable pin.

3.8.10 DEACTIVATE_JOINT

The DEACTIVATE_JOINT command turns off all the calculations associated with a single joint, but does not change the joint's amp enable output pin.

3.8.10.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.10.2 Results

Calculations for the specified joint are enabled. The amp enable pin is not changed, and subsequent ENABLE or DISABLE commands will not modify the joint's amp enable pin.

3.8.11 ENABLE_WATCHDOG

The ENABLE_WATCHDOG command enables a hardware based watchdog (if present).

3.8.11.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.11.2 Results

Currently nothing. The old watchdog was a strange thing that used a specific sound card. A new watchdog interface may be designed in the future.

3.8.12 DISABLE_WATCHDOG

Команда DISABLE_WATCHDOG отключает аппаратный сторожевой таймер (если он есть).

3.8.12.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.12.2 Results

Currently nothing. The old watchdog was a strange thing that used a specific sound card. A new watchdog interface may be designed in the future.

3.8.13 PAUSE

Команда PAUSE останавливает планировщик траектории. Она не действует в свободном режиме или режиме телеоперации. На данный момент я не знаю, останавливает ли он все движения немедленно или завершает текущее движение, а затем делает паузу, прежде чем вытащить другое движение из очереди.

3.8.13.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.13.2 Results

Планировщик траектории останавливается.

3.8.14 RESUME

Команда RESUME перезапускает планировщик траектории, если он был приостановлен. Она не действует в свободном или телеоперативном режиме, а также если планировщик не поставлен на паузу.

3.8.14.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.14.2 Results

Планировщик траектории возобновляет работу.

3.8.15 STEP

Команда STEP перезапускает планировщик траектории, если он был приостановлен, и указывает планировщику снова остановиться, когда он достигает определенной точки. Он не действует в свободном режиме или режиме телеоперации. На данный момент я точно не знаю, как это работает. Я добавлю сюда дополнительную документацию, когда углублюсь в планировщик траектории.

3.8.15.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.15.2 Results

Планировщик траектории возобновляет работу, а затем приостанавливается, когда достигает определенной точки.

3.8.16 SCALE

Команда SCALE масштабирует все ограничения скорости и команды на указанную величину. Она используется для реализации переопределения скорости подачи и других подобных функций. Масштабирование работает в свободном, телеоперативном и координатном режимах и влияет на все, включая скорость позиционирования в исходную позицию и т. д. Однако ограничения скорости отдельных сочленений не затрагиваются.

3.8.16.1 Requirements

None. The command can be issued at any time, and will always be accepted.

3.8.16.2 Results

Все команды скорости масштабируются указанной константой.

3.8.17 OVERRIDE_LIMITS

Команда OVERRIDE_LIMITS предотвращает срабатывание пределов до окончания следующей команды JOG. Обычно она используется для того, чтобы станок мог быть отведен с концевого выключателя после срабатывания. (На самом деле эту команду можно использовать для переопределе ограничений или для отмены предыдущего переопределения.)

3.8.17.1 Requirements

Никаких. Команда может быть подана в любое время и всегда будет принята. (Я думаю, что это должно работать только в свободном режиме.)

3.8.17.2 Results

Ограничения для всех сочленений переопределяются до окончания следующей команды JOG. (В настоящее время это не работает… как только получена команда OVERRIDE_LIMITS, пределы игнорируются до тех пор, пока другая команда OVERRIDE_LIMITS не активирует их снова.)

3.8.18 HOME

Команда HOME инициирует последовательность возврата в исходное положения указанного сочленения. Фактическая последовательность возврата определяется рядом параметров конфигураци и может варьироваться от простой установки текущего положения на ноль до многоступенчатого поиска концевого выключателя исходной позиции и индексного импульса с последующим перемещени в произвольное исходное положение. Для получения дополнительной информации о последовательно возврата см. раздел возврат в исходную позицию в Руководстве системного разработчика.

3.8.18.1 Requirements

Команда будет молча игнорироваться, если станок не находится в свободном режиме.

3.8.18.2 Results

Любая медленная подача или другое движение сочленения прерывается, и начинается последователь возврата в исходное положение.

3.8.19 JOG_CONT

The JOG_CONT command initiates a continuous jog on a single joint. A continuous jog is generated by setting the free mode trajectory planner's target position to a point beyond the end of the joint's range of travel. This ensures that the planner will move constantly until it is stopped by either the joint limits or an ABORT command. Normally, a GUI sends a JOG_CONT command when the user presses a jog button, and ABORT when the button is released.

3.8.19.1 Requirements

The command handler will reject the JOG_CONT command with an error message if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.

3.8.19.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, with a target position beyond the end of joint travel, and a velocity limit of emcmotCommand->vel. This starts the joint moving, and the move will continue until stopped by an ABORT command or by hitting a limit. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit when it stops.

3.8.20 JOG_INCR

The JOG_INCR command initiates an incremental jog on a single joint. Incremental jogs are cumulative, in other words, issuing two JOG_INCR commands that each ask for 0.100 inches of movement will result in 0.200 inches of travel, even if the second command is issued before the first one finishes. Normally incremental jogs stop when they have traveled the desired distance, however they also stop when they hit a limit, or on an ABORT command.

3.8.20.1 Requirements

The command handler will silently reject the JOG_INCR command if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.

3.8.20.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, the target position is incremented/decremented by emcmotCommand->offset, and the velocity limit is set to emcmotCommand->vel. The free mode trajectory planner will generate a smooth trapezoidal move from the present position to the target position. The planner can correctly handle changes in the target position that happen while the move is in progress, so multiple JOG_INCR commands can be issued in quick succession. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit to stop at the target position.

3.8.21 JOG_ABS

The JOG_ABS command initiates an absolute jog on a single joint. An absolute jog is a simple move to a specific location, in joint coordinates. Normally absolute jogs stop when they reach the desired location, however they also stop when they hit a limit, or on an ABORT command.

3.8.21.1 Requirements

The command handler will silently reject the JOG_ABS command if machine is not in free mode, or if any joint is in motion (GET_MOTION_INPOS_FLAG() == FALSE), or if motion is not enabled. It will also silently ignore the command if the joint is already at or beyond its limit and the commanded jog would make it worse.

3.8.21.2 Results

The free mode trajectory planner for the joint identified by emcmotCommand->axis is activated, the target position is set to emcmotCommand->offset, and the velocity limit is set to emcmotCommand->vel. The free mode trajectory planner will generate a smooth trapezoidal move from the present position to the target position. The planner can correctly handle changes in the target position that happen while the move is in progress. If multiple JOG_ABS commands are issued in quick succession, each new command changes the target position and the machine goes to the final commanded position. The free mode planner accelerates at the joint accel limit at the beginning of the move, and will decelerate at the joint accel limit to stop at the target position.

3.8.22 SET_LINE

The SET_LINE command adds a straight line to the trajectory planner queue.

(More later)

3.8.23 SET_CIRCLE

Команда SET_CIRCLE добавляет круговое движение в очередь планировщика траектории.

(More later)

3.8.24 SET_TELEOP_VECTOR

Команда SET_TELEOP_VECTOR указывает контроллеру движения двигаться по определенному вектору в декартовом пространстве.

(More later)

3.8.25 PROBE

Команда PROBE указывает контроллеру движения двигаться в направлении определенной точки в декартовом пространстве, останавливая и записывая его положение, если срабатывает вход датчика.

(More later)

3.8.26 CLEAR_PROBE_FLAG

Команда CLEAR_PROBE_FLAG используется для сброса входа датчика при подготовке к команде PROBE. (Вопрос: почему команда PROBE не должна автоматически сбрасывать ввод?)

(More later)

3.8.27 SET_xix

Существует примерно 15 команд SET_xxx, где xxx — это имя некоторого параметра конфигурации. Ожидается, что будет еще несколько команд SET по мере добавления дополнительных параметров. Я хотел бы найти более аккуратный способ установки и чтения параметров конфигурации. Существую методы требуют добавления множества строк кода в несколько файлов при каждом добавлении параметра. Большая часть этого кода идентична или почти идентична для каждого параметра.

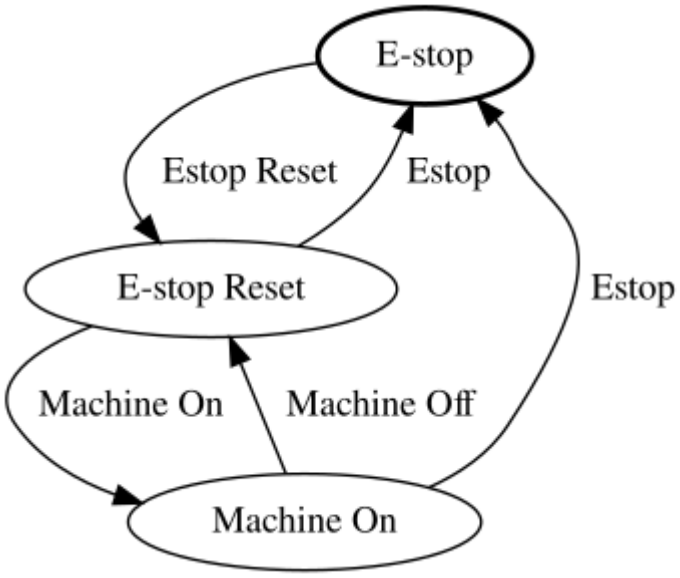
3.9 Компенсация люфта и погрешности винтов

```
+ FIXME b''Kb''b''ob''b''мb''b''пb''b''eb''b''нb''b''cb''b''ab''b''цb''b''иb''b''яb'' b' ↩
    'лb''b''юb''b''фb''b''тb''b''ab'' b''иb'' b''пb''b''ob''b''гb''b''pb''b''eb''b''шb''b' ↩
    'нb''b''ob''b''cb''b''тb''b''иb'' b''вb''b''иb''b''нb''b''тb''b''ob''b''вb''
```

3.10 Контроллер задач (EMCTASK)

3.10.1 Состояние

Задача имеет три возможных внутренних состояния: **Аварийная остановка**, **Сброс аварийной остановки** и **Станок включен**.



3.11 Контролер ввода/вывода (EMCIO)

The I/O Controller is part of TASK. It interacts with external I/O using HAL pins.

В настоящее время ESTOP/Enable, СОЖ и смена инструмента обрабатываются iocontrol. Это относительно низкоскоростные события, высокоскоростной скоординированный ввод-вывод обрабаты в движении.

emctaskmain.cc sends I/O commands via taskclass.cc.

процесс основного цикла iocontrol:

- проверяет, чтобы увидеть, что входные данные HAL изменились
- проверяет, указывает ли read_tool_inputs(), что смена инструмента завершена, и устанавливает emcioStatus.status

3.12 Интерфейсы пользователя

```
b''Иb''b''нb''b''тb''b''eb''b''pb''b''фb''b''eb''b''йb''b''cb''b''ыb'' b''пb''b''ob''b' ↩
    'лb''b''ьb''b''зb''b''ob''b''вb''b''ab''b''тb''b''eb''b''лb''b''яb''
```

3.13 libnml Introduction

libnml is derived from the NIST rcslib without all the multi-platform support. Many of the wrappers around platform specific code has been removed along with much of the code that is not required by LinuxCNC. It is hoped that sufficient compatibility remains with rcslib so that applications can be implemented on non-Linux platforms and still be able to communicate with LinuxCNC.

This chapter is not intended to be a definitive guide to using libnml (or rcslib), instead, it will eventually provide an overview of each C++ class and their member functions. Initially, most of these notes will be random comments added as the code scrutinized and modified.

3.14 LinkedList

Base class to maintain a linked list. This is one of the core building blocks used in passing NML messages and assorted internal data structures.

3.15 LinkedListNode

Base class for producing a linked list - Purpose, to hold pointers to the previous and next nodes, pointer to the data, and the size of the data.

No memory for data storage is allocated.

3.16 SharedMemory

Provides a block of shared memory along with a semaphore (inherited from the Semaphore class). Creation and destruction of the semaphore is handled by the SharedMemory constructor and destructor.

3.17 ShmBuffer

Class for passing NML messages between local processes using a shared memory buffer. Much of internal workings are inherited from the CMS class.

3.18 Timer

The Timer class provides a periodic timer limited only by the resolution of the system clock. If, for example, a process needs to be run every 5 seconds regardless of the time taken to run the process, the following code snippet demonstrates how :

```
main()
{
    timer = new Timer(5.0);    /* Initialize a timer with a 5 second loop */
    while(0) {
        /* Do some process */
        timer.wait();    /* Wait till the next 5 second interval */
    }
    delete timer;
}
```

3.19 Semaphore

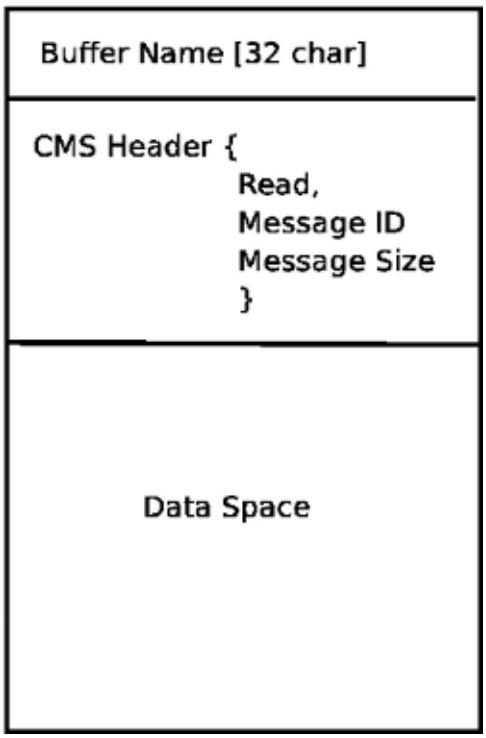
Класс Semaphore предоставляет метод взаимных исключений для доступа к общему ресурсу. Функция для получения семафора может либо блокироваться до тех пор, пока отсутствует доступ, возвращаться после тайм-аута, либо немедленно возвращаться с или без получения семафора. Конструктор создаст семафор или присоединится к существующему, если идентификатор уже используется.

Semaphore::destroy() должен вызываться только последним процессом.

3.20 CMS

В основе libnml лежит класс CMS, он содержит большинство функций, используемых libnml и по большому счету NML. Многие из внутренних функций перегружены, чтобы обеспечить определенные аппаратно-зависимые методы передачи данных. В конечном счете, все вращается вокруг центрального блока памяти (называемого «буфером сообщений» или просто «буфером»). Этот буфер может существовать как блок общей памяти, к которому обращаются другие процессы CMS/NML, или как локальный и частный буфер для данных, передаваемых по сети или последовательн интерфейсам.

Буфер выделяется динамически во время выполнения, чтобы обеспечить большую гибкость подсистем CMS/NML. Размер буфера должен быть достаточно большим, чтобы вместить самое большое сообщение, небольшой объем для внутреннего использования и позволять кодировать сообщение, если выбран этот параметр (закодированные данные будут рассмотрены позже). На следующем рисунке показан внутренний вид буферного пространства.



CMS буфер Базовый класс CMS в первую очередь отвечает за создание каналов связи и взаимодействи
с операционной системой.

3.21 Формат файла конфигурации

Конфигурация NML состоит из двух типов форматов строк. Один для буферов, а второй для процессов, которые подключаются к буферам.

3.21.1 Буферная линия

Исходный формат строки буфера NIST:

- *B name type host size neut RPC# buffer# max_procs key [type specific configs]*
- *B* - идентифицирует эту строку как конфигурацию буфера.
- *name* - это идентификатор буфера.
- *type* — описывает тип буфера — SHMEM, LOCMEM, FILEMEM, PHANTOM или GLOBMEM.
- «хост» — это либо IP-адрес, либо имя хоста для сервера NML
- *size* - размер буфера
- *neut* - логическое значение, указывающее, закодированы ли данные в буфере в машинно-независимом формате или они в необработанном виде.
- «RPC#» — устарело — заполнитель сохранен только для обратной совместимости.
- *buffer#* — уникальный идентификационный номер, используемый, если сервер управляет несколько буферами.
- *max_procs* - максимальное количество процессов, которым разрешено подключаться к этому буферу.
- *key* - числовой идентификатор для буфера общей памяти

3.21.2 Типизированные конфигурации

Тип буфера подразумевает дополнительные параметры конфигурации, в то время как операционная система хоста исключает определенные комбинации. При попытке преобразовать опубликованную документацию в согласованный формат будет рассмотрен только тип буфера **SHMEM**.

- *mutex=os_sem* - default mode for providing semaphore locking of the buffer memory.
- *mutex=none* - Not used
- *mutex=no_interrupts* - not applicable on a Linux system
- *mutex=no_switching* - not applicable on a Linux system
- *mutex=mao split* - Splits the buffer in to half (or more) and allows one process to access part of the buffer whilst a second process is writing to another part.
- *TCP=(port number)* - Specifies which network port to use.
- *UDP=(port number)* - ditto
- *STCP=(port number)* - ditto
- *serialPortDevName=(serial port)* - Undocumented.
- *passwd=file_name.pwd* - Adds a layer of security to the buffer by requiring each process to provide a password.
- *bsem* - NIST documentation implies a key for a blocking semaphore, and if bsem=-1, blocking reads are prevented.
- *queue* - Enables queued message passing.
- *ascii* - Encode messages in a plain text format
- *disp* - Encode messages in a format suitable for display (???)
- *xdr* - Encode messages in External Data Representation. (see rpc/xdr.h for details).
- *diag* - Enables diagnostics stored in the buffer (timings and byte counts ?)

3.21.3 Process line

The original NIST format of the process line is:

P name buffer type host ops server timeout master c_num [type specific configs]

- *P* - identifies this line as a Process configuration.
- *name* - is the identifier of the process.
- *buffer* - is one of the buffers defined elsewhere in the config file.
- *type* - определяет, является ли этот процесс локальным или удаленным по отношению к буферу.
- *host* - указывает, где в сети запущен этот процесс.
- *ops* - дает процессу доступ только для чтения, только для записи или для чтения/записи к буферу.
- *server* - указывает, будет ли этот процесс запускать сервер для этого буфера.
- *timeout* - устанавливает характеристики таймаута для доступа к буферу.
- *master* - указывает, отвечает ли этот процесс за создание и уничтожение буфера.
- *c_num* - целое число от нуля до (max_procs -1)

3.21.4 Комментарии к конфигурации

Некоторые из комбинаций конфигурации недействительны, в то время как другие подразумевают определенные ограничения. В системе Linux GLOBMEM устарел, в то время как PHANTOM действительно полезен только на этапе тестирования приложения, как и для FILEMEM. LOCMEM малопригоден для многопроцессорного приложения и предлагает лишь ограниченные преимущества в производительности по сравнению с SHMEM. Это оставляет SHMEM единственным типом буфера для использования с LinuxCNC.

Параметр neut используется только в многопроцессорной системе, где разные (и несовместимые) архитектуры совместно используют блок памяти. Вероятность увидеть систему такого типа за пределами музея или исследовательского учреждения невелика и имеет отношение только к буферам GLOBMEM.

Номер RPC задокументирован как устаревший и сохраняется только по соображениям совместимости.

С уникальным именем буфера наличие числового идентификатора кажется бессмысленным. Необходимо просмотреть код, чтобы определить логику. Точно так же ключевое поле на первый взгляд кажется избыточным и может быть получено из имени буфера.

Цель ограничения количества процессов, которым разрешено подключаться к любому одному буферу, неясна из существующей документации и оригинального исходного кода. Разрешить неопределенным множественным процессам подключаться к буферу реализовать не сложнее.

Типы мьютексов сводятся к одному из двух: по умолчанию «os_sem» или «mao split». Большинство сообщений NML относительно короткие и могут быть скопированы в буфер или из буфера с минимальными задержками, поэтому раздельное чтение не обязательно.

Кодирование данных имеет значение только при передаче удаленному процессу. Использование TCP или UDP подразумевает кодирование XDR. В то время как кодирование ASCII может иметь некоторое применение в диагностике или для передачи данных во встроенную систему, которая не реализует NML.

Протоколы UDP имеют меньше проверок данных и позволяют отбрасывать процент пакетов. TCP более надежен, но немного медленнее.

Если LinuxCNC должен быть подключен к сети, можно надеяться, что она локальная и находится за брандмауэром. Единственная причина разрешить доступ к LinuxCNC через Интернет — удаленная диагностика. Этого можно добиться гораздо более безопасно, используя другие средства, например, веб-интерфейс.

Точное поведение, когда время ожидания установлено равным нулю или отрицательному значению, неясно из документов NIST. Упоминаются только INF и положительные значения. Однако, погребенный в исходном коде rcslib, очевидно, что применимо следующее:

timeout > 0 Блокировка доступа до тех пор, пока не будет достигнут интервал тайм-аута или появится доступ к буферу.

timeout = 0 Доступ к буферу возможен только в том случае, если никакой другой процесс не читает и не пишет в это время.

timeout < 0 или INF Доступ заблокирован до тех пор, пока буфер не будет доступен.

3.22 базовый класс NML

Раскройте списки и взаимосвязь между NML, NMLmsg и классами cms более низкого уровня.

Не путать с NMLmsg, RCS_STAT_MSG или RCS_CMD_MSG.

NML отвечает за синтаксический анализ файла конфигурации, настройку буферов cms и является механизмом маршрутизации сообщений в правильный(е) буфер(ы). Для этого NML создает несколько списков для:

- буферы cms, созданные или подключенные к ним.
- процессы и буферы, к которым они подключаются
- длинный список функций формата для каждого типа сообщения

Этот последний пункт, вероятно, является основной причиной несоответствия libnml/rcslib и NML в целом. Каждое сообщение, передаваемое через NML, требует присоединения определенного объема информации в дополнение к фактическим данным. Для этого последовательно вызываются несколько функций форматирования для сборки фрагментов общего сообщения. Функции формата будут включать NML_TYPE, MSG_TYPE в дополнение к данным, объявленным в производных классах NMLmsg. Изменения в порядке, в котором вызываются функции форматирования, а также передаваемые переменные, нарушат совместимость с rcslib, если с ними что-то не так. Существуют причины для поддержания совместимости с rcslib и веские причины для вмешательства в код. Вопрос в том, какой набор причин превалирует?

3.22.1 Внутреннее устройство NML

3.22.1.1 Конструктор NML

NML::NML() анализирует файл конфигурации и сохраняет его в связанном списке для передачи в конструкторы cms в виде отдельных строк. Функция конструктора NML состоит в вызове соответствующего конструктора cms для каждого буфера и ведении списка объектов cms и процессов, связанных с каждым буфером.

Именно с помощью указателей, хранящихся в списках, NML может взаимодействовать с cms, и поэтому Doxygen не может показать реальные отношения.

Note
Конфиг хранится в памяти до передачи указателя на конкретную строку конструктору cms. Затем конструктор cms снова анализирует строку, чтобы извлечь пару переменных… Было бы разумнее выполнить ВЕСЬ анализ и сохранить переменные в структуре, которая передается конструктору cms. Это устранит обработку строк и уменьшит количество дубликатов кода в cms…

3.22.1.2 Чтение/запись NML

Вызовы NML::read и NML::write выполняют схожие задачи в том, что касается обработки сообщения. Единственная реальная разница заключается в направлении потока данных.

Вызов функции чтения сначала получает данные из буфера, затем вызывает format_output(), в то время как функция записи вызывает format_input() перед передачей данных в буфер. Именно в format_xxx() происходит работа по построению или деконструкции сообщения. Список различных функций вызывается по очереди для размещения различных частей заголовка NML (не путать с заголовком cms) в правильном порядке. Последней вызываемой функцией является emcFormat() в emc.cc.

3.22.1.3 Отношения NMLmsg и NML

NMLmsg — это базовый класс, от которого происходят все классы сообщений. Каждый класс сообщения должен иметь уникальный идентификатор, определенный (и переданный конструктору), а также функцию обновления (*cms). Функция update() будет вызываться функциями чтения/записи NML при вызове модуля форматирования NML — указатель на модуль форматирования будет объявлен в конструкторе NML в какой-то момент. Благодаря связанным спискам, которые создает NML, он может выбирать указатель cms, который передается форматтеру, и, следовательно, какой буфер использовать.

3.23 Добавление пользовательских команд NML

LinuxCNC довольно крут, но некоторые части нуждаются в правках. Как вы знаете, связь осуществляет через каналы NML, данные, отправляемые через такой канал, являются одним из классов, определенн в emc.hh (реализованных в emc.cc). Если кому-то нужен несуществующий тип сообщения, он должен выполнить следующие шаги, чтобы добавить новый. (Сообщение, которое я добавил в примере, называется EMC_IO_GENERIC (наследует EMC_IO_CMD_MSG (наследует RCS_CMD_MSG)))

1. добавить определение класса EMC_IO_GENERIC в emc2/src/emc/nml_intf/emc.hh
2. добавьте определение типа: #define EMC_IO_GENERIC_TYPE ((NMLTYPE) 1605)

    a. (я выбрал 1605, потому что он был доступен) в emc2/src/emc/nml_intf/emc.hh
3. добавить блок EMC_IO_GENERIC_TYPE в emcFormat в emc2/src/emc/nml_intf/emc.cc
4. добавить блок EMC_IO_GENERIC_TYPE в emc_symbol_lookup в emc2/src/emc/nml_intf/emc.cc
5. добавить функцию EMC_IO_GENERIC::update в emc2/src/emc/nml_intf/emc.cc

Перекомпилируйте, и новое сообщение должно быть там. Следующая часть — отправлять такие сообщения откуда-то, получать их в другом месте и что-то с этим делать.

3.24 Таблица инструментов и устройство смены инструмента

LinuxCNC взаимодействует с оборудованием смены инструмента и имеет внутреннюю абстракцию смены инструмента. LinuxCNC управляет информацией об инструментах в файле таблицы инструмент

3.24.1 Абстракция смены инструмента в LinuxCNC

LinuxCNC поддерживает два типа аппаратных средств смены инструмента, называемых *nonrandom* и *random*. Параметр INI [EMCIO]RANDOM_TOOLCHANGER определяет как LinuxCNC будет решать к какому из этих типов оборудования он подключен.

3.24.1.1 Неслучайная смена инструмента

Устройство автоматической смены инструмента возвращает каждый инструмент в гнездо, из которого он был первоначально взят.

Примерами неслучайного оборудования для смены инструмента являются «ручная» смена инструмент револьверные головки токарных станков и реечные устройства смены инструмента.

При настройке для неслучайного устройства смены инструмента LinuxCNC не изменяет номер гнезда в файле таблицы инструментов при загрузке и выгрузке инструментов. Внутри LinuxCNC, при смене инструмента информация об инструменте **копируется** из исходного кармана таблицы инструментов в карман 0 (который представляет шпиндель), заменяя любую информацию об инструменте, которая была там ранее.

Note
В LinuxCNC, сконфигурированном для неслучайной смены инструмента, инструмент 0 (T0) имеет особое значение: «нет инструмента». T0 может не отображаться в файле таблицы инструментов, и изменение на T0 приведет к тому, что LinuxCNC будет думать, что у него пустой шпиндель.

3.24.1.2 Устройства случайной замены инструмента

Устройство случайной смены инструмента меняет инструмент в шпинделе (если есть) на запрошенный инструмент при смене инструмента. Таким образом, карман, в котором находится инструмент, изменяется по мере того, как он вставляется в шпиндель и вынимается из него.

Примером оборудования случайного устройства смены инструмента является устройство смены инструмента карусельного типа.

При настройке случайного устройства смены инструмента LinuxCNC меняет местами номер кармана старого и нового инструмента в файле таблицы инструментов при загрузке инструментов. Внутри LinuxCNC, при смене инструмента информация об инструменте **переставляется** между исходным гнездом таблицы инструментов и гнездом 0 (которое представляет шпиндель). Таким образом, после смены инструмента карман 0 в таблице инструментов содержит информацию об инструменте для нового инструмента, а карман, из которого был получен новый инструмент, содержит информацию об инструменте для старого инструмента (инструмент, который был в шпинделе до смены инструмента) если такой есть.

Note
Если LinuxCNC настроен на случайную смену инструмента, инструмент 0 (T0) не имеет специального значения. Он обрабатывается точно так же, как и любой другой инструмент в таблице инструментов. Обычно T0 используется для обозначения «отсутствия инструмента» (т. е. инструмента с нулевым TLO), чтобы при необходимости шпиндель мог быть удобно разгружен.

3.24.2 Таблица инструментов

LinuxCNC отслеживает инструменты в файле tool table. В таблице инструментов записывается следующая информация для каждого инструмента:

номер инструмента
Целое число, которое однозначно идентифицирует этот инструмент. Номера инструментов обрабатываются LinuxCNC по-разному при настройке случайных и неслучайных устройств смены инструмента:

- Когда LinuxCNC настроен для неслучайного устройства смены инструмента, это число должно быть положительным. T0 подвергается специальной обработке и не может отображаться в таблице инструментов.
- Когда LinuxCNC настроен для случайного устройства смены инструмента, это число должно быть неотрицательным. T0 допускается в таблице инструментов и обычно используется для обозначения отсутствия инструмента, то есть пустого кармана.

номер гнезда
Целое число, идентифицирующее гнездо или паз в оборудовании устройства смены инструмента, в котором находится инструмент. Номера карманов обрабатываются LinuxCNC по-разному при настройке случайных и неслучайных устройств смены инструмента:

- Когда LinuxCNC сконфигурирован для неслучайного устройства смены инструмента, номер гнезда в файле инструмента может быть любым положительным целым числом (ячейка 0 не допускается). LinuxCNC молча уплотняет номера гнезд при загрузке файла инструмента, поэтому может быть разница между номерами гнезд в файле инструмента и внутренними номерами гнезд, используемыми LinuxCNC с неслучайным устройством смены инструмента.
- Когда LinuxCNC настроен для случайного устройства смены инструмента, номера карманов в файле инструмента должны быть в диапазоне от 0 до 1000 включительно. Гнезда 1-1000 находятся в устройстве смены инструмента, гнездо 0 - это шпиндель.

diameter
Диаметр инструмента в единицах измерения станка.

смещение длины инструмента
Коррекция длины инструмента (также называемая TLO), до 9 осей, в единицах измерения станка. Оси, у которых нет указанного TLO, получают 0.

3.24.3 G-коды, влияющие на инструменты

G-коды, которые используют или влияют на информацию об инструменте:

3.24.3.1 Txxx

Указывает оборудованию смены инструмента подготовиться к переключению на указанный инструмен
xxx.

Обрабатывается `Interp::convert_tool_select()`.

1. Станку предлагается подготовиться к переключению на выбранный инструмент, путем вызова Canon функции `SELECT_TOOL()` с номером запрошенного инструмента.
   a. (saicanon) No-op.
   b. (emccanon) Создает сообщение `EMC_TOOL_PREPARE` с запрошенным номером кармана и отправляет его в Task, который отправляет его в IO. IO получает сообщение и просит HAL подготовить карман, установив `iocontrol.0.tool-prep-pocket`, `iocontrol.0.tool-prep` и `iocontrol.0.tool-prepare`. Затем IO повторно вызывает `read_tool_inputs()` для опроса вывода HAL `iocontrol.0.tool-prepared`, который сигнализирует от оборудования смены инструмента через HAL IO, что запрошенная подготовка инструмента завершена. Когда этот вывод становится True, IO устанавливает `emcioStatus.tool.pocketPrepped` на номер кармана запрошенного инструмента.
2. Вернувшись в интерпретацию, `settings->selected_pocket` присваивается индекс tooldata запрошенного инструмента *xxx*.

Note
Устаревшие имена **selected_pocket** и **current_pocket** на самом деле ссылаются на последовательный индекс данных инструмента для элементов инструмента, загруженных из таблицы инструментов ([EMCIO]TOOL_TABLE) или через базу данных инструментов ([EMCIO]DB_PROGRAM).

3.24.3.2 M6

Сообщает устройству смены инструмента переключиться на текущий выбранный инструмент (выбранный предыдущей командой Txxx).

Обрабатывается `Interp::convert_tool_change()`.

1. Станку предлагается перейти на выбранный инструмент, путем вызова Canon функции `CHANGE_TO` с `settings->selected_pocket` (индекс данных инструмента).
   a. (saicanon) Устанавливает sai's `_active_slot` в переданный номер кармана. Информация об инструменте копируется из выбранного кармана таблицы инструментов (т. е. из sai's `_tools[_active_slot]`) на шпиндель (он же sai's `_tools[0]`).

b. (emccanon) Отправляет сообщение EMC_TOOL_LOAD в Task, которое отправляет его в IO. IO устанавливает emcioStatus.tool.toolInSpindle на номер инструмента в гнезде, указанно emcioStatus.tool.pocketPrepped (устанавливается Txxx, также известным как SELECT_TOOL(Затем он запрашивает, чтобы оборудование смены инструмента выполнило смену инструмент установив для контакта HAL iocontrol.0.tool-change значение True. Позже read_tool_inpu IO обнаружит, что контакт HAL iocontrol.0.tool_changed был установлен в True, указывая на то, что устройство смены инструмента завершило смену инструмента. Когда это происходит, он вызывает load_tool() для обновления состояния станка.

i. load_tool() с конфигурацией устройства неслучайной смены инструмента копирует информацию об инструменте из выбранного гнезда в шпиндель (карман 0).

ii. load_tool() с конфигурацией устройства случайной смены инструмента меняет местами информацию об инструменте между гнездом 0 (шпиндель) и выбранным гнездом, затем сохраняет таблицу инструментов.

2. Вернувшись в интерпретатор, settings->current_pocket назначается новый индекс tool-data из settings->selected_pocket (устанавливается Txxx). Соответствующие пронумерованные параметры (#5400-#5413) обновляются новой информацией об инструменте из гнезда 0 (шпиндель).

3.24.3.3 G43/G43.1/G49

Применить коррекцию длины инструмента. G43 использует TLO загруженного в данный момент инструмента или указанного инструмента, если H-слово задано в блоке. G43.1 получает TLO из осевых слов в блоке. G49 отменяет TLO (использует 0 для смещения для всех осей).

Обрабатывается Interp::convert_tool_length_offset().

1. Он начинается со сборки EmcPose, содержащего для использования смещения 9-и осей. Для G43.1 эти коррекции инструмента берутся из слов осей в текущем кадре. Для G43 эти коррекции исходят от текущего инструмента (инструмент в гнезде 0) или от инструмента, указанного H-словом в блоке. Для G49 все смещения равны 0.

2. Смещения передаются Canon функции USE_TOOL_LENGTH_OFFSET().

   a. (saicanon) Записывает TLO в _tool_offset.

   b. (emccanon) Создает сообщение EMC_TRAJ_SET_OFFSET, содержащее смещения, и отправляет его в Task. Task копирует смещения в emcStatus->task.toolOffset и отправляет их в Motion с помощью команды EMCMOT_SET_OFFSET. Движение копирует смещения в emcmotStatu где оно используется для смещения будущих перемещений.

3. Вернувшись в интерпретацию, смещения записываются в settings->tool_offset. Эффективное гнездо записывается в settings->tool_offset_index, хотя это значение никогда не используется

3.24.3.4 G10 L1/L10/L11

Изменяет таблицу инструментов.

Обрабатывается Interp::convert_setup_tool().

1. Picks the tool number out of the P-word in the block and finds the pocket for that tool:

   a. With a nonrandom toolchanger config this is always the pocket number in the toolchanger (even when the tool is in the spindle).

   b. With a random toolchanger config, if the tool is currently loaded it uses pocket 0 (pocket 0 means ”the spindle”), and if the tool is not loaded it uses the pocket number in the tool changer. (This difference is important.)

2. Figures out what the new offsets should be.
3. The new tool information (diameter, offsets, angles, and orientation), along with the tool number and pocket number, are passed to the Canon call SET_TOOL_TABLE_ENTRY().
   a. (saicanon) Copy the new tool information to the specified pocket (in sai's internal tool table, `_tools`).
   b. (emccanon) Build an `EMC_TOOL_SET_OFFSET` message with the new tool information, and send it to Task, which passes it to IO. IO updates the specified pocket in its internal copy of the tool table (`emcioStatus.tool.toolTable`), and if the specified tool is currently loaded (it is compared to `emcioStatus.tool.toolInSpindle`) then the new tool information is copied to pocket 0 (the spindle) as well. (FIXME: that's a buglet, should only be copied on nonrandom machines.) Finally IO saves the new tool table.
4. Back in interp, if the modified tool is currently loaded in the spindle, and if the machine is a non-random toolchanger, then the new tool information is copied from the tool's home pocket to pocket 0 (the spindle) in interp's copy of the tool table, `settings->tool_table`. (This copy is not needed on random tool changer machines because there, tools don't have a home pocket and instead we just updated the tool in pocket 0 directly.). The relevant numbered parameters (#5400-#5413) are updated from the tool information in the spindle (by copying the information from interp's `settings->tool_table` to `settings->parameters`). (FIXME: this is a buglet, the params should only be updated if it was the current tool that was modified).
5. If the modified tool is currently loaded in the spindle, and if the config is for a nonrandom toolchanger, then the new tool information is written to the tool table's pocket 0 as well, via a second call to SET_TOOL_TABLE_ENTRY(). (This second tool-table update is not needed on random toolchanger machines because there, tools don't have a home pocket and instead we just updated the tool in pocket 0 directly.)

3.24.3.5 M61

Set current tool number. This switches LinuxCNC's internal representation of which tool is in the spindle, without actually moving the toolchanger or swapping any tools.

Обрабатывается `Interp::convert_tool_change()`.

Canon: `CHANGE_TOOL_NUMBER()`

settings->current_pocket is assigned the tooldata index currently holding the tool specified by the Q-word argument.

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

Enable cutter radius compensation (usually called *cutter comp*).

Handled by `Interp::convert_cutter_compensation_on()`.

No Canon call, cutter comp happens in the interpreter. Uses the tool table in the expected way: if a D-word tool number is supplied it looks up the pocket number of the specified tool number in the table, and if no D-word is supplied it uses pocket 0 (the spindle).

3.24.3.7 G40

Cancel cutter radius compensation.

Handled by `Interp::convert_cutter_compensation_off()`.

No Canon call, cutter comp happens in the interpreter. Does not use the tool table.

3.24.4 Internal state variables

This is not an exhaustive list! Tool information is spread through out LinuxCNC.

3.24.4.1 IO

`emcioStatus` is of type `EMC_IO_STAT`

emcioStatus.tool.pocketPrepped
When IO gets the signal from HAL that the toolchanger prep is complete (after a `Txxx` command), this variable is set to the pocket of the requested tool. When IO gets the signal from HAL that the tool change itself is complete (after an `M6` command), this variable gets reset to -1.

emcioStatus.tool.toolInSpindle
Tool number of the tool currently installed in the spindle. Exported on the HAL pin `iocontrol.0.tool-r` (s32).

emcioStatus.tool.toolTable[]
An array of `CANON_TOOL_TABLE` structures, `CANON_POCKETS_MAX` long. Loaded from the tool table file at startup and maintained there after. Index 0 is the spindle, indexes 1-(CANON_POCKETS_MAX-1) are the pockets in the toolchanger. This is a complete copy of the tool information, maintained separately from Interp's `settings.tool_table`.

3.24.4.2 interp

`settings` is of type `settings`, defined as `struct setup_struct` in `src/emc/rs274ngc/interp_internal.hh`

settings.selected_pocket
Tooldata index of the tool most recently selected by `Txxx`.

settings.current_pocket
Original tooldata index of the tool currently in the spindle. In other words: which tooldata index the tool that's currently in the spindle was loaded from.

settings.tool_table[]
An array of tool information. The index into the array is the "pocket number" (aka "slot number"). Pocket 0 is the spindle, pockets 1 through (CANON_POCKETS_MAX-1) are the pockets of the toolchanger.

settings.tool_offset_index
Unused. FIXME: Should probably be removed.

settings.toolchange_flag
Interp sets this to true when calling Canon's CHANGE_TOOL() function. It is checked in `Interp::conve` to decide which tooldata index to use for G43 (with no H-word): `settings->current_pocket` if the tool change is still in progress, tooldata index 0 (the spindle) if the tool change is complete.

settings.random_toolchanger
Set from the INI variable `[EMCIO]RANDOM_TOOLCHANGER` at startup. Controls various tool table handling logic. (IO also reads this INI variable and changes its behavior based on it. For example, when saving the tool table, random toolchanger save the tool in the spindle (pocket 0), but non-random toolchanger save each tool in its "home pocket".)

settings.tool_offset
This is an `EmcPose` variable.

- Used to compute position in various places.

- Отправляется в Motion через сообщение EMCMOT_SET_OFFSET. Все, что движение делает со смещениями, — это экспортирует их на контакты HAL motion.0.tooloffset.[xyzabcuvw]. FIXME: экспортируйте их куда-нибудь ближе к таблице инструментов (вероятно, io или interp) и удалите сообщение EMCMOT_SET_OFFSET.

settings.pockets_max
Used interchangeably with CANON_POCKETS_MAX (a #defined constant, set to 1000 as of April 2020). FIXME: This settings variable is not currently useful and should probably be removed.

settings.tool_table
This is an array of CANON_TOOL_TABLE structures (defined in src/emc/nml_intf/emctool.h), with CANON_POCKETS_MAX entries. Indexed by "pocket number", aka "slot number". Index 0 is the spindle, indexes 1 to (CANON_POCKETS_MAX-1) are the pockets in the tool changer. On a random toolchanger pocket numbers are meaningful. On a nonrandom toolchanger pockets are meaningless; the pocket numbers in the tool table file are ignored and tools are assigned to tool_table slots sequentially.

settings.tool_change_at_g30 , settings.tool_change_quill_up , settings.tool_change_with_spindle_

These are set from INI variables in the [EMCIO] section, and determine how tool changes are performed.

3.25 Reckoning of joints and axes

3.25.1 In the status buffer

The status buffer is used by Task and the UIs.

FIXME: axis_mask and axes overspecify the number of axes

status.motion.traj.axis_mask
A bitmask with a "1" for the axes that are present and a "0" for the axes that are not present. X is bit 0 with value $2^0 = 1$ if set, Y is bit 1 with value $2^1 = 2$, Z is bit 2 with value 4, etc. For example, a machine with X and Z axes would have an axis_mask of 0x5, an XYZ machine would have 0x7, and an XYZB machine would have an axis_mask of 0x17.

status.motion.traj.axes (removed)
This value was removed in LinuxCNC version 2.9. Use axis_mask instead.

status.motion.traj.joints
A count of the number of joints the machine has. A normal lathe has 2 joints; one driving the X axis and one driving the Z axis. An XYYZ gantry mill has 4 joints: one driving X, one driving one side of the Y, one driving the other side of the Y, and one driving Z. An XYZA mill also has 4 joints.

status.motion.axis[EMCMOT_MAX_AXIS]
An array of EMCMOT_MAX_AXIS axis structures. axis[n] is valid if (axis_mask & (1 << n)) is True. If (axis_mask & (1 << n)) is False, then axis[n] does not exist on this machine and must be ignored.

status.motion.joint[EMCMOT_MAX_JOINTS]
An array of EMCMOT_MAX_JOINTS joint structures. joint[0] through joint[joints-1] are valid, the others do not exist on this machine and must be ignored.

Things are not this way currently in the joints-axes branch, but deviations from this design are considered bugs. For an example of such a bug, see the treatment of axes in src/emc/ini/initraj.cc:loadTraj(). There are undoubtedly more, and I need your help to find them and fix them.

3.25.2 In Motion

Компонент реального времени контроллера движения сначала получает количество соединений из параметра времени загрузки `num_joints`. Это определяет, сколько сочленений заслуживающих контактов HAL будет создано при запуске.

Motion’s number of joints can be changed at runtime using the `EMCMOT_SET_NUM_JOINTS` command from Task.

Контроллер движения всегда работает по осям EMCMOT_MAX_AXIS. Он всегда создает девять наборов контактов `axis.*.*`.

Chapter 4

Сообщения NML

Список сообщений NML.
Подробности смотрите в `src/emc/nml_intf/emc.hh`.

4.1 OPERATOR

```
EMC_OPERATOR_ERROR_TYPE
EMC_OPERATOR_TEXT_TYPE
EMC_OPERATOR_DISPLAY_TYPE
```

4.2 JOINT

```
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 (Input/Output)

```
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 Другие

```
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 кодирования

В этой главе описывается стиль исходного кода, предпочитаемый командой разработчиков LinuxCNC.

5.1 Не навреди

При внесении небольших изменений в код в стиле, отличном от описанного ниже, соблюдайте местный стиль кодирования. Быстрые переходы от одного стиля кодирования к другому ухудшают читабельность кода.

Никогда не проверяйте код после выполнения над ним «отступа». Изменения пробелов, внесенные отступом, затрудняют отслеживание истории изменений файла.

Не используйте редактор, который вносит ненужные изменения в пробельные символы (например, заменяет 8 пробелов табулятором на строке, не измененной иным образом, или переносит строки, не измененные иным образом).

5.2 Табуляторы

Табуляция всегда соответствует 8 пробелам. Не пишите код, который правильно отображается только с другим значением позиции табуляции.

5.3 Отступ

Используйте 4 пробела на уровень отступа. Объединение 8 пробелов в один табулятор допустимо, но не обязательно.

5.4 Размещение скобок

Поместите открывающую фигурную скобку последней в строке и поставьте закрывающую фигурную скобку первой:

```
if (x) {
    // b''cb''b''дb''b''eb''b''лb''b''ab''b''тb''b''ьb'' b''чb''b''тb''b''ob''-b''тb''b' ↩
        'ob'' b''пb''b''ob''b''дb''b''xb''b''ob''b''дb''b''яb''b''щb''b''eb''b''eb''
}
```

Закрывающая фигурная скобка находится на отдельной строке, за исключением случаев, когда за ней следует продолжение того же оператора, т. е. «while» в do-операторе или «else» в if-операторе , например :

```
do {
    // b''чb''b''тb''b''ob''-b''тb''b''ob'' b''вb''b''ab''b''жb''b''нb''b''ob''b''eb''
} while (x > 0);
```

и

```
if (x == y) {
    // b''дb''b''eb''b''лb''b''ab''b''тb''b''ьb'' b''ob''b''дb''b''нb''b''ob''
} else if (x < y) {
    // b''дb''b''eb''b''лb''b''ab''b''тb''b''ьb'' b''дb''b''pb''b''yb''b''гb''b''ob''b' ↩
        'eb''
} else {
    // b''дb''b''eb''b''лb''b''ab''b''тb''b''ьb'' b''тb''b''pb''b''eb''b''тb''b''ьb''b' ↩
        'eb''
}
```

Такое размещение фигурных скобок также минимизирует количество пустых (или почти пустых) строк, что позволяет одновременно отображать большее количество кода или комментариев в терминале фиксированного размера.

5.5 Именование

C — спартанский язык, и таким же должно быть ваше именование. В отличие от программистов Modula-2 и Pascal, программисты C не используют милые имена, такие как ThisVariableIsATemporaryCounter. Программист на C назвал бы эту переменную *tmp*, что намного проще написать и не в последнюю очередь сложнее понять.

Однако описательные имена для глобальных переменных обязательны. Назвать глобальную функцию «foo» — это преступление.

ГЛОБАЛЬНЫЕ переменные (используются только в том случае, если они **действительно** нужны) должны иметь описательные имена, как и глобальные функции. Если у вас есть функция, которая подсчитывает количество активных пользователей, вы должны назвать это *count_active_users()* или аналогично, вы **не** должны называть ее *cntusr()*

Кодирование типа функции в имени (так называемая венгерская нотация) — это мозговой косяк — компилятор и так знает типы и может их проверить, а это только путает программиста. Неудивительно, что Microsoft делает глючные программы.

Имена ЛОКАЛЬНЫХ переменных должны быть короткими и точными. Если у вас есть счетчик циклов со случайными целыми числами, его, вероятно, следует назвать «i». Называть его *loop_counter* непродуктивно, если нет шансов, что его неправильно поймут. Точно так же «tmp» может быть переменной любого типа, которая используется для хранения временного значения.

Если вы боитесь перепутать имена локальных переменных, у вас есть другая проблема, которая называется синдромом дисбаланса функции-гормона-роста. См. следующую главу.

5.6 Функции

Функции должны быть короткими и приятными, и делать только одну вещь. Они должны умещаться на одном или двух экранах текста (как мы все знаем, размер экрана ISO/ANSI составляет 80x24) и делать одну вещь, и делать это хорошо.

Максимальная длина функции обратно пропорциональна сложности и уровню отступа этой функции. Таким образом, если у вас есть концептуально простая функция, представляющая собой всего лишь один длинный (но простой) case-оператор, в котором вам нужно делать множество мелких действий для множества разных случаев, можно использовать более длинную функцию.

Однако, если у вас есть сложная функция, и вы подозреваете, что менее одаренный первокурсник может даже не понимать, о чем эта функция, вам следует еще более строго придерживаться максимальных ограничений. Используйте вспомогательные функции с описательными именами (вы можете попросить компилятор встроить их (in-line), если считаете, что это критично для производительности, и он, вероятно, справится с этой задачей лучше, чем вы).

Другой мерой функции является количество локальных переменных. Их не должно быть больше 5-10, иначе вы что-то делаете не так. Переосмыслите функцию и разделите ее на более мелкие части. Человеческий мозг обычно может легко отслеживать около 7 разных вещей, если больше, то он запутается. Вы знаете, что вы гениальны, но, возможно, вы хотели бы понять, что вы сделали через 2 недели.

5.7 Комментирование

Комментарии — это хорошо, но существует опасность их чрезмерного комментирования. НИКОГДА не пытайтесь объяснить в комментарии, КАК работает ваш код: гораздо лучше написать код так, чтобы **работа** была очевидна, а объяснять плохо написанный код — пустая трата времени.

Как правило, вы хотите, чтобы ваши комментарии сообщали, ЧТО делает ваш код, а не КАК. Комментарий в рамке с описанием функции, возвращаемого значения и того, кто ее вызывает, помещается над телом — это хорошо. Кроме того, старайтесь избегать размещения комментариев внутри тела функции: если функция настолько сложна, что вам нужно отдельно комментировать ее части, вам, вероятно, следует еще раз перечитать раздел «Функции». Вы можете делать небольшие комментарии, чтобы отметить или предупредить о чем-то особенно умном (или некрасивом но старайтесь избегать излишеств. Вместо этого поместите комментарии в начале функции, рассказывая людям, что она делает и, возможно, ПОЧЕМУ она это делает.

Если используются комментарии типа /* Fix me */, пожалуйста, скажите, почему что-то нужно исправить. Если в затронутую часть кода было внесено изменение, либо удалите комментарий, либо измените его, чтобы указать, что изменение было внесено и требует тестирования.

5.8 Скрипты оболочки и файлы Makefile

Не у всех установлены одинаковые инструменты и пакеты. Некоторые люди используют vi, другие emacs. Некоторые даже избегают установки любого пакета, предпочитая легкий текстовый редактор, такой как nano или встроенный в Midnight Commander.

gawk против mawk. Опять же, не у всех будет установлен gawk, mawk почти в десять раз меньше по размеру и при этом соответствует стандарту POSIX AWK. Если требуется какая-то непонятная команда, специфичная для gawk, которую mawk не предоставляет, то у некоторых пользователей скрипт не работает. То же самое относится и к mawk. Короче говоря, используйте общий вызов awk вместо gawk или mawk.

5.9 Соглашения C++

Обсуждение стилей программирования C++ всегда заканчиваются жаркими дебатами (что-то вроде споров между emacs и vi). Одно можно сказать наверняка: общий стиль, используемый всеми, кто работает над проектом, приводит к единообразному и удобочитаемому коду.

Соглашения об именах: константы из #define или enumerations должны быть в верхнем регистре. Обоснование: упрощает обнаружение констант в момент компиляции в исходном коде, например, EMC_MESSAGE_TYPE.

Classes и Namespaces должны использовать первую букву каждого слова с заглавной буквы и избегать подчеркивания. Обоснование: Идентифицируют classes, constructors и destructors, например, GtkWidget.

Методы (или имена функций) должны соответствовать приведенным выше рекомендациям C и не должны включать имя класса. Обоснование: поддерживает общий стиль для исходных кодов C и C++, например, get_foo_bar().

Однако логические методы легче читать, если они избегают символов подчеркивания и используют префикс «is» (не путать с методами, которые манипулируют логическими значениями). Обоснование: Идентифицирует возвращаемое значение как TRUE или FALSE и ничего больше, например, isOpen, isHomed.

НЕ используйте *Not* в логическом имени, это приводит только к путанице при выполнении логических тестов, например, isNotOnLimit или is_not_on_limit - это ПЛОХО.

В именах переменных следует избегать использования верхнего регистра и символов подчеркивания, за исключением локальных или частных имен. Следует по возможности избегать использования глобальных переменных. Обоснование: разъясняет, какие из них являются переменными, а какие методами. Общедоступный: например, axislimit Private: например, maxvelocity_ .

5.9.1 Конкретные соглашения об именовании методов

Термины get и set следует использовать там, где доступ к атрибуту осуществляется напрямую. Обоснование: указывает назначение функции или метода, например, get_foo set_bar.

Для методов, использующих логические атрибуты, предпочтительнее использовать set & reset. Обоснование: как указано выше. например set_amp_enable reset_amp_fault

Методы интенсивно использующие вычисления должны использовать compute в качестве префикса. Обоснование: показывает, что это требует больших вычислительных ресурсов и загружает ЦП. например compute_PID

По возможности следует избегать сокращений в именах. Исключение составляют имена локальных переменных. Обоснование: Ясность кода. например pointer предпочтительнее, чем ptr, compute предпочтительнее, чем cmp, compare снова предпочтительнее, чем cmp.

Enumerates и другие константы могут иметь префикс общего имени типа, например, `enum COLOR { COLOR_RED, COLOR_BLUE };`.

Следует избегать чрезмерного использования макросов и определений. Предпочтительно использовать простые методы или функции. Обоснование: улучшает процесс отладки.

Файлы заголовков Include Statements должны быть включены в начало исходного файла, а не разбросаны по всему тексту. Они должны быть отсортированы и сгруппированы по их иерархическому положению в системе, при этом файлы низкого уровня должны включаться в первую очередь. Включаемые пути к файлам НИКОГДА не должны быть абсолютными. Вместо этого используйте флаг компилятора -I, чтобы расширить путь поиска. Обоснование: заголовки могут быть не в одном и том же месте во всех системах.

Указатели и ссылки должны иметь символ ссылки рядом с именем переменной, а не с именем типа. Обоснование: уменьшает путаницу, например, `float *x` или `int &i`.

Неявные проверки нуля не должны использоваться, за исключением булевых переменных, например, `if (spindle_speed != 0)` НЕ `if (spindle_speed)`.

В конструкцию for() должны быть включены только операторы управления циклом, например,
`sum = 0; for (i=0; i<10; i++) { sum += value[i]; }`
НЕ: `for (i=0, sum=0; i<10; i++) sum += value[i];`.

Точно так же следует избегать исполняемых операторов в условных операторах, например, `if (fd = open(file_name)` неверный.

Следует избегать сложных условных операторов. Вместо этого используйте временные логические переменные.

Скобки должны использоваться в математических выражениях в большом количестве. Не полагайтесь на приоритет операторов, когда лишние скобки могут прояснить ситуацию.

Имена файлов: исходники и заголовки C++ используют расширения .cc и .hh. Использование .c и .h зарезервировано для простого C. Заголовки предназначены для объявлений классов, методов и структур, а не для кода (если только функции не объявлены встроенными).

5.10 Стандарты кодирования Python

Используйте стиль PEP 8 для кода Python.

5.11 Стандарты кодирования .comp

В части объявлений файла .comp каждое объявление начинается с первого столбца. Вставляйте дополнительные пустые строки, когда они помогают сгруппировать связанные элементы.

In the code portion of a .comp file, follow normal C coding style.

Chapter 6

Сборка LinuxCNC

6.1 Введение

В этом документе описывается, как собрать программное обеспечение LinuxCNC из исходного кода. Это в первую очередь полезно, если вы разработчик, модифицирующий LinuxCNC. Это также может быть полезно, если вы являетесь пользователем, тестирующим ветки разработчиков, хотя в этом случае у вас также есть возможность просто установить пакеты Debian из buildbot (http://buildbot.linuxcnc.org) или как обычный пакет из вашего Дистрибутив Linux (https://tracker.debian pkg/linuxcnc). Следует признать, что этот раздел также существует, поскольку LinuxCNC — это работа сообщества, и вам предлагается внести свой вклад в развитие LinuxCNC. Как правило, вы хотите скомпилировать LinuxCNC самостоятельно для немедленного функционального доступа * к новой разработке LinuxCNC или * новая разработка вы возможно хотите внести вклад в LinuxCNC или помочь другим завершить его.

Например, вы можете переносить LinuxCNC на какой-то новый дистрибутив Linux или, что часто бывает, разработчик реагирует на ваш отчет о проблеме, исправление которой вы хотите протестирова Любое такое изменение не увидит buildbot, который мог бы помочь, или эта помощь будет отложена, ожидая чьего-то обзора, которого вы не хотите ждать, или вы единственный человек с определенным оборудованием для тестирования кода.

Помимо программ, управляющих вашим компьютером и созданных на основе исходного дерева, вы также можете создавать те же файлы PDF и/или HTML, которые вы, вероятно, встречали в сети по адресу https://linuxcnc.org/documents/.

Если вы хотите внести свой вклад в LinuxCNC, но не уверены, с чего начать, пожалуйста, серьезно рассмотрите возможность внесения вклада в документацию. Каждый всегда находит что-то, что можно улучшить - и если вы просто оставите ”FIXME: with a comment” в тексте в качестве ссылки для себя и других, чтобы вернуться к разделу позже. Кроме того, переводы на языки, отличные от английского, скорее всего, выиграют от вашего пристального внимания на https://hosted.weblate.org projects/linuxcnc/ .

6.2 Скачивание дерева исходников

Репозиторий git проекта LinuxCNC находится по адресу https://github.com/LinuxCNC/linuxcnc. GitHub — популярный сервис хостинга git и сайт для обмена кодом.

Для получения исходного дерева у вас есть два варианта:

Скачать архив
: На странице проекта LinuxCNC в GitHub найдите ссылку на ”релизы” или ”теги”, щелкните

гиперссылку на страницу архива и загрузите последний файл .tar. Вы найдете этот файл, сжатый как файл .tar.xz или .tar.gz. Этот файл, обычно называемый "tarball", является архивом, очень похожим на .zip. Ваш рабочий стол Linux будет знать, как обращаться с этим файлом, если дважды щелкнуть по нему.

Подготовьте локальную копию репозитория LinuxCNC
Сначала вам следует установить инструмент "git" на свой компьютер, если он еще не доступен (`sudo apt install git`). Затем подготовьте локальный экземпляр исходного дерева следующим образом: .

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

Первый аргумент команды git выдает это: Это называется "clone" репозитория LinuxCNC. Преимущест в том, что этот локальный клон поддерживает связь об изменениях, которые вы можете решить выполнить в исходном дереве.

GitHub — это инфраструктура сама по себе, и подробно она описана в другом месте. Просто чтобы мотивировать вас, если вы еще этого не знаете, предлагается выполнить клонирование для вас и сделать этот экземпляр общедоступным. GitHub называет такой дополнительный экземпляр другого репозитория "fork". Вы можете легко (и бесплатно) создать форк репозитория git LinuxCNC на GitHub и использовать его для отслеживания и публикации своих изменений. После создания собственного форка GitHub LinuxCNC клонируйте его на свой компьютер для разработки и продолжайте заниматься хакингом как обычно.

Мы из проекта LinuxCNC надеемся, что вы поделитесь с нами своими изменениями, чтобы сообщество могло извлечь пользу из вашей работы. GitHub делает этот обмен очень простым: после того, как вы отшлифуете свои изменения и отправите их в свой форк github, отправьте нам запрос Pull Request.

6.2.1 Быстрый старт

Для нетерпеливых попробуйте следующее:

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

Это, вероятно, не удастся! Это не делает вас плохим человеком, это просто означает, что вы должны прочитать весь этот документ, чтобы узнать как решить свои проблемы. Особенно раздел удовлетворение зависимостей сборки.

Если вы работаете в системе с поддержкой реального времени (например, при установке из образа LinuxCNC Live/Install Image, см. раздел В реальном времени ниже), в настоящее время требуется один дополнительный шаг сборки:

```
$ sudo make setuid
```

После того, как вы успешно собрали LinuxCNC, пришло время запустить тесты:

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

Это тоже может не получиться! Прочтите весь этот документ, но особенно раздел Настройка среды тестирования.

6.3 Поддерживаемые платформы

Проект LinuxCNC нацелен на современные дистрибутивы на основе Debian, включая Debian, Ubuntu и Mint. Мы постоянно тестируем на платформах, перечисленных на http://buildbot.linuxcnc.org.

LinuxCNC собирается на большинстве других дистрибутивов Linux, хотя управление зависимостями будет более ручным и менее автоматическим. Всегда приветствуются исправления для улучшения переносимости на новые платформы.

6.3.1 В реальном времени

LinuxCNC — это контроллер станка, и ему требуется платформа реального времени для этой работы. Эта версия LinuxCNC поддерживает следующие платформы. Первые три в списке являются операционными системами реального времени:

RTAI
: С сайта https://www.rtai.org. Ядро Linux с патчем RTAI доступно в архиве Debian по адресу https://linuxcnc.org. Инструкции по установке см. Получение LinuxCNC.

Xenomai
: С сайта https://xenomai.org. Вам придется скомпилировать или получить ядро Xenomai самостоятельно.

Preempt-RT
: С https://rt.wiki.kernel.org. Ядро Linux с патчем Preempt-RT иногда доступно в архиве Debian по адресу https://www.debian.org и в интернет-архиве по адресу https://snapshot.debian.org.

Не в реальном времени
: LinuxCNC также можно собрать и запустить на платформах, не работающих в реальном времени, таких как обычная установка Debian или Ubuntu без какого-либо специального ядра реального времени. . В этом режиме LinuxCNC бесполезен для управления станками, но он полезен для имитации выполнения G-кода и для тестирования частей системы, не работающих в реальном времени (таких как пользовательские интерфейсы и некоторые виды компонентов и драйверов устройств). . Чтобы использовать возможности LinuxCNC в реальном времени, некоторые части LinuxCNC должны работать с привилегиями root. Чтобы включить root для этих частей, запустите эту дополнительную команду после `make`, которая собирает LinuxCNC:

```
$ sudo make setuid
```

6.4 Режимы сборки

Существует два способа сборки LinuxCNC: удобный для разработчиков режим «запуска на месте» (run in place) и удобный для пользователя режим создания пакетов Debian.

6.4.1 Сборка для запуска на месте (RIP)

При сборке Run-In-Place программы LinuxCNC компилируются из исходного кода, а затем запускаются непосредственно из каталога сборки. Ничего не устанавливается за пределами каталога сборки. Это быстро и легко и подходит для быстрого внесения изменений. Набор тестов LinuxCNC работает только в сборке Run-In-Place. Большинство разработчиков LinuxCNC в основном используют этот режим.

Сборка для запуска на месте делается по шагам, описанным в разделе Быстрый старт в верхней части этого документа, возможно, с разными параметрами для `src/configure` и `make`.

6.4.1.1 `src/configure` параметры

Скрипт `src/configure` настраивает способ компиляции исходного кода. Он принимает много необязательных параметров. Вывести все аргументы `src/configure` запуском этого:

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

Наиболее часто используемые параметры:

`--with-realtime=uspace`
: Создавайте для любой платформы реального времени или для не-реального времени. Полученные исполняемые файлы LinuxCNC будут работать как на ядре Linux с патчами Preempt-RT (обеспечивающими управление машиной в реальном времени), так и на обычном (непропатченно ядре Linux (обеспечивающем симуляцию G-кода, но без управления машиной в реальном времени).

```
b''Eb''b''cb''b''лв''b''ив'' b''yb''b''cb''b''тв''b''ab''b''нв''b''ob''b''вв''b''лв''b' ↩
    'eb''b''нв''b''ыв'' b''фв''b''ab''b''йв''b''лв''b''ыв'' b''pb''b''ab''b''зв''b' ↩
    'pb''b''ab''b''бв''b''ob''b''тв''b''кв''b''ив'' b''дв''b''лв''b''яв'' Xenomai (b' ↩
    'ob''b''бв''b''ыв''b''чв''b''нв''b''ob'' b''ив''b''зв'' b''пв''b''ab''b''кв''b' ↩
    'eb''b''тв''b''ab'' libxenomai-dev) b''ив''b''лв''b''ив'' RTAI (b''ob''b''бв''b' ↩
    'ыв''b''чв''b''нв''b''ob'' b''ив''b''зв'' b''пв''b''ab''b''кв''b''eb''b''тв''b' ↩
    'ab'' b''cb'' b''ив''b''мв''b''eb''b''нв''b''eb''b''мв'', b''нв''b''ab''b''чв''b' ↩
    'ив''b''нв''b''ab''b''юв''b''щв''b''ив''b''мв''b''cb''b''яв'' b''cb'' "rtai-modules ↩
    "), b''пв''b''ob''b''дв''b''дв''b''eb''b''pb''b''жв''b''кв''b''ab'' b''эв''b''тв''b ↩
    ''ив''b''хв'' b''яв''b''дв''b''eb''b''pb'' b''pb''b''eb''b''ab''b''лв''b''ьв''b' ↩
    'нв''b''ob''b''гв''b''ob'' b''вв''b''pb''b''eb''b''мв''b''eb''b''нв''b''ив'' b' ↩
    'тв''b''ab''b''кв''b''жв''b''eb'' b''бв''b''yb''b''дв''b''eb''b''тв'' b''вв''b' ↩
    'кв''b''лв''b''юв''b''чв''b''eb''b''нв''b''ab''.
```

`--with-realtime=/usr/realtime-$VERSION`
: Сборка для платформы реального времени RTAI с использованием более старой модели "ядро реального времени". Для этого необходимо, чтобы у вас было ядро и модули RTAI, установленные в `/usr/realtime-$VERSION`. Полученные исполняемые файлы LinuxCNC будут работать только на указанном ядре RTAI. Начиная с LinuxCNC 2.7, это обеспечивает наилучшую производительность в реальном времени.

`--enable-build-documentation`
: Сборка документации в дополнение к исполняемым файлам. Эта опция значительно увеличивает время, необходимое для компиляции, так как сборка документации занимает довольно много времени. Если вы не активно работаете над документацией, вы можете опустить этот аргумент.

`--disable-build-documentation-translation`
: Отключить сборку переведенной документации для всех доступных языков. Сборка переведенной документации занимает огромное количество времени, поэтому рекомендуется пропустить это, если в этом нет особой необходимости.

6.4.1.2 Параметры `make`

Команда *make* принимает два полезных опциональных параметра.

Параллельная компиляция
: `make` принимает опциональный параметр `-j` *N* (где N — число). Это позволяет выполнять параллельную компиляцию с *N* одновременными процессами, что может значительно ускорить вашу сборку.

  Полезным значением *N* является количество ЦП в вашей системе сборки.

Вы можете узнать количество процессоров, запустив `nproc`.

Сборка только конкретного элемента

Если вы хотите собрать только определенный компонент LinuxCNC, вы можете указать на то, что хотите собрать, в командной строке `make`. Например, если вы работаете над компонентом с именем `froboz`, вы можете собрать его исполняемый файл, запустив:

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

6.4.2 Сборка Debian пакетов

При сборке пакетов Debian программы LinuxCNC компилируются из исходного кода и затем сохраняются в пакете Debian вместе с информацией о зависимостях. Этот процесс по умолчанию также включает сборку документации, которая занимает время из-за всех операций ввода-вывода для многих языков, но ее можно пропустить. Затем LinuxCNC устанавливается как часть этих пакетов на тот же компьютер или на любой другйо компьютер такой же архитектуры, на которую копируются файлы .deb. LinuxCNC нельзя запустить, пока пакеты Debian не будут установлены на целевом компьютере, а затем исполняемые файлы не будут доступны в /usr/bin и /usr/lib, как и другое обычное программное обеспечение системы.

Этот режим сборки в первую очередь полезен при упаковке программного обеспечения для доставки конечным пользователям, а также при сборке программного обеспечения для компьютера, на котором не установлена среда сборки или нет доступа к Интернету.

Сборка пакетов в первую очередь полезна при упаковке программного обеспечения для доставки конечным пользователям. Разработчики между собой обмениваются только исходным кодом, который, вероятно, поддерживается репозиторием LinuxCNC GitHub, указанным ниже. Кроме того, при создании программного обеспечения для машины, на которой не установлена среда сборки или нет доступа к Интернету, можно с радостью принять готовый пакет.

Сборка пакетов Debian выполняется с помощью инструмента `dpkg-buildpackage`, который предоставля
пакетом `dpkg-dev`. Его выполнение сопровождается рядом предварительных условий, которые подробно описаны ниже: * Должна быть установлена общая инфраструктура сборки, т.е. компиляторы и т.д. * должны быть установлены зависимости на момент сборки, т.е. заголовочные файлы для используемых внешних библиотек кода, как описано в разделе Удовлетворение зависимостей сборки. * файл в папке Debian должен быть полным и описывать пакет

Инструменты сборки были собраны в виртуальный пакет с именем `build-essential`. Чтобы установить его, выполните:

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

Как только эти предварительные условия соблюдены, сборка пакетов Debian состоит из двух шагов.

Первым шагом является создание скриптов пакета Debian и метаданных из репозитория git, путем запуска:

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

Note

Скрипт`debian/configure` отличается от скрипта `src/configure`!
Скрипт `debian/configure` принимает аргументы в зависимости от платформы, на которой/для которой вы строите, см. раздел debian/configure arguments section. По умолчанию LinuxCNC работает в пользовательском пространстве (”uspace”), ожидая, что ядро preempt_rt минимизирует задержки.

После конфигурации скриптов и метаданных пакета Debian соберите пакет, запустив dpkg-buildpacka

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

Note
dpkg-buildpackage необходимо запустить из корня исходного дерева, которое вы могли назвать linuxcnc-source-dir, **не** внутри linuxcnc-source-dir/debian.
dpkg-buildpackage принимает необязательный аргумент ``-j`N` (где *N* — число). Это позволяет запускать несколько заданий одновременно.

6.4.2.1 LinuxCNC's debian/configure параметры

В исходном дереве LinuxCNC есть каталог Debian со всей информацией о том, как должен быть собран пакет Debian, но некоторые ключевые файлы в нем распространяются только как шаблоны. Скрипт debian/configure подготавливает эти инструкции по сборке для обычных утилит упаковки Debian и поэтому должен быть запущен до dpkg-checkbuilddeps или dpkg-buildpackage.

Скрипт debian/configure принимает один аргумент, который определяет базовую платформу реального или не реального времени, для которой будет выполняться сборка. Обычные значения этого аргумента:

no-docs
: Пропустить сборку документации.

uspace
: Настройте пакет Debian для режима Preempt-RT в реальном времени или для режима не в реальном времени (эти два варианта совместимы).

noauto , rtai , xenomai
: Обычно списки OCPB для поддержки uspace realtime определяются автоматически. Однако, если вы хотите, вы можете указать одну или несколько из них после uspace, чтобы включить поддержку этих OCPB. Чтобы отключить автоопределение, укажите noauto.

  Если вам нужен только традиционный «модуль ядра» RTAI в реальном времени, используйте вместо этого -r или $KERNEL_VERSION.

rtai=<package name>
: Если пакет разработки для RTAI lxrt не запускается с "rtai-modules", или если первый такой пакет, указанный поиском apt-cache, не является нужным, тогда явно укажите имя пакета.

-r
: Настройте пакет Debian для работающего в данный момент ядра RTAI. Чтобы это работало, на вашей сборочной машине должно быть запущено ядро RTAI!

$KERNEL_VERSION
: Настройте пакет Debian для указанной версии ядра RTAI (например, «3.4.9-rtai-686-pae»). На компьютере, на котором производится сборка, должен быть установлен пакет Debian с совпадающими заголовками ядра (например, «linux-headers-3.4.9-rtai-686-pae»). Обратите внимание, что вы можете *собирать* LinuxCNC в этой конфигурации, но если вы не используете соответствующее ядро RTAI, вы не сможете *запустить* LinuxCNC, включая набор тестов.

6.4.2.2 Удовлетворение зависимостей сборки

На платформах на основе Debian мы предоставляем метаданные о пакетах , которые знают, какие внешние программные пакеты необходимо установить для сборки LinuxCNC. Это называется

зависимостями сборки LinuxCNC. Они называются *зависимостями сборки* LinuxCNC, т.е. теми пакетами, которые должны быть доступны * сборка прошла успешно и * сборка может быть сделана воспроизводимо.

Вы можете использовать эти метаданные для легкого перечисления требуемых пакетов, отсутствующи в вашей системе сборки. Сначала перейдите в исходное дерево LinuxCNC и инициируйте его самонастройку по умолчанию, если она еще не выполнена:

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

Это подготовит файл debian/control, содержащий списки пакетов Debian, которые необходимо создать, с зависимостями времени выполнения для этих пакетов, а для нашего случая также и зависимости сборки для этих пакетов, которые необходимо создать.

Самый простой способ установить все зависимости сборки — просто выполнить (из того же каталога):

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

которая установит все требуемые зависимости, еще не установлены, но доступны. `.` является частью командной строки, т.е. инструкцией по извлечению зависимостей для исходного дерева, а не для зависимостей другого пакета. Это завершает установку build-dependencies.

Оставшаяся часть этого раздела описывает полуручной подход. Список зависимостей в debian/-control длинный, и сравнивать с ним текущее состояние уже установленных пакетов утомительно. В системах Debian есть программа под названием dpkg-checkbuilddeps, которая анализирует метаданные пакета и сравнивает пакеты, перечисленные как зависимости сборки, со списком установленных пакетов, а также сообщает вам, чего не хватает.

Сначала установите программу `dpkg-checkbuilddeps`, запустив:

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

Это создает файл `debian/control` в формате yaml, доступном для чтения пользователем, в котором перечислены зависимости сборки ближе к началу. Вы можете использовать эти метаданные для простого перечисления требуемых пакетов, отсутствующих в вашей системе сборки. Вы можете решить вручную проверить эти файлы, если у вас есть хорошее представление о том, что уже установлено.

В качестве альтернативы системы Debian предоставляют программу `dpkg-checkbuilddeps`, которая анализирует метаданные пакета и сравнивает пакеты, перечисленные как зависимости сборки, со списком установленных пакетов и сообщает вам, чего не хватает. Кроме того, `dpkg-buildpackage` сообщит вам о том, чего не хватает, и это должно быть нормально. Однако она сообщает об отсутствующих зависимостях сборки только после того, как исправления в каталоге `debian/patches` были автоматически применены (если таковые имеются). Если вы новичок в Linux и управлении версиями git, чистый старт может быть предпочтительнее, чтобы избежать осложнений.

Программу `dpkg-checkbuilddeps` (также из пакета dpkg-dev, который устанавливается как часть зависимостей build-essential) можно попросить выполнить свою работу (обратите внимание, что ее нужно запустить из каталога `linuxcnc-source-dir`, **не** из `linuxcnc-source-dir/debian`):

```
$ dpkg-checkbuilddeps
```

Он выдаст список пакетов, которые требуются для сборки LinuxCNC на вашей системе, но пока не установлены. Теперь вы можете установить отсутствующие зависимости сборки

вручную
: Установите их все с помощью `sudo apt-get install`, а затем имена пакетов. Вы можете повторно запустить `dpkg-checkbuilddeps` в любое время, чтобы получить список отсутствующих пакетов, что не повлияет на исходное дерево.

автоматизированный
Запустите `sudo apt build-dep . .`

Если вы сомневаетесь в том, что может предоставлять конкретный пакет build-dep, проверьте описание пакета с помощью ``apt-cache show`` *packagename*.

6.4.2.3 Варианты для `dpkg-buildpackage`

Для сборки типичного пакета Debian вам нужно запустить dpkg-buildpackage без каких-либо аргументов. Как было указано выше, команде переданы две дополнительные опции. Как и для всех хороших инструментов Linux, на странице руководства есть все подробности с помощью `man dpkg-buildpackage`.

`-uc`
Не подписывайте цифровыми подписями полученные двоичные файлы. Вам следует подписывать пакеты своим ключом GPG только в том случае, если вы хотите распространять их среди других. Если эта опция не установлена, а затем пакет не подписан, это не повлияет на файл .deb.

`-b`
Компилирует только архитектурно-зависимые пакеты (например, двоичные файлы `linuxcnc` и GUI). Это очень полезно, чтобы избежать компиляции того, что не зависит от оборудования. Для LinuxCNC это документация, которая в любом случае доступна онлайн.

Если у вас возникли трудности при компиляции, посетите онлайн-форум LinuxCNC.

В настоящее время появляется поддержка переменной среды DEB_BUILD_OPTIONS. Установите ее в

`nodocs`
чтобы пропустить сборку документации, лучше использовать флаг `-B` для dpkg-buildpackage.

`nocheck`
для пропуска самотестирования процесса сборки LinuxCNC. Это экономит время и снижает потребность в нескольких программных пакетах, которые могут быть недоступны для вашей системы, например, в xvfb. Не следует устанавливать эту опцию, чтобы получить дополнительную уверенность в том, что ваша сборка будет работать так, как ожидается, если только вы не сталкиваетесь с чисто техническими трудностями с зависимостями программного обеспечения, специфичными для тестирования.

Переменная окружения может быть установлена вместе с выполнением команды, например:

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

объединит все варианты, представленные в этом разделе.

6.4.2.4 Установка самостоятельно собранных пакетов Debian

Пакет Debian можно распознать по его расширению .deb. Инструмент, устанавливающий его, `dpkg`, является частью каждой установки Debian. Файлы .deb, созданные `dpkg-buildpackage`, находятся в каталоге выше linuxcnc-source-dir, т. е. в `...` Чтобы увидеть, какие файлы предоставляются в пакете, запустите

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

Версия LinuxCNC будет частью имени файла, которое должно соответствовать звездочке. Возможно, в списке слишком много файлов, чтобы поместиться на экране. Если вы не можете прокрутить вверх в своем терминале, добавьте `| more` к этой команде, чтобы ее вывод прошел через так называемый ”pager”. Выйдите, нажав ”q”.

Чтобы установить пакеты, запустите

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

6.5 Настройка среды

В этом разделе описываются специальные шаги, необходимые для настройки станка для запуска программ LinuxCNC, включая тесты.

6.5.1 Увеличьте лимит заблокированной памяти

LinuxCNC пытается улучшить свою задержку в реальном времени, блокируя используемую память в ОЗУ. Это делается для того, чтобы не дать операционной системе выгрузить LinuxCNC на диск, что может плохо сказаться на задержке. Обычно блокировка памяти в ОЗУ не приветствуется, и операционная система устанавливает строгие ограничения на объем памяти, который пользователь может заблокировать.

При использовании платформы реального времени Preempt-RT LinuxCNC запускается с достаточными привилегиями, чтобы самостоятельно повысить свой лимит блокировки памяти. При использовании платформы реального времени RTAI у него недостаточно привилегий, и пользователь должен повысить лимит блокировки памяти.

Если LinuxCNC при запуске отображает следующее сообщение, проблема заключается в установленно в вашей системе ограничении заблокированной памяти:

```
RTAPI: ERROR: failed to map shmem
RTAPI: Locked memory limit is 32KiB, recommended at least 20480KiB.
```

Чтобы исправить эту проблему, добавьте файл с именем `/etc/security/limits.d/linuxcnc.conf` (как root) с помощью вашего любимого текстового редактора (например, `sudo gedit /etc/security/l` Файл должен содержать следующую строку:

```
* - memlock 20480
```

Выйдите из системы и войдите снова, чтобы изменения вступили в силу. Убедитесь, что предел блокировки памяти повышен, используя следующую команду:

```
$ ulimit -l
```

6.6 Сборка на Gentoo

Сборка на Gentoo возможна, но не поддерживается. Убедитесь, что вы используете профиль рабочего стола. Этот проект использует Tk Widget Set, asciidoc и имеет некоторые другие зависимости Они должны быть установлены как 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
```

Вы можете переключиться обратно на обычного пользователя для большей части оставшейся части установки. Как этот пользователь, создайте виртуальную среду для pip, затем установите пакеты pip:

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

Затем вы можете продолжить как обычно:

```
(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
```

Нет необходимости запускать ”make suid”, просто убедитесь, что ваш пользователь находится в группе ”dialout”. Чтобы запустить linuxcnc, вы должны находиться в виртуальной среде Python и настроить среду linuxcnc:

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

6.7 Варианты проверки репозитория git

Инструкции Quick Start в верхней части этого документа клонируют наш репозиторий git с https://github.com/LinuxCNC/linuxcnc.git. Это самый быстрый и простой способ начать. Однако есть и другие варианты, которые стоит рассмотреть.

6.7.1 Форкните нас на GitHub

Git-репозиторий проекта LinuxCNC находится по адресу https://github.com/LinuxCNC/linuxcnc. GitHub — популярный сервис хостинга git и сайт для обмена кодом. Вы можете легко (и бесплатно) создать форк (второй экземпляр, содержащий копию, которую вы контролируете) git-репозитория LinuxCNC на GitHub. Затем вы можете использовать этот форк для отслеживания и публикации своих изменений, получения комментариев к своим изменениям и принятия патчей от сообщества.

После создания собственного форка LinuxCNC на GitHub клонируйте его на свой компьютер для разработки и продолжайте заниматься хакингом как обычно.

Мы из проекта LinuxCNC надеемся, что вы поделитесь с нами своими изменениями, чтобы сообщество могло извлечь пользу из вашей работы. GitHub делает этот обмен очень простым: после того, как вы отшлифуете свои изменения и отправите их в свой форк GitHub, отправьте нам запрос на Pull Request.

Chapter 7

Добавление элементов выбора конфи

Примеры конфигураций можно добавить в селектор конфигураций двумя способами:

- Дополнительные приложения. Приложения, установленные независимо с пакетами deb, могут размещать подкаталоги конфигурации в указанном системном каталоге. Имя каталога задается с помощью сценария оболочки linuxcnc_var:

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

- Параметры среды выполнения — селектор конфигурации также может предлагать подкаталоги конфигурации, указанные в момент выполнения с использованием экспортированной переменной среды (LINUXCNC_AUX_CONFIGS). Эта переменная должна быть списком путей одного или нескольких каталогов конфигурации, разделенных знаком (:). Обычно эта переменная устанавливае в оболочке, запускающей `linuxcnc` , или в пользовательском сценарии запуска `~/.profile`. Пример:

```
export LINUXCNC_AUX_CONFIGS=~/myconfigs:/opt/otherconfigs
```

Chapter 8

Contributing to LinuxCNC

8.1 Введение

This document contains information for developers about LinuxCNC infrastructure, and describes the best practices for contributing code and documentation updates to the LinuxCNC project.

Throughout this document, ”source” means both the source code to the programs and libraries, and the source text for the documentation.

8.2 Communication among LinuxCNC developers

The two main ways that project developers communicate with each other are:

- Via IRC, at #linuxcnc-devel on Libera.chat.
- Via email, on the developers’ mailing list

8.3 The LinuxCNC Source Forge project

We use Source Forge for mailing lists.

8.4 The Git Revision Control System

All of the LinuxCNC source is maintained in the Git revision control system.

8.4.1 LinuxCNC official Git repo

The official LinuxCNC git repo is at https://github.com/linuxcnc/linuxcnc/

Anyone can get a read-only copy of the LinuxCNC source tree via git:

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

If you are a developer with push access, then follow github's instructions for setting up a repository that you can push from.

Note that the clone command put the local LinuxCNC repo in a directory called `linuxcnc-dev`, instead of the default `linuxcnc`. This is because the LinuxCNC software by default expects configs and G-code programs in a directory called `$HOME/linuxcnc`, and having the git repo there too is confusing.

Issues and pull requests (abbreviated PRs) are welcome on GitHub: https://github.com/LinuxCNC/-linuxcnc/issues https://github.com/LinuxCNC/linuxcnc/pulls

8.4.2 Use of Git in the LinuxCNC project

We use the "merging upwards" and "topic branches" git workflows described here:

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

У нас есть ветка разработки под названием «мастер» и одна или несколько стабильных веток с именами вроде «2.6» и «2.7», указывающими номер версии выпусков, которые мы делаем из нее.

Исправления идут в самой старой применимой стабильной ветке, и эта ветка объединяется со следующей, более новой стабильной веткой, и так далее до «мастера». Коммиттер исправления может сделать слияние самостоятельно или оставить слияние для кого-то другого.

Новые функции обычно помещаются в ветку master, но некоторые виды функций (в частности, хорошо изолированные драйверы устройств и документация) могут (по усмотрению менеджеров выпуска стабильной ветки) помещаться в стабильную ветку и объединяться, как это делается с исправлением ошибок.

8.4.3 git учебники

В Интернете есть много отличных бесплатных руководств по git.

Первое место, куда стоит заглянуть, это, вероятно, man-страница gittutorial. Доступ к этой справочной странице можно получить, запустив «man gittutorial» в терминале (если у вас установлены справочные страницы git). Gittutorial и последующая документация также доступны онлайн здесь:

- git tutorial: https://www.kernel.org/pub/software/scm/git/docs/gittutorial.html
- git tutorial 2: https://www.kernel.org/pub/software/scm/git/docs/gittutorial-2.html
- Ежедневный git с примерно 20 командами: https://www.kernel.org/pub/software/scm/git/docs/-giteveryday.html
- Руководство пользователя Git: https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

Более подробную документацию по git см. в книге «Pro Git»: https://git-scm.com/book

Еще один рекомендуемый онлайн-учебник — «Git для ленивых»: https://wiki.spheredev.org/index.php/-Git_for_the_lazy

8.5 Обзор процесса

Общий обзор того, как внести изменения в исходный код, выглядит следующим образом:

- Communicate with the project developers and let us know what you're hacking on. Explain what you are doing, and why.

- Clone the git repo.
- Make your changes in a local branch.
- Добавление документации и Написание тестов — важная часть добавления новой функции. В противном случае другие не будут знать, как использовать вашу функцию, а если другие изменения сломают вашу функцию, это может остаться незамеченным без проверки.
- Поделитесь своими изменениями с другими разработчиками проекта одним из следующих способов:
  - Push your branch to github and create a github pull request to https://github.com/linuxcnc/-linuxcnc (this requires a github account), or
  - Push your branch to a publicly visible git repo (such as github, or your own publicly-accessible server, etc) and share that location on the emc-developers mailing list, or
  - Email your commits to the LinuxCNC-developers mailing list (<emc-developers@lists.sourceforge.net>) (use `git format-patch` to create the patches).
- Advocate for your patch:
  - Explain what problem it addresses and why it should be included in LinuxCNC.
  - Будьте восприимчивы к вопросам и отзывам от сообщества разработчиков.
  - Патч нередко подвергается нескольким ревизиям, прежде чем он будет принят.

8.6 Конфигурация git

Чтобы коммиты рассматривались для включения в исходный код LinuxCNC, они должны иметь правильные поля Author, идентифицирующие автора коммита. Хороший способ убедиться в этом — настроить глобальную конфигурацию git:

```
git config --global user.name ”Your full name”
git config --global user.email ”you@example.com”
```

Используйте свое настоящее имя (не псевдоним) и незашифрованный адрес электронной почты.

8.7 Эффективное использование git

8.7.1 Содержимое коммита

Держите свои коммиты небольшими и по существу. Каждый коммит должен выполнять одно логическое изменение репо.

8.7.2 Пишите хорошие сообщения коммитов

Сохраняйте сообщения коммитов шириной около 72 столбцов (чтобы в окне терминала по умолчанию они не переносились при отображении с помощью `git log`).

Используйте первую строку как краткое изложение цели изменения (почти как строку темы электронного письма). За ним следует пустая строка, а затем более длинное сообщение, объясняющее изменение. Пример:

8.7.3 Коммит в нужную ветку

Исправления должны идти в самой старой применимой ветке. Новые функции должны идти в основной ветке. Если вы не уверены, куда относится изменение, спросите в irc или в списке рассылки.

8.7.4 Используйте несколько коммитов для организации изменений

Когда это уместно, организуйте свои изменения в ветку (серию коммитов), где каждый коммит является логическим шагом к вашей конечной цели. Например, сначала вынесите какой-нибудь сложный код в новую функцию. Затем, во втором коммите, исправьте базовую ошибку. Затем в третьем коммите добавьте новую функцию, которая упрощается благодаря рефакторингу и которая не работала бы без исправления этой ошибки.

Это полезно для рецензентов, потому что легче увидеть, что шаг «вынести код в новую функцию» был правильным, когда нет других смешанных правок; легче увидеть, что ошибка исправлена, когда исправляющее ее изменение отделено от новой функции; и так далее.

8.7.5 Следуйте стилю окружающего кода

Постарайтесь следовать преобладающему стилю отступов окружающего кода. В частности, изменения пробелов затрудняют другим разработчикам отслеживание изменений с течением времени. Когда необходимо выполнить переформатирование кода, делайте это как коммит отдельно от каких-либо семантических изменений.

8.7.6 Избавьтесь от RTAPI_SUCCESS, вместо этого используйте 0

Тест «retval < 0» должен показаться вам знакомым; это тот же тест, который вы используете в пользовательском пространстве (возвращает -1 в случае ошибки) и в пространстве ядра (возвращает -ERRNO в случае ошибки).

8.7.7 Упростите сложную историю, прежде чем делиться ею с другими разработчиками

С git можно записывать каждое редактирование и фальстарт как отдельный коммит. Это очень удобно как способ создания контрольных точек во время разработки, но часто вы не хотите делиться этими фальстартами с другими.

Git предоставляет два основных способа очистки истории, оба из которых можно использовать свободно, прежде чем делиться изменениями:

`git commit --amend` позволяет вам внести дополнительные изменения в последнюю вещь, которую вы закоммитили, а также при необходимости изменить сообщение коммита. Используйте это, если вы сразу поняли, что что-то упустили из коммита, или если вы опечатались в сообщении коммита.

`git rebase --interactive` upstream-branch позволяет вернуться к каждому коммиту, сделанному с тех пор, как вы разветвили свою ветку функций из восходящей ветки, возможно, отредактировав коммиты, отбросив коммиты или смешав (объединив) коммиты с другими. Rebase также можно использовать для разделения отдельных коммитов на несколько новых коммитов.

8.7.8 Убедитесь, что каждый коммит собирается

Если ваше изменение состоит из нескольких исправлений, можно использовать `git rebase -i`, чтобы переупорядочить эти исправления в последовательность коммитов, которая более четко описывает этапы вашей работы. Потенциальным последствием переупорядочивания патчей является то, что можно неправильно понять зависимости — например, ввести использование переменной, а объявление этой переменной следует только в более позднем патче.

Хотя ветвь HEAD будет собираться, не каждый коммит может быть собран в таком случае. Это ломает `git bisect` - то, что кто-то другой может использовать позже, чтобы найти фиксацию, которая привела к ошибке. Таким образом, помимо обеспечения сборки вашей ветки, важно также обеспечить сборку каждого отдельного коммита.

Существует автоматический способ проверки ветки для каждого коммита, который можно построить — см. https://dustin.sallings.org/2010/03/28/git-test-sequence.html и код на https://github.com/ dustin/bind test-sequence. Используйте следующим образом (в этом случае тестируйте каждый коммит от origin/master до HEAD, включая выполнение регрессионных тестов):

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

Это либо сообщит *All is well*, либо *Broke on <commit>*

8.7.9 Переименование файлов

Пожалуйста, используйте возможность переименования файлов очень осторожно. Подобно применен отступов для отдельных файлов, переименования по-прежнему затрудняют отслеживание изменений с течением времени. Как минимум, вы должны искать консенсус в irc или списке рассылки, что переименование является улучшением.

8.7.10 Предпочитаю ”rebase”

Используйте `git pull --rebase` вместо простого `git pull`, чтобы сохранить красивую линейную историю. Когда вы перебазируете, вы всегда сохраняете свою работу как ревизии, которые опережают origin/master, поэтому вы можете делать такие вещи, как `git format-patch`, чтобы делиться ими с другими, не отправляя их в центральный репозиторий.

8.8 Переводы

Проект LinuxCNC использует `gettext` для перевода программного обеспечения на многие языки. Мы приветствуем вклад и помощь в этой области! Улучшать и расширять переводы легко: вам не нужно знать программирование, и вам не нужно устанавливать какие-либо специальные программы для перевода или другое программное обеспечение.

Самый простой способ помочь с переводами — использовать Weblate, веб-сервис с открытым исходным кодом. Наш переводческий проект находится здесь:

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

Документация по использованию Weblate находится здесь: https://docs.weblate.org/en/latest/user/-basic.html

8.9 Другие способы внести свой вклад

Есть много способов внести свой вклад в LinuxCNC, которые не рассматриваются в этом документе. Эти способы включают в себя:

- Отвечая на вопросы на форуме, в списках рассылки и в IRC
- Сообщение об ошибках в системе отслеживания ошибок, на форуме, в списках рассылки или в IRC
- Помощь в тестировании экспериментальных функций

Chapter 9

Glossary

A listing of terms and what they mean. Some terms have a general meaning and several additional meanings for users, installers, and developers.

Acme Screw
A type of lead-screw that uses an Acme thread form. Acme threads have somewhat lower friction and wear than simple triangular threads, but ball-screws are lower yet. Most manual machine tools use acme lead-screws.

Axis
One of the computer controlled movable parts of the machine. For a typical vertical mill, the table is the X axis, the saddle is the Y axis, and the quill or knee is the Z axis. Angular axes like rotary tables are referred to as A, B, and C. Additional linear axes relative to the tool are called U, V, and W respectively.

AXIS(ГИП)
One of the Graphical User Interfaces available to users of LinuxCNC. It features the modern use of menus and mouse buttons while automating and hiding some of the more traditional LinuxCNC controls. It is the only open-source interface that displays the entire tool path as soon as a file is opened.

GMOCCAPY (GUI)
A Graphical User Interfaces available to users of LinuxCNC. It features the use and feel of an industrial control and can be used with touch screen, mouse and keyboard. It support embedded tabs and hal driven user messages, it offers a lot of hal beens to be controlled with hardware. GMOCCAPY is highly customizable.

Backlash
The amount of ”play” or lost motion that occurs when direction is reversed in a lead screw. or other mechanical motion driving system. It can result from nuts that are loose on leadscrews, slippage in belts, cable slack, ”wind-up” in rotary couplings, and other places where the mechanical system is not ”tight”. Backlash will result in inaccurate motion, or in the case of motion caused by external forces (think cutting tool pulling on the work piece) the result can be broken cutting tools. This can happen because of the sudden increase in chip load on the cutter as the work piece is pulled across the backlash distance by the cutting tool.

Backlash Compensation
Any technique that attempts to reduce the effect of backlash without actually removing it from the mechanical system. This is typically done in software in the controller. This can correct the final resting place of the part in motion but fails to solve problems related to direction changes while in motion (think circular interpolation) and motion that is caused when external forces (think cutting tool pulling on the work piece) are the source of the motion.

Ball Screw
A type of lead-screw that uses small hardened steel balls between the nut and screw to reduce friction. Ball-screws have very low friction and backlash, but are usually quite expensive.

Ball Nut
A special nut designed for use with a ball-screw. It contains an internal passage to re-circulate the balls from one end of the screw to the other.

CNC
Computer Numerical Control. The general term used to refer to computer control of machinery. Instead of a human operator turning cranks to move a cutting tool, CNC uses a computer and motors to move the tool, based on a part program.

Comp
A tool used to build, compile and install LinuxCNC HAL components.

Configuration(n)
A directory containing a set of configuration files. Custom configurations are normally saved in the users home/linuxcnc/configs directory. These files include LinuxCNC's traditional INI file and HAL files. A configuration may also contain several general files that describe tools, parameters, and NML connections.

Configuration(v)
The task of setting up LinuxCNC so that it matches the hardware on a machine tool.

Coordinate Measuring Machine
A Coordinate Measuring Machine is used to make many accurate measurements on parts. These machines can be used to create CAD data for parts where no drawings can be found, when a hand-made prototype needs to be digitized for moldmaking, or to check the accuracy of machined or molded parts.

Display units
The linear and angular units used for onscreen display.

DRO
A Digital Read Out is a system of position-measuring devices attached to the slides of a machine tool, which are connected to a numeric display showing the current location of the tool with respect to some reference position. DROs are very popular on hand-operated machine tools because they measure the true tool position without backlash, even if the machine has very loose Acme screws. Some DROs use linear quadrature encoders to pick up position information from the machine, and some use methods similar to a resolver which keeps rolling over.

EDM
EDM is a method of removing metal in hard or difficult to machine or tough metals, or where rotating tools would not be able to produce the desired shape in a cost-effective manner. An excellent example is rectangular punch dies, where sharp internal corners are desired. Milling operations can not give sharp internal corners with finite diameter tools. A *wire* EDM machine can make internal corners with a radius only slightly larger than the wire's radius. A *sinker* EDM can make internal corners with a radius only slightly larger than the radius on the corner of the sinking electrode.

EMC
The Enhanced Machine Controller. Initially a NIST project. Renamed to LinuxCNC in 2012.

EMCIO
The module within LinuxCNC that handles general purpose I/O, unrelated to the actual motion of the axes.

EMCMOT
The module within LinuxCNC that handles the actual motion of the cutting tool. It runs as a real-time program and directly controls the motors.

Энкодер
A device to measure position. Usually a mechanical-optical device, which outputs a quadrature signal. The signal can be counted by special hardware, or directly by the parport with LinuxCNC.

Feed
Relatively slow, controlled motion of the tool used when making a cut.

Feed rate
The speed at which a cutting motion occurs. In auto or MDI mode, feed rate is commanded using an F word. F10 would mean ten machine units per minute.

Feedback
A method (e.g., quadrature encoder signals) by which LinuxCNC receives information about the position of motors.

Feedrate Override
A manual, operator controlled change in the rate at which the tool moves while cutting. Often used to allow the operator to adjust for tools that are a little dull, or anything else that requires the feed rate to be ”tweaked”.

Floating Point Number
A number that has a decimal point. (12.300) In HAL it is known as float.

G-code
The generic term used to refer to the most common part programming language. There are several dialects of G-code, LinuxCNC uses RS274/NGC.

GUI
Graphical User Interface.

General
A type of interface that allows communications between a computer and a human (in most cases) via the manipulation of icons and other elements (widgets) on a computer screen.

LinuxCNC
An application that presents a graphical screen to the machine operator allowing manipulation of the machine and the corresponding controlling program.

HAL
Hardware Abstraction Layer. At the highest level, it is simply a way to allow a number of building blocks to be loaded and interconnected to assemble a complex system. Many of the building blocks are drivers for hardware devices. However, HAL can do more than just configure hardware drivers.

Home
A specific location in the machine’s work envelope that is used to make sure the computer and the actual machine both agree on the tool position.

INI file
A text file that contains most of the information that configures LinuxCNC for a particular machine.

Instance
One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the ”Lassie” object is an instance of the ”Dog” class.

Joint Coordinates
These specify the angles between the individual joints of the machine. See also Kinematics

Jog
Manually moving an axis of a machine. Jogging either moves the axis a fixed amount for each key-press, or moves the axis at a constant speed as long as you hold down the key. In manual mode, jog speed can be set from the graphical interface.

kernel-space
Код, выполняемый внутри ядра, в отличие от кода, выполняемого в пользовательском пространст
Некоторые системы реального времени (например, RTAI) выполняют код реального времени в ядре, а код не в реальном времени — в пользовательском пространстве, в то время как другие системы реального времени (например, Preempt-RT) выполняют код как в реальном времени, так и не в реальном времени в пользовательском пространстве.

Kinematics
The position relationship between world coordinates and joint coordinates of a machine. There are two types of kinematics. Forward kinematics is used to calculate world coordinates from joint coordinates. Inverse kinematics is used for exactly the opposite purpose. Note that kinematics does not take into account, the forces, moments etc. on the machine. It is for positioning only.

Lead-screw
An screw that is rotated by a motor to move a table or other part of a machine. Lead-screws are usually either ball-screws or acme screws, although conventional triangular threaded screws may be used where accuracy and long life are not as important as low cost.

Machine units
The linear and angular units used for machine configuration. These units are specified and used in the INI file. HAL pins and parameters are also generally in machine units.

MDI
Manual Data Input. This is a mode of operation where the controller executes single lines of G-code as they are typed by the operator.

NIST
National Institute of Standards and Technology. An agency of the Department of Commerce in the United States.

NML
Neutral Message Language provides a mechanism for handling multiple types of messages in the same buffer as well as simplifying the interface for encoding and decoding buffers in neutral format and the configuration mechanism.

Offsets
An arbitrary amount, added to the value of something to make it equal to some desired value. For example, G-code programs are often written around some convenient point, such as X0, Y0. Fixture offsets can be used to shift the actual execution point of that G-code program to properly fit the true location of the vice and jaws. Tool offsets can be used to shift the ”uncorrected” length of a tool to equal that tool’s actual length.

Part Program
A description of a part, in a language that the controller can understand. For LinuxCNC, that language is RS-274/NGC, commonly known as G-code.

Program Units
The linear and angular units used in a part program. The linear program units do not have to be the same as the linear machine units. See G20 and G21 for more information. The angular program units are always measured in degrees.

Python
General-purpose, very high-level programming language. Used in LinuxCNC for the Axis GUI, the StepConf configuration tool, and several G-code programming scripts.

Rapid
Fast, possibly less precise motion of the tool, commonly used to move between cuts. If the tool meets the workpiece or the fixturing during a rapid, it is probably a bad thing!

Rapid rate
The speed at which a rapid motion occurs. In auto or MDI mode, rapid rate is usually the maximum speed of the machine. It is often desirable to limit the rapid rate when testing a G-code program for the first time.

Real-time
Software that is intended to meet very strict timing deadlines. On Linux, in order to meet these requirements it is necessary to install a realtime kernel such as RTAI or Preempt-RT, and build the LinuxCNC software to run in the special real-time environment. Realtime software can run in the kernel or in userspace, depending on the facilities offered by the system.

RTAI
Real Time Application Interface, see https://www.rtai.org/, the real-time extensions for Linux that LinuxCNC can use to achieve real-time performance.

RTLINUX
See https://en.wikipedia.org/wiki/RTLinux, an older real-time extension for Linux that LinuxCNC used to use to achieve real-time performance. Obsolete, replaced by RTAI.

RTAPI
A portable interface to real-time operating systems including RTAI and POSIX pthreads with realtime extensions.

RS-274/NGC
The formal name for the language used by LinuxCNC part programs.

Servo Motor
Generally, any motor that is used with error-sensing feedback to correct the position of an actuator. Also, a motor which is specially-designed to provide improved performance in such applications.

Servo Loop
A control loop used to control position or velocity of an motor equipped with a feedback device.

Signed Integer
A whole number that can have a positive or negative sign. In HAL it is usually a s32, but could be also a s64.

Spindle
The part of a machine tool that spins to do the cutting. On a mill or drill, the spindle holds the cutting tool. On a lathe, the spindle holds the workpiece.

Spindle Speed Override
A manual, operator controlled change in the rate at which the tool rotates while cutting. Often used to allow the operator to adjust for chatter caused by the cutter's teeth. Spindle Speed Override assumes that the LinuxCNC software has been configured to control spindle speed.

StepConf
An LinuxCNC configuration wizard. It is able to handle many step-and-direction motion command based machines. It writes a full configuration after the user answers a few questions about the computer and machine that LinuxCNC is to run on.

Stepper Motor
A type of motor that turns in fixed steps. By counting steps, it is possible to determine how far the motor has turned. If the load exceeds the torque capability of the motor, it will skip one or more steps, causing position errors.

TASK
The module within LinuxCNC that coordinates the overall execution and interprets the part program.

Tcl/Tk
A scripting language and graphical widget toolkit with which several of LinuxCNCs GUIs and selection wizards were written.

Traverse Move
A move in a straight line from the start point to the end point.

Units
See ”Machine Units”, ”Display Units”, or ”Program Units”.

Unsigned Integer
A whole number that has no sign. In HAL it is usually a u32 but could be also a u64.

World Coordinates
This is the absolute frame of reference. It gives coordinates in terms of a fixed reference frame that is attached to some point (generally the base) of the machine tool.

Chapter 10

Legal Section

Translations of this file provided in the source tree are not legally binding.

10.1 Copyright Terms

10.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

0. PREAMBLE

Цель настоящей лицензии — сделать руководство, учебник или другой письменный документ ”свободным” в смысле свободы: гарантировать каждому эффективную свободу копировать и распространять его с модификацией или без нее, как в коммерческих, так и в некоммерческих целях. Во-вторых, настоящая лицензия оставляет за автором и издателем возможность получить признание за свою работу, не неся при этом ответственности за изменения, внесенные другими.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

«Второй раздел» — это поименованное приложение или основной раздел Документа, который касается исключительно отношений издателей или авторов Документа к общей теме Документа (или к связанным с ней вопросам) и не содержит ничего, что могло бы напрямую относиться к в рамках этой общей темы. (Например, если Документ частично является учебником по математике, Второй раздел не может объяснять какую-либо математику.) Отношения могут быть вопросом исторической связи с предметом или связанными с ним вопросами, а также юридическими, коммерческими, философскими, этическая или политическая позиция по отношению к ним.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

”Прозрачная” копия Документа означает машиночитаемую копию, представленную в формате, спецификация которого доступна широкой публике, содержание которого может быть просмотрено и отредактировано прямо и понятно с помощью обычных текстовых редакторов или (для изображений, состоящих из пикселей) общедоступных программ рисования или (для рисунков) какой-либо широко доступный редактор рисунков, который подходит для ввода в форматировщики текста или для автоматического перевода в различные форматы, подходящие для ввода в форматировщики текста. Копия, сделанная в формате файла ”Прозрачный”, разметка которого была разработана для предотвращения или избегания последующего изменения читателями, не является ”Прозрачной” Копия, которая не является ”Прозрачной”, называется ”Непрозрачной”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add

other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

Просьба, но не обязательно, чтобы вы связались с авторами Документа задолго до распространения большого количества копий, чтобы дать им возможность предоставить вам обновленную версию Документа.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Используйте на титульном листе (и на обложках, если таковые имеются) заголовок, отличный от названия Документа и предыдущих версий (которые должны, если таковые имеются, быть указаны в разделе «История» Документа) . Вы можете использовать то же название, что и в предыдущей версии, если первоначальный издатель этой версии дает разрешение. Б. Укажите на титульном листе в качестве авторов одно или несколько физических или юридических лиц, ответственных за авторство изменений в Модифицированной версии, а также не менее пяти основных авторов Документа (всех его основных авторов, если он имеет менее пяти). C. Укажите на титульном листе имя издателя Модифицированной версии в качестве издателя. D. Сохраните все уведомления об авторских правах на Документ. E. Добавьте соответствующее уведомление об авторских правах для ваших изменений рядом с другими уведомлениями об авторских правах. F. Включите сразу после уведомлений об авторских правах уведомление о лицензии, дающее публичное разрешение на использование Модифицированной версии в соответствии с условиями настоящей Лицензии, в форме, показанной в Приложении ниже. G. Сохраните в этом уведомлении о лицензии полные списки Неизменяемых разделов и обязательных сопроводительных текстов, приведенных в уведомлении о лицензии к Документу. H. Включите неизмененную копию настоящей Лицензии. I. Сохраните раздел, озаглавленный «История», и его название, и добавьте к нему пункт, указывающий как минимум название, год, новых авторов и издателя Модифицированной версии, как указано на титульном листе. Если в Документе нет раздела под названием ”История”, создайте его, указав название, год, авторов и издателя Документа, как указано на его титульной странице, затем добавьте элемент, описывающий Модифицированную версию, как указано в предыдущем предложении. J. Сохраните сетевое местоположение, если таковое имеется, указанное в Документе, для публичного доступа к прозрачной копии Документа, а также сетевые местоположени указанные в Документе для предыдущих версий, на которых он был основан. Их можно разместить в разделе ”История”. Вы можете опустить сетевое местоположение для произведения, которое было опубликовано не менее чем за четыре года до самого Документа, или если первоначальный издатель версии, на которую оно ссылается, дает разрешение. K. В любом разделе, озаглавленном ”Благодарности” или ”Посвящения”, сохраните название раздела и сохраните в нем всю суть и тон каждого из приведенных в нем благодарностей и/или посвящений участников. L. Сохраните все неизменяемые разделы Документа без изменений в их тексте и названиях. Номера разделов или их эквиваленты не считаются частью названий разделов. M. Удалить любой раздел, озаглавленны ”Подтверждения”. Такой раздел не может быть включен в Модифицированную версию. N. Не

переименовывайте существующие разделы как ”Подтверждения” и не конфликтет по названию с каким-либо неизменным разделом.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled ”History” in the various original documents, forming one section entitled ”History”; likewise combine any sections entitled ”Acknowledgements”, and any sections entitled ”Dedications”. You must delete all sections entitled ”Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an ”aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

If you have no Invariant Sections, write ”with no Invariant Sections” instead of saying which ones are invariant. If you have no Front-Cover Texts, write ”no Front-Cover Texts” instead of ”Front-Cover Texts being LIST”; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.